Page segmentation

In common with most other document understanding tasks, we first need to separate the content into atomic blocks, each of which somehow represent one logical unit in the document's structure.

Whilst many document analysis methods perform segmentation on the bitmap level, we choose to cluster together segments on the object level, as they are represented on the PDF.

We start by merging text characters horizontally to form lines, and then merge these lines vertically to form paragraphs, table cells or other atomic blocks. The problem is, if we were to just use a threshold distance, certain lines would be over- or under-segmented, as shown below.

The human reader is able to determine the most likely segmentation of a page by considering the structure at different levels of granularity simultaneously. We choose to under-segment at this stage as, due to the principle of rectangular containment, lines belonging to a paragraph (such as in Fig. 1) can easily be found later. A special heuristic is used to take care of cases such as the one illustrated in Fig. 2.

Rectangular containment

An underlying principle in our algorithms is that, for the vast majority of documents, all items on the page, in particular tables and their constituent rows and columns, have a rectangular or nearly rectangular shape. This allows us to make use of the procedure of rectangular containment expansion when looking for tables.

As an example, let's imagine that we have a candidate table that contains two objects. We find the smallest bounding rectangle, or bounding box, of this object group, and therefore of the candidate table. We then find that this bounding box intersects other objects on the page, so we proceed to add them to our candidate table, and find its new, enlarged bounding box again. We repeat this procedure until the bounding box is not grown any more, and therefore no more items are added.

Our table-finding process

First, we look for candidate columns (Fig. 3) by looking for connected components of vertically adjacent text lines. We join these lines only when they are close enough, of similar width, and do not split as in the case of a spanning column. A special heuristic is used to take care of cases such as the one illustrated in Fig. 2.

For non-ruled and horizontally ruled tables, we aim to use rectangular containment expansion to cluster the individual candidate columns into complete tables (Fig. 4). If the table is determined to be horizontally ruled, we assume that the horizontal rules span the complete width of the table. If no horizontal rules are detected, we attempt to add neighbouring columns left and right, until the table no longer passes a heuristic validity check.

For tables with horizontal and vertical ruling lines, this grid structure is usually sufficient to detect the boundary of the table. However, we still examine the candidate columns to look for sub-columns that may not be separated by ruling lines.

Finally, we detect the row boundaries and output the table in HTML (Fig. 5).