PROJECT REPORT

INTRODUCTION

Recognition of tables consists of two main tasks, namely table detection and table structure recognition. This project has been built over a multistage pipeline named Multi-TypeTD-TSR (SOTA). It utilises state-of-the-art deep learning models for table detection and differentiates between 3 different types of tables based on the tables' borders. For the table structure recognition they have used a deterministic non-data driven algorithm, which works on all table types. In this project we have worked on the latter stage of Table Structure Recognition to improve it further by attempting to solve problems such as column spanning, colour invariance etc. and include a data driven approach.

Tables are used to structure information into rows and columns to compactly visualise multidimensional relationships between information units. In order to convert an image of a table, i.e. a scanned document that contains a table into machine readable characters, it is important to structure the information in such a way that the original relationships between the information units and their semantics is preserved. Developing algorithms that can handle such conversion tasks is a major challenge, since the appearance and layout of tables can vary widely and depends very much on the style of the table author. The already mentioned row and column structure of tables goes hand in hand with different sizes and layouts of table elements, changing background colours and fonts per cell, row or column and changing borders of the table as a whole or its individual entries.

Table Detection

ResNeXt-152 based architecture is used to detect bounding boxes for tables.

Colour Invariance

The further TSR algorithm requires a white background and a black font. Thus for the pipeline to be robust and inclusive we need to deal with dark background images.
The algorithm proposed in the paper isn't robust and fails a lot of times when applied to images with dark backgrounds. We propose a new algorithm. We can use the information about pixel values of text and background. When considering histogram of pixel values we'll find two peaks one for background colour another for text value. The peak for the background pixel values will be

much higher than text pixels. This information can now be used to find a split threshold to binarize the image.

Partially Bordered TSR

The algorithm proposed was not able to detect column spanning or row spanning. We have built upon this algorithm and further improved it to address the issue of column spanning. We first use operations such as erosion and dilution over binarized images to identify any borders present and remove them from the original table. We first identify all the rows present in the image by the help of contours. Once all the rows are identified we apply a similar procedure to find columns. Now since we are finding columns for an individual row and not the whole table, the chances of false cases(extra columns) increase, owing to the fact that there might be unnecessary space within a column which the algorithm misinterprets and splits into two columns. To rectify this we can make use of OCR algorithms, to get bounding boxes for text identified in each row (in particular we have used EasyOCR). And now we can merge these two different sets of bounding boxes.

Web App

A basic flask app where users can upload images, to identify tables which can be later converted to HTML format.

Further Developments

The algorithm proposed above fails to identify empty cells, this can be further worked upon. One potential solution is to make use of table statistics and bounding boxes.

The algorithm hasn’t been optimised for row spanning.