KU LEUVEN

Knowledge Discovery on Spreadsheet Data using Semantic Information

Sarah Binta Alam Shoilee

Thesis Defense
Master of Science in Artificial Intelligence
Option: Big Data Analytics

September 28, 2020

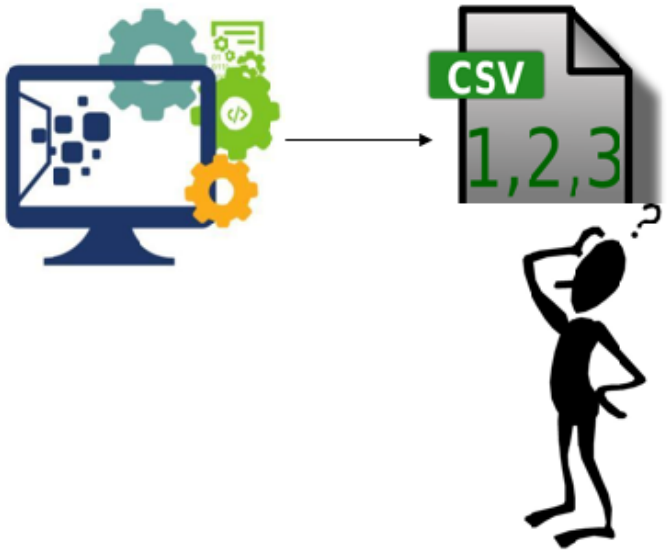
Why Spreadsheet?

- It is estimated that 90% of all analysts use spreadsheets for their calculations[1].
- 95% of US firms and 80% in Europe use spreadsheets in some form of financial reporting[2].
- Some studies report that up to 90% of real-world spreadsheets contain errors[3].
- chances of making mistakes in the data-sheet are 63%, even when the creator and maintainer is the same individual[4].

1. W.L. Winston. Executive education opportunities. OR/MS Today, 28(4):8–10, 2001.
2. R. Panko. Facing the problem of spreadsheet errors. Decision Line, 37(5), 2006.
3. Rajalingham, K., Chadwick, D. R., & Knight, B. (2008). Classification of spreadsheet errors.
4. R. Panko. What We Don't Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. In Proceedings of the EuSpRIG 2015 Conference on Spreadsheet Risk Management, 2016.

Use Case

- poorly filed spreadsheet
- imported spreadsheet from other software
- comma separated file (CSV)
- inexperienced user



Objective

Problem: Find tabular constraints (or re-discover mathematical functions or formulas)

- Inspiration: TaCLe, introduced by Kolb et al. (2017)
- Improve TaCLe's precision & run-time
- Formula discovery using header information

ID	Salesperson	1st	2nd	3rd	4th	Total	Rank	Label
1	Diana Coolen	353	378	396	387	1514	2	Great
2	Marc Desmet	370	408	387	386	1551	1	Great
3	Kris Goossens	175	146	167	203	691	3	Low
4	Birgit Kenis	93	98	96	105	392	4	Low

Formula: $T1[:, 7] = SUM_{row}(T1[:, 3{:}6])$

Thesis Contribution

Introduction of a semantic heuristic approach for formula reconstruction on a spreadsheet

- a predictive method on learning constraint existence
- a priority-based constraint search algorithm

ID	Salesperson	1st	2nd	3rd	4th	Total	Rank	Label
1	Diana Coolen	353	378	396	387	1514	2	Great
2	Marc Desmet	370	408	387	386	1551	1	Great
3	Kris Goossens	175	146	167	203	691	3	Low
4	Birgit Kenis	93	98	96	105	392	4	Low

Terminology[5]

- **vector** (v): single row or column
- **header word** (w_i): header word assigned for vector index i
- **block** (β): group of neighbouring cell arrays of same type
- **maximal blocks** (β): largest possible groups of type consistent neighbouring cell
- **Constraint Template** (c): Canonical form representation of formula
- **Target Variable**: left-hand-side argument of mathematical formula; the right-hand-side argument(s) are **data variable**.

Set of corresponding components will be denoted in BOLD, Capital letters.

5. Kolb, S., Paramonov, S., Guns, T., & De Raedt, L. (2017). Learning constraints in spreadsheets and tabular data. Machine Learning, 106(9-10), 1441-1468.

Terminology[5] *(Cont.)*

Concept of Blocks and Sub-blocks

A block β is a super-block of a sub-block β', iff β contains all the vectors from the β' in the same orientation from the same table.

Example of Constraint Template

- **Syntax**: $B_2 = SUM_{row} \boldsymbol{B_1}$
- **Signature**: B_2 and $\boldsymbol{B_1}$ are numeric, columns($\boldsymbol{B_1}$) ≤ 2, and rows($\boldsymbol{B_1}$) = length(B_2)
- **Definition**: $B_2[i] = \sum_{j=1}^{columns(\boldsymbol{B_1})} row(i, \boldsymbol{B_1})[j]$

Application Overview

Algorithm 1: The Semantic-TaCLe approach

Step 1: Identify the tables($\boldsymbol{T}$) and their headers($\boldsymbol{H}$)

Step 2: Generate and pre-process vector-header word pairs (v_i, w_i) from $\boldsymbol{T}$ and $\boldsymbol{H}$

Step 3: For each vector $v \in \boldsymbol{V}$:

(a) Predict 'any' constraint presence on v

(b) If (a) positive: semantically rank supported constraint list $\boldsymbol{C}$

(c) for each constraint $c \in \boldsymbol{C}$

1. Consider v as a target variable
2. Find remaining 'valid' input argument for c
3. Validate arguments variables assignment

Step-1: Table & Header Extraction

ID	Salesperson	1st Quarter	2nd Quarter	3rd Quarter	4th Quarter	Total
1	Diana Coolen	353	378	396	387	1514
2	Marc Desmet	370	408	387	386	1551
3	Kris Goossens	175	146	167	203	691
4	Birgit Kenis	93	98	96	105	392
Total		991	1030	1046	1081	4148
Average		247.75	257.5	261.5	270.25	1037
Max		370	408	396	387	1551
Min		93	98	96	105	392

Figure: Example of a spreadsheet data

Table	**Header**	**Orientation**
T1 [0:5, 0:7]	H1(horizontal) = [0:1, 0:7]	vertical
T2 [7:11, 0:1]	H2 = []	vertical, horizontal
T3 [7:11, 2:7]	H3(vertical) = [7:11, 0:1]	horizontal

Step-2: Pre-processioning

Algorithm 2: Feature Extraction

```
while token in header text do
    if token.dep = "adjectival modifier" then
        return token ;
    else if token.pos = "adj" or "det" then
        return token ;
    else if token.dep = "root" then
        return token ;
```

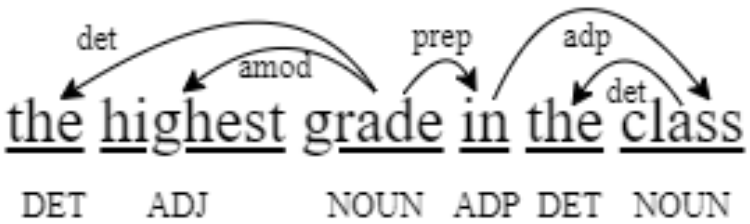


Figure: Example of a Dependency Parse Tree of a Header Text

Step-3: Constraint Learning

Algorithm 3: Constraint Learning using Header Information

Procedure LEARNCONSTRAINTS ($tuple(v, w), \boldsymbol{C}$)

```
1   if φ(w) = 0 then
2       return; ▷ no constraint for vector v
    end
3   else
4       C ← ψ(w, C); ▷ Rank constraint set C
5       for c ∈ C do
6           if ASSIGN(v, c) = True then
7               S ← VALIDATION(v, c, β)
8               return S; ▷ valid constraint found
            end
        end
    end
```

Implementation

- *Table Extraction*: Autoextract, introduced in TaCLe
- *Header Identification and vector mapping*: Python class implementation
- *Feature Extraction*: Python library Spacy's dependency parse tree and POS tagger
- *Constraint existence prediction*: Decision Tree from Scikit learn
- *Constraint ranking*: Python class implementation with help of word2vec similarity score
- *Final Constraint validation*: TaCLe's python classes and constraint solver

Evaluation

1. Effectiveness of Constraint Ranking
2. Effectiveness of Constraint Existence Prediction
3. Effectiveness of Proposed Approach

Evaluation Setup:

- Synthetic spreadsheet (10)
- Real-world spreadsheet (10)
- Supported Constraint (Aggregate Formula)
  - SUM, PRODUCT, AVERAGE, MAX, MIN, COUNT

1. Effectiveness of Constraint Ranking

Search Space Improvement:

$$score_{c_j} = \frac{\sum_{w \in \boldsymbol{W}} Rank_{w,\boldsymbol{C}}(c_j)}{\|\boldsymbol{W}\|} \quad (1)$$

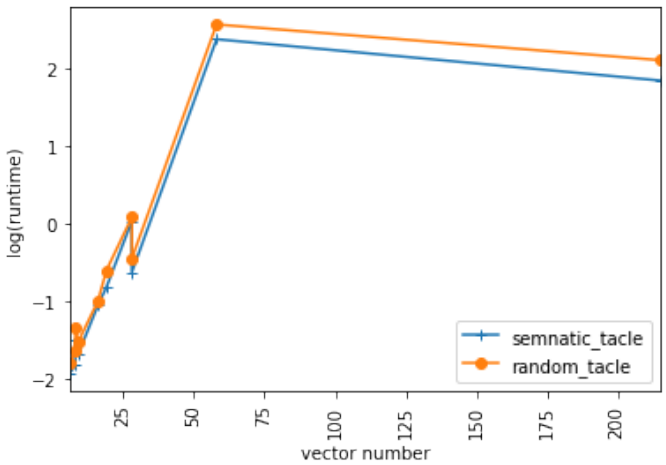
where c_j is the j^{th} constraint in the supported constraint list $\boldsymbol{C}$; w is a word that belongs to the word list ($\boldsymbol{W}$) for constraint c_j and "Rank" is a function that returns the constraint position in the sorted constraints list for word w.

	Naive Ranking	Substituted Ranking	Final Ranking
Overall	0.439	0.253	0.218

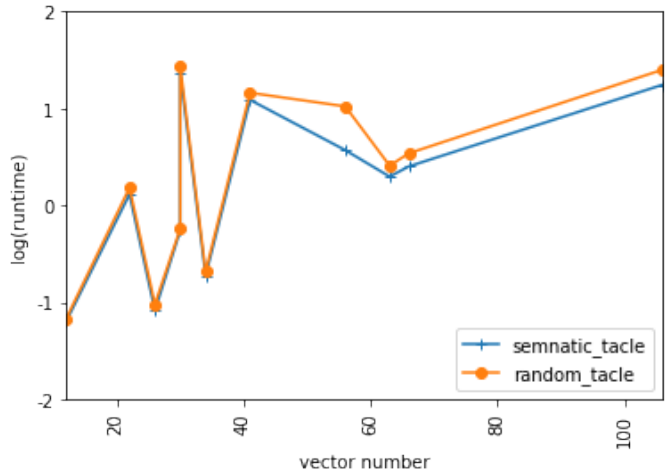
Table: Average semantic score of all supported constraint

1. Effectiveness of Constraint Ranking *(Cont.)*

Computation-time Improvement:



(a) Synthetic Spreadsheet



(b) Real Spreadsheets

Figure: Log plots of computation-time for semantic similarity ranked constraints implementation and baseline randomly ranked constraint implementation

2. Effectiveness of Existence Prediction

Predictability of Constraint from the Header Text:

	Word2Vec			Tf-Idf		
	Logistic Regression	Naive Bayes	Decision Tree	Logistic Regression	Naive Bayes	Decision Tree
Accuracy:	0.98	0.98	0.93	0.96	0.44	0.99
Precision:	0.88	0.80	0.5	1	0.11	0.89
Recall:	0.88	1	0.88	0.5	0.88	1.0

Table: Performance metric comparison between word2vec and tf-idf

2. Effectiveness of Existence Prediction *(Cont.)*

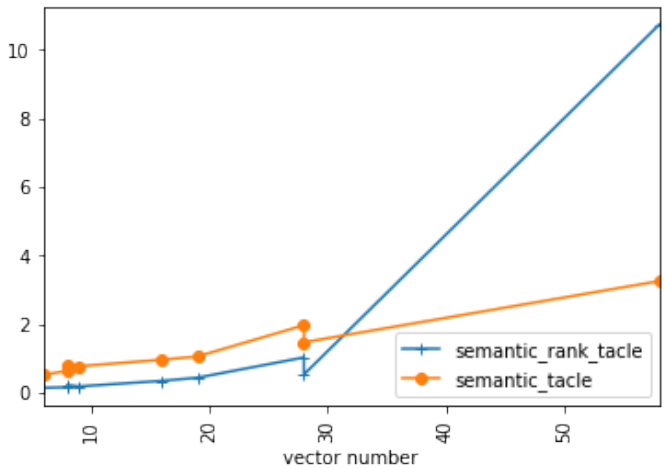
Predictability of Constraint from the Header Text:

Classifier	Accuracy	Precision	Recall
DecisionTree (depth= 4, feature= 100)	0.927	0.755	0.907
DecisionTree (depth= 4, feature= 200)	0.889	0.59	0.815
LinearSVM (loss ='squared hinge', C=1.0)	0.918	0.672	0.79
RBF SVM (gamma=1.0, C=1.0)	0.942	0.765	0.836

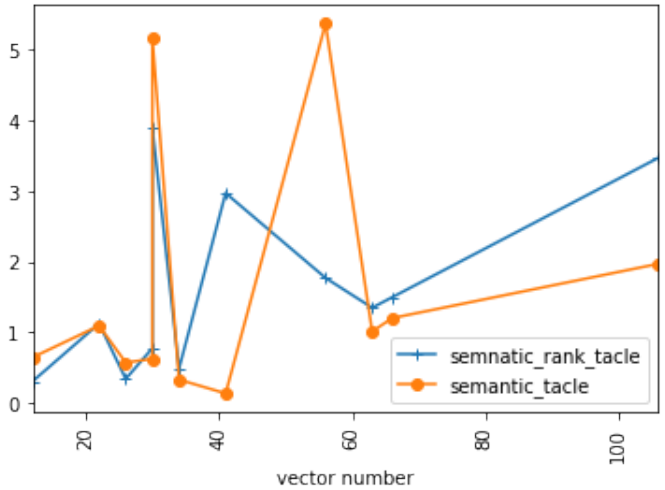
Table: Leave one out validation score for different algorithm accepted with their best learn parameter

2. Effectiveness of Existence Prediction *(Cont.)*

Predictability of Constraint from the Header Text:



(a) Synthetic Spreadsheet



(b) Real Spreadsheets

Figure: Computation time Semantic-TaCLe without Classifier vs Semantic-TaCLe

3. Effectiveness of Proposed Approach

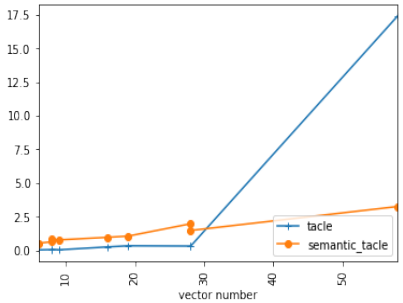
Influence in Precision and Recall:

	Precision	Recall
TaCLe	0.857	1.0
Semantic-TaCLe without classifier	0.846	0.917
Semantic-TaCLe (classifier + ranking)	0.923	1.0

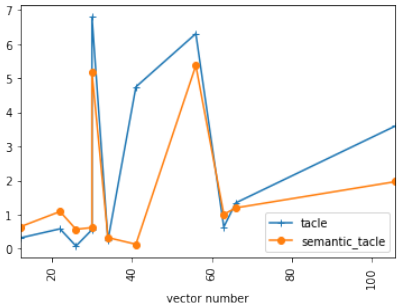
Table: Performance metric of TaCLe, Semantic-TaCLe without classifier and Semantic-TaCLe in terms of precision and recall.

3. Effectiveness of Proposed Approach *(Cont.)*

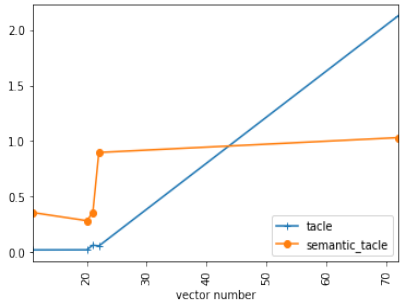
Influence in Time:



(a) Synthetic Spreadsheet



(b) Real Spreadsheet



(c) Test Spreadsheet

Figure: Computation time comparison between TaCLe and Semantic TaCLe

	Synthetic Spreadsheet	Real Spreadsheet	Test Spreadsheet
TaCLe	2.091	2.294	0.4568
Semantic-TaCLe	1.275	1.646	0.5846

Table: Run-time Comparison in seconds

Applications

- Formula suggestions
- Error Correction
- Auto-completion

Future Work

- Predicting constraint only using header labels and classifier
- Trained word-embedding for spreadsheet words
- Relational formula prediction from header text

Thank You!

ANY QUESTIONS?