Advanced Data Management (CSCI 640/490)

Data Cleaning

Dr. David Koop

Three Ways to Present the Same Data

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Initial Data

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Transpose

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Tidy Data

[H. Wickham, 2014]

Messy Dataset Problems

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of observational units are stored in the same table
- A single observational unit is stored in multiple tables

Problem: Variables stored in both rows & columns

Mexico Weather, Global Historical Climatology Network

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

[H. Wickham, 2014]

Problem: Variables stored in both rows & columns

Mexico Weather, Global Historical Climatology Network

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

Variable in columns: day; Variable in rows: tmax/tmin

[H. Wickham, 2014]

Unpivot/Melt

- Many columns (wider) become two columns (longer): one with column name (variable), other with value

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
str	i32	i32	str	f64	f64	f64	f64	f64	f64	f64	f64
"MX000017004"	1955	4	"tmax"	31.0	31.0	31.0	32.0	33.0	32.0	32.0	33.0
"MX000017004"	1955	4	"tmin"	15.0	15.0	16.0	15.0	16.0	16.0	16.0	16.0
"MX000017004"	1955	5	"tmax"	31.0	31.0	31.0	30.0	30.0	30.0	31.0	31.0
"MX000017004"	1955	5	"tmin"	20.0	16.0	16.0	15.0	15.0	15.0	16.0	16.0
"MX000017004"	1955	6	"tmax"	30.0	29.0	28.0	27.0	28.0	26.0	23.0	27.0
…	…	…	…	…	…	…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
"MX000017004"	2011	3	"tmax"	NaN	NaN	NaN	NaN	33.2	NaN	NaN	NaN
"MX000017004"	2011	3	"tmin"	NaN	NaN	NaN	NaN	14.8	NaN	NaN	NaN
"MX000017004"	2011	4	"tmax"	NaN	35.0	NaN	NaN	NaN	NaN	NaN	NaN
"MX000017004"	2011	4	"tmin"	NaN	16.8	NaN	NaN	NaN	NaN	NaN	NaN

id	year	month	element	variable	value
str	i32	i32	str	str	f64
"MX000017004"	1955	4	"tmax"	"d1"	31.0
"MX000017004"	1955	4	"tmin"	"d1"	15.0
"MX000017004"	1955	5	"tmax"	"d1"	31.0
"MX000017004"	1955	5	"tmin"	"d1"	20.0
"MX000017004"	1955	6	"tmax"	"d1"	30.0
…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	"d31"	NaN
"MX000017004"	2011	3	"tmax"	"d31"	36.5
"MX000017004"	2011	3	"tmin"	"d31"	17.0
"MX000017004"	2011	4	"tmax"	"d31"	NaN
"MX000017004"	2011	4	"tmin"	"d31"	NaN

Unpivot/Melt

- Many columns (wider) become two columns (longer): one with column name (variable), other with value

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
str	i32	i32	str	f64	f64	f64	f64	f64	f64	f64	f64
"MX000017004"	1955	4	"tmax"	31.0	31.0	31.0	32.0	33.0	32.0	32.0	33.0
"MX000017004"	1955	4	"tmin"	15.0	15.0	16.0	15.0	16.0	16.0	16.0	16.0
"MX000017004"	1955	5	"tmax"	31.0	31.0	31.0	30.0	30.0	30.0	31.0	31.0
"MX000017004"	1955	5	"tmin"	20.0	16.0	16.0	15.0	15.0	15.0	16.0	16.0
"MX000017004"	1955	6	"tmax"	30.0	29.0	28.0	27.0	28.0	26.0	23.0	27.0
…	…	…	…	…	…	…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
"MX000017004"	2011	3	"tmax"	NaN	NaN	NaN	NaN	33.2	NaN	NaN	NaN
"MX000017004"	2011	3	"tmin"	NaN	NaN	NaN	NaN	14.8	NaN	NaN	NaN
"MX000017004"	2011	4	"tmax"	NaN	35.0	NaN	NaN	NaN	NaN	NaN	NaN
"MX000017004"	2011	4	"tmin"	NaN	16.8	NaN	NaN	NaN	NaN	NaN	NaN

id	year	month	element	variable	value
str	i32	i32	str	str	f64
"MX000017004"	1955	4	"tmax"	"d1"	31.0
"MX000017004"	1955	4	"tmin"	"d1"	15.0
"MX000017004"	1955	5	"tmax"	"d1"	31.0
"MX000017004"	1955	5	"tmin"	"d1"	20.0
"MX000017004"	1955	6	"tmax"	"d1"	30.0
…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	"d31"	NaN
"MX000017004"	2011	3	"tmax"	"d31"	36.5
"MX000017004"	2011	3	"tmin"	"d31"	17.0
"MX000017004"	2011	4	"tmax"	"d31"	NaN
"MX000017004"	2011	4	"tmin"	"d31"	NaN

Pivot

- Inverse of unpivot: two columns (longer) become many columns (wider) one column becomes column names (variable), other becomes values

id	year	month	element	variable	value
str	i32	i32	str	str	f64
"MX000017004"	1955	4	"tmax"	"d1"	31.0
"MX000017004"	1955	4	"tmin"	"d1"	15.0
"MX000017004"	1955	5	"tmax"	"d1"	31.0
"MX000017004"	1955	5	"tmin"	"d1"	20.0
"MX000017004"	1955	6	"tmax"	"d1"	30.0
…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	"d31"	NaN
"MX000017004"	2011	3	"tmax"	"d31"	36.5
"MX000017004"	2011	3	"tmin"	"d31"	17.0
"MX000017004"	2011	4	"tmax"	"d31"	NaN
"MX000017004"	2011	4	"tmin"	"d31"	NaN

id	year	month	variable	tmax	tmin
str	i32	i32	str	f64	f64
"MX000017004"	1955	4	"d1"	31.0	15.0
"MX000017004"	1955	5	"d1"	31.0	20.0
"MX000017004"	1955	6	"d1"	30.0	16.0
"MX000017004"	1955	7	"d1"	27.0	15.0
"MX000017004"	1955	8	"d1"	23.0	14.0
…	…	…	…	…	…
"MX000017004"	2010	12	"d31"	NaN	NaN
"MX000017004"	2011	1	"d31"	NaN	NaN
"MX000017004"	2011	2	"d31"	NaN	NaN
"MX000017004"	2011	3	"d31"	36.5	17.0
"MX000017004"	2011	4	"d31"	NaN	NaN

Pivot

- Inverse of unpivot: two columns (longer) become many columns (wider) one column becomes column names (variable), other becomes values

id	year	month	element	variable	value
str	i32	i32	str	str	f64
"MX000017004"	1955	4	"tmax"	"d1"	31.0
"MX000017004"	1955	4	"tmin"	"d1"	15.0
"MX000017004"	1955	5	"tmax"	"d1"	31.0
"MX000017004"	1955	5	"tmin"	"d1"	20.0
"MX000017004"	1955	6	"tmax"	"d1"	30.0
…	…	…	…	…	…
"MX000017004"	2011	2	"tmin"	"d31"	NaN
"MX000017004"	2011	3	"tmax"	"d31"	36.5
"MX000017004"	2011	3	"tmin"	"d31"	17.0
"MX000017004"	2011	4	"tmax"	"d31"	NaN
"MX000017004"	2011	4	"tmin"	"d31"	NaN

id	year	month	variable	tmax	tmin
str	i32	i32	str	f64	f64
"MX000017004"	1955	4	"d1"	31.0	15.0
"MX000017004"	1955	5	"d1"	31.0	20.0
"MX000017004"	1955	6	"d1"	30.0	16.0
"MX000017004"	1955	7	"d1"	27.0	15.0
"MX000017004"	1955	8	"d1"	23.0	14.0
…	…	…	…	…	…
"MX000017004"	2010	12	"d31"	NaN	NaN
"MX000017004"	2011	1	"d31"	NaN	NaN
"MX000017004"	2011	2	"d31"	NaN	NaN
"MX000017004"	2011	3	"d31"	36.5	17.0
"MX000017004"	2011	4	"d31"	NaN	NaN

Types of Transformations

1 Pivot

Name	Alice
Age	17
Name	Bob
Age	19

→

Name	Age
Alice	17
Bob	19

2 Transpose

Name	Alice	Bob
Age	17	19

→

Name	Age
Alice	17
Bob	19

3 Subtitle

Name	Age
Student	
Alice	17
Bob	19
Teacher	
Claire	32

→

Name	Age	Role
Alice	17	Student
Bob	19	Student
Claire	32	Teacher

4 Ffill

Role	Name
Student	Alice
	Bob
Teacher	Claire
	David

→

Role	Name
Student	Alice
Student	Bob
Teacher	Claire
Teacher	David

5 Explode

Teacher	Student
Claire	Alice, Bob
David	Eva, Fiona

→

Teacher	Student
Claire	Alice
Claire	Bob
David	Eva
David	Fiona

6 Stack

Name	2013	2014	2015
Alice	A	B	A

→

Name	Year	Grade
Alice	2013	A
Alice	2014	B
Alice	2015	A

7 Wide_to_long

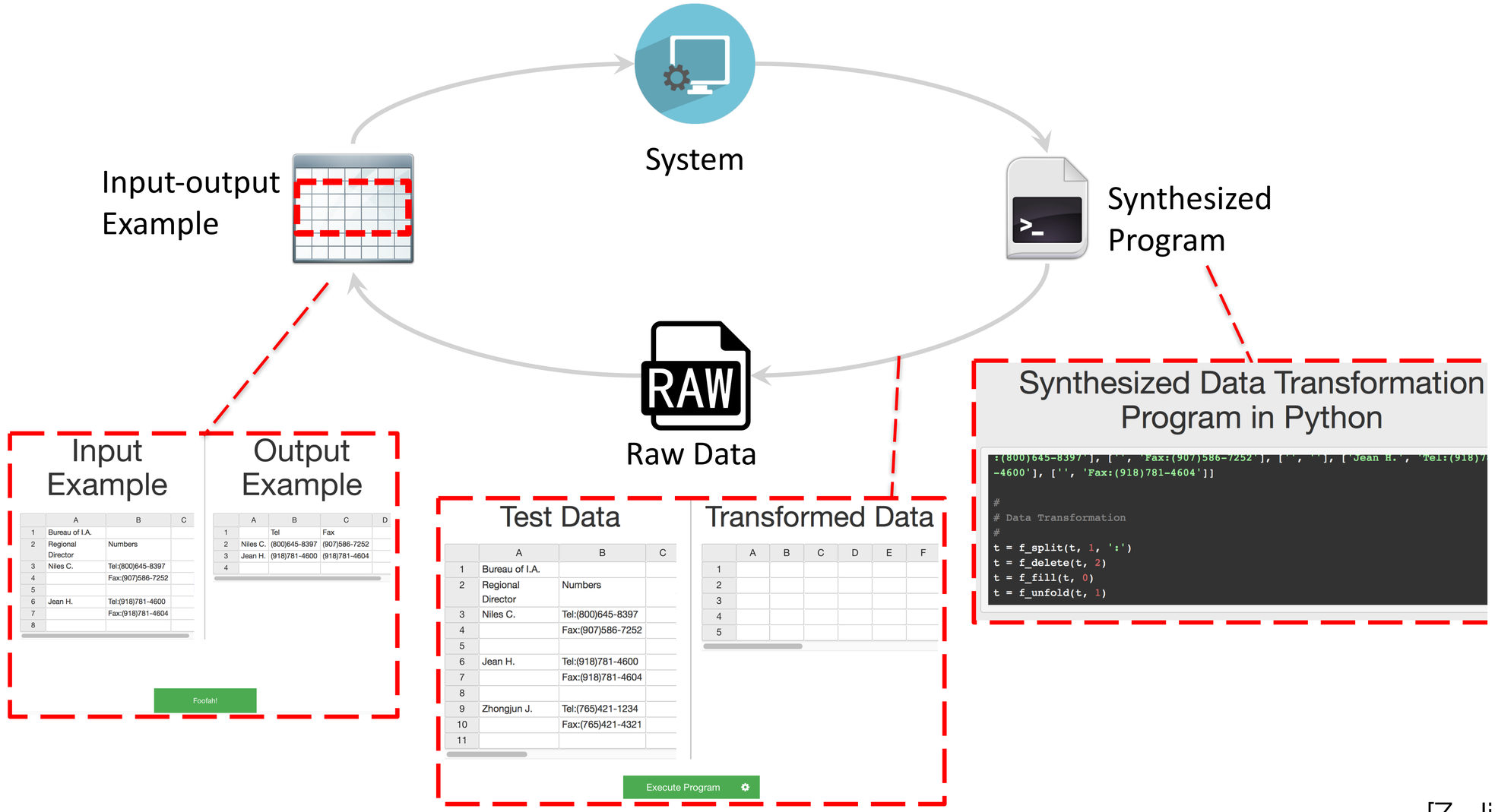
Name	2013 English	2014 English	2015 English	2013 Math	2014 Math	2015 Math
Alice	A	B	A	B	B	A

→

Name	Year	Course	Grade
Alice	2013	English	A
Alice	2014	English	B
Alice	2015	English	A
Alice	2013	Math	B
Alice	2014	Math	B
Alice	2015	Math	A

[Z. Huang & E. Wu, 2024]

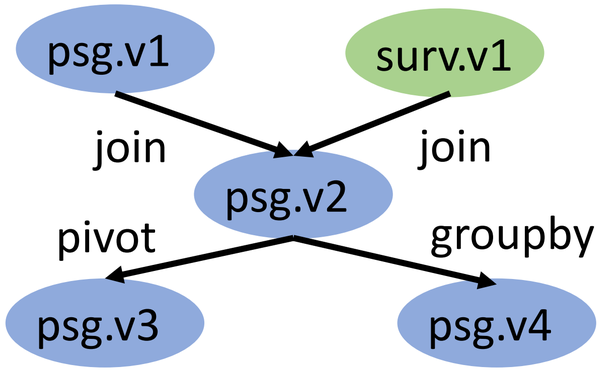
Foofah Design: Programming by Example



[Z. Jin et al., 2017]

AutoSuggest: Mine Data Pipelines

- Crawl, reapply, and analyze data pipelines from Jupyter+pandas
- Tasks:
  - Single-Operator Prediction: Given two tables and an operation, decide how to best apply the operation (what are the parameters)
  - Next-Operator Prediction: Given all operations performed so far, predict the next one




```
1 import pandas as pd
2
3 psg=pd.read_csv('passenger_data.csv')
4 surv=pd.read_csv('survive.csv')
5 psg=psg.merge(surv,on='PassengerId',
          how='left')
6 psg.pivot(header=['Survived, Pclass'],
          index='Sex', aggrfunc='count')
7 psg.groupby('Sex',aggrfunc='count')
```

[C. Yan & Y. He]

Pivot/Unpivot

- Pivot is hard to get right
  - Index
  - Header
  - Aggregation Function
  - Aggregation Columns
- Use GroupBy Prediction
- Look for NULLs and use **affiinity**
- Affinity-Maximizing Pivot Table
- Unpivot requires **compatibility**

Sector	Ticker	Company	Year	Quarter	Market Cap	Revenue
Aerospace	AJRD	AEROJET ROCKETD	2006	Q1	1442.67	472.07
Aerospace	AJRD	AEROJET ROCKETD	2006	Q2	1514.80	489.22
...	...	...	...	...	...	...
Aerospace	BA	BOEING CO	2006	Q1	343.41	210.66
...	...	...	...	...	...	...
Utilities	YORW	YORK WATER CO	2008	Q4	600.19	271.73

Sector	Ticker	Company	2006	2007	2008
Aerospace	AJRD	AEROJET ROCKETD	6218.09	6342.45	7088.62
	ATRO	ASTRONICS CORP	1050.97	1071.99	1198.11
Business Services	HHS	HARTE-HANKS INC	2473.75	2523.22	2820.07
	NCMI	NATL CINEMEDIA	856.92	874.06	976.89
Consumer Staples	YTEN	TIELD10 BIOSCI	533.13	543.79	607.77
	...	...	...	...	...
Utilities	YORW	YORK WATER CO	1902.37	1940.42	2168.70

Ticker	Company	Year	Aerospace	Business Services	...	Utilities
AJRD	AEROJET ROCKETD	2006	6218.09	NULL	...	NULL
AJRD	AEROJET ROCKETD	2007	6342.45	NULL	...	NULL
AJRD	AEROJET ROCKETD	2008	7088.62	NULL	...	NULL
ATRO	ASTRONICS CORP	2006	1050.97	NULL	...	NULL
...	...	...	...	...	...	...
HHS	HARTE-HANKS INC	2006	NULL	2473.75	...	NULL
...	...	...	...	...	...	...
YORW	YORK WATER CO	2008	NULL	NULL	...	2168.7

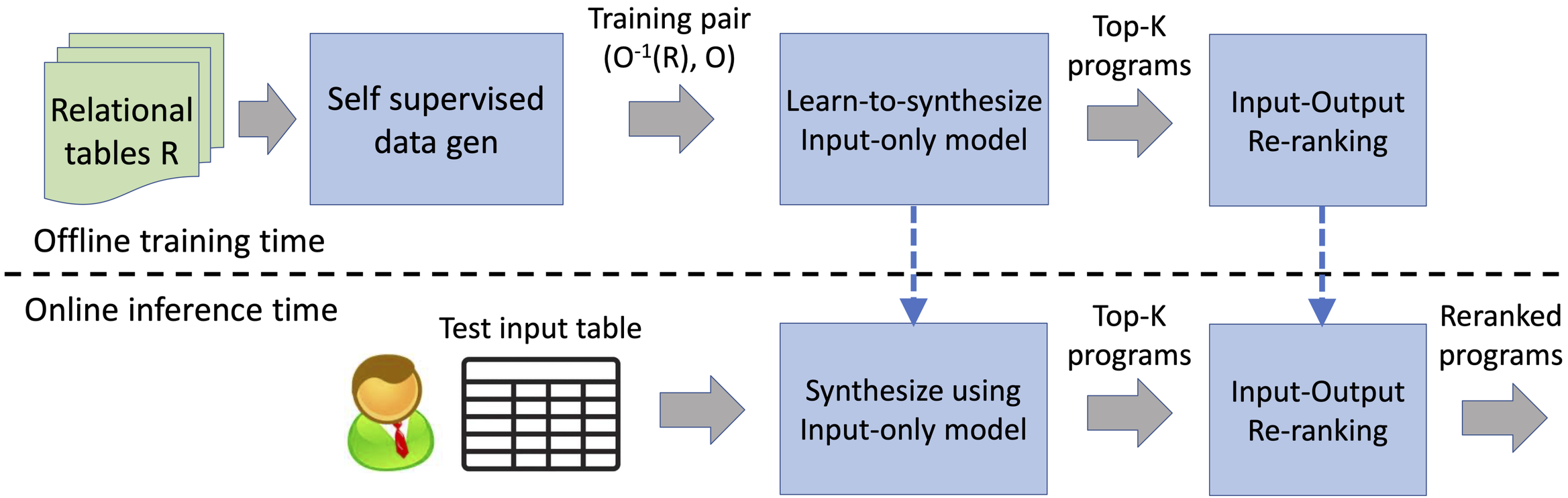
[C. Yan & Y. He]

AutoTables

- Problem:
  - Non-relational tables are common but hard to query
  - Non-relational tables are hard to "relationalize" (aka tidy)
- Steps:
  1. Identify structural issues
  2. Map the visual pattern to an operator
  3. Parameterize the operator correctly
  4. Potentially add more operators (go back to 1 or 2)
- Solution: Use LLMs to relationalize tables

[P. Li et al., 2023]

AutoTables Architecture



[P. Li et al., 2023]

Test 1

- Wednesday, October 8, 12:30-1:45pm in PM 103
- In-Class, paper/pen & pencil
- Covers material through this week
- Format:
  - Multiple Choice
  - Free Response
  - One extra 2-sided page for CSCI 640 Students
- Info will be on the course webpage

Assignment 3

- Upcoming, won't be due until after the first test

Data Cleaning

Data Cleaning Types

- How can statistical techniques improve efficiency or reliability of data cleaning? (Data Cleaning **with** Statistics)
  - Example: Trifacta
- How how can we improve the reliability of statistical analytics with data cleaning? (Data Cleaning **for** Statistics)
  - Example: SampleClean

[D. Haas et al., 2016]

Misconceptions about Data Cleaning

- Surveyed Technology Professionals
- The end goal of data cleaning is clean data
  - "We typically clean our data until the desired analytics works without error."
- Data cleaning is a sequential operation
  - "[It's an] iterative process, where I assess biggest problem, devise a fix, re-evaluate. It is dirty work."
- Data cleaning is performed by one person
  - "There are often long back and forths with senior data scientists, devs, and the business units that provided the data on data quality."

[D. Haas et al., 2016]

Misconceptions about Data Cleaning

- Data quality is easy to evaluate
  - "I wish there were a more rigorous way to do this but we look at the models and guess if the data are correct"
  - "Other than common sense we do not have a procedure to do this"
  - "Usually [a data error] is only caught weeks later after someone notices."

[D. Haas et al., 2016]

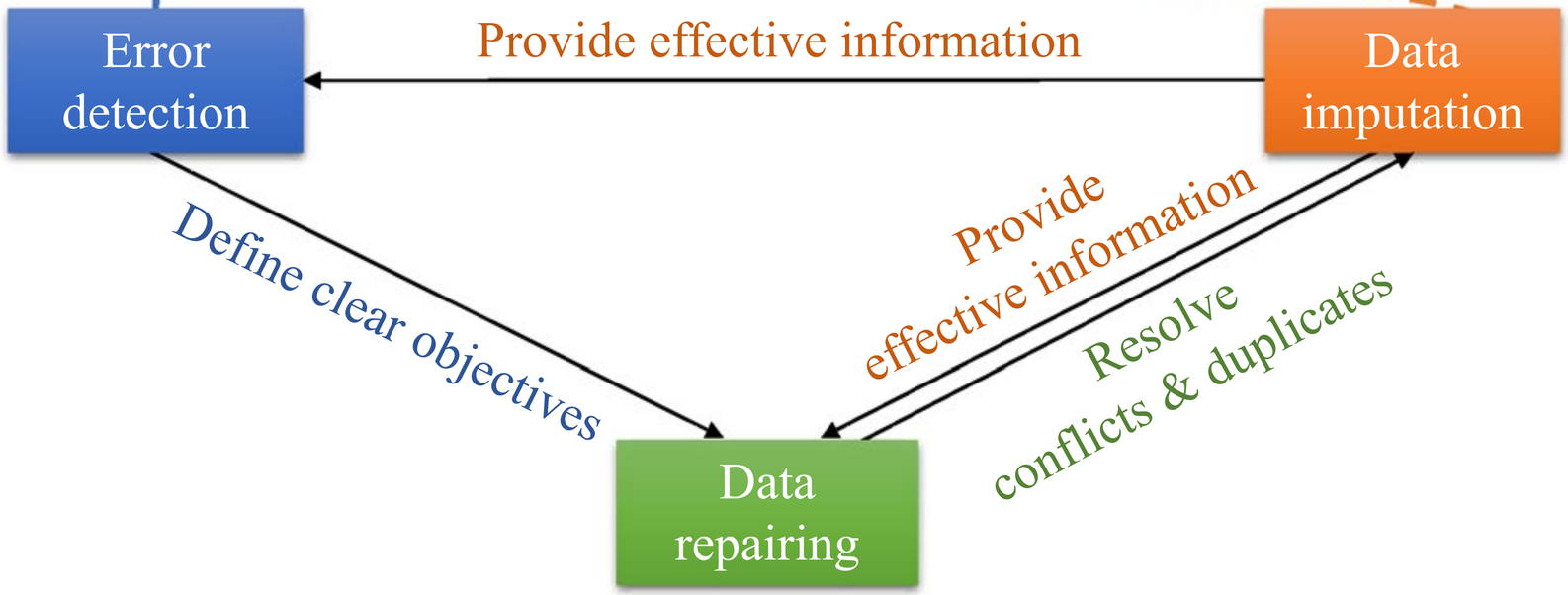
Data Cleaning

- Three key tasks:
  - Error Detection: Identify potentially erroneous values within the data
  - Data Repairing: Resolve conflicts and duplicates
  - Data Imputation: Fill in missing values
- These tasks are not unrelated

[J. Zhu et al., 2024]

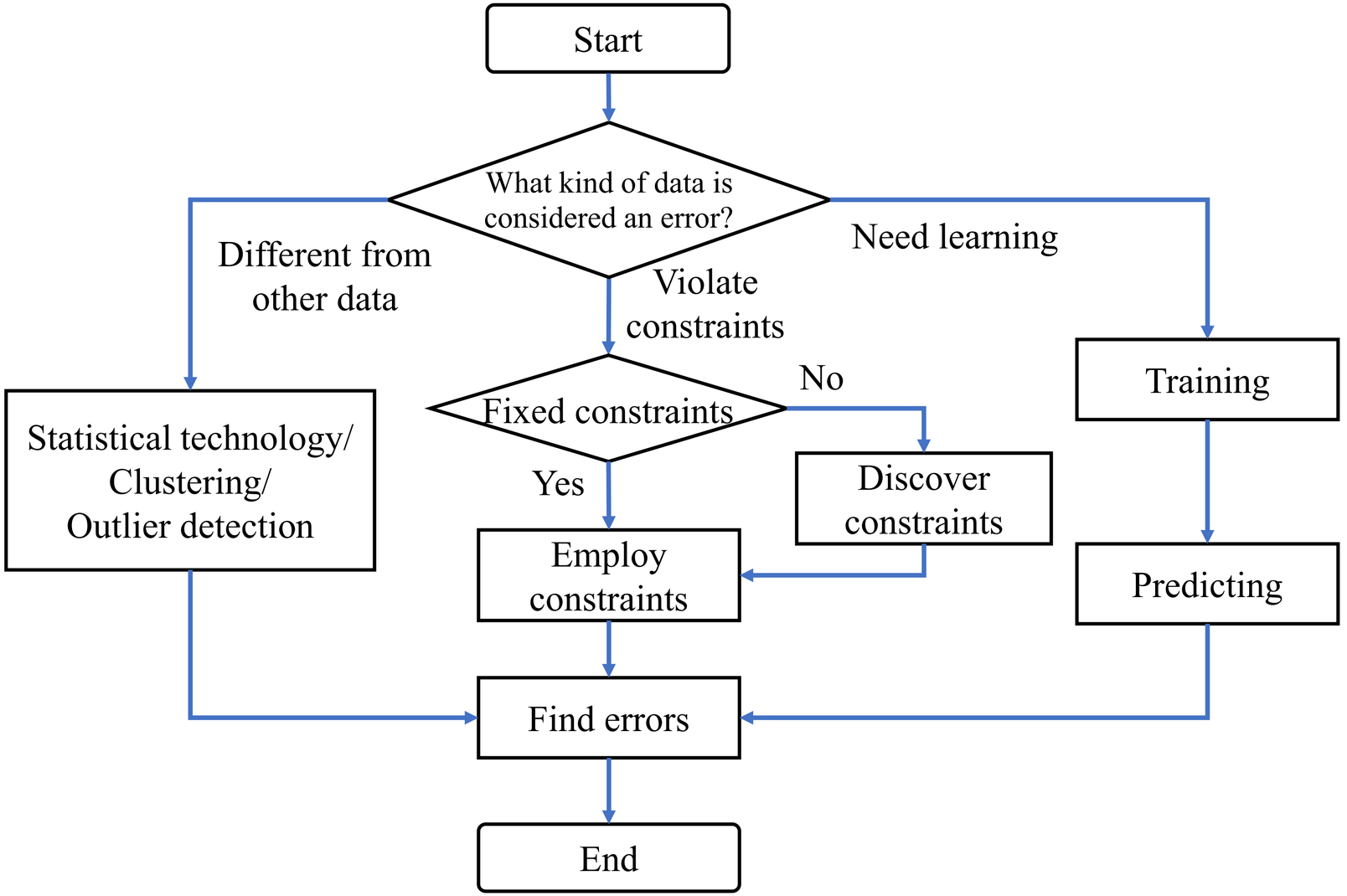
Data Cleaning Overview

	Length (cm)	Size	Material	Price ($)
Pants 1	95	XL	Silk	70
Pants 2	0.9	M	Cotton	30
Pants 3	100	L	?	35



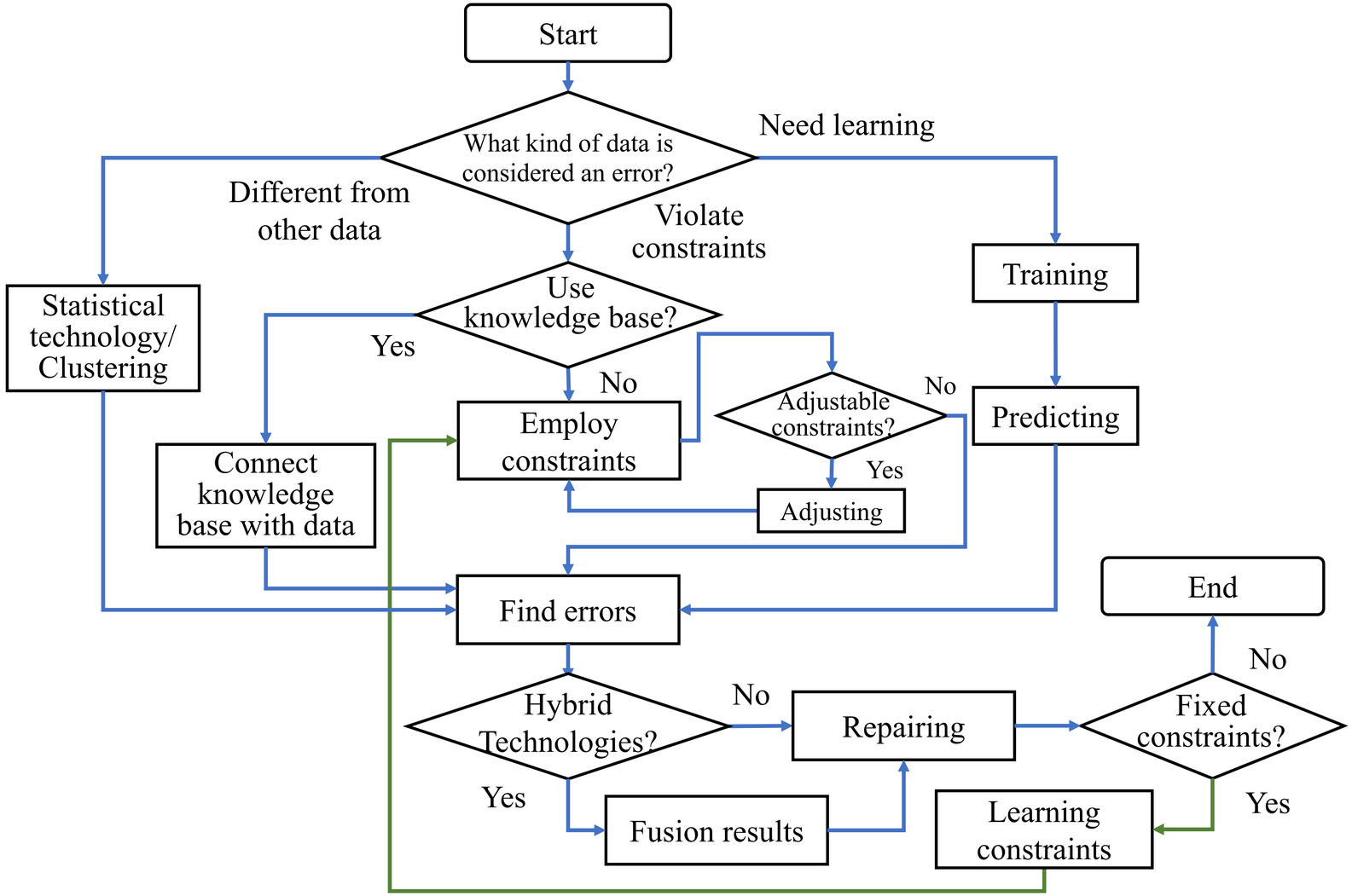
[J. Zhu et al., 2024]

Error Detection



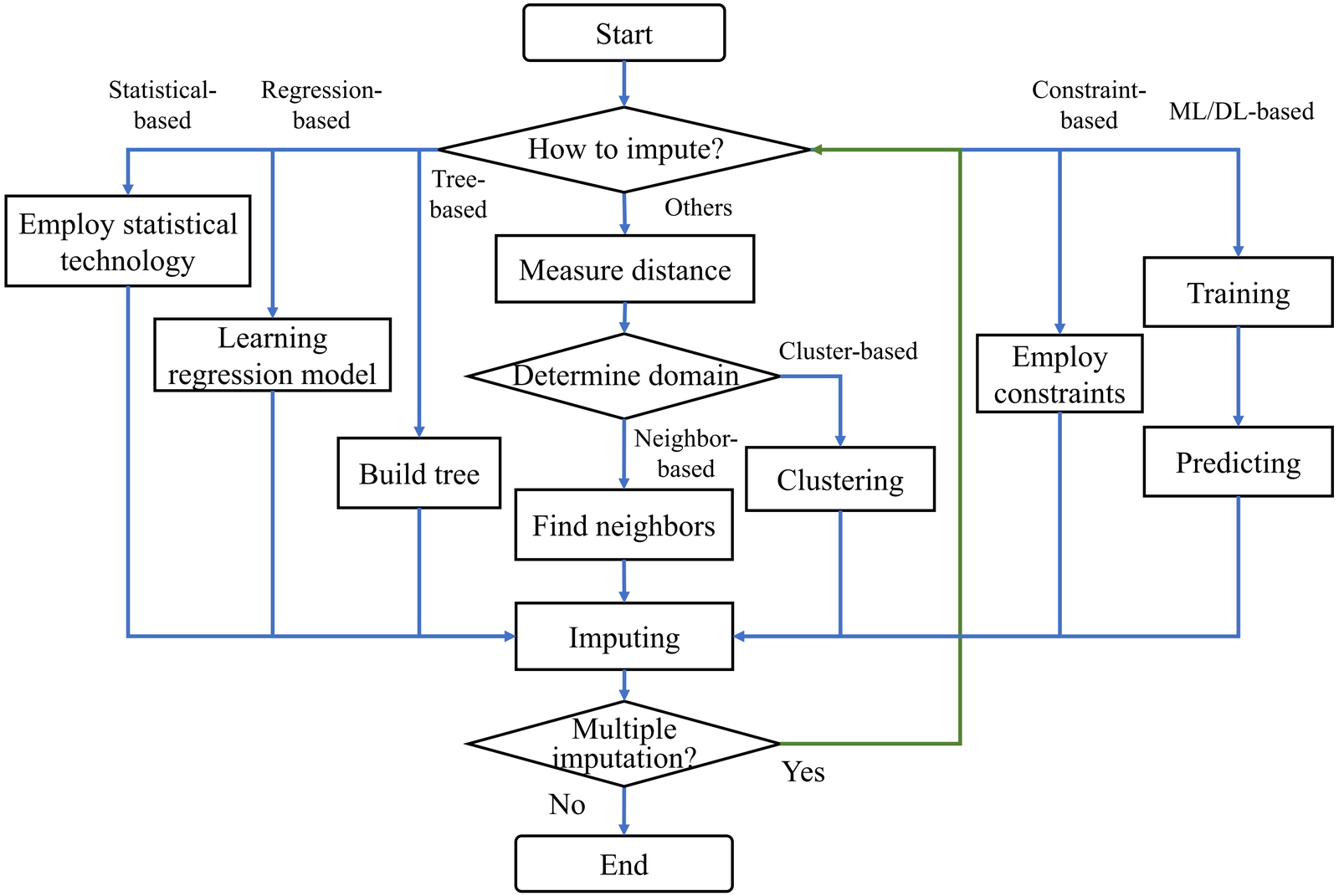
[J. Zhu et al., 2024]

Data Repairing



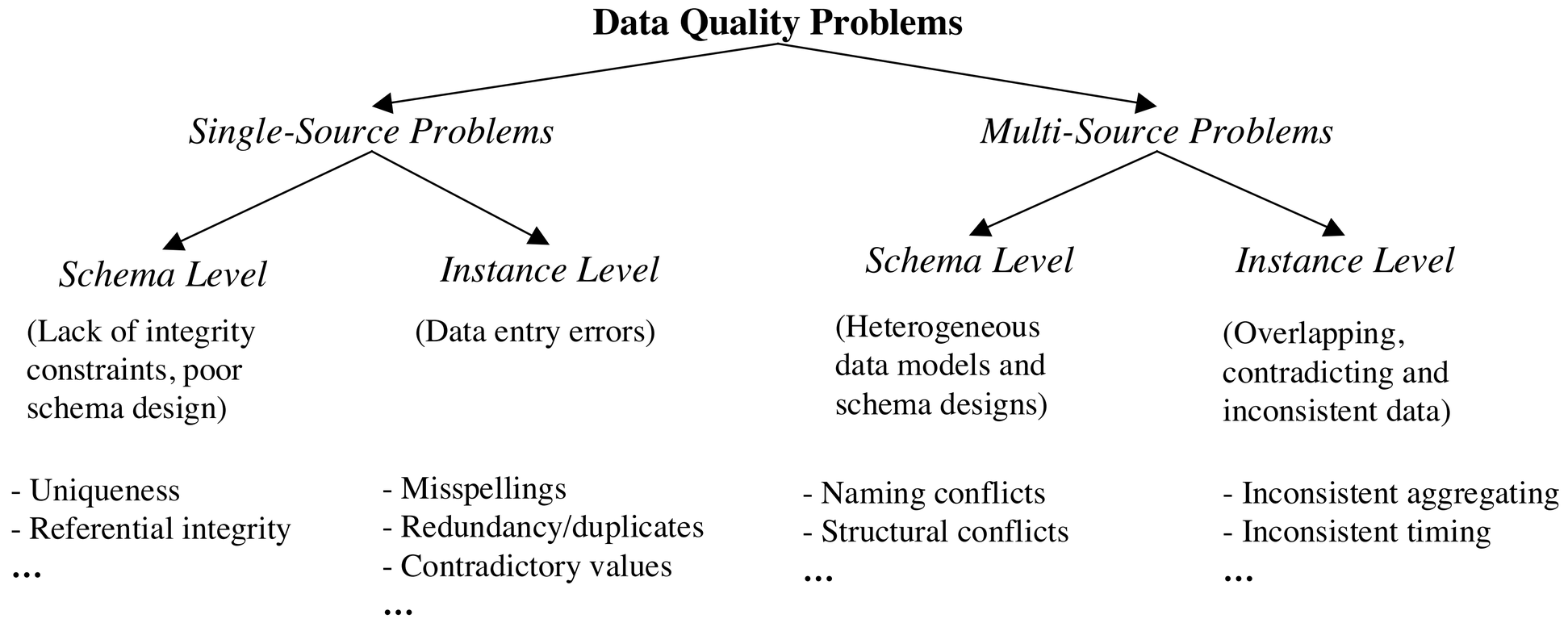
[J. Zhu et al., 2024]

Data Imputation



[J. Zhu et al., 2024]

Classifying Data Quality Problems



[E. Rahm & H. H. Do, 2000]

Single-Source Schema Problems

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Illegal values	bdate=30.13.70	values outside of domain range
Record	Violated attribute dependencies	age=22, bdate=12.02.70	age = (current date – birth date) should hold
Record type	Uniqueness violation	emp_1=(name="John Smith", SSN="123456") emp_2=(name="Peter Miller", SSN="123456")	uniqueness for SSN (social security number) violated
Source	Referential integrity violation	emp=(name="John Smith", deptno=127)	referenced department (127) not defined

[E. Rahm & H. H. Do, 2000]

Single-Source Instance Problems

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city=”Liipzig”	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience=”B”; occupation=”DB Prog.”	
	Embedded values	name=”J. Smith 12.02.70 New York”	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city=”Germany”	
Record	Violated attribute dependencies	city=”Redmond”, zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= “J. Smith”, $name_2$=”Miller P.”	usually in a free-form field
	Duplicated records	emp_1=(name=”John Smith”,...); emp_2=(name=”J. Smith”,...)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name=”John Smith”, bdate=12.02.70); emp_2=(name=”John Smith”, bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name=”John Smith”, deptno=17)	referenced department (17) is defined but wrong

[E. Rahm & H. H. Do, 2000]

Multi-Source Schema & Instance Problems

Customer (source 1)

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Client (source 2)

Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Customers (integrated target with cleaned data)

No	*LName*	*FName*	*Gender*	*Street*	*City*	*State*	*ZIP*	*Phone*	*Fax*	*CID*	*Cno*
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

[E. Rahm & H. H. Do, 2000]

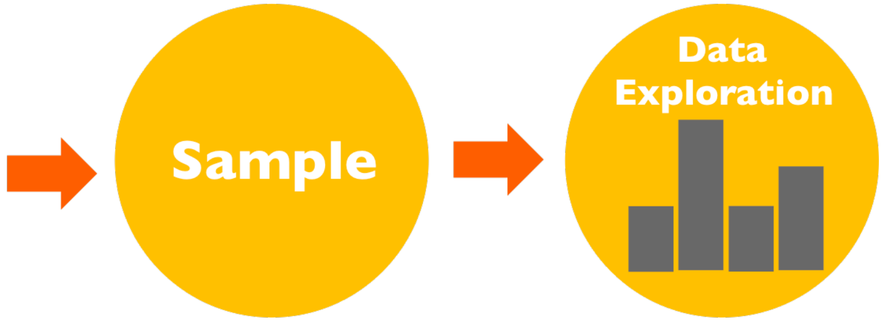
SampleClean (and Variants)

- Dirty Data?
  - Missing Values
  - Duplicate Values
  - Incorrect Values
  - Inconsistent Values
- Estimate query results using a sample of the data
- Two ideas:
  - Direct Estimate
  - Correction

Typical Data Cleaning Steps

Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6

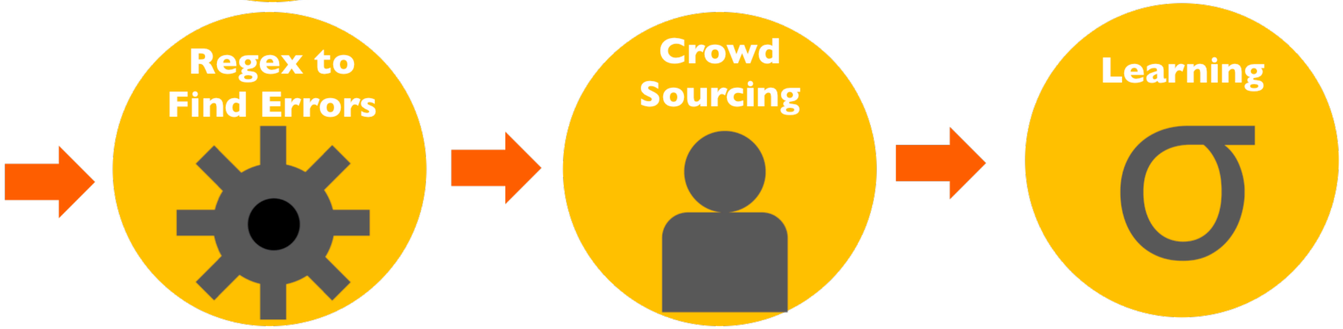


Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6

Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6



[sampleclean.org]

Dirty and Cleaned Data

(a) Dirty Data

id	title	pub_year	citation_count
t_1	CrowdDB	11	18
t_2	TinyDB	2005	1569
t_3	YFilter	Feb, 2002	298
t_4	Aqua		106
t_5	DataSpace	2008	107
t_6	CrowdER	2012	1
t_7	Online Aggr.	1997	687
...	...	...	...
t_{10000}	YFilter - ICDE	2002	298

(b) Cleaned Sample

id	title	pub_year	citation_count	#dup
t_1	CrowdDB	**2011**	**144**	**2**
t_2	TinyDB	2005	1569	1
t_3	YFilter	**2002**	298	**2**
t_4	Aqua	**1999**	106	1
t_5	DataSpace	2008	107	1
t_6	CrowdER	2012	**34**	1
t_7	Online Aggr.	1997	687	**3**

[J. Wang et al., 2014]

Two Sources of Error

- Approximate Query Processing (AQP): Don't process the entire dataset, but use samples to get an approximate result
- Now add dirty data
- Two sources of error:

If R is dirty, then there is a true relation R_{clean}.

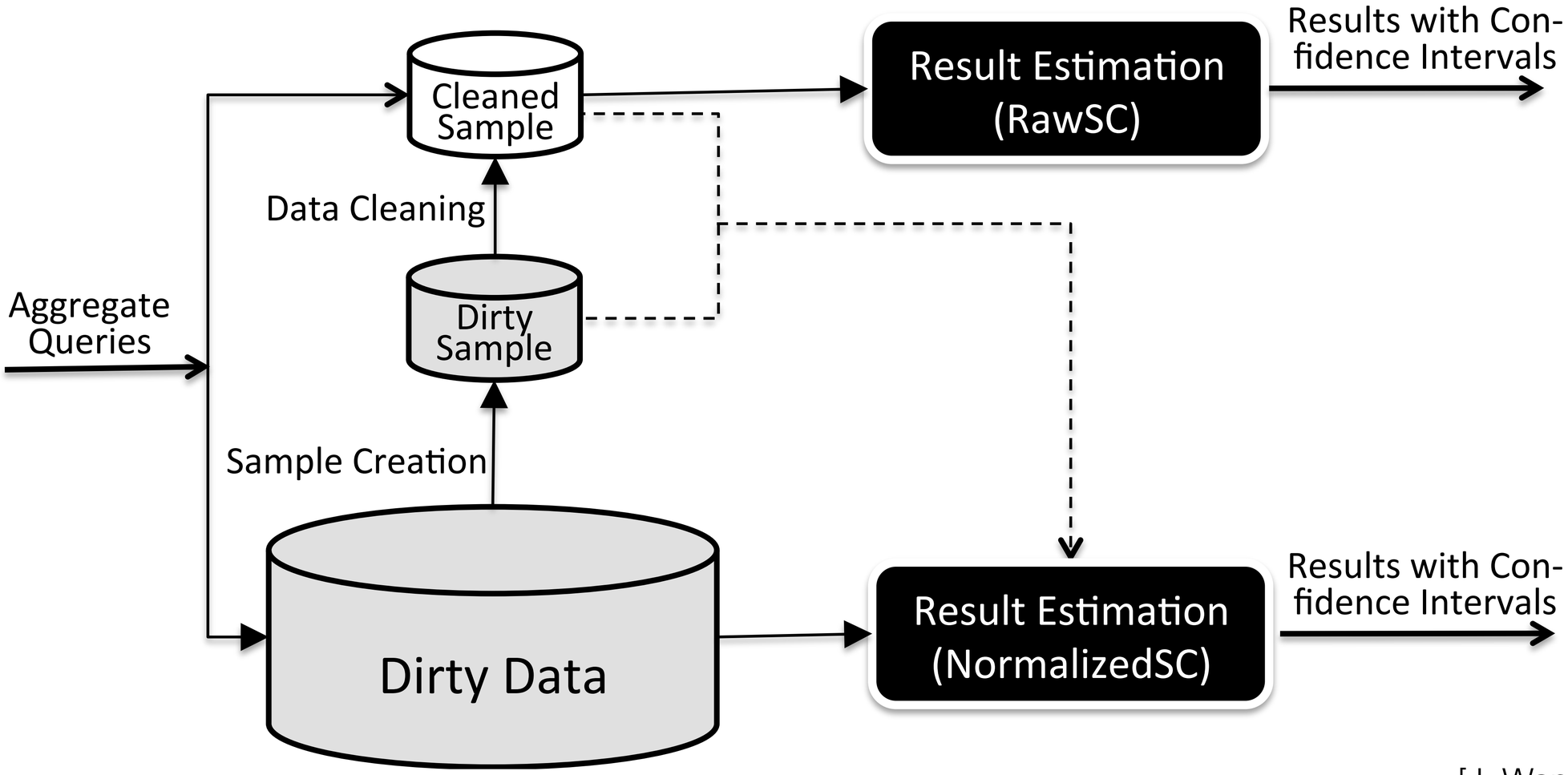
$$Q(R_{clean}) \neq Q(R) \approx est = c \cdot Q(S)$$

The error in est has two components: error due to sampling ϵ_s and error due to the difference with the cleaned relation $\epsilon_c = Q(R_{clean}) - Q(R)$:

$$| Q(R_{clean}) - est | \leq \epsilon_s + \epsilon_c$$

[S. Krishnan et al., 2015]

SampleClean Framework



[J. Wang et al., 2014]

Types of Direct Estimation Errors

- Attribute Errors:
  - value of one attribute is wrong
  - affect a single row
  - does not affect sampling
- Duplication Errors
  - same items appear multiple times
  - those items are over-represented
  - count up duplicates and divide the influence

Direct Estimation with Errors

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi_{\texttt{clean}}(\cdot)$ to each $t_i \in S$ and call the resulting set $\phi_{\texttt{clean}}(S)$
3. Calculate the mean μ_c, and the variance σ_c^2 of $\phi_{\texttt{clean}}(S)$
4. Return $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$

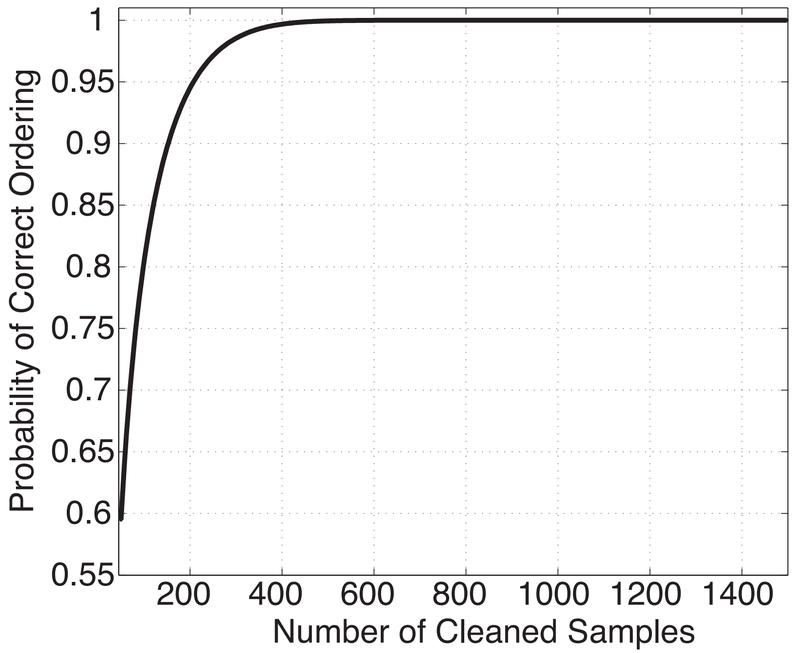
[S. Krishnan et al., 2015]

Correction with Data Errors

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi(\cdot)$ and $\phi_{\texttt{clean}}(\cdot)$ to each $r_i \in S$ and call the set of differences $Q(S)$.
3. Calculate the mean μ_q, and the variance σ_q of $Q(S)$
4. Return $(f(R) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{k}}$

[S. Krishnan et al., 2015]

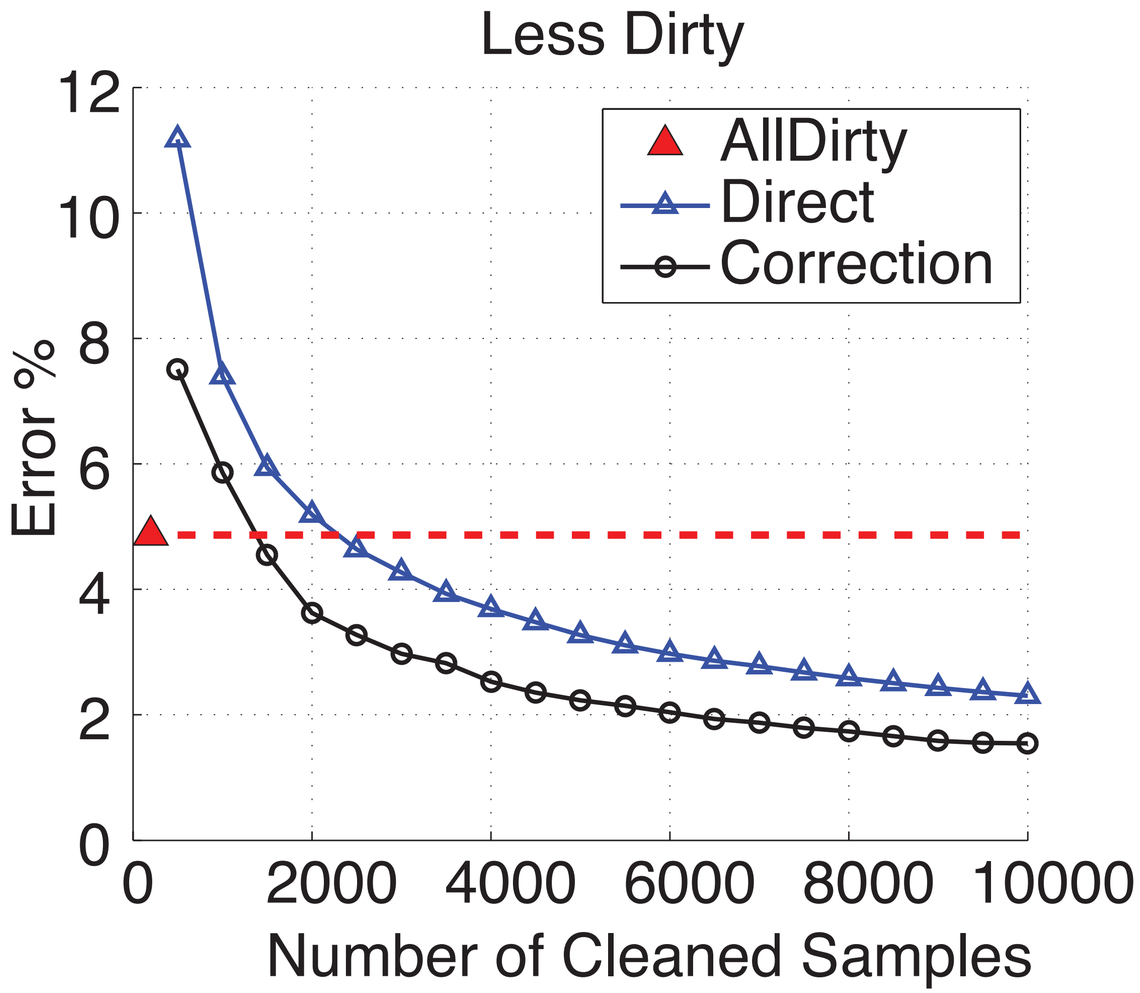
Example

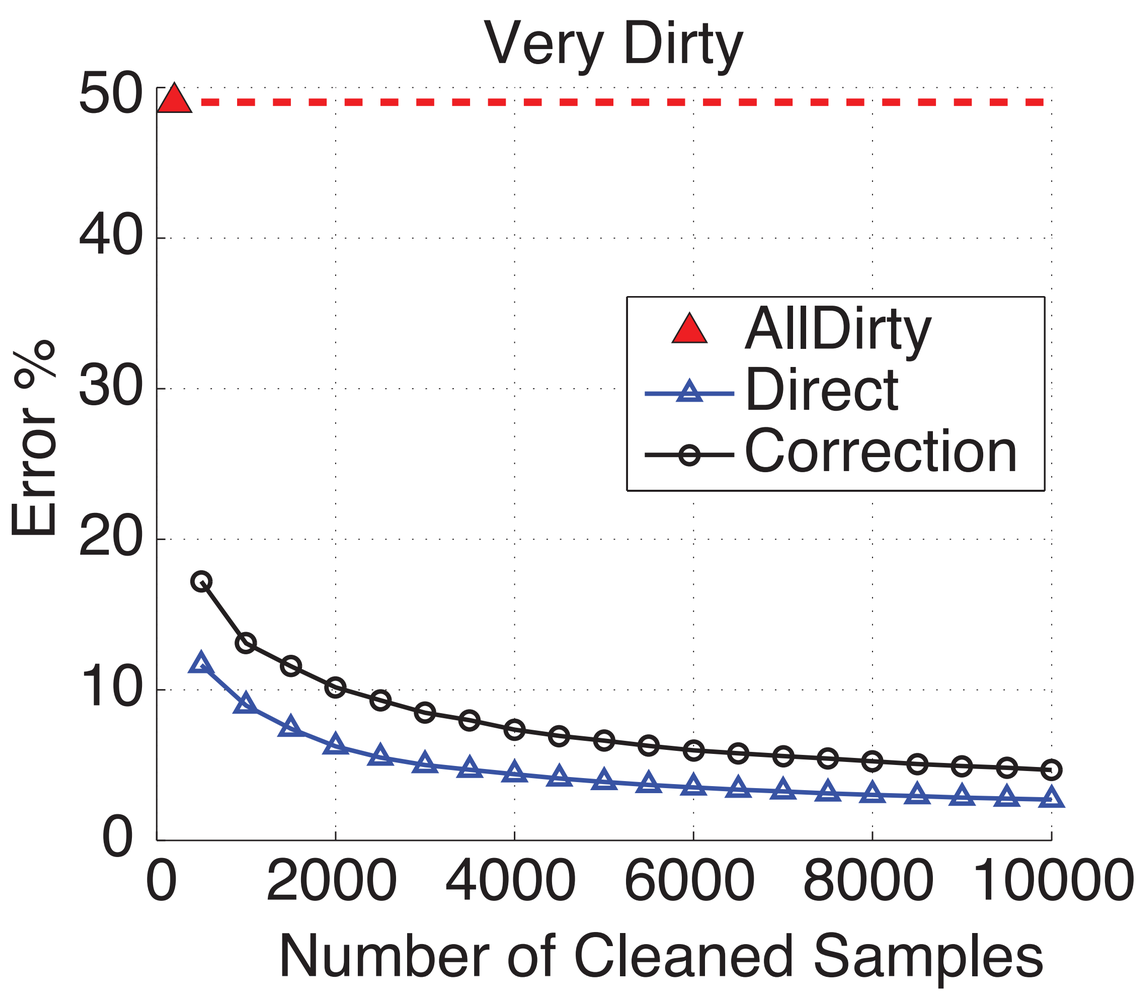


Name	Dirty	Clean	Pred %	Dup
Rakesh Agarwal	353	211	18.13%	1.28
Jeffery Ullman	460	255	05.00%	1.65
Michael Franklin	560	173	65.09%	1.13

[S. Krishnan et al., 2015]

Comparing the Two Approaches





[S. Krishnan et al., 2015]

Notes

- Duplicate Problem
- Focuses on aggregate measures
- How do we actually clean the data?

[S. Krishnan et al., 2016]

ML Data Cleaning Operations

Data cleaning activity	Data error type
Feature cleaning	Incorrect feature values
Label cleaning	Incorrect label values
Entity matching	Duplicate records
Outlier detection	Out-of-distribution records
Imputation	Missing values
Holistic data cleaning	More than one error type at the same time

Data Cleaning for Machine Learning

Problem: Simpson's Paradox

[S. Krishnan et al., 2016]

ActiveClean

- Given dirty data and a mapping from the data to a feature vector and label, we want a **reliable** estimate of the **clean** model
  - reliable = bounded estimate
- Solution: Use stochastic gradient descent (uses sampling!)

[S. Krishnan et al., 2016]

Machine Learning and Data Cleaning

- Data cleaning important for machine learning
  - Filter dirty Data
  - Make learning robust to noise (early stopping?)
- …but machine learning can also help data cleaning
  - No need for rules, just learn
  - Can include lots of features like statistical properties, integrity constraints
  - What about explainability?

HoloClean

- A holistic data cleaning framework that combines qualitative methods with quantitative methods:
  - Qualitative: use integrity constraints or external data sources
  - Quantitative: use statistics of the data
- Driven by probabilistic inference. Users only need to provide a dataset to be cleaned and describe high-level domain specific signals.
- Can scale to large real-world dirty datasets and perform automatic repairs with high accuracy

[T. Rekatsinas et al., 2017]

Example: Input Data

Functional Dependency:
Constraint between two attribute sets
where one set uniquely determines the other set

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(B) Functional Dependencies

c1: DBAName $\rightarrow$ Zip
c2: Zip $\rightarrow$ City, State
c3: City, State, Address $\rightarrow$ Zip

[T. Rekatsinas et al., 2017]

Example: Fixing via Minimality

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(B) Functional Dependencies

c1: DBAName $\rightarrow$ Zip
c2: Zip $\rightarrow$ City, State
c3: City, State, Address $\rightarrow$ Zip

(E) Repair using Minimality w.r.t FDs

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Cicago**	IL	60608

[T. Rekatsinas et al., 2017]

Example: Fixing via External Matches

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(C) Matching Dependencies

$$\text{m1: } \text{Zip} = \text{Ext\_Zip} \rightarrow \text{City} = \text{Ext\_City}$$

$$\text{m2: } \text{Zip} = \text{Ext\_Zip} \rightarrow \text{State} = \text{Ext\_State}$$

$$\text{m3: } \text{City} = \text{Ext\_City} \wedge \text{State} = \text{Ext\_State} \wedge \text{Address} = \text{Ext\_Address} \rightarrow \text{Zip} = \text{Ext\_Zip}$$

(D) External Information (Address listings in Chicago)

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

(F) Repair using Matching Dependencies

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

[T. Rekatsinas et al., 2017]

Example: Fixing via Statistics

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(G) Repair that leverages Quantitative Statistics

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t4	**John Veliotis Sr.**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

[T. Rekatsinas et al., 2017]

HoloClean

Input

Dataset to be cleaned

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t4	Johnnyo's	3465 S Morgan ST	Cicago	IL	60608

Denial Constraints

c1: DBAName → Zip
c2: Zip → City, State
c3: City, State, Address → Zip

External Information

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

Matching Dependencies

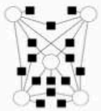
m1: Zip = Ext_Zip → City = Ext_City
m2: Zip = Ext_Zip → State = Ext_State
m3: City = Ext_City ∧ State = Ext_State∧
∧ Address = Ext_Address → Zip = Ext_Zip

The HoloClean Framework

1. Error detection module

2. Automatic compilation to a probabilistic graphical model

3. Repair via statistical learning and infernece



Output

Proposed Cleaned Dataset

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
t4	**John Veliotis Sr.**	3465 S Morgan ST	**Chicago**	IL	60608

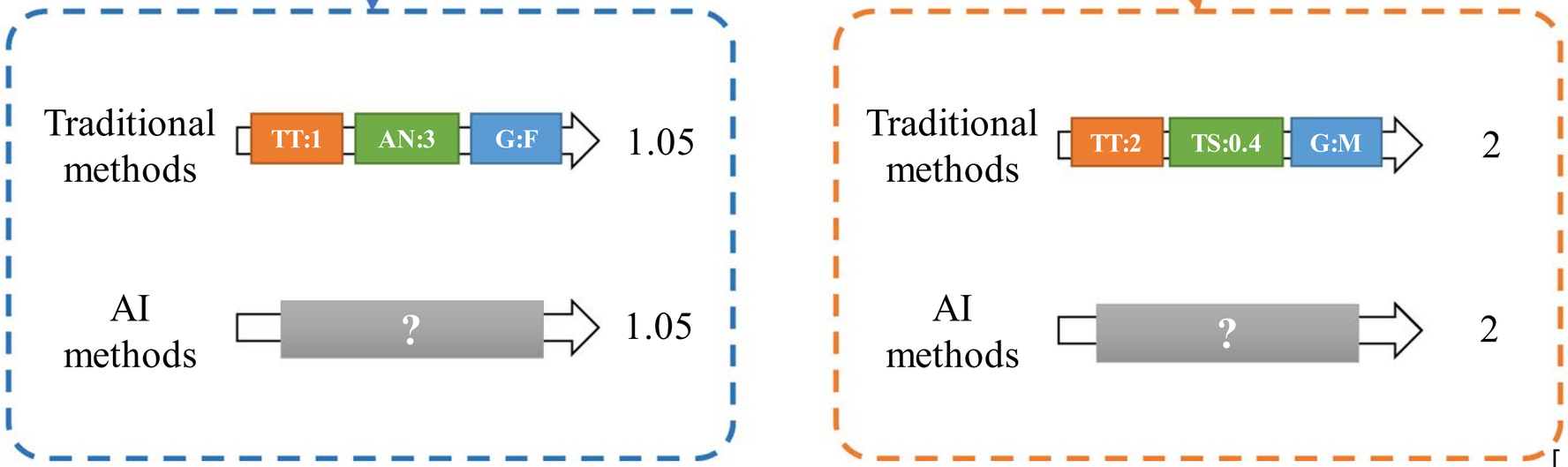
Marginal Distribution of Cell Assignments

Cell	Possible Values	Probability
t2.Zip	60608	0.84
	60609	0.16
t4.City	Chicago	0.95
	Cicago	0.05
t4.DBAName	John Veliotis Sr.	0.99
	Johnnyo's	0.01

[T. Rekatsinas et al., 2017]

Data Cleaning and AI

	Gender (G)	Tumor size (TS)	Axillary lymph nodes (AN)	Age (A)	Type of treatment (TT)	Survival status (SS)
Patient 1	M	1.2	2	58	3	0
Patient 2	F	105	3	45	1	1
Patient 3	F	0.3	1	37	3	1
Patient 4	M	0.4	?	62	2	1



[J. Zhu et al., 2024]

Data Cleaning and AI

- Traditional Methods are often efficient and interpretable
- Deep Learning is expensive and hard to understand but can be more effective
- Machine Learning provides a balance?

		Cost	Generalization	Interpretability	Efficiency	Effectiveness
	Traditional	¥	⚛	💬💬💬	⚡⚡⚡	🌷🌷
AI	ML	¥¥	⚛⚛	💬💬	⚡	🌷🌷
AI	DL	¥¥¥	⚛⚛⚛	💬	⚡	🌷🌷🌷

[J. Zhu et al., 2024]

Anomaly Detection using LLMs

Q: “Data 0 is 48.10. Data 1 is 51.51. Data 2 is 51.35. Data 3 is -57.44. Data 4 is 49.18. Data 5 is -52.49. Data 6 is 54.04. Data 7 is 51.36. Data 8 is 52.74. Data 9 is 47.49. Data 10 is 4.37. Data 11 is 50.28. Data 12 is 55.29. Data 13 is 51.40. Data 14 is -80.43. Data 15 is 46.53.

Abnormal data are different from the majority. Which data are abnormal?”

Llama2: “Data 3, 14, and 15 are abnormal.” (https://www.llama2.ai/)

Llama2-AD: “Data 3, 5, 10 and 14 are abnormal.”

[A. Li et al., 2024]

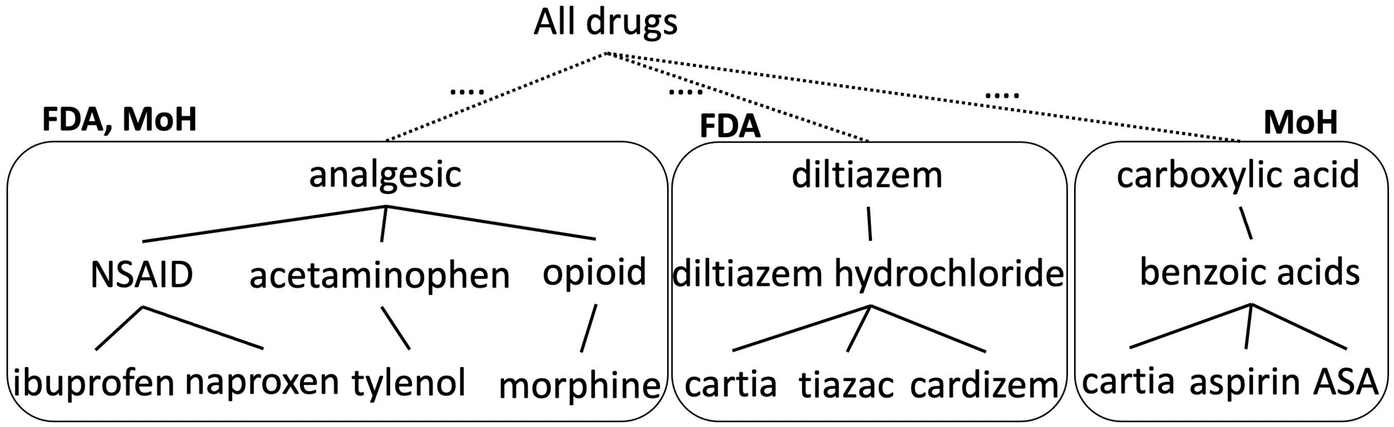
Anomaly Detection using LLMs

	Proposed Methods						Baselines	
	GPT-3.5	GPT-4	Llama2	Llama2-AD	Mistral	Mistral-AD	KNN	ECOD
abalone	78.4±15.2	84.4±7.4	67.2±16.9	49.7±13.3	73.0±15.6	75.1±9.0	88.0±8.8	83.9±10.1
annthyroid	65.1±13.5	82.8±4.5	50.8±1.1	61.5±11.9	64.7±13.0	82.3±9.0	76.5±7.0	81.4±3.3
arrhythmia	73.1±1.6	75.9±3.5	47.2±0.1	58.7±4.9	55.2±2.7	61.0±4.8	69.6±5.2	66.3±6.7
breastw	63.1±34.4	98.7±0.5	50.4±2.4	74.3±2.6	62.7±4.6	93.6±2.0	97.5±1.0	99.0±0.3
cardio	83.3±2.5	87.1±1.4	45.5±3.6	71.7±10.5	68.4±18.5	71.7±1.5	92.5±0.4	95.8±1.2
ecoli	78.7±5.1	73.5±2.4	52.3±9.7	78.9±7.3	79.5±6.2	79.1±4.4	89.3±13.6	79.1±10.1
forest cover	82.5±11.2	85.9±8.1	53.9±5.5	58.1±25.2	68.7±24.3	52.4±18.8	48.5±18.9	83.3±3.4
glass	69.5±11.4	64.2±14.1	45.4±7.7	56.3±4.7	59.3±8.3	65.9±3.9	86.7±3.0	68.6±8.9
ionosphere	83.5±2.5	88.8±2.0	50.7±1.4	59.9±9.4	64.1±2.3	69.4±8.1	94.7±2.5	85.8±1.8
kdd	66.1±28.8	87.4±1.6	52.4±3.4	58.0±3.1	65.3±1.7	60.1±5.6	59.8±4.8	88.3±1.7
kddrev	58.5±16.8	72.8±5.1	53.3±4.7	60.7±12.6	56.8±9.0	50.2±14.1	45.0±1.2	74.4±5.3
				...				
vertebral	57.9±3.0	51.6±11.4	48.8±3.9	48.2±4.5	54.1±5.7	45.3±2.6	34.6±2.3	44.0±3.3
vowels	40.9±8.1	65.9±3.1	51.9±6.3	51.4±19.1	47.3±3.5	52.5±5.2	96.1±3.9	62.6±14.3
wbc	79.2±5.6	93.4±2.2	48.2±4.8	61.3±5.4	68.5±7.3	88.6±8.2	93.7±2.0	91.9±2.3
wine	47.6±11.5	51.3±10.2	50.6±9.3	51.2±3.9	55.5±8.4	59.7±12.0	30.0±0.0	64.9±0.0
average	68.3±1.2	74.1±2.2	51.1±2.5	60.0±3.2	62.4±2.7	69.1±1.0	70.7±0.9	75.5±1.0

[A. Li et al., 2024]

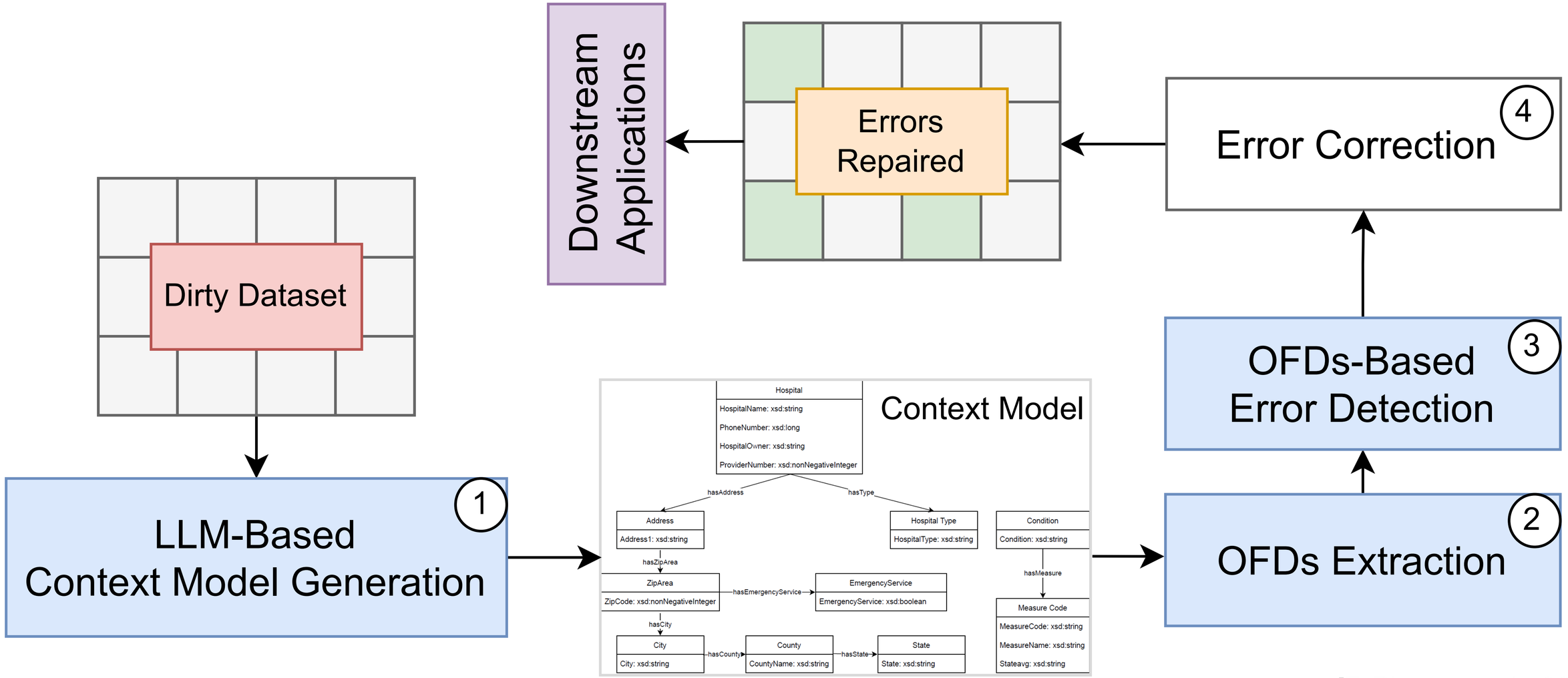
Error Detection using LLMs (LLMClean)

- Ontology: set of concepts and categories in a subject area or domain that shows their properties and the relations between them
- Use Ontological Functional Dependencies (OFDs)
  - Functional dependency from ontology (synonyms, inheritance)
  - Capture semantic relationships between attributes
  - Reduce the incidence of false positives
- Use an automated context model to generate OFDs
- Can run data repair after



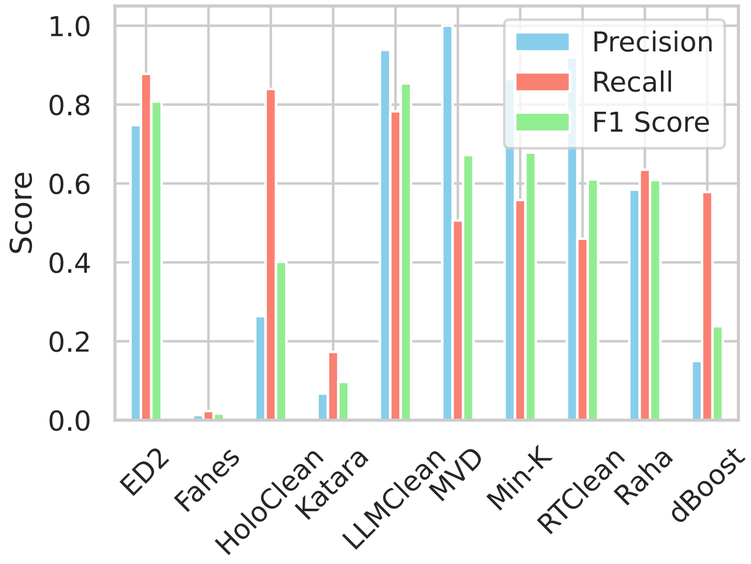
[Z. Zheng et al., 2015]

Error Detection using LLMs (LLMClean)

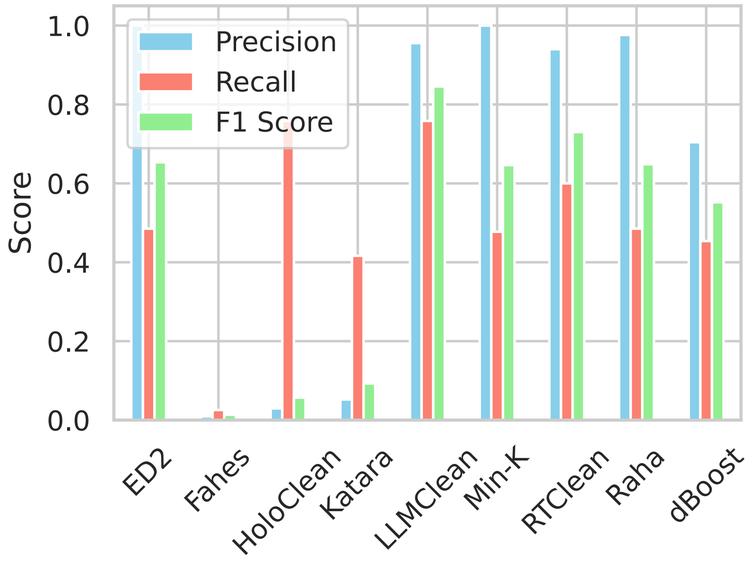


[F. Biester et al., 2024]

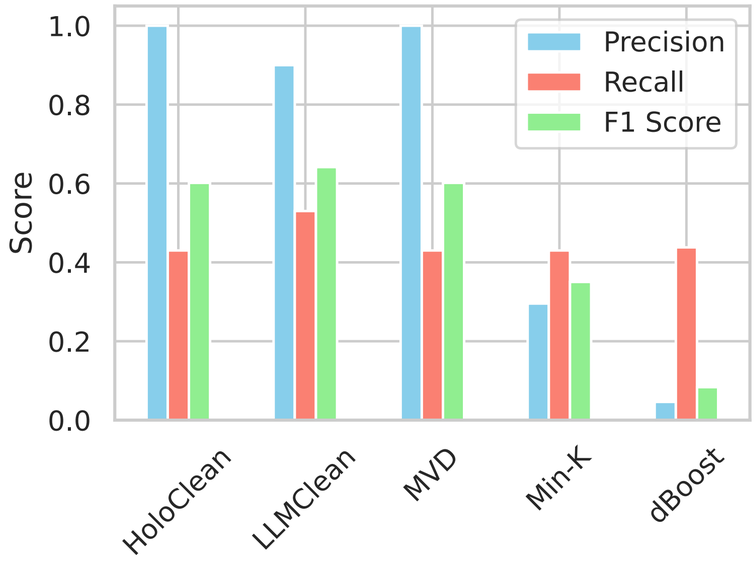
LLMClean Error Detection Evaluation



(a) IoT dataset



(b) Hospital dataset



(c) Context dataset

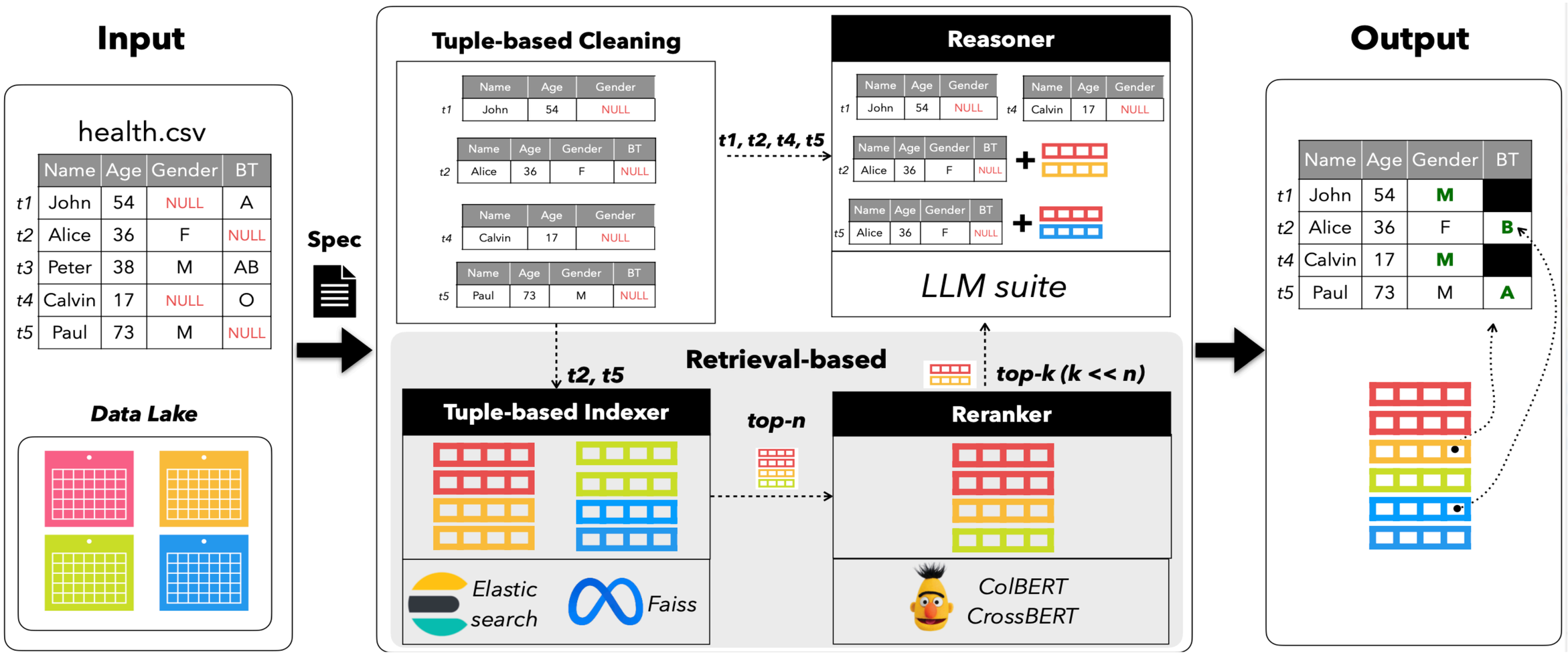
[F. Biester et al., 2024]

Data Repair using LLMs (RetClean)

- Non-retrieval based: Send tuple to LLMs and identify tuple(s) and column(s) to be fixed
- Retrieval-based:
  - Indexer: Get top-k relevant tuples from a database/data lake
  - Reranker: Rank relevance using ColBERT/CrossBERT
  - Reasoner: Determine, using LLM, which tuple and value to use for fix
  - Reasoner keeps track of lineage

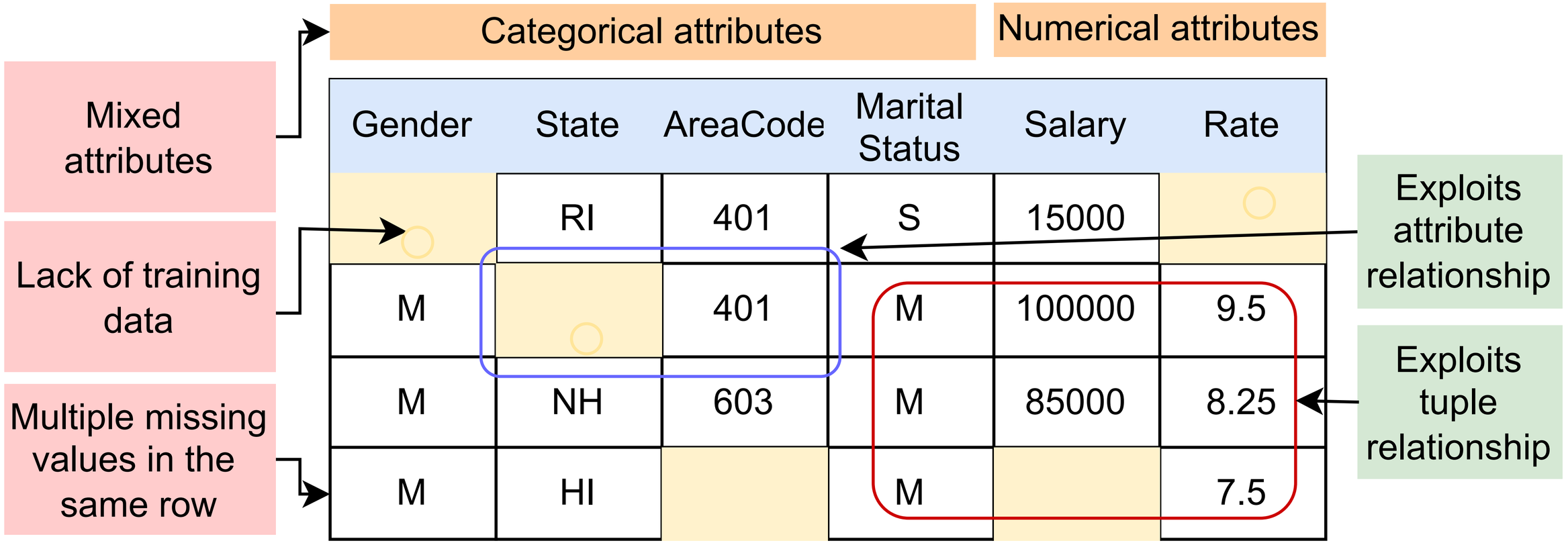
[Naeem et al., 2024]

Data Repair using LLMs (RetClean)



[Naeem et al., 2024]

Data Imputation Challenges

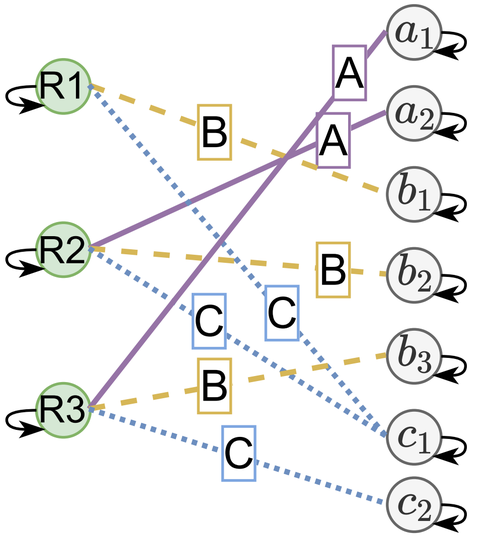


[R. Cappuzzo et al., 2024]

Data Imputation via Graph Learning

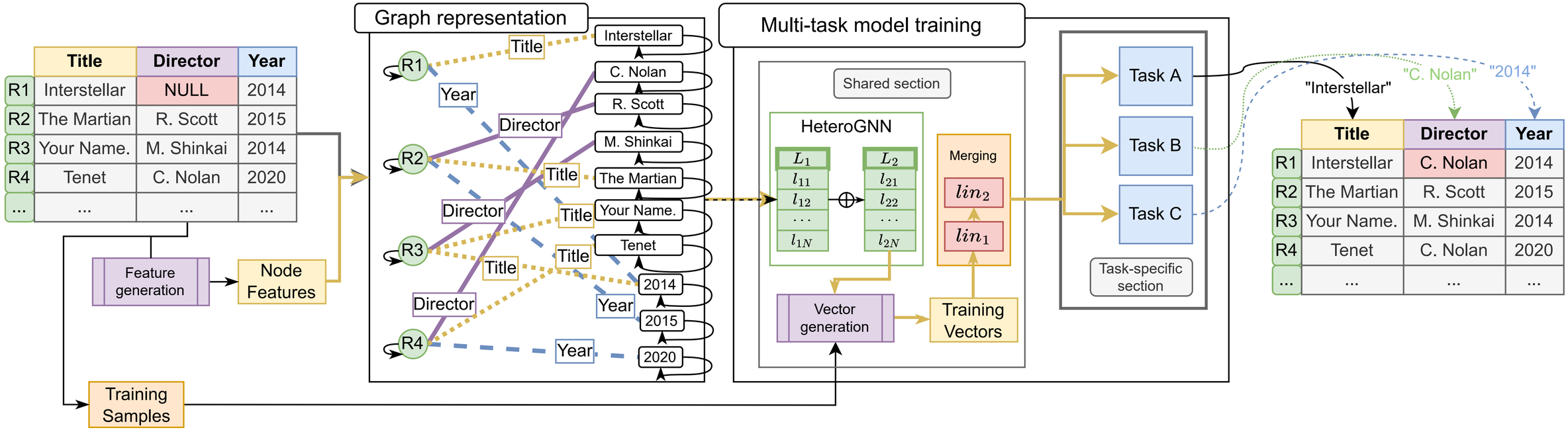
- Represent relational data as a heterogeneous graph
  - relationships between tuples, attributes, and values
- Use graph representation learning
- Self-supervised multi-task learning
- Does not require explicit training data, uses given data with missing values

	Title (A)	Country (B)	Year (C)
R1	NULL	$France$ (b_1)	2014 (c_1)
R2	$Your\ Name$ (a_1)	$Japan$ (b_2)	2014 (c_1)
R3	$Tenet$ (a_2)	USA (b_3)	2020 (c_2)



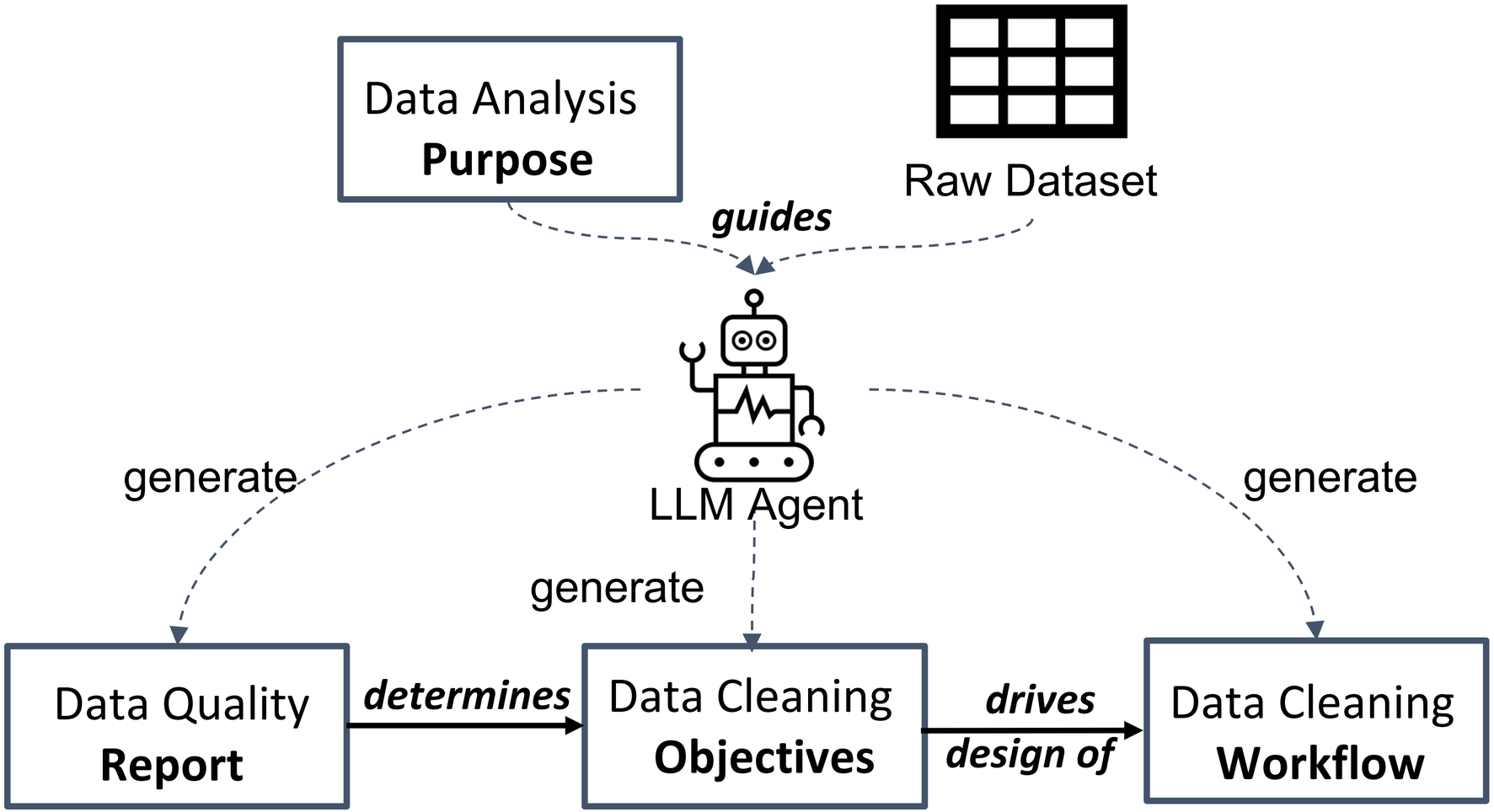
[R. Cappuzzo et al., 2024]

GRIMP Overview



[R. Cappuzzo et al., 2024]

Data Cleaning Workflows using LLMs



[L. Li et al., 2024]