# Classification and Text Extraction of Visualization Images

Jing Yuan
Building Science
jingyuan10@berkeley.edu

## Abstract

*Visualizations are a popular method to convey data. However, most visualizations are stored in bitmap image format, which makes information extraction difficult. In this project, we developed techniques to computationally parse these visualization images. Specifically, we extracted the chart type (via classification) and text elements. We used ResNet18 for visualization type classification, where we achieved a high validation accuracy of 98 percent across 28 chart types, with the model also showing strong generalizability to charts from other data sources. We built a pipeline for text extraction, where we first detected word bounding boxes, merged them, and then extracted the text content using optical character recognition. Furthermore, we built a LSTM-based model to classify the role of the extracted text (e.g. chart title, x-axis label, y-axis title, etc). We applied our methods to parse two large existing visualization datasets. We conclude with suggestions to improve the performance of the text role classifier and text bounding box detection.*

## 1. Introduction

Visualizations are a common way for people to present their data, and are found all over the web, academic papers, textbooks, etc [7]. While visualizations can be in various formats (e.g. Scalable Vector Graphics for charts on the web), the most common format is bitmap images [1]. Unlike charts in XML-based Scalable Vector Graphics (SVG) format, it is quite difficult to extract information from these raster visualization images, such as the chart type (e.g. scatter plot) and useful text, such as the title, axes labels, etc. However, being able to computationally parse these visualization images will enable users to annotate and index this largely available group of visualizations. This will enable more in-depth analysis of visualization trends on a large scale, and could lead to interesting applications, such as automated generation of visualizations.

We utilized the latest developments in image classification [4], object detection [12], and optical character recognition (OCR) [11] to tackle this challenging task of computationally parsing bitmap visualization images. In particular, we developed methods to extract the chart type (via classification) and all text elements, which are classified by role (e.g. chart title). We then applied our techniques to parse and annotate two large visualization datasets: Beagle [1] and Revision [9]. Specifically, we applied ResNet-18 [4] pretrained on ImageNet [3] to classify chart images by type, achieving 98 percent validation accuracy across 28 distinct chart types. This classifier also showed strong generalization capabilities to other visualization datasets. Then, we used the EAST text detector [12] to find the bounding boxes for text in chart images and merged aligned text elements. After that, we implemented Tesseract OCR [11] on the detected bounding boxes to extract the words from the chart images. Finally, to classify the extracted text element by type (e.g. chart title), we built a LSTM-based model that takes in the bounding box of the target text element, as well as the bounding boxes of all other text elements, and outputs a prediction for the text element type.

## 2. Related Work

There has been a lot of prior work on visualization extraction, classification, and visualization text extraction. Battle et. al. built a system (Beagle) [1] to extract SVG visualizations from the web and then classify the visualizations by type (e.g. bar chart). The authors then used this system to extract a large corpus of SVG visualizations from the web. However, [1] classified visualizations in SVG format, from which detailed features could be extracted. Furthermore, charts in SVG format are considerably rarer than bitmap images, as [1] were unable to find many examples by filtering random urls on the web.

[9, 7, 8, 10] classified visualizations in bitmap image format, achieving 80-90 percent accuracy. Furthermore, these studies also extracted text from the chart images. Our study is perhaps most similar to [7], as the authors classified visualization images by type and extracted text from the images; we also utilized some techniques from their text extraction pipelines. However, [9, 7, 8, 10] were all limited to at most 10 distinct chart types, while we were able to classify 28

| Dataset | Chart Types |
|---|---|
| Beagle | line, scatter, bar, heat map, box, bubble, sankey, chord, radial, area, donut, geographic map, treemap, pie, stream graph, hexabin, graph, parallel coordinates, sunburst, waffle, voronoi, word cloud, contour, filled-line |
| ReVision | area, bar, line, geographic map, pareto, pie, radar, scatter, table, venn diagram |

Table 1. All visualization types found in the Beagle and ReVision datasets.

different types with higher (98 percent) accuracy. Furthermore, their text extraction methods relied heavily on heuristics [9, 7, 8, 10], and hence require extensive feature engineering and computation, while our pipeline just took in visualization images as input and utilized the current state of the art bounding box detection [12] and OCR [11] to extract the text and also classify the text by role.

## 3. Datasets

We trained and evaluated our visualization type classifier on two large existing visualization datasets, and also applied our text extraction techniques to identify and annotate the text in those two corpora. The datasets are described in this section.

### 3.1. Beagle

The Beagle dataset [1] contains over 41,000 visualizations across 24 distinct chart types. These visualizations were collected from 5 different visualizations tools, such as D3 [2] and Plotly [5]. For each chart, the corpus contains the bitmap image, chart type, and the SVG code. All chart types in this dataset can be found in Table 1.

### 3.2. Revision

The ReVision dataset [9] has over 2,200 visualizations across 10 chart types. These visualizations were collected by querying Google image search. For each chart, the corpus contains it's bitmap image and chart type. Table 1 shows all chart types found in this corpus.

## 4. Visualization Type Classification

For visualization type classification, we combined the Beagle the ReVision datasets for a total of over 43,000 visualizations across 28 distinct chart types. We will now discuss our classification model and results.

### 4.1. Method

We used the ResNet18 [4] architecture that has been pretrained on the ImageNet [3] dataset. ImageNet contains

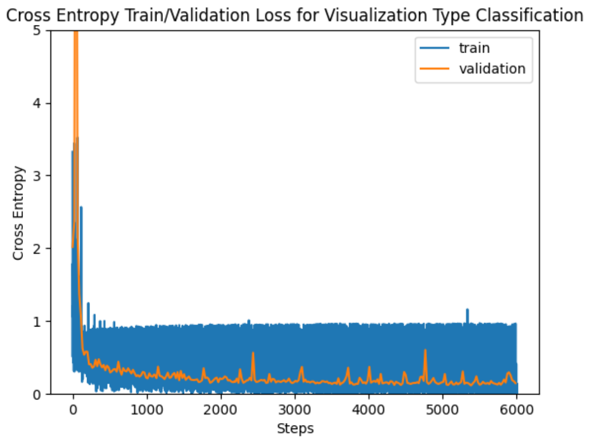

Figure 1. Training and validation plot of the chart type classifier.

| Model | Validation Accuracy | Number of chart types |
|---|---|---|
| ResNet18 (pretrained) | 98% | 28 |
| Beagle (SVG) | 85% | 24 |
| ReVision | 96%, 80% | 10, 24 |

Table 2. Validation accuracies of our model (ResNet18) compared with the models from the Beagle and ReVision studies. Note that Beagle took in detailed features computed from the visualization's SVG code, as opposed to the bitmap image.

over a million photos from 1,000 categories. To regularize the model, we added a dropout layer of probability 0.4 after the residual blocks. We split our visualization dataset into 90 percent training and 10 percent validation, and fine-tuned ResNet18 with cross entropy loss using the Adam optimizer, a learning rate of 3e-4, and a batch size of 32. We standardized all input chart images to be of size 224x224 pixels and also normalized their RGB values.

### 4.2. Results

We achieved a very high validation accuracy of 98 percent. This is higher than the validation accuracies of both classifiers from the Beagle and ReVision studies, as shown in Table 2. This is significant as [1] and [9] both had fewer categories. Furthermore, [1] took in detailed features computed from the visualization's SVG code, such as the number of SVG circle elements, whereas [9] and our model just took in the chart image as input. Figure 1 shows the training and validation curves of our model.

### 4.3. Generalizability to other Visualization Datasets

In order for our classifier to be useful, it should be able to generalize to visualizations from any dataset. To assess our model's generalizability, we trained on the Beagle dataset

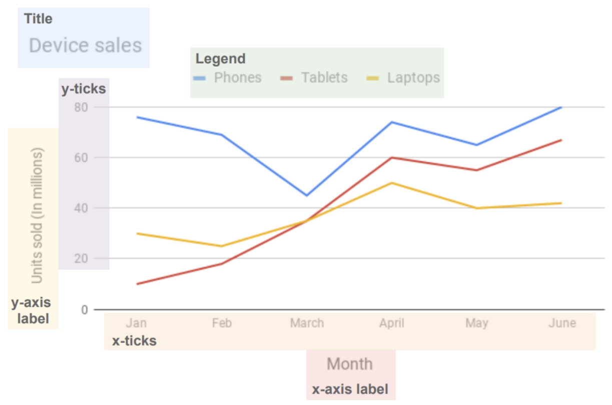

Figure 2. A line chart marked with all the text labels to be extracted: title (blue), legend (green), x-axis label (red), y-axis label (yellow), x-ticks (orange), y-ticks (purple)

only and evaluated on visualizations from ReVision (from overlapping chart types). We achieved a good test accuracy of 86.2 percent.

## 5. Text Label Extraction

The Beagle and ReVision datasets are missing text data from the visualizations. Hence, we developed methods to extract text labels from bitmap images of charts and used this methods to annotate those two corpora. We extracted the following types of text labels: chart title, legend title, legend item label, x-axis label, x-axis tick label, y-axis label, and y-axis tick label. Figure 2 shows a visualization with all these text labels marked. We followed the process from [7] for text label bounding box extraction, where we first detected word bounding boxes and then merged the word bounding boxes to obtain the bounding boxes for the text labels. We then built a classifier to identify the type of text label, and finally, utilized OCR to extract the text from the bounding box image.

### 5.1. Bounding Box Detection

We used EAST to detect the bounding box of text in the chart. EAST stands for "Efficient and Accurate Scene Text" detection pipeline. The model is a fully-convolutional neural network adapted for text detection that outputs dense per-pixel predictions of words or text lines [12]. This eliminates intermediate steps such as candidate proposal, text region formation, and word partition. The post-processing steps only include thresholding and NMS on predicted geometric shapes. We used a pre-trained EAST model for text detection. This model was initially trained to detect text in natural scenes, which means it is quite robust – capable of localizing text even when it's blurred, reflective, or partially obscured. We applied this model on our visualization data

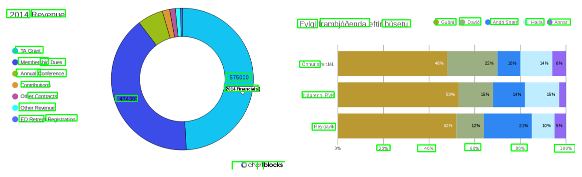

Figure 3. Example of text bounding boxes detection.(Green box represent the detected bounding box)

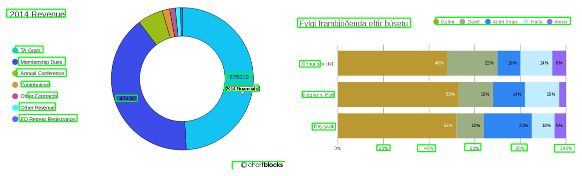

Figure 4. Example of merged bounding boxes.(Green box represent the detected bounding box)

sets, which are in bitmap format and clearer than the natural scenes, but lacking in variation of pixel values. Text box detection examples are shown in Figure 3.

### 5.2. Bounding Box Merging

However, the EAST text detector can only find the bounding box for individual words. Text labels, such as the chart title, often contain multiple words. Hence, we would need to merge the extracted word bounding boxes to obtain the bounding box for the entire text label.

In the bounding box merging process, we merged boxes that are aligned, have identical orientation, and are close in proximity. For example, we merged the words of the title, but not the axes ticks labels nor legend items. These are the rules we used for bounding box merging:

If there are two single words bounding boxes ($B_1$ and $B_2$):

1. When $B_1$ and $B_2$ are close to each other, we merged bounding boxes if $Distance(B_1 - B_2) < D$ (D was optimized based on the merging performance.)

2. When $B_1$ and $B_2$ are overlap to each other, we merged bounding boxes if $Distance(B_1 - B_2) < 0$

Then, we used the minimum x,y coordinates of the upper left points as the start point of the new bounding boxes and used the maximum x, y coordinates of the lower right points as the end point of the new bounding boxes. Examples of the merged bounding boxes are shown in Figure 2. We extracted over 383,000 text labels from the Beagle + Revision dataset.

### 5.3. Text Label Role Classification

We built a simple LSTM-based model to classify the bounding boxes of each text label by role. Since the loca-

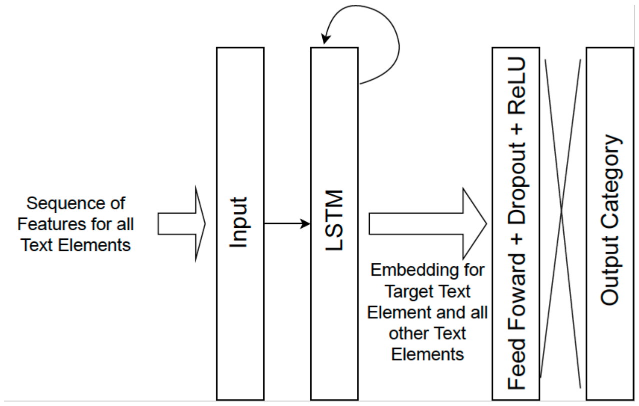
Sequence of Features for all Text Elements
Input
LSTM
Embedding for Target Text Element and all other Text Elements
Feed Foward + Dropout + ReLU
Output Category

Figure 5. Architecture for the text label classification model.

tions of other text elements provide clues to the target text element's type, the model takes in the bounding boxes of all text labels when making the prediction.

### 5.3.1 Model Architecture

The model contains a LSTM layer that takes in the sequence of features (one for each text element) with the target text element marked. After the last element, the LSTM hidden layer is used as an embedding that is input to a feed forward layer, followed by a dropout layer of probability 0.2 and a Rectified Linear layer before the final output layer. Figure 5 shows the model's architecture.

### 5.3.2 Model Features

As described earlier, the LSTM encoder takes in a sequence of feature vectors, one for each text element bounding box. The text elements in the sequence are sorted by visual scanning order (top-down then left-right) of their top left corner. For text element $b_i$, the feature vector is as follows:

$$b_i = [x_1, y_1, x_2, y_2, target]$$

where $x_1$, $y_1$ are the x,y coordinates of the text bounding box's top left corner and $x_2$, $y_2$ are the coordinates of the bottom right corner. $target$ is 1 if $b_i$ is the target text element to be classified, and is 0 otherwise.

### 5.3.3 Results

We trained and evaluated our model on the dataset from [7], which had over 113,000 text elements from over 5,000 visualizations. We again, used cross entropy loss with the Adam optimizer, a learning rate of 3e-4, and a batch size of 32. With this setup, we achieved a validation accuracy of 80 percent. Figure 6 shows the training and validation curves.

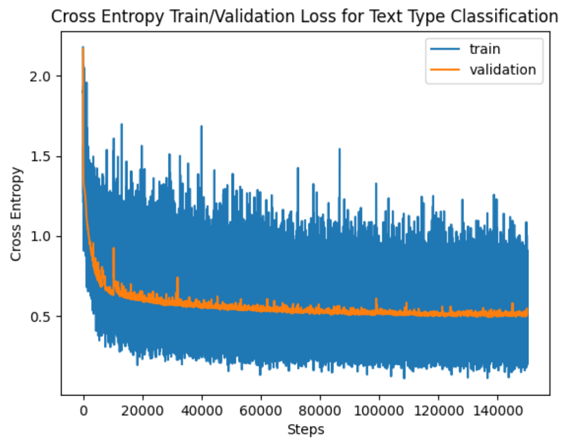

Figure 6. Training and validation plot of the text type classifier.

## 5.4. Text Extraction with OCR

Finally, we perform optical character recognition for each merged candidate text label using the open source Tesseract [11] engine. Initially, the capability of the Tesseract was mostly limited to structured text data. More recently, deep learning based method performs better for unstructured data. Tesseract 4 incorporated deep-learning based capability with LSTM network Recurrent Neural Network based OCR engine. Various languages were also included in the training data. Here we just assumed texts are in English to evaluate its performance. It is worth noting that in general, in order to get better OCR results, it is recommended to have images with high resolution, horizontal text, high contrast and noiseless. However, real images are usually compressed and bounding boxes of the texts are small and sometimes overlapped. Besides, the orientations of the texts are varied typically. Thus, we performed some preprocessing steps.

Firstly, we cropped a rectangle of the bounding box from the image. Then we turned the three RGB channels into grey scale and upscale it by factor 3. Then we applied binarization to the images with a global threshold approach. The Otsu's method [6] is used to calculate the optimum threshold, which assumes all pixels belong to two classes and pick threshold to maximize the inter-class variance. To deal with texts with different orientations, we ran Tesseract multiple times for rotated images with angles including 0°, 90°and -90°. We chose the right orientation and recognized text strings based on the confidence scores returned by Tesseract. Higher confidence scores are preferred.

## 5.5. Results

We ran this pipeline to extract and annotate text elements from the Beagle + Revision dataset. The results were not

| Text Element Type | Percent |
|---|---|
| x-axis tick | 29.3 |
| x-axis label | 7.76 |
| y-axis tick | 56.2 |
| y-axis label | 0.18 |
| legend label | 0.79 |
| legend title | 0.061 |
| text label | 1.19 |
| chart title | 4.46 |

Table 3. Distribution of predicted text types in the Beagle + ReVision dataset.

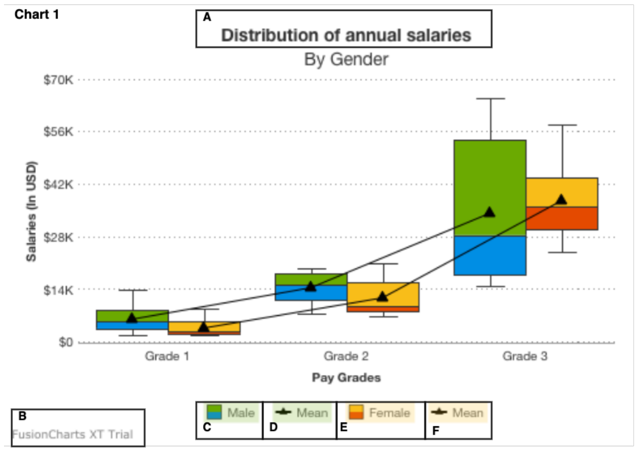

Figure 7. Some example outputs of the text extraction pipeline applied to the Beagle + ReVision dataset. The extracted text (OCR) and type classification are as follows:
A (text: ”Distribution of annual salaries”, type: chart title), B (text: ”FusionCharts XT Trials”, type: x-axis label), C (text: ”Male”, type: legend label), D (text: ”Mean”, type: legend label), E (text: ”Female”, type: legend label), F (text: ”Mean”, type: legend label)

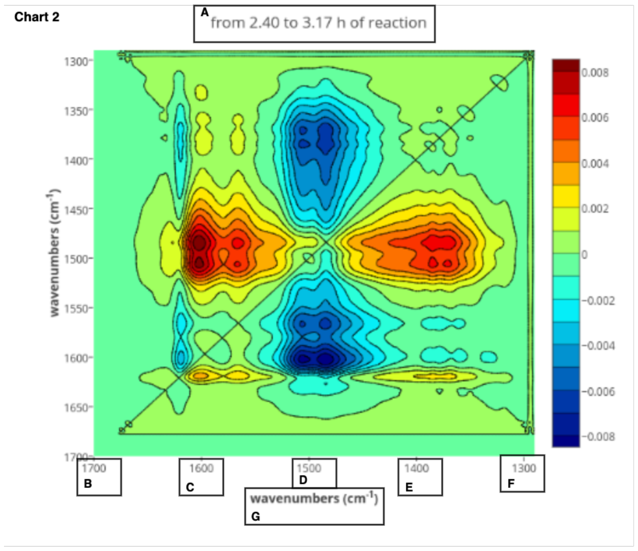

Figure 8. An example output of the text extraction pipeline applied to the Beagle + ReVision dataset. The extracted text (OCR) and type classification are as follows:
A (text: ”from 2.40 to 3.17 h of reaction”, type: y-axis label), B (text: ”1700”, type: x-axis tick), C (text: ”1600”, type: x-axis tick), D (text: ”1500”, type: x-axis tick), E (text: ”1400”, type: x-axis tick), F (text: ”1300”, type: x-axis tick), G (text: ”wavenumbers (cm”)”, type: x-axis label)

as good as expected, with some results shown in Figures 7-9. In Figures 7-9, each detected text element is marked in the figure, and the OCR extracted text and the text type classification are included in caption. Table 3 shows the distribution of the predicted text types. We notice a large discrepancy between the percentages of x-axis and y-axis labels, where they are expected to be similar. A possible explanation for this is that the Fusion Charts logo (”FusionCharts XT Trial”) is often misclassified as an x-axis label, as shown in Figure 7. Also, there is an extremely small fraction of classified legend titles, with true legend titles being misclassified as shown in Figure 9. This is likely due to the fact that there were only two examples of legend titles in the training set from [7]. More significant causes of poor performance and ideas for improvement are discussed in the next section.

## 6. Discussion and Future Work

There are many implications of our study. The superior performance of ResNet18 on visualization type classification compared to other state of the art classifiers [1, 9, 7, 8, 10] indicates that it is a suitable model choice for type classification of bitmap visualization images. Furthermore, ResNet’s high accuracy when evaluated on ReVision when trained only on the Beagle dataset shows that the model is able to generalize to new visualization images from other data sources. This suggests that when trained on a sufficiently large dataset with examples from a vast array of chart types, the model would be able to classify any input chart image with reasonable accuracy.

For the chart text role classifier, a significant reason for it’s subpar performance on the Beagle + ReVision dataset is that the training set was small and only contained four distinct chart types [7]. The training set likely lacked examples where there were symbols in the chart title, a search bar (Figure 9), and a color gradient (Figure 8), which resulted in misclassification of those text elements. Furthermore, the training set only had four basic chart types, and did not have any examples for most of the 28 chart types in the Beagle + ReVision dataset, such as heatmaps (Figure 8) and venn diagrams (Figure 9). This likely led to the model’s poor performance on those two examples of unseen chart types. Future work to improve the model’s accuracy include training

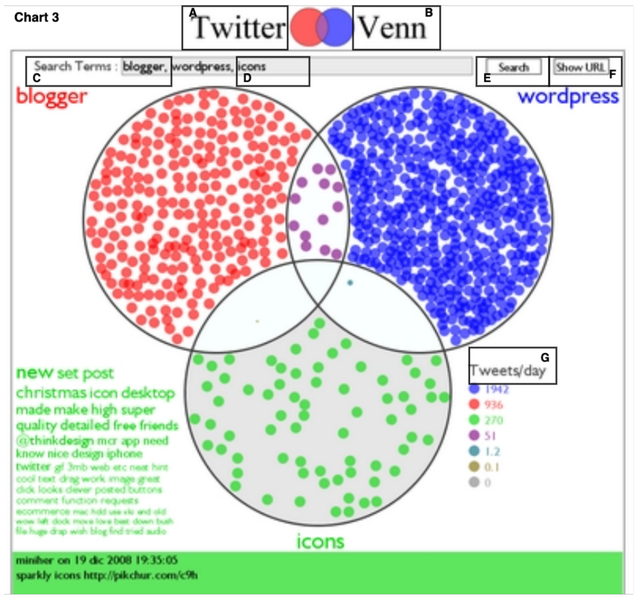

Figure 9. An example output of the text extraction pipeline applied to the Beagle + ReVision dataset. The extracted text (OCR) and type classification are as follows:
A (text: ”Twitter”, type: y-axis label), B (text: ”Venn”, type: y-axis label), C (text: ”Search Terms: blogger”, type: legend label), D (text: ”Icons”, type: y-axis tick), E (text: ”Search”, type: y-axis tick), F (text: ”Show URL”, type: y-axis tick), G (text: ”Tweets/day”, type: y-axis tick)

on a much larger dataset with examples of those aforementioned edge cases and with significantly more chart types. Furthermore, features of the text itself, such as word2vec embeddings or whether the text is a number or word, could be input to the classifier to help determine the text element’s role.

In bounding box detection, we found that EAST can not find the bounding box for languages other than English (Figure 10). And EAST text detectors also failed to find the text which are rotated so that they are not vertically nor horizontally aligned with the chart edge; i.e. the detected bounding boxes were parallel to the edge of the chart. This might be because the rotated word lacks dimensional information as the real world texts. In reality, when we see texts on the side, the farther the alphabet the smaller it is. However, in bitmaps, we just rotate it without changing the size of alphabets. These shortcomings made our text label type classification even harder, because we do not have a good quality of input data due to the inaccurate text area detection. For the next step, we plan to annotate part of the dataset by manually finding the text bounding box and fine-tune the EAST model. After that, the trained model will be more accurate on detecting the texts in visualizations.

In our workflow, we decided to merge the text bounding boxes before OCR. This decision led to failures when

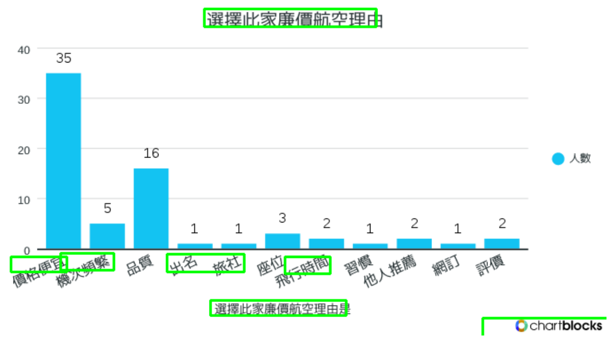

Figure 10. An example of text bounding boxes detection failure when detecting languages other than English and has rotated texts in the chart. (Green box represent the detected bounding box)

merging bounding boxes for vertical lines or lines that were not horizontal orientated. In the future, if we could get the orientation of the text through OCR first, it will help us group the aligned texts and improve the accuracy of bounding box merging. For example, we would know whether the texts are horizontally aligned or vertically aligned, or rotated with the same angles. Then, based on the text orientation, we could find the texts with the same orientation at the beginning and then decide whether we should merge the bounding box or not based on the distance between boxes.

For the optical character recognition, we could fix some common errors from some simple logic. For examples, the confusions between ’O’ and ’0’ also ’l’ and ’1’ could be corrected by the judgement of the type of surrounding characters. And the boundary noise should be considered since some bounding boxes is small and only partial character is included. A contrary problem is that parts of the characters from other words are contained. A simple improvement can be achieved if text strings with low confidence scores are excluded. Lastly, more language datasets or packages are supposed to be added to accommodate the various languages in the chart images.

## References

[1] 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*, CHI ’18, page 1–8, New York, NY, USA, 2018. Association for Computing Machinery.

[2] M. Bostock. D3.js - data-driven documents, 2012.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[5] P. T. Inc. Collaborative data science.

[6] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[7] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Comput. Graph. Forum*, 36(3):353–363, June 2017.

[8] V. S. N. Prasad, B. Siddiquie, J. Golbeck, and L. S. Davis. Classifying computer generated charts. In *2007 International Workshop on Content-Based Multimedia Indexing*, pages 85–92, 2007.

[9] 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*, UIST '11, page 393–402, New York, NY, USA, 2011. Association for Computing Machinery.

[10] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi. Figureseer: Parsing result-figures in research papers. In *ECCV*, 2016.

[11] R. Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, 2007.

[12] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: an efficient and accurate scene text detector. *CoRR*, abs/1704.03155, 2017.