TabbyXL: Software Platform & DSL for Rule-Based Spreadsheet Data Extraction

Presentation · April 2021

DOI: 10.6084/m9.figshare.14837571.v1

Alexey Shigarov
Institute for System Dynamics and Control Theory of Siberian Branch of Russian A…

Viacheslav Paramonov
Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of …

TabbyXL: Software Platform & DSL for Rule-Based Spreadsheet Data Extraction*

Alexey Shigarov Vasiliy Khristyuk Viacheslav Paramonov

Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences

shigarov@icc.ru

Apr 29th, 2021

* This work was supported by the Russian Science Foundation # 18-71-10001

Table of Contents

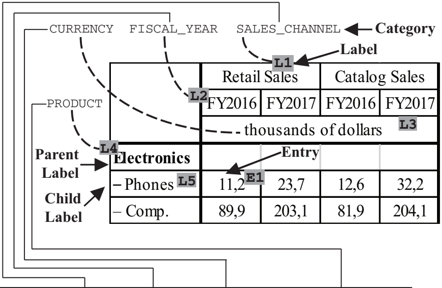
- Spreadsheets are Everywhere
  - 80M end-users in U.S. in 2005 [Scaffidi et al., 2005]
  - A large volume of valuable data
    - Government statistics (SAUS, CIUS)
    - Enterprise data (ENRON)
  - User friendly, Semi-structured, Multimedia
  - Only one rule — '**THERE ARE NO RULES!**'
- Applications
  - Business Intelligence
  - Data Science
- Spreadsheet Data Extraction
  - Refine & Cleanse tabular data
  - Recover missing semantics
  - Map tabular data to a structured form

Spreadsheet Data Extraction fits well with the *Table Understanding*[1]

5 Steps from Sheets to Relations

1. Detection
2. Structure Recognition
3. **Functional analysis**
4. **Structural analysis**
5. **Interpretation**

a



DATA	SALES CHANNEL	FISCAL YEAR	CURRENCY	PRODUCT
E1 11200	L1 retail	L2 2016	L3 u.s. dollars	L4 electronics/L5 phones
23700	retail	2017	u.s. dollars	electronics/phones
12600	catalog	2016	u.s. dollars	electronics/phones
32200	catalog	2017	u.s. dollars	electronics/phones

b

[1] See the definition in [Hurst, 2001]

- **AI Approach**
  - Ad-hoc heuristics [Embley et al., 2016, Koci et al., 2017, Koci et al., 2018]
  - ML-based models for some popular layouts [Chen, 2016, Koci et al., 2016]
  - DL-based models for Spreadsheet Table Understanding [Dong et al., 2019, Ghasemi-Gol et al., 2019, Ghasemi-Gol et al., 2020]
  - **Projects**
    - **TANGO**[2] (Brigham Young Univ.)
    - **Senbazuru**[3] (Univ. Michigan)
    - **DeExcelerator**[4] (TU Dresden)
  - **Limitation**
    - Predefined tricks of table design ('critical cells', header hierarchies)
    - Build-in functional cell regions (head, stub, body, footer)
    - Many tricks remain out of scope
    - Structured cells NOT SUPPORTED

[2] https://tango.byu.edu
[3] http://dbgroup.eecs.umich.edu/project/sheets
[4] https://wwwdb.inf.tu-dresden.de/research-projects/deexcelarator

- **End-User Programming Approach**
  - Spreadsheet-based domain-specific languages [Hung et al., 2011, Adam and Schultz, 2015]
  - Programming by examples [Harris and Gulwani, 2011, Gulwani et al., 2012, Barowy et al., 2015, Jin et al., 2017]
  - User-provided clues [Kandel et al., 2011, Swidan and Hermans, 2017]
  - **DSL**
    - **TranSheet** (Univ. New South Wales)
    - **TableProg** (Microsoft Research)
    - **FlashRelate** (Microsoft Research)
    - **Foofah**[5] (Univ. Michigan)
  - **Limitation**
    - **Fixed cell structure**
      (GOOD when tables have an identical cell structure)
      (NOT SCALED when tables vary the cell structure)
    - NO OPEN SOFTWARE (in most cases)

[5] https://github.com/umich-dbgroup/foofah

GOAL: Spreadsheet data extraction driven by user-defined rules

Our proposal

- Table object model
  - NO predefined tricks of table design
  - NO build-in functional cell regions
  - SUPPORT structured cells
- **CRL**, a DSL of rules for table analysis and interpretation
  - **“Different cell structures, the same tricks”**
  - Declarative (WHEN-THEN)
  - Well-defined terminology of Wang’s model [Wang, 1996]
  - Java imports AVAILABLE
  - **Drools** rule engine COMPATIBLE
- **TabbyXL**[a], a software platform for spreadsheet data extraction
  - Translation of CRL-rules to Java programs
  - Open Source & Free License

Table of Contents

Table Object Model

USE: Representation of facts on a table

Physical Layer

Cells characterized by layout, style, and content features

Logical Layer

Functional data items and their relationships:

- entries (values)
- labels (keys)
- categories (concepts)
- entry-label pairs
- label-label pairs
- label-category pairs

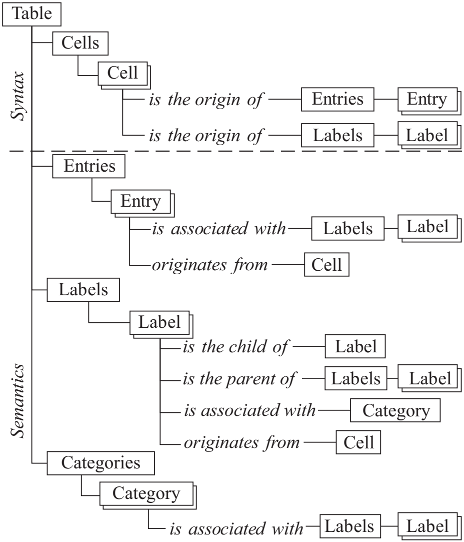


Table of Contents

CRL rules

USE: Mapping the physical layer into the logical one

FORM: **when** LHS **then** RHS

- **LHS** queries facts satisfying constraints
  - **EXISTS**:
    `cell | entry | label | category var: constraints, assignments`
  - **NOT EXIST**:
    `no cells | no entries | no labels | no categories: constraints`
- **RHS** modifies available facts and asserted new ones

4 kinds of rules

- Cell Cleansing
- Functional Analysis
- Structural Analysis
- Interpretation

CRL rules

Cell Cleansing

ACTIONS: to correct an inaccurate layout and content of a hand-coded table

- `merge` combines two adjacent cells
- `split` divides a merged cell that spans *n*-tiles into *n*-cells
- `set text` modifies a text of a cell
- `set indent` modifies a text indentation of a cell

Example

```
when
  cell corner: cl == 1, rt == 1, blank
  cell c: cl > corner.cr, rt > corner.rb
then
  split c
```

CRL rules
Functional Analysis

ACTIONS: to create entries and labels as functional data items

- `set tag` annotates a cell with a user-defined tag
- `new entry` (`new label`) creates an entry (label) from a cell text

Example

```
when
  cell corner: cl == 1, rt == 1, blank
  cell c: cl > corner.cr, rt > corner.rb
then
  new entry c
```

ACTIONS: to recover entry-label and label-label pairs

- `add label` associates an entry with a label
- `set parent` binds two labels as parent-child

Example

```
when
  cell c1: cl == 1
  cell c2: cl == 1, rt > c1.rt, indent == c1.indent + 2
  no cells: cl == 1, rt > c1.rt, rt < c2.rt, indent == c1.indent
then
  set parent c1.label to c2.label
```

CRL rules

ACTIONS: to recover label-category pairs

- `set category` associates a label with a category
- `group` places two labels to one group (an anonymous category)

Example

```
when
  label l1: cell.tag == "stub"
  label l2: cell.tag == "stub", cell.rt == l1.cell.rt
then
  group l1 with l2
```

CRL rules

Illustrative Example[6]

- **GOAL**: from (*a* and *c*) to (*b* and *d*)
- **Source tables** (*a* and *c*): **“different cell structures, the same tricks”**

a1		a2	
b1	1	b4	4
b2	2	b5	NA
b3		b6	6

a

DATA	A	B
1	a1	b1
2	a1	b2
4	a2	b4
6	a2	b6

b

a1		a2		a3	
b1		b3	3	b5	5
b2	2	b4	NA	b6	6

c

DATA	A	B
2	a1	b2
3	a2	b3
5	a3	b5
6	a3	b6

d

- **Ruleset**: cell cleansing — (*a*), role analysis — (*b*, *c*), structural analysis — (*d*, *e*), and interpretation — (*f*, *g*)

a

```
when cell c: c.text.matches("NA")
then set text "" to c
```

b

```
when cell c: (cl % 2) == 0, !blank
then new entry c
```

c

```
when cell c: (cl % 2) == 1
then new label c
```

d

```
when
   entry e
   label l: cell.cr == e.cell.cr
then add label l to e
```

e

```
when
   entry e
   label l: cell.rt == e.cell.rt, cell.cl == e.cell.cl - 1
then add label l to e
```

f

```
when label l: cell.rt == 1
then set category "A" to l
```

g

```
when label l: cell.rt > 1
then set category "B" to l
```

[6]This example is reproducible with CodeOcean, https://codeocean.com/capsule/5326436

Table of Contents

Software Platform

CRL Implementation

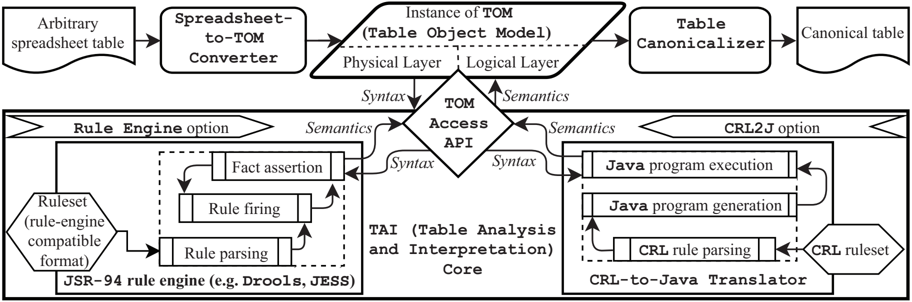
```
rule        = 'rule' <a Java integer literal> 'when' condition
              'then' action 'end' <EOL> {rule} <EOF>
condition   = query identifier [':' constraint {',' constraint}
              [',' assignment {',' assignment}]] <EOL> {condition}
constraint = <a Java boolean expr>
assignment = identifier ':' <a valid Java expr>
query       = 'cell' | 'entry' | 'label' | 'category' | 'no cells' |
              'no entries' | 'no labels' | 'no categories'
action      = merge | split | set text | set indent | set tag  |
              new entry | new label | add label | set parent |
              set category | group <EOL> {action}
merge       = 'merge' identifier 'with' identifier
split       = 'split' identifier
set text    = 'set text' <a Java string expr> 'to' identifier
set indent = 'set indent' <a Java integer expr> 'to' identifier
set mark    = 'set mark' <a Java string expr> 'to' identifier
new entry   = 'new entry' identifier ['as' <a Java string expr>]
new label   = 'new label' identifier ['as' <a Java string expr>]
add label   = 'add label' identifier | (<a Java string expr>
              'of' identifier | <a Java string expr>)
              'to' identifier
set parent    = 'set parent' identifier 'to' identifier
set category = 'set category' identifier | <a Java string expr>
                'to' identifier
group         = 'group' identifier 'with' identifier
identifier    = <a Java identifier>
```

- **CRL Grammar**[7] in ANTLR3 format
- DSL specification of CRL-dialect for **Drools**

[7] https://github.com/tabbydoc/tabbyxl/wiki/crl-language#implementation

Software Platform

Architecture



Two options are provided

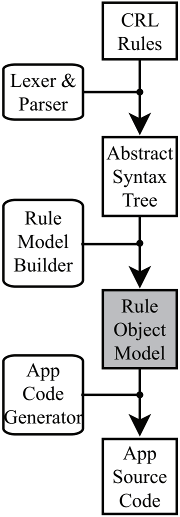
Rule Engine option

Executing a ruleset in an appropriate format with a JSR-94 compatible rule engine (e.g. **Drools**, **Jess**)

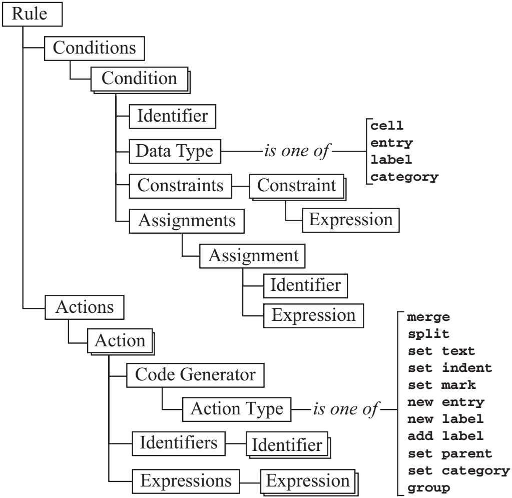
CRL2J option

Translating a ruleset expressed in CRL to an executable Java program

CRL-to-Java



Rule Object Model



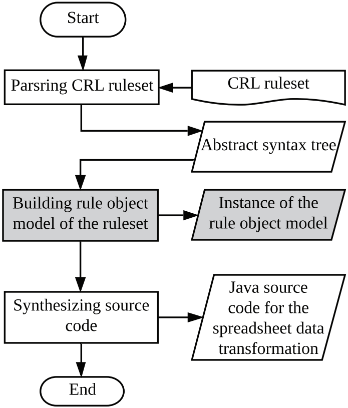
Example (Source Rule)

```
when
  cell corner: cl == 1, rt == 1, blank
  cell c: cl > corner.cr, rt > corner.rb, !tagged
then
  set tag "@entry" to c
  new entry c
```

Example (Fragment of the Generated Java Code)

```
...
Iterator<CCell> iterator1 = getTable().getCells();
while (iterator1.hasNext()) {
  corner = iterator1.next();
  if ((corner.getCl() == 1) && (corner.getRt() == 1) && ...
    Iterator<CCell> iterator2 = getTable().getCells();
    while (iterator2.hasNext()) {
...
```

GET: Java code from CRL rules (**Ready for compilation**)



GET: Maven-project (**Ready for build**)

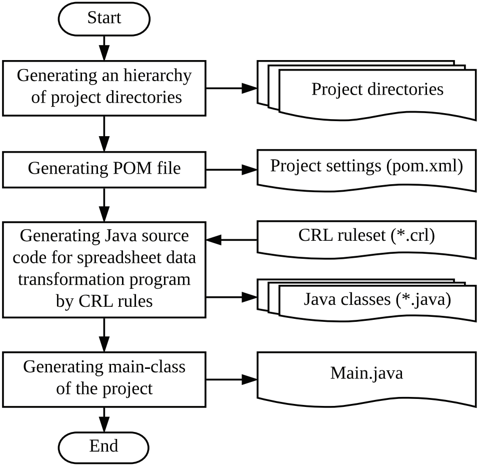


Table of Contents

Empirical Results

Performance Evaluation[8]

SETUP: 200 tables of **Troy200** dataset [Nagy, 2016] + **16** CRL rules

Metrics	Functional analysis		Structural analysis	
	Type of instances			
	entries	labels	entry-label pairs	label-label pairs
Recall	0.9813 $\frac{16602}{16918}$	0.9965 $\frac{4842}{4859}$	0.9773 $\frac{34270}{35066}$	0.9389 $\frac{1951}{2078}$
Precision	0.9996 $\frac{16602}{16609}$	0.9364 $\frac{4842}{5171}$	0.9965 $\frac{34270}{34389}$	0.9784 $\frac{1951}{1994}$
F-score	**0.9904**	**0.9655**	**0.9868**	**0.9582**

Metrics

$$\text{recall} = {}^{|R \cap S|}/_{|S|} \quad \text{precision} = {}^{|R \cap S|}/_{|R|}$$

S is a set of instances in a source table, R is a set of instances in its canonical form

[8] All data and steps to reproduce the results are available at http://dx.doi.org/10.17632/ydcr7mcrtp.5

Process Time

The comparison of the running time by using **TabbyXL** with three different options for transforming 200 tables of **Troy200** dataset [Nagy, 2016]

Running time of	**CRL2J**	**Drools**	**Jess**
Ruleset preparation (t_1)	2108* ms	1711† ms	432† ms
Ruleset execution (t_2)	367** ms	1974‡ ms	4149‡ ms

* t_1 — a time of parsing and compiling the original ruleset into a Java program
** t_2 — a time of executing the generated Java program

† t_1 — a time of parsing the original ruleset and adding the result into a rule engine session
‡ t_2 — a time of asserting facts into the working memory and matching rules against the facts

For testing, we used 3.2 GHz 4-core CPU

Empirical Results

Performance Evaluation

SETUP: 200 tables of **SAUS** dataset[9] + **13** CRL rules

Metrics	Functional analysis		Structural analysis	
	Type of instances			
	entries	labels	entry-label pairs	label-label pairs
Recall	0.9928 $\frac{135785}{136766}$	0.9360 $\frac{18804}{20089}$	0.9550 $\frac{370022}{387499}$	0.8391 $\frac{15058}{17946}$
Precision	0.9420 $\frac{135785}{144148}$	0.9446 $\frac{18804}{19906}$	0.9275 $\frac{370022}{398967}$	0.8636 $\frac{15058}{17437}$
F-score	**0.9667**	**0.9403**	**0.9410**	**0.8512**

Metrics

$$\text{recall} = {}^{|R \cap S|}/_{|S|} \quad \text{precision} = {}^{|R \cap S|}/_{|R|}$$

S is a set of instances in a source table, R is a set of instances in its canonical form

[9] http://dbgroup.eecs.umich.edu/project/sheets/datasets.html

Comparison with Ad-hoc Solutions

Empirical Results

Functional Analysis

- Dataset: **Troy200**[a]
- **TANGO** — *accuracy* $= 0.990$ (for detecting critical cells) [Embley et al., 2016]
- **16 CRL** rules & **TabbyXL** — $F_1 = 0.995$ (for extracting entries & labels)

[a] http://tc11.cvc.uab.es/datasets/Troy_200_1

Structural Analysis

- Dataset: A random subset of **SAUS**[a]
- **Senbazuru** — $F_1 = 0.886$ (for predicting parent-child relationships in stub hierarchies of 100 tables) [Chen and Cafarella, 2014]
- **18 CRL** rules & **TabbyXL** — $F_1 = 0.851$ (for extracting label-label pairs from 200 tables)

[a] http://dbgroup.eecs.umich.edu/project/sheets/datasets.html

Empirical Results

Case Study[10]

GOAL: Populating a web-based statistical atlas of the Irkutsk region — (*b*) via extracting data from government statistical reports — (*a*)

Corner cell

Head part

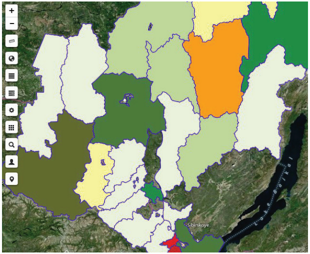
	h1	h2				h3	
		h4		h5			
		h6	h7	h8	h9	h10	h11
s1							
··s2	d1	d2	d3	d4	d5	d6	d7
····s3	d8	d9	d10	d11	d12	d13	d14
····s4	d15	d16	d17	d18	d19	d20	d21
··s5	d22	d23	d24	d25	d26	d27	d28
····s6	d29	d30	d31	d32	d33	d34	d35
··s7	d36	d37	d38	d39	d40	d41	d42
····s8	d43	d44	d45	d46	d47	d48	d49
s9	d50	d51	d52	d53	d54	d55	d56

Stub part

Body part

a

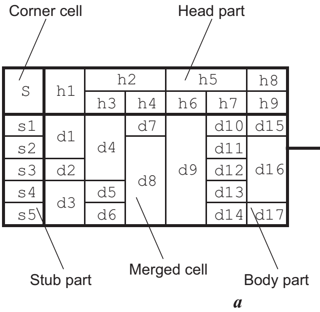
DATA	HEAD	STUB
d1	h1	s1|s2
d2	h2|h4|h6	s1|s2
d3	h2|h4|h7	s1|s2
d4	h2|h5|h8	s1|s2
d5	h2|h5|h9	s1|s2
d6	h3|h10	s1|s2
d7	h3|h11	s1|s2
d8	h1	s1|s2|s3
d9	h2|h4|h6	s1|s2|s3
d10	h2|h4|h7	s1|s2|s3
d11	h2|h5|h8	s1|s2|s3
d12	h2|h5|h9	s1|s2|s3
d13	h3|h10	s1|s2|s3
d14	h3|h11	s1|s2|s3
...	...	...
d56	h3|h11	s9



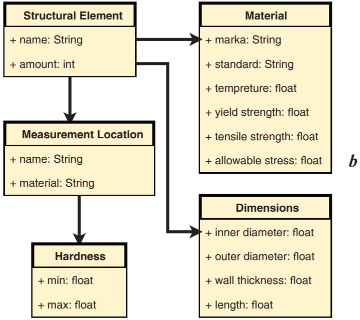
b

[10]The more detail can be found at https://github.com/tabbydoc/tabbyxl/wiki/statistical-atlas

GOAL: Generating conceptual models — (*b*) from arbitrary tables presented in industrial safety inspection reports — (*a*)



DATA	HEAD	S
d1	h1	s1
d1	h1	s2
d2	h1	s3
d3	h1	s4
d3	h1	s5
d4	h2|h3	s1
d4	h2|h3	s2
d4	h2|h3	s3
d5	h2|h3	s4
d6	h2|h3	s5
...	...	...
d17	h8|h9	s5



[11] The more detail can be found at https://github.com/tabbydoc/tabbyxl/wiki/industrial-safety-inspection

Table of Contents

Conclusions

- **GOOD NEWS**: To write the rules can be cheaper than to train ML models or to hard-code ad-hoc heuristics
- **BAD NEWS**: Need to learn the rule language

- **Limitation**
  - Table Detection NOT PROVIDED
  - Hand-coded tables — The Structure Recognition First
  - HARD TO SCALE due to ambiguity of table tricks
  - Simple interpretation without KGs
  - CLEAR language for Pivot Tables, NOT for Entity-Focused Tables

- **Further work**
  - TABLE EXTRACTION — Spreadsheet Intelligence
  - Support Entity-Focused Tables CLEARLY

Table of Contents

References I

Adam, S. and Schultz, U. P. (2015).
Towards tool support for spreadsheet-based domain-specific languages.
In *Proc. 2015 ACM SIGPLAN Int. Conf. Generative Programming: Concepts and Experiences*, page 95–98.

Barowy, D. W., Gulwani, S., Hart, T., and Zorn, B. (2015).
FlashRelate: Extracting relational data from semi-structured spreadsheets using examples.
SIGPLAN Not., 50(6):218–228.

Chen, Z. (2016).
Information Extraction on Para-Relational Data.
PhD thesis, University of Michigan, US.

Chen, Z. and Cafarella, M. (2014).
Integrating spreadsheet data via accurate and low-effort extraction.
In *Proc. 20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 1126–1135.

Dong, H., Liu, S., Han, S., Fu, Z., and Zhang, D. (2019).
Tablesense: Spreadsheet table detection with convolutional neural networks.
Proc. AAAI Conf. Artificial Intelligence, 33(01):69–76.

Embley, D. W., Krishnamoorthy, M. S., Nagy, G., and Seth, S. (2016).
Converting heterogeneous statistical tables on the web to searchable databases.
Int. J. Document Analysis and Recognition, 19(2):119–138.

Ghasemi-Gol, M., Pujara, J., and Szekely, P. (2019).
Tabular cell classification using pre-trained cell embeddings.
In *2019 IEEE Int. Conf. Data Mining*, pages 230–239.

References II

Ghasemi-Gol, M., Pujara, J., and Szekely, P. (2020).
Learning cell embeddings for understanding table layouts.
Knowl Inf Syst, 33(01).

Gulwani, S., Harris, W. R., and Singh, R. (2012).
Spreadsheet data manipulation using examples.
Commun. ACM, 55(8):97–105.

Harris, W. R. and Gulwani, S. (2011).
Spreadsheet table transformations from examples.
In *Proc. 32nd ACM SIGPLAN Conf. Programming Language Design and Implementation*, page 317–328.

Hung, V., Benatallah, B., and Saint-Paul, R. (2011).
Spreadsheet-based complex data transformation.
In *Proc. 20th ACM Int. Conf. Information and Knowledge Management*, pages 1749–1754.

Hurst, M. (2001).
Layout and language: Challenges for table understanding on the web.
In *Proc. 1st Int. Workshop Web Document Analysis*, pages 27–30.

Jin, Z., Anderson, M. R., Cafarella, M., and Jagadish, H. V. (2017).
Foofah: Transforming data by example.
In *Proc. ACM Int. Conf. Management of Data*, pages 683–698.

Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J. (2011).
Wrangler: Interactive visual specification of data transformation scripts.
In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, page 3363–3372.

References III

Koci, E., Thiele, M., Lehner, W., and Romero, O. (2018).
Table recognition in spreadsheets via a graph representation.
In *Proc. 13th IAPR Int. Workshop on Document Analysis Systems*, pages 139–144.

Koci, E., Thiele, M., Romero, O., and Lehner, W. (2016).
A machine learning approach for layout inference in spreadsheets.
In *Proc. 8th Int. Joint Conf. Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 77–88.

Koci, E., Thiele, M., Romero, O., and Lehner, W. (2017).
Table identification and reconstruction in spreadsheets.
In *Proc. 29th Int. Conf. Advanced Information Systems Engineering*, pages 527–541.

Nagy, G. (2016).
TANGO-DocLab web tables from international statistical sites (Troy_200), 1, ID: Troy_200_1.

Scaffidi, C., Shaw, M., and Myers, B. (2005).
Estimating the numbers of end users and end user programmers.
In *2005 IEEE S. Visual Languages and Human-Centric Computing*, pages 207–214.

Swidan, A. and Hermans, F. (2017).
Semi-automatic extraction of cross-table data from a set of spreadsheets.
In Barbosa, S., Markopoulos, P., Paternò, F., Stumpf, S., and Valtolina, S., editors, *End-User Development*, pages 84–99.

Wang, X. (1996).
Tabular Abstraction, Editing, and Formatting.
PhD thesis, University of Waterloo, Waterloo, Ontario, Canada.

Thanks

Read more about the project at
http://td.icc.ru

The project source code is available at
https://github.com/tabbydoc/tabbyxl