A Rectangle Mining Method for Understanding the Semantics of Financial Tables

*Xilun Chen**, Laura Chiticariu†, Marina Danilevsky†, Alexandre Evfimievski† and Prithviraj Sen†
*: Cornell University †: IBM Research - Almaden

Structures and Semantics of Tabular Data

CONSOLIDATED STATEMENT OF COMPREHENSIVE INCOME (LOSS)

Unaudited (Canadian dollars in millions)		Three months ended June 30 2015	Three months ended June 30 2014	Six months ended June 30 2015	Six months ended June 30 2014
Comprehensive income (loss)					
Net income (loss) for the period		$ 296	$ 223	$ (13)	$ (118)
Other comprehensive income (loss), net of taxes of nil:					
Items that will not be reclassified to net income					
Remeasurements on employee benefit liabilities	Note 5	542	(90)	790	(293)
Items that will be reclassified to net income					
Loss on derivatives designated as cash flow hedges	Note 9	(1)	-	(1)	-
Total comprehensive income (loss)		**$ 837**	**$ 133**	**$ 776**	**$ (411)**
Comprehensive income (loss) attributable to:					
Shareholders of Air Canada		$ 836	$ 132	$ 774	$ (413)
Non-controlling interests		1	1	2	2
		$ 837	$ 133	$ 776	$ (411)

Table Headers / Column Headers / Row Headers / Data

Cell Semantics: The unaudited comprehensive income (loss) attributable to shareholders of Air Canada in the six months ended June 30, 2015 is: $774 million Canadian dollars.

All cells circled in *solid* lines are related to the semantics of this cell

Figure: Dataframe Tables

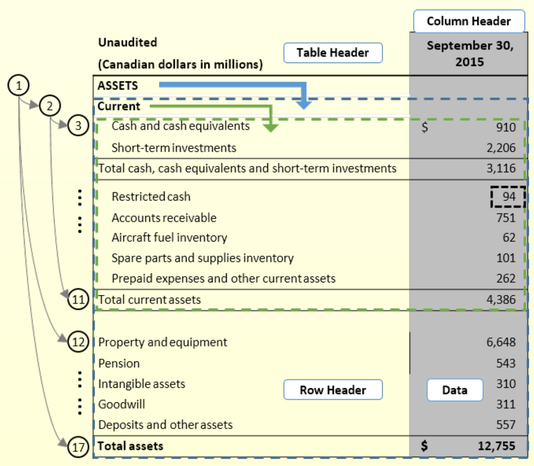
Input: HTML tables converted from PDF documents (A separate open research problem)
Output: Tuples containing the semantics of each body cell

Unaudited	(Canadian dollars in millions)	Comprehensive Income (loss) attributable to:	Shareholders of Air Canada	Six months ended June 30	2015	**$774**
Unaudited	(Canadian dollars in millions)	Comprehensive Income (loss)	Total comprehensive Income (loss)	Six months ended June 30	2015	**$776**

In practice, *semantic hierarchy* of **row headers** is the most complex!

Row header semantics hierarchy: Pair-wise vs. Rectangle

- **Existing approach: Pair-wise Classification**
  - For each pair of rows, classify whether they form a *parent-child relation*
  - The result is a tree (left)
  - **Fragmentation**: non-consecutive children
    - (1) has children (2), and (12) to (17).
    - Fragmented predictions; difficulty in detecting section boundaries; needs post-processing
  - **Pair-local Features**: can only utilize features local to the pair itself
    - (11) and (12) has almost identical features
    - (11) is a child of (2), (12) is not
    - Correct prediction involves properly inferring the boundary of section headed by (2)



- **Our approach: Rectangle Mining**
  - Shown on the right
    - Row (1) is the header of the blue rectangle; row (2) heads the green one.
  - Makes decisions based on contiguous blocks of cells (i.e., rectangles)
  - Contiguous blocks of cells are assigned to a parent in one shot (no fragmentation)
  - Features global within a rectangle become available (no pair-local decisions)

Rectangle Mining (ReMine) Rectangle Features and Feature Partial Order

- **Assumption 1.** All the cells semantically headed (explained) by a single cell form a contiguous rectangular region, defined as the support rectangle of that cell.

ReMine mines all *interesting* rectangles (support rectangle of some cell). A rectangle is represented as a vector features $\phi(r) = \Phi$.

- **Definition 1**. A partial order $\preccurlyeq$ defined on the feature space Φ is called a feature partial ordering.

Intuitively, $\preccurlyeq$ is a comparator that returns one of {$\succ$, $\prec$, $\asymp$, non-comparable}. $\phi_1 \succ \phi_2$ means the header row of rectangle r_1 tends to semantically head r_2

In this work, ReMine works with only a few intuitive features:

- Two rectangle-wise global features: isEndedSection, isEmptySection
- In the future: more sophisticated features; data-driven feature learning

Single-row Features (local)		**Rectangle Features** (global)	
Feature Name	**Order**	**Feature Name**	**Order**
Boldness	T $\succ$ F	isEndedSection	T $\prec$ F
Indentation	G $\prec$ L	isEmptySection	T $\succ$ F
Blankness	T $\prec$ F		
Capitalization	T $\succ$ F		
isSectionHeader	T $\succ$ F		
isTotalRow	*See text*		

TABLE I: Features and Feature Partial Order used in REMINE. For the partial order, "T" means the feature is true while "F" indicates false. For numeric features such as indentation, "G" means greater value while "L" means less.

- isEndedSection: Rectangles headed by a section header row and terminated by a paired total row (e.g. Rectangle of rows 2-11 in the example table)
- isEmptySection: Rectangles of only one row, a section header defined by isSectionHeader (e.g. Rectangle of only row 1)
- For the order on isTotalRow, $\preccurlyeq$ always return non-comparable so that a total row cannot take children. Otherwise, when considered as child candidate, the feature isTotalRow is ignored

ReMine Algorithm

ReMine is a two-stage iterative algorithm. Larger rectangles are built by iteratively combining smaller rectangles until convergence. Feature partial order $\preccurlyeq$ is utilized to make decisions.

- **Definition 2**. A rectangle r is *minimal* within a list of rectangles R under partial order $\preccurlyeq$, if $\forall r'$ after r in R, it satisfies $r \not\succ r'$.

  *We only work on minimal rectangles in each iteration because they are "safe" to be combined or attached with no potential children remaining in R.

- The **Combination** stage combines consecutive rectangles with equal features, and all the header rows of the combined rectangles become headers of the new rectangle.
- The **Attachment** stage attaches a rectangle r_2 to the preceding rectangle r_1 as a child if $r_1 \succ r_2$, and the header row of r_1 remains as the header of the new rectangle

Algorithm 1 REMINE Algorithm for Extracting Row Header Semantic Hierarchy. Functions such as `findMinimalRects` are explained later.

Input: List of rows $\mathcal{C}$ starting from the top; Partially ordered set $(\mathbf{\Phi}, \preccurlyeq)$
Output: Set of parent-child relations $\mathcal{P}$
State: Maintains *ordered list* $\mathcal{R}$ of active top-level rectangles

$\mathcal{P} = \varnothing$
$\mathcal{R} = \mathcal{C}$ ▷ Initially each row is a rectangle
repeat
  $combineR =$ new $List()$ ▷ Init with empty list
  $\mathcal{M}$ = `findMinimalRects`$(\mathcal{R}, \mathbf{\Phi})$
  ▷ Combination Stage
  for all $r_i \in \mathcal{R}$ **do** ▷ r_i is the i-th rectangle in $\mathcal{R}$
    if $r_i \in \mathcal{M}$ and $r_i \asymp r_{i+1} \asymp \cdots \asymp r_j$ **then**
      $combineR$.append(`Combine`$(r_i, \ldots, r_j)$)
    else $combineR$.append(r_i)
  $attachR =$ new $List()$
  $\mathcal{M}$ = `findMinimalRects`$(combineR, \mathbf{\Phi})$
  ▷ Attaching Stage
  for all $r_i \in combineR$ **do**
    if $r_{i+1} \in \mathcal{M}$ and $r_i \succ r_{i+1}$ **then**
      $attachR$.append(`Attach`(r_i, r_{i+1}))
      $\mathcal{P} = \mathcal{P} \cup$ `addRelations`(r_i, r_{i+1})
    else $attachR$.append(r_i)
  $\mathcal{R} = attachR$
until $\mathcal{R}$ not changing
return $\mathcal{P}$

ReMine Example

ReMine starts with each row as a rectangle.

- Iteration 1 Combination:
  - Minimal rectangles: row 3-17
  - Row 3,4 will be combined into a single rectangle $Rect(3,4)$; similarly, we have $Rect(6,10), Rect(12,16)$
- Iteration 1 Attachment:
  - Minimal rectangles: $Rect(3,4), Rect(6,10), Rect(12,16), Rect(5), Rect(11), Rect(17)$
  - $Rect(3,4)$ is attached to $Rect(2)$ becoming $Rect(2,4)$
- Iteration 2 Combination: no adjacent rectangles with equal features
- Iteration 2 Attachment: $Rect(2,4)$ continues to consume $Rect(5)$ becoming $Rect(2,5)$
- Iterations 3 & 4 are similar to Iteration 2: $Rect(2,5)$ further consumes $Rect(6,10)$ and $Rect(11)$
  - At the end of Iteration 4, $Rect(2,11)$ has feature isEndedSection and stops taking children
- Iteration 5: $Rect(2,11)$ is attached to $Rect(1)$
- Iteration 6: $Rect(1,11)$ consumes $Rect(12,16)$
- Iteration 7: $Rect(17)$ is attached to $Rect(1,16)$

Experiments and Results

Systems	Financial Tables Dataset						ICDAR 2013 Dataset					
	Direct			Transitive			Direct			Transitive		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SVM	76.27	62.03	68.42	76.27	50.68	60.89	54.69	47.30	50.72	41.86	43.37	42.60
RIPPER	78.40	48.27	59.74	81.31	47.15	59.69	92.31	36.36	52.17	87.61	36.40	51.43
REMINE	82.45	85.84	**84.11**	86.30	89.56	**87.90**	91.43	86.49	**88.89**	85.88	87.95	**86.90**

TABLE II: Leave-one-company-out cross validation results. Section III defines *Direct* and *Transitive* evaluation schemes.

- **Datasets**
  - **Financial Table Dataset**
    - 2015 Q3 financial statements of 6 companies
    - 72 tables; manually labeled semantics
    - We publicly release the annotated dataset
  - **ICDAR 2013 Table Competition Dataset**
    - Only 7 tables exhibit non-trivial row hierarchy structures

Feature Name	Pair-wise	REMINE	Ablation
Single-row Features (local)			
Boldness	✓	✓	85.36
Indentation	✓	✓	83.73
Blankness	✓	✓	87.90
Capitalization	✓	✓	88.94
isSectionHeader	✓	✓	87.50
isTotalRow	✓	✓	85.87
Row-pair Features (local)			
isTotalRowPair	✓	N/A	N/A
Rectangle Features (global)			
isEndedSection	N/A	✓	86.21
isEmptySection	N/A	✓	84.33

TABLE III: Feature set comparison and features ablation results on the Financial Table dataset. A feature is used (✓) or not applicable to the system (N/A). Feature ablation is conducted on REMINE and indicates *Transitive* F1.

- **Experimental Results**
  - Results shown in Table II
  - **Baseline methods**
    - A SVM pair-wise classifier trained to predict parent-child relations
    - RIPPER, which learns rules of logical expressions of features, and is more transparent (explainable) than SVM
  - **Evaluation Metrics**
    - *Direct*: Used in literature and evaluates only direct parent-child relations
      - In example table, row 1 has 8 direct children {2, 11-17}
    - *Transitive*: Our proposal, where all descendants are counted during evaluation
      - In example table, row 1 has 16 descendants {2-17}
      - Intuition: mistakes on certain relations are more costly in understanding the entire table semantics. E.g. failing to predict row 2 as a child of row 1
- **Feature Ablation**
  - Shown in Table III
  - When removing both rectangle features, transitive F1 decreases to 82% (down 5%)
- **Error Analysis**
  - Hand-crafted feature partial order: lacks the flexibility to account for outliers.
    - capitalized acronyms ("EBITDA") trigger the Capitalization feature
  - The extraction of certain complex features such as isTotalRow and isTotalRowPair