Table retrieval project report [CM-UCL I]

Contentmine

2017-05-01

- Corpus
- Metrics
    - Table alignment
    - Table structure and cell integrity
    - Normalized discrepancies
- Documentation

The current project aims to extract information from scholarly tables (semi-)automatically. Scholarly articles frequently contain vital information/statistics in tabular format, but given that the PDF format is a visually oriented tool instead of also being machine-readable, the information from these tables is not readily exportable. For example, simply copy-pasting a table into a spreadsheet is virtually impossible. This greatly decreases the efficiency of data collection in systematic reviews. This document is the final report for the project that ran from circa March 2017 through April 2017, contracted by the EPPI-centre (https://www.ucl.ac.uk/public-policy/spotlight/eppi-centre).

We developed software for table extraction in two stages: (1) testing and (2) validation. Throughout this report the results of both will be presented alongside each other in order to improve comparison of the software's performance. In this report we do not present the results of the automated table identification procedure, but these can be found here (https://github.com/ContentMine/cm-ucl/blob/master/interim-reports/grobid-table-retrieval.pdf). Additionally, all individual tables and the extracted results can be inspected via the project's webpage (https://contentmine.github.io/cm-ucl).

The current report showcases some of the vital metrics of the resulting software's performance. However, in no way is this meant to be exhaustive of the data collected. The data are freely available (https://github.com/ContentMine/cm-ucl/tree/master/data) for reanalysis and the code behind this report (https://github.com/ContentMine/cm-ucl/blob/master/final-report.Rmd) as well. Additionally, the steps taken to come to this end-product are more extensive than a summary report can contain; more extensive technical details are available on the project's Open Notebook (http://discuss.contentmine.org/t/ami-eppi-cm-ucl-table-extraction-project/322/47) and Github repository (https://github.com/contentmine/cm-ucl). However, documentation is available at the end of the report.

Corpus

After coordination with EPPI, we selected Open Access papers with tables from a selection of 26 journals (https://github.com/ContentMine/cm-ucl/blob/master/data/test-pubs-jrnls.csv). Figure 1 depicts the flowchart of the table extraction from both the test and validation corpus. The raw corpora are available on Github (test set (https://github.com/ContentMine/cm-ucl/tree/master/corpus-oa), validation set (https://github.com/ContentMine/cm-ucl/tree/master/corpus-oa-validation)).

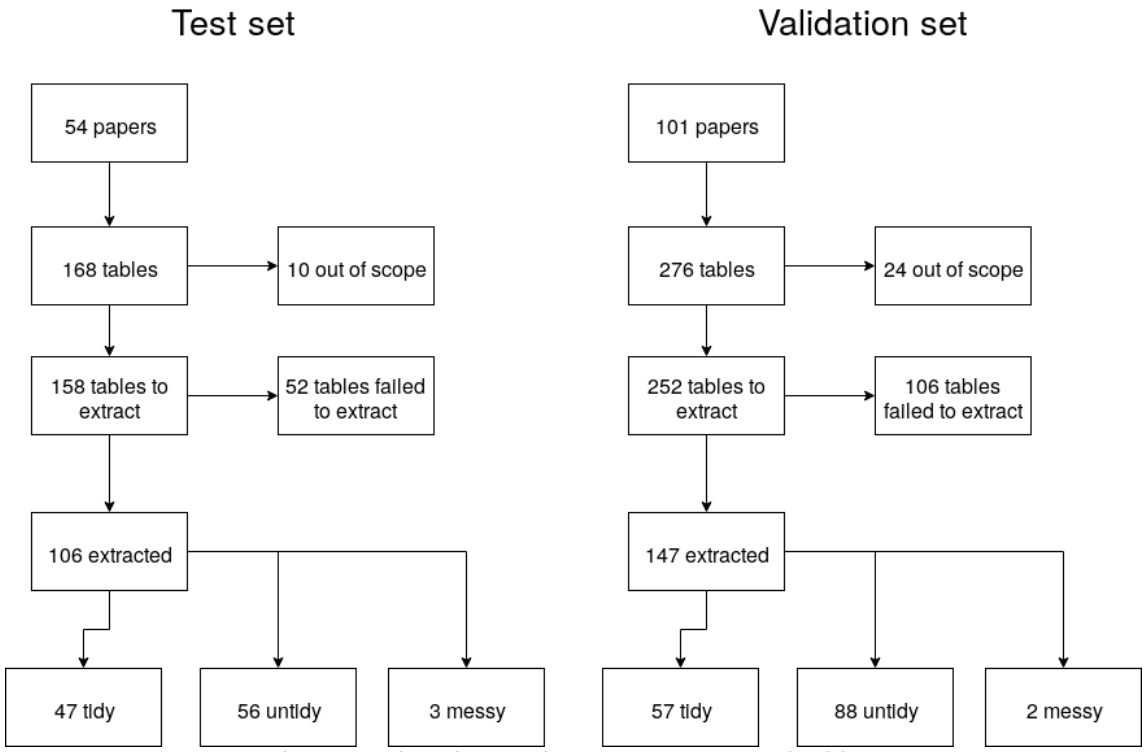

Figure 1. Flowcharts of corpus to extracted tables.

Metrics

Each table has certain aspects that can be taken into account for retrieval rates. In this report we focus on the following for the tables we were able to extract (see Figure 1):

1. Are the body and header sections of a table aligned?
2. Is the row and column structure of the table correctly retrieved?
3. How many discrepancies in the cells of extracted table are there?

For these metrics, tables were classified into three categories: (1) messy, (2) untidy, and (3) tidy. “Messy” includes tables that contain subtables (for example), whereas tidy are simple matrix-like tables (*i* rows and *j* columns); untidy tables include nested columns/rows but no messy structuring beyond that.

Table alignment

The tables below indicate the proportion of bodies and headers that were extracted from the respective sets of tables in an aligned manner. For the test and validation set, the results are fairly similar. Unexpectedly, the alignment in the tidy tables is less than in the untidy tables.

Table 1. Automated alignment of table body and header for both the test and validation sets

	test; not aligned	test; aligned	validation; not aligned	validation; aligned
messy	0.000	0.028	0.000	0.007
untidy	0.113	0.415	0.141	0.465
tidy	0.094	0.349	0.077	0.303

Table structure and cell integrity

The tables below indicate the proportion of tables that contain several categories of errors. The bottom-right hand cell is the one that denotes perfectly retrieved tables (i.e., perfectly maintains both table row/column structure and cell integrity). This amounts ~30% in both the test and validation set. Perfect might be too strict given that the nesting of untidy tables was out of scope; relaxing it to one structure error or a cell error raises the proportion of tables to 0.584 for the test set and 0.501 for the validation set.

Table 2. Table structure retrieval for test set, based on rows and cell contents.

	>=4 cell discrepancies	**1<x<4 cell discrepancies**	**1 cell discrepancy**	**no cell discrepancies**
>=4 row/column discrepancies	0.198	0.019	0.009	0.028
1<x<4 row/column discrepancies	0.028	0.075	0.000	0.028
1 row/column discrepancy	0.009	0.019	0.066	0.198
no row/column discrepancies	0.000	0.000	0.009	0.311

Table 3. Table structure retrieval for validation set, based on rows and cell contents.

	>=4 cell discrepancies	**1<x<4 cell discrepancies**	**1 cell discrepancy**	**no cell discrepancies**
>=4 row/column discrepancies	0.296	0.007	0.000	0.028
1<x<4 row/column discrepancies	0.028	0.042	0.000	0.028
1 row/column discrepancy	0.014	0.056	0.085	0.085
no row/column discrepancies	0.000	0.000	0.021	0.310

The tables can be divided in to four quadrants, where we see a logical clustering of structure and cell integrity. Few structure and cell errors (lower right-hand) cluster, and multiple structure and cell errors cluster as well. As would be expected, the software doesn't create large discrepancies in the structure but not in the cells — the co-occurence is a sign the software does not systematically act up. Some tables might contain more errors in both the structure and the cells, but that indicates that there is ground for improvement in the coverage of the software (as in UCL-II).

Normalized discrepancies

Not each table is of the same size or complexity, hence, one could argue that one error is worse in a 3x3 table than in a 42x42 table. Considering this, we computed normalized discrepancy scores (i.e., number of cell discrepancies divided by the amount of cells). As a result, we can directly compare the values across tables of

variable complexity. Figure 2 indicates that this is a severely skewed distribution, as would be expected, and that most tables fall within a 0-20% range of discrepancies, with a few (far) above those. The larger discrepancies can be investigated further to determine optimal ways to improve the software in the future.

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```

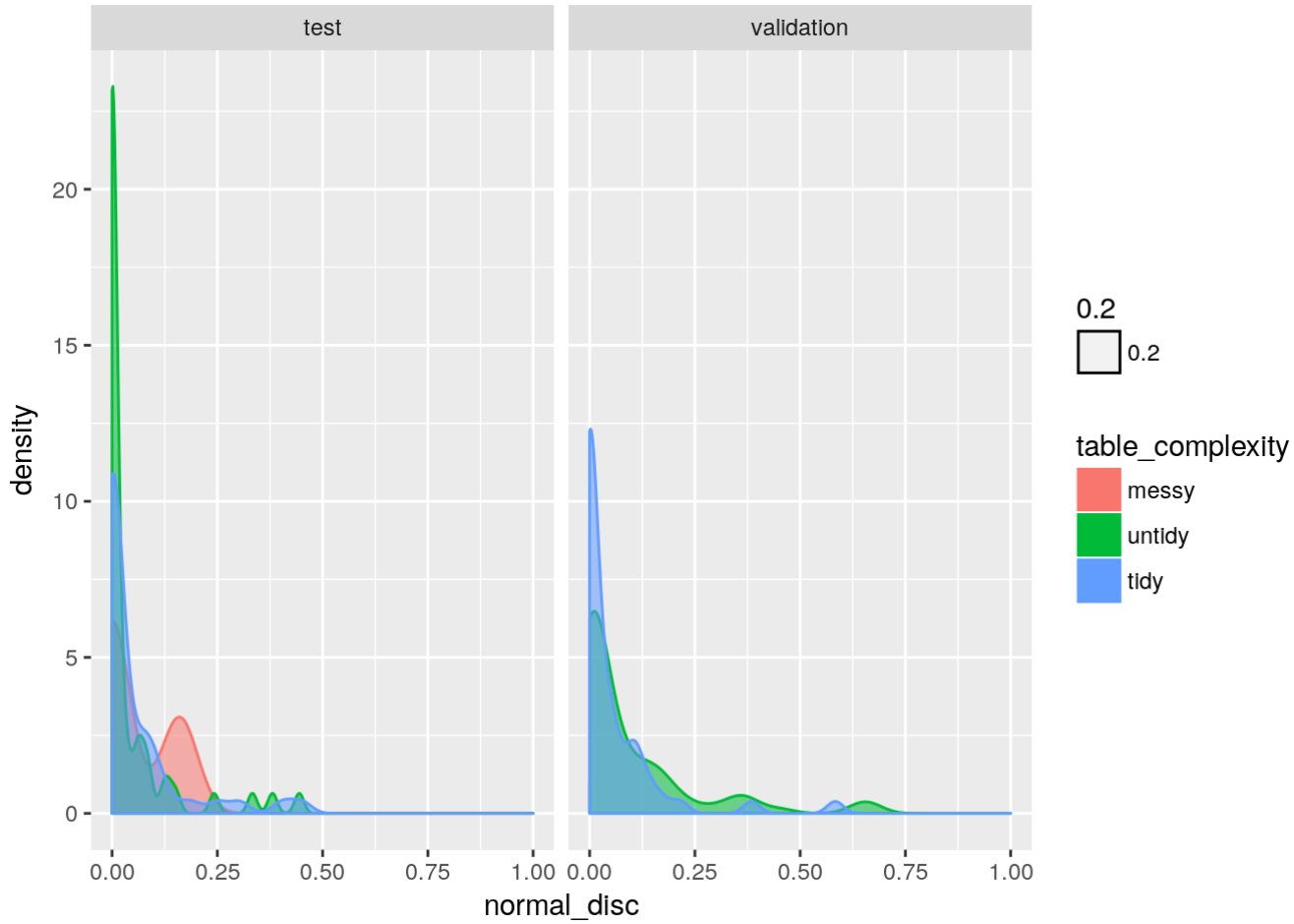

Figure 2. Density distributions of normalized discrepancy scores (i.e., number of cell discrepancies divided by the amount of cells). The more to the right in a plot, the larger the normalized discrepancy and a larger problem for the software. Validation set contained too few data to plot messy tables.

Documentation

In order to extract information from the raw scholarly articles, a set of 7 steps are taken. The following commands are based on the validation set in the folder `corpus-oa-validation` and assumes the Github repository is cloned. Steps 1 and 2 are included for cases where the PDFs are yet to be downloaded and organized.

1. Download all PDFs of the article
2. Organize the PDFs into a `cproject` structure

a. A `cproject` structure looks as follows

```
corpus-oa-validation
├── 10.1007_s00213-016-4512-6
│   ├── fulltext.pdf
├── 10.1007_s00213-016-4518-0
│   ├── fulltext.pdf
├── 10.1007_s00213-016-4519-z
│   ├── fulltext.pdf
...
...
...
├── 10.1371
│   ├── fulltext.pdf
```

b. If you have a set of PDFs in a folder, this command will reformat it into a `cproject` structure.

```
./norma-2017-03-16/bin/norma --project corpus-oa-validation --makeProject (\1)/fulltex
t.pdf --fileFilter .*/(.*)\.pdf
```

3. Convert the PDFs to SVGs with the following command

```
./norma-2017-04-11/bin/norma --project corpus-oa-validation/ --input fulltext.pdf --tra
nsform pdf2svg --outputDir corpus-oa-validation/
```

4. Manually cut out the tables in an SVG editor (recommended: Inkscape) and organize them in a subfolder `tables/table1/table.svg` structure.

5. Run the software to convert the SVG into HTML and CSV

```
./norma-2017-03-16/bin/norma --project  corpus-oa-validation --fileFilter ^.*tables/tab
le\(\\d+\)/table.svg  --outputDir corpus-oa-validation --transform stable2html

./norma-2017-03-16/bin/norma --project  corpus-oa-validation --fileFilter ^.*tables/tab
le\(\\d+\)/table.svg  --outputDir   corpus-oa-validation --output table.svg.csv  --tran
sform svgtable2csv
```

6. To present the extracted tables in an easy to consume way, first convert the SVGs to PNGs (make sure to have `svg2png` installed)

```
for FILE in `ls */tables/*/table.svg`; do inkscape --verb=FitCanvasToDrawing --verb=Fil
eSave --verb=FileClose $FILE; done

for FILE in `ls */tables/*/table.svg`; do svg2png $FILE -o $FILE.png; done
```

7. Create an overview file (`corpus-oa-validation/tableView.html`) with side-by-side comparisons of the extracted tables and the original

```
./norma-2017-03-16/bin/norma --project corpus-oa-validation --fileFilter ^.*tables/tabl
e\\d+$  --output  ./tableRow.html  --htmlDisplay  ^.*/table.svg.png ^.*/table.svg.html

./norma-2017-03-16/bin/norma --project corpus-oa-validation --output  tables/tableView.
html  --htmlAggregate ^.*tables/table\\d+/tableRow.html

./norma-2017-03-16/bin/norma --project corpus-oa-validation --output tableViewList.html
 --projectMenu .*/tables/tableView.html
```