Robust Machine-Learning Approaches for Efficient Functional Dependency Approximation

Philipp Jung

Beuth University of Applied Sciences

philippjung@posteo.de

November 27, 2019

Overview

Motivation

Example of an FD

	left hand side			right hand side
Id	First Name	Surname	Zip	Town
1	Alice	Smith	19139	Munich
2	Peter	Meyer	19139	Munich
3	Ana	Parker	19139	Munich
4	John	Pick	12055	Berlin
5	John	Pick	19139	Munich

Table: Example of the non-minimal FD
{Id, First Name, Surname, Zip} $\rightarrow$ Town.

FD Fields of Application

FDs are constraints on a relational scheme commonly used for

- schema normalization of relational databases
- data cleaning, e.g. HoloClean[1]
- data exploration

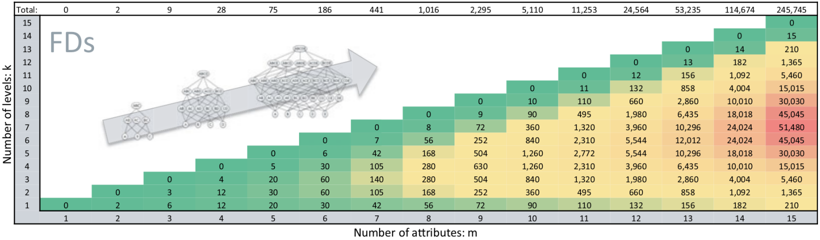
[1] Heidari et al. 2019.

Challenges of FD Detection

Remark

FD detection is a particularly complex problem to solve.

FD Detection Search Space Size



Total:	0	2	9	28	75	186	441	1,016	2,295	5,110	11,253	24,564	53,235	114,674	245,745
15															0
14														0	15
13													0	14	210
12												0	13	182	1,365
11											0	12	156	1,092	5,460
10										0	11	132	858	4,004	15,015
9									0	10	110	660	2,860	10,010	30,030
8								0	9	90	495	1,980	6,435	18,018	45,045
7							0	8	72	360	1,320	3,960	10,296	24,024	51,480
6						0	7	56	252	840	2,310	5,544	12,012	24,024	45,045
5					0	6	42	168	504	1,260	2,772	5,544	10,296	18,018	30,030
4				0	5	30	105	280	630	1,260	2,310	3,960	6,435	10,010	15,015
3			0	4	20	60	140	280	504	840	1,320	1,980	2,860	4,004	5,460
2		0	3	12	30	60	105	168	252	360	495	660	858	1,092	1,365
1	0	2	6	12	20	30	42	56	72	90	110	132	156	182	210
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Number of attributes: m

The number of FD candidates for m attributes is $\mathcal{O}(\frac{m}{2} \cdot 2^m)$.

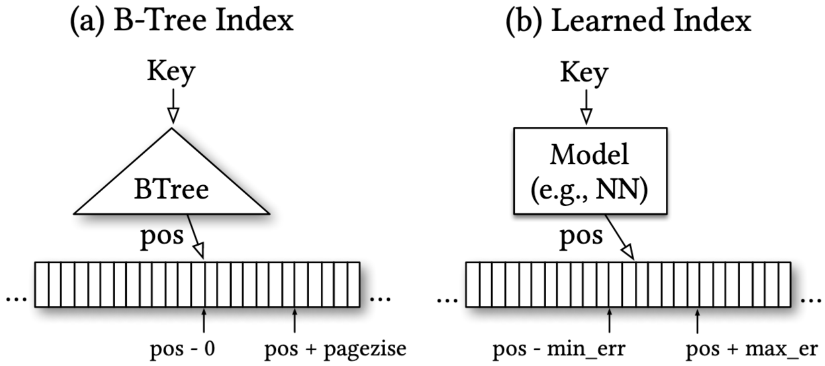
[1]Image from Abedjan et al. 2019

Learned Algorithms

Kraska et al. showed in 2018 that a Binary Tree can be interpreted as a learned index structure.[2]

[2] Kraska et al. 2018.

BTree as a Learned Model



[2] Image from Kraska et al. 2018

Learning FDs

Remark

Learned algorithms offer a new research-directory when solving old algorithmic problems.

Objectives

Research Objectives

- Interpret FDs as *learned* constraints
- Train data imputation models with DataWig and benchmark them against FD-based models
- Identify similarities and leverage them to derive FDs from learned models
- Detect minimal FDs

DataWig

- DataWig is a framework for learning models to impute missing values in tables
- Data imputation: Replace missing or faulty data
- Models trained by DataWig use either regression or multi-label classification
- DataWig models can be benchmarked against FD Imputer

FD Imputer

FD Imputer: FDs as Models

- Interpret FDs as rules for data imputation
- Write FD Imputer: An imputation model entirely based on FDs
- Measure *Robustness*: Either the F1-Score or the MSE that FD Imputer obtains for a FD

How FD Imputer works

1. Split dataset in train-set and test-set
2. Detect FDs on train-set using HyFD[3]
3. For each FD, impute the right hand side for each row in the **test**-set. Do so by finding a tuple with equal left hand side in the **train**-set
4. Compute *Robustness* by evaluating FD Imputer's performance for each FD (F1-Score or Mean Squared Error)

[3] Papenbrock and Naumann 2016.

FD Imputer Functionality Example

A	B	C	D
Green	Bus	Portugal	Lisbon
Yellow	Car	Portugal	Lisbon

(a) Train set

A	B	C	D
Yellow	Bus	Spain	?
Blue	Bus	Portugal	?

(b) Test set

In this example, FD Imputer uses the FD C $\rightarrow$ D.

FD Imputer Functionality Example

A	B	C	D
Green	Bus	Portugal	Lisbon
Yellow	Car	Portugal	Lisbon

(a) Train set

A	B	C	D
Yellow	Bus	Spain	-
Blue	Bus	Portugal	Lisbon

(b) Test set

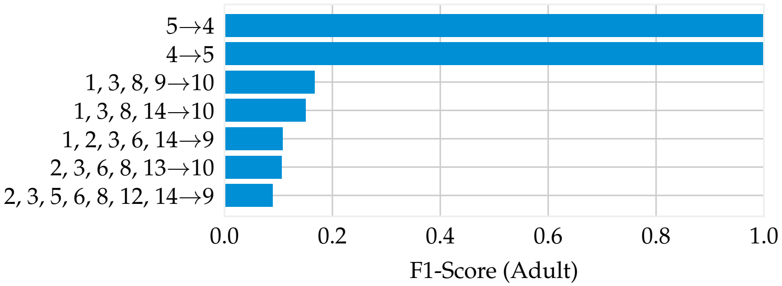
In this example, FD Imputer uses the FD C $\rightarrow$ D.

Benchmarking FD Imputer

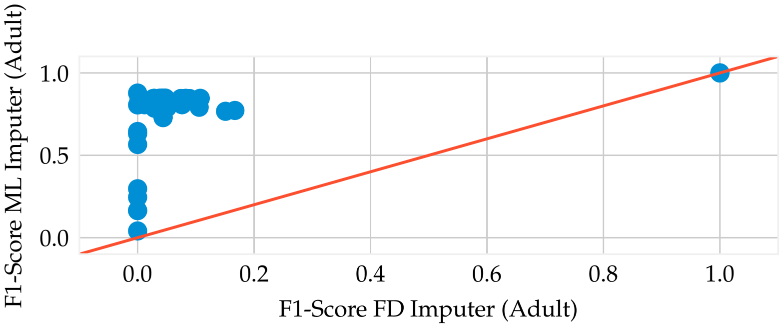
Dataset	$\#FDs_{train}$	$\#FD\ (F1 = 0)$	$F1_{mean}$	$F1_{max}$
Abalone	193	45	0.0008	0.0048
Adult	88	10	0.0669	1.0000
Balance S.	7	6	0.0000	0.0000
Breast C. W.	77	10	0.2198	0.7539
Chess	9	8	0.0000	0.0000
Iris	8	1	0.1274	0.2252
Letter	80	17	0.2347	0.3737
Nursery	11	10	0.0000	0.0000

Table: Performance of the FD Imputer on a selection of UCI datasets.

Benchmarking FD Imputer: Ranking for Robustness



Benchmarking FD Imputer: Comparison with DataWig Model



Summary of FD Imputer Results

- Some FDs are more robust than others
- FD Imputer performs generally worse than the model trained with DataWig
- This concerns only classifiable data – it is generally impossible to impute continuous numerical data with FD Imputer

FD Imputer is Overfitting by Design

Haykin 2008

"[Overfitting] is essentially a 'look-up table', which implies that the input-output mapping [. . .] is not smooth."[4]

[4] Haykin 2008, p. 165.

FD Imputer is Overfitting by Design

Haykin 2008

"[Overfitting] is essentially a 'look-up table', which implies that the input-output mapping [. . .] is not smooth."[4]

- Due to the implementation of FD Imputer, the train-set is merely a table to look up imputation values
- No generalization takes place whatsoever
- No empirical risk minimization (ERM) is applied!

[4] Haykin 2008, p. 165.

Deriving FDs from Trained Models

- When overfitting DataWig models, one cannot control that overfitting takes place on LHS FD-attributes

Proposition

In consequence, it does not appear to be possible to derive FDs with ERM-based imputation models.

Deriving FDs from Trained Models

But what about Relaxed Functional Dependencies (RFDs)?

DepDetector

RFDs as Models

- An RFD is based on the definition of an FD
- It alters that definition to serve a specific purpose
- There are many RFDs defined, such as Metric Functional Dependencies, Conditional Functional Dependencies or Approximate Functional Dependencies.

Example of an RFD

Example

Koudas et al. introduce Metric Functional Dependencies (MFDs) to find constraints on tables that contain rows with slightly different formatting or slightly deviating values.[5]

[5] Koudas et al. 2009.

Example for Noisy Data

Id	First name	Last name	Zip	Town
1	Alice	Smith	19139	Munich
2	**Peter**	**Meyer**	**19139**	**Muinch**
3	Ana	Parker	19139	Munich
4	John	Pick	12055	Berlin

The FD Zip $\rightarrow$ Town is violated by the second row. A MFD that considers the Levenshtein distance between two entries rather than exact equality can still detect the constraint.

RFDs as Models

- Due to their their training being ERM-based, DataWig models are capable of learning constraints on noisy data as well
- During training, DataWig models “learn” relaxations – there is no need to manually set up a threshold of a Levenshtein distance as in the previous example

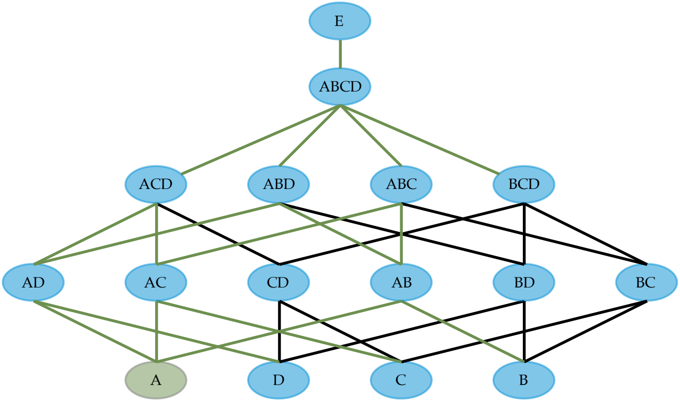
RFDs as Models

- Due to their their training being ERM-based, DataWig models are capable of learning constraints on noisy data as well
- During training, DataWig models “learn” relaxations – there is no need to manually set up a threshold of a Levenshtein distance as in the previous example

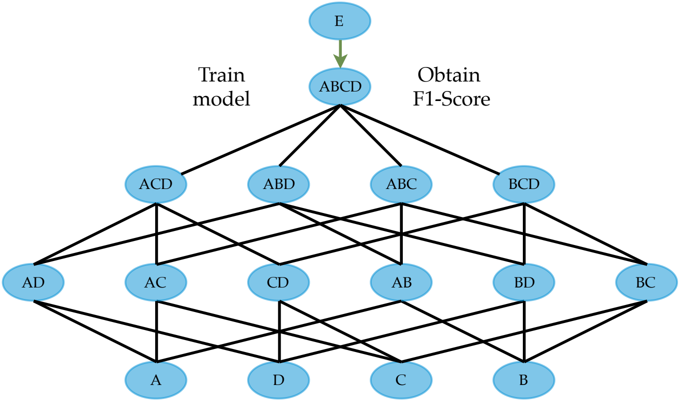
Question

How does one find a *minimal* dependency with this approach?

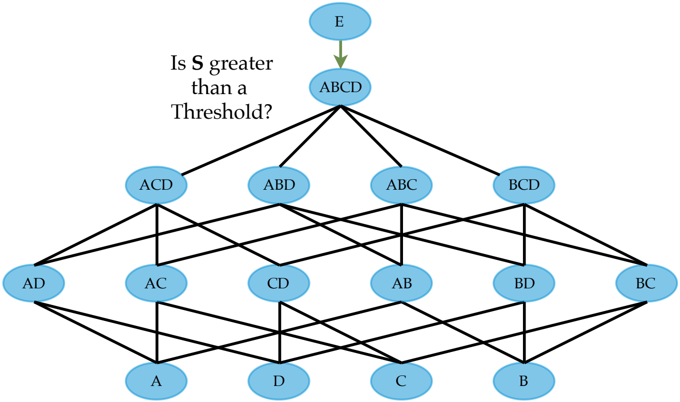
Functionality of DepDetector



Functionality of DepDetector

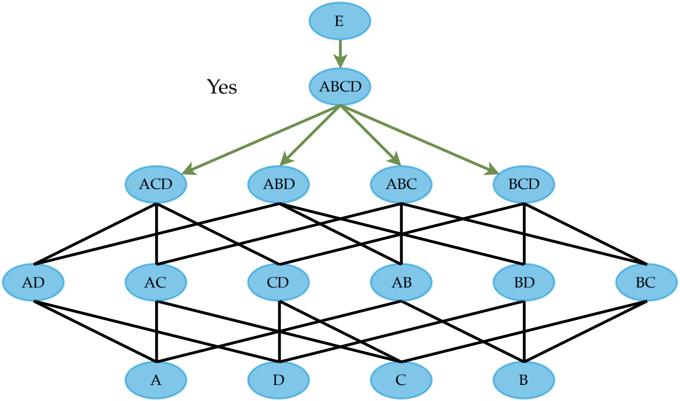


Functionality of DepDetector

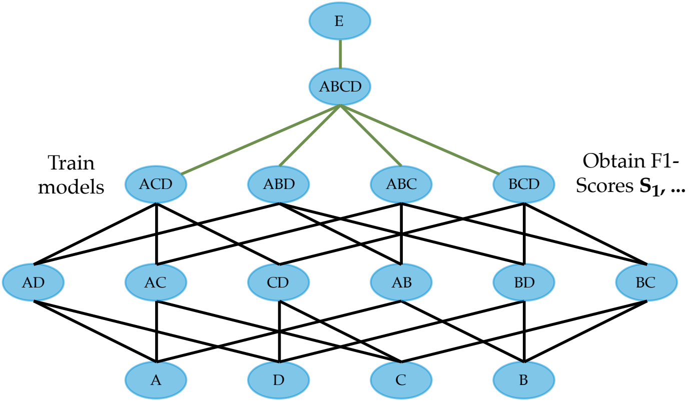


Functionality of DepDetector

Yes

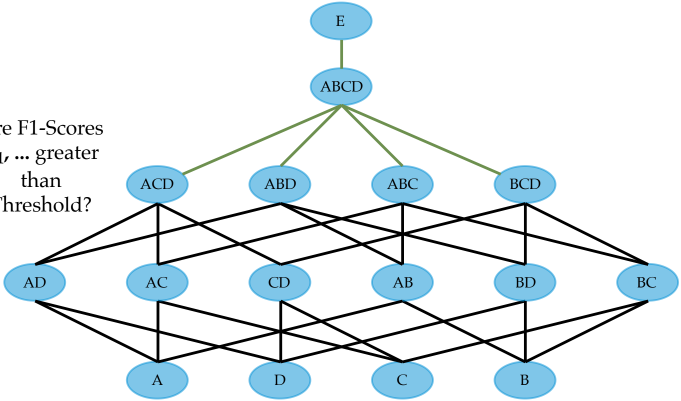


Functionality of DepDetector

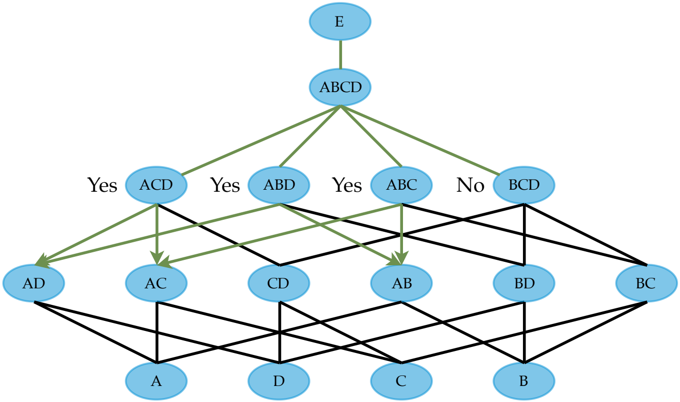


Functionality of DepDetector

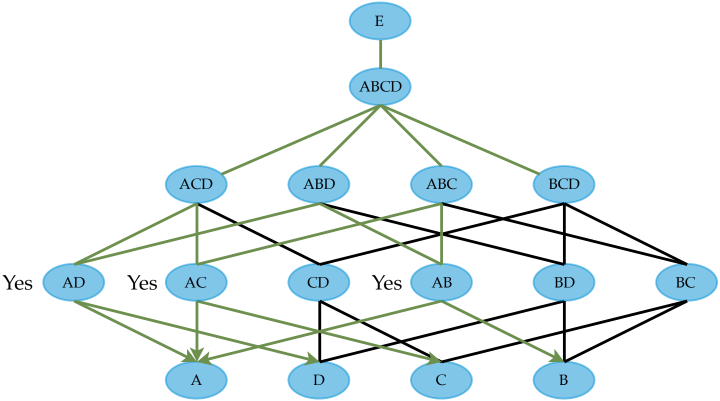
Are F1-Scores **S_1, ...** greater than Threshold?



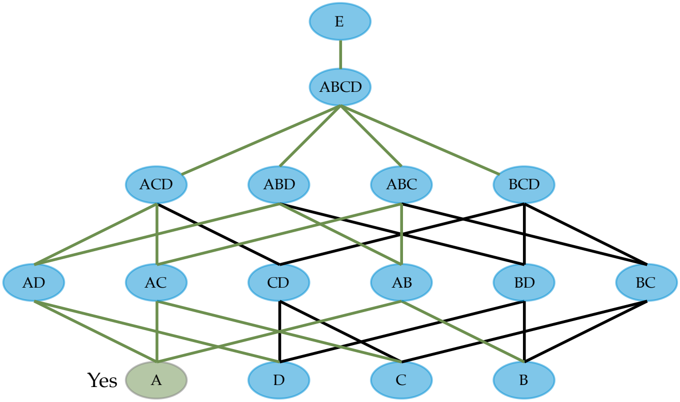
Functionality of DepDetector



Functionality of DepDetector



Functionality of DepDetector



DepDetector Properties

- DepDetector solves an optimization problem on a directed graph
- DepDetector depends on one threshold for classifiable data and one threshold for continous numerical data

DepDetector Properties

Proposition

Machine learning classifier/regressor models may have the potential to unify a number of RFDs.

Summary of DepDetector Results

Dataset	Cols	Rows	# FDs	Greedy dependencies	Complete dependencies
Abalone	10	4177	175	7 (42 min)	TL
Adult	16	32561	93	TL	TL
Balance-S.	6	625	7	3 (67 s)	3 (80 s)
Chess	8	28056	9	1 (117 min)	1 (340 min)
Iris	6	150	9	5 (38 s)	8 (43s)
Letter	18	20000	78	TL	TL
Nursery	11	12960	11	3 (110 min)	TL

'TL' indicates a time limit of 350 min.

Prospects

Prospects for further Research

- Minimal dependency detection algorithms for learned relaxations can be further optimized (Concurrency, applying Graph-Theory)
- A complexity analysis for DepDetector can be performed
- Approaches to calculate thresholds from the data can be introduced

Thank you for your attention!

References I

Ziawasch Abedjan et al. *Data Profiling*. 2019. isbn: 9781681734477. doi: https://doi.org/10.2200/S00878ED1V01Y201810DTM052.

Simon Haykin. *Neural Networks and Learning Machines Third Edition*. Pearson Prentice Hall, 2008. isbn: 9780131471399.

References II

Alireza Heidari et al. "HoloDetect: Few-Shot Learning for Error Detection". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. ACM, 2019, pp. 829–846. isbn: 978-1-4503-5643-5. doi: 10.1145/3299869.3319888. url: http://doi.acm.org/10.1145/3299869.3319888.

N. Koudas et al. "Metric Functional Dependencies". In: (Mar. 2009), pp. 1275–1278. issn: 1063-6382. doi: 10.1109/ICDE.2009.219.

References III

Tim Kraska et al. "The Case for Learned Index Structures". In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA: ACM, 2018, pp. 489–504. isbn: 978-1-4503-4703-7. doi: 10.1145/3183713.3196909. url: http://doi.acm.org/10.1145/3183713.3196909.

Thorsten Papenbrock and Felix Naumann. "A Hybrid Approach to Functional Dependency Discovery". In: SIGMOD '16 (2016), pp. 821–833. doi: 10.1145/2882903.2915203. url: http://doi.acm.org/10.1145/2882903.2915203.

Benchmarking FD Imputer Continuous Data

Dataset	# $\text{cFDs}_{\text{train}}$	# 0-Coverage cFDs	Coverage (%)
Abalone	139	84	0.1277
Adult	11	5	0.1217
Balance S.	1	1	0.0000
Breast C. W.	1	1	0.0000
Chess	1	1	0.0000
Iris	4	4	0.0000
Letter	0	0	-
Nursery	1	1	0.0000

Table: Imputation coverage of FD Imputer on all UCI datasets for which FDs with continuous data in the RHS were detected.

FD Imputer Continuous Data Definitions

$$\text{mean missing values per cFD} = \sum_i \frac{\text{missing imputations}}{\text{cFD}_i} \cdot (\#\ \text{cFDs})^{-1}$$

$$\text{mean coverage} = \left(1 - \frac{\text{mean missing values per cFD}}{\#\ \text{rows in } r_{test}}\right) \cdot 100$$

Machine Learning Models

data representation

↓

model class

↓

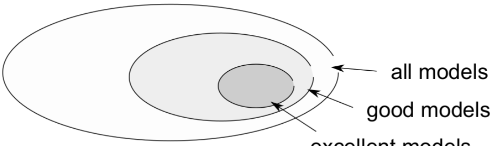
performance measure

↓

optimization

↓

validation



[5] Image from Prof. Obermeyer, Neural Information Processing Group TU Berlin

Definition Robustness Continous Data

If A contains continuous numerical data, the MSE is determined to measure robustness:

$$\text{robustness}_{\text{MSE}} = \frac{1}{p-m} \sum_{i=m+1}^{p} \left(t_i^*[A] - t_i'[A]\right)^2 . \qquad (1)$$

Here, $p \in \mathbb{N}$ denotes the number of tuples in a relational instance and m is the number of imputed tuples.

Definition Robustness Classifiable Data

If A contains classifiable data, the F1-Score is calculated to measure robustness:

$$\text{robustness}_{\text{F1-Score}} = \left(\frac{Recall(r_{imp}, r_{test})^{-1} + Precision(r_{imp}, r_{test})^{-1}}{2} \right)^{-1} \quad (2)$$

Empirical Risk Minimization

- We cannot know how an algorithm will be influenced when it is used in practice (*risk*)
- We thus simulate the fact that the true distribution of data is unknown by creating models on a subset of the whole dataset (*empirical risk*)
- By minimizing the empirical risk and preventing overfitting by cross-validation or train/test validation, we make models resistant towards noise

ERM and Overfitting