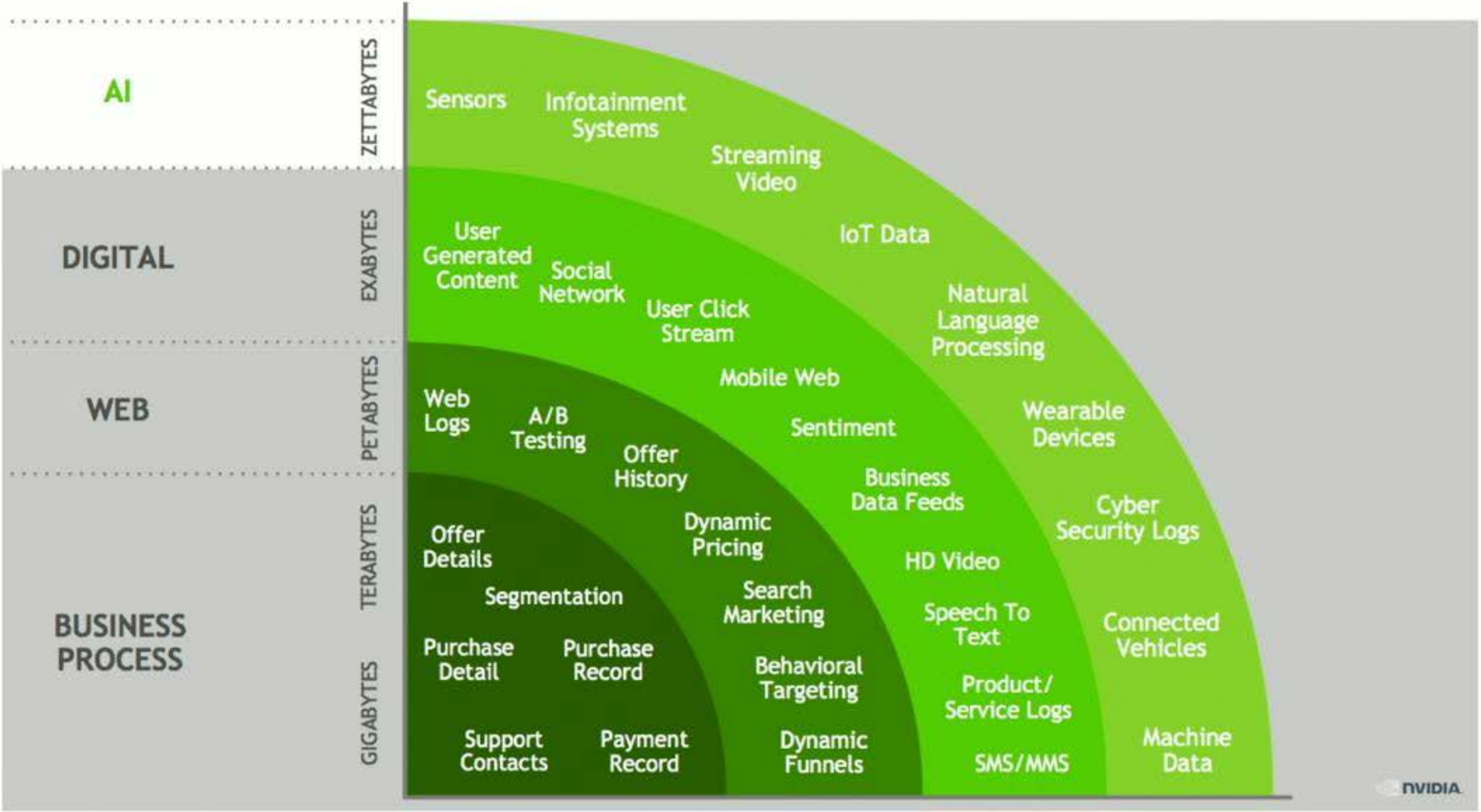
A Machine Learning Perspective on Managing Noisy Data

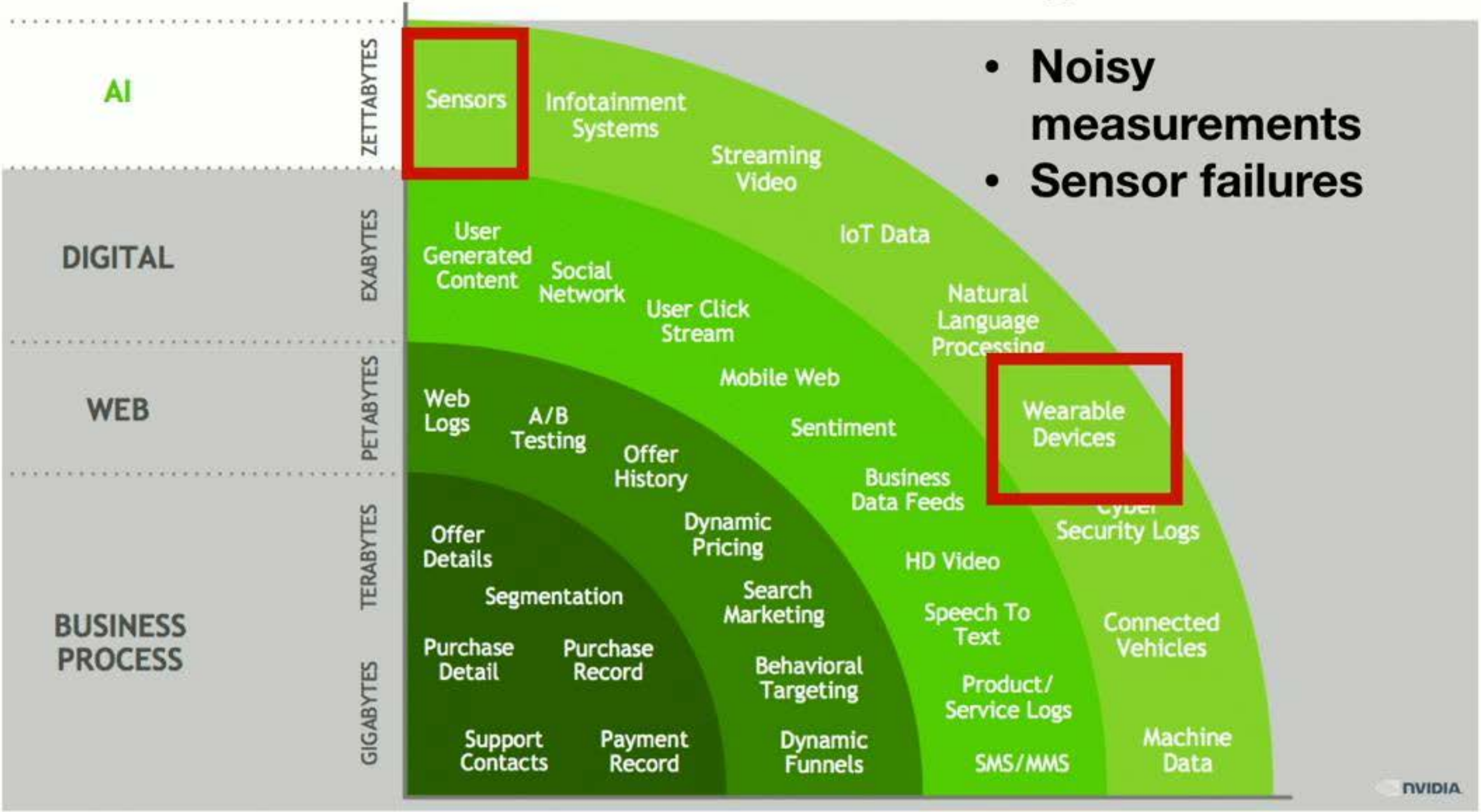
Theodoros Rekatsinas | UW-Madison
@thodrek

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Data-hungry applications are taking over



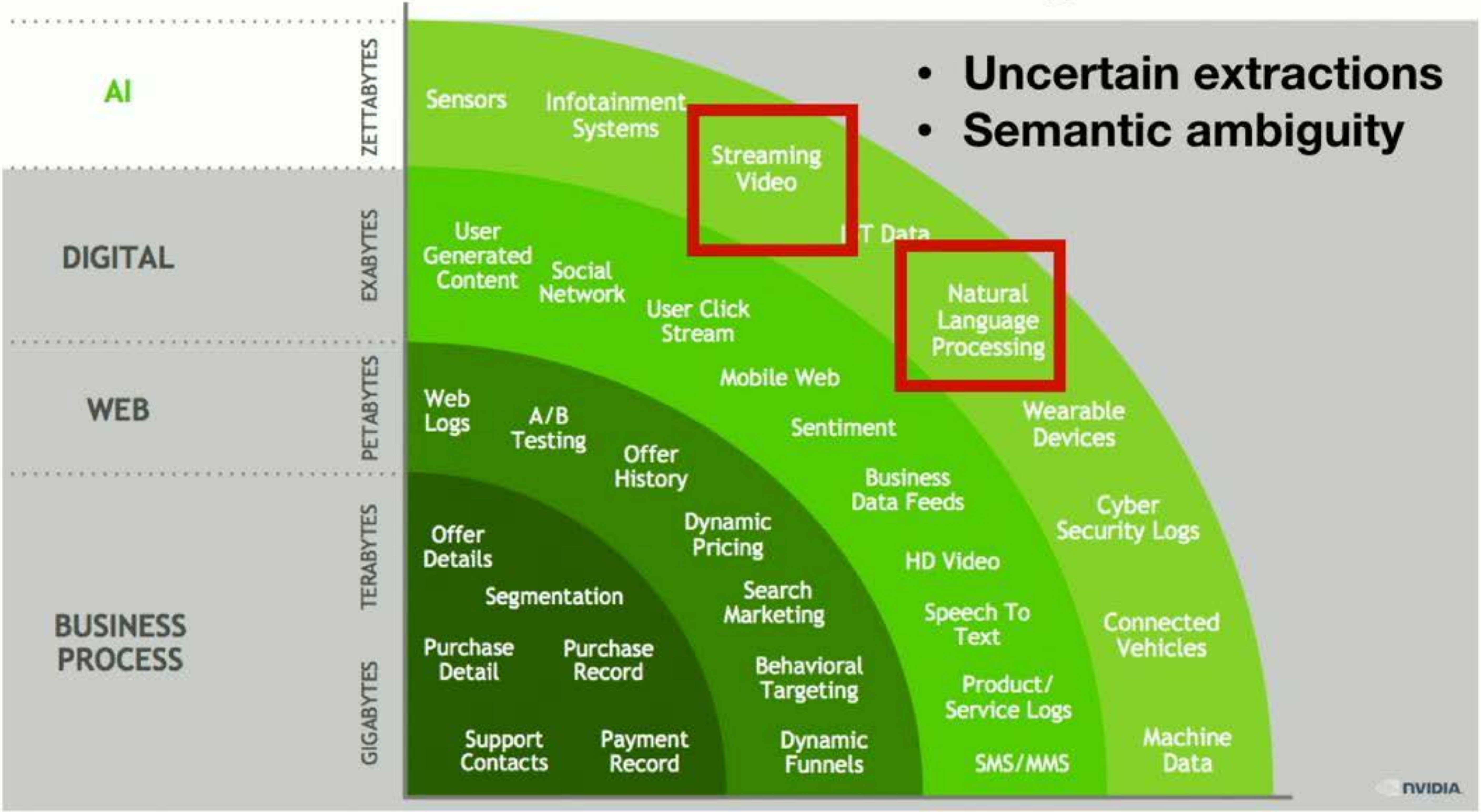
Data errors are everywhere



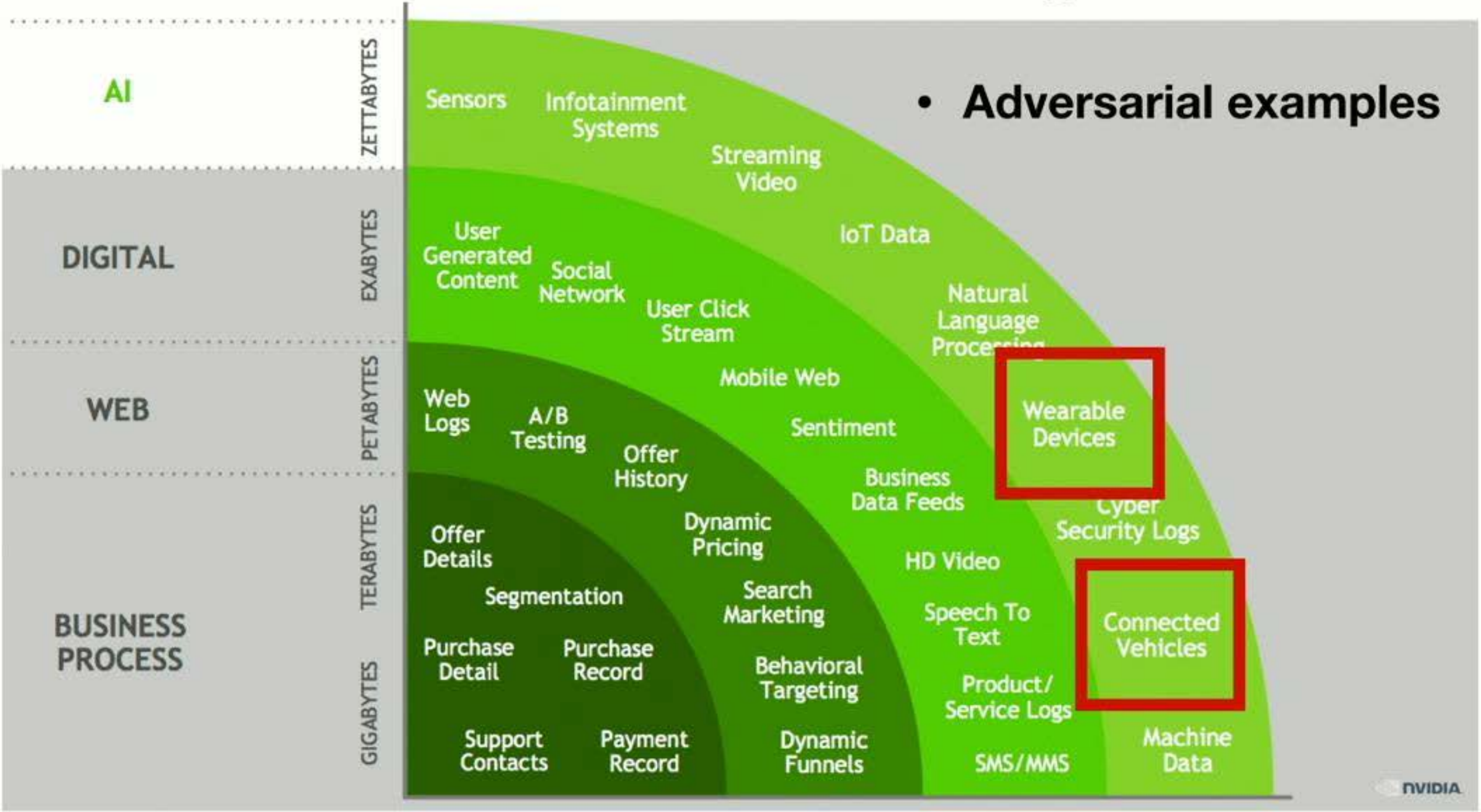
- **Noisy measurements**
- **Sensor failures**

Data errors are everywhere

- **Uncertain extractions**
- **Semantic ambiguity**

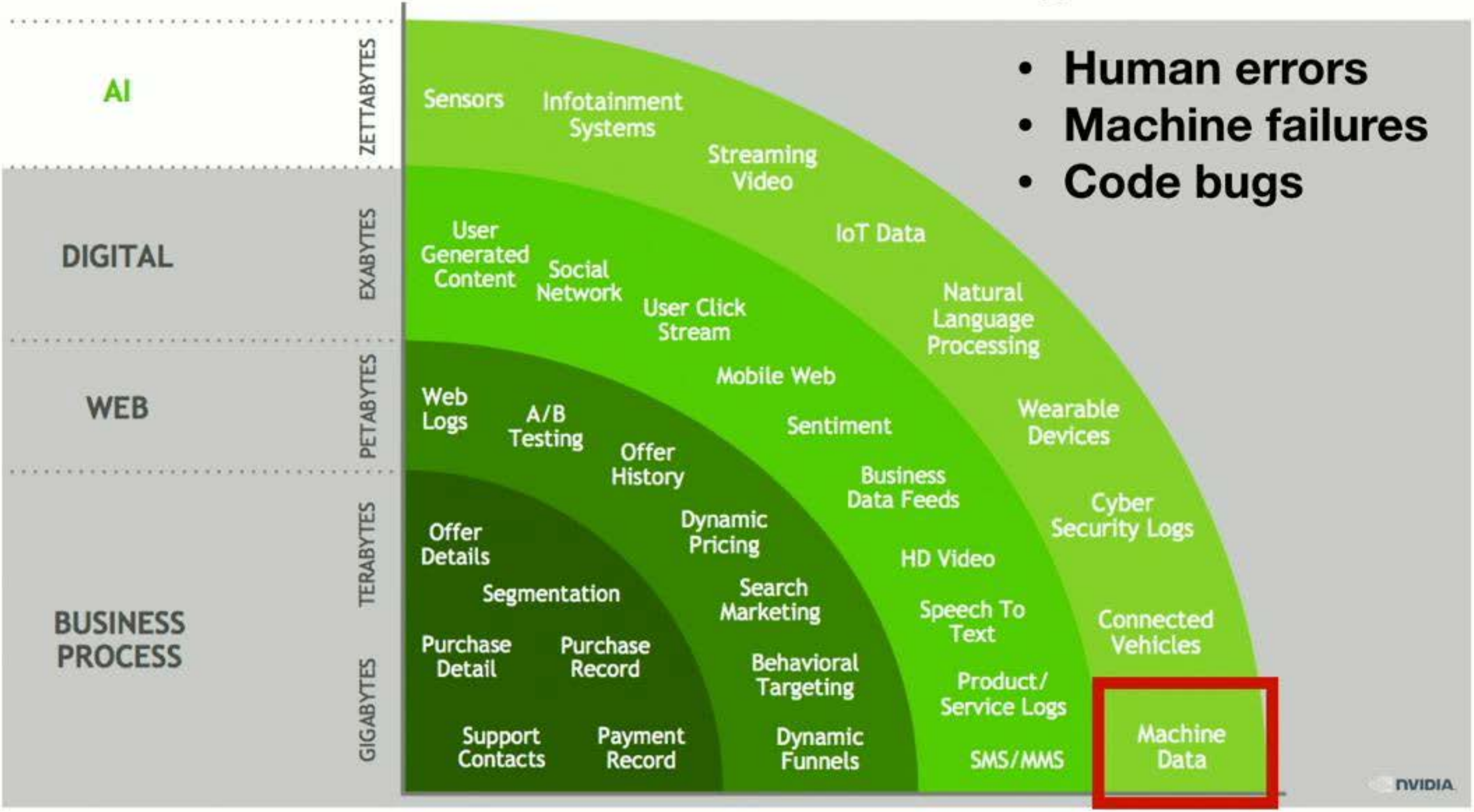


Data errors are everywhere



- **Adversarial examples**

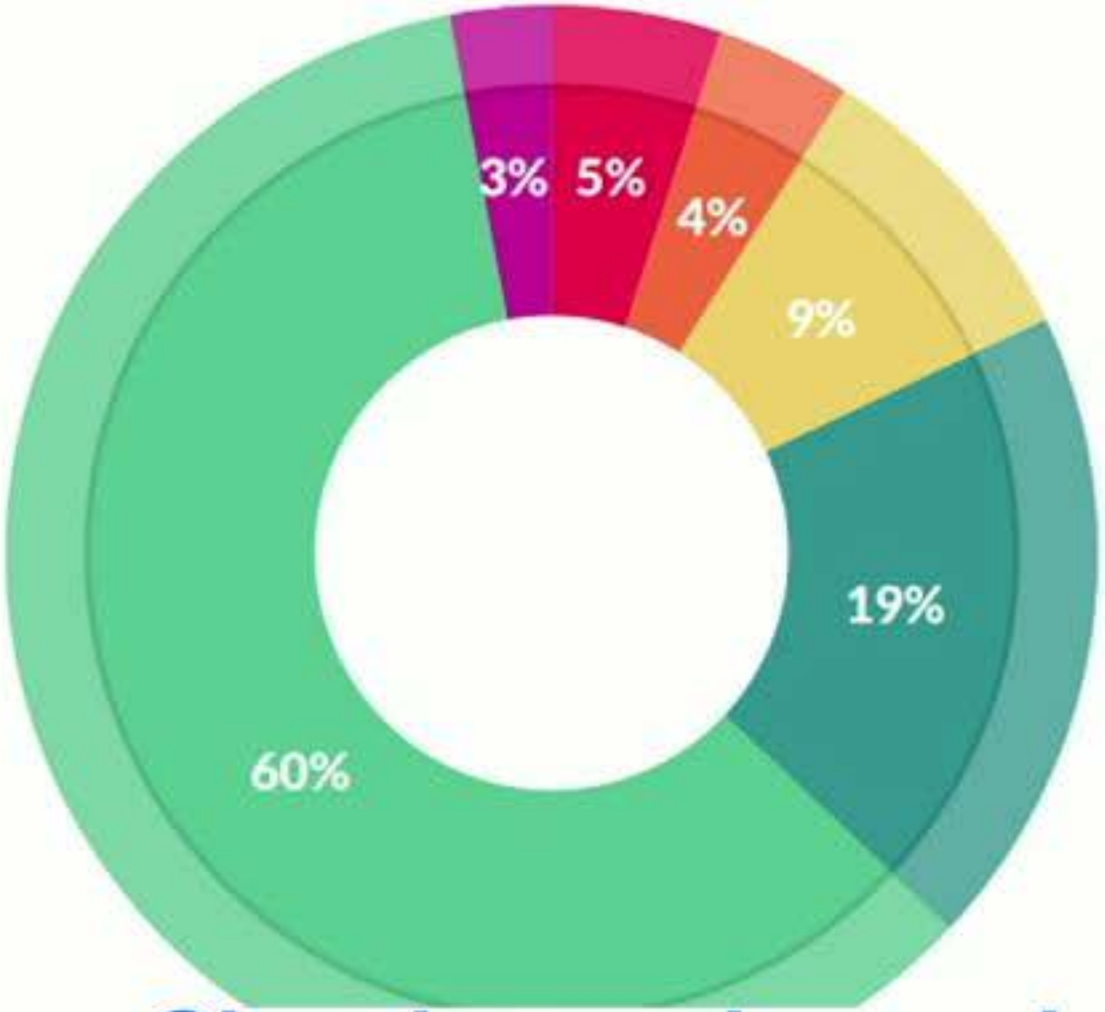
Data errors are everywhere



- Human errors
- Machine failures
- Code bugs

The Achilles' Heel of Modern Analytics

is low quality, erroneous data



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

Cleaning and organizing the data comprises 60% of the time spent on an analytics or AI project.

The Achilles' Heel of Modern Analytics

is low quality, erroneous data

Many modern data management systems are being developed to address aspects of this issue:

Stanford's Snorkel: A System for Fast Training Data Creation

Google's TFX: TensorFlow Data Validation

Amazon's SageMaker

Amazon's Deequ: Data Quality Validation for ML Pipelines

HoloClean: Weakly-supervised data cleaning

Question:

What is an appropriate (formal) framework for managing noisy data?

Things to consider:

Simplicity and generality

Talk outline

- Managing Noisy Data (Background)
- The Probabilistic Unclean Databases (PUDs) Framework
- From Theory to Systems

Managing Noisy Data

A simple example of noisy data

	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

c1: DBAName $\rightarrow$ Zip

c2: Zip $\rightarrow$ City, State

c3: City, State, Address $\rightarrow$ Zip

Does not obey data distribution

Conflict

A simple example of noisy data

	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

c1: DBAName $\rightarrow$ Zip

c2: Zip $\rightarrow$ City, State

c3: City, State, Address $\rightarrow$ Zip

Does not obey data distribution

Conflict

Computational problems: ***Detect*** errors, ***repair*** errors, compute ***"consistent" query answers***.

The case for **inconsistent** data

	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

c1: $\text{DBAName} \rightarrow \text{Zip}$

c2: $\text{Zip} \rightarrow \text{City, State}$

c3: $\text{City, State, Address} \rightarrow \text{Zip}$

An example unclean database *J*

- Errors correspond to tuples/cells that introduce inconsistencies (violations of integrity constraints).
- Inconsistencies are typical in data integration, extract-load-transform workloads, etc.
- **Data repairs:** A theoretical framework for coping with inconsistent databases [Arenas et al. 1999]

Minimal data repairs

Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)

Σ a set of integrity constraints and I an inconsistent database.
A database J is a *repair* of I w.r.t. Σ if

- J is a consistent database (i.e., $J \models \Sigma$);
- J differs from I in a minimal way.

Fact

Several different types of repairs have been considered:

- Set-based repairs (subset, superset, $\oplus$-repairs).
- Cardinality-based repairs
- Attribute-based repairs
- Preferred repairs

Slide by Phokion Kolaitis
[SAT 2016]

Minimal data repairs

Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)

Σ a set of integrity constraints and I an inconsistent database.
A database J is a *repair* of I w.r.t. Σ if

- J is a consistent database (i.e., $J \models \Sigma$);
- J differs from I in a minimal way.

Fact

Several different types of repairs have been considered:

- Set-based repairs (subset, superset, $\oplus$-repairs).
- Cardinality-based repairs
- Attribute-based repairs
- Preferred repairs

Plethora of fundamental results on tractability of repair-checking and consistent query answering.

Slide by Phokion Kolaitis
[SAT 2016]

Minimal data repairs

Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)

Σ a set of integrity constraints and I an inconsistent database.
A database J is a *repair* of I w.r.t. Σ if

- J is a consistent database (i.e., $J \models \Sigma$);
- J differs from I in a minimal way.

Fact

Several different types of repairs have been considered:

- Set-based repairs (subset, superset, $\oplus$-repairs).
- Cardinality-based repairs
- Attribute-based repairs
- Preferred repairs

Slide by Phokion Kolaitis
[SAT 2016]

Plethora of fundamental results on tractability of repair-checking and consistent query answering.

Limited adoption in practice.

Minimal data repairs

	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

$\text{c1: DBAName} \rightarrow \text{Zip}$

$\text{c2: Zip} \rightarrow \text{City, State}$

$\text{c3: City, State, Address} \rightarrow \text{Zip}$

Minimal data repairs

	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

An example repaired database *I*

$c1\colon\ \text{DBAName} \rightarrow \text{Zip}$

$c2\colon\ \text{Zip} \rightarrow \text{City, State}$

$c3\colon\ \text{City, State, Address} \rightarrow \text{Zip}$

Minimal subset repair:
We remove t1

Minimal data repairs

	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

c1: $\text{DBAName} \rightarrow \text{Zip}$

c2: $\text{Zip} \rightarrow \text{City, State}$

c3: $\text{City, State, Address} \rightarrow \text{Zip}$

Minimal subset repair:
We remove t1

Errors remain:

(1) Cicago should clearly be Chicago

(2) Non-obvious errors: 60609 is the wrong Zip

Minimal data repairs

	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

$\mathrm{c1:\ DBAName} \rightarrow \mathrm{Zip}$

$\mathrm{c2:\ Zip} \rightarrow \mathrm{City,\ State}$

$\mathrm{c3:\ City,\ State,\ Address} \rightarrow \mathrm{Zip}$

Errors remain:

(1) Cicago should clearly be Chicago

(2) Non-obvious errors: 60609 is the wrong Zip

Minimal subset repair:
We remove t1

Several variations of minimal repairs. E.g., update the minimum number of cells.

Minimal data repairs

	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

c1: DBAName $\rightarrow$ Zip

c2: Zip $\rightarrow$ City, State

c3: City, State, Address $\rightarrow$ Zip

Minimal subset repair:
We remove t1

Several variations of minimal repairs. E.g., update the minimum number of cells.

Errors remain:

(1) Cicago should clearly be Chicago

(2) Non-obvious errors: 60609 is the wrong Zip

Minimality can be used as an operational principle to prioritize repairs but these repairs are not necessarily correct with respect to the ground truth.

The case for most probable data [Gribkoff et al., 14]

	DBAName	AKAName	Address	City	State	Zip	p
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***	0.9
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608	0.8

c1: DBAName $\rightarrow$ Zip

c2: Zip $\rightarrow$ City, State

c3: City, State, Address $\rightarrow$ Zip

Most probable world, conditioned on integrity constraint satisfaction

The case for **most probable** data [Gribkoff et al., 14]

	DBAName	AKAName	Address	City	State	Zip	*p*	*Factor (f)*
~~t1~~	~~John Veliotis Sr.~~	~~Johnnyo's~~	~~3465 S Morgan ST~~	~~***Chicago***~~	~~IL~~	~~***60608***~~	0.9	1 - 0.9
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4	0.4
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4	0.4
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608	0.8	0.8

c1: $\text{DBAName} \rightarrow \text{Zip}$

c2: $\text{Zip} \rightarrow \text{City, State}$

c3: $\text{City, State, Address} \rightarrow \text{Zip}$

Optimization Objective

$$\max_{I} \left(\prod_{t \in I} p(t) \prod_{t \notin I} (1 - p(t)) \right)$$

The case for **most probable** data [Gribkoff et al., 14]

	DBAName	AKAName	Address	City	State	Zip	*p*	*Factor (f)*
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***	0.9	0.9
~~t2~~	~~John Veliotis Sr.~~	~~Johnnyo's~~	~~3465 S Morgan ST~~	~~Chicago~~	~~IL~~	~~***60609***~~	0.4	1 - 0.4
~~t3~~	~~John Veliotis Sr.~~	~~Johnnyo's~~	~~3465 S Morgan ST~~	~~Chicago~~	~~IL~~	~~***60609***~~	0.4	1 - 0.4
~~t4~~	~~***Johnnyo's***~~	~~Johnnyo's~~	~~3465 S Morgan ST~~	~~***Cicago***~~	~~IL~~	~~60608~~	0.8	1 - 0.8

c1: $\mathrm{DBAName} \rightarrow \mathrm{Zip}$

c2: $\mathrm{Zip} \rightarrow \mathrm{City, State}$

c3: $\mathrm{City, State, Address} \rightarrow \mathrm{Zip}$

Optimization Objective

$$\max_{I} \left(\prod_{t \in I} p(t) \prod_{t \notin I} (1 - p(t)) \right)$$

Most probable repairs

	DBAName	AKAName	Address	City	State	Zip	*p*	*Factor (f)*
~~t1~~	~~John Veliotis Sr.~~	~~Johnnyo's~~	3465 S Morgan ST	~~***Chicago***~~	~~IL~~	~~***60608***~~	0.9	1 - 0.9
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4	0.4
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***	0.4	0.4
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608	0.8	0.8

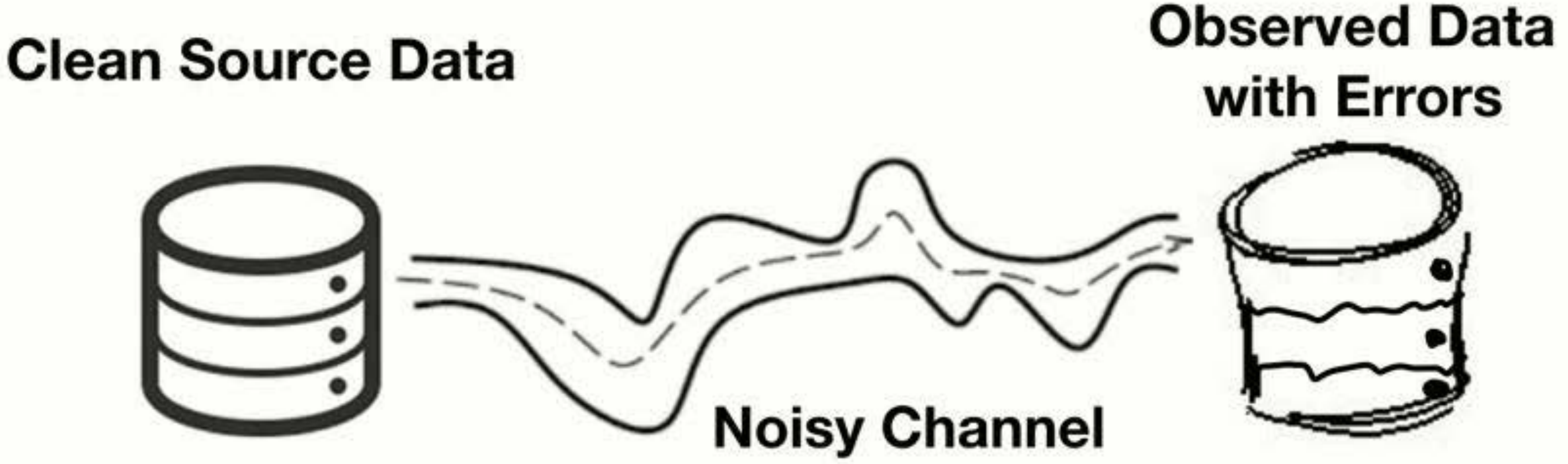
Optimization Objective $\max_{I} \left(\prod_{t \in I} p(t) \prod_{t \notin I} (1 - p(t)) \right)$

Probabilities offer clear semantics than minimality.
Fundamental question: How do we know p?

Probabilistic Unclean Databases

Christopher De Sa, Ihab Ilyas, Benny Kimelfeld, Christopher Ré, Theodoros Rekatsinas, ICDT 2019

The case of a noisy channel for data



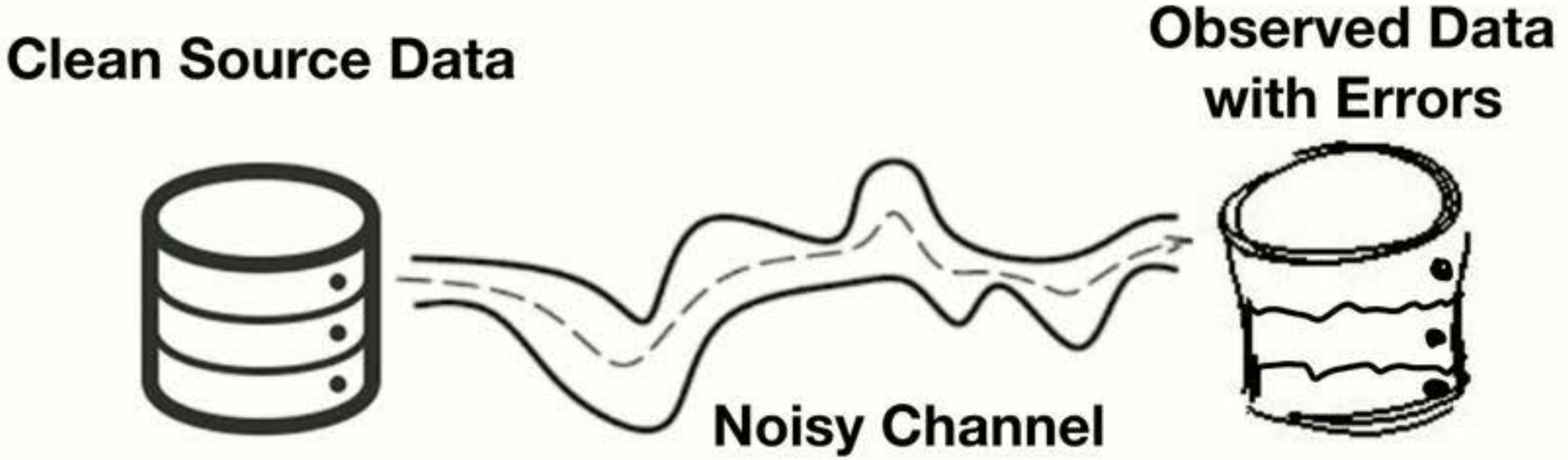
Noisy Channel Model

1. We see an observation *x* in the noisy world
2. Find the correct world *w*

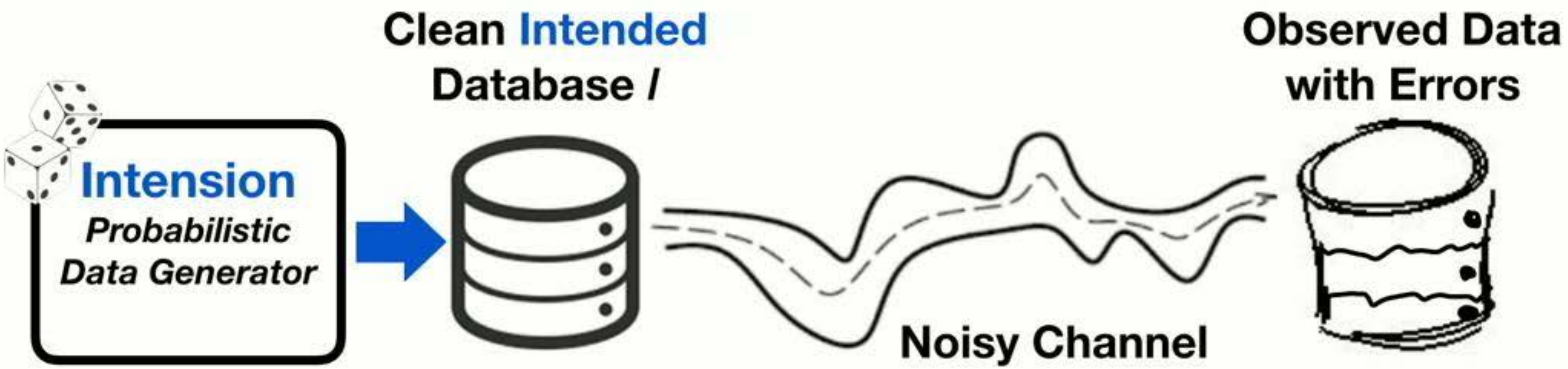
$$\hat{w} = \underset{w \in W}{\arg\max}\, P(w \mid x)$$

Applications: *Speech, OCR, Spelling correction, Part of speech tagging, machine translations, etc…*

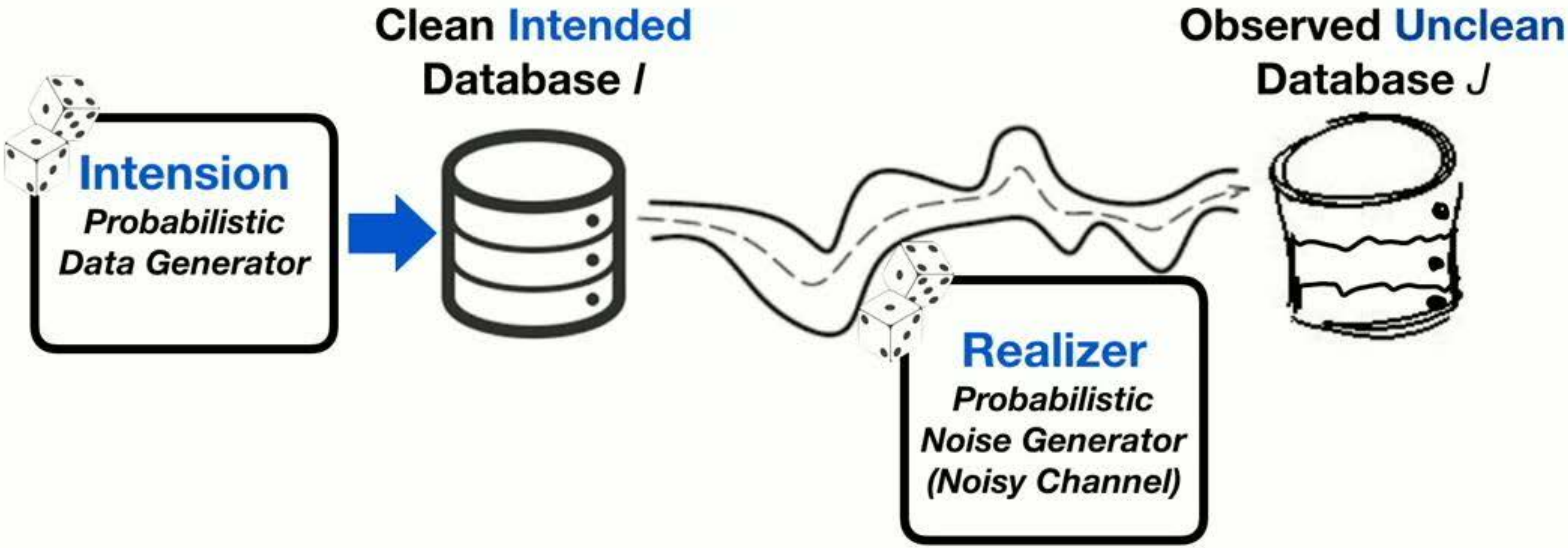
The Probabilistic Unclean Database Model



The Probabilistic Unclean Database Model



The Probabilistic Unclean Database Model



The Probabilistic Unclean Database Model

Intension

Probabilistic Data Generator

A Probability Distribution

$$\mathcal{D}(I) \stackrel{\text{def}}{=} \frac{1}{Z} \times \boxed{\mathcal{K}(I)} \times \boxed{\exp\left(-\sum_{\varphi \in \Phi} w(\varphi) \times |V(\varphi, I)|\right)}$$

Component 1:

Probability over tuple-values in I

Component 2:

Logical constraints bias towards consistency of tuples in I

The Probabilistic Unclean Database Model

Realizer

Probabilistic Noise Generator (Noisy Channel)

A Conditional Probability Distribution

$$\mathcal{R}_I(J) = Pr(J|I)$$

Captures the conditional probability of data edits and transformations

Probability of the i'th record of I changing from t to t'

Example:
Exponential Family Realizer

$$\mathcal{R}[i,t](t') = \frac{1}{Z(t)} \exp\left(\sum_{g \in G} w_g \cdot g(t,t')\right)$$

with $t \in I$, $t' \in J$ and G is a set of features where each g is an arbitrary function over (t,t') and each weight w_g is a real number.

Example PUD instances

Intended Database

Business ID	*City*	*State*	*Zip Code*
Porter	Madison	WI	53703
Graft	Madison	WI	53703
EVP Coffee	Madison	WI	53703

Dirty Database under Subset Realizer

Business ID	*City*	*State*	*Zip Code*
Porter	Madison	WI	53703
Graft	Madison	WI	53703
EVP Coffee	Madison	WI	53703
EVP Coffee	Madison	WI	53703

PUD Example 1:
Parfactor/Subset PUD

Prob. of extra tuples in ***J***

$$\mathcal{R} \circ \mathcal{I}\,(I, J) \stackrel{\text{def}}{=} \mathcal{D}(I) \times \boxed{\prod_{\substack{i \in ids(J) \setminus \\ ids(I)}} \tau[i](J[i])} \times \boxed{\prod_{\substack{i \in ids(\mathcal{D}) \setminus \\ ids(J)}} \tau[i](\perp)}$$

Prob. of no-tuples

Models the generation of duplicate data

Example PUD instances

Intended Database

Business ID	*City*	*State*	*Zip Code*
Porter	Madison	WI	53703
Graft	Madison	WI	53703
EVP Coffee	Madison	WI	53703

Dirty Database under Update Realizer

Business ID	*City*	*State*	*Zip Code*
Porter	Madison	WI	53703
Graft	Madison	WI	53704
EVP Coffee	adison	WI	53703

Prob. of edits present in *J*

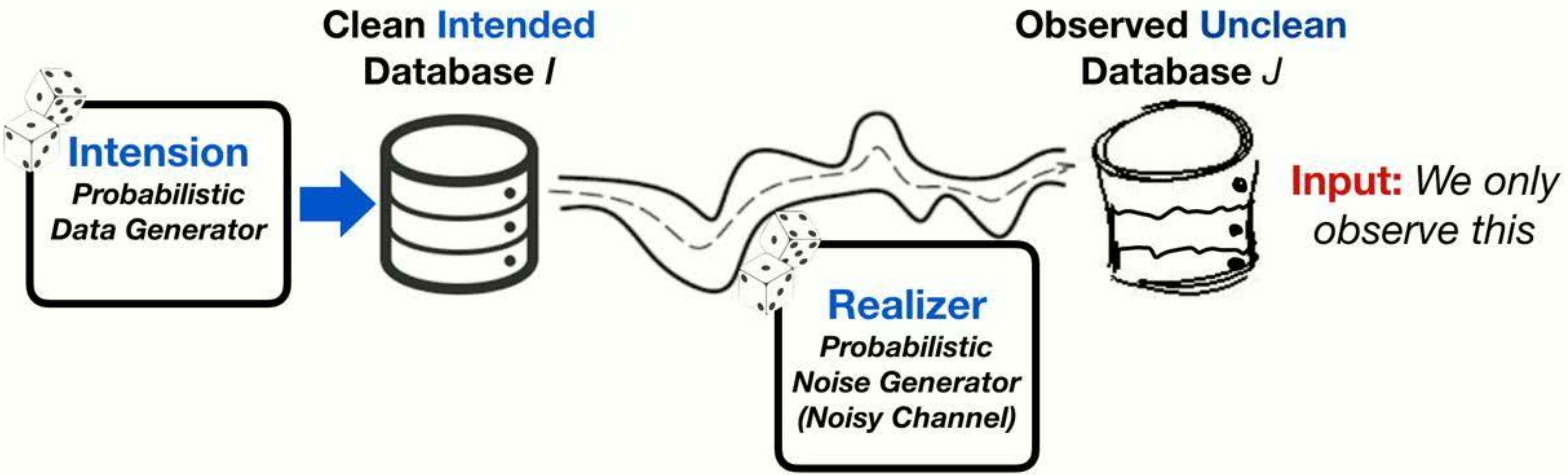
PUD Example 2:
Parfactor/Update PUD

$$\mathcal{R} \circ \mathcal{I}(I, J) \stackrel{\text{def}}{=} \mathcal{D}(I) \times \boxed{\prod_{i \in ids(I)} \kappa[i, I[i]](J[i])}$$

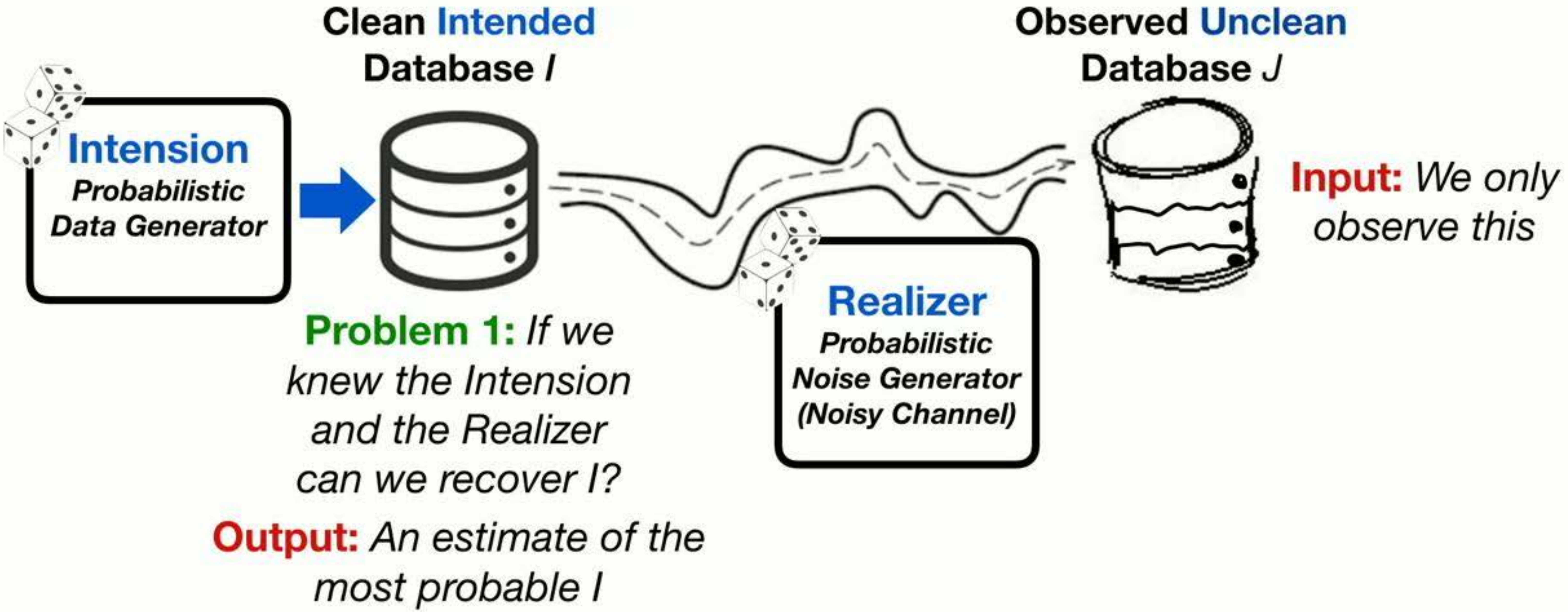
Models errors due to transformations (e.g., typos)

Computational Problems in the PUD framework

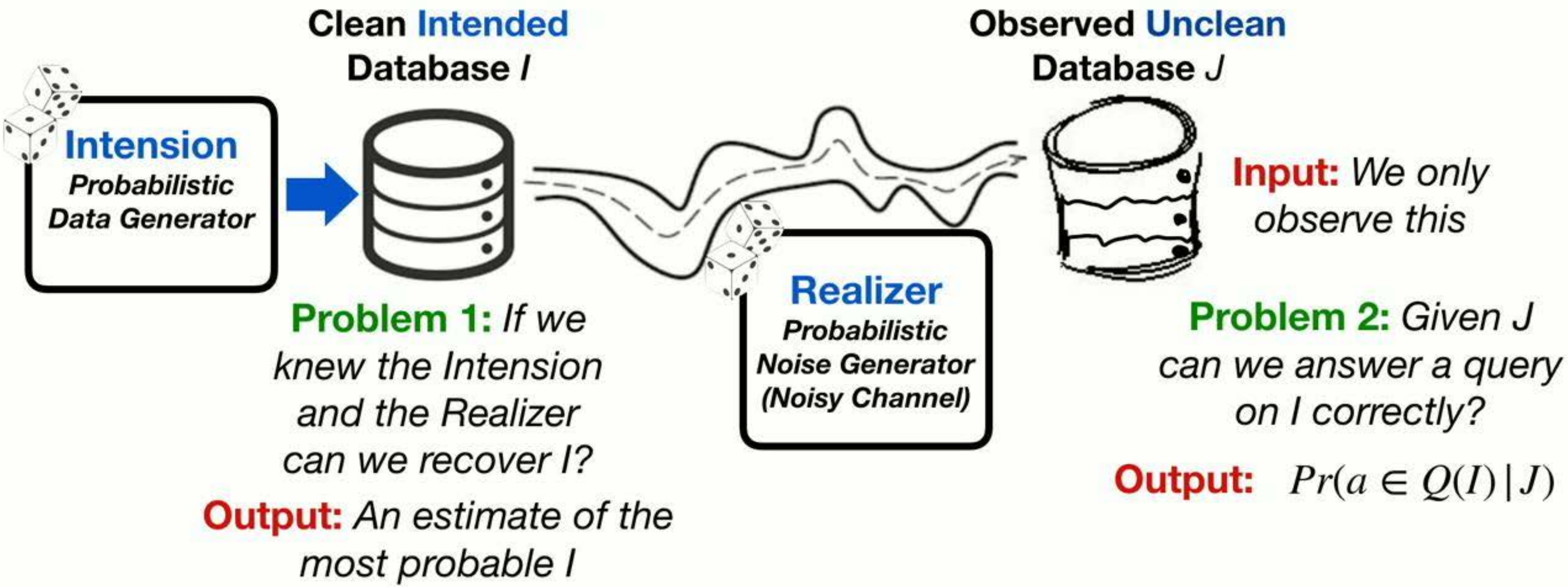
The Probabilistic Unclean Database Model



The Probabilistic Unclean Database Model



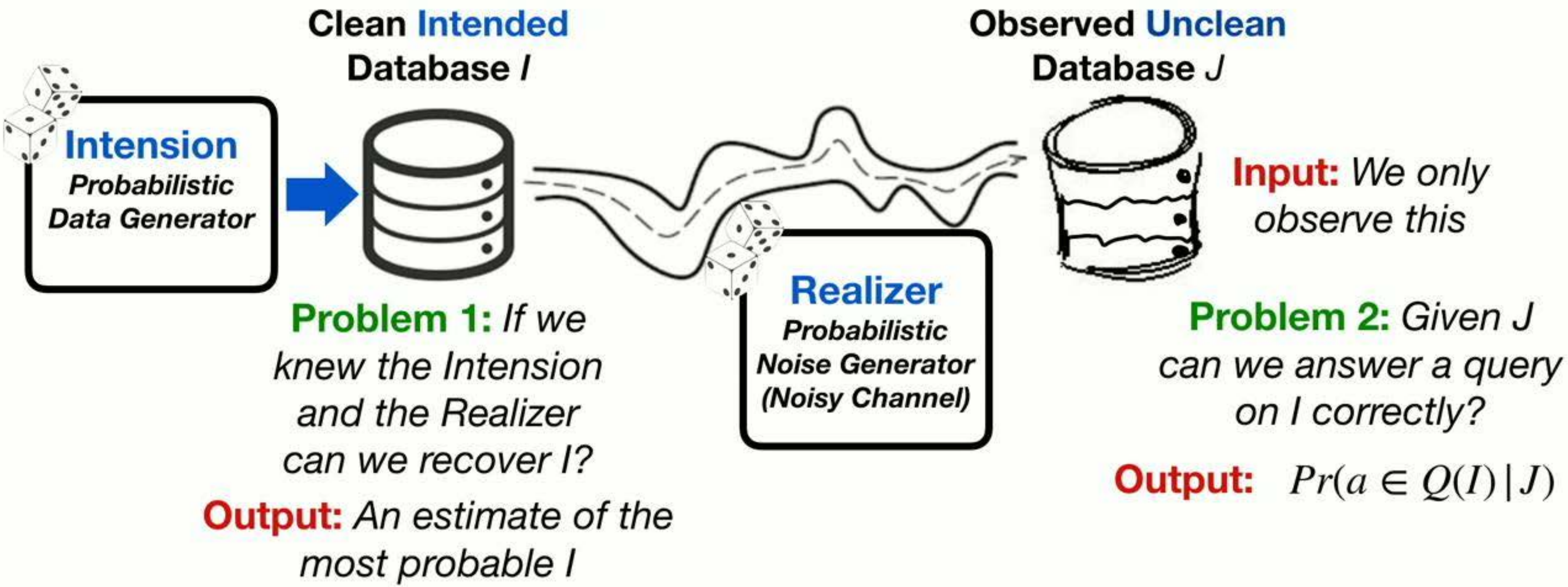
The Probabilistic Unclean Database Model



The Probabilistic Unclean Database Model

Problem 3: *Can we learn the Intension and the Realizer? Can we do that from J (i.e., **without any training data**)?*

Output: *An estimate for the Intension and the Realizer*



Data Cleaning

Problem Statement: *Given the observed noisy database instance J, compute the Most Likely intended database instance I.*

Question: *How does data cleaning in PUDs compare to existing frameworks?*

We show that PUDs generalize existing frameworks:

- MLI in parfactor/subset PUDs generalizes **cardinality repairs**
- MLI in parfactor/update PUDs generalizes **min-update repairs**

Data Cleaning

Problem Statement: *Given the observed noisy database instance J, compute the Most Likely intended database instance I.*

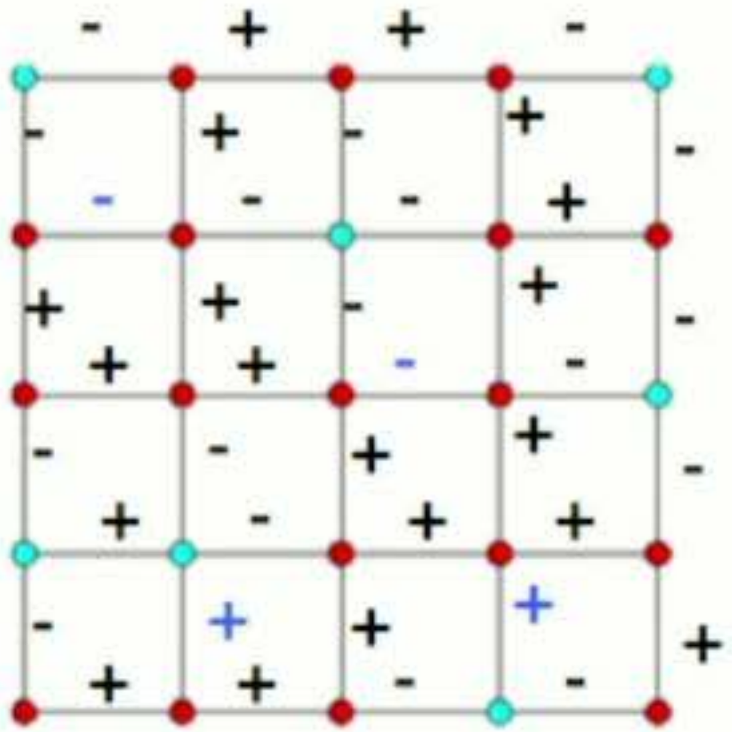
Question: *Is data cleaning in the PUD framework efficient?*

In general no. It is equivalent to probabilistic inference. However:
- For parfactor/subset PUDs with key constraints (i.e., when errors are limited to duplicates) MLI can be computed in polynomial time.
- **New result:** Approximate inference algorithm with guarantees on expected Hamming Error w.r.t. I; uniform noise model [Heidari, Ilyas, Rekatsinas UAI 2019.]

Approximate Inference in Structured Instances with Noisy Categorical Observations

Setup (with noise):

- known graph G = (V,E)
- unknown labeling X: V -> {1, 2, …, k}
- given noisy parity of each edge
  - flipped with probability p
- given noisy observations for each node
  - altered with probability q



Goal: (approximately) recover X.

Formally: want an algorithm A that finds a labeling $\hat{X}$ that minimizes the worst-case expected Hamming error:

$$\max_{X}\{E_{L\sim D(X)}[error(\hat{X}, X)]\}$$

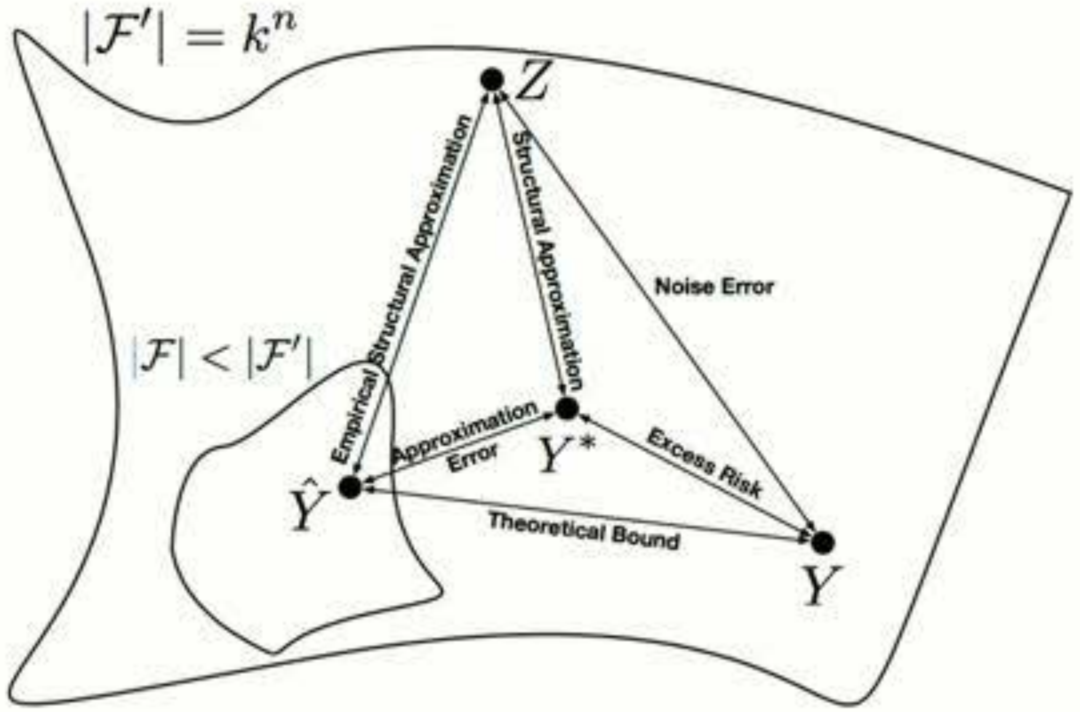
Approximate Inference in Structured Instances with Noisy Categorical Observations

New Algorithm: New approximate inference algorithm based on tree decompositions and correlation clustering.

Guarantees on worst-case expected Hamming error:

- For trees, the Hamming error is upper bounded by $\tilde{O}(\log(k) \cdot p \cdot n)$
- For low-treewidth graphs, the Hamming error is upper bounded by

$$\tilde{O}(k \cdot \log(k) \cdot p^{\lceil \frac{\Delta(G)}{2} \rceil} \cdot n)$$

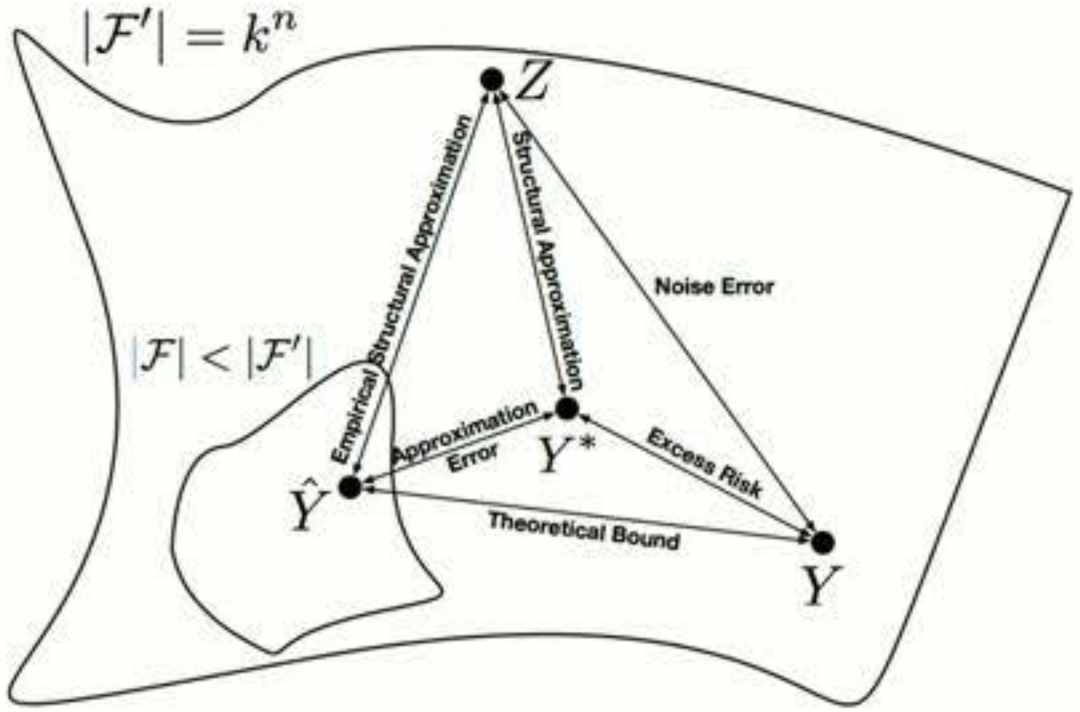


Approximate Inference in Structured Instances with Noisy Categorical Observations

New Algorithm: New approximate inference algorithm based on tree decompositions and correlation clustering.

Guarantees on worst-case expected Hamming error:

- For trees, the Hamming error is upper bounded by $\tilde{O}(\log(k) \cdot p \cdot n)$
- For low-treewidth graphs, the Hamming error is upper bounded by $\tilde{O}(k \cdot \log(k) \cdot p^{\lceil \frac{\Delta(G)}{2} \rceil} \cdot n)$



It should be $p < \sqrt{\frac{1}{k \log k}}$ for the edge side information to be useful for statistical recovery.

PUD learning

Problem Statement: *Assume a parametric representation of the Intention and the Realizer. We want to find the maximum likelihood estimates for the parameters of these representations.*

Supervised variant: *We are given examples of both unclean databases and their clean versions.*
Unsupervised variant: *We are given only unclean databases.*

Question: *Can we learn a PUD? Can we do so without any training data?*

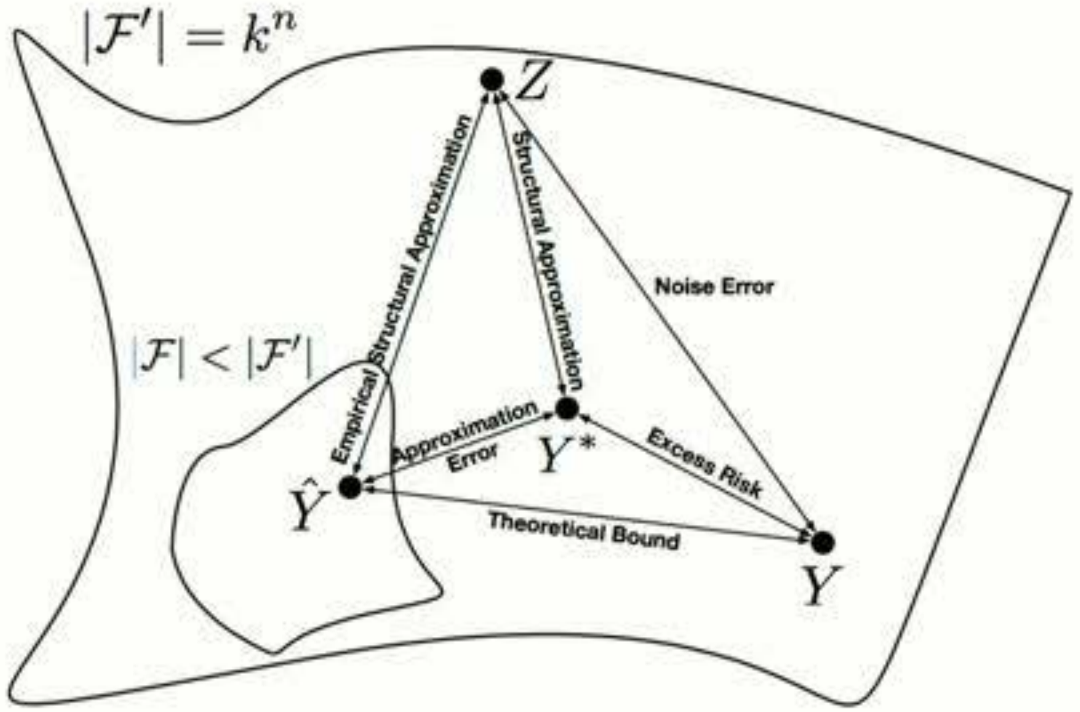
- We show standard learnability results for supervised variant
- **More interesting result:** We show that in the uniform noise model and under tuple independence we can learn a PUD without any training data when the noise is bounded. Single instance J decomposes to multiple training examples. ***Under bounded noise the log-likelihood is convex.***

Approximate Inference in Structured Instances with Noisy Categorical Observations

New Algorithm: New approximate inference algorithm based on tree decompositions and correlation clustering.

Guarantees on worst-case expected Hamming error:

- For trees, the Hamming error is upper bounded by $\tilde{O}(\log(k) \cdot p \cdot n)$
- For low-treewidth graphs, the Hamming error is upper bounded by $$\tilde{O}(k \cdot \log(k) \cdot p^{\lceil \frac{\Delta(G)}{2} \rceil} \cdot n)$$



It should be $p < \sqrt{\frac{1}{k \log k}}$ for the edge side information to be useful for statistical recovery.

PUD learning

Problem Statement: *Assume a parametric representation of the Intention and the Realizer. We want to find the maximum likelihood estimates for the parameters of these representations.*

Supervised variant: *We are given examples of both unclean databases and their clean versions.*
Unsupervised variant: *We are given only unclean databases.*

Question: *Can we learn a PUD? Can we do so without any training data?*

- We show standard learnability results for supervised variant
- **More interesting result:** We show that in the uniform noise model and under tuple independence we can learn a PUD without any training data when the noise is bounded. Single instance J decomposes to multiple training examples. ***Under bounded noise the log-likelihood is convex.***

From Theory to Systems

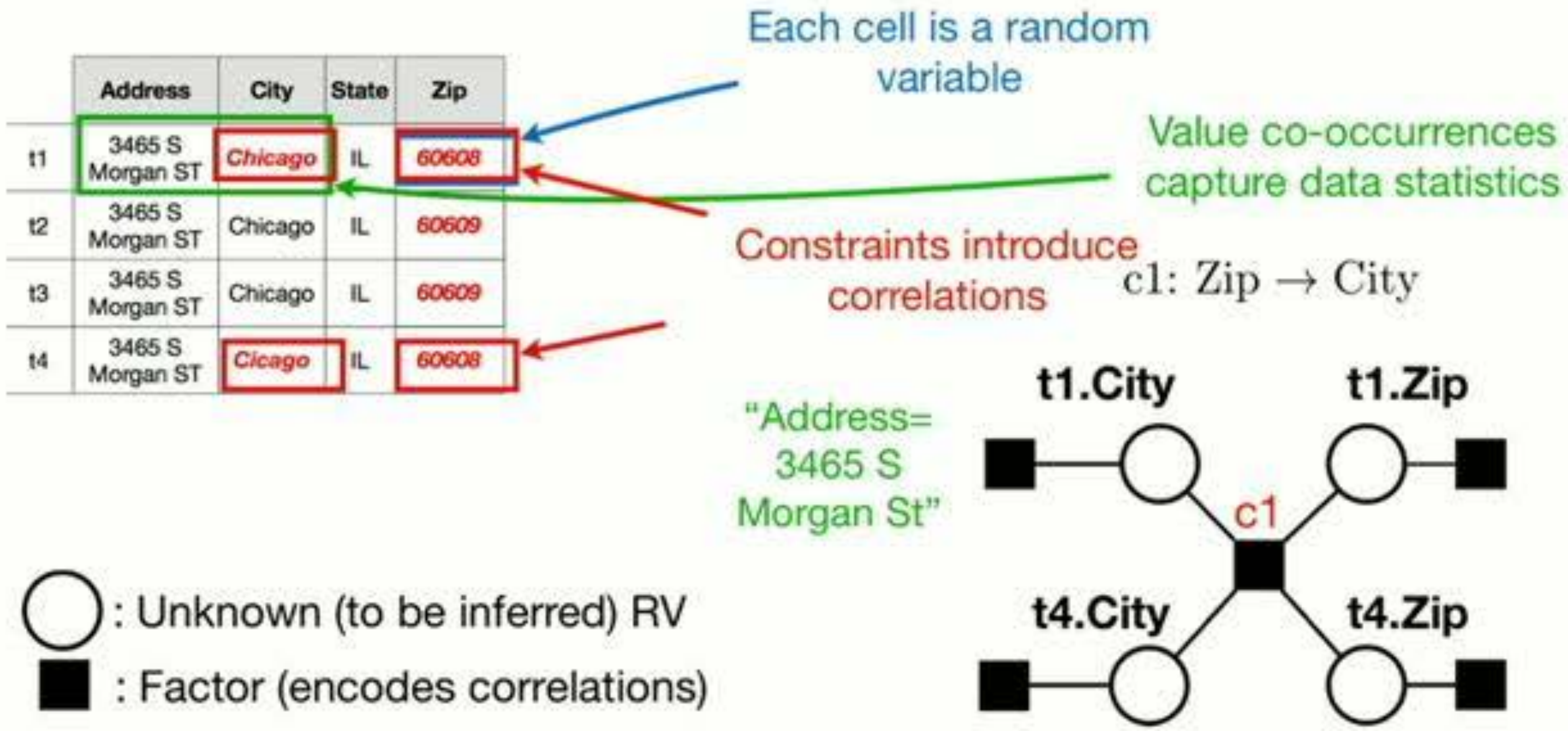
Is the PUDs framework useful in practice?

HoloClean: Probabilistic Data Repairs

HoloClean is the first practical probabilistic data repairing engine and a state-of-the-art data repairing system

HoloClean's factor-graph model is an instantiation of the PUDs Intention model.

HoloClean uses clean cells as training data to learn its PUD Intention model and uses the learned model to approximate MLI repairs.



Reference: HoloClean: Holistic Data Repairs with Probabilistic Inference
Rekatsinas, Chu, Ilyas, Ré, VLDB 2017

HoloClean: Probabilistic Data Repairs

Challenge: Inference under constraints is #P-complete

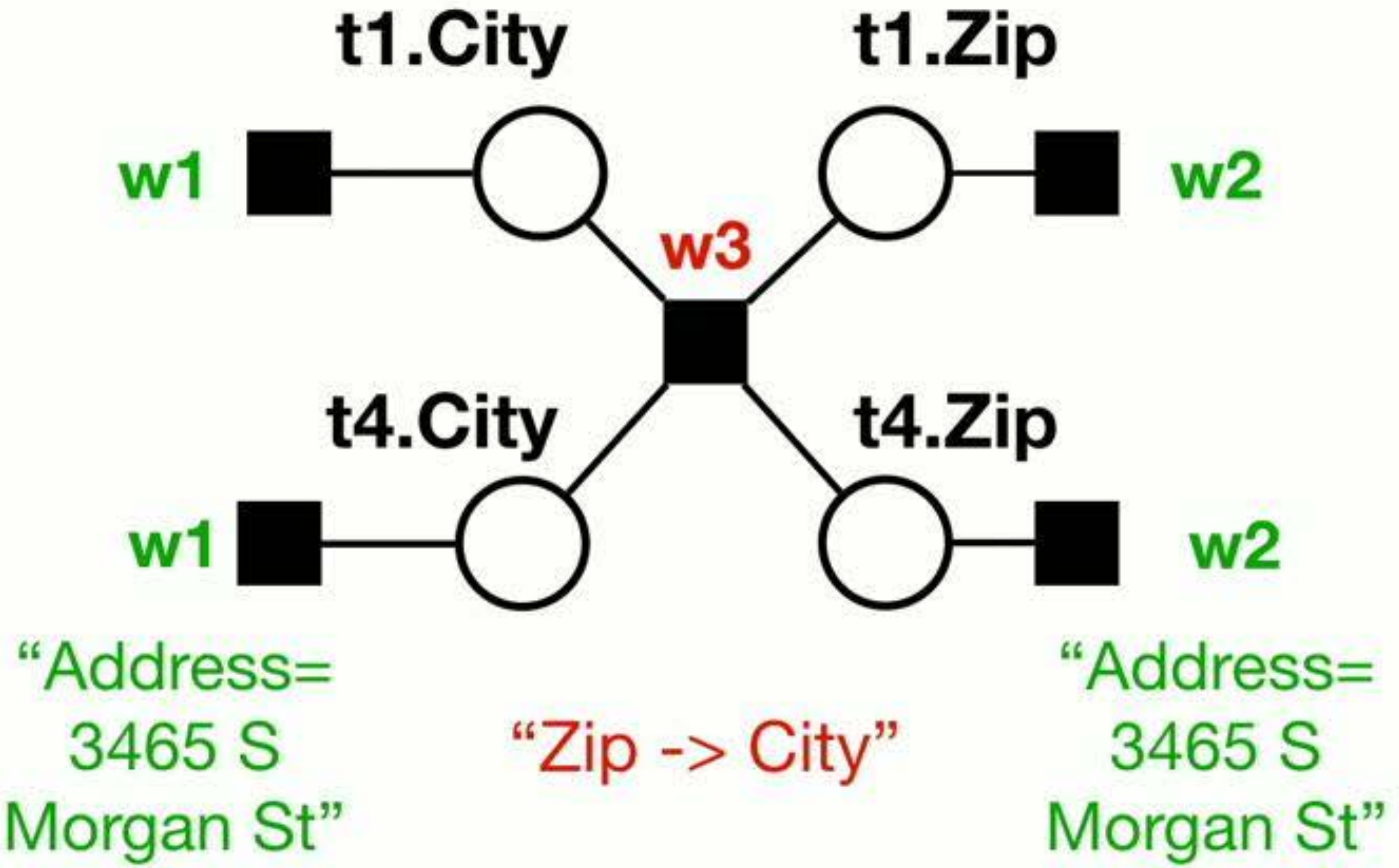
Applying probabilistic inference naively does not scale to data cleaning instances with millions of tuples

Idea 1: Prune domain of random variables.

Idea 2: Relax constraints over sets of random variables to features over independent random variables.

Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*



HoloClean: Probabilistic Data Repairs

Challenge: Inference under constraints is #P-complete

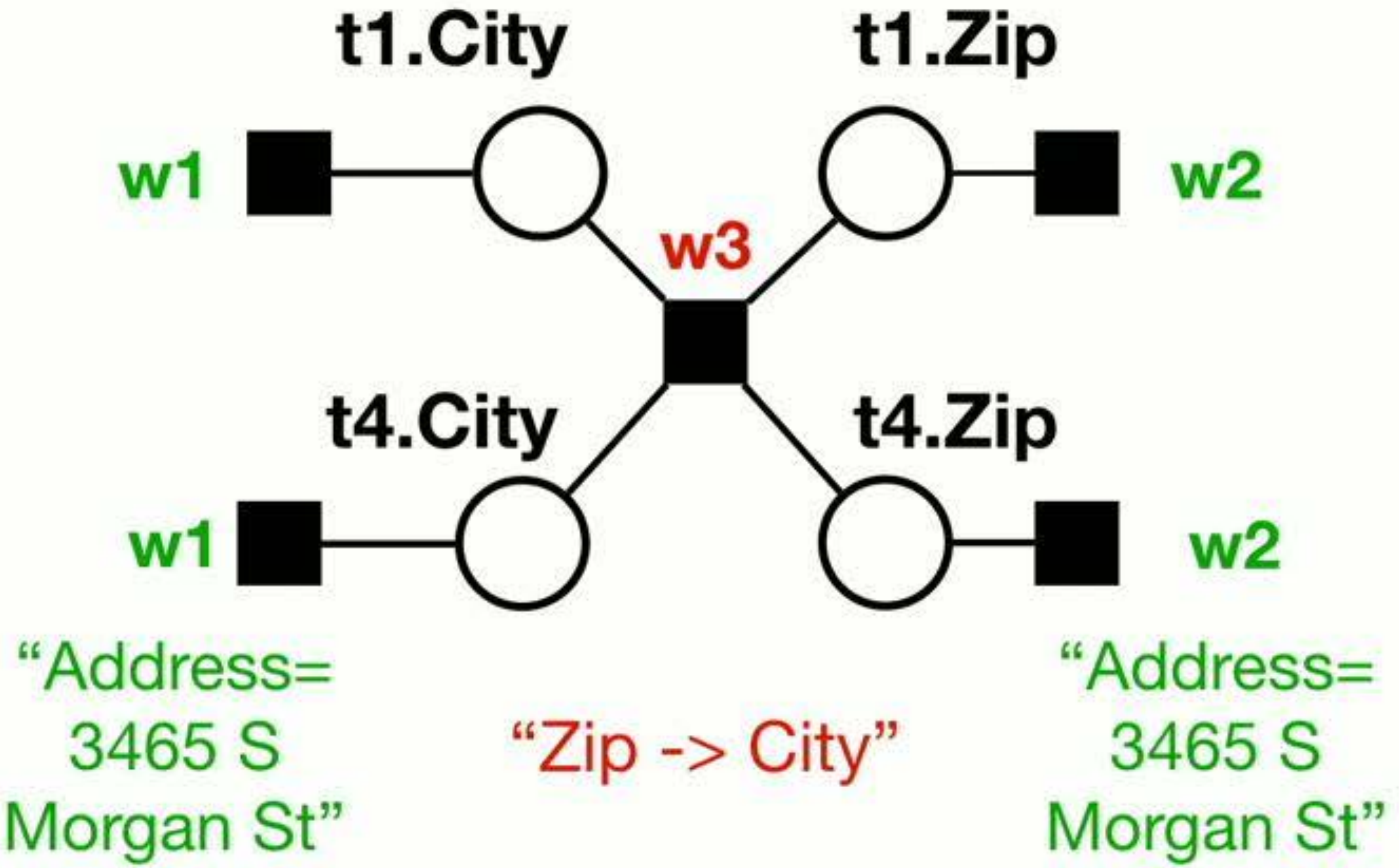
Applying probabilistic inference naively does not scale to data cleaning instances with millions of tuples

Idea 1: Prune domain of random variables.

Idea 2: Relax constraints over sets of random variables to features over independent random variables.

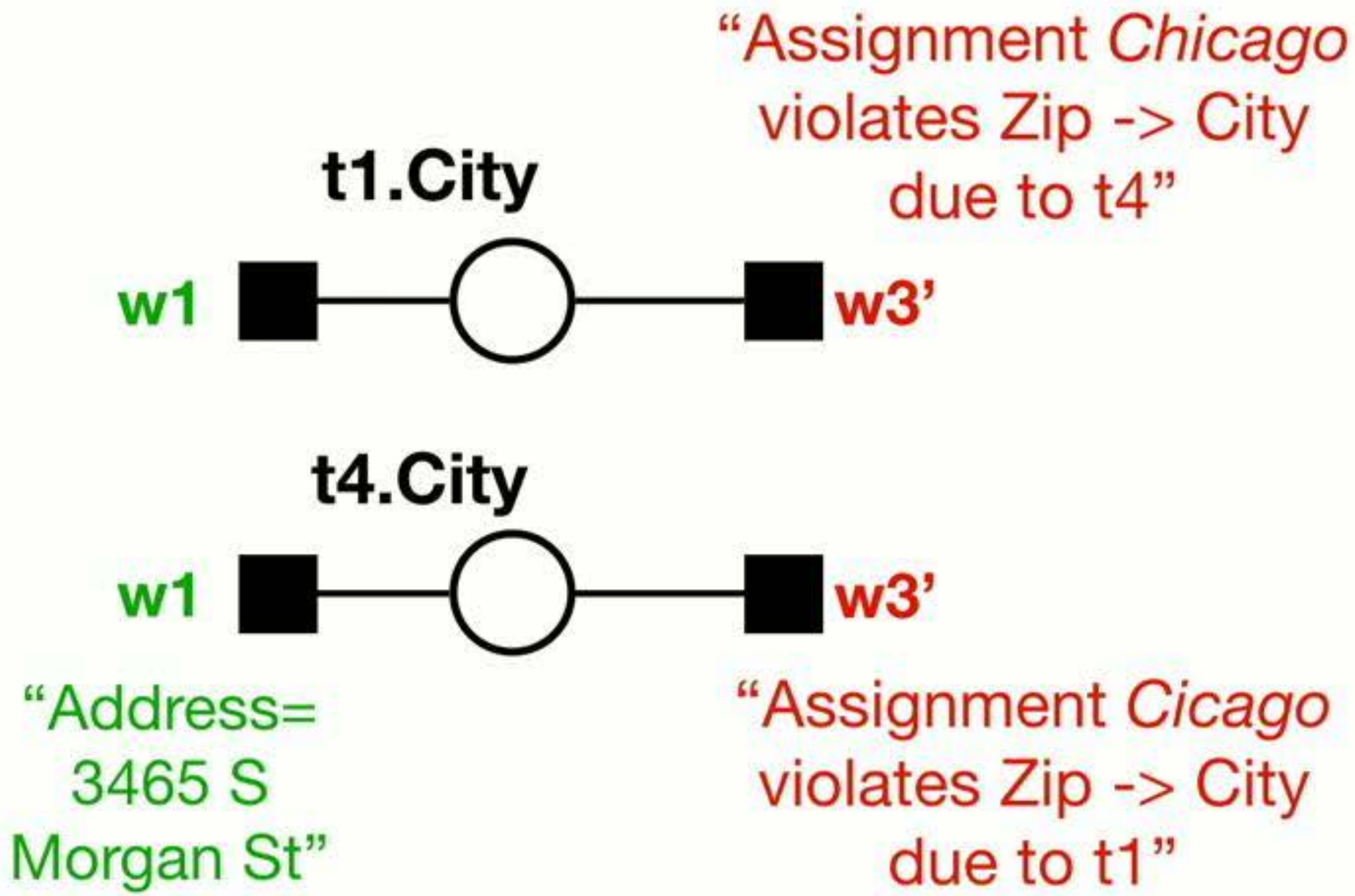
Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*



Relaxing constraints

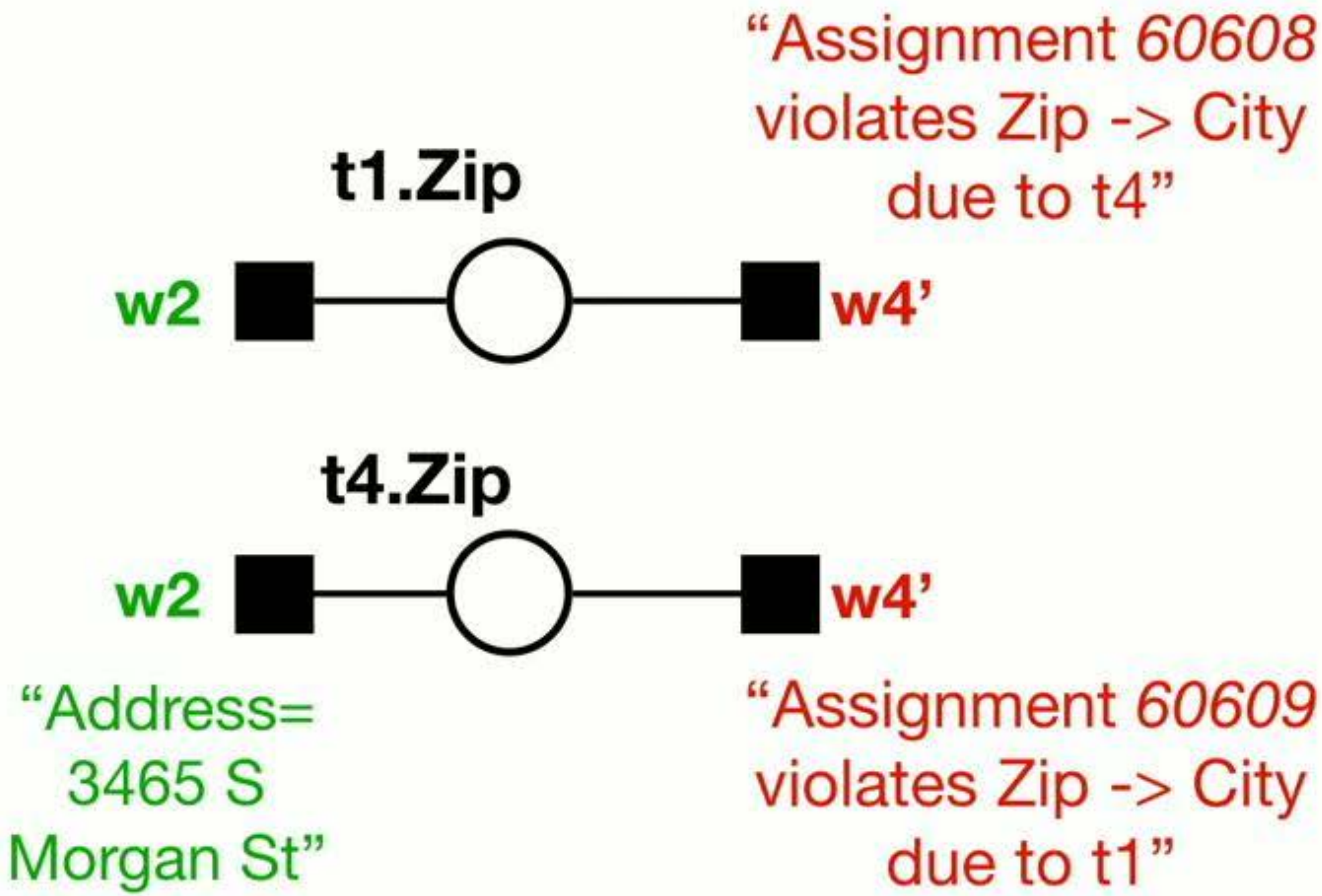
	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*



We have one ***relaxed factor*** for each value in the domain of the RV

Relaxing constraints

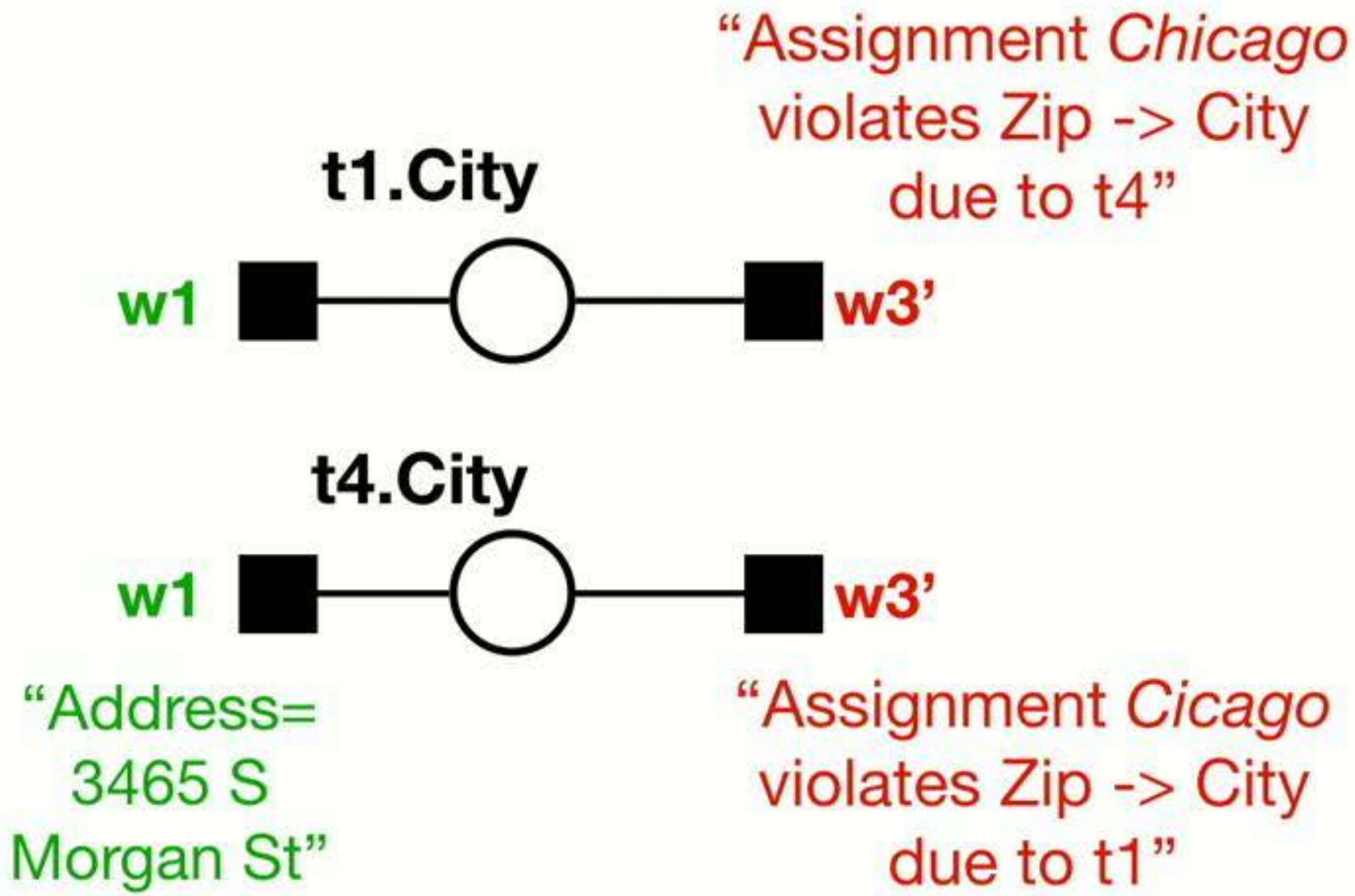
	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	3465 S Morgan ST	Chicago	IL	***60609***
t3	3465 S Morgan ST	Chicago	IL	***60609***
t4	3465 S Morgan ST	***Cicago***	IL	***60608***



We have one ***relaxed factor*** for each value in the domain of the RV

Relaxing constraints

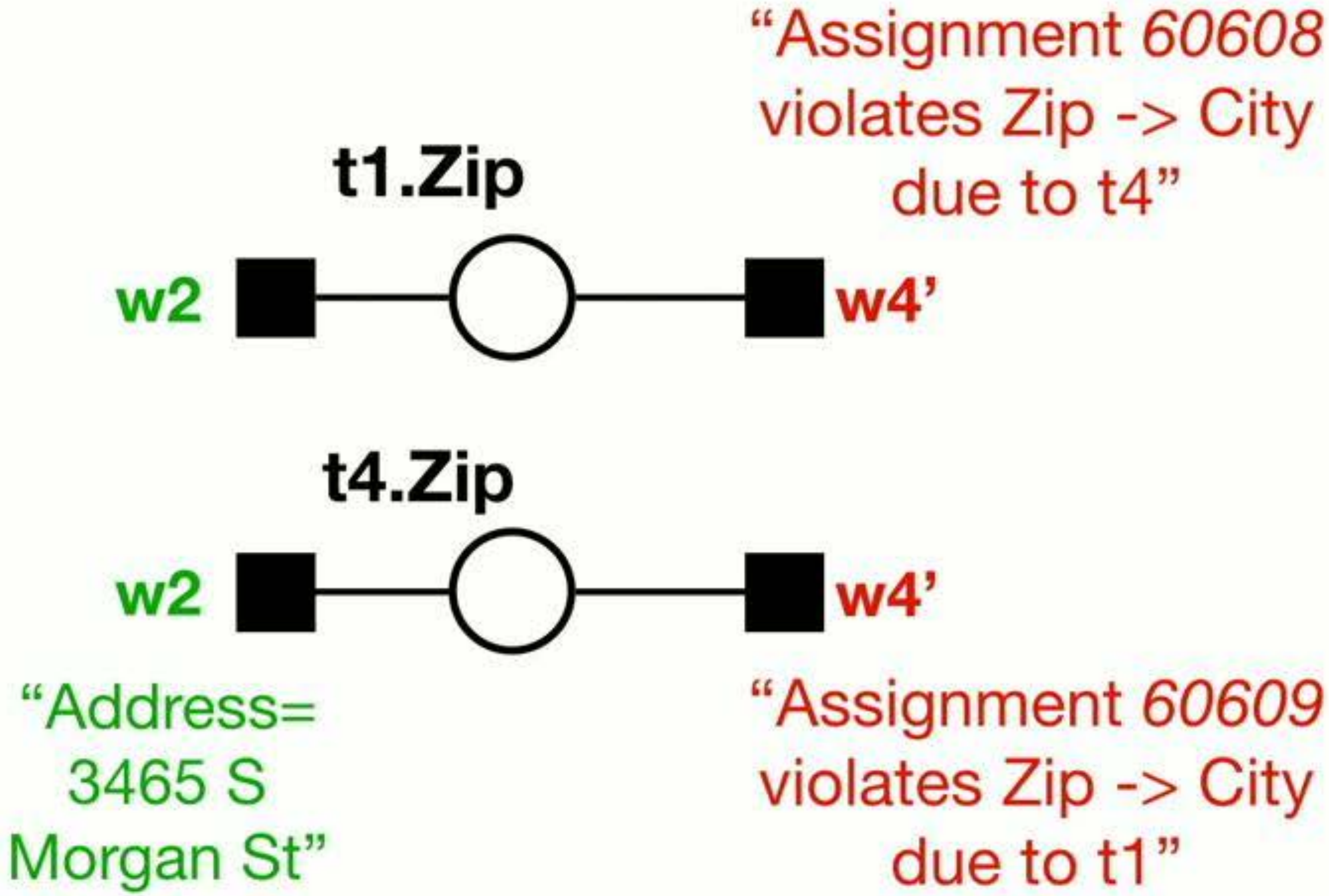
	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*



We have one ***relaxed factor*** for each value in the domain of the RV

Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*

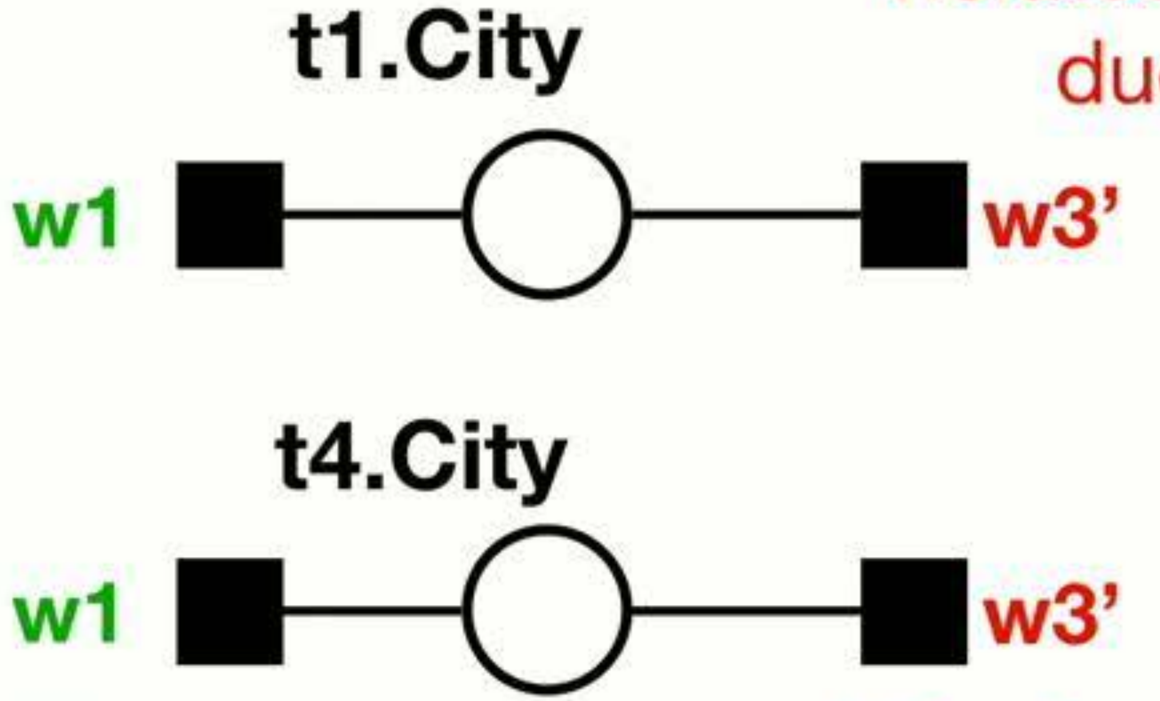


We have one ***relaxed factor*** for each value in the domain of the RV

Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*

"Assignment *Chicago* violates Zip -> City due to t4"



"Address= 3465 S Morgan St"

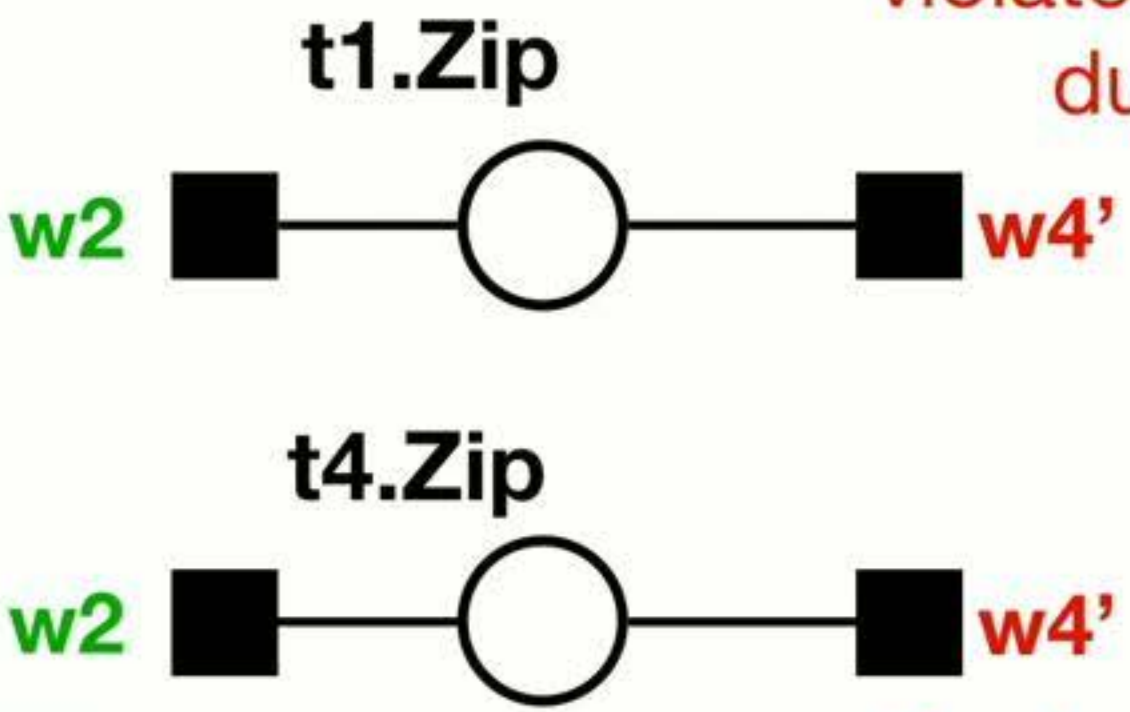
"Assignment *Cicago* violates Zip -> City due to t1"

We have one ***relaxed factor*** for each value in the domain of the RV

Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*

“Assignment *60608* violates Zip -> City due to t4”

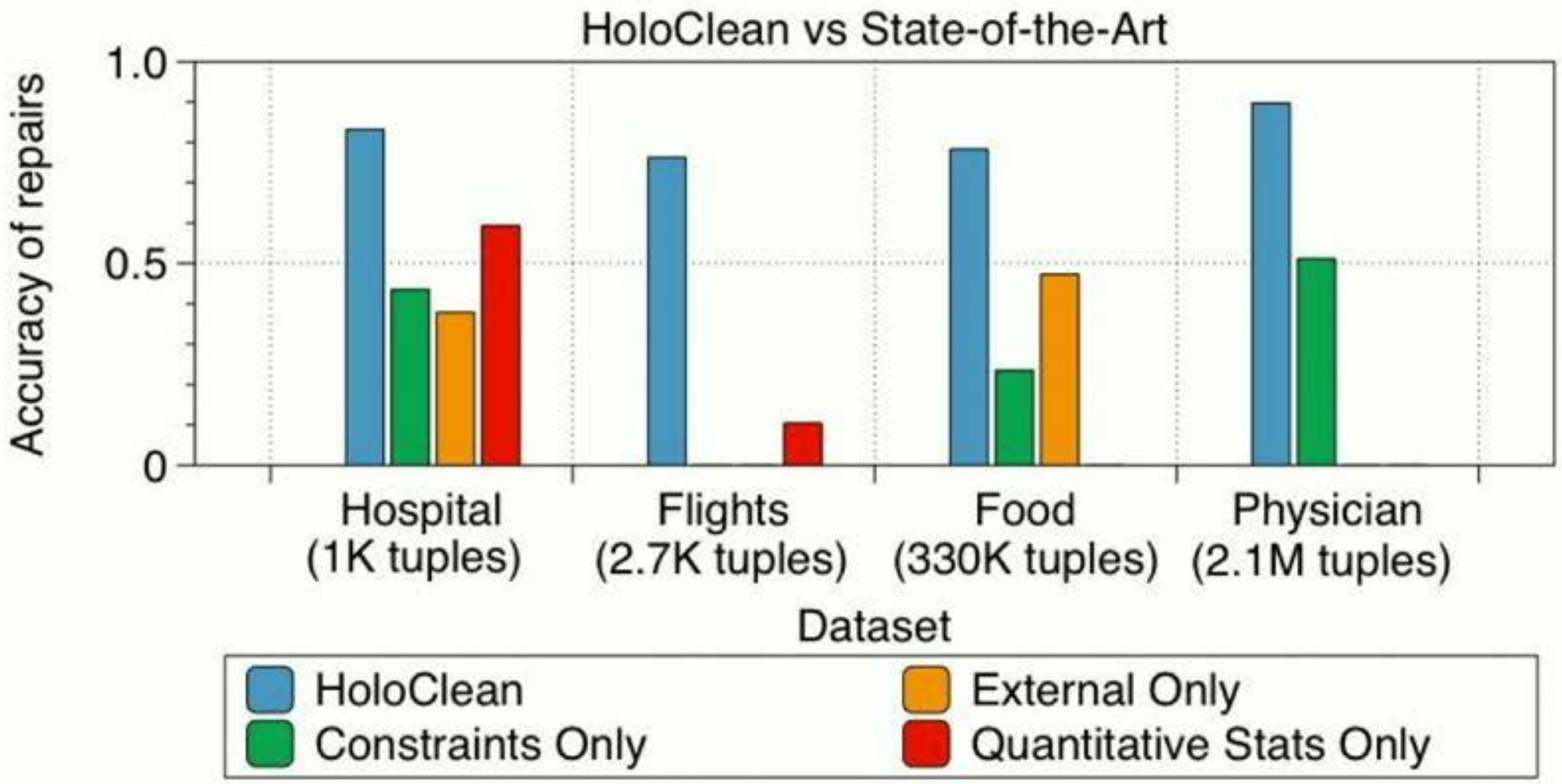


“Address= 3465 S Morgan St”

“Assignment *60609* violates Zip -> City due to t1”

We have one ***relaxed factor*** for each value in the domain of the RV

HoloClean in practice



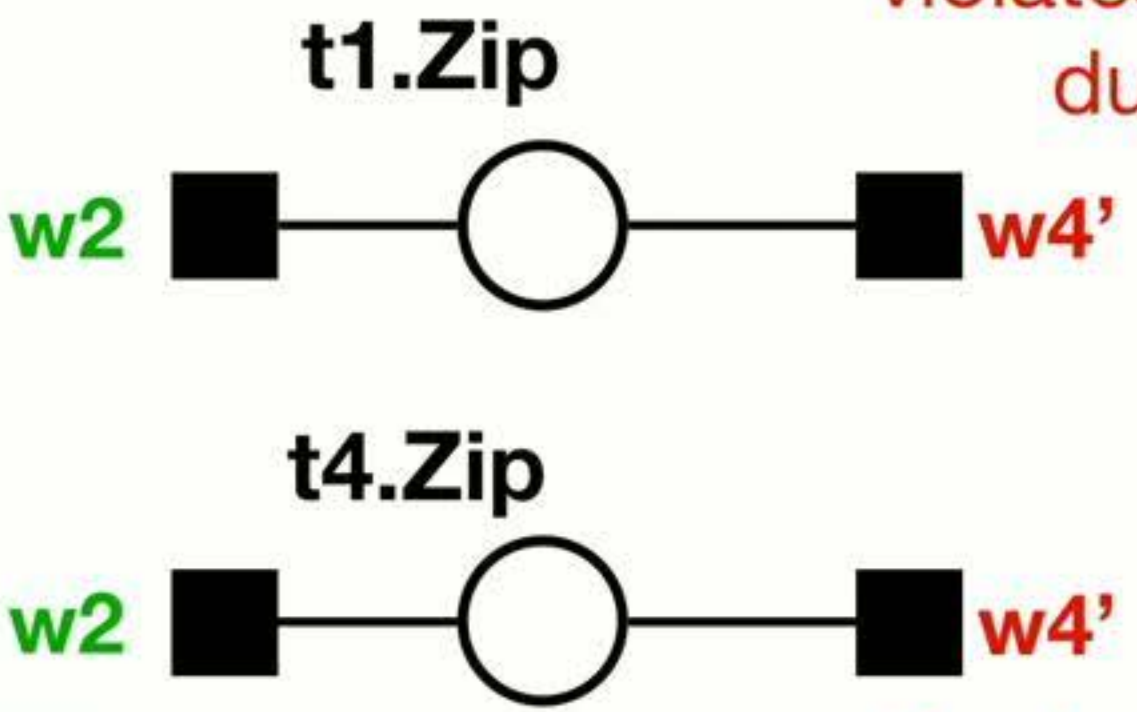
Competing methods do not scale or perform correct repairs.

HoloClean: our approach combining all signals and using inference
Holistic[Chu,2013]: state-of-the-art for constraints & minimality
KATARA[Chu,2015]: state-of-the-art for external data
SCARE[Yakout,2013]: state-of-the-art ML & qualitative statistics

Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	***Cicago***	IL	*60608*

"Assignment *60608* violates Zip -> City due to t4"



"Address= 3465 S Morgan St"

"Assignment *60609* violates Zip -> City due to t1"

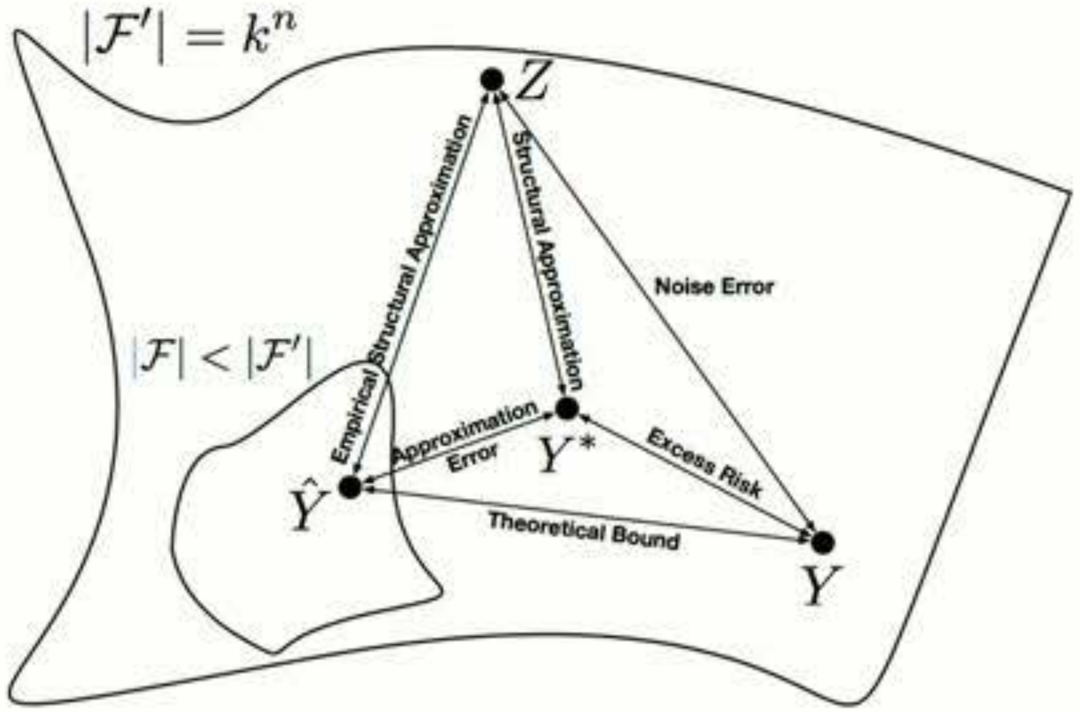
We have one ***relaxed factor*** for each value in the domain of the RV

Approximate Inference in Structured Instances with Noisy Categorical Observations

New Algorithm: New approximate inference algorithm based on tree decompositions and correlation clustering.

Guarantees on worst-case expected Hamming error:

- For trees, the Hamming error is upper bounded by $\tilde{O}(\log(k) \cdot p \cdot n)$
- For low-treewidth graphs, the Hamming error is upper bounded by
$$\tilde{O}(k \cdot \log(k) \cdot p^{\lceil \frac{\Delta(G)}{2} \rceil} \cdot n)$$



It should be $p < \sqrt{\frac{1}{k \log k}}$ for the edge side information to be useful for statistical recovery.

PUD learning

Problem Statement: *Assume a parametric representation of the Intention and the Realizer. We want to find the maximum likelihood estimates for the parameters of these representations.*

Supervised variant: *We are given examples of both unclean databases and their clean versions.*
Unsupervised variant: *We are given only unclean databases.*

Question: *Can we learn a PUD? Can we do so without any training data?*

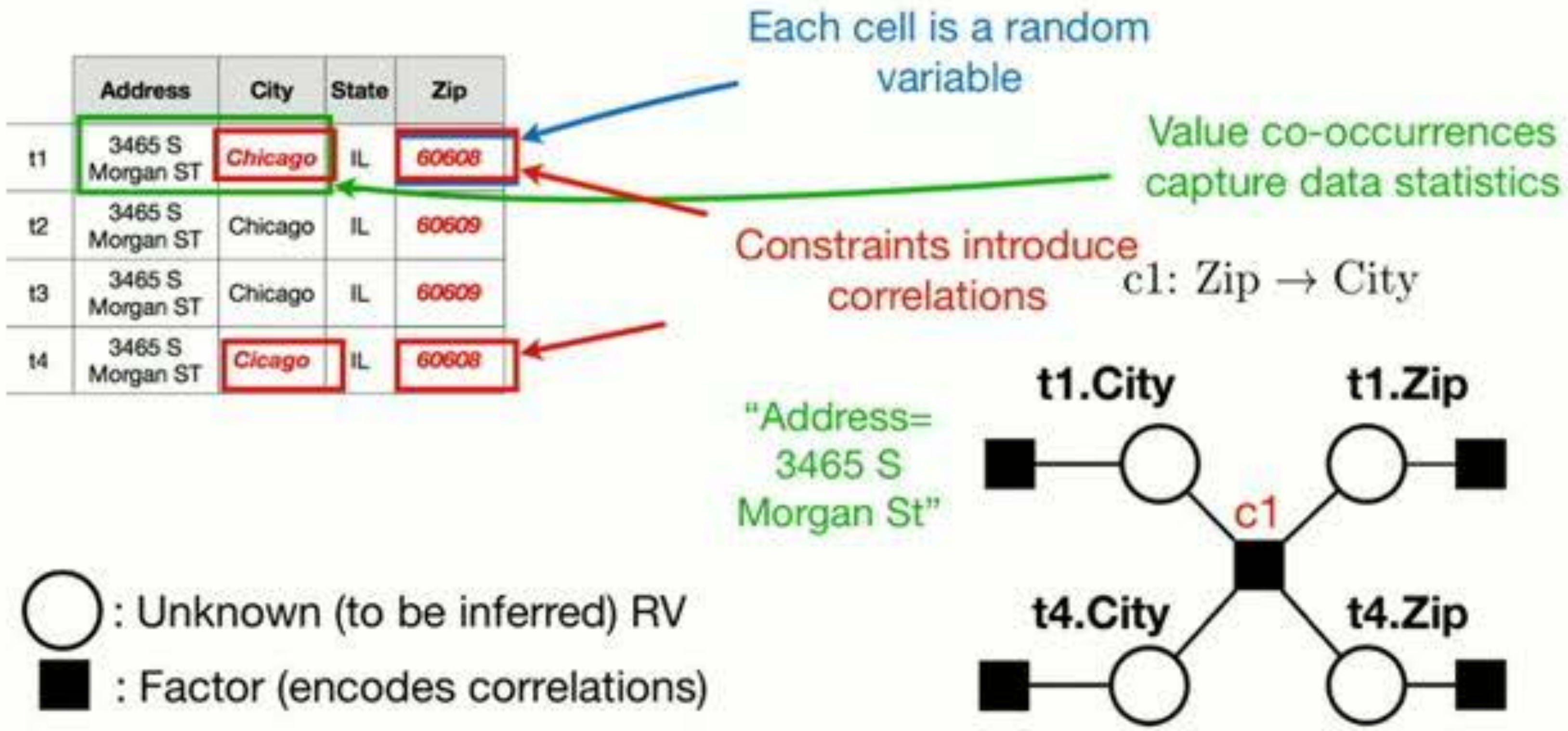
- We show standard learnability results for supervised variant
- **More interesting result:** We show that in the uniform noise model and under tuple independence we can learn a PUD without any training data when the noise is bounded. Single instance J decomposes to multiple training examples. ***Under bounded noise the log-likelihood is convex.***

HoloClean: Probabilistic Data Repairs

HoloClean is the first practical probabilistic data repairing engine and a state-of-the-art data repairing system

HoloClean's factor-graph model is an instantiation of the PUDs Intention model.

HoloClean uses clean cells as training data to learn its PUD Intention model and uses the learned model to approximate MLI repairs.



Reference: HoloClean: Holistic Data Repairs with Probabilistic Inference
Rekatsinas, Chu, Ilyas, Ré, VLDB 2017

HoloClean: Probabilistic Data Repairs

Challenge: Inference under constraints is #P-complete

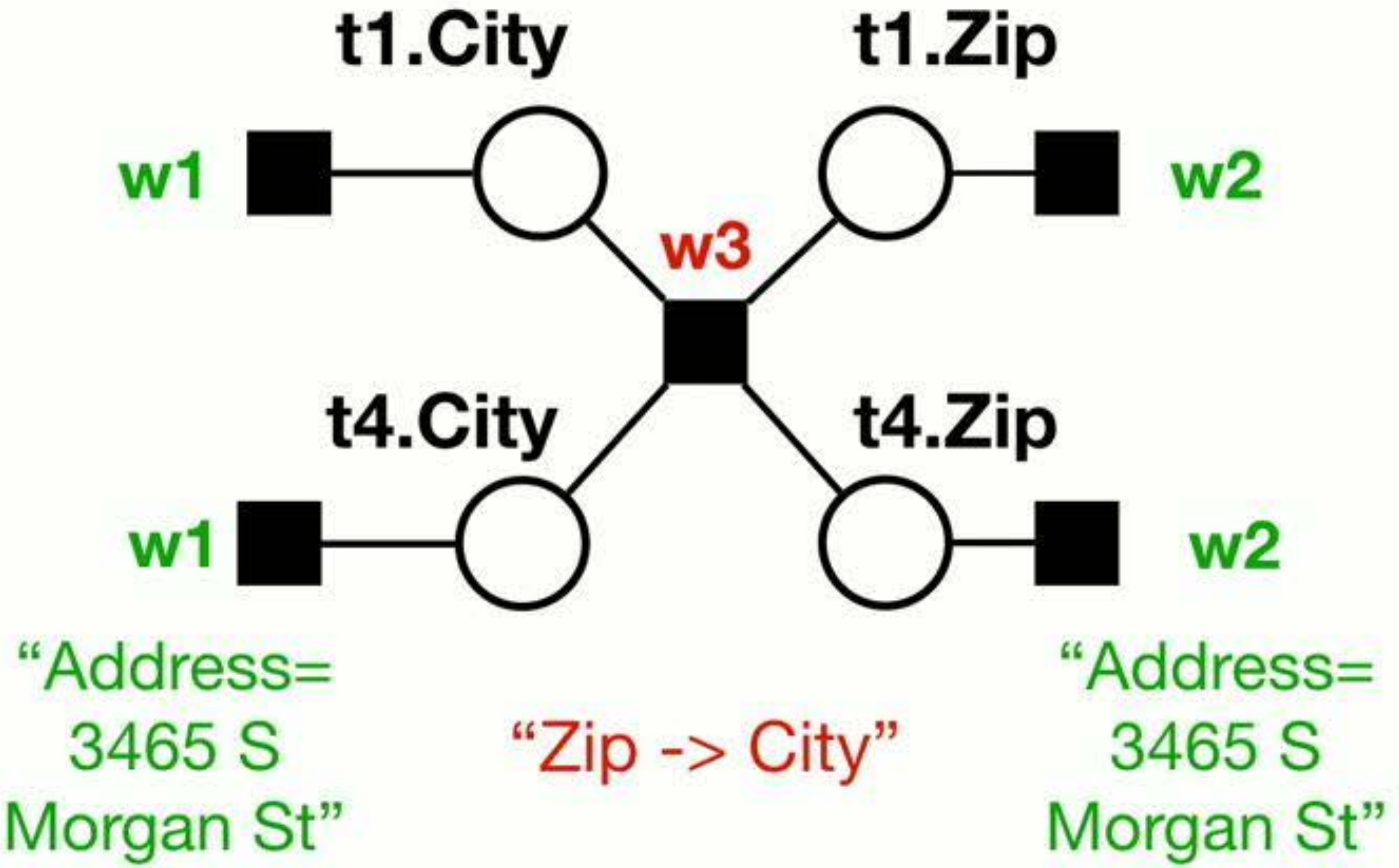
Applying probabilistic inference naively does not scale to data cleaning instances with millions of tuples

Idea 1: Prune domain of random variables.

Idea 2: Relax constraints over sets of random variables to features over independent random variables.

Relaxing constraints

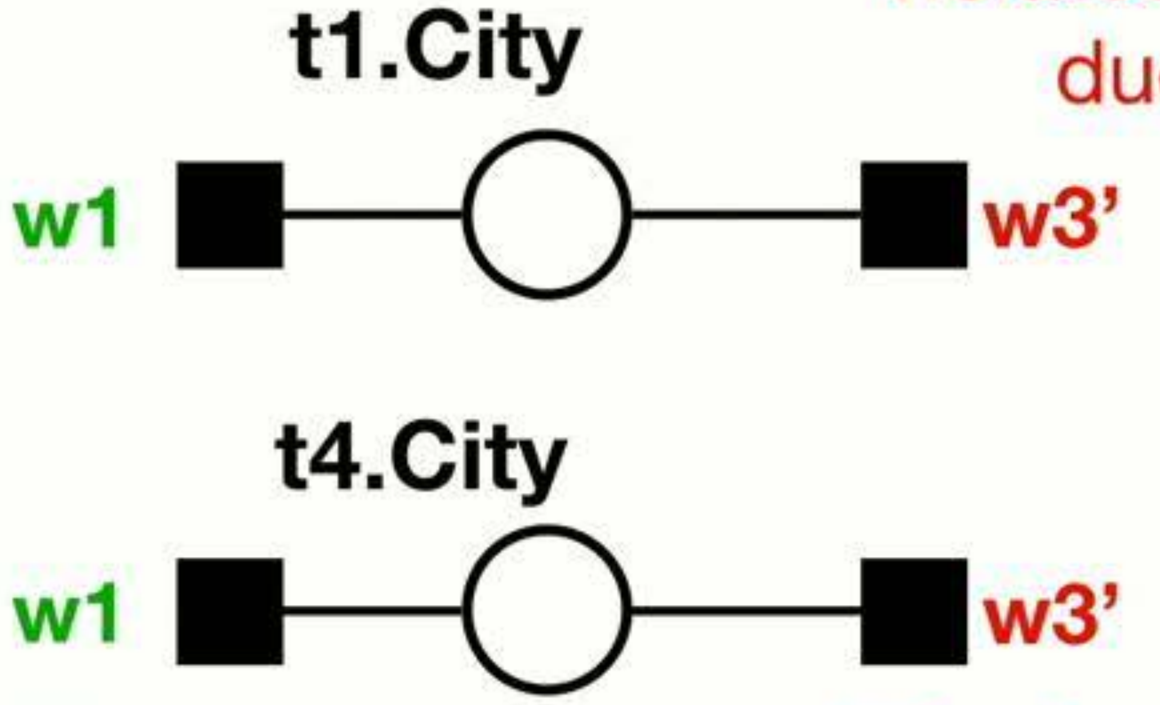
	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	*60608*
t2	3465 S Morgan ST	Chicago	IL	*60609*
t3	3465 S Morgan ST	Chicago	IL	*60609*
t4	3465 S Morgan ST	*Cicago*	IL	*60608*



Relaxing constraints

	Address	City	State	Zip
t1	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	3465 S Morgan ST	Chicago	IL	***60609***
t3	3465 S Morgan ST	Chicago	IL	***60609***
t4	3465 S Morgan ST	***Cicago***	IL	***60608***

"Assignment *Chicago* violates Zip -> City due to t4"

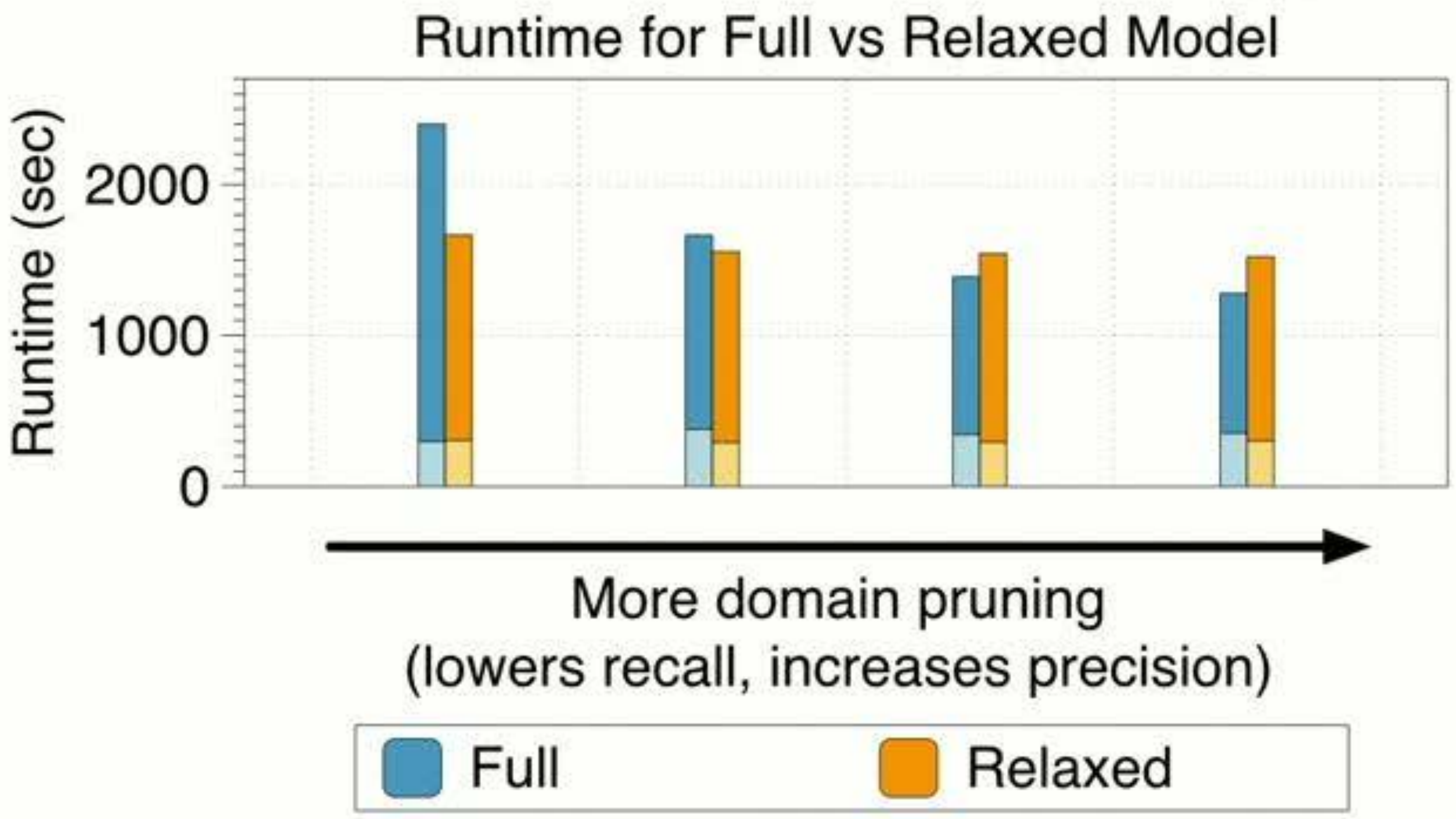


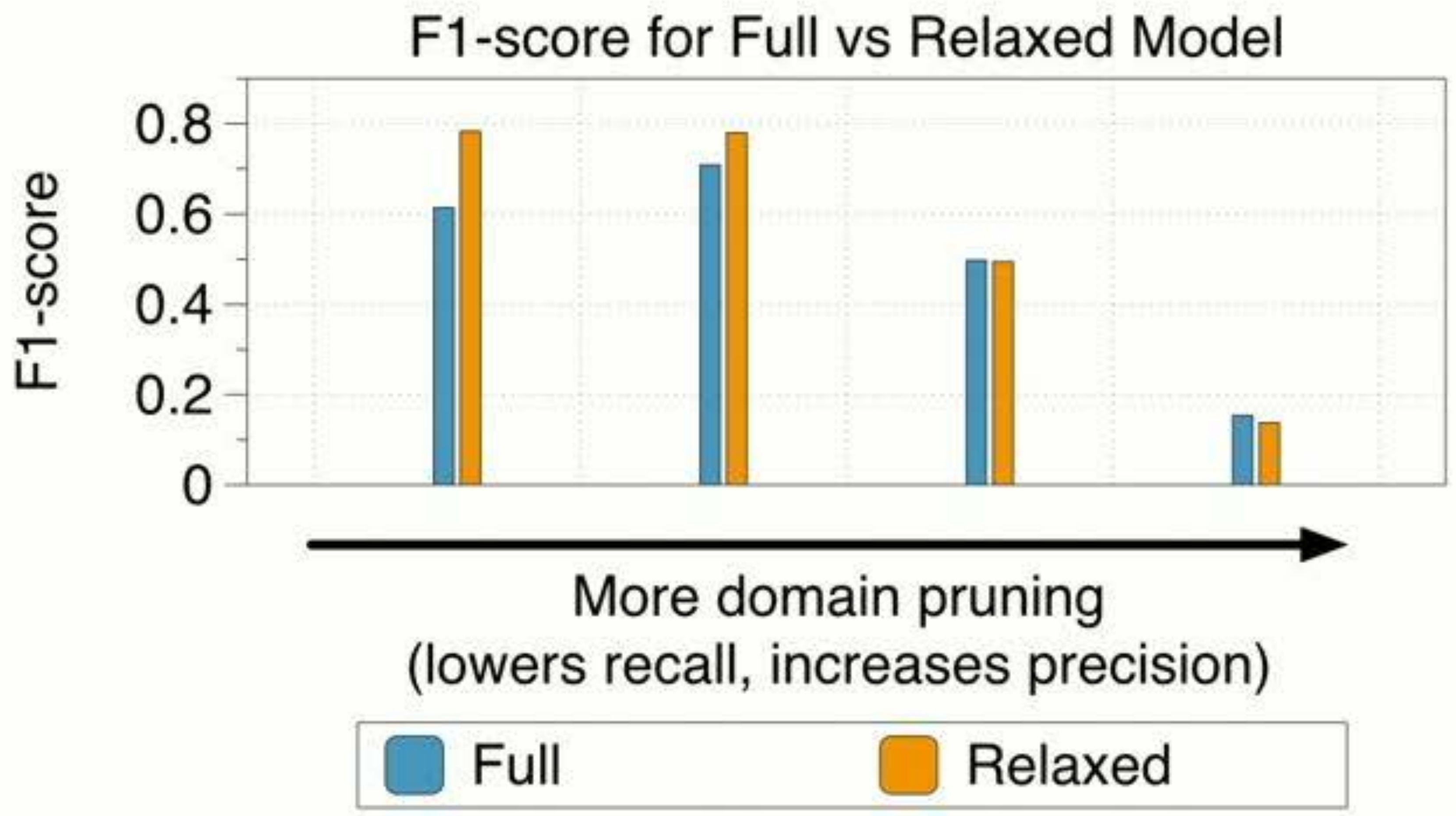
"Address= 3465 S Morgan St"

"Assignment *Cicago* violates Zip -> City due to t1"

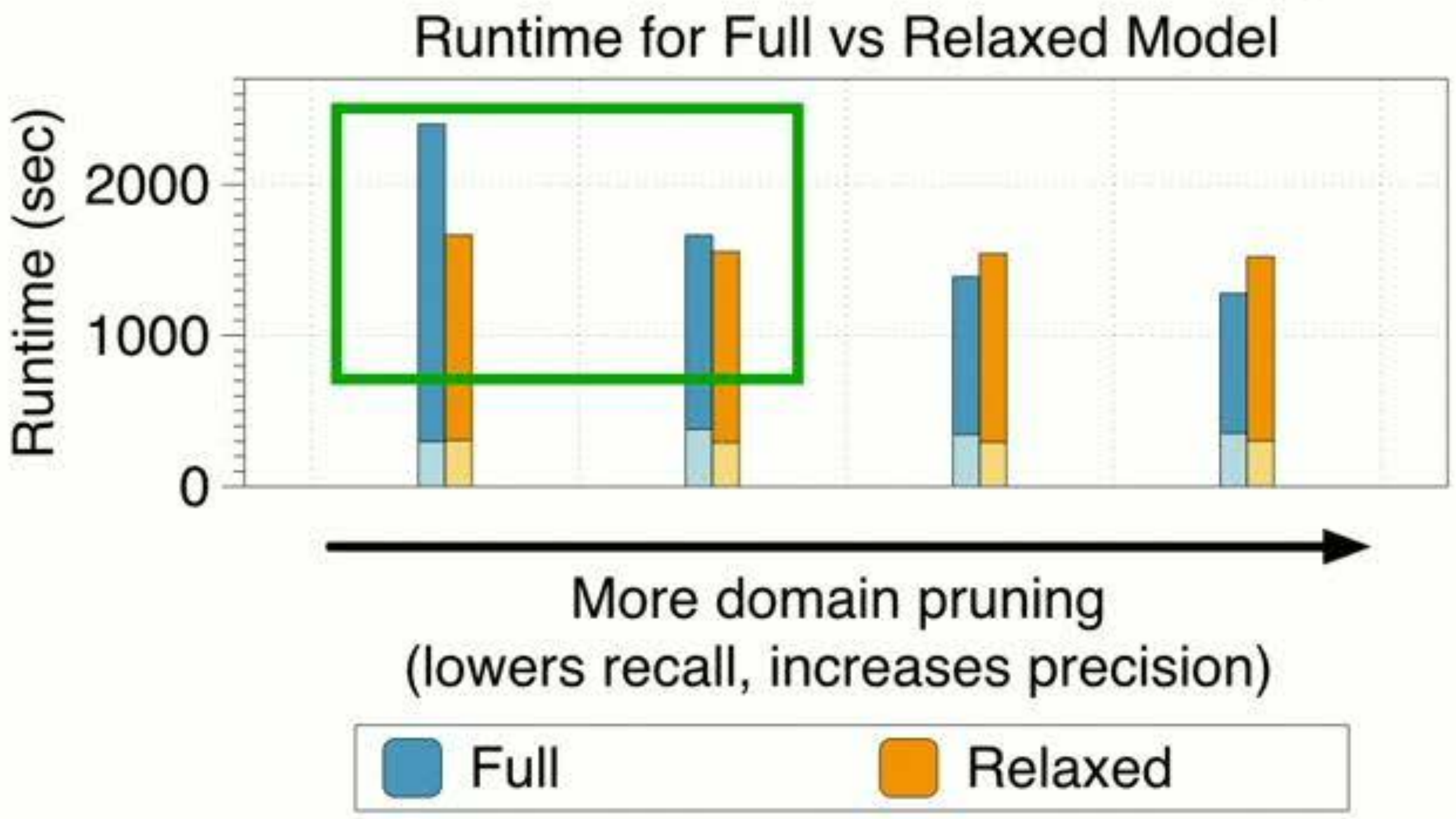
We have one ***relaxed factor*** for each value in the domain of the RV

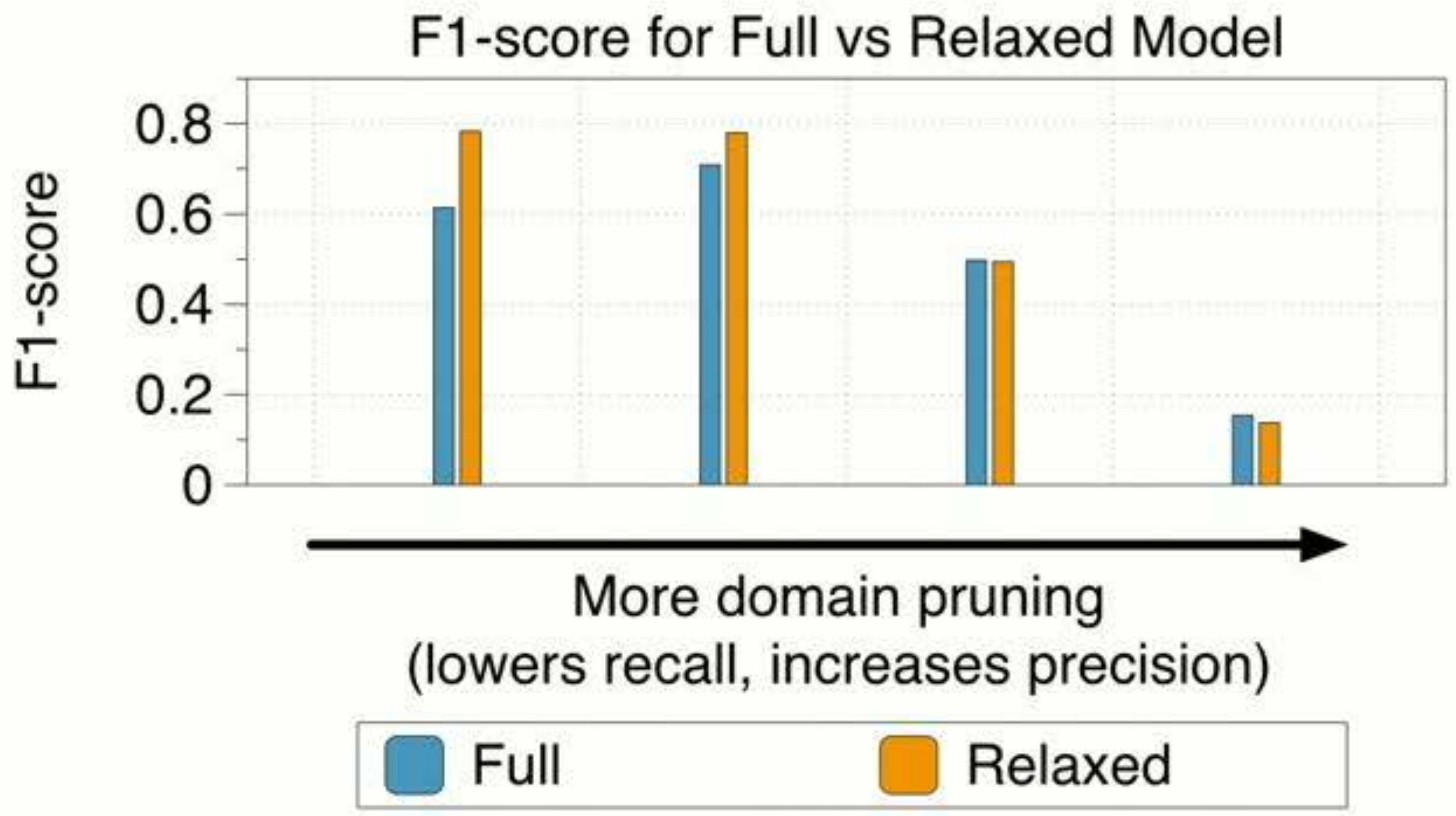
Relaxing constraints





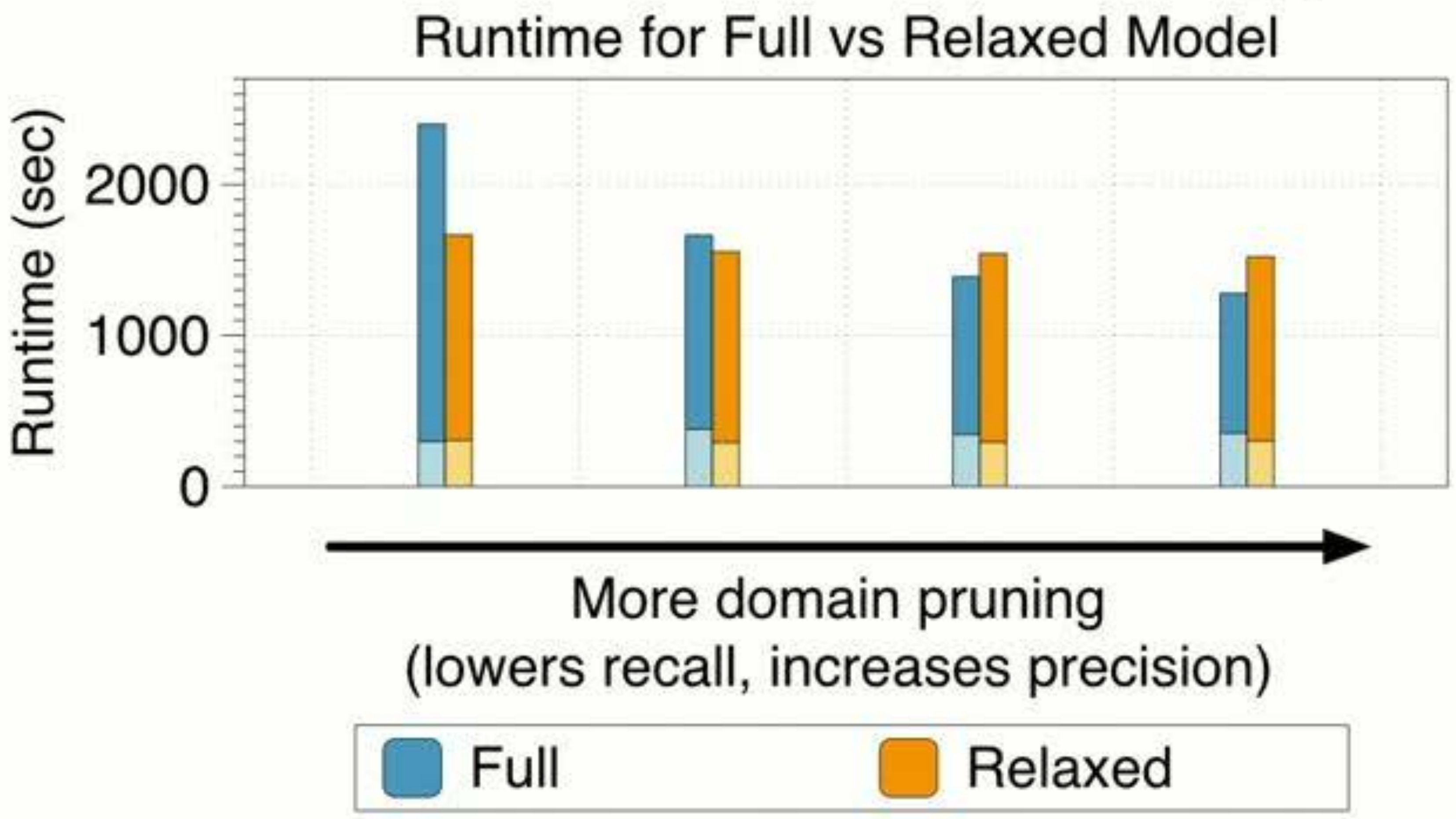
Relaxing constraints

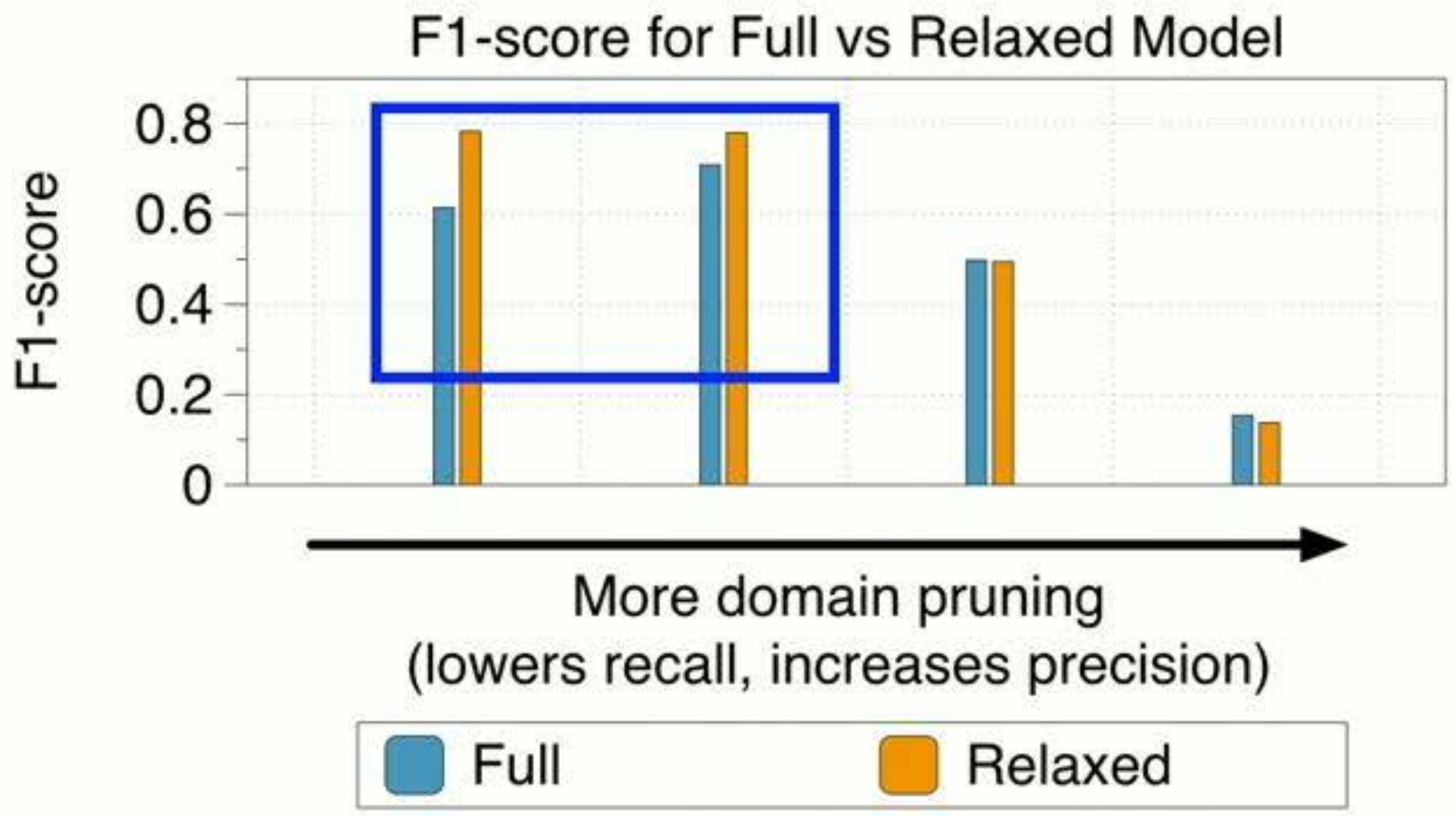




Faster compilation, learning, and inference when we prune the RV domain

Relaxing constraints





Increased robustness (more accurate repairs) when RV domain is ill-specified (no heavy pruning used)

HoloClean: Probabilistic Data Repairs

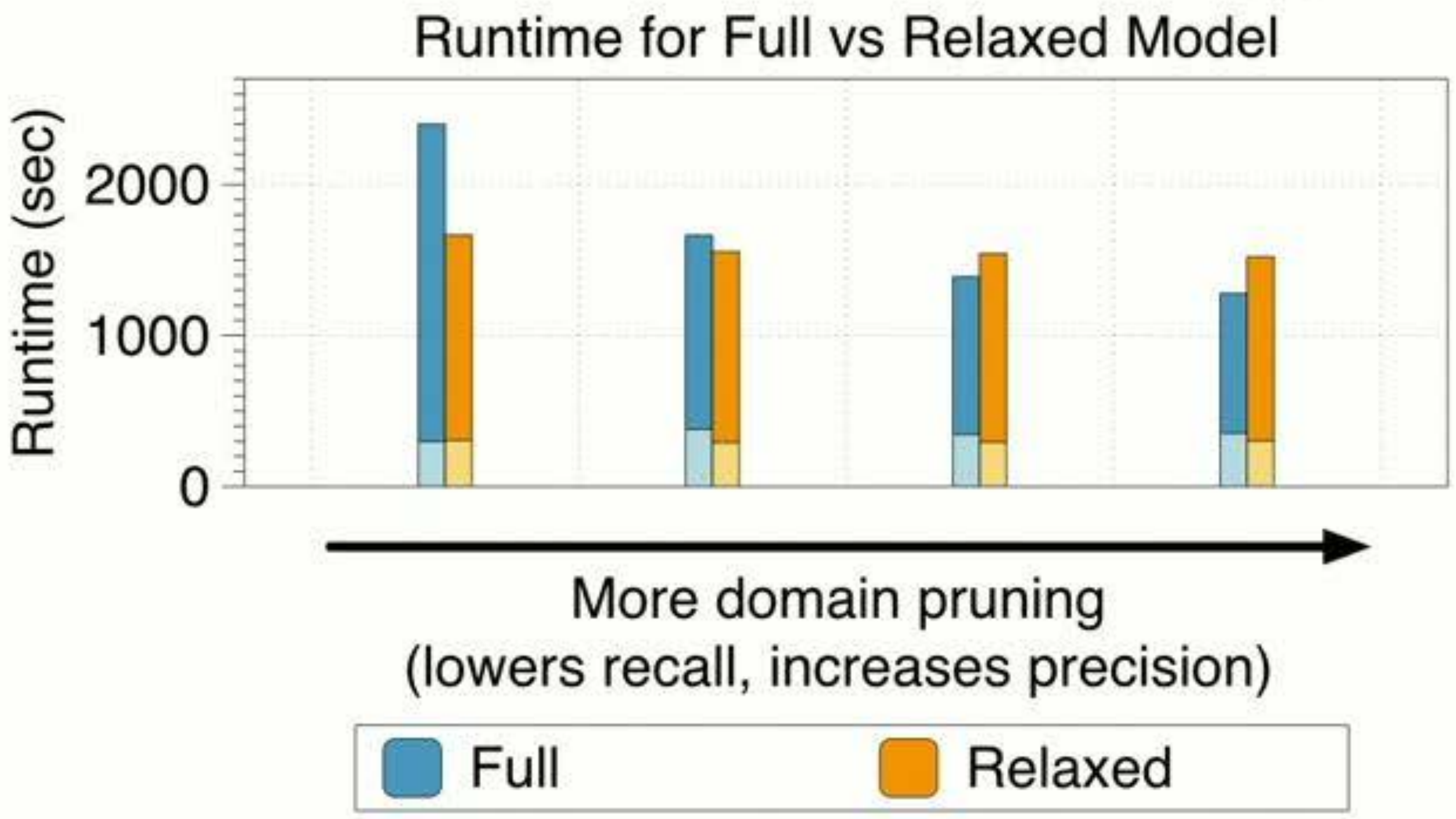
Challenge: Inference under constraints is #P-complete

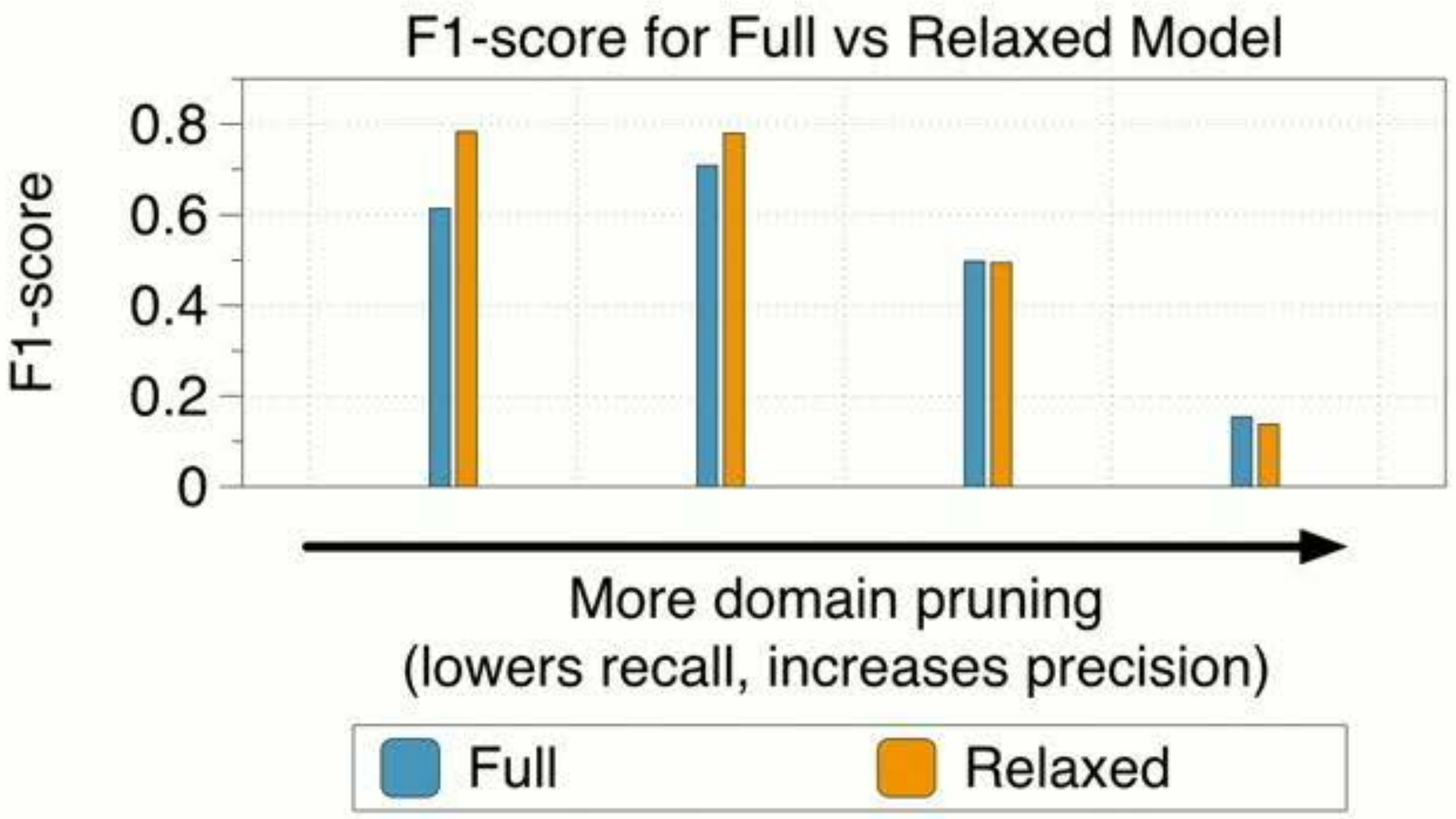
Applying probabilistic inference naively does not scale to data cleaning instances with millions of tuples

Idea 1: Prune domain of random variables.

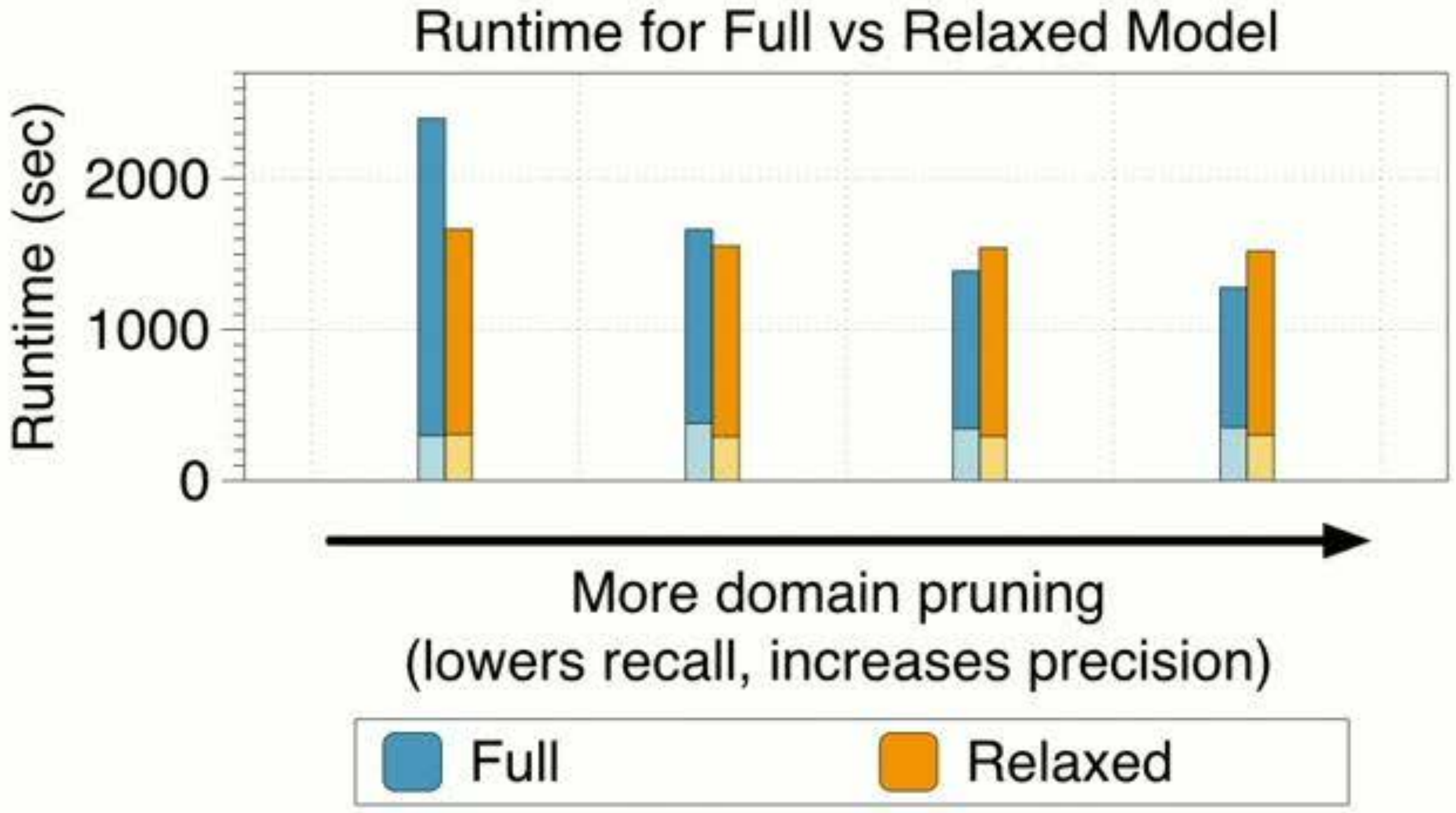
Idea 2: Relax constraints over sets of random variables to features over independent random variables.

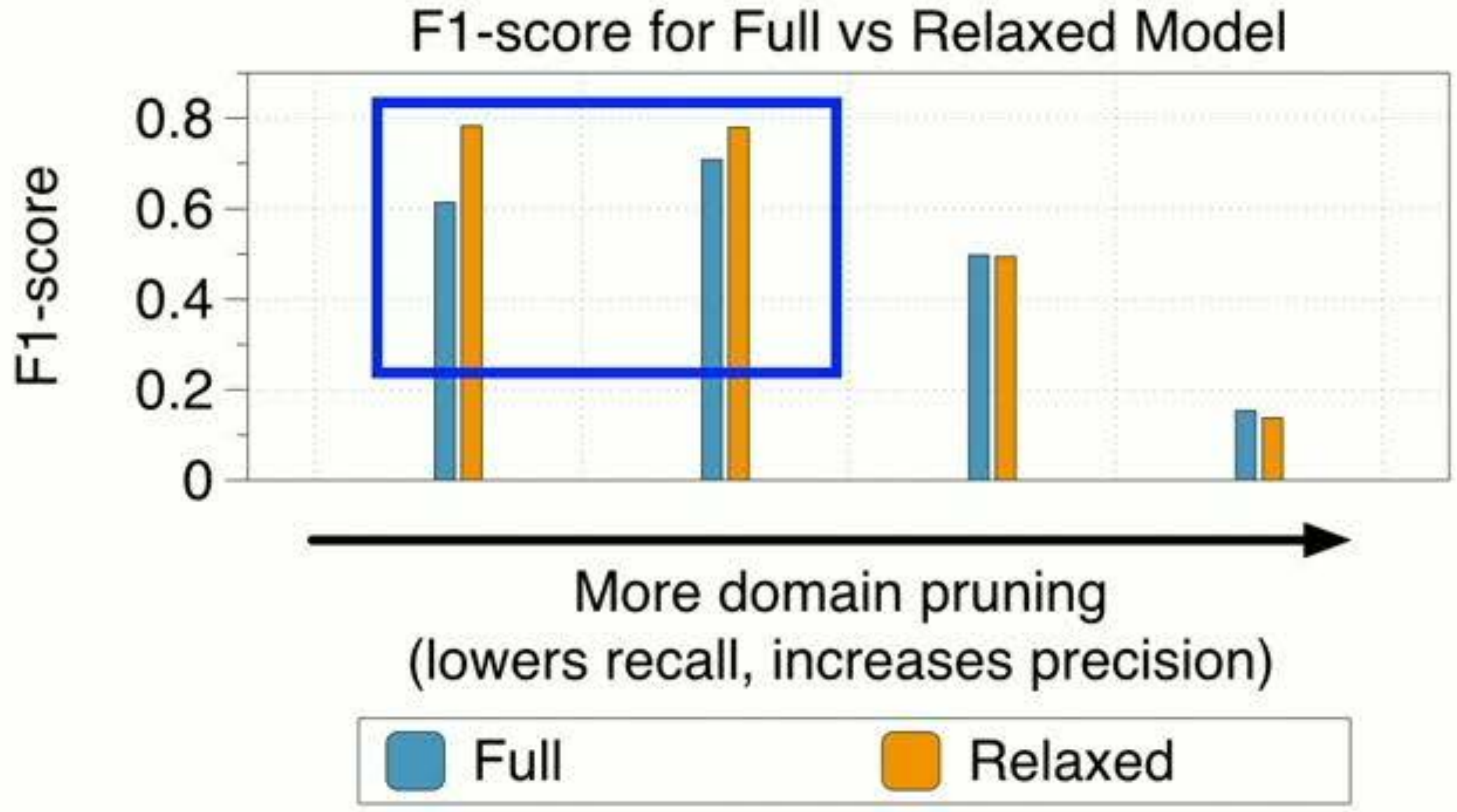
Relaxing constraints





Relaxing constraints



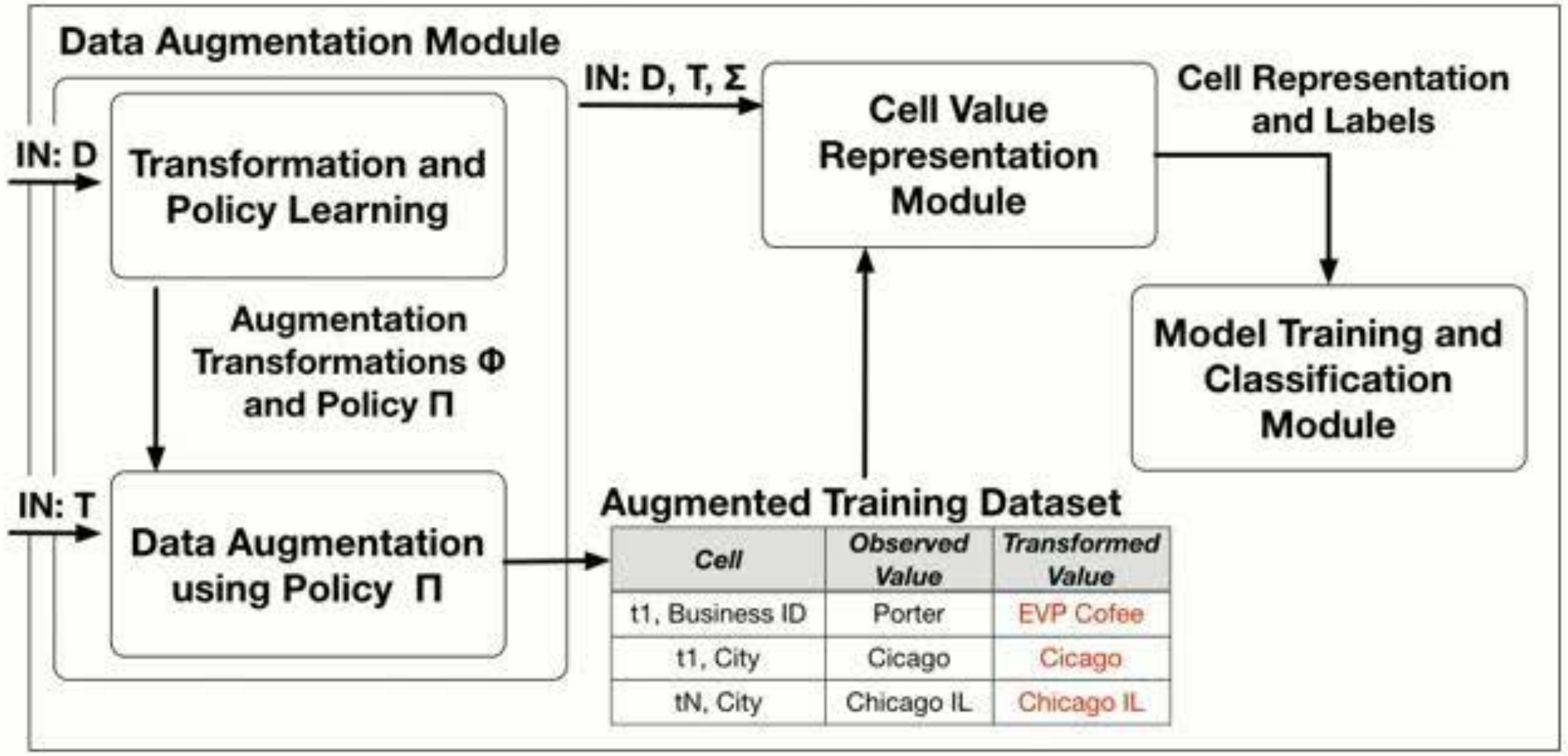


Increased robustness (more accurate repairs) when RV domain is ill-specified (no heavy pruning used)

Data Augmentation for Error Detection

HoloDetect learns a PUD realizer and uses the learned realizer to generate synthetic training data to teach a deep neural network how to detect erroneous values.

Error Detection with Data Augmentation



Reference: HoloDetect: A Few-Shot Learning Framework for Error Detection
Heidari, McGrath, Ilyas, Rekatsinas, SIGMOD 2019

Data Augmentation for Error Detection

Error Detection:

- ***Binary classification:*** for each cell decide if it's erroneous or correct.
- Severe imbalance, high heterogeneity.
- Assumption: Easy for human annotators to provide examples of correct tuples.

Challenge: How can we obtain labeled data while minimizing the input from human annotators?

Dataset D

	Business ID	*City*	*State*	*Zip Code*
t1	Porter	Chicago	IL	60612
t2	Graft	Chicago	IL	60614
t3	EVP Coffee	Cicago IL		60618
...				
tN	Dark Matter	Chicago	IL	60612

Denial Constraints Σ (optional)

Zip Code $\rightarrow$ City

Zip Code $\rightarrow$ State

Business ID $\rightarrow$ State

Business ID $\rightarrow$ City

Business ID $\rightarrow$ Zip Code

Training Dataset T

Cell	*Observed Value*	*Correct Value*
t1, Business ID	Porter	Porter
t1, City	Chicago	Chicago
t3, City	Cicago IL	Chicago
...		
tk, State	<NaN>	IL

Data Augmentation for Error Detection

Dataset D

	Business ID	City	State	Zip Code
t1	Porter	Chicago	IL	60612
t2	Graft	Chicago	IL	60614
t3	EVP Coffee	Cicago IL		60618
...				
tN	Dark Matter	Chicago	IL	60612

Training Dataset T

Cell	Observed Value	Correct Value
t1, Business ID	Porter	Porter
t1, City	Chicago	Chicago
t3, City	Cicago IL	Chicago
...		
tk, State	<NaN>	IL

Program Synthesis: Learn a program to introduce errods

- Add characters: $\varnothing \longmapsto [a-z]^+$
- Remove characters: $[a-z]^+ \longmapsto \varnothing$
- Exchange characters: $[a-z]^+ \longmapsto [a-z]^+$ (the left side and right side are different)

Augmented Training Dataset

Cell	Observed Value	Transformed Value
t1, Business ID	Porter	EVP Cofee
t1, City	Cicago	Cicago
tN, City	Chicago IL	Chicago IL

Approach: Analyze the input dataset and learn how errors are introduced (learn a noisy channel). Use the clean tuples as seeds and introduce artificial erroneous examples that obey the distribution of the noisy channel.

Data Augmentation for Error Detection

Top-10 Entries for Π('scip-inf-4') in Hospital

i	->	x: 0.159139658427
n	->	x: 0.154838586577
p	->	x: 0.081720365137
s	->	x: 0.064516077740
c	->	x: 0.064516077740
-	->	x: 0.047311790343
p	->	Ø: 0.043010718493
Ø	->	s: 0.043010718493
Ø	->	x: 0.038709646644
4	->	x: 0.038709646644

Top-10 Entries for Π('Female') in Adult

Ø	->	s: 0.105263054412
Female	->	Male: 0.084889560009
Fem	->	M: 0.064516065607
Ø	->	T: 0.054329318406
Ø	->	K: 0.054329318406
Ø	->	t: 0.044142571205
a	->	Ø: 0.033955824003
Ø	->	u: 0.030560241603
Ø	->	f: 0.030560241603
Ø	->	j: 0.030560241603

Top-10 Entries for Π('R') in Animal
This column can only take values R, O, and Empty

R	->	Empty: 0.477337556212
R	->	O: 0.380031693159
Ø	->	200: 0.037717907828
Ø	->	20: 0.028843105986
Ø	->	0: 0.027575277151
Ø	->	7: 0.024088747856
Ø	->	2: 0.001584786043
Ø	->	3: 0.001584786043
Ø	->	O: 0.001267828835
Ø	->	4: 0.001267828834

Data Augmentation for Error Detection

Dataset D

	Business ID	City	State	Zip Code
t1	Porter	Chicago	IL	60612
t2	Graft	Chicago	IL	60614
t3	EVP Coffee	Cicago IL		60618
…				
tN	Dark Matter	Chicago	IL	60612

Training Dataset T

Cell	Observed Value	Correct Value
t1, Business ID	Porter	Porter
t1, City	Chicago	Chicago
t3, City	Cicago IL	Chicago
…		
tk, State	<NaN>	IL

Program Synthesis: Learn a program to introduce errods

- Add characters: $\emptyset \longmapsto [a-z]^+$
- Remove characters: $[a-z]^+ \longmapsto \emptyset$
- Exchange characters: $[a-z]^+ \longmapsto [a-z]^+$ (the left side and right side are different)

Augmented Training Dataset

Cell	Observed Value	Transformed Value
t1, Business ID	Porter	EVP Cofee
t1, City	Cicago	Cicago
tN, City	Chicago IL	Chicago IL

Approach: Analyze the input dataset and learn how errors are introduced (learn a noisy channel). Use the clean tuples as seeds and introduce artificial erroneous examples that obey the distribution of the noisy channel.

Data Augmentation for Error Detection

Top-10 Entries for Π('scip-inf-4') in Hospital

i	->	x: 0.159139658427
n	->	x: 0.154838586577
p	->	x: 0.081720365137
s	->	x: 0.064516077740
c	->	x: 0.064516077740
-	->	x: 0.047311790343
p	->	Ø: 0.043010718493
Ø	->	s: 0.043010718493
Ø	->	x: 0.038709646644
4	->	x: 0.038709646644

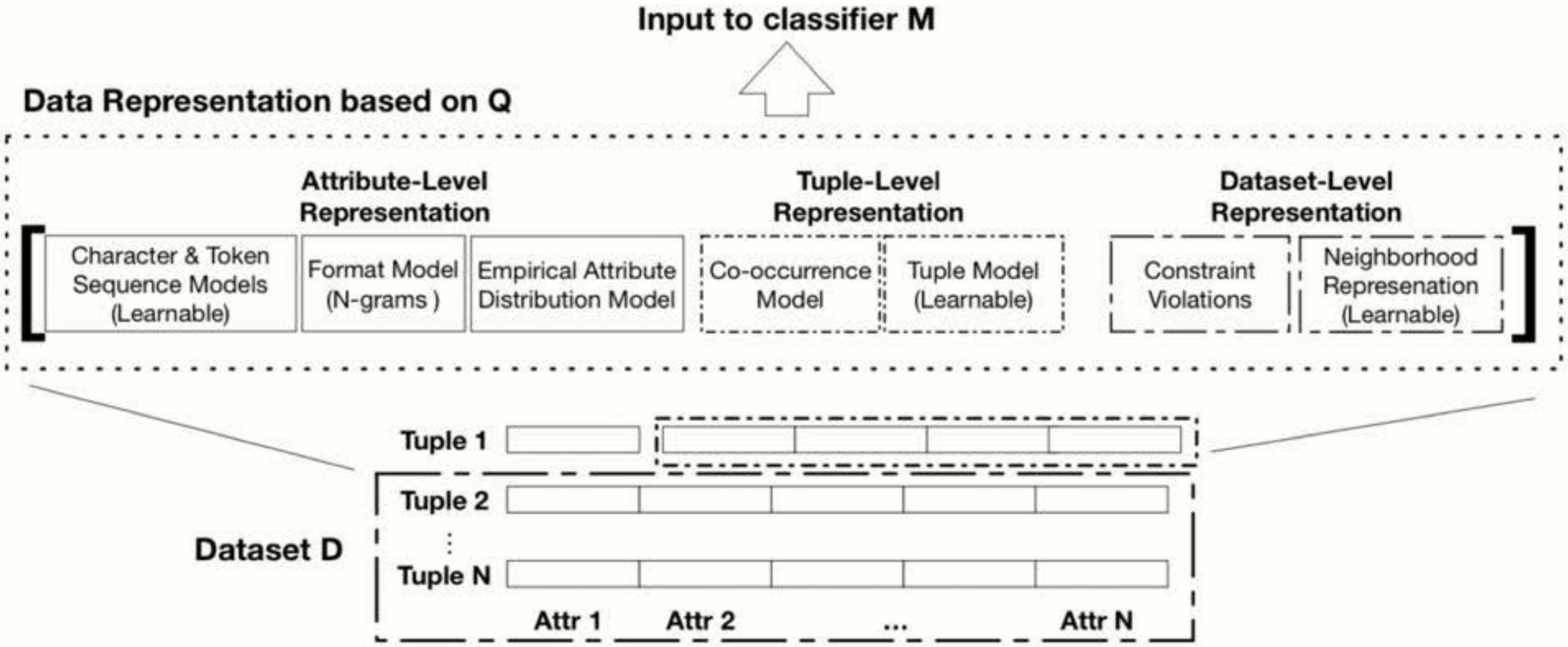
Top-10 Entries for Π('Female') in Adult

Ø	->	s: 0.105263054412
Female	->	Male: 0.084889560009
Fem	->	M: 0.064516065607
Ø	->	T: 0.054329318406
Ø	->	K: 0.054329318406
Ø	->	t: 0.044142571205
a	->	Ø: 0.033955824003
Ø	->	u: 0.030560241603
Ø	->	f: 0.030560241603
Ø	->	j: 0.030560241603

Top-10 Entries for Π('R') in Animal
This column can only take values R, O, and Empty

R	->	Empty: 0.477337556212
R	->	O: 0.380031693159
Ø	->	200: 0.037717907828
Ø	->	20: 0.028843105986
Ø	->	0: 0.027575277151
Ø	->	7: 0.024088747856
Ø	->	2: 0.001584786043
Ø	->	3: 0.001584786043
Ø	->	O: 0.001267828835
Ø	->	4: 0.001267828834

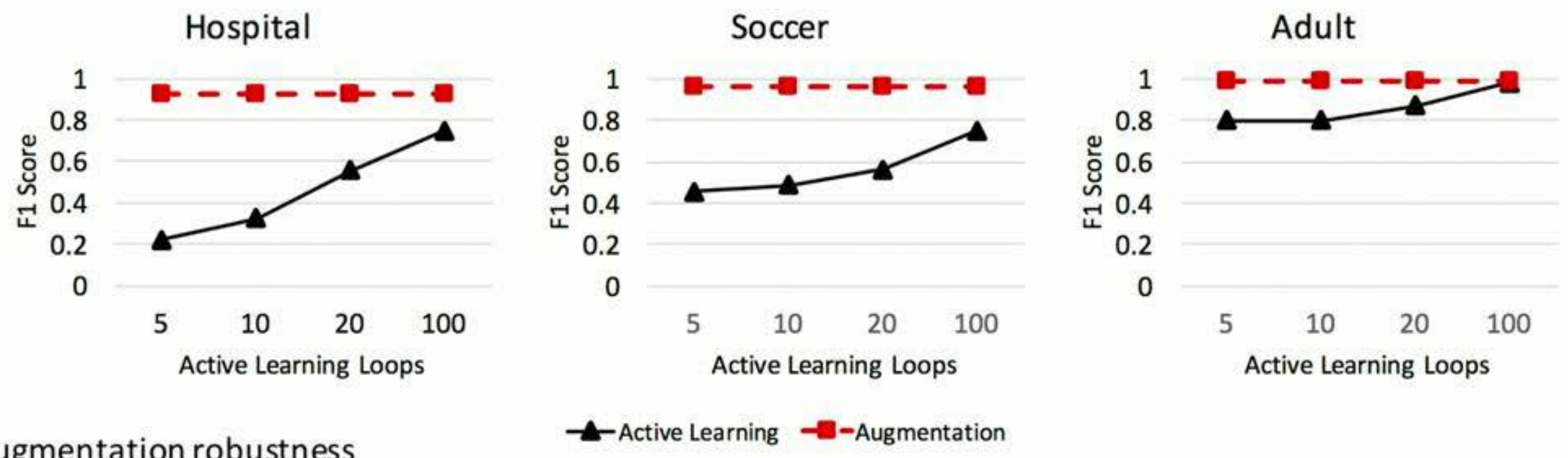
Data Augmentation for Error Detection

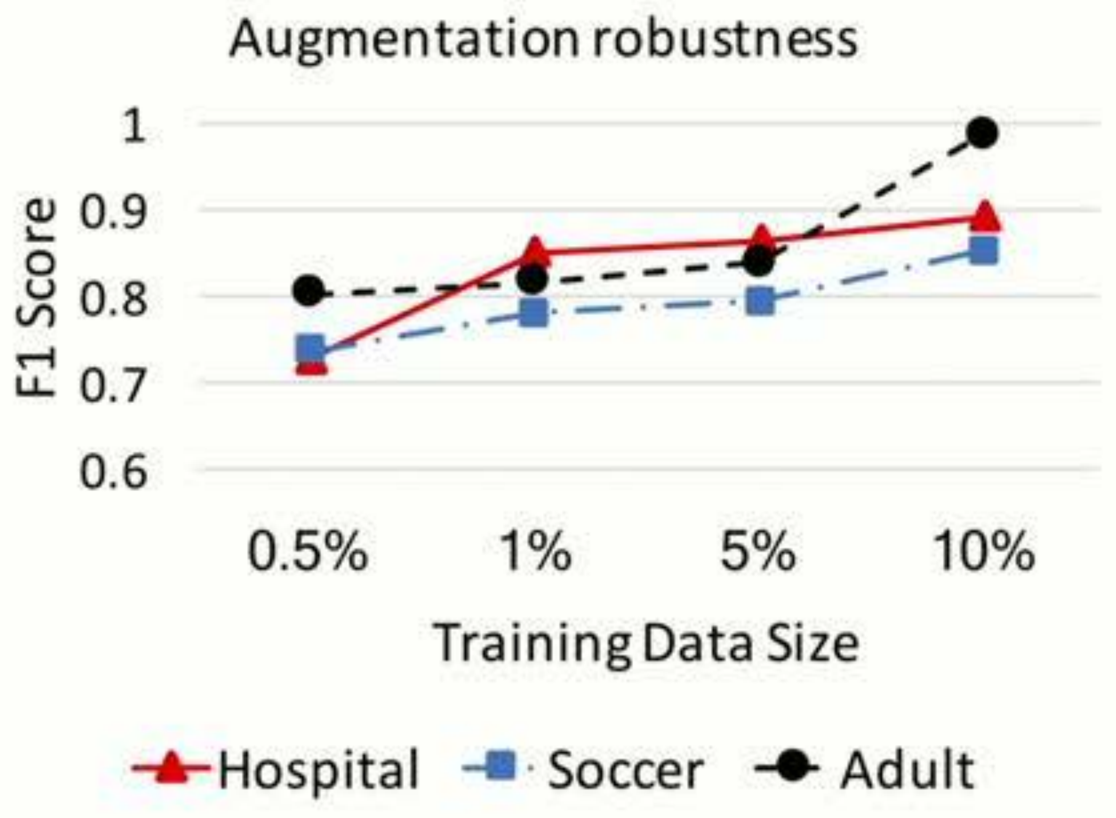


(A) Schematic Diagram of Representation Model Q

Approach: Train a classifier to identify errors in the input data set

Data Augmentation for Error Detection





HoloDetect requires fewer training examples than competing approaches

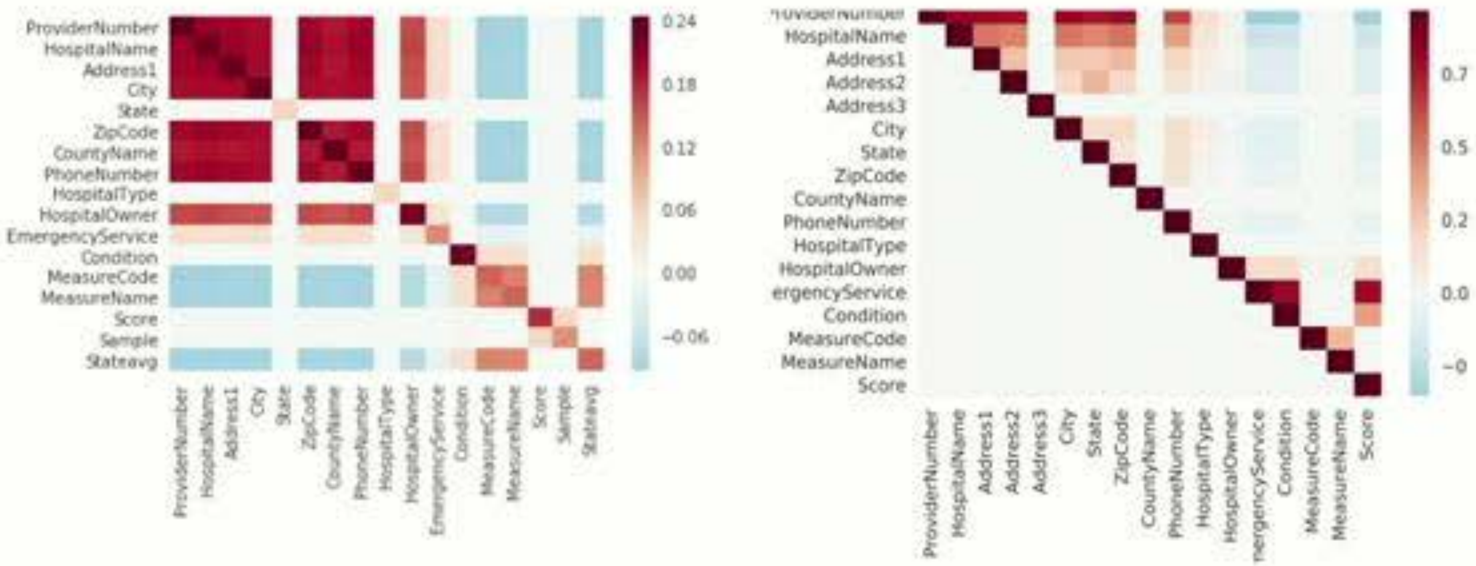
AutoFD: Functional Dependency Discovery via Structure Learning

Input Noisy Dataset

DBAName	Address	City	State	Zip Code
Foodlife	835 N Michigan Av	Chicago	IL	60608
Mity Nice Bar	835 N Michigan Av	Chicago	IL	60611
Harry Caray's	835 N Michigan	Chicago	IL	60611
Graft	3435 W Washington	Cicago	IL	60612
Pierrot	3493 Washington		IL	60612

Structure Learning

- Estimate the inverse covariance matrix of lifted model.
- Fit a linear model by decomposing the estimated inverse covariance

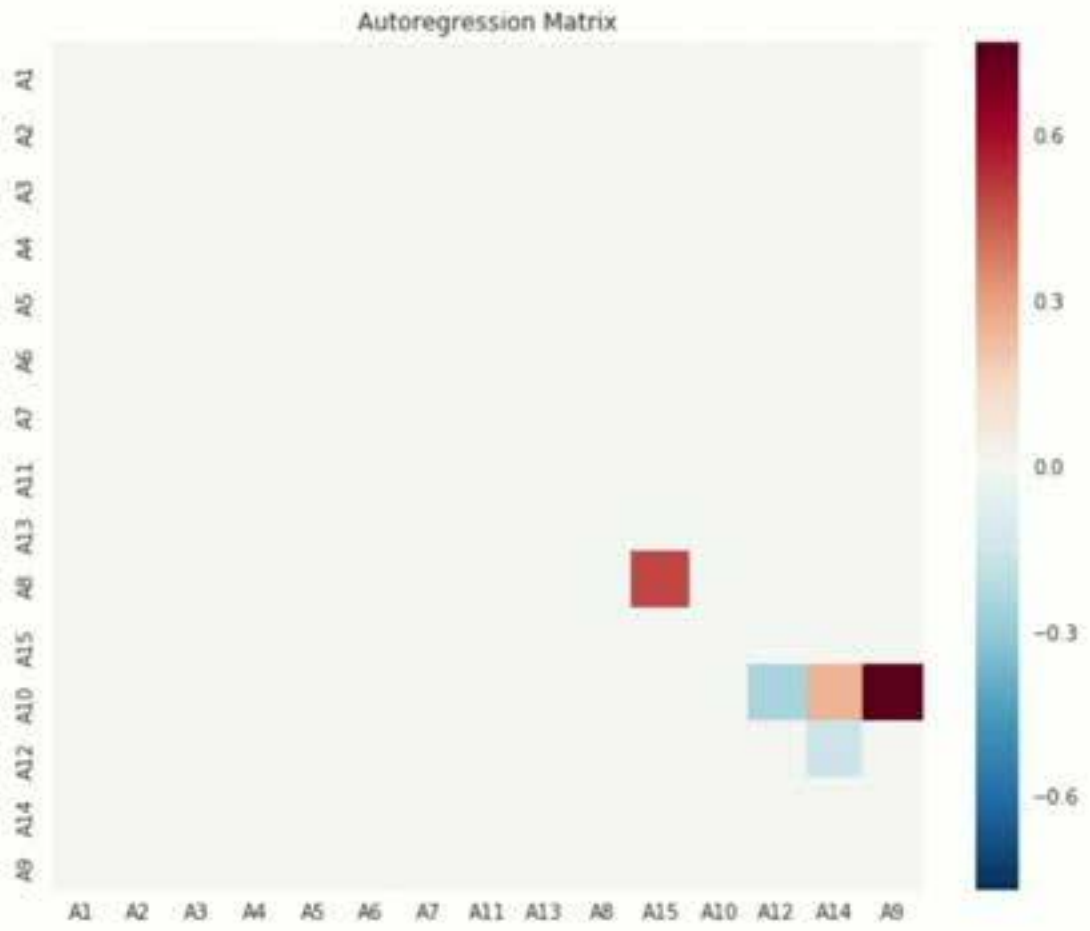


FD discovery as a structure learning problem over a linear structured model

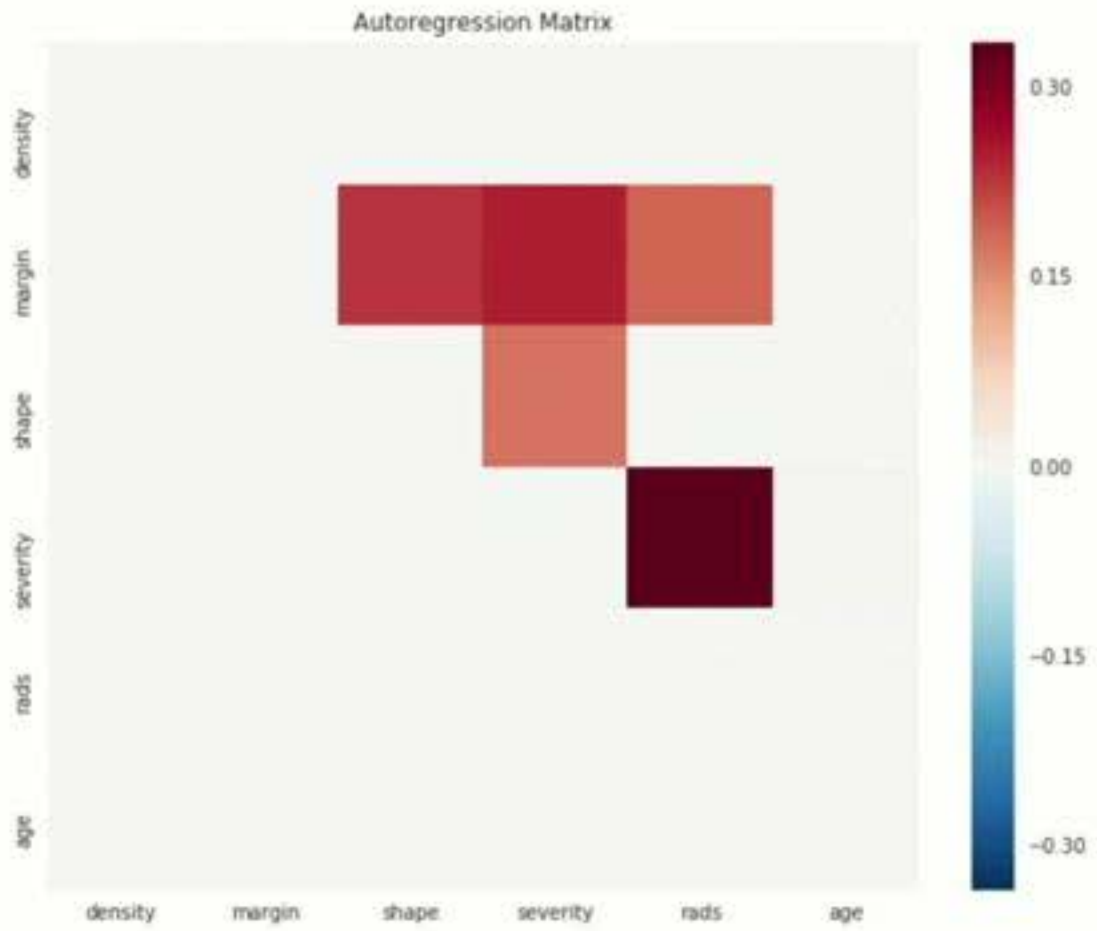
Lifted-variation of structure learning using sparse regression (L1-regularization).
2x F1 improvement over state-of-the-art (included non-lifted structure learning methods).

Guarantees on FD discovery under a weak Realizer (bounded noise).

AutoFD provides insights for downstream data preparation tasks



(A) Australian Credit Approval;
A15 is the goal attribute



(B) Mammography;
Severity is the goal attribute

Effective feature engineering

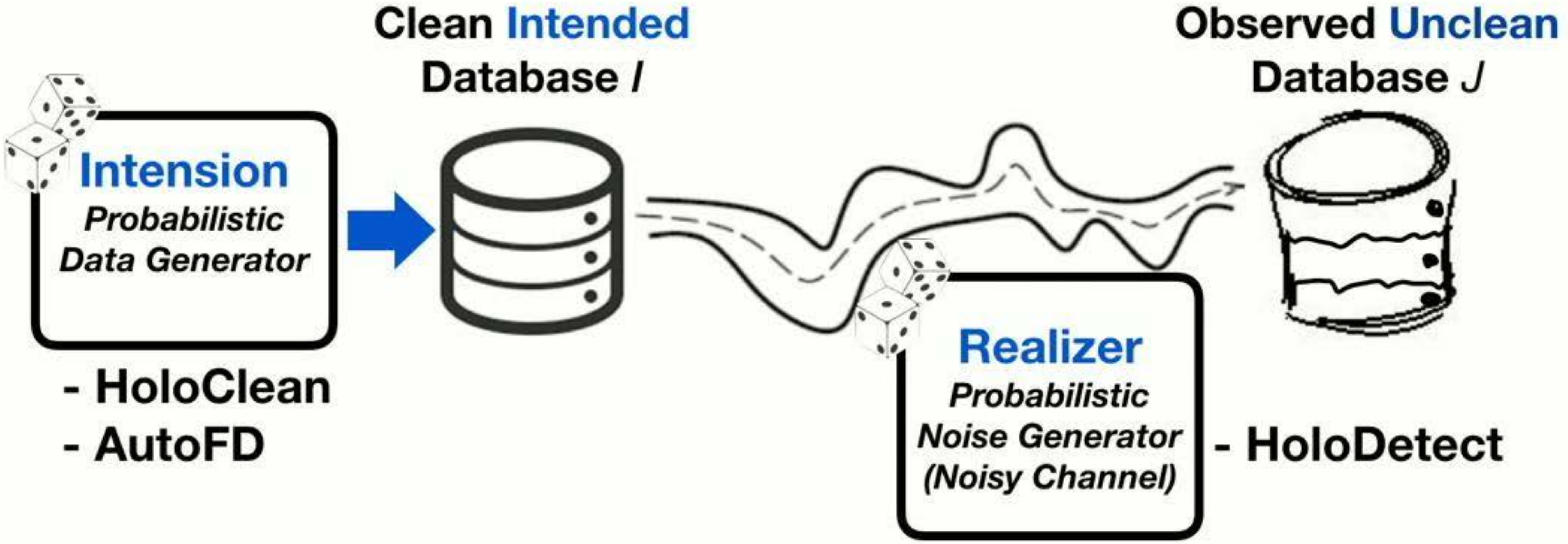
AutoFD provides insights for downstream data preparation tasks

Data set	Random Noise				Systematic Noise			
	SystemX		XGBoost		SystemX		XGBoost	
	w/o	w	w/o	w	w/o	w	w/o	w
Australian	0.41	**0.86**	0.34	**0.86**	0.42	**0.96**	0.34	**0.96**
Hospital	0.58	**1.0**	0.57	**0.97**	0.38	**1.0**	0.53	**0.99**
Mammogr.	0.63	**0.84**	0.54	**0.73**	0.44	**0.73**	0.42	**0.68**
NYPD	0.89	**0.93**	0.92	**0.94**	0.75	**0.76**	0.86	**0.90**
Thoracic	0.77	**0.82**	0.76	**0.83**	0.74	**0.91**	0.61	**0.91**
Tic-Tac-Toe	**0.6**	0.56	0.52	**0.55**	**0.48**	0.47	**0.57**	0.50

Ex: increased imputation accuracy for attributes w. dependencies (in the output of AutoFD)

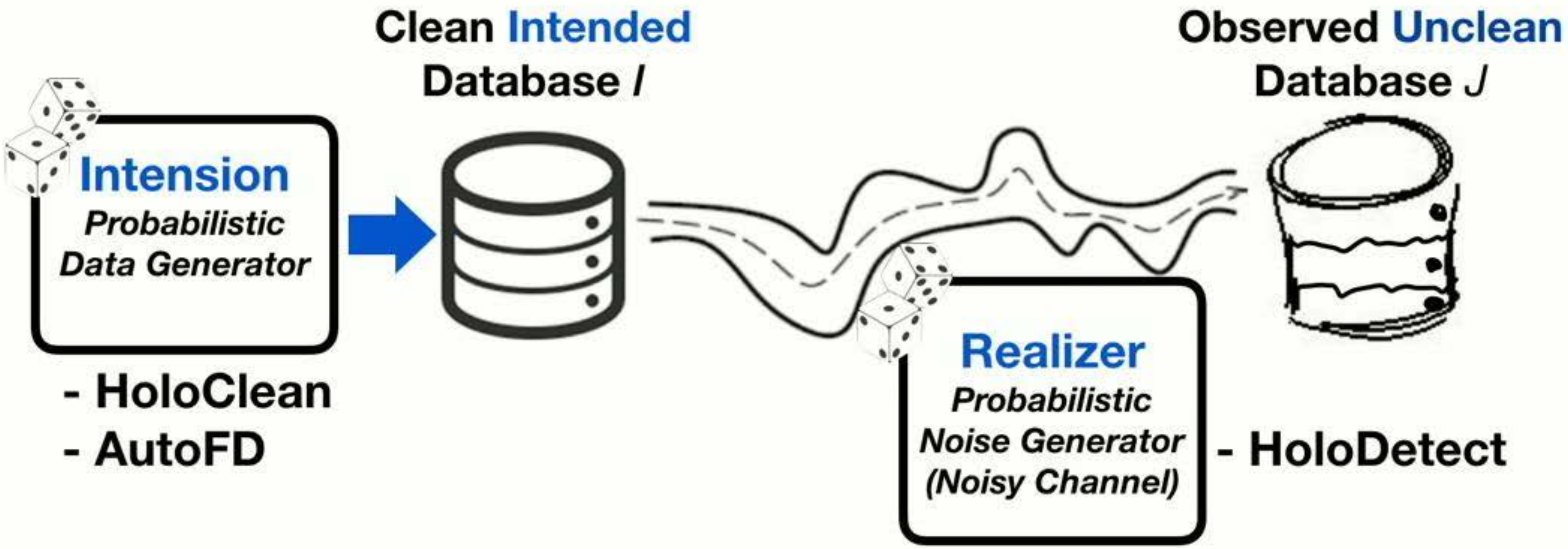
Provides insights on the effectiveness of automated data cleaning.

The Probabilistic Unclean Database Model



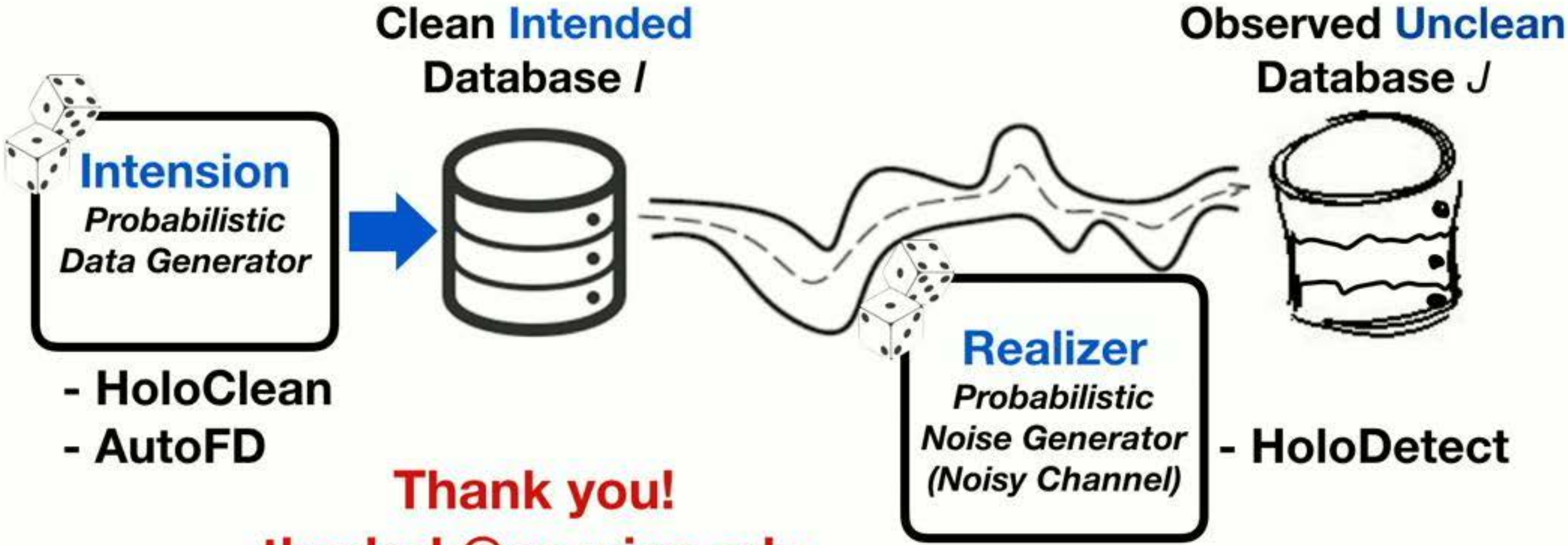
A formal noisy channel model that leads to new insights for managing noisy data and has immediate practical applications to data cleaning systems.

The Probabilistic Unclean Database Model



A formal noisy channel model that leads to new insights for managing noisy data and has immediate practical applications to data cleaning systems and exciting connections to robust ML.

The Probabilistic Unclean Database Model



Thank you!

thodrek@cs.wisc.edu

A formal noisy channel model that leads to new insights for managing noisy data and has immediate practical applications to data cleaning systems and exciting connections to robust ML.