Practical Work Report: Automatic Extraction of Information from Visualization Images

Fabio Pöschko
JKU
Linz, Austria
fabio.poeschko@gmail.com

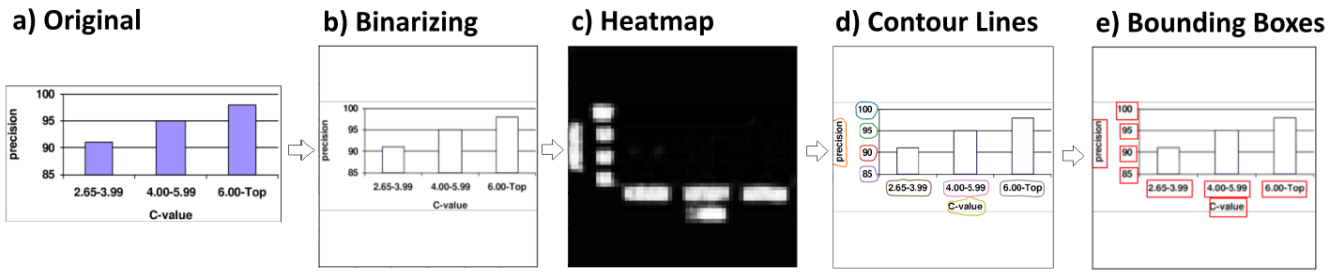

Figure 1: Visualisation of the text detection pipeline

ABSTRACT

Automatically extracting information from chart images is a useful research topic, for analysing user behavior of visualisation tools. Existing approaches work theoretically, but are written in old and outdated programming frameworks, therefore this practical work project aims to recreate these existing approaches, make them more modern, and more accurate. In this report, I propose similar approach, using a heatmap for text localisation and recovering the text with optical character recognition. This means, that with further work it is possible to fully recover visual encodings from chart images.

1 INTRODUCTION

Understanding the analytical strategies users who work with visualization tools has been an active research topic. Deriving those strategies is still a challenging research topic. Current approaches primarily focus on logging interactions, such as click events with the system, to save this workflow, but often this type of raw data is hard to access. Therefore this project aims to semi automatically extract visual encodings from chart images. The primary contribution of this work is a way to semi automatic way to extract textual information from chart images, using a heatmap for text localisation and recovering the text with optical character recognition. Together with further work this could be used to understand the analytical reasoning process of users who work with interactive visualisation tools.

2 RELATED WORK

To recover visual encodings from chart images, the first step is to remove non relevant information from the image, for this purpose it is useful to classify text and not text in an image, therefore this work builds upon prior research in the area of text localisation and classification

Text localisation has been studied extensively, in recent years, however methods that work for normal images are not accurate at localising text in chart images. Older text localisation methods usually follow an approach, like the one Huang and Tan[1] use, they separate text and graphical elements using a bottom-up, region-based approach, find connected components and merge them according to fixed rules like proximity.

A newer and more modern approach was done by Moritz [3], he explains how to classify text pixels in scientific visualisations. To do this the author uses a CNN in the Darknet [7] which takes as input the visualisation and outputs a heatmap of text pixels.

Poco and Heer [6] then used the approach from Moritz to present a way to decode a bitmap image into its visual encoding specifications, by first localising the text and using optical character recognition (OCR), then classifying the role of this text and classifying the mark type of the chart using a Convolutional Neural Networks (CNNs), OCR programs and special rules. With their method they already achieve good results, reporting an average F1-scores of 88% in text localisation, 99% in text role classification and 94% in mark type classification.

3 METHODS

As the aim of this practical work was to reverse engineer bitmaps of visualisations, I performed, research to find the best method of reverse-engineer bitmaps of visualizations. Basing the work on Poco and Heer [6], their Github repository served as a good starting point. However, some code files were missing. Consequently, to recreate their project with up-to-date programming languages and state-of-the art libraries, I decided to use python 3 instead of 2.

3.1 Preprocessing

To standardize the visualisations, the method starts by padding the original bitmap of the visualisation, such that it is a square, then resizing the image to 1200x1200 pixels and binarizing the images with a global threshold approach, the optimal threshold is calculated using Otsu's [4] method, which assumes that pixels belong to two classes and attempts to maximize the inter-class variance.

3.2 Heatmap

The heatmap is generated using a CNN which is modeled after the one Moritz [3] used in his article. This heatmap displays where the text is in the visualisation, as seen in Figure 1c. The original project used a CNN which was trained with the Darknet [7] which i thought was impractical to use, because the installation for operating systems other than linux needs a special github repository and other installation steps, so i downloaded the acl anthology part of the training data from the "Amazon Web Services S3 Bucket" [2], and trained a similar CNN with pytorch [5], there would be a lot more training data from arXiv, but this is so much data that the free storage of my computer would not be enough to store it.
I got the network architecture by looking at the file with the network architecture in the domoritz label generator [3] github repository, i only had to do some small adjustments, like leaving out the croping layer and implementing the ramp activation function, because pytorch [5] did not offer it.

3.3 Contour Lines

Contour lines are used to draw circles around the pixels where text might be, in Figure 1d contour lines are shown around text elements such as the x-axis title ("C-value") and y-axis title ("precision") and their respective axis labels. I used contour lines because, it provides an easy way to separate areas with and without text.

3.4 Bounding Boxes

Bounding boxes are used to separate different text elements, therefore the max distance of the circle pixels are used as boundaries boxes, an example can be seen in Figure 1e. With these on these boundary boxes I then use pytesseract a python wraper for tesseract [8], an optical character recognition module, to give me the text in the box, because the text sometimes has a different orientation I rotate the box and then take the text in which pytesseract is most confident in, with their confidence score.

4 DOCUMENTATION OF THE CODE

All the code I used for the project can be found on my github repository at https://github.com/FabioPoe/PW. The actual method can be found in the file final_pipeline.ipynb, as functions. The files data_preprocessing.py, domoritz.ipynb and domoritz_colab.ipynb were used to train the CNN to generate the heatmaps used for text detection. The file domoritz_colab.ipynb was made, because I realized, that a CNN would probably be way faster on a GPU than my normal CPU, and since I do not have a good GPU in my PC, I decided to use google colab for my training. With their GPU the model trained a lot faster.

5 RESULTS

I was successful in localizing text in bitmap images of visualisations, and recognizing the characters in the recognized text areas. I have tried the approach on 20 different images and compared the ground truth, the approach only worked great in about 50% of the cases.

The Method still has some flaws, which are problems with text localisation if the labels are tightly spaced and if someone uses for example "O" as a plotting symbol it is recognized as text. These flaws are also listed as flaws in the paper by Poco and Heer [6], there are some examples of these mistakes in Figure 2.

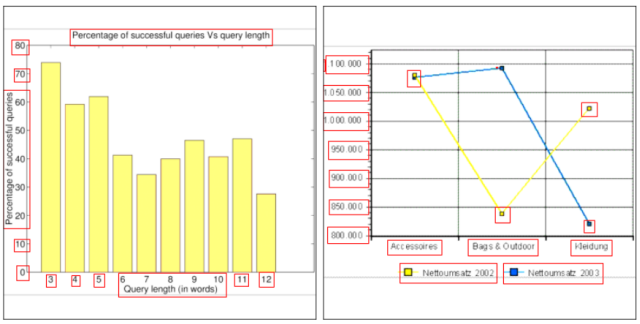

Figure 2: Common problems, on the left there are merged bounding boxes, and on the right are plotting symbols which get identified as characters

6 CONCLUSION

This report has discussed the development of a text localisation approach. The objectives of this project were to develop a way to semi automatically extract information from images of visualisations. Text localisation and recognition was done, using binarization, a CNN to generate a heatmap of text and non-text pixels, Contour Lines, Bounding Boxes, and OCR.

As future work, the model can be modified to improve the two big flaws the method still has, which are problems with text localisation if the labels are tightly spaced and if someone uses a letter as a plotting symbol it is recognized as text. Another possibility of future work would be to develop other models to classify the role of the extracted text, and the mark type of the chart.

REFERENCES

[1] Weihua Huang and Chew Lim Tan. 2007. A system for understanding imaged infographics and its applications. 9–18. https://doi.org/10.1145/1284420.1284427
[2] Dominik Moritz. 2017. Dataset extracted figures and labels. Downloaded in December 2022 from amazon S3 bucket s3://escience.washington.edu.viziometrics.
[3] Dominik Moritz. 2017. Text detection in screen images with a Convolutional Neural Network. *The Journal of Open Source Software* 2, 15 (July 2017), 235. https://doi.org/10.21105/joss.00235
[4] Nobuyuki Otsu. 1979. A Threshold Selection Method from Gray-level Histograms. IEEE Transactions on Systems, Man and Cybernetics, 5.
[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[6] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. EuroVis, Washington, 11.
[7] Joseph Redmon. 2013–2016. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/.
[8] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633. https://doi.org/10.1109/ICDAR.2007.4376991