影像表格文字化的實作與學習

作者：薛詠謙

1、 動機

在尋找合適校系的過程中，我發現 University TW 所提供的分數搜尋功能非常實用。不過，該網站的資料更新速度並不一定與官方同步，而 **大學甄選入學委員會** 所公布的錄取分數資料又多以 **圖片表格** 的形式呈現，難以直接整理或分析。

因此，我產生了想要將影像表格轉換成可數位化利用文字資料的想法，因此，我開始學習 **OCR**（光學文字辨識）技術，並嘗試實現影像中文字資料的自動化擷取與轉換。

2、 資料蒐集

根據 [1] 與 [2] ，影像表格分析流程可簡略分為以下幾個步驟：

1. **影像前處理**：對原始影像進行去噪、二值化或對比度等調整。
2. **表格分割**：偵測並標註表格的行列結構，將整體表格拆分為單元格。
3. **文字辨識**：對每個單元格進行 OCR 辨識，取得對應文字內容。
4. **轉換輸出格式**：將最終文字結果整理成結構化資料，如 `csv` 或 `json` 。

2.1 影像預處理

2.1.1 二值化

`threshold(thresh: f64)` 函式可將灰階影像轉換為純黑與純白的二值化影像。 然而，由於中文字筆劃粗細不一，閾值 (`thresh`) 的設定若不恰當，可能會降低文字辨識的準確度。

		
圖 1: 閾值=100	圖 2: 閾值=150	圖 3: 閾值=200

因此，雖然在此範例中使用 圖 2（閾值 = 150）的二值化處理結果尚可接受， 但鑒於本專案涉及多種不同文字樣式，最終我決定在文字辨識的預處理階段不採用二值化。

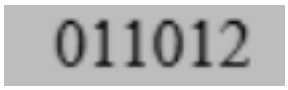
2.2 文字辨識

2.2.1 `Tesseract`

將中文文字影像直接輸入 `tesseract` 進行辨識時，即使經過灰階化等前處理，辨識效果仍不盡理想。推測原因可能在於大考中心提供的檔案 **解析度** 過低[3]，導致模型無法正確辨識文字。

`tesseract figure-1.png - -l chi_tra`
無法 辨識出文字

不過，`tesseract` 在數字的辨識上表現十分精準，能穩定且無誤地擷取出校系代碼欄位的資料。



```
tesseract figure-2.png - -c tessedit_char_whitelist=0123456789
```

成功辨識出 011012

2.2.2 PaddleOCR

PaddleOCR 是由百度開發的文字辨識工具，相較於 `tesseract`，在中文文字的擷取上具有顯著的提升。

在大多數情況下，PaddleOCR 能夠成功辨識中文文字，其特色包括：

- 少數情況下可能漏字，例如在 圖 4 中僅辨識出「(華語文教學組)」。
- 純數字辨識有時也會漏字，如在 圖 5 中僅辨識出「1012」。
- 對中英文混合及標點符號具有良好容錯性，大部分情況下都能正確辨識，例如在 圖 6 成功辨識出「(英文+數學 A)25」。

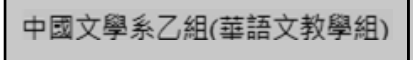
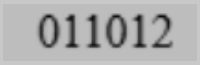
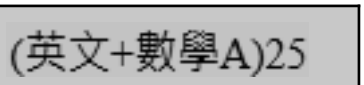
		

圖 4:	圖 5:	圖 6:

2.2.3 小結

綜上所述，將 `tesseract` 用於單元格的純數字辨識，而將 PaddleOCR 作為主要文字辨識工具，似乎為最佳組合。

3、 操作方法

根據 圖 7 所示，大考中心提供的篩選標準圖表可能包含多個區塊。 因此，需要先辨識各區域的 **外圍邊線** 並進行分割，再辨識表格內的 **分割線** ，將表格切分成單元格，最後進行 **文字辨識**，並將結果整理成結構化的 JSON 格式。

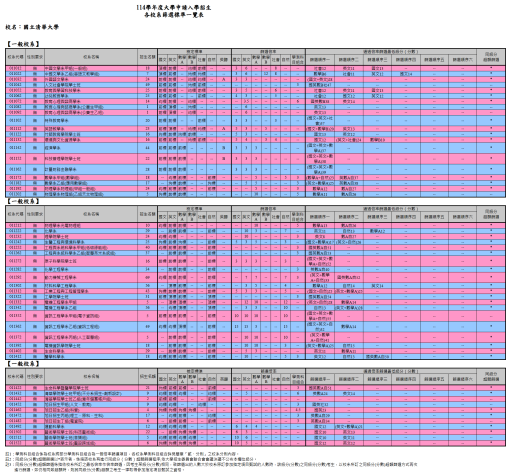


圖 7:

處理流程可簡述如下：

外圍邊線辨識 → 區域分割 → 表格邊線辨識 → 單元格分割 → 文字辨識 → 將資料結構化

3.1 外圍邊線辨識與分割

根據 [1]，可使用`find_contours_with_hierarchy`函數進行直線邊緣偵測。 然而，此方法同時會偵測到文字筆劃中的直線，導致難以準確判定外框邊界。

為消除文字干擾，可採用**膨脹與腐蝕（morphological closing）**處理。

```
let kernel =
    get_structuring_element(imgproc::MORPH_RECT,
Size::new(20, 5), Point::new(-1, -1))?;

let mut closed = Mat::default();

morphology_ex(
    &thresh,
    &mut closed,
    imgproc::MORPH_CLOSE,
    &kernel,
    Point::new(-1, -1),
    1,
    BORDER_CONSTANT,
    Scalar::default(),
)?;
```

表 1: 其中`thresh`為經過二值化的影像

由 表 1 和轉換過後的 圖 8 可見，以長 20、寬 5 的矩形結構元素（kernel）進行運算，能有效的腐蝕掉文字，不過同時也腐蝕掉了表格。 為保留表格結構，可將處理後的影像與原始影像進行`bitwise_or`運算，以刪除文字、復原表格，如 圖 9 所見。

圖 8: 經過腐蝕過後的圖像

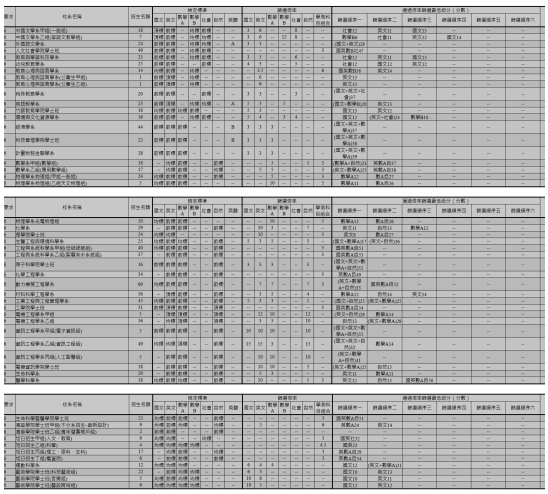


圖 9: 經 bitwise_or 合併的圖像

3.2 表格邊線辨識

3.3 單元格分割

3.4 資料結構化

參考文獻

[1] 蔡桓銘, “用 Tesseract 結合 LSTM 模型實作手填表格辨識,” 2021. doi: 10.6846/TKU.2021.00596.

[2] Johnny Chang, “使用 python 萃取掃描文件中的表格(一)切豆腐篇.” [Online]. Available: https://between2058.medium.com/%E4%BD%BF%E7%94%A8python%E8%90%83%E5%8F%96%E6%8E%83%E6%8F%8F%E6%96%87%E4%BB%B6%E4%B8%AD%E7%9A%84%E8%A1%A8%E6%A0%BC-%E4%B8%80-%E5%88%87%E8%B1%86%E8%85%90%E7%AF%87-d5b65b7ec320

[3] Willus Dotkom, “Optimal image resolution (dpi/ppi) for Tesseract 4.0.0 and eng.traineddata?.” [Online]. Available: https://groups.google.com/g/tesseract-ocr/c/Wdh_JJwnw94/m/24JHDYQbBQAJ?pli=1