CHART DECODE: AN AUTOMATED SYSTEM FOR DATA TABLE EXTRACTION AND SUMMARY GENERATION FROM CHART IMAGES

Siri Chandana Daggubati

Master of Science by Research Thesis

May 2022

International Institute of Information Technology, Bangalore

CHART DECODE: AN AUTOMATED SYSTEM FOR DATA TABLE EXTRACTION AND SUMMARY GENERATION FROM CHART IMAGES

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Science by Research

by

Siri Chandana Daggubati
MS2019005

International Institute of Information Technology, Bangalore
May 2022

Dedicated to my family

Thesis Certificate

This is to certify that the thesis titled **Chart Decode: An automated system for data table extraction and summary generation from chart images** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Science by Research** is a bona fide record of the research work done by **Siri Chandana Daggubati**, **MS2019005**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

_____

Prof. Jaya Sreevalsan Nair

Bengaluru,

The 19th of May, 2022.

CHART DECODE: AN AUTOMATED SYSTEM FOR DATA TABLE EXTRACTION AND SUMMARY GENERATION FROM CHART IMAGES

Abstract

Charts are widely used visual representations of data to communicate the information clearly and efficiently. Most marketing research and data analysis workflows popularly use simple data representations such as bar, pie, scatter, and line charts to illustrate the quantitative behavior of data and observe trends over time, thus aiding us in quickly drawing new inferences from data. However, to make a thorough and accurate data analysis, one must have access to the original data table of the chart image. But in most information sources such as scholarly materials (textbooks, publications), print media (newspapers, magazines), and the internet, charts images are typically found without an original data table. In cases of the unavailability of the source data table and the information being available only in raster format, visually impaired users are precluded from reading visualizations. To address these challenges, we propose an automated chart interpretation system to extract the underlying source data table from the chart image and a brief textual description that summarizes the extracted data table. The extracted data, thus, allows the non-visual users to interpret the chart image.

Many image recognition-based algorithms for chart image interpretation have been proposed over the years. However, due to the constraints provided by the diversity of chart types, complexity of chart content, and layout, a robust solution is yet to be developed. We propose an integrated automated algorithm for data table extraction and text summarization of images of commonly available chart types and their variants. The chart types include line and pie charts along with bar chart and scatter plot variants. Our

system is designed to handle a variety of image data, which include the ones (i) sourced from crawling websites, (ii) synthetically generated from plotting tools, and (iii) curated image datasets.

Our contributions are as follows:

- Text and legend extraction from chart images for determining a scaling factor for transforming extracted data from pixel space to data space.
- Joint geometry- and color-based data extraction from chart images, specifically with circular objects*, i.e.,* scatter plots, bubble plots, pie charts, etc.
- Implementation of design study methodology for text summarization of chart images after data extraction.

The data extraction process involves decoding the graphical objects (e.g., rectangular bars, pie sectors, etc.) of chart images that rendered source data into graphical properties based on the design mechanism implemented by each chart type. So as an initial step towards chart data extraction, we classify the chart images into their respective types and subtypes using a CNN-based classifier. Specifically, for scatter plots, we use heuristic-based approaches in subtype classification owing to data insufficiency. Based on the semantics and graphical perception of chart images, we interpret the graphical objects and textual elements, e.g., chart title, legends, axis titles, and axis labels, separately. To accomplish this, we perform chart component extraction using image processing mechanisms for canvas and legend component extraction and pre-defined deep-learning-based text detection and recognition models for text element identification. Then the detected text elements are heuristically aggregated and labeled into textual components. Image processing mechanisms include marker-based water segmentation, Hough circle transform, and color-based graphical object extraction, along with a few preprocessing morphological filters for extracting noise-free canvas images.

Finally, we propose automatic data extraction of chart images using geometry-based and color-based extraction techniques based on the chart type to exploit properties of graphical objects rendered in the pixel space, which is contextualized using the information retrieved from the text. Our data extraction process addresses challenges such as overlapping data points and transparent graphical objects in scatter points and their variants (e.g., bubble plots), and proposes default interval computation based on gradient change locations and x-axis label locations in line charts, making our method more efficient than existing methods.

We also propose a design study methodology for effective text summary generation in chart images using the templated-NLG approach. The templates are designed using sentence structure built on significant features of extracted data tables to generate the initial set of system summaries. We conducted a user study to assess and improve our summary generation process and redesign sentence structure to generate a more generalized final set of master summaries by improving the inclusiveness of the usage of these charts.

We use both image processing and deep learning approaches in our workflow, along with templatized natural language generation (NLG) approaches for interpreting chart images and representing the extracted information as data tables and text summaries, and data extraction accuracy can be represented using the reconstructed or redesigned chart image of extracted data table qualitatively. The efficiency of data extraction reflects text summary generation; our preliminary results on data extraction using F1-scores and MAPE values show the effectiveness of our proposed algorithm.

Overall, our novel techniques based on image processing, deep learning, and natural language generation improve the robustness of the state-of-the-art methods for automated interpretation of chart images.

Acknowledgements

"My sincere gratitude to my advisor Prof. Jaya Sreevalsan Nair, for her invaluable advice, continuous encouragement, and support. It has been my pleasure to work under her guidance. I wish to express my sincere gratitude to the Machine Intelligence and Robotics Center at IIIT Bangalore for the generous financial support. I would also like to thank the GVCL team for providing a healthy working environment and sharing their invaluable experience. Discussing with them was always fun and stress relieving. I would also like to thank my lab mate Ms. Komal Dadhich for helping me find my research motivation and working with me. Last but not least, I would like to thank my family for maintaining their belief and faith in me, without which completing this journey would not have been possible."

— Siri Chandana Daggubati

List of Publications

1. J. Sreevalsan-Nair, K. Dadhich, and S. C. Daggubati, "Tensor Fields for Data Extraction from Chart Images: Bar Charts and Scatter Plots," (to appear) in Topological Methods in Visualization: Theory, Software and Applications, Ingrid Hotz, Talha Bin Masood, Filip Sadlo, and Julien Tierny (Eds.). Springer-Verlag, 2020; preprint at arXiv (2020), October 2020. https://doi.org/10.48550/arXiv.2010.02319

2. K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "BarChartAnalyzer: Digitizing Images of Bar Charts," in the Proceedings of the International Conference on Image Processing and Vision Engineering (IMPROVE 2021), pp 17–28, April 2021. https://doi.org/10.5220/0010408300170028

3. K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "ScatterPlotAnalyzer: Digitizing Images of Charts Using Tensor-based Computational Model" in International Conference on Computational Science – ICCS 2021, Part V, Lecture Notes in Computer Science, volume 12746, M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, and P. M. Dongarra Jack J.and Sloot, Eds., Cham: Springer International Publishing, 2021, pp. 70–83, ISBN : 978-3-030-77977-1. https://doi.org/10.1007/978-3-030-77977-1_6

4. S. C. Daggubati, and J. Sreevalsan-Nair, and K. Dadhich, "ACCirO: A System for Analyzing and Digitizing Images of Charts with Circular Objects," in International Conference on Computational Science - ICCS 2022, Lecture Notes in Computer Science, 2022 (*to appear*).

Contents

List of Figures

List of Tables

List of Abbreviations

BLEU . . . . . Bi-Lingual Evaluation Understudy

CNN . . . . . Convolutional Neural Network

CRAFT . . . . . Character Region Awareness for Text Detection

DBSCAN . . . . . Density-Based Spatial Clustering of Applications with Noise

DPI . . . . . Dots Per Inch

IOU . . . . . Intersection Over Union

NLG . . . . . Natural Language Generation

MAPE . . . . . Mean Absolute Percentage Error

MECDG . . . . . Manufacturing Enterprise Chart Description Generation

OCR . . . . . Optical Character Recognition

RANSAC . . . . . Random Sample Consensus

RNN . . . . . Recurrent Neural Network

STR . . . . . Scene Text Recognition

SVM . . . . . Support Vector Machine

CHAPTER 1

INTRODUCTION

Charts are powerful visual representations that play a significant role in disseminating massive data sets. Chart images pique curiosity and focus the viewer's attention on the significant information like patterns, trends, and outliers in data by employing visual elements like bars, lines, and circles. So data mining experts use them in exploratory data analysis of enormous data volumes for making data-driven decisions. Even business intelligence technologies use chart images as a key element in data storytelling (Figure FC1.1) because a simple chart image may speak more volumes than a big data spreadsheet ever can. However, compelling data stories require both data and textual descriptions along with chart images to communicate insights that deliver real value and accelerate the decision-making process. The need for a data component is to reach accurate data insights, and a chart image enables us to spot underlying trends and patterns in data. Finally, the text summary acts as a vehicle to convey insights, with visualizations and data being important proof points.

Despite the potential for data storytelling, communication channels such as the internet and print media frequently transmit merely chart images without the source data table and textual summary, assuming them as an easy source of communication. As a result, the information withheld by chart images in the raster format becomes inaccessible to a few sections of people like visually impaired users, leading us to use a

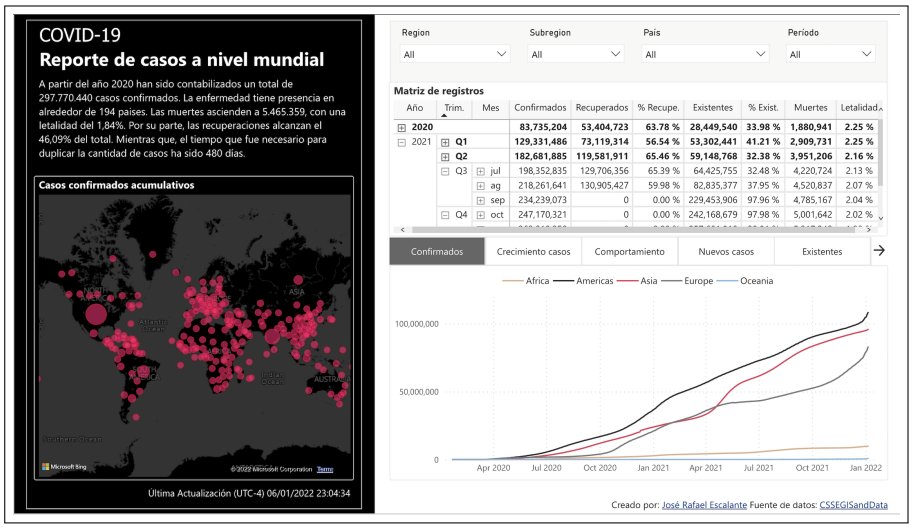


Figure FC1.1: An example of a data story, a snapshot of COVID-19 data involving all three major components: data, text, and graph visualizations. Source url: `https://community.powerbi.com/t5/COVID-19-Data-Stories-Gallery/Coronavirus-COVID-2019/td-p/961286`

non-visual mode (e.g., text, sound, or Braille) of communicating the information captured in chart images. In such circumstances, a brief text summary readout using assistive technology such as screen readers will assist non-visual users in understanding the chart image's essential information and providing the option to access the source data table to learn more about the chart. Therefore, effective source data table extraction and text summarizing of chart images may aid data analysts and business analysts make better data-driven decisions, as well as people with visual impairments and low cognitive capacities, interpret charts and perform analytical tasks using audio summaries [3].

Chart data extraction is the process of retrieving the underlying semi-structured tables from chart images. To extract the data tables and generate text summaries from chart images that capture this significant information in raster format, we need to study the human cognition behind how charts are perceived, termed graphical perception. According to Lohse's [4] cognitive model of graphical perception in humans, it is observed that early visual processes of perceiving graphs involve detection and encoding of visual primitives such as shape, position, color, and length of graphical objects. He also stated

that these apprehensions are made parallel; some require serial scanning and identification for patterns and anomalies—the final step of perceiving graphs is semantically understanding objects with the help of textual components and legend components. So, most chart interpretation mechanisms follow individual chart analysis of graphical objects (i.e., bars, pie sectors, graph lines and scatter points, etc.), which are collectively interpreted later to have a semantic understanding of the chart image.

Data extraction from chart images from semi-structured tables is a very challenging process given its stringent requirements of the proper interpretation of the text and graphical objects (e.g., bars or lines) [5]. Many image processing techniques are employed to prepare chart images for data extraction by segmenting graphical objects from the text. Some methods use connected component analysis to separate text from other graphical components, and more recently, deep neural networks are used for semantic segmentation of components. Apart from segmentation, various image preprocessing techniques are often employed for noise removal and effective text area detection in chart images. Noise removal operations involve image filters and morphological operations for handling background grids, tick marks, etc. However, it becomes challenging when graphical objects (e.g., graph lines or scatter points) exhibit similar graphical properties as these extraneous elements. Image preprocessing methods like cropping, resizing, and deskewing the image are also performed to handle multiple sizes, styles, and text orientations of text [2]. The state-of-art mechanisms even use various machine learning and deep learning models for object detection and semantic segmentation. Yet, the issue faced in such approaches is data availability for training the model. The available datasets [1,2] are synthetically-born chart images that lack other types of noise that are common in naturally available images across. So we geometry and color-based data extraction algorithm of image processing to interpret both synthetically generated and web collected chart images.

Another challenging aspect of chart data extraction is the vast design space of chart images. Hence most chart data extraction and interpretation algorithm confine their work to a single chart. Very few systems like ReVision [6] and ChartSense [7] worked on a wide variety of charts by providing a generalized approach at a few stages of the data extraction process that require similar interpretation (e.g., text labels, axis lines, etc.). However, developing a completely style-independent chart recognition system is not feasible due to the different data decoding formats used by each chart type. These chart recognition can even be semi-automatic, requiring human intervention to accomplish a few tasks [7, 8]. As a result, we focus on generating fully automated data extraction for commonly used chart types such as bar, pie, scatter, and line charts. Upon understanding the human cognition behind chart interpretation, our data extraction algorithm proposed chart component extraction as a crucial step of our workflow. As the individual interpretation of each component of a chart image is one of the early stages of graphical perception of chart images, we interpret the graphical objects and textual elements, e.g., chart title, legends, axis titles, and axis labels, separately.

Here, we define different elements of the chart image, as shown in Figure FC1.2, including the graphical objects, to formalize the inputs to our proposed algorithm.

Definition 1.1. Chart Objects: The graphical objects or marks in the chart representation that encode the data are called chart objects. Chart objects are objects with non-geometric/geometric shapes, e.g., bars/columns in a bar chart and a scatter point in a scatter plot.

Definition 1.2. Chart Text Components: The chart components are regions in the charts based on their location and roles. They include x-and y-axes that set the reference lines in two-dimensional space, x- and y-labels that provide the values being represented by the chart, title for both charts, as well as axes, a legend used for multi-series/multi-class charts. The title and legend provide meta-information of context and groups in charts.

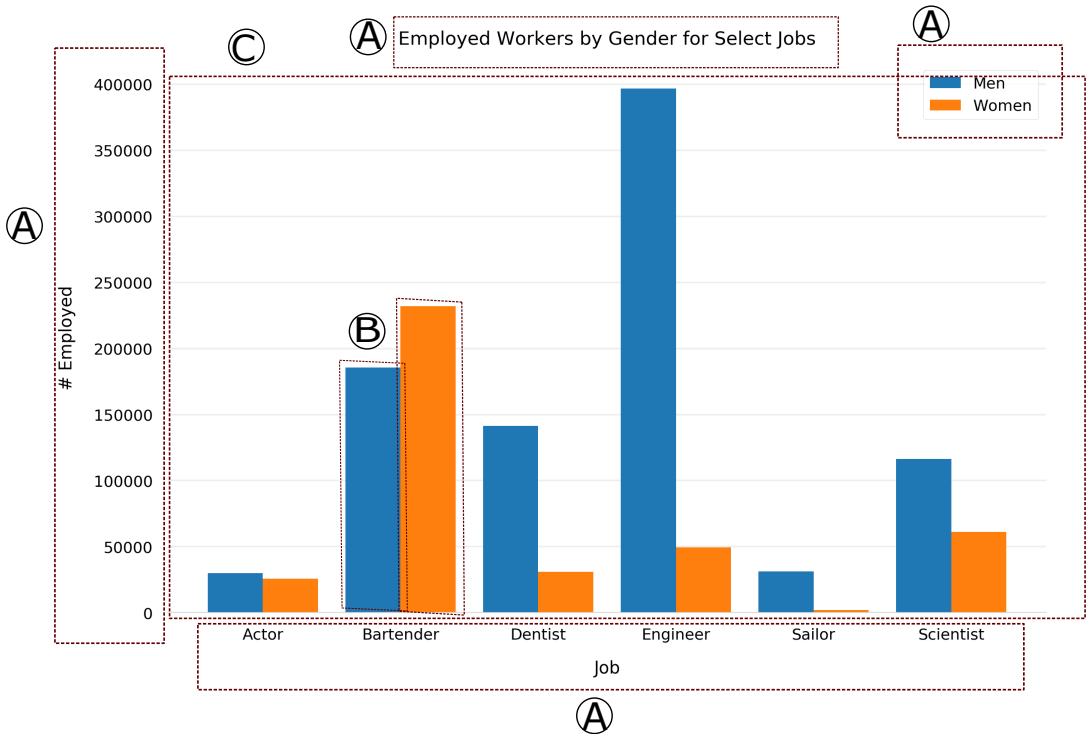


Figure FC1.2: Various elements of chart image highlighted in bounding boxes. (A) Chart text components such as XY axis labels, titles, legend, etc., (B) Chart objects such as rectangular bars/columns in this image, and (C) Chart Canvas excluding legend.

Definition 1.3. Chart Canvas: The chart canvas is a region of the image of a chart that contains only chart graphical objects, such as bars, scatter points, etc.,

A successful data table extraction will provide a substantial input for text summarization of chart images. A chart summary is more than just a language; it is also a concise representation of the chart image. The most difficult tasks in creating chart summaries are content selection and data interpretation. We need to set the scope of data and decide what goes into the summary because simply reading out the data is not ideal, as that may lose the interest of readers. So we thoroughly analyze and interpret the data using data analysis techniques of statistics or machine learning in search of significant features that need to be discussed in summary. However, these data analysis techniques often miss out on these visually perceivable details, such as bar height dif-

ferences in the grouped bar and cumulative height values in the stacked bar. The text summary will be addressed as a data summary instead of a chart summary unless it contains such visually perceivable details. We also need to make sure that a chart summary should be more generalized and engrossing to large groups of users with different levels of chart understanding. However, most of the chart summary generation mechanisms in the literature adopted statistical algorithms and template-based NLG (Natural Language Generation) methods, which due to their predefined structure, may lack generality and style variations [9]. Even the recently emerging neural network models for chart summarization fail to address this issue, as generating an appropriate dataset containing a large set of chart images and their ground truth summaries owing to various writing styles in humans is a new problem area to be explored. So, we proposed summary generation using the templated-NLG approach and designed a sentence structure based on the significant features of chart images estimated from source data table formats, data encoding styles of chart type, and user study observations.

1.1 Problem Statement

The automation of chart interpretation induces data extraction and text summarization from images of charts. The chart interpretation involves accomplishing computer vision tasks such as chart classification, object detection, text detection and recognition, semantically mapping graphical objects with text elements, etc. Our primary goal is to develop a chart interpretation system that inputs a chart image that decodes different components of the chart image based on chart type and sub-type and extracts the data represented in the chart image and its text summary. Our objective is to address the following problems:

- Extract the various components in the chart image based on semantics and usage.

  - A chart image is a collection of several meaningful components used to depict data effectively. These chart components are perceived in order and later assembled to form an overall interpretation of underlying information based on the steps involved in the graphical perception process. Identifying and extracting the essential elements based on their role in encoding data in chart images using image processing or deep learning models is crucial in interpreting chart images and extracting the source data.

- Decode graphical objects of chart image to obtain their properties and extract the underlying source data table

  - Charts render source data table values in graphical objects. So, chart interpretation systems focus on extracting properties of graphical objects by decoding, the process of storing data values based on each chart type. We need an effective data extraction technique to extract the properties of these objects in the image. Appropriate heuristic-based data transformations on data values obtained from these properties will give us a source data table.

- Generate text summaries of chart images from its extracted data table.

  - Data table representation of chart images may not be helpful in most cases when people want to have a quick interpretation of data insights demonstrated by chart images. A concise text summary will suffice for this purpose. We focus on addressing the challenges of the text summary generation process, such as content and data selection, grammar, and structure analysis, by using templated-NLG approaches.

The purpose of our work in this thesis is to explore and analyze diverse methods that have been implemented for chart interpretation and introduce our approach as an alternative and equally efficient solution for representing interpreted information into data

tables and text summaries. Our proposed methods work on popular chart images such as line and pie charts and widely used bar and scatter plots variants.

1.2 Contributions

We propose an algorithm that provides a label for image classification for a given chart image. Further, for predetermined chart types, proceeds to chart component extraction, geometry and color-based image processing for data extraction, and text summarization of extracted data table using template-based NLG approach.

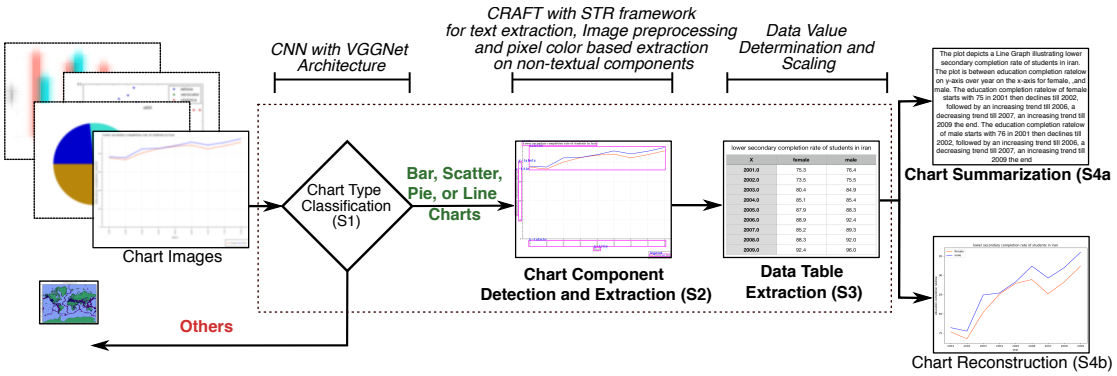


Figure FC1.3: Our proposed workflow for automatic data extraction from chart images using geometry and color-based extraction techniques based on their decoding mechanisms, extracted from multiple sources, e.g., existing data sets, internet, synthetically generated images, etc. *This thesis contributes to the tasks highlighted in the boldface format.*

In this thesis, we undertake the following tasks:

- we propose an end-to-end fully automated workflow (Figure FC1.3) for data table extraction, text summarization, and chart reconstruction across commonly-used variants such as pie and line charts along with subtypes of bar charts and scatter plots;
- we propose an automated chart component extraction method using image processing techniques for canvas and legend component extraction and text role clas-

sification using heuristic-based approaches;

- we determine appropriate image preprocessing techniques for removing extraneous elements from canvas such as gridlines, ticks marks, and undetected text fragments, especially in challenging cases like hollow bars and thin horizontal graph lines;
- we determine appropriate predefined deep-learning-based OCR models for effective text area detection in chart images with heavily aliased raster elements, numeric types, small font sizes, and non-horizontal orientations;
- we propose color-based and geometry-based object extraction techniques in image processing algorithms to decode graphical objects and extract the data table from chart image;
- we improve the efficiency of data extraction in scatter points variants and line charts by handling challenges like overlapping data points and transparent graphical objects in scatter plots and default interval computation in line charts;
- we propose a sentence structure based on the significant features of chart images estimated from source data formats and data encoding styles of chart type to generate an initial set of summaries using the templated-NLG approach;
- we conduct a user study to interpret participant understanding of chart images and comparatively assess our chart summary based on human rating, followed by redesigning our sentence structure to improve and generate a more generalized summary.

1.3 Thesis Structure

The thesis structure is as follows: in Chapter 2, we provide a literature survey on chart component analysis and chart data extraction mechanism, and different ways of

chart data representation, emphasizing the variety of implementations and their corresponding limitations. Chapter 3 discusses our preprocessing step for chart component extraction involving chart classification, canvas and legend component extraction, text detection, and recognition, along with the experiments made. Chapter 4 proposes an algorithm for data table extraction from chart images by decoding graphical objects and semantically inferring them with chart text components, and comparing the experimental results with existing methods. Chapter 5 focuses on design methodology for efficient chart summarization using sentence structure flow chart from extracted salient features of the data table, along with user study on bar charts for improving and assessing summaries. Chapter 6 concludes the thesis with the potential applications of our work, with a note on complete systems that can use our algorithm as a module and the scope of future work towards improving our algorithm.

CHAPTER 2

LITERATURE SURVEY

Over the years, a substantial amount of research has been conducted on chart image interpretation to assist people in understanding the charts better, especially for people with visual impairment or low cognitive abilities. The chart image interpretation involves unveiling the source information captured in rasterized chart image that requires the following visual recognition tasks: chart classification, text detection, text role classification, and graphical object detection. This chapter presents a detailed overview of the visual recognition tasks of source data extraction workflow over the past few years, including different ways of representing extracted data.

2.1 Chart Classification

The perceptual tasks such as judging position, angle, or slope graphical objects to interpret the chart image vary based on the type of chart visualizations. Earlier chart classification is usually done based on heuristics and domain knowledge of chart taxonomy [10]. Later, chart image classification is done using machine learning techniques like SVM (Support Vector Machine) [6, 11, 12], KNN (K-Nearest Neighbor) [12, 13], and ensembles to classify image feature vectors. Beagle [14] used object-based feature vectors from scalable vector graphics format. Much earlier than that, revision [6] used a bag of visual words approach to generate low-level feature vectors from vi-

sual dictionaries and small image patch clusters. The feature vectors computed for the statistical models fail to handle multiple variants and styles in chart images. The state-of-the-art work uses deep learning convolutional neural networks (CNN) for chart type classification. Few common CNN architectures used to classify chart images are VGG-Net [8], ResNet [15], AlexNet [16], and GoogleNet [7]. An empirical study on chart classification [17] states that the performance of chart classification using CNN models is affected by the quality of the chart images size of the training sample. Recently Region-based CNN (RCNN) [18] has been used in 2D and 3D pie charts classification.

2.2 Text Detection and Recognition

The data extraction methods mentioned in the literature predominantly analyze the graph and textual components separately [19, 20]. OCR (Optical Character Recognition) is prevalent for detecting and recognizing chart text that is further classified based on its role. OCR engines include open source and commercial systems such as Microsoft OCR, Tesseract [21], and ABBYY FineReader. Recent approaches replace end-to-end OCR systems with natural scene text detection models [12, 15] and deep learning-based CRNN models, the combination of convolution and recurrent neural networks for chart text recognition [15, 22].

2.3 Text Role Classification

Each text entity in the chart image, either individually or in a group, is interpreted according to its purpose. Based on their purpose or arrangement, the text regions of chart images are classified into the following components: chart title, axis title, axis labels (values associated with axis positions), and legend [5, 8, 15, 23, 24]. This process of classifying textual entities is often termed text role classification.

A variety of geometric based and layout based features of bounding boxes such as center coordinates, angle to image centers, the height of the box, and positioning around image borders are used to compute feature vectors [5, 15, 23, 24] and further classified using popular machine learning classifiers such as Random Forests, Naive Bayes, Decision Trees, and SVMs have been used to predict text roles. However, instead of machine learning classifiers, we heuristically identify text components based on the features mentioned above as the iGRAPH-Lite system [25].

2.4 Graphical Object Detection for Data Extraction

Chart data extraction by decoding graphical objects in chart images is the crucial step of chart interpretation. Most of the data extraction methods are specific for one chart type, e.g., iGRAPH-Lite system [25] for line charts, ScatterScanner [26] and Scatteract [27] in scatter plots, 2d-3d pie chart [18] extraction, and few systems on bar chart data extraction [10, 28–30]. Slowly there is an increased focus towards generalized systems to handle several chart types [6, 15].

However, these approaches use chart-specific graph object extraction techniques based on data decoding formats used by each chart type. A single generalized approach for all chart types is fairly possible due to the different decoding mechanisms used to represent each chart type. For example, in approaches like data extraction of chart images using tensor voting [8, 31, 32], the behavior of critical points of structure tensor is used to decode the graphical objects. The behavior of critical points varies for each chart type based on the morphology and intensity gradients of graphical objects (Figure FC2.1). The critical points are identified at the corners of the rectangular bars of the bar chart, throughout graph line in the line chart due to the staircase effect of lines in raster images, and at corners of a rectangle enclosed within the circular objects like pie and scatter points. Hence the chart data extraction systems like Choi *et al.* [15] use

real-time object detection systems like YOLO for graph bars detection and color-based extraction for decoding line and pie charts.

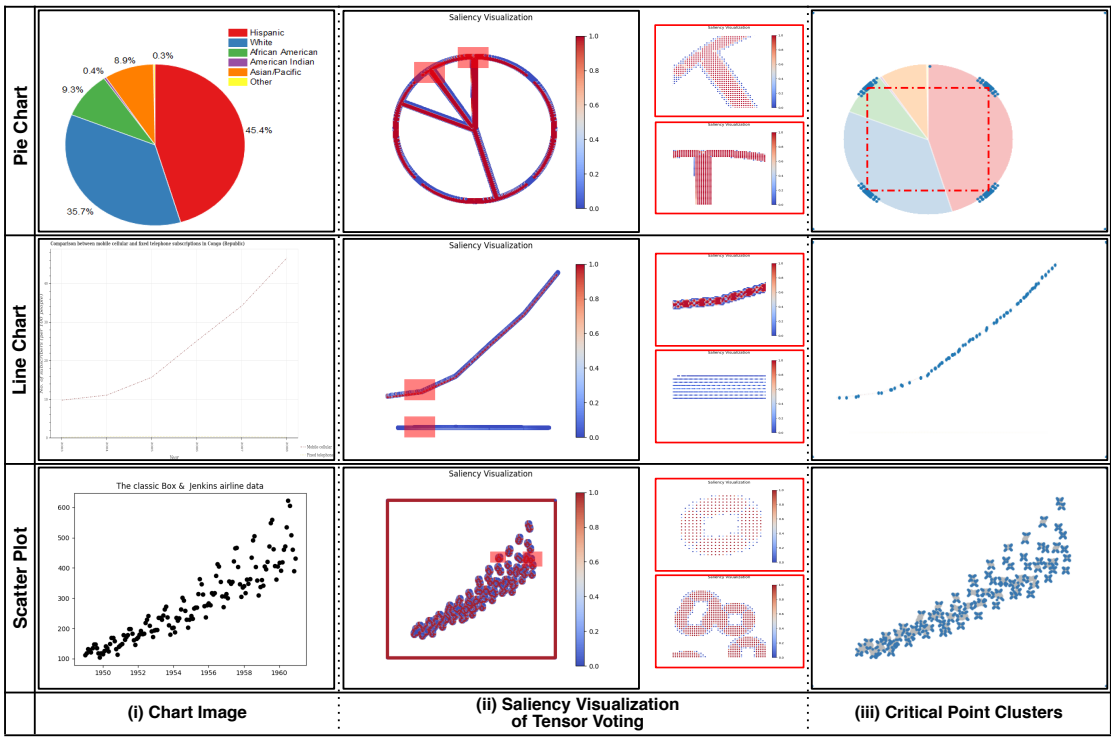


Figure FC2.1: Results of tensor voting approach in pie, line, and scatter plots (left) original chart image, (middle) saliency map visualizations of tensor voting in the canvas extracted from the image, and (right) critical points clusters formed at different locations of graphical objects.

Most color-based extraction methods [15, 22, 33] for decoding line, scatter, and pie charts work under constraints on chart images, such as having solid unique-colored graphical objects without any gradients and 3D effects. However, we often come across the chart image with a different graphical rendering of objects. The datasets released by recent data extraction competition on chart infographics like ICPR [34] used unique patterns to represent bar chart images. The data table extraction from chart images requires recognizing or isolating the graphical objects from other chart components. Traditional image processing methods, such as Hough transform and connected component labeling, have been used for graphical object detection [6, 7, 10, 24]. Instead of hough circle transform, ReVision used a curve fitting regression algorithm like RANSAC (Random

Sample Consensus) for circle detection in pie charts [6]. Apart from graphical object detection, Scatteract [27] uses RANSAC regression to predict the scale for transforming pixelated data obtained from graphical objects (Section 4.2). Scatteract [27] uses a deep learning object detection model called ReInspect [35] for scatter point detection, especially for overlapping points, as it proved very effective for person detection in crowded places. However, it doesn't focus on challenges due to the overlap of semi-transparent scatter points frequently encountered in a scatter plot variant called bubble plot. Few CNN-based image segmentation approaches [36–38] are used for isolating translucent overlapping objects with varying shapes trained on medical image datasets. However, we use best-fit circle detection techniques to segregate the overlapping scatter points and categorize circles into their respective classes based on the alpha blending mechanism of images.

2.5 Representation of Data Extracted from Chart Images

Representing the extracted data from chart images in human-readable format is the final step of the chart image interpretation. Instead of tabulating the data extracted from graphical objects, Chester *et al.* [10] used XML format to represent the data extracted from chart images. These XML formats have also been used to retrieve core messages from bar charts [29] for indexing graphics and summarizing multimodal documents. Recently, there has been growing interest in generating natural language descriptions of chart images [3,22,28,39–41] from the extracted source data table. The textual descriptions have been shown to assist visually impaired (VI) users read the visual summary of chart images using screen readers. Besides assistive technologies for the visually impaired, the generation of text summaries of chart images is used in an automated chart question-answering system [42, 43] based on user queries.

The natural language description challenges stem from content determination, selec-

tion of appropriate sentence structures, sentence organization, and text comprehension, followed by Linguistic Realization to improve the readability of text. These challenges are either addressed while designing language templates manually [8, 22, 28] or automatically taken care of while training deep learning models of natural language generation (NLG) processes. The templatized sentence creation in bar and line charts used a centrality-based PageRank algorithm [44]. An alternative approach for content selection is based on the design study methodology used in human-computer interaction (HCI) [45]. Further, the core message has been used for constructing the discourse and sentence structure in a brief text summary for bar charts [28], and a user study is designed in its improved version [9] to determine the level of importance of the different aspects of the content. Al-Zaidy [24] proposed building a semantic graph for computing salient features in chart images and predefined templates for text summary generation. Recent deep learning approaches like residual network (ResNet) [39], long short-term memory network (LSTM) [41], and transformer-based encoder-decoder architecture [40] have been used for chart summarization. The template-based is more predominant and more rigid content than advanced deep learning methods. As the former is easier to implement and sufficient for generic applications, we use a template-based NLG approach to generate master summaries for chart images.

In our workflow, we adapt the VGG-based convolutional neural network in BarChartAnalyzer [8] for chart image classification and bar chart classification. We then used image-processing techniques for noise removal and graphical object detection along with predefined deep learning OCR models to extract chart text such as titles and labels. These chart text components are heuristically classified according to their roles. We then implemented geometry-based approaches like second-order tensor fields [31] to extract corner points in bars and color-based image processing techniques to address the challenges faced in the data extraction of line, pie, and scatter plots. Finally, we generate text summaries from extracted data tables using templated NLG approaches.

CHAPTER 3

CHART COMPONENT EXTRACTION

A chart image is an association of several meaningful elements used for effective data representation. Each element has its special property that helps us interpret the data represented by the chart better. Hence, analyzing each image element and their association is a more meaningful and effective way to interpret the chart and reconstruct the underlying chart data rather than analyzing the entire image as a single entity.

Based on the semantics and usage of these elements chart image is segregated into seven components namely *canvas*, *legend*, *chart title*, *XY-axis titles*, and *XY-axis labels*. The process of locating and retrieving these components in chart images is termed chart component extraction. Unlike other computer vision, task charts component extraction is quite complex to handle as the components in the image are a combination of textual and non-textual elements (graphical objects and their markers). So the extraction of both textual and non-textual components needs to be handled separately. We use text detection and recognition models (Section 3.1) for textual component extraction such as chart titles, XY-axis titles, and XY-axis labels. The non-textual components, such as the canvas region contain the graphical objects, namely bars, graph lines, pie segments, and scatter points that are used to represent data. In the case of categorical data, the color and shape of graphical objects (graphical markers) as a visual channel are used to encode the data [46]. The legend component contains markers of these graphical

objects with their associated class labels. As canvas and legend components contain geometrical graph objects, they require separate extraction algorithms, which vary based on the plotting mechanism used in each chart type. So identifying the category of chart image is a crucial step for chart component extraction.

Apart from our focus on chart component extraction, analysis and interpretation of chart images vary based on how the chart is plotted. Data representation of each chart type gives us different insights into the same data. So identifying the type of chart before processing the chart image help us derive an insightful summary. Hence, the first step of our workflow is to classify the chart image into its respective category bar, scatter, line, and pie charts for the rest of our algorithm. We use the convolutional neural network (CNN) model with VGGNet architecture for chart classification, which has been used in the BarChartAnalyzer [8]. The model has been trained on four chart types bar, scatter, pie, line charts collected from the internet, and synthetic bar charts generated using the Python plotting library, `matplotlib`. Among these four chart types, bar and scatter plots are further visualized in multiple ways due to their vast design space. Hence, once we classify the chart image as the bar chart, we further train the classifier model to subclassify it into simple, grouped, stacked bars in horizontal and vertical orientations. Histograms with rectangular bins use the exact plotting mechanism as the bar and hence are considered one of its subcategories. Similar to bar charts, there is a need for sub-type chart classification of scatter plots. Owing to the scarcity of subtype data in scatter plots, we heuristically classify the scatter plot categories based on the properties of graphical objects and their taxonomy. The chart image classification hierarchy of different chart types and their subtypes chart images is shown in Figure FC3.1.

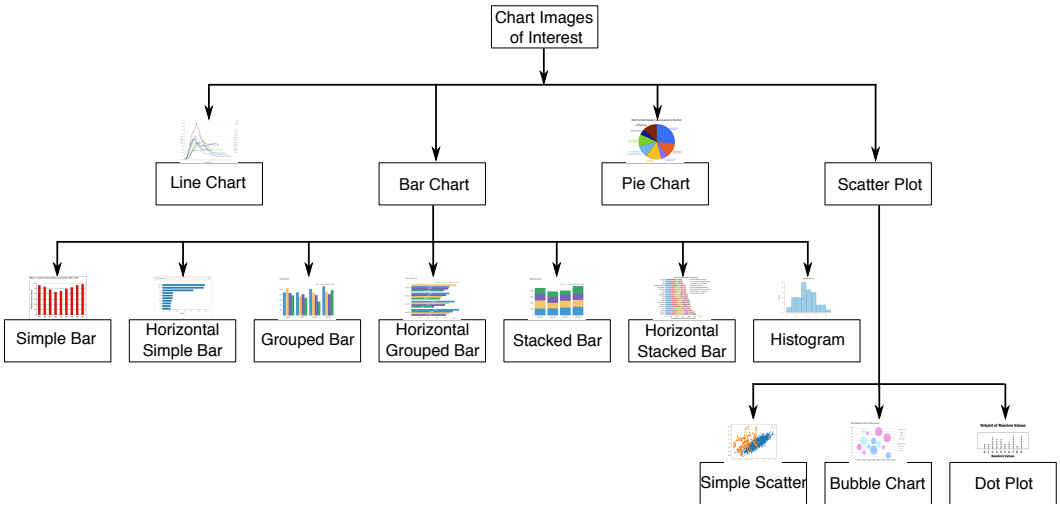


Figure FC3.1: Different types of chart images and its subtypes used in our proposed workflow

3.1 Text Detection and Recognition

A chart image without text labels is nothing but a drawing of symmetrical objects. The text labels add meaning to these objects and help us better understand the data and their insights. In our workflow, we use text components for data extraction and summarization processes. The main challenge is to extract the text labels encoded in pixels to machine-readable form. This process of converting digitally encoded text to machine-readable representation is called Optical character recognition or reader (OCR).

The OCR technology evolved in the early 1990s for text detection and recognition purposes. We have checked the suitability of an open-source OCR engine called Tesseract [47] with chart images. Tesseract has been found to work effectively with scanned document images with a clean background with regular font, plain layout, and single uniform color. However, the chart images we come across from scanned documents (Figure FC3.2(ii)) may often contain noise like a blur, illumination, etc., much similar to challenges faced in natural scene text, which deteriorates the performance of Tesseract-OCR in both text detection and recognition. We often encounter text elements with multiple numeric types and non-horizontal orientations of axis labels in

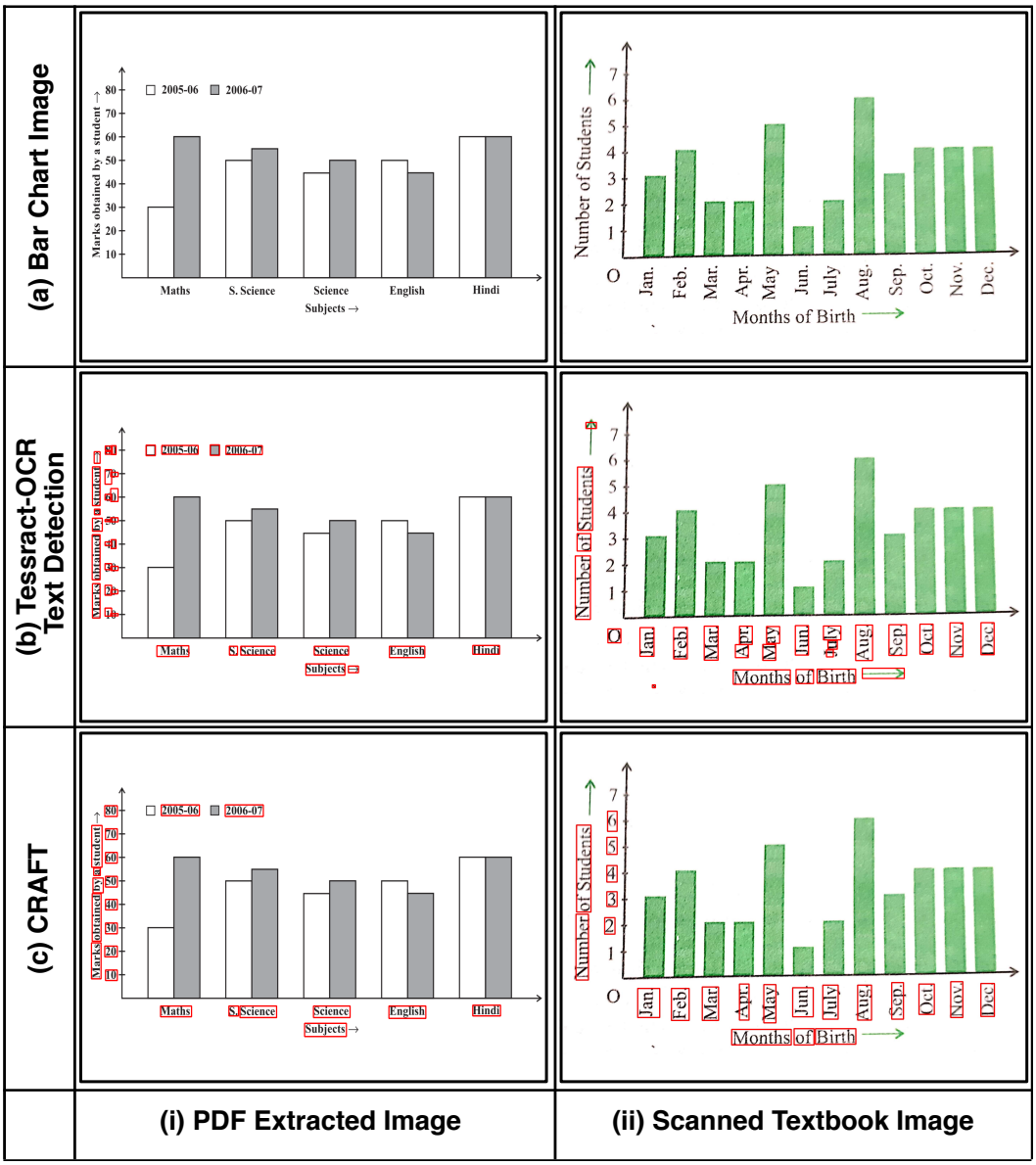


Figure FC3.2: Comparsion of text detection results using Tesseract-OCR and CRAFT pre-trained model. The red rectangular boxes are the bounding boxes of text predicted using text detection methods

chart images. All such variations in a single chart image make text recognition a challenging task. Also, the tesseract text recognition system tends to misinterpret non-text elements, such as symbols (e.g., arrows, currency) and color boxes in the legend to text labels which causes errors in the data transformation process (Section 4.2).

To address the challenges mentioned above, we have used a deep-learning-based OCR model, namely Character Region Awareness for Text Detection, CRAFT [48]. CRAFT text detection model is followed by a unified framework for scene text recognition (STR) that fits all scenarios [49] of text images. Given its proven efficiency, we use CRAFT with the STR framework together instead of end-to-end Tesseract-OCR for text detection and recognition purposes.CRAFT is designed for complicated scene text images, and it works by exploring each character region and affinity scores between characters. A CNN is designed to produce the character region score and affinity score. The character region score is used to localize individual characters and affinity scores to group each character to a single instance. There is no publicly available dataset to fit this method's criteria, so the model is trained using a weakly-supervised manner.

We use the pre-trained CRAFT model for our text detection process as a post-processing step. Then, we rotate predicted bounding boxes of the CRAFT model based on orientation computed from its coordinates for an effective text recognition using the STR model. The STR model is a four-stage framework with the following stages: transformation, feature extraction, sequence modeling, and prediction. STR resembles the combination of computer vision tasks like object detection and sequence prediction. So the framework includes a combination of CNN and RNN (Recurrent Neural Network) architectures. CRNN extracts CNN features from text images and reconfigures them with an RNN for robust sequence prediction.CRAFT, along with the STR model, effectively detects and recognizes text in chart images, including arbitrarily-oriented text found in charts with long x-axis tick labels (Figure FC3.2(c). On the other hand, tesseract falsely detects non-text elements like arrows legend color boxes in the image and fails to predict bounding box coordinates in numerical text regions (Figure FC3.2(b)).

3.2 Canvas and Legend Component Extraction

We now focus on extracting non-textual chart components such as canvas and legend components based on the chart type obtained from the chart classification. The canvas region is a crucial part of the chart image as it contains graphical objects that encode the data. The legend region contains the graphical markers of objects and their text labels associated with each category. Apart from graphical objects, the canvas region also contains axis lines, grid lines, and tick marks to aid chart reading by highlighting data in the chart image. However, the presence of these elements in the chart image will affect the performance of the system while decoding the objects present in the canvas region in data table extraction. Hence, the chart component extraction task includes canvas region extraction, which only contains graphical objects free from extraneous elements considered as noise. We use image processing algorithms for canvas extraction of images based on the geometry of graphical objects to identify and locate them in the image. We also use a few preprocessing techniques to remove extraneous elements such as gridlines, ticks, and small text fragments unidentified by text detection algorithms. However, the extraction of other chart components is independent of chart type.

3.2.1 Canvas Extraction from Bar Chart Images

The graphical object in the bar chart and its variants is a solid rectangular object which can be detected using object detection techniques. However, we encounter hollow rectangular bars filled with a background texture in a few cases, which are treated as a set of the edge lines of bars rather than objects. Also, distinguishing between gridlines and edge lines of the bar is a challenging task. Edge strength is the significant and relevant geometric property that distinguishes the gridlines from the bar lines. The gridlines have comparatively weaker edges than the bar edge lines. Advanced edge detection techniques like Improved Canny edge detector for color images [50] perform

well to segment edge lines of the bar from gridlines properly, as the Canny operator helps us remove weaker edges using the Non-Maxima Suppression technique. The Canny operator computes the edge strength based on the pixel intensities in the image that make the less saturated bars segmented along with gridlines. However, the canny operator may tend to add noisy edges in cases of the bar with varying gradients, e.g., stacked bar. Also, the noise-like text fragments obtained after removing text boxes from the image are not handled by edge detection techniques like canny.

Apart from edge and object detection techniques, simple segmentation like thresholding followed by contour detection does not effectively segment noise from the image. The marker-based watershed algorithm is used for complex image segmentation, which works effectively by labeling the sure foreground and background objects. Watershed transformation on grayscale pixel intensities focuses on balancing the intensities at local minima with its neighboring gradients by assigning minimal regions with distinct labels accordingly, thereby giving us the segmented result. However, noise and irregularities in the image cause over-segmented results. So to make segmentation more robust, we use marker-based watershed transformations [51]. Markers are the foreground and background labels given to regions of the image by morphological operations. Apart from foreground/background, we come across regions that do not belong to both, usually edges of objects which are labeled as unknown. The foreground, background, and unknown regions are obtained with the help of the following morphological operations defined below. Morphological operations apply a structuring element to an input image, creating an output image of the same size. Each output pixel corresponds to the neighbors of input pixels. Our proposed approach uses an 8-point neighborhood for all the morphological operations.

Definition 3.1. Erosion: The value of the output pixel is the minimum value of all pixels in the neighborhood. In a binary image, a pixel is set to 0 if any of the neighboring pixels have the value 0. This operation shrinks the foreground objects in the image.

Definition 3.2. Dilation: The value of the output pixel is the maximum value of all pixels in the neighborhood. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1. This operation thickens the foreground objects in the image.

Definition 3.3. Morphological Opening: The opening operation erodes an image and then dilates the eroded image using the same structuring element. This operation removes noise in the image.

Definition 3.4. Morphological Closing: The closing operation dilates an image and then erodes the dilated image using the same structuring element. This operation fills small holes in objects.

Definition 3.5. Distance Transform: For each pixel in an input image, the distance transform assigns a number that is the distance between that pixel and the nearest nonzero pixel of the image. This operation highlights the centers of foreground objects in the image and draws the boundaries between closer objects.

We perform the following sequence of image preprocessing operations on the chart image to compute the markers of the chart image:

- **Histogram Equalization**

  We perform histogram equalization on grayscale images to improve the saturation of bars encoded in lighter colors (Figure FC3.3) to prevent them from misinterpreting as background regions due to their low intensities.

- **Thresholding**

  Thresholding helps us obtain a binary image used in morphology operations followed (Figure FC3.4(b)). We use a combination of Otsu's and Binary Inverse thresholding that helps us determine the optimal global threshold, which is the mean of peaks in the bimodal histogram.

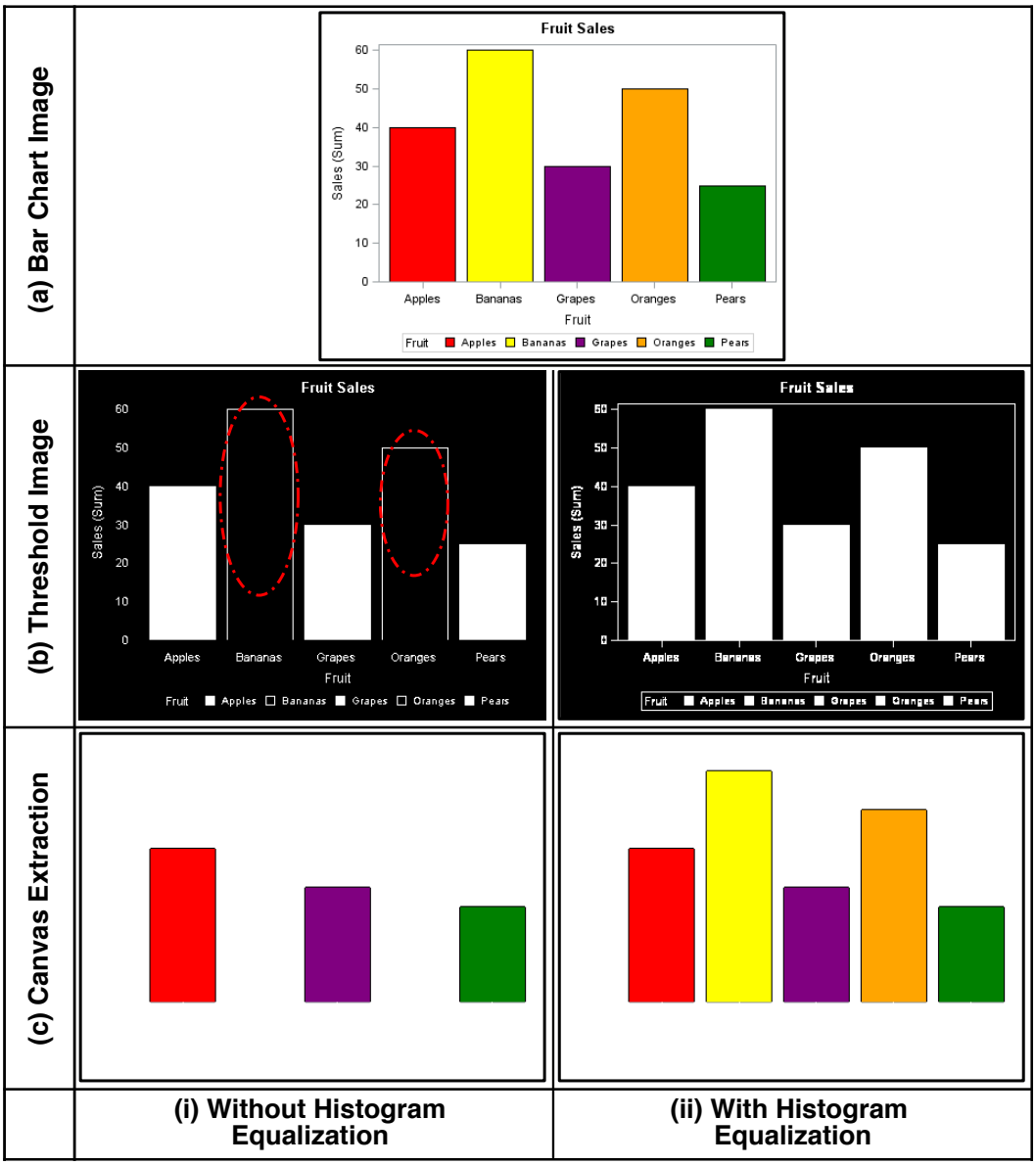


Figure FC3.3: Effect of histogram equalization on canvas extraction using marker-based watershed transformation on images with low saturation color fill

- **Morphological Opening**

  The binary image obtained from thresholding contains extraneous elements which are misinterpreted as white-colored foreground objects. These white noises, such as gridlines and tick marks, are thin lines compared to rectangular bars of chart images with less than 3-pixel width removed using morphological opening. The morphological opening erodes the boundaries of the white regions (foreground

object), followed by extending the boundaries of regions to revert them to their original size. In this process, very small and thin objects are completely removed during erosion, giving us a noise-free image (Figure FC3.4(c)).

- **Erosion Followed by Distance Transform**

  After noise removal, we are now sure the white regions only belong to rectangular bars. These white regions without boundaries (the unknown regions) can be marked as foreground regions. We perform erosion to remove the boundaries of objects and mark the region as the foreground region in the image. In some cases, when objects touch each other, we can't rule out the boundaries of objects using erosion. Hence, we apply distance transform along with the local thresholding to erode their boundaries

- **Dilation**

  Similar to the case of the foreground region, the entire black region apart from the area nearby boundaries if foreground objects are considered as background. So dilation allows foreground objects to expand their boundaries, and the remaining black region is marked as background. The remaining region, which is neither marked as background nor foreground, is marked as unknown.

Finally, after the above sequence of operations, we perform watershed segmentation on the grayscale image with its markers (Figure FC3.4(d)). We can see that watershed transformation can effectively segment the rectangular bars from the image, even in the case of gradients (Figure FC3.4(ii)). However, in the case of chart images with hollow bars, we consider background pixels to occupy more than 80% of pixels in the image. The foreground objects obtained from the threshold image, i.e., the rectangular bars, are filled with their edge colors bars with their edge colors and recompute the threshold image, followed by the same sequence of preprocessing operations to obtain the segmented image. To get the color of the edge lines of the bar, we perform a contour

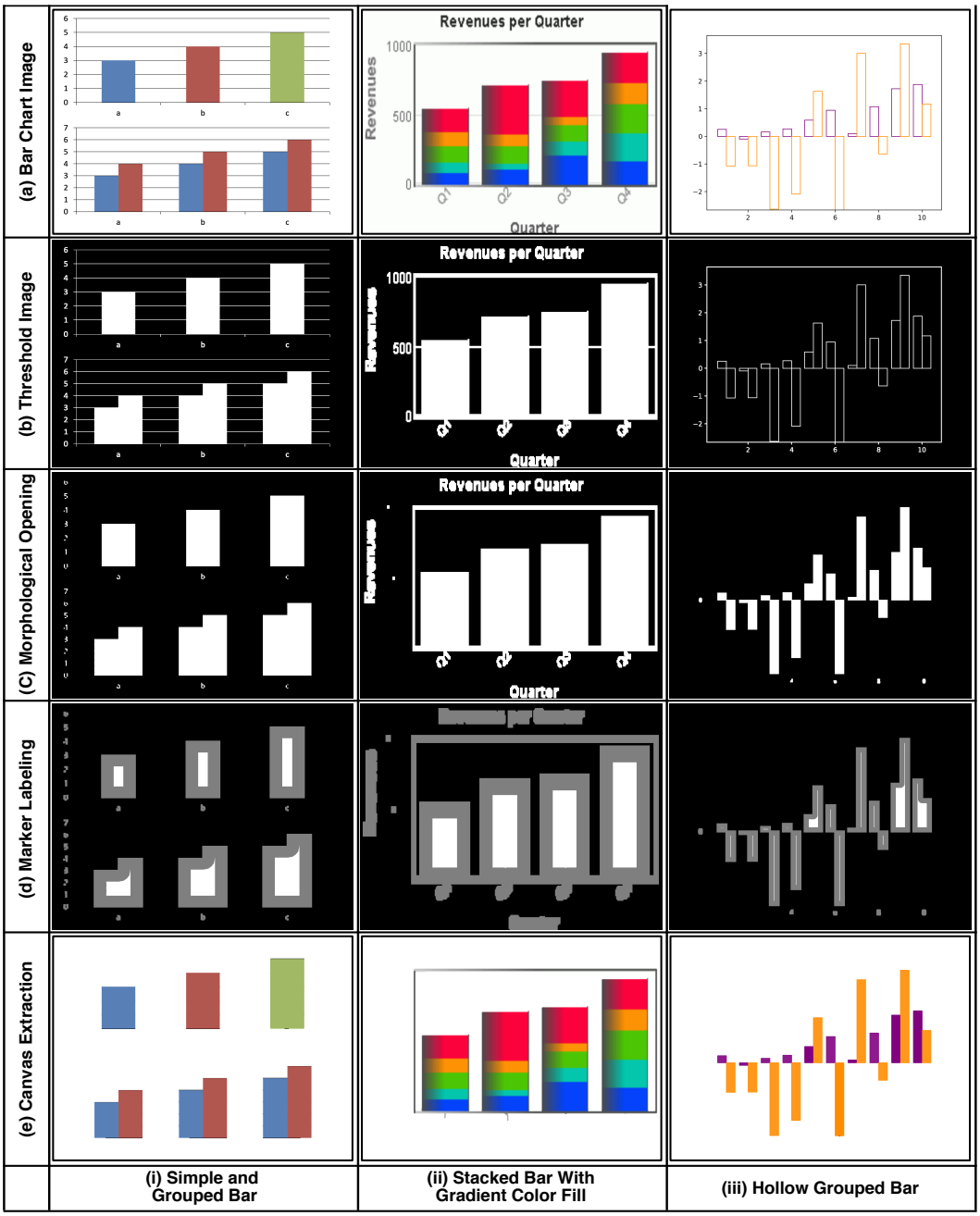


Figure FC3.4: Various image preprocessing stages of canvas extraction in the bar chart (a) Images of bar charts and their variants. (b) Threshold image of histogram equalized image. (c) Morphological opening for noise removal. (d) Marking foreground (white), background (black), and unknown regions (gray). (e) Extracted canvas image (only rectangular objects) by watershed segmentation of the marked image. *Hollow bars: (iii) Morphological opening for filled contours of threshold image with 80% of background pixels.*

detection technique that gives boundary lines of objects. We obtain contours colors and then compute the high pixel frequency border color of the graphical object to fill the contours with that color. The less frequent color pixels of contours are regarded as those belonging to extraneous elements; hence those contours are left unfilled. We can see that filled hollow bars are segmented effectively from the chart image (Figure FC3.4(iii)). We also use contours to highlight border pixels of bars to address the issue of highly pixelated edges in aliased images leading to uneven edges in each bar object which affects our data extraction process (in Section 4.1.1).

3.2.2 Canvas Extraction from Pie Chart Images

In the plotting mechanism of pie charts, only three chart components, canvas, legend, and chart title, are used to plot data. The canvas region of pie charts uses sectors that constitute the pie circle as graphical objects. So based on the geometric morphology of graphical objects, a simple circle detection technique, hough circle transform, is used to detect the canvas region.

Definition 3.6. Hough Circle Transform: The hough circle transform detects the circle like objects by a voting procedure in hough parameter space (a,b,r) with (a,b) being the center of the circle and r being the radius of the circle. Given an input grayscale image, the algorithm generates the canny edge image for a given threshold value. For each radius value in the estimated range, we traverse through each pixel and draw circles of that radius with edge pixels as the center. These values are stored in an accumulator matrix, and the voting is done to find the best candidate circles by selecting local maxima in the accumulator matrix.

The parameters like radius range, threshold values of canny edge detection, and accumulator play a significant role in candidate circle detection. We noticed pie circle detection performs well for the radius ranging from 1 to 1000, along with canny edge

detection and accumulator have threshold values of 50 and 30, respectively. The algorithm returns the best candidate circles in sorted order based on the no. of votes. So the region corresponding to the first circle, the one with maximum votes, is the canvas region of the pie chart. Apart from pie sectors, the canvas region often contains the text labels embedded within its associated pie sectors in the absence of a legend component. In such cases, we remove extracted text boxes as shown in Figure FC3.9(ii)(c) obtained from the text detection technique (Section 3.1) from the detected canvas region to extract the canvas region, which only contains pie sectors.

3.2.3 Component Extraction from Scatter Plot and Line Chart Images

The graphical objects in scatter plots and line charts use various shapes and line styles, respectively. Hence, geometry-based canvas extraction that works with predefined geometry of graphical objects is not helpful in the case of scatter plots and line charts. Moreover, in both scatter points and graph lines, the graphical objects exhibit similar properties as extraneous elements in the chart, e.g., in the case where the scatter points are smaller than the font size of textual components, and graph lines have similar saturation and width as gridlines (Figure FC3.9(iv)). Therefore, noise removal in the canvas region becomes more challenging, unlike rectangular bars and circular pies, where graphical objects are distinguishable from extraneous elements.

Firstly we get rid of textual elements for images with text boxes obtained from the text detection technique (Section 3.1). Later, we exploit the properties of periodicity and structure (conventional) in the grid- and axis lines, respectively, to distinguish them from the graph lines. We search for patterns in lines that demonstrate these properties and remove such lines, and the region enclosing the boundaries of those lines will help us locate graphical objects faster. We now focus on location graph objects from noise-free preprocessed images. Considering the size of graphical objects and diverse marker

styles, state-of-the-art segmentation and objects detection models fail in case scatter points and graph lines detection. So we implement color based extraction technique to extract graph objects where we obtain high-frequency color pixels from the chart image histogram. The clusters of these color pixels constitute graphical objects. The smaller pixel clusters of each color with a textual element in its closer proximity are considered legend markers and their associated text labels, thereby giving the bounding box coordinates of the legend component. The remaining clusters are enclosed together to give the bounding box coordinates of the canvas component.

The legend component extraction of bar and pie charts are done using the similar color-based extraction mentioned in scatter plots and line charts. We obtain high-frequency color pixels from the canvas region histogram of the chart image. The smaller pixel clusters of these colors, which don't contribute to graphical objects of canvas components and have textual elements in their proximity, are considered legend markers. The region enclosing these legend markers and their associated text labels gives the bounding box coordinates of the legend component.

3.3 Text Role Classification

After color-pixel-based and geometry-based extraction of canvas and legend components, the extracted text boxes from the CRAFT text detection model (Section 3.1) are now heuristically aggregated and labeled into textual components of chart images, namely chart title, axis labels, and axis titles. Firstly, the text boxes associated with to legend component are excluded. Now, based on the structural arrangement of these boxes around the canvas region, we segregate text boxes. The components' y-axis labels followed by the y-axis title are placed to the left; the chart title is above, and the x-axis labels followed by the x-axis title are found below the canvas component (Figure FC3.9(d)). These components segregated are clustered and labeled using this spe-

cific layout pattern with their text box centers. The boxes with the same y-axis values are grouped as y-axis labels and y-axis titles. The boxes with the same x-axis values are grouped as x-axis labels, x-axis titles, and chart title. In pie charts without legend, the text label association is done based on the proximity of text box centers with arc centers of the pie sector.

3.4 Evaluation Metrics

This section discusses and defines several metrics and terms that we are going to use for quantitative evaluation of chart component extraction and chart data table extraction of pipeline discussed in Chapters 3, 4.

Definition 3.7. IOU Score: IOU (Intersection Over Union) score or Jaccard index is the extent of overlap between the ground truth region and predicted region, which is nothing but the ratio of the area of intersection to the area of union of actual and predicted bounding box. In object detection problems, the IOU score is used to evaluate the prediction coordinates of the bounding box. We predict the accuracy of canvas component extraction using the IOU score.

$$IOU = \frac{\text{area of intersection}}{\text{area of union}}$$

Definition 3.8. True Positive (TP): It is a case where a test result correctly indicates the presence of a condition or characteristic. In terms of class prediction using a model, a true positive is an outcome where the model correctly predicts the positive class.

Definition 3.9. True Negative (TN): It is a case where a test result correctly indicates the absence of a condition or characteristic. In terms of class prediction using a model, a true negative is an outcome where the model correctly predicts the negative class.

Definition 3.10. False Positive (FP): It is a case where a test result wrongly indicates

that a particular condition or attribute is present. In terms of class prediction using a model, a false positive is an outcome where the model incorrectly predicts the positive class. The false-positive cases lead to the type-1 or commission error. The type-1 error refers to the non-acceptance of a hypothesis that ought to be accepted.

Definition 3.11. False Negative (FN): It is a case where a test result wrongly indicates that a particular condition or attribute is absent. In terms of class prediction using a model, a false positive is an outcome where the model incorrectly predicts the negative class. The false-negative cases lead to type-2 or omission errors. The type-2 error refers to accepting a hypothesis that ought to be rejected.

Definition 3.12. Precision: Precision or positive predictive value is the fraction of relevant instances among the retrieved instances. It is the ratio of the number of true positives to the total number of elements labeled as belonging to the positive class.

$$\text{Precision } = \frac{TP}{(\text{TP} + \text{FP})}$$

Definition 3.13. Recall: Recall or sensitivity is the fraction of relevant instances that were retrieved. It is the ratio of the number of true positives to the total number of elements that actually belong to the positive class.

$$\text{Recall } = \frac{TP}{(\text{TP} + \text{FN})}$$

Definition 3.14. F1-Score: The traditional F-measure is a measure that combines precision and recall is the harmonic mean of precision and recall. We F1-Score to predict the accuracy of our data table extraction process.

$$\text{F1-Score } = \frac{2 \times \text{ Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}}$$

Definition 3.15. MAPE: Mean absolute percentage error (MAPE) is the relative error that measures the prediction accuracy by getting the average absolute percentage of ratio

differences between actualA_t and estimated valuesF_t to the actual values. In machine learning, MAPE is used as a loss function to measure the total deviation of model predictions from the actual values. We use MAPE to evaluate the error rate of our data table extraction process.

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

3.5 Experiments and Results

In this section, we have evaluated the performance of the canvas extraction module and text detection and recognition models in chart images. For evaluating the canvas extraction process, We have taken a test set of 250 images each for bar, pie, line, and dot-line chart images of FigureQA [1] *validation dataset-1*. FigureQA dataset is a synthetical generated visual question answering and reasoning dataset on chart images. It contains bounding box annotations for all plot elements and question-answer pairs to chart images. The chart images in the dataset use colors to identify graphical objects, with a resolution of 72 DPI (Dots Per Inch). The validation dataset-1 uses color schemes from the X11 color set with 100 colors to generate chart images on a white background. The bar chart images in the test set include simple and grouped bars in both horizontal and vertical orientations along with grid lines. We include dot-line plots in our canvas extraction test set as they are similar to simple scatter points in visual presentation despite the difference in chart interpretations.

We evaluate the effectiveness of our canvas extraction process quantitatively by using the IOU scores as shown in Table TC3.1. It is one of the popular metrics for evaluating the performance of segmentation problems in computer vision. IOU score is often termed the Jaccard index, and it quantifies the percent overlap between the ground truth region and predicted region. The equation (Eqn 3.1) defines the IOU computation between the predicted bounding box, B_p of the canvas region, and the ground truth

bounding box, B_{gt} obtained from graphical object annotations of the test set. These IOU values range from 0 to 1, and if it is greater than 0.5 is considered a PASCAL VOC metric, representing the system's better performance. Although the predicted bounding of the canvas region in the pie chart image almost coincides with the ground truth box with 0.55 IOU (Figure FC3.5(i)), the color pixels belonging to pie sectors outside the prediction box will make a greater impact on the data table extraction (Section 4.1.2). In the line charts, although the pixel information belonging to graphical objects is preserved (Figure FC3.5(iv),(v)), it causes a problem in text role classification tasks, as the structural arrangement of other components is dependent on the position of canvas region. The greater the IOU value better the prediction. So, we consider IOU thresholds at 0.9, 0.75, and 0.5 and evaluate the performance of the canvas extraction process for each chart type based on precision value. When the IOU value of the chart image is greater than the threshold, we consider it a true positive, otherwise a false positive.

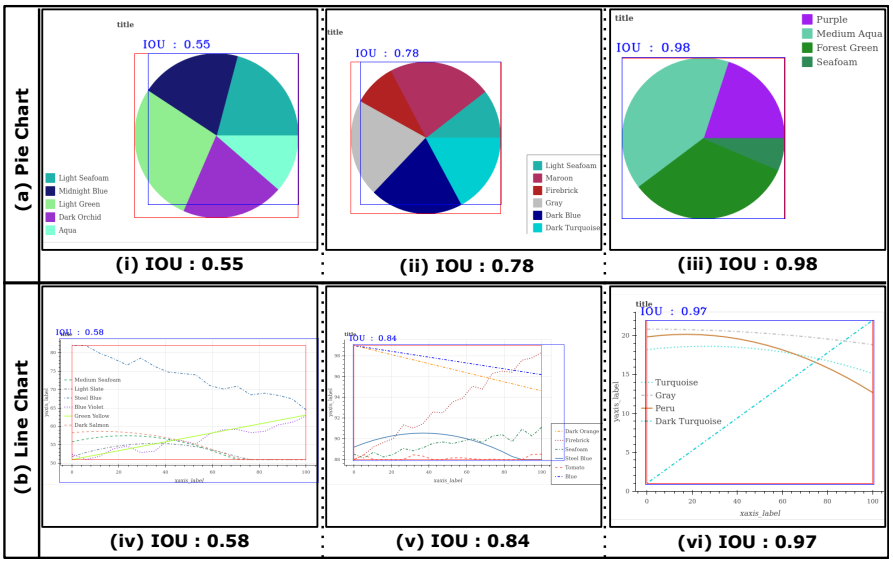


Figure FC3.5: The predicted (blue color) and ground truth (red color) bounding boxes for the canvas region with their IOU scores in (a) pie chart, (b) line chart images of FigureQA dataset.

$$IOU = \frac{\text{area}\,(B_p \cap B_{gt})}{\text{area}\,(B_p \cup B_{gt})} \qquad \text{(Eqn 3.1)}$$

Table TC3.1: The accuracy of canvas region extraction for each chart type at threshold IOU values 0.5, 0.75, and 0.9.

Chart Type	$IOU > 0.5$	$IOU > 0.75$	$IOU > 0.9$
Pie	88.8%	83.6%	76.4%
Line	90.4%	78.8%	77.6%
Bar	90.4%	86%	82%
Scatter	**99.6%**	**99.2%**	**96.8%**
Overall	**92.3%**	**86.9%**	**83.2%**

The canvas extraction process of the pie chart using hough circle transform is parameter dependent. The values like canny edge and accumulator thresholds of radius values affect the circle detection process as in Figure FC3.6. Although the chosen parameters are effective with ∼76.4% accuracy for a threshold IOU of 0.9, a few exceptional cases of chart images are not handled well. Moreover, the hough circle transform has a time complexity of $\mathcal{O}(n^4)$, where n × n is the size of the image. Thereby arising a need for advanced circle detection methods such as adaptive hough circle transform and RANSAC in the future.

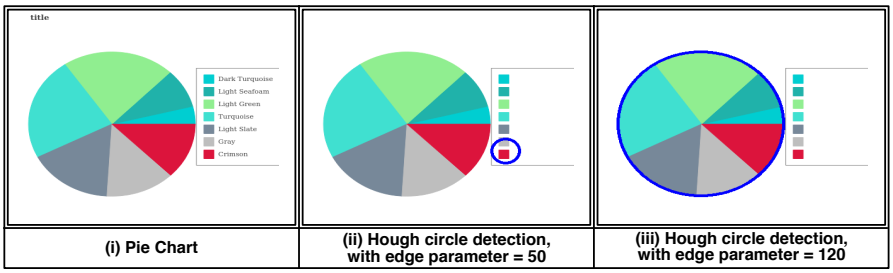


Figure FC3.6: Impact of canny edge detection threshold parameter in circle detection using hough circle transform on pie chart image from FigureQA [1] dataset.

The ineffective noise removal is a major contribution to the failure of the canvas extraction process. The canvas extraction using a marker-based watershed algorithm gives ∼82% accuracy in bar chart images. Although the test dataset images do not have exceptional cases like hollow bars, we still notice omission errors in the extracted canvas region due to the bars with small heights, as shown in Figure FC3.7(i).

The use of color-based extraction techniques in scatter plots, and line charts give

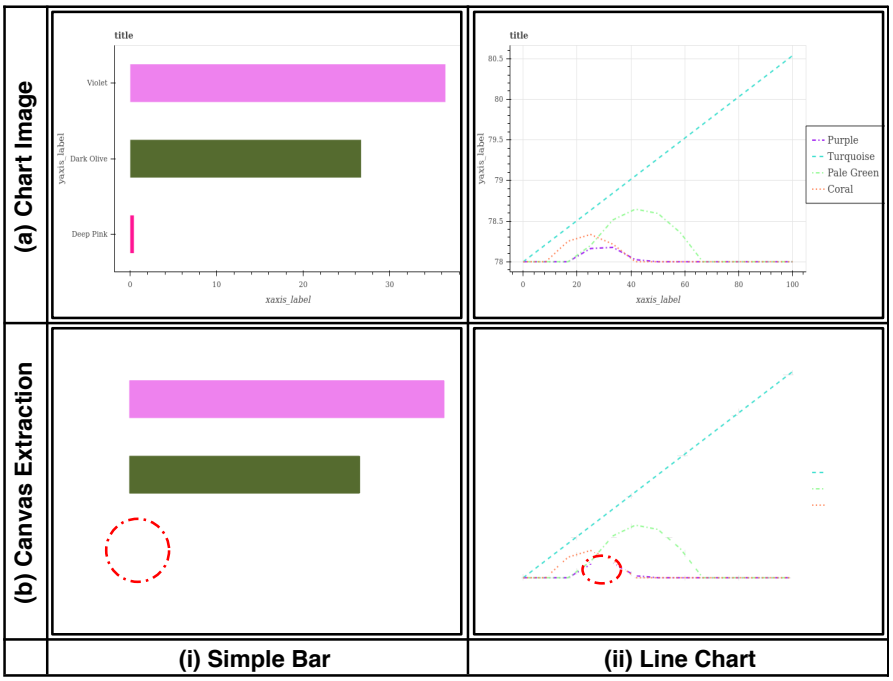


Figure FC3.7: The exceptional cases in canvas extraction of the bar and line chart images of the FigureQA dataset reported omission errors (i) missing bar with small height (ii) missing graph line due super-imposition of other lines.

$\sim$96.8% and $\sim$77.6% accuracy for canvas extraction, respectively. Despite the better performance of color-based canvas extraction over morphology and geometry-based techniques proposed in bar and pie charts, the line charts exhibit low performance because of exceptional cases where the graph lines are imposed on one another, as in Figure FC3.7. The lines imposed on one another cause the color blend of pixels to generate an unidentified graph line color apart from legend colors gets omitted as noise. Apart from the color blend, the graph line with small dots will have less pixel coverage area than other graph lines and gets unidentifed due to less frequency of graphical object color pixels. However, these superimposed dotted graph lines, small dotted lines, are even hard to interpret by the naked eye. Unlike line charts, the scatter plots have no transparency in the color of the graphical element and thereby giving the highest accuracy of $\sim$ 99.6%. It clearly states the canvas component extraction works exceptionally well in the case of color-based extraction on chart images that represent the graphi-

cal objects with plain colors. However, our proposed method may perform low in the case of *Adobe Synth Datasets* [34] of chart reading competition which uses textures and patterns along with color to represent graphical elements.

Our canvas extraction algorithm fails due to object sizes, such as bars with small heights and small dotted graph lines. However, we noticed that a dataset with synthetically generated high-resolution images hardly fails in such cases. So, we propose that improving the image resolution may increase canvas extraction efficiency.

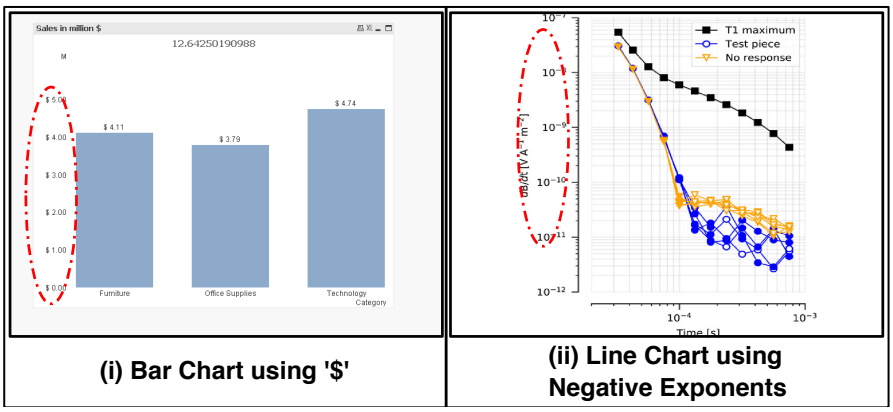


Figure FC3.8: Charts images using special characters in axis labels that our pre-trained text recognition model fails to identify, resulting in improper data extraction.

The CRAFT text detection model [48] and STR text recognition model [49] used in our algorithm yield an F1 score of 0.952 and 0.936, respectively, on the ICDAR-2013 dataset [52]. Despite high F1 scores, the text recognition model has issues in handling special characters such as $,%,£, sign (-), and cannot handle superscript symbols, e.g., degrees and exponents, as shown in Figure FC3.8. Also, the model misidentifies and confuses the alphabet 'O' or 'o', irrespective of the case, as the numeral '0' and vice versa in chart images and those with tiny font sizes, affecting the accuracy of the data scale. The model also doesn't recognize the minus sign, and negative values go unidentified in the chart images. Hence, we introduce conditions such as finding duplicate values that help us to determine negative values if the chart image has such data. Using a better text recognition model designed for recognizing chart text rather than natural

scene text is a plausible solution.

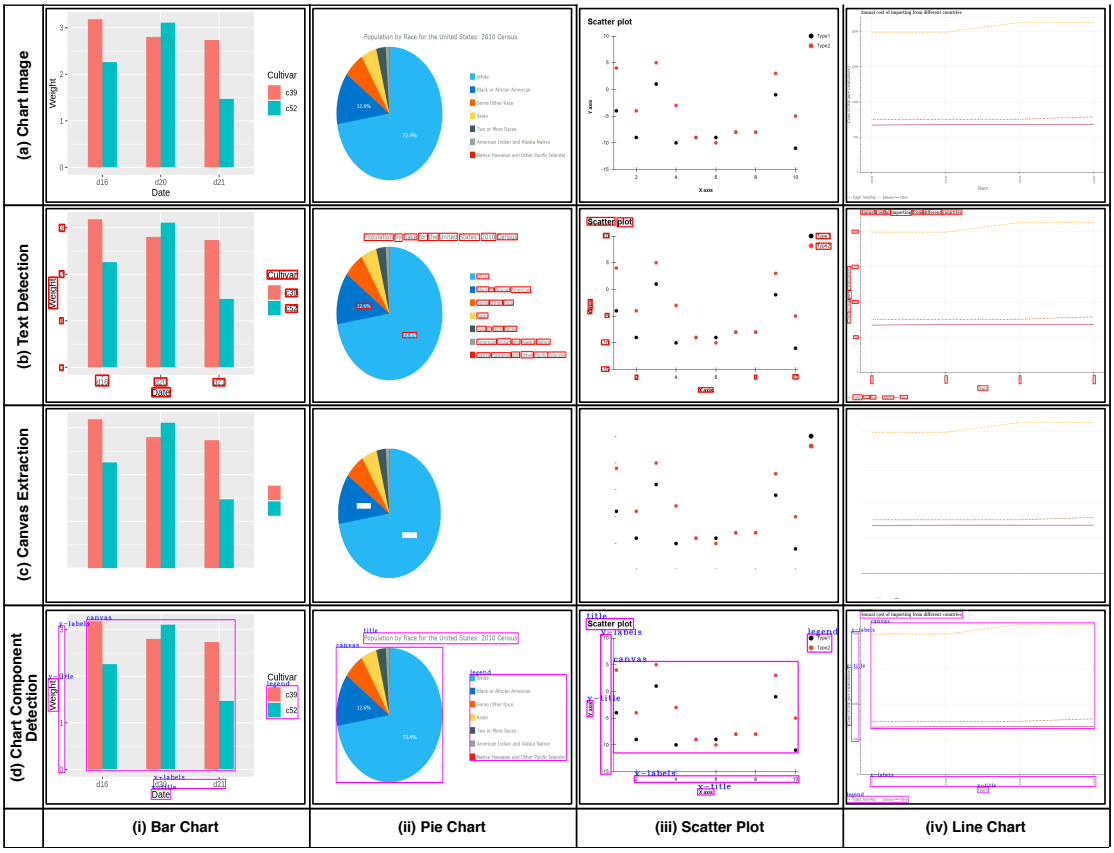


Figure FC3.9: Various stages of chart component detection and extraction (a) Input images of charts. (b) Textual elements detection using CRAFT pre-trained model along with its bounding boxes. (c) Canvas extraction of bar and pie charts based on their geometry, and scatter plot and line charts based on the frequency of color pixels. (d) Detected chart components with their predicted bounding boxes.

3.6 Summary

Overall, we conclude that the noise-free canvas extraction employed by image pre-processing techniques and robust natural scene text detection and recognition models like CRAFT and STR help us speed up the component-wise chart interpretation process resulting in data table extraction and text summarization of chart images. Apart from the synthetically generated question-answer dataset, our chart component extraction pro-

cess works effectively even in the case of chart images collected from different sources such as the web, documents, scanned images from textbooks, etc. Consequently, the improper canvas component extraction due to a few constraints like object size, noise, color blend, parameters of circle detection, and unusual text characters may impact the performance of the chart interpretation system.

CHAPTER 4

CHART DATA TABLE EXTRACTION

Chart images are capable of capturing large amounts of data efficiently in a graphical format and are a vital component of data analysis. Different chart types are used to represent data, but the trickiest part of data analysis is choosing the right way to represent data. It depends on the category of variables that need to be visualized and the purpose of our visual exploration process. Each chart used different visual encoding to represent these data variables. For example, bar charts use the length (height) and positional attributes of rectangular bars, and the pie chart uses the angle of pie sectors. The line chart uses position along with slope to encode the data values. In the case of categorical variables, the color of objects is used to encode each category. Scatter points also use color and positional encoding, but in special cases like bubble charts and dot plots, they use the size and arrangement of scatter points as their encoding, respectively. So one key aspect of chart data table extraction is to understand graphical notation based on its chart type. The graphical notation is visual encodings and attributes used by graphical objects to render source data variables. Decoding graphical object renders will help us extract raw data values in pixel coordinates. The raw data in pixel coordinates is transformed back to its original dimensions and units based on XY axis labels. The remaining chart components legend and XY axis title components will help us associate the data column with its associated label (variable name).

4.1 Data Extraction from Graphical Objects

The primary step of chart data table extraction is to extra raw data values in pixel coordinates based on the encoding used by graphical objects for each chart type. The canvas extraction in section discusses various geometry-based and color-based extraction techniques to isolate graphical objects in the chart image. We now process the extracted canvas image using geometry-based and color-based techniques according to its chart type to extract pixel data from these graphical objects.

4.1.1 Decoding Rectangular Bars

Bar chart images used the position and length/height of the bar to represent its data. In rectangular objects, corner point pixel coordinates are used to compute the position and length values. The popular corner detection techniques like Harris or Shi–Tomasi corner detection algorithms exploit the geometric and spatial properties of objects in images and use local geometric descriptors (i.e., the structure tensor of the image) computed from eigenvalues to extract corner points of objects in the image. However, these scale-invariant techniques either fail to detect a few corners of bars for a smaller scale or falsely detect edge points of bars along with corner points on a larger scale. So we use the closed-form tensor voting approach for chart images [31] which determines the likelihood of a point/pixel belonging to an edge line or corner point. Thereby helping us to identify point type features (whose saliency value, $CL < 0.4$) from the chart image structure tensor. The pixels with point features form small clusters at the corners of bars. We then compute the cluster centroid to get the corners of the bars (Figure FC4.1).

We now can compute data values from corner points, the bar heights along the y-axis, and bar center values along the x-axis. In grouped bars and stacked bars, these data points are further divided into groups based color of the center pixel of the bar. The

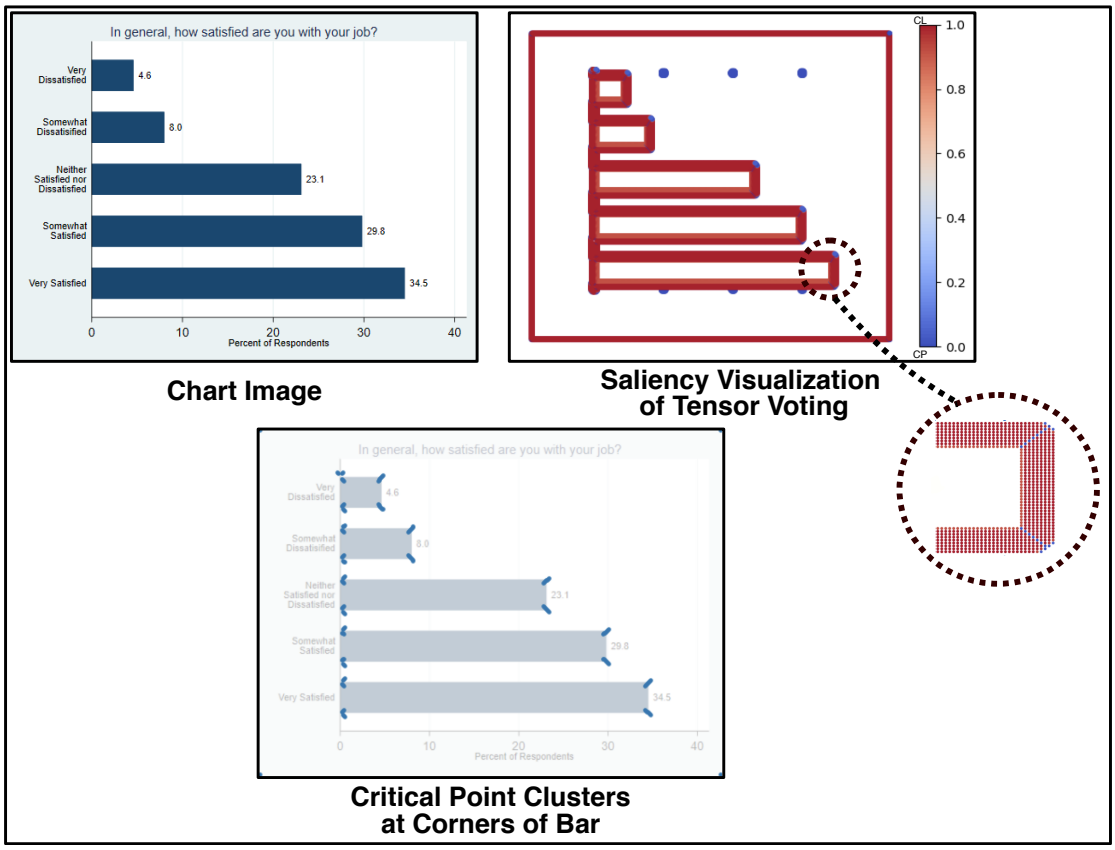


Figure FC4.1: The point type features, *i.e.,* blue-colored point clusters that occur at the corners of the bar in the saliency visualization of the tensor voting field.

corner point arrangement in stacked bars and histograms is different from the simple bar as in Figure FC4.2. In stacked bars, the bars are stacked upon one another and share corners at their junctions. At the same time, in histograms, the neighboring bins are attached, resulting in the extraction of only the top two corners of the bin, each for the starting and ending bin that has on its bottom corner. Hence, every bar subtype follows different corner point traversals for data values extraction. Corner detection using tensor voting works efficiently in the case of bar charts. Even in the case of stacked bars and histograms, critical point clusters are accurately formed at junctions of bar stacks and histogram bins, resulting in an effective data table extraction process in bar charts.

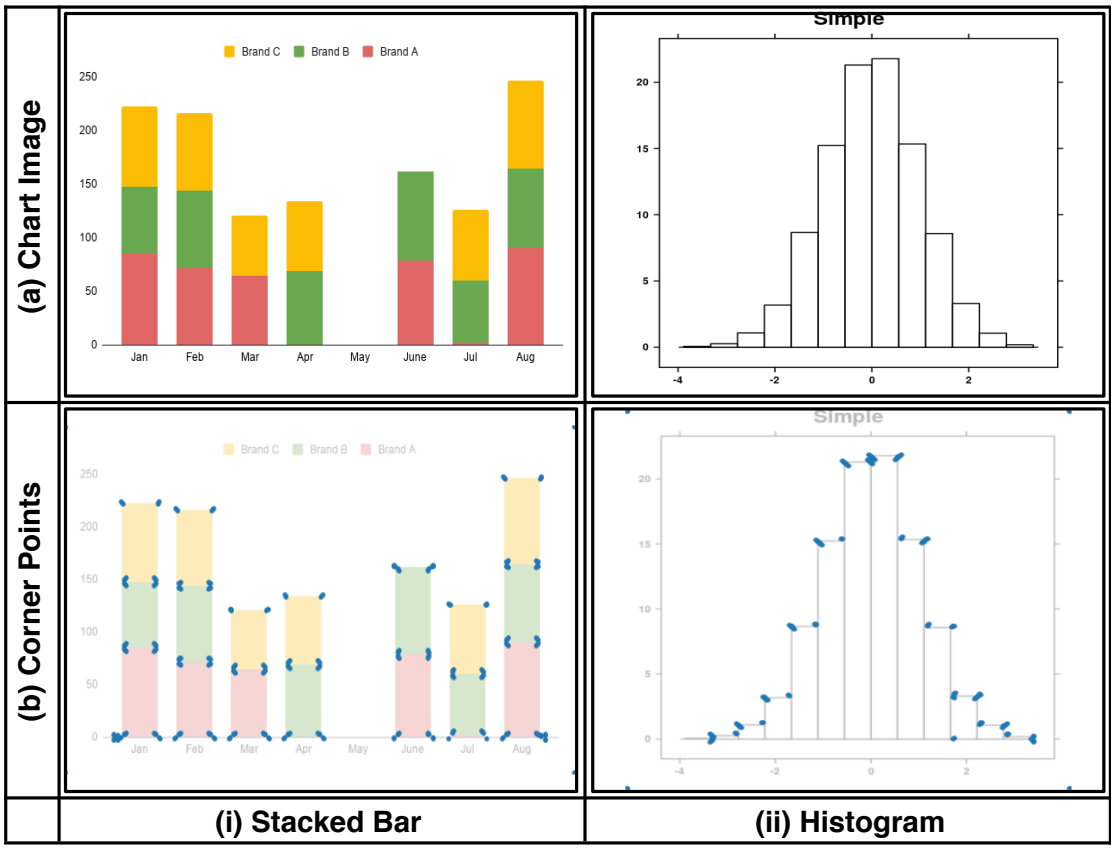


Figure FC4.2: Differences in automatic corner determination in stacked bars and histogram, owing to the clusters of critical points formed in the corners of the bar segments and the bars.

4.1.2 Decoding Pie Sectors

A pie chart is a graph with a dominant circular object, and its sectors represent portions of the entire pie, the 100 percent of a whole. They encode a set of continuous data variables using the angles made in the center of the circle. These angles are proportionate to the area of pie sectors which is constituted by the set of pixels in the sector regions of the canvas image [6, 15]. So we extract pixel counts of each sector based on its color histogram as raw data values of pie. Now the data values of each color are labeled using color and text labels association in the legend component.

4.1.2.1 Handling Pie Chart without Legend

We often come across pie charts without an explicit legend component. Instead, the text labels are implicitly embedded within or around the pie sectors. So the data values are labeled by mapping text clusters to their nearby/within pixel clusters formed based on their color. Another issue is the text labels embedded in pie sectors are considered noise and are removed in the canvas extraction process (Section 3.2.2). These pruned-out text boxes affect each sector's pixel counts, resulting in poor data extraction. To retrieve proper angle proportions of the pie, we get pixel counts of a hollow pie with one-third of the circle radius width to ensure accurate data extraction.

4.1.3 Decoding Graph Lines

Line charts are often called time-series graphs that visualize information that changes over time. The units of time are often plotted along the x-axis to observe data change along the y-axis. The plotting mechanism of the line graph initially plots a set of (x,y) data points and then draws a continuous line connecting points. The gradient change of line signifies the data change between variables. To extract data from graph lines, we first need to obtain a set of (x,y) discrete data points passing through graph lines in the canvas region. In the chart image, we process each graph line formed by a pixel cluster based on its unique color. Each graph line may have a continuous set of y-values for a given x-value (Figure FC4.3(b)), owing to the line width in the image. So, we average these sets of pixel locations along the y-direction to obtain all (x,y) pixel tuples of the graph line (Figure FC4.3(c)). Among all (x,y) pixel tuples, only a few contribute to source data values showing significant gradients on the graph lines. We may also come across a straight line with zero gradients for a constant variable of source data. So we compute a default interval value that covers all locations with significant gradient change on the graph lines and the x-axis label locations (Figure FC4.3(a)). The (x, y)

pixel tuples picked at uniform intervals from pixels of each graph line pertains to data points in pixel space. The use of significant gradient change values while computing default intervals to extract data points, thus improving the state-of-the-art [15].

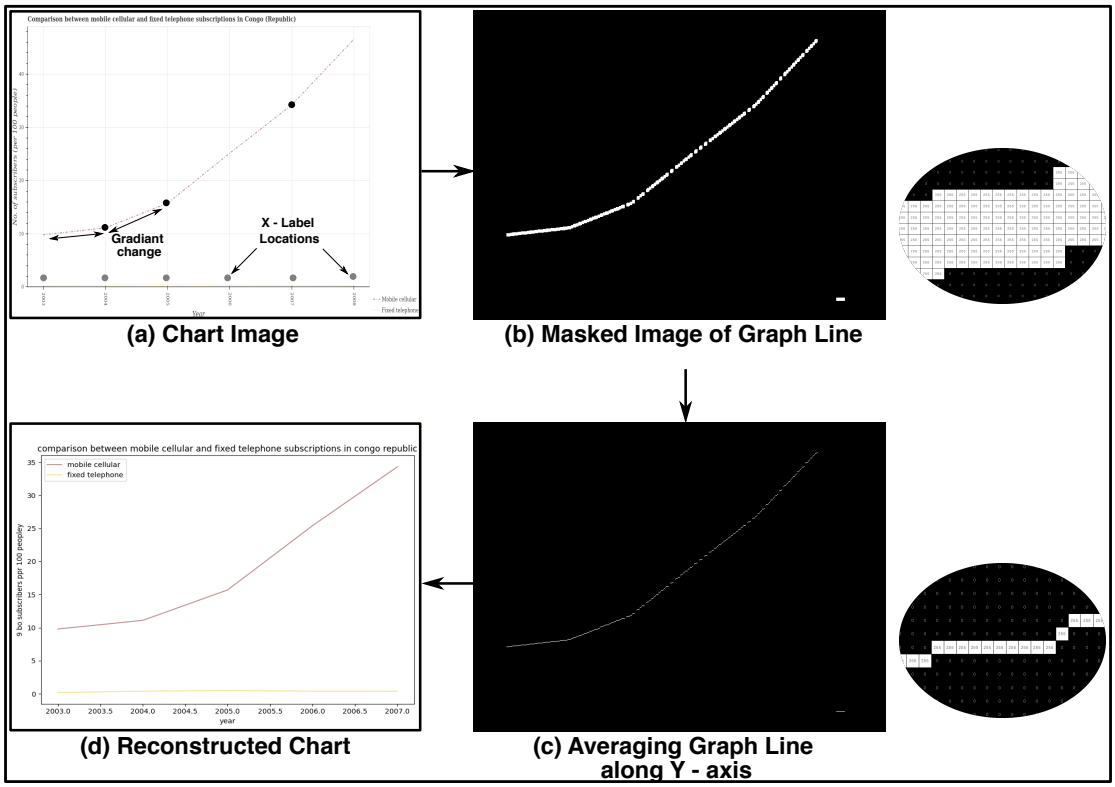


Figure FC4.3: Different stages of decoding line charts (a) A line chart image with x-axis label locations and gradient change locations are highlighted (b) The masked image of each graph line obtained based on pixel-color clustering (c) Graph line points averaged along the y-axis (d) Reconstructed chart image from the extracted data table.

4.1.4 Decoding Scatter Points

A scatter plot is a simple cartesian representation of two data variables generally used to observe the relationship between two variables, like identifying the correlation between the data variables, which helps us investigate the causality of events. The scatter points further encode new variables using the color, shape, or size of the scatter point. The bubble chart is a circular scatter point representing a new variable using size

encoding. The dot plot is another variant of scatter plots that shows the distribution of a single variable, similar to histograms. The difference is they use scatter points to visualize the count of data points instead of bin heights. However, the dot plot encodes colors to points to visualize the cumulatively stacked distribution of categorical variables. Based on the encoding of scatter points and the purpose of their usage, we use different data extraction techniques in scatter points.

So our first step of the data extraction process is to subclassify scatter plot images into their respective types. We extract contours scatter points in the canvas region. The chart image is classified as a dot plot if the contours are arranged in uniform stacks. If the contours are circular with varying sizes and have unidentified colors apart from the legend colors, the image is classified as a bubble plot. The image having contours without the above properties is classified as a simple scatter plot.

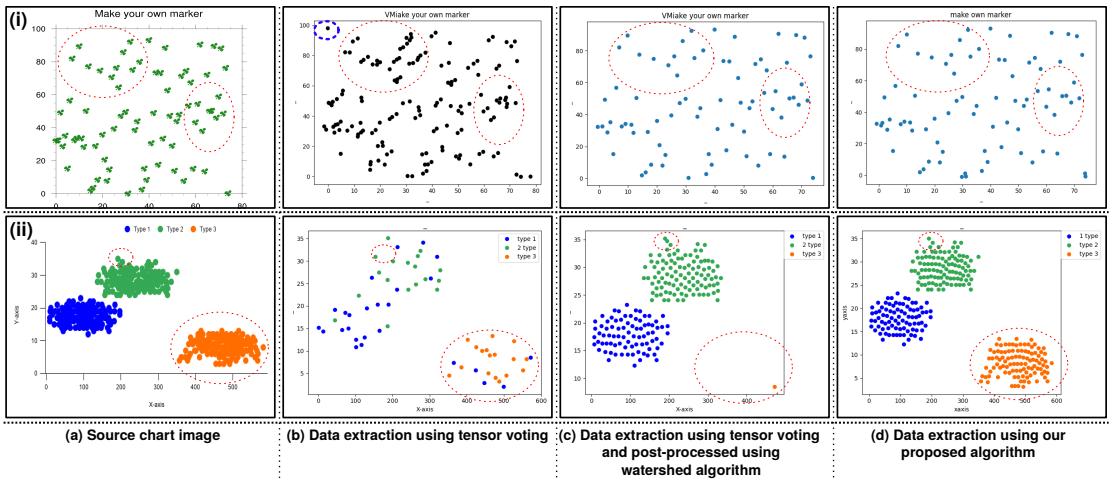


Figure FC4.4: Chart reconstruction using extracted data tables for sample source images of scatter plots. The comparisons between methods (red dotted ellipses), and the axis points misidentified as scatter points (blue ellipse) are highlighted. The top image is characterized by a clover-shaped scatter point, and the bottom one has cluttered circular scatter points.

4.1.4.1 Simple Scatter Plot

A simple scatterplot represents two or three variables in the two-dimensional cartesian coordinate system using color encoding for the third variable. The scatter point is represented in various shapes and colors. So, the data extraction process should be irrespective of geometry, unlike in bar charts (Section 4.1.1). Our ultimate goal is to get the pixel locations of each scatter point, which is nothing but the center pixel of the scatter point. The centroid of contours extracted in the subclassification step will give the center of individual scatter points or the location of overlapping scatter point clusters. If the contour area is n times greater than the area of the smallest contour*, i.e.,* the individual scatters point. Then we consider that contour is formed by a cluster of N overlapping scatter points where N is the nearest largest integer of n. So we now find N centers in counter using k-means clustering. Now the pixel locations of all centers are computed individually for each color, and data values are grouped based on color. The color-based extraction of contours on a simple scatter plot yields better results irrespective of the shape of scatter points and handles the overlapping scatter points more efficiently as in Figure FC4.4(d), which is a limitation of tensor field-based approaches [32]. We see the area proportions of cluttered scatter points with respect to individual scatter point used in our approach handles the overlapping scatter points better than instead of a clustering technique like DBSCAN (Density-Based Spatial Clustering of Applications with Noise) used in *ScatterPlotAnalyzer* [32].

4.1.4.2 Bubble Chart

Unlike simple scatter plots, bubble charts use circular scatter points with varying sizes. So handling overlapping scatter points with no fixed size becomes challenging as we can't use contour area to overcome this issue. Another issue we face in bubble plots is their transparency. Due to this transparent property of scatter points, the over-

lapping region of points renders into a new color based on the transparency value α of graphical objects using the equation (Eqn 4.1). This process of rendering pixels by combining foreground objects with background objects based on the α value is called alpha blending. The alpha blending of overlapping scatter points in the image makes it harder to categorize data points based on their color (Figure FC4.5(i)(a)).

$$I = \alpha F + (1 - \alpha)B \tag{Eqn 4.1}$$

where F, B, and I represent foreground, background, and the result pixels color, respectively; and $0 \leq \alpha \leq 1$.

We need to handle two main challenges in the data extraction of bubble charts: overlapping scatter points of varying size and scatter points of unknown color due to transparency. So we now use the circular shape of scatter points and alpha blending formula to address these challenges. First, we need to find the number of overlapping points in each contour extracted in the subclassification step. For each contour region, if the contour is encoded with pixels of a single color, *i.e.,* a scatter point without any overlap. For such contour regions, we compute extreme ends of contour that lie on opposite ends of the circle (scatter point) to get the center and radius of the scatter point. In the other case, we segment contours based on legend colors, and the no. of segments will give the no. of overlapping points (n_{sc}) within the contour. We then perform the spectral clustering method on an overlapping contour region given the estimated no. of overlapping points (n_{sc}) in that region and find the best fit circle using circular regression as mentioned in [53] to obtain circle parameters like center and radius.

The circle parameters like center and radius will contribute to data encoded by scatter points and are grouped based on their legend colors. We may notice a scatter point with unidentified color as a result of alpha blending, which arises due to the complete overlap of multiple scatter points with the same center. In such a case, we compute the

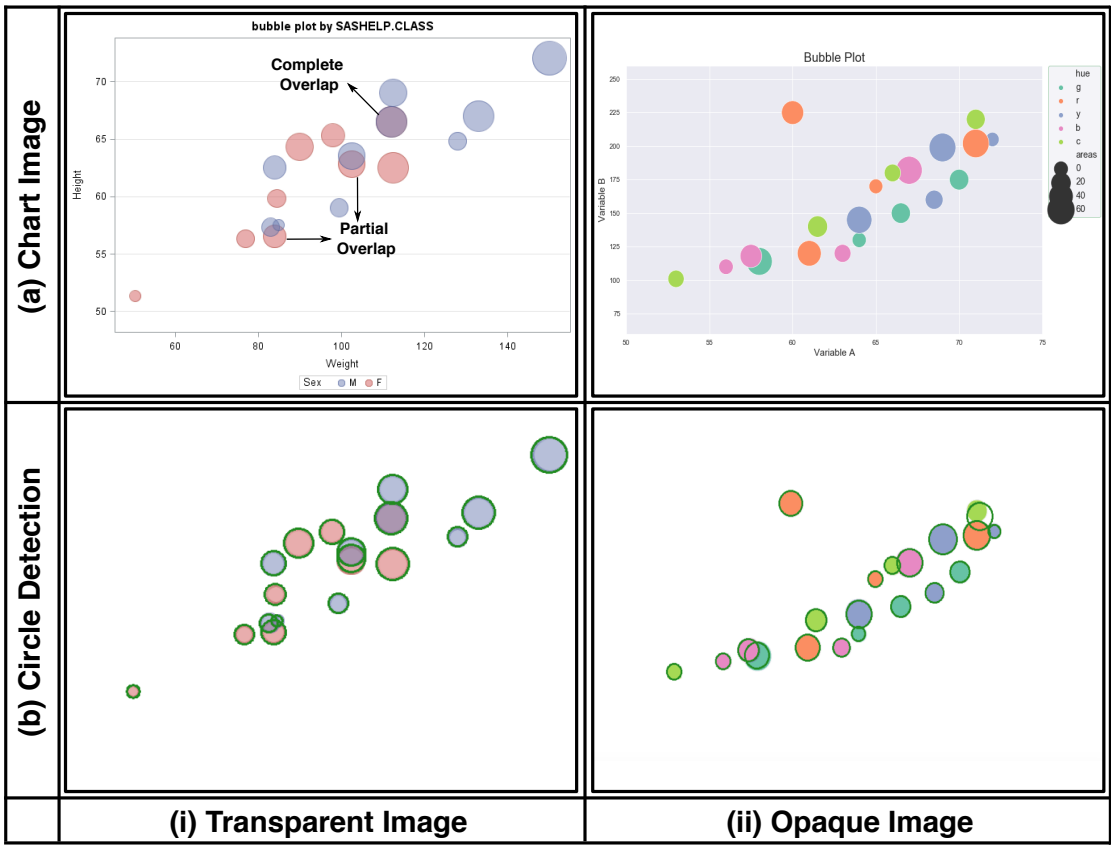


Figure FC4.5: Circle detection in bubble chart images by handling challenges like overlapping, varying size, and transparency using the proposed workflow.

combination of overlapped legend colors using the alpha blending equation (Eqn 4.1). The legend colors obtained from the image are the results of alpha blending the source color with the background color. But transparency value α is unknown in the image. So we compute a set of possible source color values for possible alpha values ranging from $(0.5 - 0.95)$ with a step difference of 0.5 using equation (Eqn 4.2).

$$C_{src} = \frac{C_{lg} - (1-\alpha)C_{bg}}{\alpha} \qquad \text{(Eqn 4.2)}$$

where C_{src} is the source color; C_{lg} is scatter point color in the image obtained from the legend component; C_{bg} is the background color of the image that contributes to almost 70% of pixels in the canvas region.

We now use a brute force approach to recognize the unidentified overlapping region colors by comparing them with rendered colors obtained from alpha blending all possible source color combinations. For each source color generated from a unique tuple of $(C_{src}, C_{lg}, \alpha)$ we generate a combination set of two source colors and compare unidentified colors with newly rendered colors generated using equation (Eqn 4.3). If any rendered color matches with unidentified colors, then both alpha and source color values are known, and the respective legend color pair replaces the unidentified color. We repeat this process of searching for other unidentified colors even by increasing the no. of source colors in combination set for every iteration until we find the unidentified colors or stop after five iterations and label the unidentified point as noise.

$$B_n = \alpha(C_{src_n}) + (1-\alpha)B_{n-1} \qquad \text{(Eqn 4.3)}$$

where α and B_0 are estimated transparency and background color in chart image, B_n is overlapping region color obtained from rendering a set of source colors $(C_{src_1}, C_{src_2}, .., C_{src_n})$, which may match unidentified colors in overlapping scatter point regions.

4.1.4.3 Dot Plot

In the dot plot, we do not encounter challenges like transparency and overlapping points. So we compute centroids of contours obtained in the subclassification process. We then count the no. of centers of scatter point contours with the same x-location values. We then group points based on their color identified by the legend component and map them with their respective labels. Finally, x-location values and their counts have been extracted in pixels coordinates.

4.2 Data Transformation

The data extracted from decoding the graphical objects is the raw data in pixel units. This raw data without text add no meaning to the data. The final step of the data table extraction process from chart image is adding semantics to raw data by labeling them textual components and legend component and scaling it back to their original unit obtained from XY axis labels. We first map legend labels for data groups segregated based on color. We label x-location and y-location pixel values in raw data with x-axis and y-axis titles, respectively. We rename the raw data table with the chart title.

After assigning the title, we now need to transform the data into its original unit. Before we transform data, we need to recognize negative text labels as the STR text recognition model does not recognize minus signs. In duplicate text labels, the first label we encounter and the numbers before them are converted as negative values. We apply the transformation equation (Eqn 4.4) on x-pixel (x_{pix}), and y-pixel (y_{pix}) location values derived from min-max normalization in data mining to give our extracted data table values in their original units/dimension.

$$\begin{aligned} x &= \frac{x_{pix} - xloc_{min}}{xloc_{max} - xloc_{min}} * (xl_{max} - xl_{min}) + xl_{min} \\ y &= \frac{y_{pix} - yloc_{min}}{yloc_{max} - yloc_{min}} * (y_{max} - yl_{min}) + yl_{min} \end{aligned} \qquad \text{(Eqn 4.4)}$$

where xl, yl are the x-, and y-axis labels obtained from the text recognition module corresponding to the center locations of their bounding boxes $xloc$ and $yloc$, respectively.

Sometimes charts use names instead of numeric as x-axis labels. In such cases, the x-location pixel values are replaced with text-label of its nearest x-value of the text box center. The data transformation in the pie chart varies due to the whole-part representation of data values; the data is stored in percentage format, *i.e.,* the percentages of each part in its whole 100%. So the percentage of pixel counts in the sum of all pixel counts

is our extracted data table values of the pie chart. In the case of bubble charts, the labels signifying the size of the radius are unidentified in most cases; hence the data values (radius of scatter points) corresponding to the size variable are left in pixel coordinates.

4.3 Implementation

This section discusses the overall implementation and the complexities of heuristic-based approaches and algorithms used in chart component extraction and data table extraction of our pipeline. The workflow is implemented in the `Pycharm 2018.3` tool on an Intel i5 processor equipped with 8 GB RAM in Mac OS 10.14.3. The complexity of our workflow varies based on algorithms implemented in each chart type. The canvas and legend component extraction process in chart component extraction has a time complexity of $\mathcal{O}(n^2)$, where n × n is the size of the image. However, in pie charts, the complexity increases to $\mathcal{O}(n^4)$ due to the hough circle transform. The text role classification of text elements has a time complexity of $\mathcal{O}(n)$, where n is the no. of text boxes in the image. In data table extraction, the tensor voting approach in the bar chart and the extraction of overlapping points bubble plots are exhaustive. In bubble plots finding the set of overlapping points using the alpha blending equation has a time complexity of $\mathcal{O}(2^n)$, where n is the list of expected source colors for unknown transparency. The tensor voting computation takes $\sim$2 minutes for an image of 192 DPI. Our current implementation of the tensor voting computation is serial and has the scope of parallel implementation in the future to reduce the execution time in the bar chart. The exhaustive algorithm such as hough circle transform and the extraction of overlapping points bubble plots need to be less time-consuming.

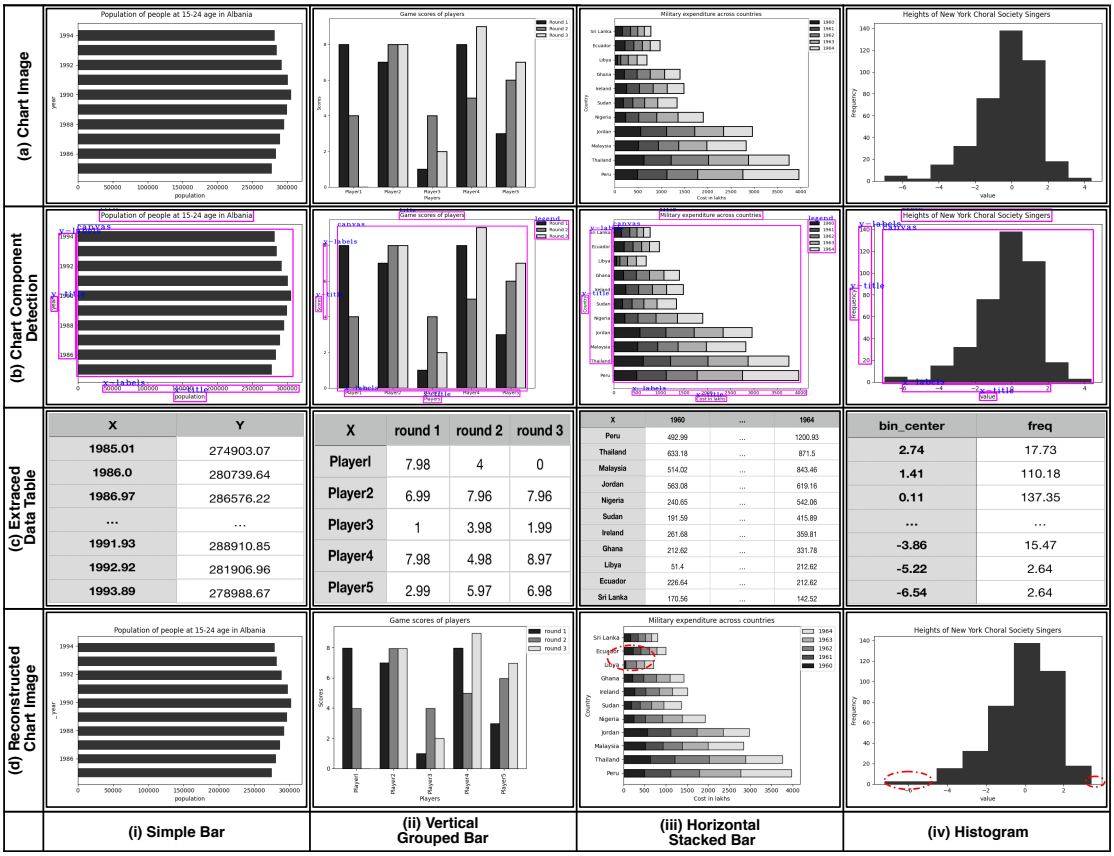
4.4 Experiments and Results

In this section, we have discussed the data table extraction results in chart images. We have tested our proposed workflow on chart images collected from FigureQA [1] and PlotQA [2] datasets and images downloaded from multiple sources on internet (Appendix A). We also use images of charts synthetically generated using `matplotlib` to compare extracted data table with the source data table used to generate the chart image. Our proposed data table extraction workflow in chart images is assessed based on two aspects: (i) how accurate is our extracted data table from the chart image to its source data table? (ii) to what extent can we reconstruct the chart image from the extracted data table, classified as quantitative and qualitative analysis, respectively.

4.4.1 Qualitative Assessment

The effectiveness of our algorithm is qualitatively analyzed by visually comparing the original chart image to the reconstructed chart obtained from the data table extracted using our proposed method. We observe these results at different stages of our proposed workflow for a sample of each chart type, both individually and collectively (Figures FC4.6 - FC4.9).

Visual comparison of original chart images of the synthetic dataset with reconstructed plots (Figure FC4.6) achieves near-perfect accuracy owing to high-resolution bar chart images created with standard or minimal formatting commonly across plotting libraries. The morphological methods for image preprocessing in Section 3.2.1 improve data extraction accuracy from low-fidelity images. However, we notice errors in data extraction in a few limited cases, specifically when DBSCAN clustering of corner points does not directly capture small/insignificant height differences between bars/bins [31]. We have identified two such cases. The first case with smaller bar/bin height, which



X	Y
1985.01	274903.07
1986.0	280739.64
1986.97	286576.22
...	...
1991.93	288910.85
1992.92	281906.96
1993.89	278988.67

X	round 1	round 2	round 3
Playerl	7.98	4	0
Player2	6.99	7.96	7.96
Player3	1	3.98	1.99
Player4	7.98	4.98	8.97
Player5	2.99	5.97	6.98

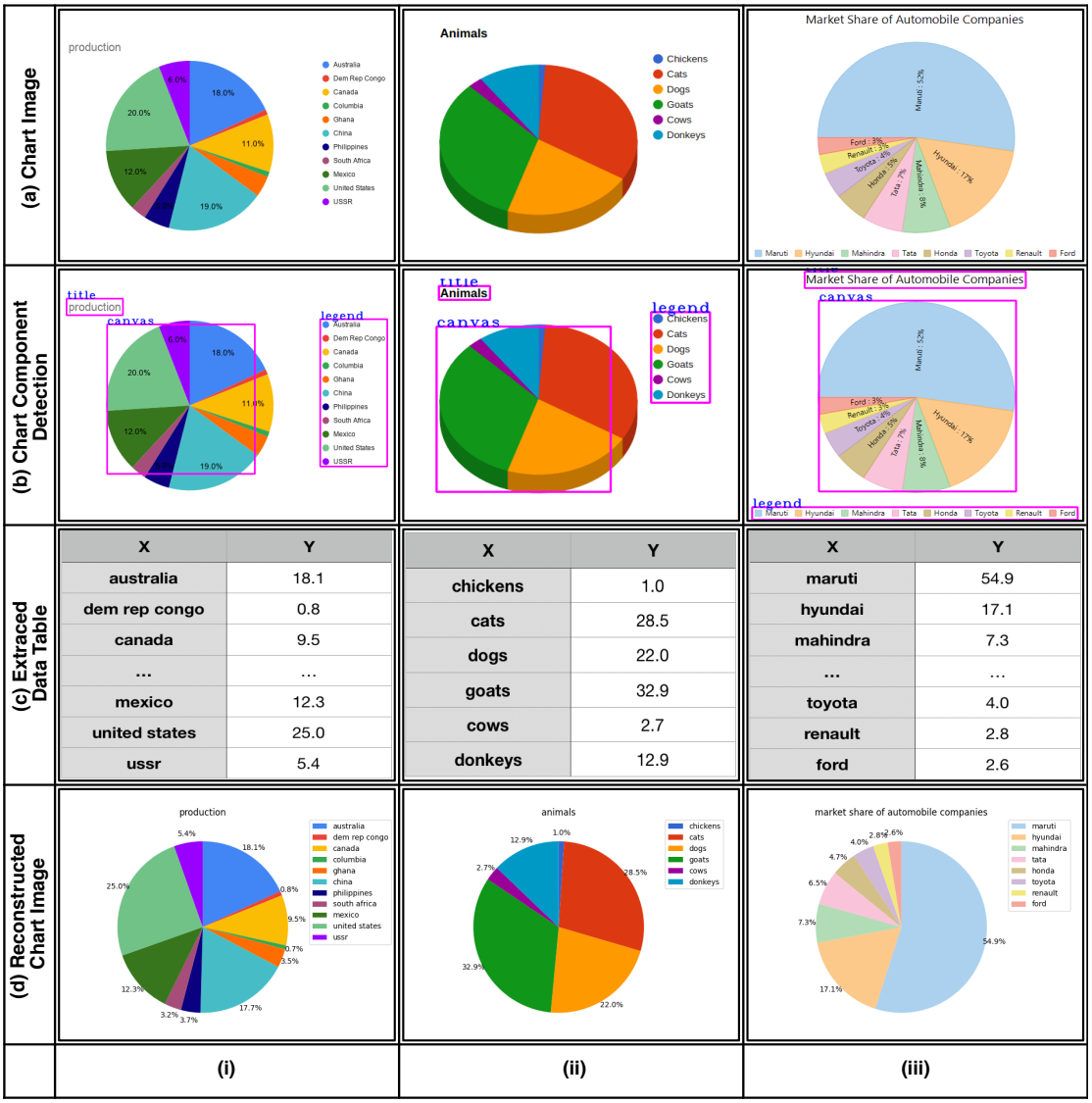
X	1960	...	1964
Peru	492.99	...	1200.93
Thailand	633.18	...	871.5
Malaysia	514.02	...	843.46
Jordan	563.08	...	619.16
Nigeria	240.65	...	542.06
Sudan	191.59	...	415.89
Ireland	261.68	...	359.81
Ghana	212.62	...	331.78
Libya	51.4	...	212.62
Ecuador	226.64	...	212.62
Sri Lanka	170.56	...	142.52

bin_center	freq
2.74	17.73
1.41	110.18
0.11	137.35
...	...
-3.86	15.47
-5.22	2.64
-6.54	2.64

Figure FC4.6: Different stages of bar chart reconstruction from the extracted data table of the chart images for sample source images of (i) simple bar, (ii) vertical grouped bar, (iii) horizontal stacked bar, and (iv) histogram.

due to improper clustering is almost considered as zero (Figure FC4.6(iii)(d), (iv)(d)). The second case is when heights of adjacent bins/bars have relatively small height differences, and the extracted data does not capture the differences (Figure FC4.6(iv)(d)). We may adopt optimal clustering mechanisms or sparsify the corner points obtained from tensor voting computing to handle the above-mentioned data extraction errors. We can even use effective corner detection techniques instead of the compute-intensive tensor field computation process for more efficient data extraction.

In pie charts, a test sample of 200 images from the FigureQA dataset [1] gives $\sim$90.11% accuracy for successfully reconstructing the plot without type-I and type-II



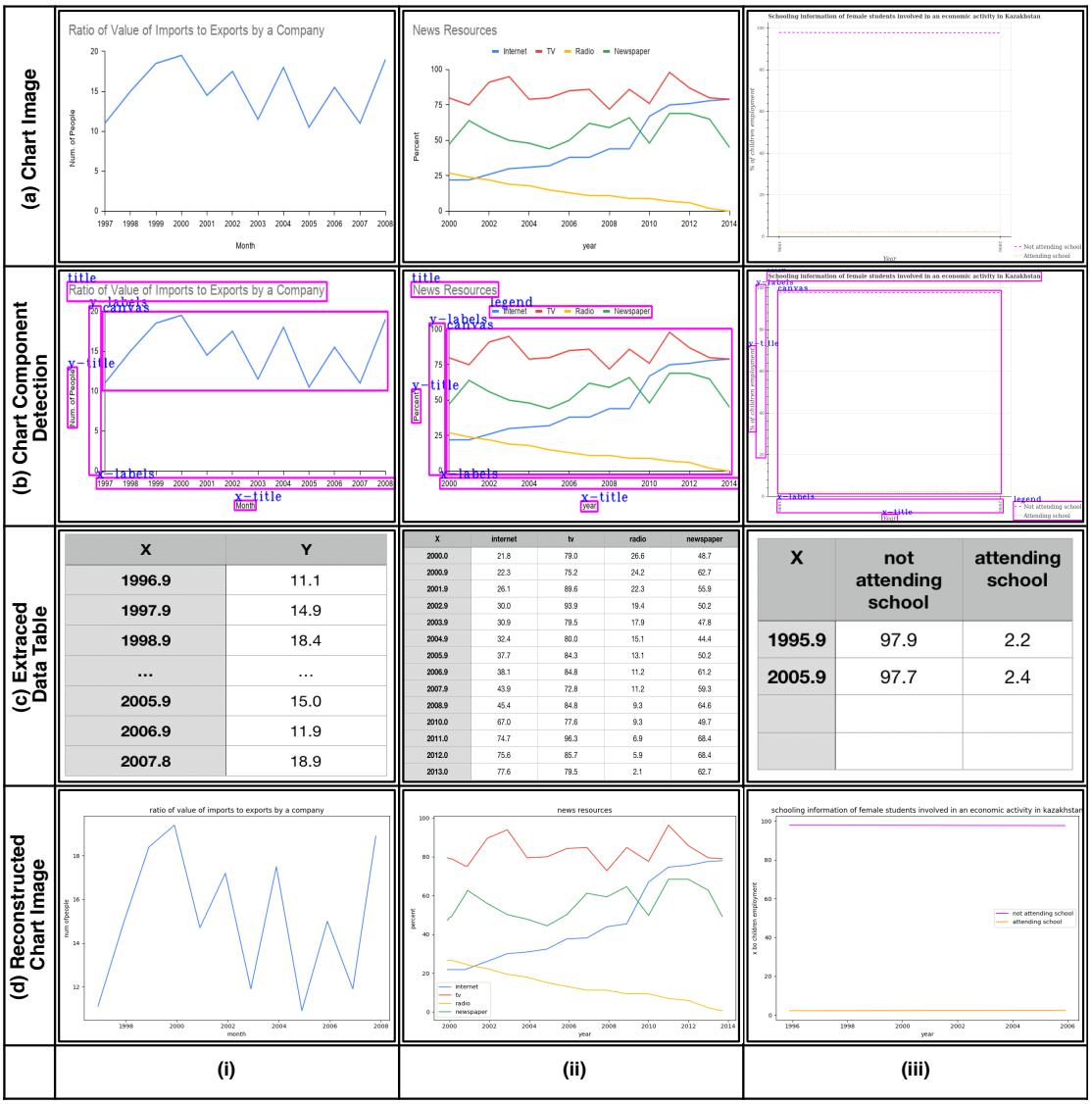
X	Y
australia	18.1
dem rep congo	0.8
canada	9.5
...	...
mexico	12.3
united states	25.0
ussr	5.4

X	Y
chickens	1.0
cats	28.5
dogs	22.0
goats	32.9
cows	2.7
donkeys	12.9

X	Y
maruti	54.9
hyundai	17.1
mahindra	7.3
...	...
toyota	4.0
renault	2.8
ford	2.6

Figure FC4.7: Different stages of pie chart reconstruction from the extracted data table of the chart images, for a sample of (i) synthetically generated images and (ii, iii) web collected images.

errors. Besides FigureQA test images having an appropriate plotting mechanism, the data extraction is accurate even in exceptional cases of pie charts with text embedding and visually appealing graphical objects (Figures FC4.7). Using more advanced circle detection methods like RANSAC [22] may further improve data extraction in pie charts.



X	Y
1996.9	11.1
1997.9	14.9
1998.9	18.4
...	...
2005.9	15.0
2006.9	11.9
2007.8	18.9

X	internet	tv	radio	newspaper
2000.0	21.8	79.0	26.6	48.7
2000.9	22.3	75.2	24.2	62.7
2001.9	26.1	89.6	22.3	55.9
2002.9	30.0	93.9	19.4	50.2
2003.9	30.9	79.5	17.9	47.8
2004.9	32.4	80.0	15.1	44.4
2005.9	37.7	84.3	13.1	50.2
2006.9	38.1	84.8	11.2	61.2
2007.9	43.9	72.8	11.2	59.3
2008.9	45.4	84.8	9.3	64.6
2010.0	67.0	77.6	9.3	49.7
2011.0	74.7	96.3	6.9	68.4
2012.0	75.6	85.7	5.9	68.4
2013.0	77.6	79.5	2.1	62.7

X	not attending school	attending school
1995.9	97.9	2.2
2005.9	97.7	2.4

Figure FC4.8: Different stages of line chart reconstruction from the extracted data table of the chart images for a sample of (i, ii) synthetically generated images and (iii) an image from PlotQA dataset [2].

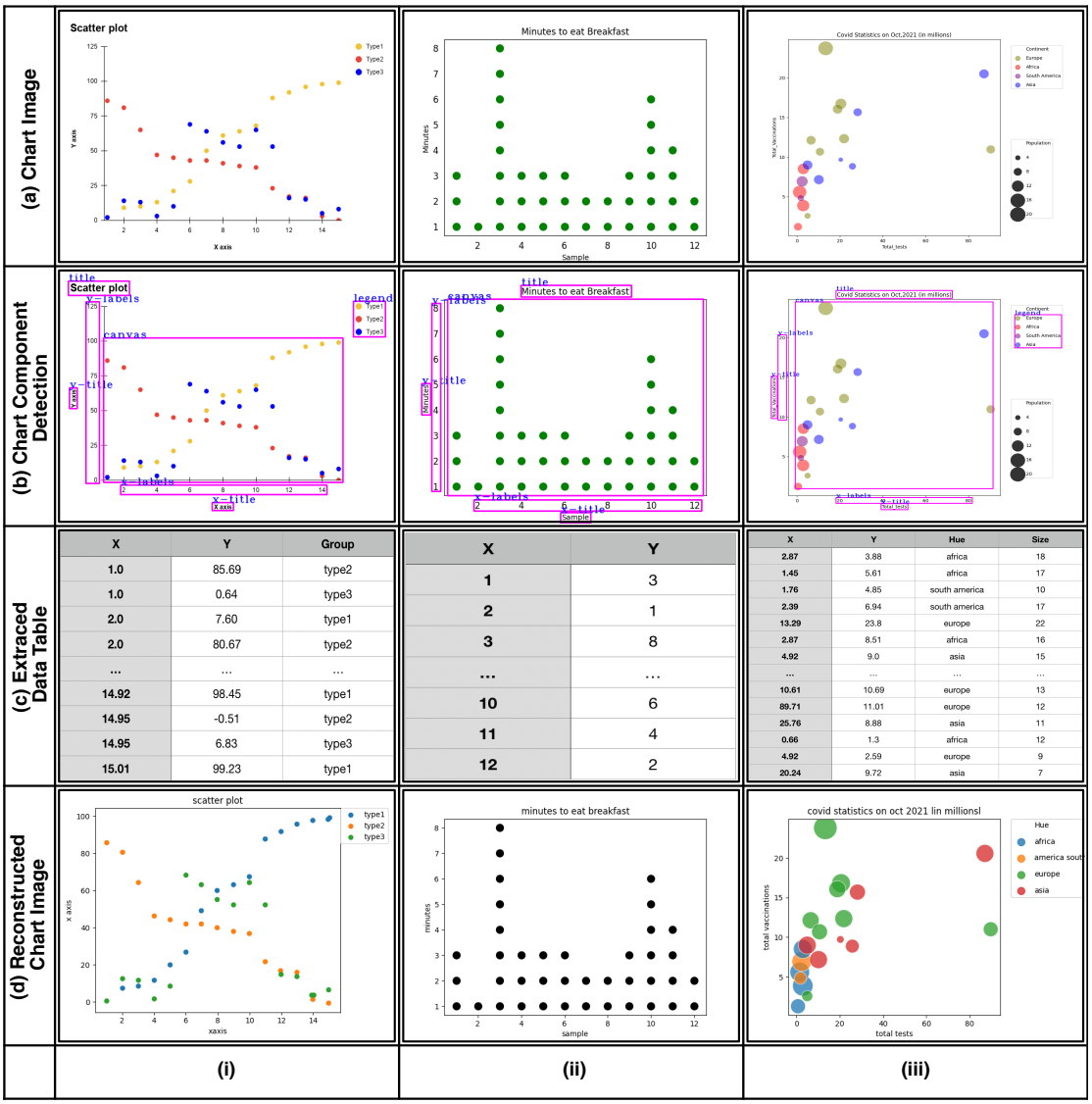
For line charts, for a test sample of 200 images from the PlotQA dataset [2], we observe individual graph line extraction based on the most frequent color pixels gives $\sim$92% success rate, which is far better than the FPN based object extraction methods with 37.65% success rate [2]. The neural network-based object extraction methods fail due to similar characteristics of graph lines and grid lines, i.e., thin dotted lines, horizon-

tal straight lines, and lower intensities of objects, as shown in Figure FC4.8(iii). However, our color-based extraction extracts graph lines accurately, resulting in successful chart data extraction and reconstruction with $\sim$74.56% accuracy. We also improve the data extraction technique over existing work [15] data value extraction interval based on the x-axis labels location, which overlooks the gradient change locations that are crucial in capturing the changes in the slope of the line. Capturing such locations is essential for improving the accuracy of the data extraction algorithm, so we propose a default interval that fits both gradient change and x-axis labels text locations to obtain better results (Figure FC4.8(ii)).

Our algorithm is successful in scatter point extraction for arbitrary shapes as well as cluttered scatter points, which has been a limitation in the tensor field-based data extraction approaches [8, 31]. We observe that the overlapping points in scatter plots are extracted accurately (Figure FC4.9(i)(iii)) despite the additional problem faced due to transparency. However, in bubble plots, the size of scatter points (Figure FC4.9(iii)) in the reconstructed plot is proportional to the original sizes but not the replica of the source chart image. This arises due to the conversion mechanism used by *matplotlib* library based on the DPI value of the image used to transform data values into the size of the bubble. Despite having the original dimensions of the size variable obtained, raw data values in pixels are proportionate to the original values.

4.4.2 Quantitative Assessment

We use the F1-Score and MAPE metrics to measure our workflow's success and failure rates (Table TC4.2). For quantitative assessment using F1-Score and MAPE metrics, we use synthetically generated chart images using *matplotlib* library for publically available data sources like *Kaggle* listed in Appendix A. We generated a set of 4 images for each variant of bar chart images and a set of 15 images for both line and pie



X	Y	Group
1.0	85.69	type2
1.0	0.64	type3
2.0	7.60	type1
2.0	80.67	type2
...	...	...
14.92	98.45	type1
14.95	-0.51	type2
14.95	6.83	type3
15.01	99.23	type1

X	Y
1	3
2	1
3	8
...	...
10	6
11	4
12	2

X	Y	Hue	Size
2.87	3.88	africa	18
1.45	5.61	africa	17
1.76	4.85	south america	10
2.39	6.94	south america	17
13.29	23.8	europe	22
2.87	8.51	africa	16
4.92	9.0	asia	15
...	...	...	...
10.61	10.69	europe	13
89.71	11.01	europe	12
25.76	8.88	asia	11
0.66	1.3	africa	12
4.92	2.59	europe	9
20.24	9.72	asia	7

Figure FC4.9: Different stages of scatter plot reconstruction from the extracted data table of the chart images, for sample source images of (i) simple scatter, (ii) dot, and (iii) bubble plots.

chart images. Histograms are not included in this analysis as the source, and extracted data in its case is a frequency table, different from a data table in the case of other sub-types of bar charts. For scatter plot variants, we generated 24 images for simple scatter and a set of 15 images for both bubble and dot plots. In total, we use a test set of 110 images of all chart types (excluding histogram) for quantitative evaluation of the source data table with the extracted data table.

The F1-Score value is usually the harmonic mean of precision and recall values used to rate the system's performance based on predicted and actual data. The F1-Score is a comprehension of precision and recall values, varying from 0 to 1; the greater the F1-Score, the lower the system's false positives and negatives, portraying the system's best performance. The computation of the precision and recall for extracted data table to its source data table is shown in [27]. The data extraction of the chart image is a success when F1-Score>0.8. The comparison of data table extraction accuracies of each chart with the state-of-the-art data extraction methods can be seen in Table TC4.1. The data table extraction in bar chart images is slightly lower than the state-of-the-art method despite the limitations in corner point extraction using tensor voting [8].

Our system outperforms the state-of-art techniques in the case of scatter plots and line charts. The extraction of dot plots is 100% accurate due to the simplicity of scatter points arrangements. Even in a bubble plot, despite the complex challenges due to transparency and overlapping points, it gives 100% for data table extraction with an F1-Score >0.8. As the variable concerning to size of the bubble is in pixel format, we exclude them in the F1-Score computation. However, we observe that the normalized radius values generated based on the conversion mechanism used by *matplotlib* library are closer to the original data values.

The data table extraction process involves transforming raw data, which may cause numerical precision errors predominantly. Hence, to compare the difference between the source and extracted data values, the MAPE value ranges between 0 and 1 for the synthetic images. For N data items with source data value x_i and its corresponding extracted value $x_i^{(e)}$, we compute

$$MAPE = \frac{1}{N}.\sum_{i=1}^{N}\left|\frac{x_i - x_i^{(e)}}{x}\right| \qquad \text{(Eqn 4.5)}$$

So, we define data extraction as success when the error rate MAPE<0.2. MAPE is

Table TC4.1: Comparison of F1-Score accuracy of data table extraction for each chart type with the state-of-the-art.

Chart Type	Method	Precision	Recall	F1-Score
Bar	ReVision [6]	78.3%	84.6%	81.3%
	ChartSense [7]	90.7%	92.1%	91.3%
	MECDG [41]	**91.2**%	**94.6**%	**92.9**%
	Our Method	91%	86%	91.7%
Scatter	ReVision [6]	79.1%	87.1%	82.9%
	ChartSense [7]	86.9%	90.4%	88.6%
	Scatteract [27]	88%	87%	89.2%
	MECDG [41]	90.5%	**95.1**%	92.7%
	Our Method	**97**%	95%	**96**%
Line	ReVision [6]	73.8%	79.8%	76.6%
	ChartSense [7]	78.2%	85.3%	81.5%
	MECDG [41]	88.7%	92.4%	90.5%
	Our Method	**95**%	**98**%	**100**%

augmented in the case of omission errors, owing to relatively short bars, especially in the case of stacked bars. We encounter higher MAPE values even in overlapping or closer graphical objects where the combined area is equally divided while retrieving their objects' locations, causing a slight deviation from the original data value. Apart from overlapping graphical objects, the data is skewed because of its transformation from pixels values to significantly larger numbers ($\approx 10^7$), causing the precision error. However, in pie charts, we record the proportion values of data items (normalized form) rather than their original data counts, resulting in error-free data extraction.

Besides F1-Score and MAPE values, we additionally compute the differences between the Pearson correlation coefficient values of the original data table to the extracted data table in simple scatter plots. Our algorithm results in 84% scatter plots with <0.1 difference in the correlation coefficient. That states the spread and direction of data are still preserved despite the error rate, which is considered as a more significant feature for text summarization than exact data values.

Table TC4.2: Data table extraction results for different chart types using our proposed workflow.

Chart Type	Average Precision	Average Recall	Success Rate	
			F_1 score >0.8	MAPE<0.2
Simple Bar	0.87	0.83	87.5%	87.5%
Grouped Bar	1.0	0.99	100%	100%
Stacked Bar	0.86	0.76	87.5%	50%
All Bar Charts	0.91	0.86	91.7%	76%
Pie Charts	0.94	1	93%	100%
Simple Scatter	0.97	0.95	96%	84%
Bubble Plot	1.0	0.99	100%	93%
Dot Plot	1.0	1.0	100%	100%
All Scatter Plots	0.98	0.97	98%	91%
Line Charts	0.95	0.98	100%	80%
Overall	**0.96**	**0.95**	**96.4%**	**88.7%**

4.5 Summary

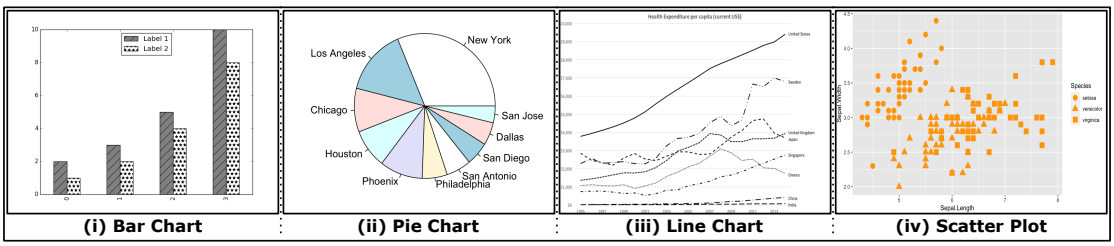


Figure FC4.10: Images of multi-class charts using visual channels other than color to distinguish the classes known to not work with our data extraction workflow.

Despite a few shortcomings in the chart component extraction step of the workflow, our data extraction method improves over the state-of-the-art color-based extraction techniques in chart images. We came across a few limitations in the data extraction of bars using the tensor voting due to improper DBSCAN clustering mechanism in exceptional cases where the graphical objects are closer and needed to be handled by choosing optimal clustering techniques or sparsifying corner points. The color-based mechanism used scatter points, and line charts handled the overlapping scatter points, and extracted the gradient change location of graph lines efficiently. Also, the color-based approach in simple scatter plots outperforms the state-of-the-art ScatterPlotAnalyzer [32] system

that uses tensor voting for data extraction. Our color based approach, on the other hand, fails because it relies on legend mapping of graphical objects based on their color value in cases where color information is insufficient to represent the graph objects, e.g., multi-class charts using shape as a visual channel with the same color (Figure FC4.10). For such cases, it is imperative that we focus on texture information along with color information of graphical objects for efficient data extraction. In the future scope of our work, our proposed methods pave the way for a feature-based data extraction approach that takes into account all graphical object properties, such as color, texture, etc.,

CHAPTER 5

CHART DATA TABLE SUMMARIZATION

An enormous amount of data is generated daily across the world. The pictorial representation of the processed data in charts allows analysts to visualize new concepts and patterns and derive valuable insights. However, in many scenarios, the natural language text is preferred to chart visualizations that can quickly tell a rich and complex story better. These crisp textual descriptions juxtaposed with chart visualizations or source data tables will create more magic in the knowledge exploration process.

There is a need for us to create an effective summary that concisely represents the significant information to keep busy readers informed without demanding more time than necessary to get the information they need and by allowing non-visual readers to get an idea of the chart image. The information perceived as significant by a reader varies based on various aspects, such as education, work, and reasoning. Given our requirement of creating a generic summary without serving any specific intent and sparsity of dataset for training the natural language model, the traditional template-based NLG approach suits our requirement better than advanced NLG approaches that involve deep learning solutions [41]. As we have chosen to use templatized NLG approach to generate the system summary, we now need to ensure that, in addition to the accurate description of the extracted data table, the summary should contextually represent the visual perception of the chart image. To achieve this goal, we design our

system-summary generation process motivated by the iterative waterfall model in the software development cycle into the following five stages:

- Exploration and discovery of significant features of charts
- Design of sentence structure and initial summary generation
- User study for readability of initial summary
- Analysis of the user study
- Redesign of sentence structure and final summary generation

To arrive at the predefined template for the chart summary, we employ the 9-stage design study methodology [45]. Although the design study technique focuses on visualization as human-computer interaction (HCI) output, it can also be extended to design chart summaries similar to visualizations as chart summaries are computer-generated but used by humans. The nine stages are organized into three top-level phases: preconditioning, core, and analysis, with the nine stages being split across these phases. The precondition phase, which focuses on the design's preparation, includes *learning*, *winnow*, and *cast* stages in sequence. We refer to the preconditioning phase as the determination of the problem statement pertaining to automatically interpreting chart images which involve the design of chart classification, chart component extraction, and data table extraction stages of our workflow. The core phase, central to the actual design process, includes *discover*, *design*, *implement*, and *deploy* stages in sequence. The core phase refers to the first three stages of our summary generation process, which entails gathering requirements from experts and users, designing and implementing sentence structure to generate a summary that satisfies the requirements, and conducting a user study to gather feedback. The analysis phase, which is done retrospectively, includes *reflect* and *write* stages in sequence, which refers to the last two stages of our summary generation process analysis of user study, followed by refining and reflecting the new

sentence structure design. Finally, our detailed documentation serves as the written report of the entire design study.

5.1 Exploration and Discovery of Significant Features of Charts

In the initial stage, we explore and discover the significant features that chart images intend to exhibit in summaries based on the source data formats and encoding styles used in each chart type. For instance, pie charts are used to picture the whole-part relationship among data items, highlighting the user's focus on the proportions of data items (Figure FC5.11(i)). Similarly, a line chart allows the users to observe the rise-fall patterns in series data, with changing trendlines as significant information. Likewise, a quick glimpse of the scatter plot discloses the spread of data points by signifying interdependence among data variables. The visual perception of data in a chart varies furthermore on its subtype in bar charts. We focus on features like frequency distribution and mode for the histogram type chart. When it comes to other categories of bar charts (Figure FC5.8(ii), FC5.9(ii)), we check for ordered attributes like age, date, day, month, and year in the chart image and focus on the trend patterns of the chart across the timeline. We also focus on other relation features like the attributes range, correlations between categorical attributes, standard deviations, and mean values of chart data to encounter any exceptional cases.

5.2 Design of Sentence Structure and Initial Summary Generation

The next stage is to design a sentence structure to embed the data values that entail these significant features. The sentence structure design constitutes a flow chart in Figure FC5.1. The flow chart inputs the extracted data table besides chart type and then traverses through the template that gives a primary description of what the chart

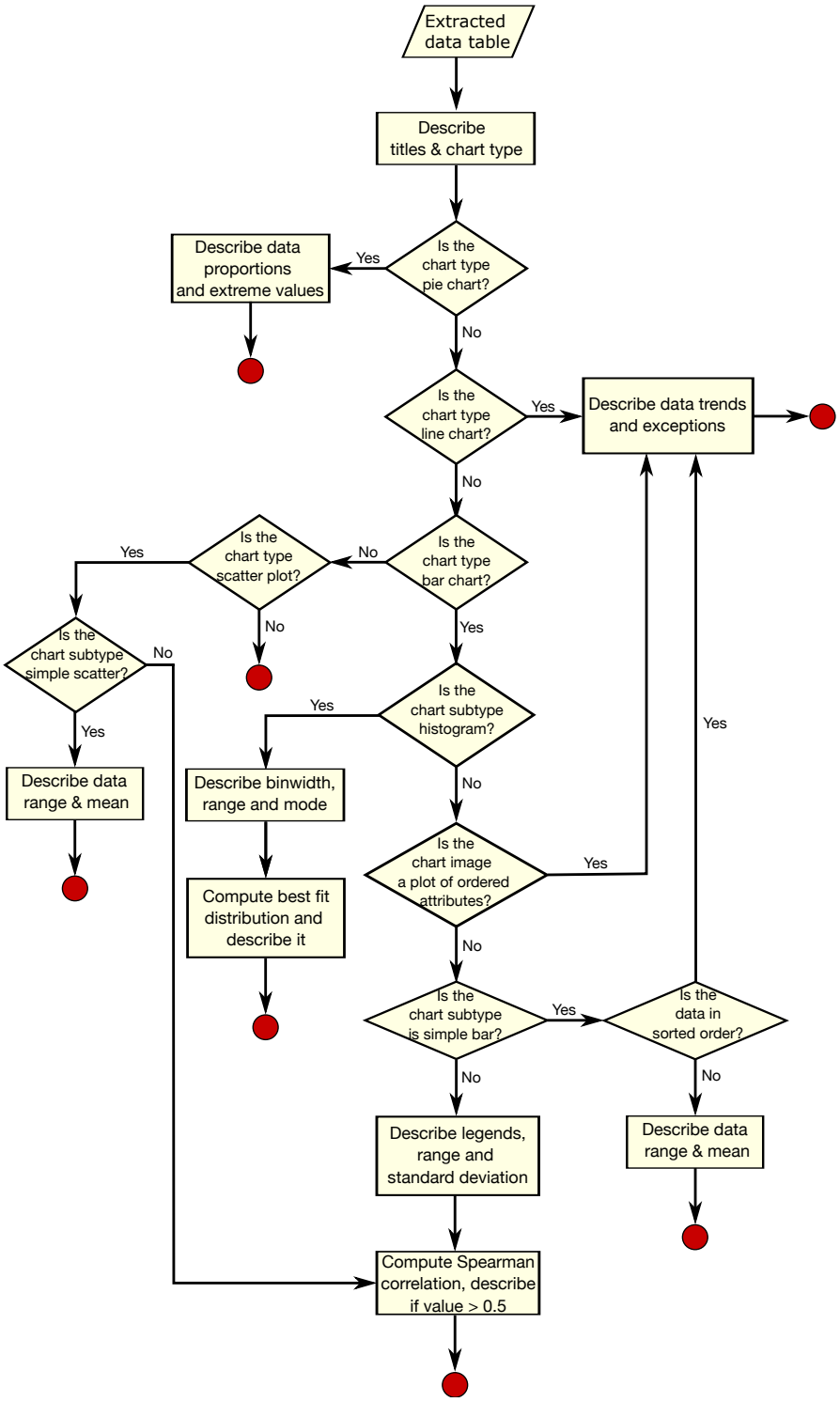


Figure FC5.1: Our proposed sentence structure formation flow chart to generate an initial chart summary.

image illustrates obtained from chart components such as chart titles, axis titles, and other text labels. Later on, we traverse across a series of decision-making boxes based on chart type and extracted data formats. The order of decisions made will lead to templates entailing their respective significant features. Then, the templated-NLG in the sentence structure flow chart is used for generating the initial summary, referred to as the system-generated summary.

The initial set of system summaries needs to be evaluated to assess the usefulness of the summaries generated from the designed sentence structure and to assure that proposed significant features based on each chart type are sufficient to meet users' perception of the chart images. In such a scenario, human ratings, the popular NLG evaluation technique, is the best approach to assess the quality and usefulness of our system-generated summary. Therefore we conduct a user study that allows the users to rate and provide feedback on the system-generated summary to assess the effectiveness and further improve the initial system-generated summaries in case of an irrelevant proposition on significant features.

5.3 User Study for Readability of Initial Summary

We designed a web-based survey application hosted online using *pythonanywhere* web hosting service (Figure FC5.2). The web-based survey is required owing to the social distancing during the pandemic. Nevertheless, a web-based survey helps us collect diverse responses from various sections of the population, thereby making generalized summaries satisfying all categories of users. The survey is designed into two modules, each owing to the main objectives of our user study. One is to learn about the user's understanding and interpretation of chart images, and the other is to assess the completeness and effectiveness of the initial system-generated summary and help us further improve it. The second module designed is inspired by a user study for identifying the

intended message of a simple bar chart [9] as our objective is partially similar to the purpose of the study.

The survey participants are invited to take the survey through email, social media, etc., and are informed about the purpose of the user study. The responses to the survey are recorded in data tables, along with the unique user id created while collecting and validating the participant response. Each participant is made to go through the following sections in sequence for a single survey session.

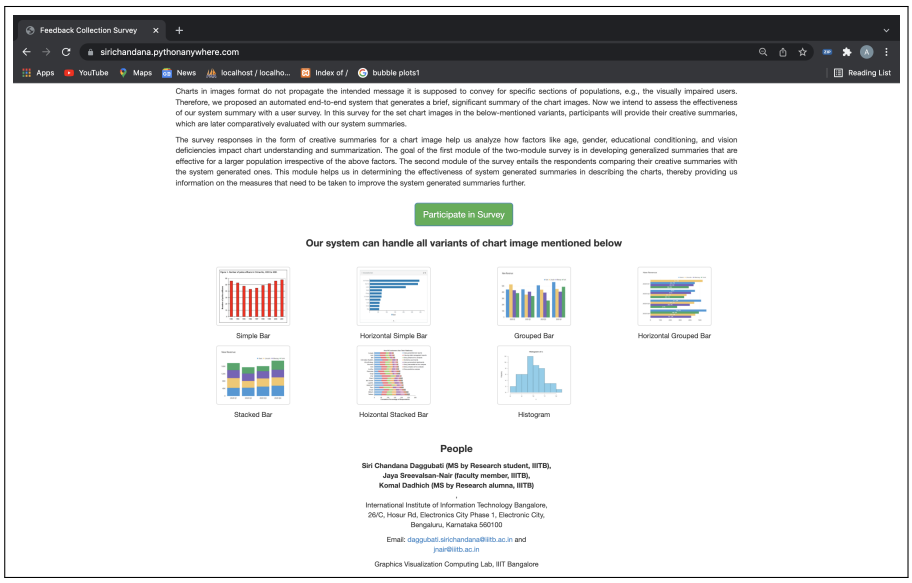


Figure FC5.2: The landing page of the web-based survey form designed to assess the readability of initial system-generated summaries.

5.3.1 Participant Profile

We collect the details of participants prior to starting the survey (Figure FC5.3) and use these details to eliminate duplication in feedback data collection. The participant profile has details, such as his/her name, email ID, age, gender, educational degree, profession (if worked/working as an educator), vision deficiency, and self-assessment of familiarity with usage of charts. The details on age, gender, and educational con-

ditioning help us assess their impact on chart understanding and interpretation of participants. The vision information will aid us in assessing if there are any difficulties in color perception (e.g., color blindness, etc.) that caused issues with chart interpretation while attempting the survey. All these details will allow us to conduct a fair analysis of users' chart interpretation when influenced by such factors.

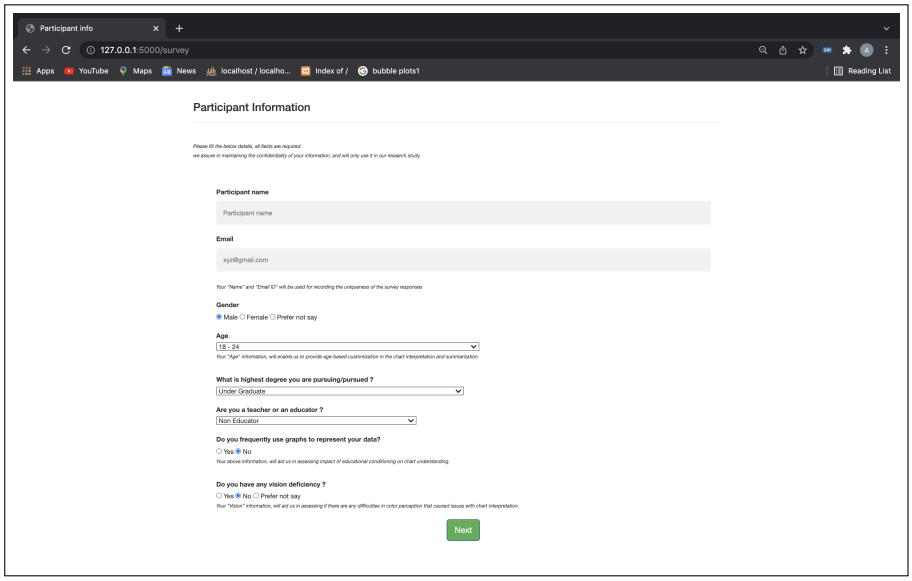


Figure FC5.3: The web page for collecting details of participant profiles to perform a de-duplicated analysis of their responses in the user study.

5.3.2 Instructions for Participants

We provide the participants with a detailed set of instructions (Figure FC5.4), including the expected time commitment for the survey. The instruction list help participants to make their tasks much more manageable and get more effective responses by answering their queries and clearly stating the tasks they are expected to complete. In addition to these instructions, further queries have been answered through telephonic calls, emails, social media correspondences, etc.

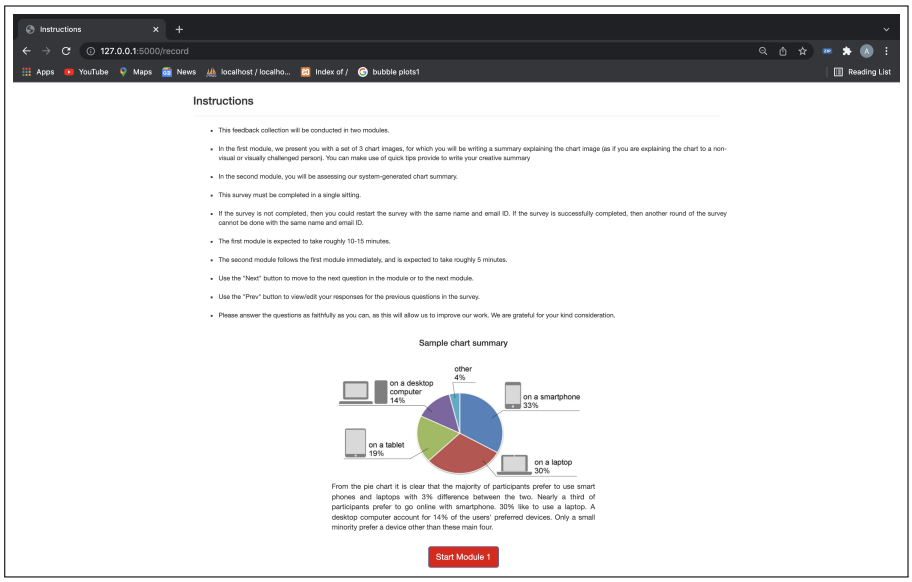


Figure FC5.4: The web page displaying the set instructions and sample chart summary to guide participants in taking the survey.

5.3.3 Module-1 of the Survey

We now move on to the first module, the core part of the survey designed upon the first objective of the user study. Here, each participant is shown a set of three chart images. These images are randomly selected from 26 images from the test corpus. We ensure that the set presented to each participant includes a grouped bar chart, a stacked bar chart, and a simple bar/histogram. The participants see an image and complete the assigned task before being shown the following image. The task for each image entails writing one's own *creative* chart summary for the image. The participant is advised to prepare the summary in a way he/she would narrate chart image information to a visually impaired person.

Due to the variable time complexity of creative writing in participants, we limit our test set size to 3 chart images and test corpus to bar charts as it covers larger variants and cases in the flowchart. However, the average time to complete the survey still takes 45 minutes longer, with module-1 consuming three-fourths of the time. To further encour-

age people to participate in the survey despite their busy lives, we provide small text suggestions in the form of quick tips near the creative summary text boxes in module-1. The web page sample of module-1 for a simple horizontal bar chart is displayed in Figure FC5.5

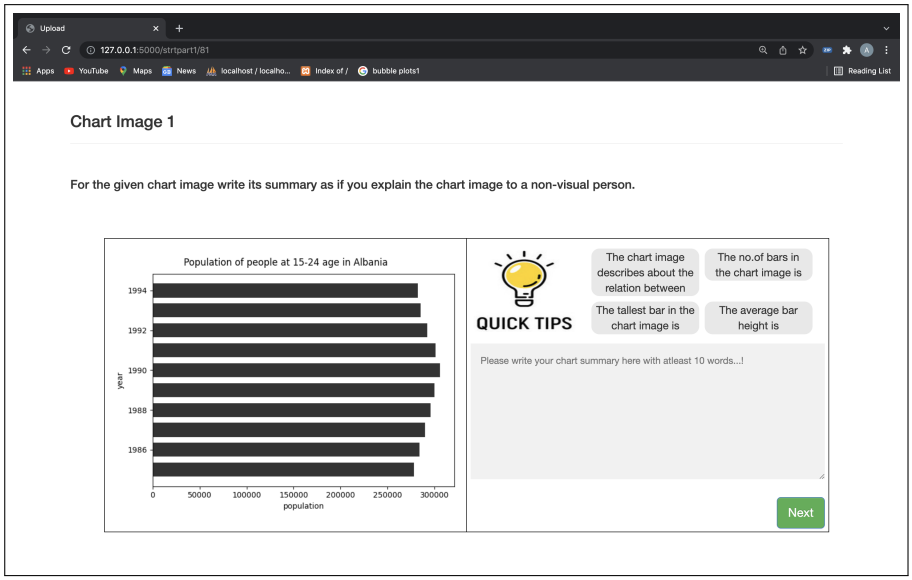


Figure FC5.5: The web page displaying one of the sample chart images in the Module-1 of the survey, asking participants to write their creative summaries for given image.

5.3.4 Module-2 of the Survey

The second module was designed based on another objective of the study by assessing our summaries using human ratings. We present the user with the same set of chart images from the first module, their creative summaries by the participant prepared in Module-1, along with corresponding initial system-generated summaries from the sentence structure flowchart. These three entities are juxtaposed for the participant to compare the two summaries. The task entails comparing both the summaries and assessing using Likert-like scales. We use three components of the scores, *i.e.,* compare-, sufficiency-, and grammar scores, to rate our system-generated summaries qualitatively against their creative summaries. This comparison task is done for one image at a time,

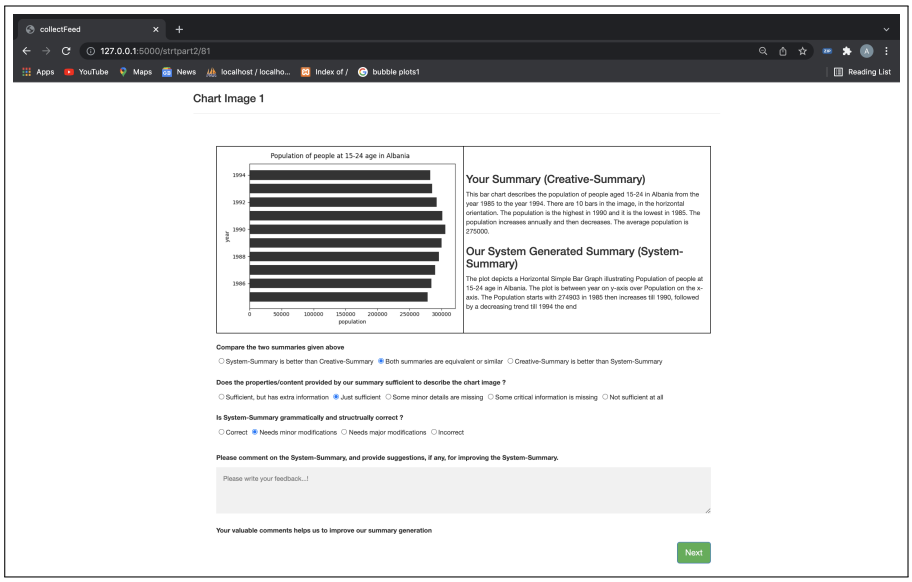


Figure FC5.6: The web page displaying one of the sample chart images in Module-2 of the survey, asking participants to assess system-generated summaries in comparison to their creative summaries written from Module-1 of the survey.

as done in Module-1. The three comparative assessment questions used in our study are given below, with rating range in parentheses and interpretation of each rating.

Compare-Score (1-3) :

3. System-Summary is better than Creative-Summary
2. Both summaries are equivalent or similar
1. Creative-Summary is better than System-Summary

Sufficiency-Score (1-5) :

5. System-Summary is sufficient but has extra information
4. System-Summary is just sufficient
3. Some minor details are missing in System-Summary
2. Some critical information is missing in System-Summary

1. System-Summary is not sufficient at all

Grammar-Score (1-4) :

4. System-Summary is grammatically/structurally correct
3. System-Summary needs minor modifications
2. System-Summary needs major modifications
1. System-Summary is grammatically/structurally incorrect

Detailed Feedback

Apart from comparison scores, if the participant intends to provide short feedback with suggestions for improving the initial system-generated summary as a final step of the user survey. we provide them with a small feedback box at the end of a web page in module-2 while assessing each chart image. The web page sample of module-2 for a simple horizontal bar chart is displayed in Figure FC5.6.

5.4 Analysis of the User Study

The end goal of our user study is to know users' visual perception and find out the information of the chart image that users feel is significant enough to be explained in summary. So, there is a need for us to analyze the user responses on both their creative summaries and initial system summaries generated to improve initial system summaries further and generate more generalized summaries that fulfill the users' requirements.

Generally, feedback analysis involves the following tasks: (i) data cleaning and aggregation into a single database table, (ii) selecting and analyzing data values based on categories and subcategories, and (iii) finally, using automated tools to explore interesting findings and their root causes. As the first task of the procedure mentioned

above, once the participants complete both the modules of the web survey, the system records the individual participants' responses with unique record names assigned while collecting their personal information. We avoid incomplete data values by restricting participants' word limit in creative summary entry and default options in data comparison scores. Then all the individual data tables are joined into a single table with the help of their unique ids. In total, *thirty* participants, primarily undergraduate and graduate college students in the age group 18-45, have participated in our user study owing to the longer duration of the survey on average. Due to the smaller data table size and open-ended answers for creative summaries, we manually explore and analyze user response data for each bar chart subtype instead of using automated statistical tools for the third task in the analysis process. The following observations are made from the user responses:

1. In most cases, the participants were satisfied with the system-generated summary.
2. They stated that the non-visual statistical description (Figure FC5.9(i)), such as distribution of histogram, standard and correlations values in the stacked and grouped bar, etc., improved their understanding of the data of the chart image. However, a few participants contradicted that the statistical description in the system-generated summary was extraneous. They also pointed out that the system-generated summaries had not included a few visually perceivable details such as intra-class inferences, overall bar heights, and inter-class differences in heights in grouped and stacked bars.
3. Some of the participants misinterpreted the histogram chart as a simple bar chart without considering the semantics of the bar being counts or frequencies and that the chart is a visualization of the data distribution.
4. Apart from the suggestions on improving the content of the summary, there were suggestions on editorial corrections to the summaries, e.g., proper noun usage,

capitalization of words, etc. These summaries also had instances of a few words being misspelled due to improper text recognition (Section 3.1).

5. Some participants suggested that summaries with detailed data representation are desirable for chart images with fewer bars, say <7.

6. There was another feedback on the use of *compound subjects* in sentences to shorten the summary, e.g., "A is correlated to B, B is correlated to C, and C is correlated to A" is to be rephrased as "A, B, and C are correlated to each other."

7. We quantify the survey outcomes, and the average grammar score is 3.33 out of 4.0, which states minor grammatical modifications are required in initial summaries. The average compare-score is 2.33 out of 3.0, which states that our initial system-generated summaries cover similar details as the creative summaries by the participants. The participants reported that the system-generated summaries outperformed creative summaries, especially in charts without any context, such as missing chart titles or axes titles. The average sufficiency score is 3.97 out of 5.0, which states that system-generated summaries are sufficient.

5.5 Redesign of Sentence Structure and Final Summary Generation

The outcomes of the user response analysis led to the redesign of sentence structure and final summary generation based on the new sentence structure. This is an iterative stage for the second step in the summary generation process. Considering the second observation of user responses along with satistical description, we intend to provide visually perceivable details in the final summary serving all categories of users. The text organization in the final summary has three parts in sequence [44] – the chart specifications, the visual summary, the data table summary*, i.e.,* the statistical description of the

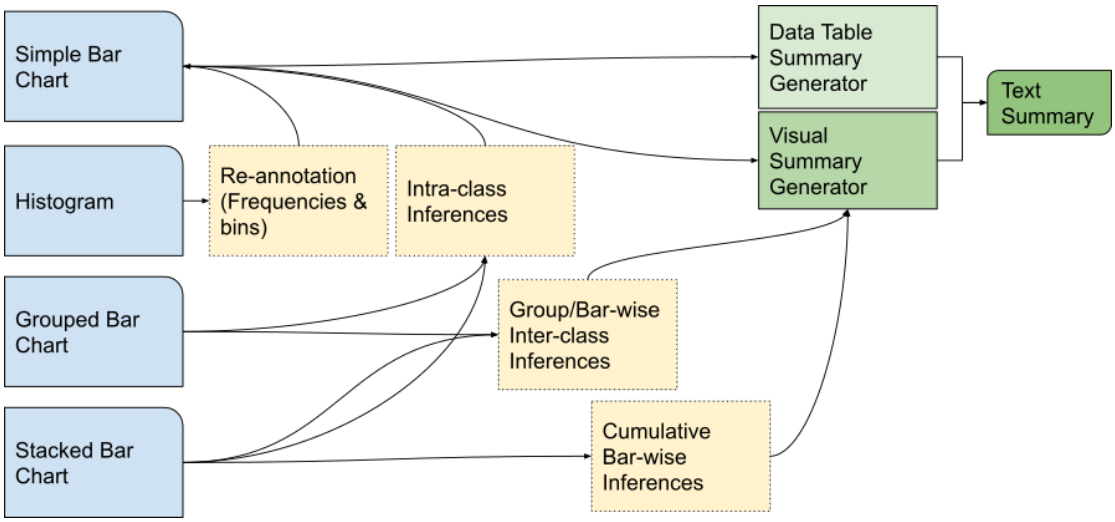


Figure FC5.7: Our proposed algorithm for generating the text summary in the Bar Charts, refined by the responses from the user study.

initial system summary.

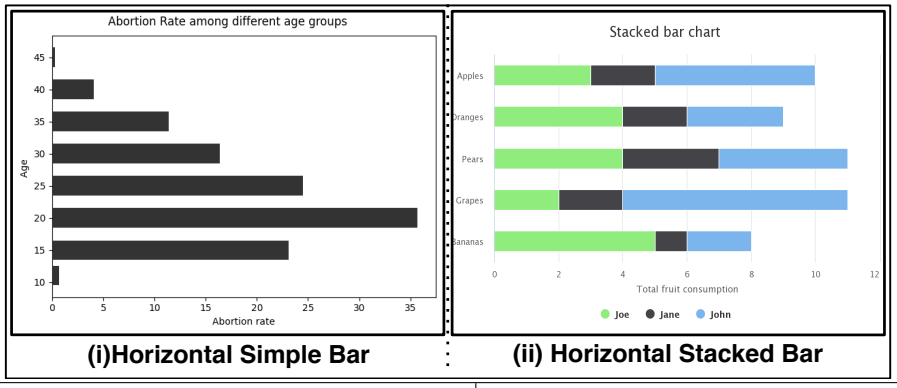
We implement the sentence structure generation using modular design for a visual summary of bar chart images, as shown in Figure FC5.7. In our design, the most reusable component simple bar module, either speak about trends for a given list of data values (heights and frequencies) or reads out all the values in the case of a smaller dataset (< 7). In histograms, as the bar heights signify frequencies and visually perceive the exact details as bars, we re-annotate bar heights as frequencies and then invoke the simple bar module for a visual summary generation. In the grouped and stacked bar charts, we compute bar heights differences between categorical variables and then invoke the simple bar module for each pair to represent interclass inferences. We also compute the sum of all stacks in a bar to discuss cumulative bar heights in the stacked bar. We invoke the simple bar module for each category for intraclass inferences in grouped and stacked bar charts. We now merge data table summaries from statistical descriptions of initial summaries with visual summaries and deduplicate them to generate final summaries. Apart from including visual descriptions, we also address the issues of compound subjects in sentences and grammatical errors. We make use of an open-

source Python library called *GingerIt* [54], a wrapper class of the *gingersoftware.com* API to correct spelling and grammar mistakes in context of complete sentences.

Based on our observations of the summary of bar charts from our user study, we extend the same process for other chart image types such as scatter and line charts, especially in the cases where they exhibit similar characteristics as in bar charts. We added inter-class visual inferences for line charts (Figure FC5.10) and modified the sentence structure to make it look more compact and informative. We generate an initial summary of dot and bubble plots of the newly added variants of scatter plots (Figure FC5.11(iii),(iv)). As dot plots exhibit similar properties as a histogram, we use the same modular components to generate their summaries. In bubble plots, we discuss intra-class and inter-class inferences similar to simple scatter plots and exclude inferences on the size variations of the plot as we only have raw data values given as radius in pixels and not the value in data space. Finally, we use the initial summaries of pie charts that are left unchanged as they already provide complete data descriptions, which are also readable.

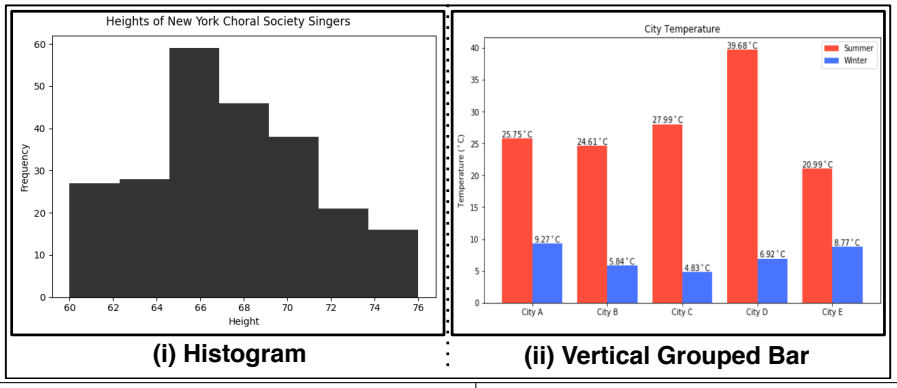
Thus, our chart summarization provides a complete summary, which we refer to as the *master* summary. The final set of master summaries is added with more details based on the extracted features, making them longer than the initial summaries. The master summaries capture most of the significant details to have a generic set of summaries that piques the interest of larger groups of the population. We need to focus on developing a selective pruning mechanism of master summaries based on user inputs, thus customizing this component for intended purposes. The parts of the modified text organization can also be considered as the granularity of information, which can be used in an interactive framework for providing high-to-low level details of the graphic, e.g., interactive SIGHT (Summarizing Information GrapHics Textually) [55].

Figure FC5.8, FC5.9 shows the sample results of our workflow for text summary



	Initial Summary	Master Summary
(i)	The plot depicts a Horizontal Simple Bar Graph illustrating Abortion Rate among different age groups. The plot is between Age on y-axis over Abortion rate on the x-axis. The Abortion rate starts with 22 in 14 then increases till 19, followed by a decreasing trend till 39 the end.	The plot depicts a Horizontal Simple Bar Graph illustrating Abortion Rate among different age groups. The plot is between Age on y-axis over Abortion rate on the x-axis. **In the Age ranging form 15-40 at the interval 5, the Abortion rate are 22.8, 35.4, 24.2, 16.07, 11.07, and 3.73 respectively**. The overall mean and standard deviation values of Abortion rate are 18.88 and 10.12 respectively.
(ii)	The plot depicts a Horizontal Stacked Bar Graph illustrating Stacked bar chart. The plot is having Total fruit consumption on x-axis for joe, jane, and john. The list of 'Y-axis' values is Bananas, Grapes, Pears, Oranges, and Apples. The 'joe' range from 1.94 to 5.21, with a standard deviation of 1.11. The 'jane' range from 0.79 to 2.98, with a standard deviation of 0.69. The 'john' range from 1.92 to 7.39, with a standard deviation of 1.88. The categories 'joe' and 'john' are negatively correlated by -0.97 Spearman rank correlation	The plot depicts a Horizontal Stacked Bar Graph illustrating Stacked bar chart. The plot is having Total fruit consumption on x-axis for joe, Jane, and John. **The Total fruit consumption of joe are 5.21, 1.94, 4.11, 4.11, and 3.02 for the labels Bananas, Grapes, Pears, Oranges, and Apples respectively. The Total fruit consumption of Jane are 0.79, 1.87, 2.98, 1.9, and 1.9 for the labels Bananas, Grapes, Pears, Oranges, and Apples respectively. The Total fruit consumption of junk are 1.92, 7.39, 4.12, 3.01, and 5.18 for the labels Bananas, Grapes, Pears, Oranges, and Apples respectively. The Total fruit consumption of all categories cumulatively are 7.93, 11.21, 11.21, 9.01, and 10.1 for the labels Bananas, Grapes, Pears, Oranges, and Apples respectively.** The standard deviation values of Total fruit consumption for categories 'joe', Jane', and 'John' are 1.11, 0.69, and 1.88 respectively. The categories 'joe', and 'John' are negatively correlated with one another.

Figure FC5.8: Comparison of initial and master summaries generated in our design study for (i) simple bar and (ii) stacked bar charts. The boldface formatted excerpts in row E2 are added during the redesigning phase of the summary generation process.



(i) Histogram **(ii) Vertical Grouped Bar**

	Intial Summary	Master Summary
(i)	The plot depicts a Histogram of Heights of New York Choral Society Singers. The bins range from 61.0 to 75.0 with 2.33 bin width. The mode of a histogram is 66.0 with a frequency of 60. The frequency distribution of histogram is the norm with following parameters loc=0.00, scale=1.00.	The plot depicts a Histogram illustrating the frequency of 'Heights of New York Choral Society Singers'. **The frequency starts with 27 in Height 61 then increases till 66 the maximum value, and ends with a decreasing trend till 75 the minimum values, frequency bins of the histogram range from 61.0 to 75.0 with 2.33 bin width**. The mode of a histogram is 66.0 with a frequency of 60. The frequency distribution the histogram is the norm with following parameters loc=0.00, scale=1.00.
(ii)	The plot depicts a Vertical Grouped Bar Graph illustrating City Temperature. The plot is having Temperature (*C) on y-axis for summer, and winter. The list of 'X-axis' values is City A, City B, City C, City D, and City E. The 'summer' range from 20.92 to 39.44, with a standard deviation of 6.31. The 'winter' range from 4.84 to 9.25, with a standard deviation of 1.68.	The plot depicts a Vertical Grouped Bar Graph illustrating City Temperature. The plot is having Temperature (*C) on y-axis for summer, and winter. **The Temperature (*C) of summer are 25.62, 24.44, 27.78, 39.44, and 20.92 for the labels City A, City B, City C, City D, and City E respectively. The Temperature (*C) of winter are 9.25, 5.82, 4.84, 6.9, and 8.76 for the labels City A, City B, City C, City D, and City E respectively. The Temperature (*C) difference between summer and winter starts with 16 at City at then increases till City D the maximum value, and finally ends with 12 in City E the minimum value**. The standard deviation values of Temperature (*C) for categories 'summer', and 'winter' are 6.31, and 1.68 respectively.

Figure FC5.9: Comparison of initial and master summaries generated in our design study for (i) histogram and (ii) grouped bar charts. The boldface formatted excerpts in row E2 are added during the redesigning phase of the summary generation process.

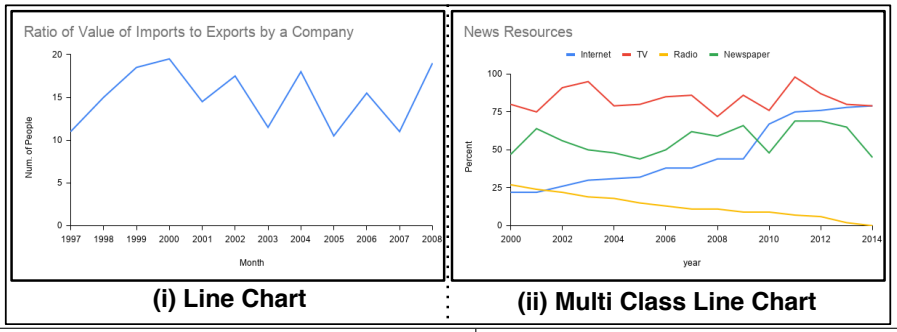


	Chart Image	Master Summary
(iii)	The plot depicts a Simple Line Graph illustrating ratio of value of imports to exports by a company. The plot is between num ofpeople on y-axis over month on the x-axis. The num ofpeople starts with 11 in 1997 then increases till 2000, followed by a decreasing trend till 2001, an increasing trend till 2002, a decreasing trend till 2003, an increasing trend till 2004, a decreasing trend till 2005, an increasing trend till 2006, a decreasing trend till 2007, and finally ends with 18 in 2008.	The plot depicts a Simple Line Graph illustrating ratio of the value of imports to exports by a company. The plot is between NUM of people on y-axis over a month on the x-axis. **The name of people of has its maximum and minimum values 19 and 10 at 2000, and 2005 respectively. The overall mean and standard deviation values of the names of people are 15.15 and 3.0 respectively**.
(ii)	The plot depicts a Line Graph illustrating news resources. The plot is between percent on y-axis over year on the x-axis for internet, tv, radio, and newspaper. The percent of internet has an overall increasing trend from 2000 to 2013. The percent of tv starts with 79 in 2000 then declines till 2001, followed by an increasing trend till 2003, a decreasing trend till 2004, an increasing trend till 2007, a decreasing trend till 2008, an increasing trend till 2009, a decreasing trend till 2010, an increasing trend till 2011, a decreasing trend till 2013, the end. The percent of radio has an overall decreasing trend from 2000 to 2013. The percent of newspaper starts with 48 in 2000 then increases till 2001, followed by a decreasing trend till 2005, an increasing trend till 2007, a decreasing trend till 2008, an increasing trend till 2009, a decreasing trend till 2010, a decreasing trend till 2013, the end.	The plot depicts a Line Graph illustrating news resource. The plot is between percent on y-axis over a year on the x-axis for internet, tv, radio, and newspaper. The percent of the internet has an overall increasing trend from 2000 to 2013. **The percent of tv has its maximum and minimum values 96 and 72 at 2011, and 2008 respectively. The percent of radio has an overall decreasing trend from 2000 to 2013. The percent of newspaper has its maximum and minimum values 68 and 44 at 2011, and 2005 respectively. The percent difference between internet and radio has an overall increasing trend from 2000 to 2013. The standard deviation values of percent of categories 'internet', tv', radio', and 'newspaper' are 19.74, 6.54, 7.05, and 7.86 respectively. The category 'internet', and 'radio' are negatively correlated with one another**.

Figure FC5.10: Comparison of initial and master summaries generated in our design study for line chart images. The boldface formatted excerpts in row E2 are based on the observation made on user response analysis on bar charts.

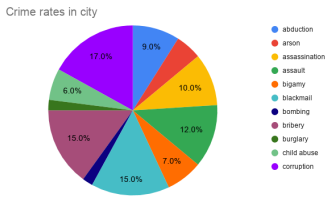
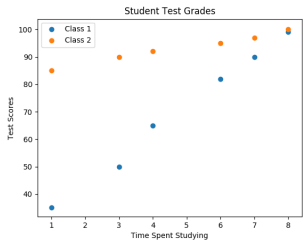
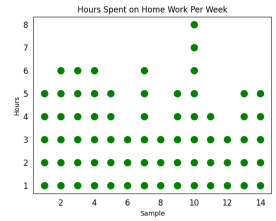
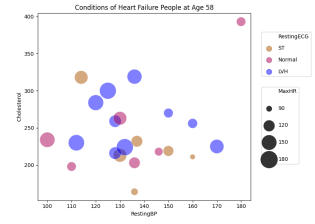
	Chart Image	Master Summary
(i)		The plot depicts a Pie Graph illustrating crime rates in city that compares across the following categories: abduction, arson, assassination, assault, bigamy, blackmail, bribery, child abuse, with proportions 8.5%, 5.5%, 9.8%, 12.1%, 6.1%, 15.2%, 15.7%, 5.3%, respectively where corruption contributing to majority of 17.4% and bombing, and burglary contributing to minority of 2.1%.
(ii)		The plot depicts a Scatter Plot illustrating student test grades. The plot is between test scores on y-axis over time spent studying on the x-axis for class 2, and class 1. The 'time spent studying' range from 1.01 to 7.96, with a mean value of 4.81. The 'class 2' range from 35.1 to 99.02, with a mean value of 70.26. The 'class 1' range from 85.29 to 100.39, with a mean value of 93.46. The categories 'time spent studying' and 'class 2' are positively correlated by 1.0.The categories 'class 2' and 'class 1' are positively correlated by 0.99.
(iii)		The plot depicts a Simple Dot Plot Graph illustrating hours spent on homework per week. The plot is between the hours on y-axis after sample on the x-axis. The house has its maximum and minimum values 8 and 3 at 10, and 6 respectively. The overall mean and standard deviation values of hours are 5.0 and 1.36 respectively.
(iv)		The plot depicts a Bubble Plot Graph illustrating conditions of heart failure people at age 58. The plot is between cholesterol on y-axis over restingbp on the x- axis for lvh, normal, and St variants. The cholesterol of lvh has its maximum and minimum values 320 and 216 at restingbp 136, and 128 respectively. For the restingbp ranging from 130-179 at the interval 30, the cholesterol of normal are 263, 235, 218, 204, 198, and 393 respectively. For the restingbp ranging from 130-160 at the interval 7, the cholesterol of the site are 211, 232, 219, 319, 165, and 212 respectively. The restingbp and cholesterol don't exhibit any correlation for categories 'lvh', normal', and 'St'.

Figure FC5.11: The final set of master summaries generated based on the observation made on bar charts for the following chart images (i) pie chart, (ii) simple scatter plot, (iii) dot plot, and (iv) bubble plot.

generation for different bar chart types. We observe that the summaries are qualitatively complete after the design study. The summaries for grouped and stacked bar charts are longer and contain more details, serving well as the master summary. However, we need to carefully design a sequence of user studies for quantitative evaluation of summaries based on human ratings and improve and test the selective mechanism of master summaries among different categories of population based on their level of understanding and purpose of usage of summary. Unlike user studies, metrics like BLEU (Bi-Lingual Evaluation Understudy) are used to assess sentence prediction tasks using similarity scores between ground truth sentences and predicted sentences. These evaluation methods used in NLG models usually have fixed lengths of text for ground truth and predicted sentences. Moreover, a dataset on chart summaries with ideal ground truth or candidate summaries is challenging, thereby paving the way for increased focus on developing advanced NLG approaches and chart-summary dataset generation in the chart interpretation area [40, 41].

5.6 Summary

Overall, we conclude that the assessment module of the user study substantiates the efficiency of the initial set of system summaries, along with a set of improvements to redesign sentence structure for a master summary generation. Our system-generated master summaries comprehensively represent the significant features and visually perceivable details of the extracted data table and its chart image, respectively, using predefined templates derived from our design study methodology of summary generation.

Improving chart summarization further requires an iterative process of feedback gathering and evaluation through a user study and the refinement of the summary based on a new set of observations. Hence, a careful design of subsequent user studies is within the scope of our future work. An alternative evaluation of summaries is based

on automatic evaluation metrics such as BLEU score [56] work by comparing words of summaries with a set of good quality candidate summaries. However, this approach is best suited for machine translation tasks from one natural language to other natural languages; still, several of the state-of-the-art methods [40, 41] use BLEU for summary evaluation. We have not used the BLEU evaluation in our work. Even though we applied BLEU with respect to the creative summaries of users, BLEU underperformed owing to the variety and language styles in the creative summaries. An alternative that can be pursued in the future is to generate candidate summaries for all chart images manually and to compute BLEU scores of the system-generated summaries based on content selection, relation generation, and content ordering.

CHAPTER 6

CONCLUSIONS

We have proposed an end-to-end, fully automated system for data table extraction and text summarization in chart images. In the preprocessing stage of our workflow, we have proposed a chart component extraction mechanism to facilitate the data extraction process by performing chart component-wise analysis inspired by the cognitive mechanism of the graphical perception. The image processing methods proposed for noise-free canvas component extraction improve the performance of the data extraction module. The predefined deep learning-based text detection and recognition models used are more robust than the open-source Tesseract-OCR engine by google due to training in natural scene text images. However, due to extensive training in natural scene text, the text recognition model fails to recognize a few special symbols used with numerical notations, which can further be improved by retraining the model with chart-text datasets.

Our data extraction mechanism proposed using color-based decoding of graphical objects, tackle the challenges in both line charts and scatter plot variants, and outperforms the state-of-the-art object detection models for graphical object detection in computer vision. In pie charts, regardless of a few failures in circle detection due to hough transform, it performs exceptionally well with a very low error rate in data extraction. Meanwhile, the bar chart exhibits a bit high error rate in data extraction compared to

other chart types due to inefficiency in the segmentation of bars with small heights and improper clustering of corner points in neighboring bars with minor height differences.

Our design study methodology for summary generation provides the predefined template with information from the chart type details, perceivable visual features in the chart, and significant information about data extracted from the chart image. The comparative scores generated by the assessment module of the user study state that system-generated summaries perform exceptionally well, especially in images with less chart text. Apart from a few minor modifications, our system-generated summaries are sufficient to describe the chart images. The proposed grammatical suggestions in the user study are later corrected using an open-source python library called *GingerIt* [54]. Summarization achieved from our system has the potential of being used in a language processing module, such as `gtts` [57] in Python, to generate an audio summary of the given chart image for the visually impaired audience. Our text summary can be pruned to provide crisper summaries per user input. Overall, our research demonstrates encouraging results in autonomously interpreting chart images and subtypes, using largely human-out-of-the-loop approaches.

6.1 Future Work

As discussed in the above chapters 3, 4, 5, our workflow has constraints due to its dependency on object size, parametric based methods, color-based extraction, and dataset availability, thereby establishing the groundwork for future research. The object size affects the canvas extraction and corner point clustering processes of our workflow. On the other hand, high-resolution chart images are rarely affected by their size because more pixels are used to represent an object. So further improving the quality of chart images may yield better results for the data extraction process. Besides object size and image quality, parameter-centric processes like hough circle transform also affect the

canvas extraction process, further raising the need for adaptive parameter selection of algorithm.

The color-based decoding of graphical object properties also affects data extraction in chart images that uses other encodings apart from color to represent the data. In such cases, a feature-based data extraction system that considers all properties like color and texture while decoding the graphical objects will make the system much more robust.

The master summaries generated based on sentence structure improvements made from user observations need to be further evaluated using another user study along with a human rating-based evaluation to evaluate using automatic metrics like BLEU. Our future scope lies in collecting a data set of ground truth summaries along with further evaluations of our master summary. The dataset collection will further motivate our research towards advanced NLG techniques for chart summary generation, a highly under-explored research area. Finally, we would like to build applications such as an interactive tool to access data tables and chart summaries, especially for improving the accessibility of charts among visually impaired people via audio.

Bibliography

[1] S. E. Kahou, V. Michalski, A. Atkinson, Á. Kádár, A. Trischler, and Y. Bengio, "FigureQA: An annotated figure dataset for visual reasoning," *arXiv preprint arXiv:1710.07300*, 2017.

[2] N. Methani, P. Ganguly, M. M. Khapra, and P. Kumar, "PlotQA: Reasoning over Scientific Plots," in *The IEEE Winter Conference on Applications of Computer Vision*, 03 2020, pp. 1516–1525, doi: 10.1109/WACV45572.2020.9093523.

[3] L. Ferres, G. Lindgaard, L. Sumegi, and B. Tsuji, "Evaluating a tool for improving accessibility to charts and graphs," *ACM Trans. Comput.-Hum. Interact.*, vol. 20, no. 5, nov 2013. [Online]. Available: https://doi.org/10.1145/2533682.2533683

[4] G. L. Lohse, "A cognitive model for understanding graphical perception," *Hum.-Comput. Interact.*, vol. 8, no. 4, p. 353–388, dec 1993. [Online]. Available: https://doi.org/10.1207/s15327051hci0804_3

[5] W. Huang and C. L. Tan, "A system for understanding imaged infographics and its applications," in *Proceedings of the 2007 ACM Symposium on Document Engineering*, ser. DocEng '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 9–18. [Online]. Available: https://doi.org/10.1145/1284420.1284427

[6] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings*

of the 24th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 393–402. [Online]. Available: https://doi.org/10.1145/2047196.2047247

[7] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo, "ChartSense: Interactive Data Extraction from Chart Images," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 6706–6717. [Online]. Available: https://doi.org/10.1145/3025453.3025957

[8] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "BarChartAnalyzer: Digitizing Images of Bar Charts," in *International Conference on Image Processing and Vision Engineering (IMPROVE), in press*. INSTICC, 2021, [Online; accessed 19-May-2022]. [Online]. Available: https://www.iiitb.ac.in/gvcl/pubs/2021_DadhichDaggubatiSreevalsanNair_IMPROVE_preprint.pdf

[9] S. Demir, S. Carberry, and K. F. McCoy, "Summarizing Information Graphics Textually," *Computational Linguistics*, vol. 38, no. 3, pp. 527–574, 09 2012. [Online]. Available: https://doi.org/10.1162/COLI_a_00091

[10] D. Chester and S. Elzer, "Getting computers to see information graphics so users do not have to," in *Proceedings of the 15th International Conference on Foundations of Intelligent Systems*, ser. ISMIS'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 660–668. [Online]. Available: https://doi.org/10.1007/11425274_68

[11] R. P. Futrelle, M. Shao, C. Cieslik, and A. E. Grimes, "Extraction, layout analysis and classification of diagrams in pdf documents," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ser. ICDAR '03. USA: IEEE Computer Society, 2003, p. 1007, doi : 10.5555/938980.939530.

[12] J. Gao, Y. Zhou, and K. E. Barner, "View: Visual information extraction widget for improving chart images accessibility," in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 2865–2868, doi : 10.1109/ICIP.2012.6467497.

[13] V. Karthikeyani and S. Nagarajan, "Machine learning classification algorithms to recognize chart types in portable document format (pdf) files," *International Journal of Computer Applications*, vol. 39, pp. 1–5, 2012, doi : 10.5120/4789-6997.

[14] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker, "Beagle: Automated extraction and interpretation of visualizations from the web," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–8. [Online]. Available: https://doi.org/10.1145/3173574.3174168

[15] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist, "Visualizing for the non-visual: Enabling the visually impaired to use visualization," in *Computer Graphics Forum*, vol. 38, no. 3. Wiley Online Library, 2019, pp. 249–260. [Online]. Available: https://doi.org/10.1111/cgf.13686

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1, doi : 10.1109/CVPR.2005.177.

[17] J. Thiyam, S. R. Singh, and P. K. Bora, *Chart Classification: An Empirical Comparative Study of Different Learning Models*. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3490035.3490291

[18] P. De, “Automatic data extraction from 2d and 3d pie chart images,” in *2018 IEEE 8th International Advance Computing Conference (IACC)*, 2018, pp. 20–25, doi : 10.1109/IADCC.2018.8692104.

[19] R. A. Al-Zaidy and C. L. Giles, “Automatic extraction of data from bar charts,” in *Proceedings of the 8th International Conference on Knowledge Capture*, ser. K-CAP 2015. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2815833.2816956

[20] K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju, “Chart mining: A survey of methods for automated chart analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3799–3819, 2021, doi : 10.1109/TPAMI.2020.2992028.

[21] T. API, “Google open Source, url: https://tesseract-ocr.github.io,” 2005, , accessed 19-May-2022]. [Online]. Available: https://tesseract-ocr.github.io

[22] A. Balaji, T. Ramanathan, and V. Sonathi, “Chart-text: A fully automated chart image descriptor,” *CoRR*, vol. abs/1812.10636, 2018. [Online]. Available: http://arxiv.org/abs/1812.10636

[23] J. Poco and J. Heer, “Reverse-engineering visualizations: Recovering visual encodings from chart images,” *Comput. Graph. Forum*, vol. 36, no. 3, p. 353–363, jun 2017. [Online]. Available: https://doi.org/10.1111/cgf.13193

[24] R. A. Al-Zaidy and C. L. Giles, “A Machine Learning Approach for Semantic Structuring of Scientific Charts in Scholarly Documents,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI’17. San Francisco, California, USA: AAAI Press, 2017, p. 4644–4649, doi : 10.5555/3297863.3297868.

[25] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, “FigureSeer: Parsing result-figures in research papers,” in *European Conference on*

Computer Vision. Springer, 2016, pp. 664–680. [Online]. Available: https://doi.org/10.1007/978-3-319-46478-7_41

[26] A. Baucom and C. Echanique, "Scatterscanner: Data extraction and chart restyling of scatterplots," in *Proc. SIGCHI*, 2013.

[27] M. Cliche, D. Rosenberg, D. Madeka, and C. Yee, "Scatteract: Automated extraction of data from scatter plots," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 135–150. [Online]. Available: https://doi.org/10.1007/978-3-319-71249-9_9

[28] S. Demir, S. Carberry, and K. F. McCoy, "Generating textual summaries of bar charts," in *Proceedings of the Fifth International Natural Language Generation Conference*, ser. INLG '08. USA: Association for Computational Linguistics, 2008, p. 7–15, doi : 10.5555/1708322.1708327.

[29] S. Carberry, S. Elzer, and S. Demir, "Information graphics: An untapped resource for digital libraries," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 581–588. [Online]. Available: https://doi.org/10.1145/1148170.1148270

[30] Y. Zhou and C. L. Tan, "Hough-based Model for Recognizing Bar Charts in Document Images," in *Document Recognition and Retrieval VIII*, vol. 4307. International Society for Optics and Photonics, 2000, pp. 333–340, doi : 10.1109/ICIP.2000.899506.

[31] J. Sreevalsan-Nair, K. Dadhich, and S. C. Daggubati, "Tensor Fields for Data Extraction from Chart Images: Bar Charts and Scatter Plots," in *Topological Methods in Visualization: Theory, Software and Applications, in press*, I. Hotz, T. B. Masood, F. Sadlo, and J. Tierny, Eds. Springer-Verlag, and arXiv preprint, 2020. [Online]. Available: https://arxiv.org/abs/2010.02319

[32] K. Dadhich, S. C. Daggubati, and J. Sreevalsan-Nair, "ScatterPlotAnalyzer: Digitizing Images of Charts Using Tensor-based Computational Model," in *International Conference on Computational Science (ICCS), in press*. Springer, LNCS, 2021. [Online]. Available: https://doi.org/10.1007/978-3-030-77977-1_6

[33] S. Ray Choudhury, S. Wang, and C. L. Giles, "Curve separation for line graphs in scholarly documents," in *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, ser. JCDL '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 277–278. [Online]. Available: https://doi.org/10.1145/2910896.2925469

[34] K. Davila, C. Tensmeyer, S. Shekhar, H. Singh, S. Setlur, and V. Govindaraju, "Icpr 2020 - competition on harvesting raw tables from infographics," in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, Eds. Cham: Springer International Publishing, 2021, pp. 361–380.

[35] R. Stewart and M. Andriluka, "End-to-end people detection in crowded scenes," *CoRR*, vol. abs/1506.04878, 2015. [Online]. Available: http://arxiv.org/abs/1506.04878

[36] A. Böhm, A. Ücker, T. Jäger, O. Ronneberger, and T. Falk, "Instance segmentation of overlapping biological objects using deep learning," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 1225–1229, doi: 10.1109/ISBI.2018.8363792.

[37] T. L. Mahyari and R. M. Dansereau, "Deep learning methods for image decomposition of cervical cells," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1110–1114, doi : 10.23919/Eusipco47968.2020.9287435.

[38] Y. Zhou, H. Chen, J. Xu, Q. Dou, and P. Heng, "Irnet: Instance relation network for overlapping cervical cell segmentation," *CoRR*, vol. abs/1908.06623, 2019. [Online]. Available: http://arxiv.org/abs/1908.06623

[39] C. Liu, L. Xie, Y. Han, D. Wei, and X. Yuan, "Autocaption: An approach to generate natural language description from visualization automatically," in *Proceedings of 2020 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2020, pp. 191–195, doi : 10.1109/PacificVis48177.2020.1043.

[40] J. Obeid and E. Hoque, "Chart-to-Text: Generating Natural Language Descriptions for Charts by Adapting the Transformer Model," in *Proceedings of the 13th International Conference on Natural Language Generation*. Dublin, Ireland: Association for Computational Linguistics, Dec 2020, pp. 138–147. [Online]. Available: https://aclanthology.org/2020.inlg-1.20

[41] L. Chen and K. Zhao, "An Approach for Chart Description Generation in Cyber–Physical–Social System," *Symmetry*, vol. 13, no. 9, p. 1552, 2021. [Online]. Available: https://doi.org/10.3390/sym13091552

[42] H. Singh and S. Shekhar, "STL-CQA: Structure-based transformers with localization and encoding for chart question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3275–3284. [Online]. Available: https://aclanthology.org/2020.emnlp-main.264

[43] A. P. Deshpande and C. Mahender, "Summarization of Graph Using Question Answer Approach," in *Information and Communication Technology for Sustainable Development*. Springer, 2020, pp. 205–216. [Online]. Available: https://doi.org/10.1007/978-981-13-7166-0_20

[44] P. Moraes, G. Sina, K. McCoy, and S. Carberry, "Evaluating the accessibility of line graphs through textual summaries for visually impaired users," in

Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility. ACM, 2014, pp. 83–90. [Online]. Available: https://doi.org/10.1145/2661334.2661368

[45] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012. [Online]. Available: https://doi.org/10.1109/TVCG.2012.213

[46] T. Munzner, *Visualization Analysis and Design*. CRC press, 2014.

[47] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, 2007, pp. 629–633, doi : 10.1109/ICDAR.2007.4376991.

[48] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," *CoRR*, vol. abs/1904.01941, 2019. [Online]. Available: http://arxiv.org/abs/1904.01941

[49] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis," *CoRR*, vol. abs/1904.01906, pp. 4714–4722, 10 2019.

[50] G. Xin, C. Ke, and H. Xiaoguang, "An improved canny edge detection algorithm for color image," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 113–117. [Online]. Available: https://doi.org/10.1109/INDIN.2012.6301061

[51] A. M. . A. K., "Image segmentation with watershed algorithm," pp. 142–148, 2017, accessed 19-May-2022. [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/_/downloads/en/stable/pdf/

[52] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras, "Icdar 2013 robust reading competition," in *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 1484–1493. [Online]. Available: https://doi.org/10.1109/ICDAR.2013.221

[53] D. Kirchhoff, S. Kuhnt, L. Bloch, and C. Müller, "Detection of circlelike overlapping objects in thermal spray images," *Quality and Reliability Engineering International*, vol. 36, 07 2020. [Online]. Available: https://doi.org/10.1002/qre.2689

[54] Tim Kleinschmdit, "Correcting spelling and grammar mistakes based on the context of complete entences. wrapper around the gingersoftware.com api," https://pypi.python.org/pypi/gingerit, 2019, [Online; accessed 19-May-2022].

[55] S. Demir, D. Oliver, E. Schwartz, S. Elzer, S. Carberry, and K. F. McCoy, "Interactive sight into information graphics," in *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, ser. W4A '10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1805986.1806009

[56] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[57] Pierre Nicolas Durette, "gtts (google text-to-speech), a python library and cli tool to interface with google translate's text-to-speech api." https://pypi.org/project/gTTS/, 2020, [Online; accessed 19-May-2022].

APPENDIX A

CHART DATA COLLECTION

The following website references are a few data sources used in the synthetic chart dataset generation for our system. These data sources are the datasets published in the online public data platform *Kaggle* which is used by many data analysts and users to find, publish, and explore datasets.

- https://www.kaggle.com/datasets/baranb/fut-22-ultimate-team-dataset
- https://www.kaggle.com/datasets/abhimaneukj/covid19-dataset
- https://www.kaggle.com/datasets/semihyilmaz/turkish-airlines-daily-stock-prices-since-2013
- https://www.kaggle.com/datasets/agajorte/zoo-animals-extended-dataset
- https://www.kaggle.com/datasets/khalidative/crimeanalysis
- https://www.kaggle.com/datasets/htagholdings/property-sales
- https://www.kaggle.com/datasets/edumagalhaes/quality-prediction-in-a-mining-process
- https://www.kaggle.com/datasets/annieichen/top-20-largest-california-wildfires
- https://www.kaggle.com/datasets/stefanoleone992/filmtv-movies-dataset
- https://www.kaggle.com/datasets/greeshmagirish/road-accidents-in-india-20142017
- https://www.kaggle.com/datasets/greeshmagirish/worldbank-data-on-gdp-population-and-military
- https://www.kaggle.com/datasets/mpwolke/cusersmarildownloadsetudiantscsv
- https://www.kaggle.com/datasets/smid80/weatherww2
- https://www.kaggle.com/datasets/nitinsss/military-expenditure-of-countries-19602019
- https://www.kaggle.com/datasets/airqualityanthony/leamington-aurn-air-quality-data
- https://www.kaggle.com/datasets/pavan9065/internet-usage

- https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset
- https://www.kaggle.com/datasets/cityapiio/world-cities-average-internet-prices-2020

The following website references are sources of bar chart images that we have used as test data in our web-based user study conducted to assess the effectiveness of our system-generated summary of the chart.

- http://ritsokiguess.site/docs/2018/01/20/displaying-grouped-bar-charts-in-ggplot/
- https://stackoverflow.com/questions/70239850/coding-a-bivariate-barplot-with-a-non-numerical-variable
- https://r-graphics.org/recipe-bar-graph-adjust-width
- https://viz-ggplot2.rsquaredacademy.com/ggplot2-histogram.html
- https://machinelearningmastery.com/data-visualization-methods-in-python/
- https://sscc.wisc.edu/sscc/pubs/stata_bar_graphs.htm
- https://devexpress.github.io/dotnet-eud/interface-elements-for-desktop/articles/charting.html