Journal of Visual Languages and Computing 48 (2018) 101–109

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

View Points

Chart decoder: Generating textual and numeric information from chart images automatically

Wenjing Dai, Meng Wang, Zhibin Niu, Jiawan Zhang*

School of Computer Software, Tianjin University, No. 135, Yaguan Road, Jinnan District, Tianjin, China

ARTICLE INFO

Keywords:
Visualization
Computer vision
Text recognition
Information extraction

ABSTRACT

Charts are commonly used as a graphical representation for visualizing numerical data in digital documents. For many legacy charts or scientific charts, however, underlying data is not available, which hinders the process of redesigning more effective visualizations and further analysis of charts. In response, we present Chart Decoder, a system that implements decoding of visual features and recovers data from chart images. Chart Decoder takes a chart image as input and generates the textual and numeric information of that chart image as output through applying deep learning, computer vision and text recognition techniques. We train a deep learning based classifier to identify chart types of five categories (bar chart, pie chart, line chart, scatter plot and radar chart), which achieves a classification accuracy over 99%. We also complement a textual information extraction pipeline which detects text regions in a chart, recognizes text content and distinguishes their roles. For generating textual and graphical information, we implement automated data recovery from bar charts, one of the most popular chart types. To evaluate the effectiveness of our algorithms, we evaluate our system on two corpora: 1) bar charts collected from the web, 2) charts randomly made by a script. The results demonstrate that our system is able to recover data from bar charts with a high rate of accuracy.

1. Introduction

Charts, as a way of visual perception, are widely used in economic reports, textbooks, scholarly documents, etc. The simplicity, usability and intuitiveness of charts make people use various types of charts in all kind of documents. Billions of chart images have been made and more charts would be produced in future. Many of these charts can be reused for other purposes such as redesign and further analysis. For example, iVolVER [1] implements chart transformation and allows users to create visualizations using graphical components in charts.

For most static charts, however, people do not have access to the raw data. Particularly in scholarly documents, the experimental data presented by different types of charts is an important source of valuable information. One way for people to obtain raw data of charts is to leverage the ability of human visual processing which is exhausting when faced with many charts. In addition, machines cannot decode textual and numeric information in charts like human as chart depictions are not designed for the machine. The lack of machine readability hinders further analysis, redesign and reuse of visualizations.

To implement reuse of chart data, many data extraction tools and systems [2–6] have been proposed. However, these tools and systems require a large amount of user operation, or exclude textual information extraction and only focus on data value extraction. Some tools like ChartSense [2] rely on manual annotation on textual components. They assume text localization and content are given by users. As a result, users have to manually type in text to accomplish the complete underlying data extraction. Some semi-automatic tools [2,3,5,6] require users to identify visual features such as x-axis, y-axis and coordinate range, while handling multiple series data is time-consuming and monotonous. Other previous works focus on extracting textual information from chart images. For example, Poco and Heer [7] recover the visual encodings from a chart image. However, they only recover the textual information and exclude the data value recovery of charts.

We present an end-to-end system that implements chart decoding based on graphical and textual features, and recovers textual and numeric information of chart images. Our system first identifies chart types of chart images (five categories), then performs specialized textual components localization, text extraction and recognition, and text role classification for textual information. Simultaneously, graphical components are detected and converted into corresponding numeric information. Then the mapping of values of bars and text elements is conducted for generating data files of chart images.

We make the following research contributions:

* Corresponding author.
E-mail addresses: meng.wang@tju.edu.cn (M. Wang), zniu@tju.edu.cn (Z. Niu), jwzhang@tju.edu.cn (J. Zhang).

https://doi.org/10.1016/j.jvlc.2018.08.005
Received 20 July 2018; Received in revised form 8 August 2018; Accepted 8 August 2018
Available online 11 August 2018
1045-926X/

1. A larger number of charts are collected for training.
2. Higher accuracies are achieved than previous work in terms of chart classification.
3. Classification of text roles in bar chart images based on text content and geometric properties of text elements.
4. Good accuracy is achieved for data interpretation in corpus 2.

In the remaining sections, we review related work in automated classification, text detection and recognition, and data extraction. Then, the design principles and implementation process of chart type classification and automated data extraction from chart images are described. To evaluate the effectiveness of our algorithms we compile two types of corpora of chart images. We also demonstrate successful examples in recovering chart information and discuss failure cases of our system. Finally, We finish with a discussion of the limitations and future works in this area.

2. Related work

Our work is built upon three areas of related research work: (1) chart image classification; (2) text detection and recognition; (3) data extraction from charts;

2.1. Chart type classification

Image classification [8–11] for natural scenes has become a well-studied problem. Many image classification algorithms based on image processing and machine learning have been proposed. And deep learning techniques [12–15] in image classification have higher recognition accuracy. For computer generated charts, some algorithms in classification of natural scenes can be applied and specialized techniques [4,16,17] for classification have also been developed.

Most chart classification approaches are proposed based on the extraction of artificially defined image features, which may be low-level, middle-level or high-level features. Shao and Futrelle [18] and Huang and Tan [19] extract graphic objects from vector graphics as figure attributes. Through detecting low-level shapes such as horizontal and vertical lines, they detect and classify six kinds of charts in scholarly documents. But their approach is not robust for chart image with noises and requires an accurate vectorization of charts.

Prasad et al. [17] apply Scale Invariant Feature Transform (SIFT) [20] and Histogram of Oriented Gradient (HOG) [21] features to represent the structural information of chart images. Then a multi-class Support Vector Machine (SVM) [22] is trained to achieve chart classification of five common categories. Savva et al. [4] design low-level features for classification combined with image features (like contour) and textual features. Zhou et al. [23] adopt Hough transform [24] for feature extraction and classification to recognize bar charts. ReVision [4] applies a classification method combined low-level image features and text-level features, which achieves an accuracy of 96% over 10 categories. Low-level image features extracted from chart images are performed using SVM. Then the extracted text-level features of chart images are used to improve the accuracy of classification.

Other approaches apply deep learning techniques to the classification of chart types. For example, ChartSense [2] applies neural networks to implement chart type classification; Tang et al. [16] combine convolutional networks (ConvNets) [13] and deep belief networks (DBNs) [25,26] to classify charts. They use natural images to train the convolutional networks and then fine-tune parameters by chart images.

2.2. Text localization, detection and recognition

Chart images contain text structural information, such as chart titles, legends and axis labels. These textual components in chart images can be extracted using text localization, detection and recognition techniques. For text localization, Neumann et al. [27] have proposed a real-time text localization and recognition approach for natural scenes. Huang et al. [28] separate textual and graphical components in chart images by detecting connected components. Jayant et al. [29] develop the tactile graphics process to separate text from images and generate location file.

To identify textual components in charts, the most common method is to use optical character recognition (OCR) engines like Tesseract [30]. Many existing systems [4,31,32] detect and separate textual components from the chart images, and then pass the text regions to OCR engines. For example, ReVision [4] extracts the text image region and then performs OCR using the Tesseract. This system also combines user interaction to correct OCR recognition results. All these systems are highly dependent on the accuracy of the OCR. However, due to the text sparsity [33] the accuracy of OCR on chart images is not very satisfactory. The inaccuracies of OCR tools result in incorrectly calculated X/Y-axis scale values.

For natural images, Jaderberg et al. [34] present a text spotting system, which adopts a region proposal mechanism to detect text and a deep Convolutional Neural Network (CNN) to recognize text string. Gupta et al. [35] train a Fully-Convolutional Regression Network to perform text detection. Their method outperforms current methods for text detection in natural images.

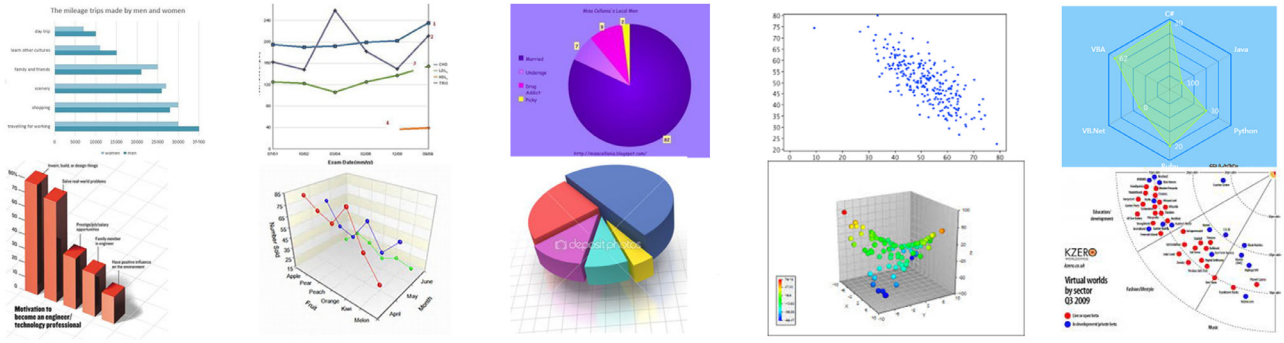
For chart images, Ray Choudhury and Giles [36] use a K-means based approach to group text segments, and then apply natural language processing techniques to scholarly figures for generating a machine-readable representation. Kataria et al. [33] create an edge map of images and extract connected components as the candidates of text characters. In their method, the components with an area greater than 20% of the image are discarded, and the distance between components is calculated to merge words. Poco and Heer [7] propose a more robust method for separating text and graphical elements of chart images. They use a CNN to identify and remove not-text pixels. In our method, we use the same network to remove not-text pixels and a similar text pipeline to extract textual information from chart images inspired by Poco and Heer [7].

2.3. Data extraction

Data extraction process of chart images is to detect graphical components and assign them to corresponding numerical values. Previous studies on data extraction from charts have developed several techniques which can be fully-automatic or interactive, such as connected component analysis, edge detection [37–39] and k-median filtering [19,33,40]. Shao and Futrelle [18] use parsing algorithms to define graphemes. Zhou and Tan [23] combine boundary tracing with Hough transform for bar graphical feature extraction. Huang et al. [19,41] separate the graphical and textural components into two separate images, and then generate edge maps from the graphical image. ChartSense [2] implements semi-automatic data extraction which makes users mark the baseline of x-axis and coordinate range. It detects bars and converts them into values according to the length of bars. For batch processing, however, ChartSense is monotonous and time-consuming.

3. Methodology

In this section we describe our approach on four pipelines: (1) Chart classification, (2) Textual component extraction, (3) Graphical component extraction, (4) Recovering chart data. We first implement chart type classification using convolutional networks. Although in this paper we only address data recovery of bar charts, we plan to implement data extraction for more chart types in future. So chart type classification is a necessary step in our system. In the process of textual component extraction, we use a similar text analysis pipeline with [7] but with different word detection and text role classification methods. We first separate text components from bar chart images and detect words. Then



(a) Bar charts (2501) (b) Line charts (2153) (c) Pie charts (2978) (d) Scatter plots (2107) (e) Radar charts (1435)

Fig. 1. Examples of chart images in our corpus.

detected words are passed to the OCR engine for recognition. Finally, we merge "close" words and classify text elements according to their roles in bar charts. In graphical component extraction, we recover numerical values of bars in bar charts. Finally, we combine text information and values of bars in bar chart images to recover the final chart data.

3.1. Chart classification

3.1.1. Our corpus for classification

To achieve chart image classification, we built a chart image corpus, which contains 11,174 chart images collected from six search engines (Google, Baidu, Yahoo, Bing, Aol. and Sogou). Five types of charts (bar chart, pie chart, line chart, scatter plot and radar chart) are included in our corpus. For each research engine, we input chart types as the search keywords and collected all images returned by the search engine. Then we manually removed incorrect or inappropriate search results, such as handmade sketches, images that include multiple types of charts.

Example chart images from each chart type are given in Fig. 1. Both 3D and 2D chart images are included in our corpus. Although we collected chart images from different search engines, about a quarter of images are duplicate. To get the exact number of chart images we removed the exactly same chart images from our corpus. But we did not remove duplicate images that have been rotated, blurred, or slightly edited. We treat these similar images as different individuals, which would improve the generalization of chart classification models. The number of chart images in five chart types is shown in Table 1.

3.1.2. Chart type classification

As we are interested in data extraction of chart images, we trained a CNN model for chart type classification. There are many CNN variations [12–15], which show a good performance in the ImageNet Recognition Competition (ILSVRC) competition. Alexnet [13] contains eight layers, which the first five layers are convolutional layers followed by fully connected layers. The VGG network architecture [15] use only 3×3 convolutional layers. Max pooling operation is done after several layers of convolutions to reduce volume size. GoogLeNet [14] uses combinations of inception modules, each including some pooling, convolutions at different scales and concatenation operations. ResNet [12] introduces residual modules and demonstrates that extremely deep networks can be trained using standard SGD through the use of residual modules.

We compare the accuracies of the above-mentioned CNN models based on our chart image corpus. We use a deep learning framework TFLearn [42] to experiment with the classification. TFLearn as a modular and transparent deep learning library supports most of recent deep learning models. In the training process, we first scaled each image in our corpus to a dimension of $224 \times 224 \times 3$ using OpenCV library. For chart images, 80% of the data was used for training and the rest was used as the verification set. Then we trained all four network models with the learning rate of 0.001 and the epoch number of 100. We used the GPU of NVIDIA Titan X to train the models, which took approximately 3 h for each model.

3.1.3. Chart classification performance

We first compare the accuracy of the four classification models (Table 2). As shown in Table 2, the training results of the four neural network models all have high accuracy rates, over 98%. Among them, the accuracy of ResNet is relatively lower than others but still very high. The accuracies of the four models have little difference with high accuracy rates over 98%, indicating that deep learning techniques can solve the problem of chart type classification perfectly. In our method, we choose GoogLeNet to implement chart classification based on the consideration of classification accuracy and the number of parameters.

Then we compare our classification results with ChartSense [2], a semi-automatic data extraction system. Our system and ChartSense both use deep learning network models to implement chart classification. ChartSense trains three CNN models in 10 chart categories (area, bar, line, map, Pareto, pie, radar, scatter plot, table, and Venn diagram). As shown in Table 3, although the chart categories of ChartSense are five more than ours, our training results are more accurate. This is mainly due to the relatively larger amount of data we have. In the same five categories of chart types, ChartSense has 2905 chart images while we have 11,174 chart images.

3.2. Corpora for data recovery

After automatically categorizing charts by type, our system proceeds to recover the underlying data which is represented by textual

Table 1
Chart corpus statistics.

Chart type	Collected
Bar chart	2501
Pie chart	2978
Line chart	2153
Scatter plot	2107
Radar chart	1435

Table 2
Training results of four network models (AlexNet, GoogLeNet, VGG16, ResNet) in chart classification.

CNN Model	Classification Accuracy
AlexNet	99.48%
GoogLeNet	99.07%
VGG16	99.55%
ResNet	98.89%

Table 3
Chart type classification accuracies for our models and ChartSense.

CNN Model	Ours	ChartSense(6997 chart images)
LeNet-1	–	44.2%
AlexNet	99.48%	88.8%
GoogLeNet	99.07%	91.3%

Table 4
The numbers of bar charts in two corpora.

Corpora	Amount
Corpus 1	59
Corpus 2	500

components and graphical components. We currently focus on implementing the underlying data extraction for bar charts, one of the most popular chart types. Before introducing our methods, we explain how we built two chart corpora for evaluating the results of data extraction.

We evaluate our techniques of chart data extraction on two chart corpora. One (Corpus 1) is a subset of the corpus used for chart type classification, the other (Corpus 2) contains chart images made randomly by a script. The number of bar charts in each corpus is shown in Table 4.

Our algorithms are based on a few simplifying assumptions regarding bar charts:

1. Charts do not have 3D effects.
2. There are gaps between different series.
3. Bars are not overlapped by text and text elements are not overlapped.
4. Bar charts do not contain stacked bars.
5. Axes appear at the left and bottom of charts.
6. Chart titles are on the top of charts.
7. X-axis labels are at the left of horizontal bars or at the bottom of vertical bars.

For the image corpus used in chart classification, we first filtered out the images without the ground truth. After that, the number of bar charts with the ground truth is 84. Finally, 59 bar charts satisfy the assumptions, which consist of Corpus 1. Most chart images collected for the web are heavily compressed and noisy. To reduce the impact of noise, we apply the bilateral filter [43] to chart images as a pre-processing step, which removes noise and retains sharp edges in the image.

As many charts in Corpus 1 are blurred and hard for recognition, to effectively evaluate our algorithm we randomly generated 500 bar charts satisfying the assumptions. We developed a script to randomly select values and generate bars. Fig. 2 (a) shows a sample of bar charts generated by our script.

3.3. Textual component extraction

In this section, our goal is to extract text elements in a bar chart and classify them according to their roles. We first separate text components from bar chart images using a text pixel classification. Then connected components of text are detected and passed to the OCR engine for recognition. Finally, we merge 'close' words and classify the text elements according to their roles in bar charts.

3.3.1. Text detection and recognition

To detect text elements in a bar chart image, we first separate the textual components from graphical components. We adopt a similar text pixel classifier method proposed in [7], which removes pixels that do not correspond to text. Once textual components have been separated from graphical components, we proceed in text detection and recognition. Fig. 2 summarizes our method about textual component separation and detection.

Textual component separation: First, we resize the input chart (Fig. 2 (a)) but maintain the aspect ratio of images. Then we binarize the image using OpenCV library (Fig. 2 (b)). Next, we use a trained convolutional network [44] to implement text detection and separation from chart images. The network is implemented in Darknet [45], which is a CNN framework written in C and CUDA. 500,000 labelled figures extracted from research papers are used to train the network. Given a pixel image as input, the network outputs a heatmap of the probability of text in the image. Then the output heatmap is converted to the original size for following text detection. Fig. 2 (c) shows an example of the results after applying the text pixel classifier.

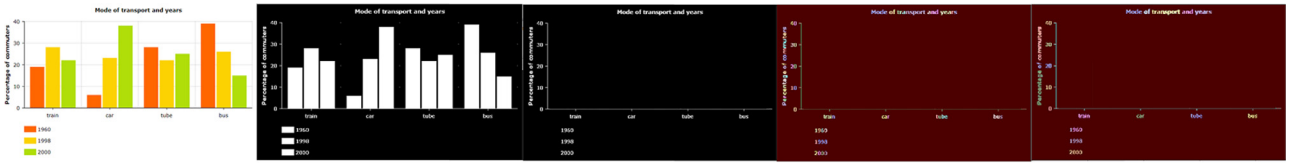
Locating text regions: In this step, we apply the connected components algorithm to find all connected components by 8-way connectivity, which recognizes letters in words as connected components. The result is shown in Fig. 2 (d). Each connected component is labeled in a random color. For the letters detected, we assume that the distances between letters centroids are closer than the distances between words. Based on this assumption we extract the centroids of components and calculate their distances between each other. The distances between letters vary in a small range. So we set a threshold to discard the distances larger than the threshold and draw a histogram of letter distances (Fig. 3). We extract the distance range most distributed in the histogram and set two letters into one label if the distance between them locates in the range. After resetting labels, a relabeled image is generated (Fig. 2 (e)).

Text recognition: Next, we split the text image regions and perform OCR to the text regions. We use Google's open source OCR engine, Tesseract. In some special cases, text elements are rotated and not horizontal. To increase the accuracy of OCR, we rotate the text regions by an angle so that the text is horizontally oriented before passing to Tesseract. The skew angle of the text region is computed using the Probabilistic Hough Transform [46].

3.3.2. Text role classification

For the extracted words in the previous steps, we merge 'close' words into phrases and classify the text elements based on their roles. First, we set a minimum threshold for the distance between words. If the distance between two words is less than the minimum distance threshold and the words have the same orientation, the two words would be merged into one phrase.

We define text elements in a bar chart into six different roles based on their geometric properties and text content: (1) Title, (2) X-axis label, (3) Y-axis value, (4) Y-axis label, (5) Legend, and (6) Value label.



(a)Input chart (b)Binary image (c)Text pixels (d)Connected components (e)Words

Fig. 2. The process of textual component separation.

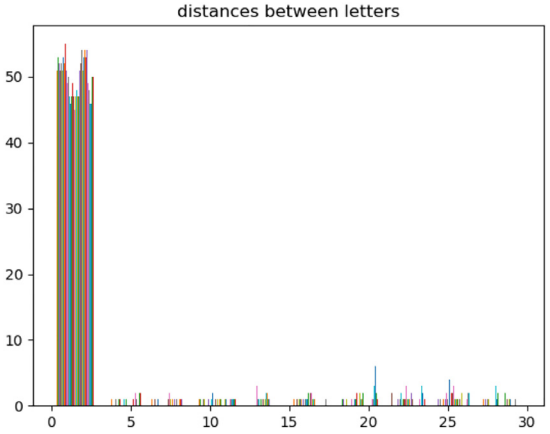


Fig. 3. The distribution histogram of distances between letters.

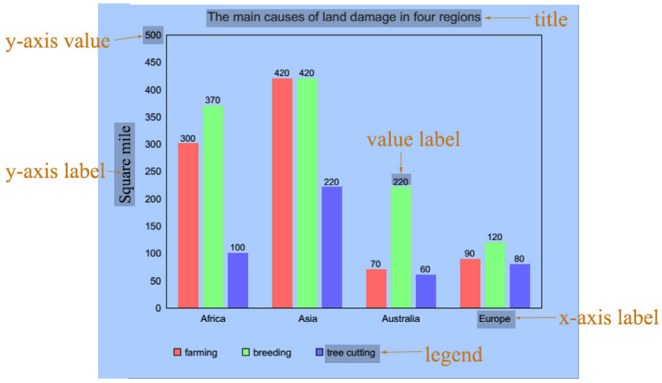


Fig. 4. Six text roles in a bar chart.

The six text roles are illustrated in Fig. 4. Most words in bar chart images can be classified as one of the roles. Given a set of text elements as input, our method classifies each according to its role. We discard the recognition of value labels because most charts do not include them. That is why we design our system.

In the text role classification process, we define a series of features based on text content and geometric properties of text elements. Then, we train a multi-class classifier using Support Vector Machines (SVM) [22] to implement text role classification in bar charts. Compared with previous work, we increase the baseline detection (x-axis and y-axis), which plays an important role in feature extraction and improves the accuracy of text role classification.

First, we identify the locations of x-axis and y-axis by Hough transform. We distinguish which axis is y-axis according to the bar orientation (the bar orientation inferring is interpreted in the next section). Based on the assumption that specific text roles are located in specific regions in chart images, we define text role features according to text contents and the spatial information of text elements. The role of a text element is inferred by four main features: its distance to axes, its distance and parallel relationship with other elements, its position in the chart image and word type. As shown in Fig. 5, the rectangular boxes of a, b, c, d and e represent the related regions of title, y-axis label, y-axis value, x-axis label and legend, respectively. Among these text elements, we first evaluate the positions of their centroids. The centroids of y-axis values would be parallel to the y-axis. Similarly, the centroids of x-axis labels would be parallel to the x-axis. This kind of parallel relationship is shown by the dotted boxes through the red dots in Fig. 5. Bar charts in our corpora can be vertical or horizontal. For vertical or horizontal charts, the rules are similar. Taking a horizontal chart as an example, sample features of six text types are shown as follows:

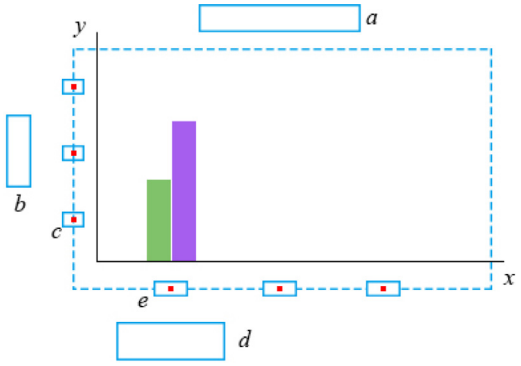


Fig. 5. geometric information and relative positions of text roles in horizontal bar charts.

1. Title: Title is at the top of the image within 10% of the image height and there is no text above it.
2. Y-axis label: Compared with y-axis values, the y-axis label is closer to the left edge of the image and further to the y-axis. There is no text on the left of y-axis label.
3. Y-axis value: First, y-axis values are numbers. Then through finding the parallel line to y-axis, text elements whose centroid falls in the line are y-axis values.
4. X-axis label: x-axis labels are near the x-axis and their centroids are parallel to x-axis.
5. Legend: Legends do not have a fixed position.

For Legends, they do not have specific regions in chart images and the direction of legend labels can be horizontal or vertical. However, the common point is that labels in legends lie on a vertical or horizontal line as y-axis values and x-axis labels. In addition, the vertical or horizontal span of legend labels is smaller than the whole span of y-axis values and x-axis labels. Therefore, we include this span feature to identify legends.

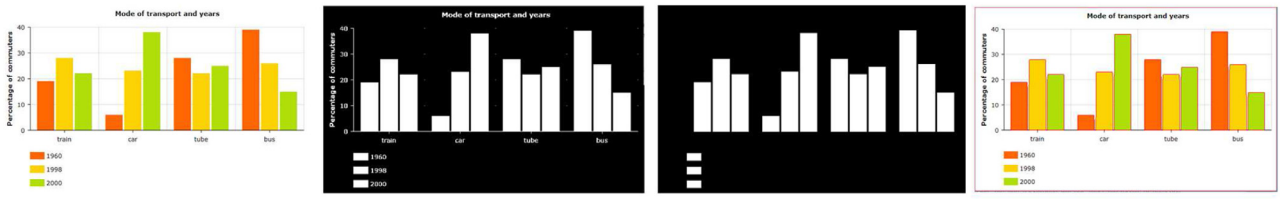
3.4. Graphical component extraction

In this section, we extract bars in bar chart images and infer their values. For the graphical components in a bar chart, We first find connected components as bars. Then we infer the chart orientation and determine which axis is the y-axis. Finally, we recover the numerical value of each bar by multiplying its height in pixels by a scaling factor.

Bar detection: Given a bar chart as input, we first convert the image to a binary image using a global threshold approach. The optimal threshold value is find using Otsu thresholding [47]. Then we apply an open morphological transform with kernel size 3 to the binary image. This operation removes small white noises, axes or assistance grid lines. Next, to find the bars we perform the connected component labelling algorithm. As shown in Fig. 6, the detected bars are labelled by a red bounding box. As we have assumed, we do not require each series of bars have a distinct colour. The bar series are distinguished by the gap between them. The bars in the same series are recognized according to their order. In this case, our method can extract data from a bar chart within one color.

Chart orientation inferring: In the chart assumptions, the chart orientation could be horizontal or vertical. For different types of chart orientation, the widths of bars in the same image are always equal. For example, vertical bars vary in height but maintain a fixed width. Therefore, we could compare the variety of the width and height of rectangles to infer the chart orientation. If one side of the rectangles varies little, we mark the axis that parallels to the side as the x-axis. In contrast, the other axis is the y-axis.

Data recovery: For majority bar charts, the heights of bars are linearly proportional to their values. Based on this assumption, we can



(a) Input chart (b) Binary image (c) Open morphological transform (d) Bars detected

Fig. 6. Bar detection procedure. (a) The input chart. (b) Initial binarized image. (b) The result after open morphological transform. (d) bars detected in red boxes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

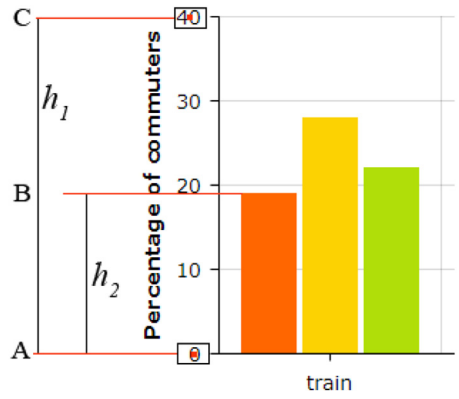


Fig. 7. Data scale of a bar.

recover the numerical values for each bar by multiplying its height by a scaling factor, which represents the mapping scale between pixels and data values. We use two data values (such as '40' and '0' in Fig. 7) in the y-axis to measure the y-axis scale. The data values of the y-axis are recognized in the section of textual component extraction. Then we calculate the y-axis scale according to the distance between A and C (Fig. 7). The scaling factor can be calculated as $(40 - 0)/h_1$. And the data value of that bar is $((40 - 0)/h_1) \times h_2$.

3.5. Recovering chart data

After finishing the extraction of textual and graphical components, we generate the final data files of chart images, which include textual information and data values of bars. The example results are shown in the right of Fig. 8.

After text role classification, text elements are written to corresponding fields. If legend labels are correctly classified, the field of "legend" will be labelled as "True". Fields without corresponding text information will be set as null. The data of bar charts is described in the field of "dataTable". Then, we build the mapping rules between data values of bars and text labels in bar chart images. As we have assumed, there are gaps between different series, so we identify bar series according to the gap between them. For the mapping rules, we consider the sequence of bar series to be consistent with the sequence of x-axis labels. Similarly, we consider the sequence of bars in one series to be consistent with the sequence of legend labels. If legend labels are horizontal, the label in the left will be considered as the category of the first bar in each series. For vertical legend labels, the label in the top will be the mapping text of the first bar in each series. Each legend label is recorded as "category" in "dataTable". The mapping rules apply to both horizontal and vertical bar charts.

We do not follow the color rules of bars series for data mapping cause in some bar charts the color changes according to different values. On the other hand, the mapping rules do not apply to charts with zero values (no bar). We checked the chart images in our two corpora, the percentage of these charts is less than 2%, so we exclude the discussion of this type of charts.

4. Extraction results

To evaluate the effectiveness of our system for automated data extraction, we conduct our extraction algorithms on our two corpora (Table 4). In Corpus 1, We manually identified the labelled values above bars as raw data and labelled text roles. Chart images with labelled values that cannot be recognized by human eyes are discarded. Finally, we manually generated 59 underlying data files for bar chart images. Raw data and labelled text of bar charts in Corpus 2 are generated by our script that randomly generates bar charts.

Taking a bar chart image as input, our system generates a JSON format file for this chart. Fig. 8 shows two successful examples of generating textual and numeric information by our system. As shown in Fig. 8, (a) and (b) are two examples of input chart images. (a) is an example without legends and (b) is an example with legends. The right side is the output results of (a) and (b), respectively. If the chart has legends, the field of "legend" will be labelled as "True" and each legend is represented in the field of "category".

We also measured and analyzed the performance of three parts in our algorithms: (1) the performance of text recognition, (2) the performance of text role classification, (3) the accuracy of bar detection and data extraction.

Text recognition performance: To evaluate the text recognition performance, we first evaluate the recognition rate of text strings in bar chart images. In Corpus 1, we identify 90% of text strings. In Corpus 2, about 98% of text strings can be recognized. Then to calculate the accuracy rates of OCR, we compare the similarity of text strings recognized with the ground truth. We use Damerau–Levenshtein method to measure the similarity of two strings. If the similarity between the extracted word and the ground truth exceeds 0.9, we consider these two words are the same. Table 5 reports the accuracy of extracted words. Compared with images in Corpus 2, the accuracy of text recognition in Corpus 1 is lower because the images collected from the web are small blurred. But for the charts images made by our script, the performance of our text recognition methods is good.

Text role classification performance: With the correct recognized text string, we classify them to five text roles. To validate our text role classification approach, we use the precision, recall and F1-score metrics. All precision, recall and F1 scores are reported in Table 6 and Table 7, respectively. The average F1-score of Corpus 2 achieves 93%, which demonstrates the effectiveness of our methods. For the classification rules we defined, the performance of classification about Corpus 2 is better than Corpus 1. We believe this is because the bar charts randomly generated follow a more logical layout structure than charts from the web.

Data extraction results: For bar detection, our system detects all bars for 44/59 (74%) of bar charts in Corpus 1. In Corpus 2, 459/500 (92%) of bar charts is correctly detected. When validating the results, we found that small bars are hard to detect. Some failures occur when Otsu binarization fails to find the optimal threshold. If the difference between the extracted value and the ground truth lays in the deviation

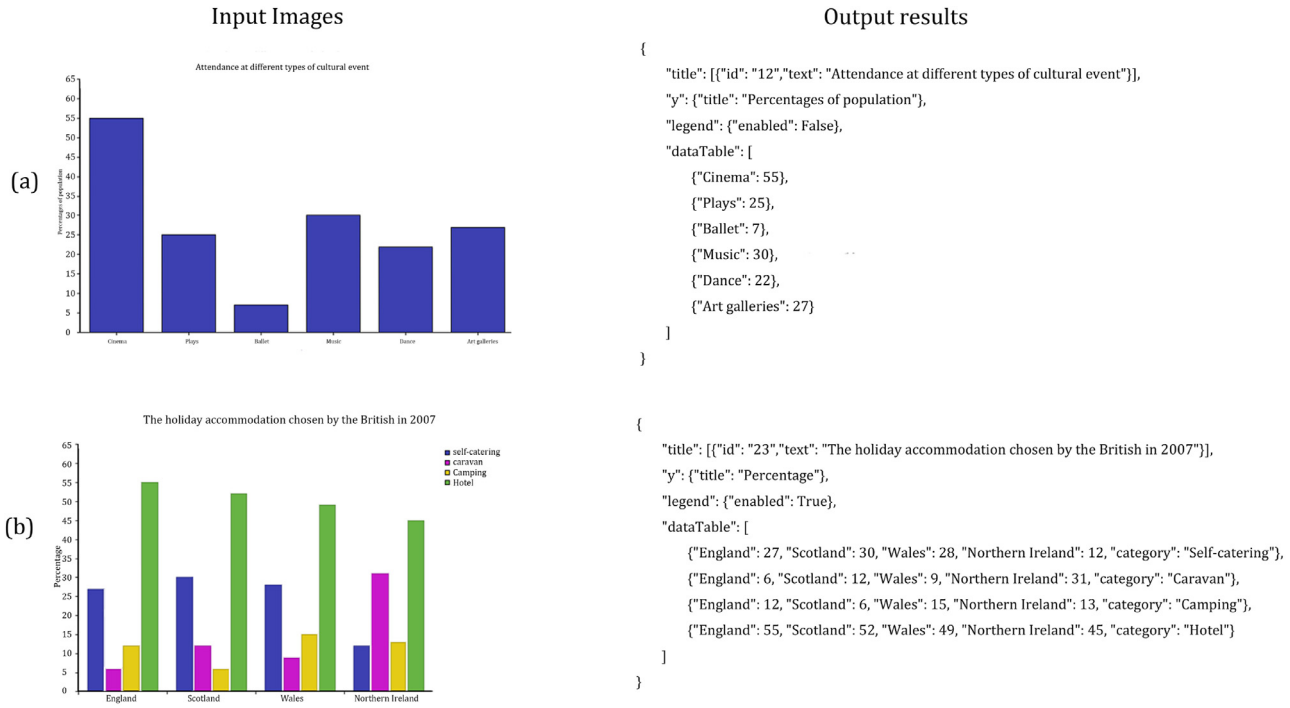


Fig. 8. Taking a bar chart image as input, Chart Decoder generates a data file of this chart image. (a) is a bar chart image without legends. (b) is a bar chart image with legends.

Table 5
Accuracy of text recognition in Corpus 1 and Corpus 2.

Corpora	Accuracy
Corpus 1	75%
Corpus 2	93%

Table 6
Performance of text role classification in Corpus 1.

Text role	Precision	Recall	F1-score
Title	80%	72%	76%
X-axis label	79%	86%	82%
Y-axis value	82%	74%	78%
Y-axis label	87%	80%	83%
Legend	83%	74%	78%

Table 7
Performance of text role classification in Corpus 2.

Text role	Precision	Recall	F1-score
Title	98%	95%	96%
X-axis label	95%	97%	96%
Y-axis value	92%	92%	92%
Y-axis label	93%	90%	91%
Legend	97%	93%	94%

of 2% of the ground truth, we consider the extracted value is correct. Based on this ratio, for Corpus 1 about 77% of extracted values are accurate. For Corpus 2, 89% of extracted values are accurate.

We also noted some errors may occur due to the failures of bar detection, text localization, OCR and text role classification. Fig. 9 shows the four common failure cases occurred in our system. As shown in Fig. 9 (a), we failed to detect small bars in bar detection, especially for small and unclear images. Another error arises due to the lack of corresponding bars. In Fig. 9 (b), there are three series of bar type while the value of one bar is zero (no bar). After data recovery, we conduct the mapping of x-axis labels and corresponding values. The lack of bars results in error mapping of values and text. In the process of text role classification, text elements located in unusual locations are misclassified (Fig. 9 (c)). In addition, text elements with tight spacing are incorrectly merged especially for x-axis labels. For example, in Fig. 9 (d) the spacing of different labels is closer than the spacing of different words in one label.

5. Limitations and future work

Currently, our system has a high accuracy in chart classification and implement the textual and graphical component extraction for bar charts. However, our algorithms work under a set of assumptions of chart types and styles. More work is needed to make our approach more extensible and overcome limitations of extraction methods.

For textual and graphical component extraction, the scope can be extended, such as charts with 3D effects and other chart types. For textual elements in charts, we could further improve the accuracy of text detection and recognition for data recovery. For the process of generating data files, the case of Fig. 9 (b) can be discussed in detail and color rules of different series could be applied to improve the data results. On the other hand, compared with heuristic methods built around a set of assumptions, deep learning methods can be used for text detection and text role classification.

We also plan to implement other applications based on chart data extraction, such as chart redesign of ReVision and chart style transfer of iVolVER. For some legacy charts, the visual perception is poor so these charts can be redesigned. The redesign of charts can be user-interactive or automatically design based on a set of aesthetic rules.

6. Conclusion

We presented a system that automatically identifies the chart type, extracts textual and graphical components, and then infers the underlying data of input chart images. By collecting a larger data set, the

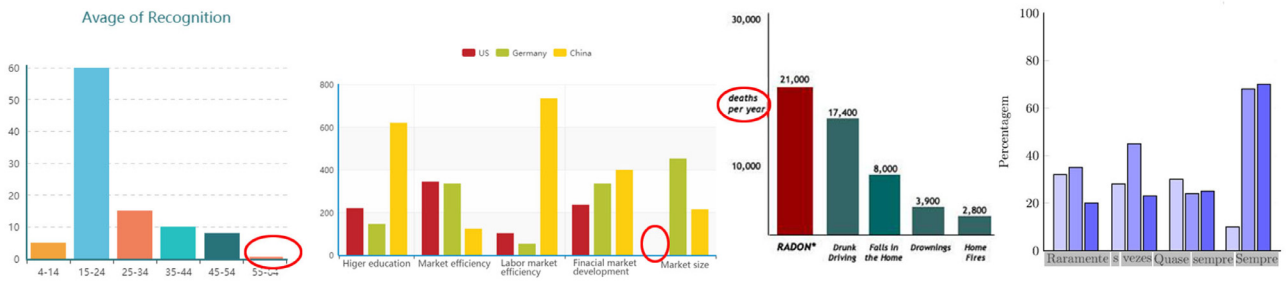


Fig. 9. The four common failure cases. If bars are very small, our algorithms may fail to detect those small bars, e.g., (a). Error mapping of values and text occurs when chart images like (b) lack bars. Text elements in unusual location are classified incorrectly, e.g., (c). When labels are in tight spacing like (d), text elements may be merged incorrectly.

trained classification models of chart types achieve higher accuracy than ChartSense. Our algorithms of data recovery extract and analyze the textual and graphical data separately. We also design spatial features of text elements for text role classification. Then we generate data files for input chart images using mapping rules. Our system is the first for automatically generating textual and numeric information from chart images. We only implemented the data extraction of bar charts. In future, we plan to apply data extraction algorithms to more chart types and improve the accuracy of data extraction.

Acknowledgements

We would like to thank the support of NVIDIA Corporation with the donation of the GeForce Titan X used for this research. This work was supported by the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1824), Zhejiang University.

References

[1] G.G. Méndez, M.A. Nacenta, S. Vandenheste, ivolver: Interactive visual language for visualization extraction and reconstruction, Proceedings of the CHI Conference on Human Factors in Computing Systems, ACM, 2016, pp. 4073–4085.
[2] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, J. Seo, Chartsense: Interactive data extraction from chart images, Proceedings of the CHI Conference on Human Factors in Computing Systems, ACM, 2017, pp. 6706–6717.
[3] A. Rohatgi, Webplotdigitizer, URL http://arohatgi.info/WebPlotDigitizer/app (2011).
[4] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, J. Heer, Revision: automated classification, analysis and redesign of chart images, Proceedings of the Twenty Forth Annual ACM Symposium on User Interface Software and Technology, ACM, 2011, pp. 393–402.
[5] W.R. Shadish, I.C. Brasil, D.A. Illingworth, K.D. White, R. Galindo, E.D. Nagler, D.M. Rindskopf, Using ungraph to extract data from image files: verification of reliability and validity, Behav. Res. Methods 41 (1) (2009) 177–183.
[6] L. Yang, W. Huang, C.L. Tan, Semi-automatic ground truth generation for chart image recognition, Proceedings of the International Workshop on Document Analysis Systems, Springer, 2006, pp. 324–335.
[7] J. Poco, J. Heer, Reverse-engineering visualizations: Recovering visual encodings from chart images, Computer Graphics Forum, 36 Wiley Online Library, 2017, pp. 353–363.
[8] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 3360–3367.
[9] O. Chapelle, P. Haffner, V.N. Vapnik, Support vector machines for histogram-based image classification, IEEE Trans. Neural Netw. 10 (5) (1999) 1055–1064.
[10] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 3642–3649.
[11] R.M. Haralick, K. Shanmugam, et al., Textural features for image classification, IEEE Trans. Syst. Man Cybern. (6) (1973) 610–621.
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016), pp. 770–778.
[13] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, (2012), pp. 1097–1105.
[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2015), pp. 1–9.
[15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[16] B. Tang, X. Liu, J. Lei, M. Song, D. Tao, S. Sun, F. Dong, Deepchart: combining deep convolutional networks and deep belief networks in chart classification, Signal Process. 124 (2016) 156–161.
[17] V.S.N. Prasad, B. Siddiquie, J. Golbeck, L.S. Davis, Classifying computer generated charts, Proceedings of the International Workshop on Content-Based Multimedia Indexing, IEEE, 2007, pp. 85–92.
[18] M. Shao, R.P. Futrelle, Recognition and classification of figures in pdf documents, Proceedings of the International Workshop on Graphics Recognition, Springer, 2005, pp. 231–242.
[19] W. Huang, C.L. Tan, A system for understanding imaged infographics and its applications, Proceedings of the ACM Symposium on Document Engineering, ACM, 2007, pp. 9–18.
[20] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1 IEEE, 2005, pp. 886–893.
[22] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[23] Y.P. Zhou, C.L. Tan, Hough technique for bar charts detection and recognition in document images, Proceedings of the International Conference on Image Processing, 2 IEEE, 2000, pp. 605–608.
[24] J. Illingworth, J. Kittler, A survey of the hough transform, Comput. Vis. Gr. Image Process. 44 (1) (1988) 87–116.
[25] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[26] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[27] L. Neumann, J. Matas, Real-time lexicon-free scene text localization and recognition, IEEE Trans. Pattern Anal. Mach. Intell. 38 (9) (2016) 1872–1885.
[28] W. Huang, Z. Lin, J. Yang, J. Wang, Text localization in natural images using stroke feature transform and text covariance descriptors, Proceedings of the IEEE International Conference on Computer Vision (ICCV), IEEE, 2013, pp. 1241–1248.
[29] C. Jayant, M. Renzelmann, D. Wen, S. Krisnandi, R. Ladner, D. Comden, Automated tactile graphics translation: in the field, Proceedings of the Ninth International ACM SIGACCESS Conference on Computers and Accessibility, ACM, 2007, pp. 75–82.
[30] R. Smith, An overview of the tesseract ocr engine, Proceedings of the Ninth International Conference on Document Analysis and Recognition, 2 IEEE, 2007, pp. 629–633.
[31] R.A. Al-Zaidy, C.L. Giles, Automatic extraction of data from bar charts, Proceedings of the Eighth International Conference on Knowledge Capture, ACM, 2015, p. 30.
[32] J. Poco, A. Mayhua, J. Heer, Extracting and retargeting color mappings from bitmap images of visualizations, IEEE Trans. Vis. Comput. Graph. 24 (1) (2018) 637–646.
[33] S. Kataria, W. Browuer, P. Mitra, C.L. Giles, Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents. Proceedings of the AAAI, 8 (2008), pp. 1169–1174.
[34] M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Reading text in the wild with convolutional neural networks, Int. J. Comput. Vis. 116 (1) (2016) 1–20.
[35] A. Gupta, A. Vedaldi, A. Zisserman, Synthetic data for text localisation in natural images, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016), pp. 2315–2324.
[36] S. Ray Choudhury, C.L. Giles, An architecture for information extraction from figures in digital libraries, Proceedings of the Twenty Forth International Conference on World Wide Web, ACM, 2015, pp. 667–672.
[37] C. Bi, Y. Yuan, R. Zhang, Y. Xiang, Y. Wang, J. Zhang, A dynamic mode decomposition based edge detection method for art images, IEEE Photon. J. 9 (6) (2017) 1–13.
[38] C. Bi, Y. Yuan, J. Zhang, Y. Shi, Y. Xiang, Y. Wang, R. Zhang, Dynamic mode decomposition based video shot detection, IEEE Access 6 (2018) 21397–21407.
[39] L. Yang, B. Wang, R. Zhang, H. Zhou, R. Wang, Analysis on location accuracy for the binocular stereo vision system, IEEE Photon. J. 10 (1) (2018) 1–16.
[40] R.R. Nair, N. Sankaran, I. Nwogu, V. Govindaraju, Automated analysis of line plots in documents, Proceedings of the Thirteenth International Conference on Document

Analysis and Recognition (ICDAR), IEEE, 2015, pp. 796–800.
[41] W. Huang, C.L. Tan, W.K. Leow, Model-based chart image recognition, Proceedings of the International Workshop on Graphics Recognition, Springer, 2003, pp. 87–99.
[42] Y. Tang, Tf. learn: Tensorflow's high-level module for distributed machine learning, arXiv preprint arXiv: 1612.04251(2016).
[43] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, Proceedings of the Sixth International Conference on Computer Vision, IEEE, 1998, pp. 839–846.
[44] D. Moritz, Text detection in screen images with a convolutional neural network, J. Open Sour. Softw. 2 (15) (2017) 235, https://doi.org/10.21105/joss.00235.
[45] J. Redmon, Darknet: Open source neural networks in c, http://pjreddie.com/darknet 2016 (2013).
[46] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic hough transform, Pattern Recognit. 24 (4) (1991) 303–316.
[47] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man Cybern. 9 (1) (1979) 62–66.