CS839: Probabilistic Graphical Models

Lecture 23: Applications in Data Management

Theo Rekatsinas
Logistics

1. Project presentations next Tuesday

2. 10 Groups: 10 - 15 mins presentation per group (We will run late)

3. Things to cover:
   • What is the problem?
   • Why is it interesting and important?
   • Why is it hard? What are the baselines (E.g., why do naive approaches fail?)
   • Why hasn't it been solved before? (Or, what's wrong with previous proposed solutions? How does yours differ?)
   • What are the key components of your approach and results?
Snorkel + Data Programming
MOTIVATION:

In practice, training data is often:

• *The bottleneck*

• *The practical injection point for domain knowledge*
We can use higher-level, weaker supervision to program ML models.
Outline

• The Labeling Bottleneck: *The new pain point of ML*

• Data Programming + Snorkel: *A framework for weaker, more efficient supervision*

• In practice: *Empirical results & user studies*
The ML Pipeline Pre-Deep Learning

Collection → Labeling → Feature Engineering → Training

Feature engineering *used to be* the bottleneck…
The ML Pipeline Today

Collection → Labeling → Representation Learning → Training

New pain point, new injection point
Training Data: Challenges & Opportunities

• Expensive & Slow:
  • Especially when domain expertise needed

• Static:
  • Real-world problems change; hand-labeled training data does not.

• An opportunity to inject domain knowledge:
  • Modern ML models are often too complex for hand-tuned structures, priors, etc.

How do we get—and use—training data more effectively?
Data Programming + Snorkel

A Framework + System for Creating Training Data with Weak Supervision
KEY IDEA:

Get users to provide *higher-level (but noisier)* supervision,

Then model & de-noise it (using *unlabeled* data) to train *high-quality* models
Data Programming Pipeline in Snorkel

**Input:** Labeling Functions, Unlabeled data

1. Users write *labeling functions* to generate noisy labels
2. We model the labeling functions’ behavior to de-noise them
3. We use the resulting prob. labels to train a model

**Generative Model**

\[ \lambda_1 \quad \lambda_2 \quad \lambda_3 \]

**Noise-Aware Discriminative Model**

\[ h_{1,1} \quad h_{1,2} \quad h_{1,3} \]

\[ y_1 \]

\[ x_{1,1} \quad x_{1,2} \]

**Ex. Application:** Knowledge Base Creation (KBC)
Surprising Point:

No hand-labeled training data!
Step 1: Writing Labeling Functions

A Unifying Framework for Expressing Weak Supervision

```python
def lf1(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0

def lf2(x):
    m = re.search(r'.*cause.*', x.between)
    return 1 if m else 0

def lf3(x):
    m = re.search(r'.*not cause.*', x.between)
    return 1 if m else 0
```
Example: Chemical-Disease Relation Extraction from Text

**TITLE:**
Myasthenia gravis presenting as weakness after magnesium administration.

**ABSTRACT:**
We studied a patient with no prior history of neuromuscular disease who became virtually quadriplegic after parenteral magnesium administration for preeclampsia. The serum magnesium concentration was 3.0 mEq/L, which is usually well tolerated. The magnesium was stopped and she recovered over a few days. While she was weak, 2-Hz repetitive stimulation revealed a decrement without significant facilitation at rapid rates or after exercise, suggesting postsynaptic neuromuscular blockade. After her strength returned, repetitive stimulation was normal, but single fiber EMG revealed increased jitter and blocking. Her acetylcholine receptor antibody level was markedly elevated. Although paralysis after magnesium administration has been described in patients with known myasthenia gravis, it has not previously been reported to be the initial or only manifestation of the disease. Patients who are unusually sensitive to the neuromuscular effects of magnesium should be suspected of having an underlying disorder of neuromuscular transmission.

- We define candidate entity mentions:
  - **Chemicals**
  - **Diseases**
- Goal: Populate a relational schema with relation mentions

<table>
<thead>
<tr>
<th>ID</th>
<th>Chemical</th>
<th>Disease</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>magnesium</td>
<td>Myasthenia gravis</td>
<td>0.84</td>
</tr>
<tr>
<td>01</td>
<td>magnesium</td>
<td>quadriplegic</td>
<td>0.73</td>
</tr>
<tr>
<td>02</td>
<td>magnesium</td>
<td>paralysis</td>
<td>0.96</td>
</tr>
</tbody>
</table>
Labeling Functions

- Traditional “distant supervision” rule relying on external KB

"Chemical A is found to cause disease B under certain conditions..."

```
def lf1(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0
```

Label = TRUE

This is likely to be true... but
Labeling Functions

- Traditional “distant supervision” rule relying on external KB

"Chemical A was found on the floor near a person with disease B..."

```python
def lf1(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0
```

Label = TRUE

...can be false!

We will learn the accuracy of each LF (next)
Writing Labeling Functions in Snorkel

• Labeling functions take in **Candidate** objects:

• Three levels of abstraction for writing LFs in Snorkel:

  • Python code

  ```python
def lf1(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0
```

  ```python
lf1 = LF_DS(KB)
```

  • LF templates

  ```python
for lf in LF_DS_hier(KB, cut_level=2):
    yield lf
```

  • LF generators

  ```python
A knowledge base (KB) with hierarchy
```

**Context Hierarchy**

Key Point: **Supervision as code**
Supported by Simple Jupyter Interface

---

**Applying Labeling Functions**

First we construct a `LabelManager`.

```python
from snorkel.annotations import LabelManager

label_manager = LabelManager()
```

Next we run the `LabelManager` to apply the labeling functions to the training `CandidateSet`. We'll start with some of our labeling functions:

```python
In [ ]:

spouses = ['wife', 'husband', 'ex-wife', 'ex-husband']
family = ['father', 'mother', 'sister', 'brother', 'son', 'daughter', 'grandfather', 'grandmother', 'uncle', 'aunt', 'cousin']

other = ['boyfriend', 'girlfriend', 'boss', 'employee', 'secretary', 'co-worker']

def LF_two_far_apart(c):
    return -1 if len(get_between_tokens(c)) > 10 else 0

def LF_third_wheel(c):
    return -1 if 'PERSON' in get_between_tokens(c, attrib='ner_tags', case_sensitive=True) else 0

def LF_husband_wife(c):
    return 1 if len(spouses.intersection(set(get_between_tokens(c))))) > 0 else 0
```
Broader Perspective:

A Template for Weak Supervision
A Unifying Method for Weak Supervision

- Distant supervision
- Crowdsourcing
- Weak classifiers
- Domain heuristics / rules

\[ \lambda : X \mapsto Y \cup \{\emptyset\} \]
How to handle such a diversity of weak supervision sources?
Step 2: Modeling Weak Supervision
Weak Supervision: Core Challenges

- Unified input format

- Modeling
  - Accuracies of sources
  - Correlations between sources
  - Expertise of sources

- Using to train a wide range of models
Weak Supervision: Core Challenges

• Unified input format

• Modeling
  • Accuracies of sources
  • Correlations between sources
  • Expertise of sources

• Using to train a wide range of models

Intuition: We use agreements / disagreements to learn without ground truth
Basic Generative Labeling Model

Labeling propensity:
\[ \beta_j = p_\theta(\Lambda_{i,j} \neq \emptyset) \]
\[ f_{j}^{\text{lab}}(\Lambda_i, Y_i) = \exp(\theta_{j}^{\text{lab}}\Lambda_{i,j}) \]

Accuracy:
\[ \alpha_j = p_\theta(\Lambda_{i,j} = Y_i \mid Y_i, \Lambda_{i,j} \neq \emptyset) \]
\[ f_{j}^{\text{acc}}(\Lambda_i, Y_i) = \exp(\theta_{j}^{\text{acc}}\Lambda_{i,j}Y_i) \]

Correlations

ICML 2017
Intuition: Learning from Disagreements

Learn the model $\pi = P(y, \Lambda)$ using MLE
- LFs have a hidden accuracy parameter
- Intuition: Majority vote--estimate labeling function accuracy based on overlaps / conflicts
  - Similar to crowdsourcing but different scaling.
  - small number of LFs, large number of labels each

Produce a set of noisy training labels
$$\mu_\pi(y, \lambda) = P_{(y,\Lambda)\sim\pi}(y \mid \Lambda = \lambda(x))$$
Step 2: Training a Noise-Aware Model

In a supervised learning setting, we would learn from ground-truth labels:

$$\hat{w} = \arg\min_w \frac{1}{N} \sum_{i=1}^{N} l(w, x^{(i)}, y^{(i)})$$

Here, we learn from the noisy labels:

$$\hat{w} = \arg\min_w \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{(y, \mathcal{A}) \sim \pi}[l(w, x^{(i)}, y^{(i)} = y)]$$

$T = \{(x_1, 0), (x_2, 1), (x_3, 0), ...\}$

$T = \{(x_1, 0.1), (x_2, 0.6), (x_3, 0.3), ...\}$

Only requires simple tweak to loss function works over many models including Logistic Regression, SVMs and LSTMs.
Theory: Scaling with *Unlabeled* Data

• We show that with:

  • $O(1)$ labeling functions of sufficient quality / expressiveness
  
  • $\tilde{O}(\epsilon^{-2})$ *unlabeled* training data points

  • $\rightarrow$ We get $O(\epsilon)$ generalization risk

This is the same asymptotic scaling as in supervised methods!
When is modeling the noise worthwhile?

- Can look at label density:
  - Low: Too sparse to beat MV
  - High: MV approaches optimal
  - Medium: Just right!

- Can use conditional decision rule to safely skip gen. modeling stage
  - E.g. during early LF dev cycles
Users write labeling functions to generate noisy labels

We model the labeling functions' behavior to de-noise them

We use the resulting prob. labels to train a model

Putting it All Back Together

Input: Labeling Functions, Unlabeled data

Generative Model

Noise-Aware Discriminative Model

Output: Probabilistic Training Labels

Def l1f(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0

Def l2f(x):
    m = re.search(r'.*cause.*', x.between)
    return 1 if m else 0

Def l3f(x):
    m = re.search(r'.*not cause.*', x.between)
    return 1 if m else 0

Users write labeling functions to generate noisy labels

We model the labeling functions' behavior to de-noise them

We use the resulting prob. labels to train a model

Def l1f(x):
    cid = (x.chemical_id, x.disease_id)
    return 1 if cid in KB else 0

Def l2f(x):
    m = re.search(r'.*cause.*', x.between)
    return 1 if m else 0

Def l3f(x):
    m = re.search(r'.*not cause.*', x.between)
    return 1 if m else 0
How well does this work in practice?

Empirical Results
Results on Chemical-Disease Relations

- Distant Supervision: Precision: 25.5, Recall: 34.8, F1: 29.4
- Generative Model: Precision: 52.3, Recall: 30.4, F1: 38.5 (+ 9.1)
- Discriminative Model: Precision: 38.8, Recall: 54.3, F1: 45.3 (+ 6.8)
- Hand Supervision: Precision: 39.9, Recall: 58.1, F1: 47.3 (+ 2.0)
How easy is this to use in practice?

User Study
Snorkel User Study

We recently ran a Snorkel biomedical workshop in collaboration with the NIH Mobilize Center.

15 companies and research groups attended.

How well did these new Snorkel users do?

71% New Snorkel users matched or beat 7 hours of hand-labeling.

2.8x Faster than hand-labeling data.

45.5% Average improvement in model performance.

3rd Place Score
No machine learning experience
Beginner-level Python
Conclusion

• Snorkel provides a unifying framework for combining and modeling weak supervision
  
  • Allows us to rapidly generate training data for modern ML models
  
  • Labeling functions: supervision as code
  
• For more check out snorkel.stanford.edu: Code, tutorials, blogs, papers
Fonduer: Knowledge Base Construction from Richly Formatted Data
Knowledge bases are everywhere…

But, troves of "richly formatted" information remains untapped
NPN Silicon Switching Transistors

- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

### Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td>mA</td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td>mA</td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation $T_S \leq 71^\circ C$</td>
<td>$P_{tot}$</td>
<td>330</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_J$</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65 ... 150</td>
<td>°C</td>
</tr>
</tbody>
</table>
NPN Silicon Switching Transistors

- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

<table>
<thead>
<tr>
<th>Maximum Ratings</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{tot}$</td>
<td>330</td>
<td>mV</td>
</tr>
<tr>
<td>$T_S \leq 71^\circ C$</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_j$</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65 ... 150</td>
<td></td>
</tr>
</tbody>
</table>

Knowledge Base

- **SMBT3904**
  - Current: 200 mA
- **MMBT3904**
  - Current: 200 mA

Goal: extract maximum collector current from transistor datasheets

Transistor Datasheet

Knowledge base construction from richly formatted data
Knowledge base construction from richly formatted data

Transistor Datasheet

SMBT3904...MMBT3904

NPN Silicon Switching Transistors
- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>VCEO</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>VCB</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>VEB</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>IC</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>Ptot</td>
<td>S330</td>
<td>mV</td>
</tr>
<tr>
<td>Junction temperature</td>
<td>Tj</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>Tstg</td>
<td>65...150</td>
<td>°C</td>
</tr>
</tbody>
</table>

In richly formatted data, semantics are expressed in **textual, structural, tabular, and visual** modalities throughout a document.

**Conventional approach 1:** Filter out other modalities besides unstructured text

**Conventional approach 2:** Limit the context scope to sentences or tables.

**Problem:** Misses important relations if you neglect multimodal information

**Up to 97% missed relations!**
Deep learning is very successful in many domains

Software 2.0

I sometimes see people refer to neural networks as just “another tool in your machine learning toolkit.” Sure, there are many other tools there, and some more. Unfortunately, neural networks of a fundamentally different kind.

Alibaba’s artificial intelligence bot beats humans at reading in a first 100 questions.

A deep neural network model developed by Alibaba has scored higher than humans in a reading comprehension test, paving the way for bots to replace people in customer service jobs.
Can we take advantage of this powerful tool and apply it to our problem?
Keys to utilizing deep learning

How do we gather enough labeled, richly formatted data?

How do we model the characteristics of richly formatted data in DL?
Fonduer

A weakly supervised deep learning framework for knowledge base construction from richly formatted data
Fonduer in practice!

- Anti-Human Trafficking
- Search Engine
- Genome-wide Association Studies
- Internet of Things
- Paleontology
Fonduer pipeline

Data Input → Preprocess data → Candidate Generation → Generate Training data → Featurization & Classification → Output

<table>
<thead>
<tr>
<th>Transistor Part</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMBT3904</td>
<td>200mA</td>
</tr>
<tr>
<td>MMBT3904</td>
<td>200mA</td>
</tr>
</tbody>
</table>

NPN Silicon Switching Transistors
- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td>V</td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td>V</td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{TOT}$</td>
<td>≤71°C</td>
<td>°C</td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_j$</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65</td>
<td>°C</td>
</tr>
</tbody>
</table>

Candidate Generation

Generate Training data
Generating richly formatted training data
### Transistor Datasheet

**NPN Silicon Switching Transistors**
- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

### Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{tot}$</td>
<td>330</td>
<td>mV</td>
</tr>
<tr>
<td>$T_S \leq 71^\circ C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_S \leq 115^\circ C$</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_J$</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65 ... 150</td>
<td></td>
</tr>
</tbody>
</table>

**Weak supervision**: express any supervision signal via labeling functions to generate training data

```python
# Check if current is in the same row with keyword 'collector'
def in_the_same_row_with(candidate):
    if 'collector' in row_ngrams(candidate.current):
        return 1
    else:
        return -1
```
Modeling Weak Supervision

<table>
<thead>
<tr>
<th>Doc. level Candidates</th>
<th>Multimodal Supervision</th>
<th>Output: Probabilistic Training Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMBT3904 100</td>
<td>Vertically aligned with ‘Value’</td>
<td>SMBT3094 100 0.5</td>
</tr>
<tr>
<td>SMBT3904 200</td>
<td>Row ngrams contain ‘mA’</td>
<td>SMBT3094 200 0.85</td>
</tr>
<tr>
<td>SMBT3904 150</td>
<td>‘current’ in sentence</td>
<td>SMBT3094 150 0.15</td>
</tr>
</tbody>
</table>

Ø = Abstain

Intuition: Use agreements / disagreements to learn the accuracy of LFs without ground truth

Output: Probabilistic Training Labels

Data programming/MeTal
Multimodal supervision is key to quality

For transistor datasheets...

Different supervision resources’ effect

<table>
<thead>
<tr>
<th>Modality</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual</td>
<td>0.1</td>
</tr>
<tr>
<td>All modalities</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Users intuitively rely on multimodal information for supervision
Featurization and Classification for Richly Formatted Data
Transistor Datasheet

NPN Silicon Switching Transistors
- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

<table>
<thead>
<tr>
<th>Maximum Ratings</th>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td></td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td></td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td></td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{tot}$</td>
<td></td>
<td>330</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td>$T_S \leq 71^\circ C$</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$T_S \leq 115^\circ C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_j$</td>
<td></td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td></td>
<td>-65 ... 150</td>
<td></td>
</tr>
</tbody>
</table>

LSTM excels at relation extraction from text
Xu et al., 2015; Miwa et al., 2016; Zhang et al., 2016

Problem: LSTM networks struggle to capture the multimodal characteristics of richly formatted data.
Augmenting LSTM with Multimodal Features

NPN Silicon Switching Transistors

- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

<table>
<thead>
<tr>
<th>Maximum Ratings</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200 mA</td>
<td></td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{tot}$</td>
<td>250 mW</td>
<td></td>
</tr>
<tr>
<td>$T_S \leq 71 ^\circ C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_S \leq 115 ^\circ C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_J$</td>
<td>150 °C</td>
<td></td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65...150 °C</td>
<td></td>
</tr>
</tbody>
</table>

We use the multimodal information stored in the document to extract basic multimodal features:

- Structural features
- Tabular features
- Visual features

Augmentation with multimodal features captures signals a traditional LSTM would miss.
Signals from different modalities can be useful to find the information.

Fonduer: a KBC system that takes advantage of both techniques to reason about all available signals.
The impact of multimodal features

For transistor datasheets...

Multimodal features significantly impact the quality of extraction
Fonduer in the wild
Empirical results & real-world uses
### Fonduer vs. Human-curated Knowledge Bases

<table>
<thead>
<tr>
<th>Human-created</th>
<th>Machine-created</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 years</td>
<td>&lt;6 months</td>
</tr>
<tr>
<td>1.0x extractions</td>
<td>1.59x extractions</td>
</tr>
</tbody>
</table>

**Same set of documents**

**Fonduer**

- **Precision**: 0.89
Input: User-customized HTML auction pages → Output: Structured knowledge base

Extract key facts (make, model, license, etc.)

Improve auction search quality and UX

How people use Fonduer in industry
Knowledge Base Construction from Richly Formatted Data

- Fonduer helps build high-quality KBC from richly formatted data
- Allows users to leverage multimodal signals
- Augments deep learning model with features from each data modality to achieve high quality
- Fonduer is supporting real world applications

Thank you!

Sen Wu
(senwu@cs.stanford.edu)

https://github.com/HazyResearch/fonduer
The Fonduer data model

Richly formatted data

SMBT3904...MMBT3904

NPN Silicon Switching Transistors
- High DC current gain: 0.1 mA to 100 mA
- Low collector-emitter saturation voltage

Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage</td>
<td>$V_{CEO}$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Collector-base voltage</td>
<td>$V_{CBO}$</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{EBO}$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Collector current</td>
<td>$I_C$</td>
<td>200</td>
<td>mA</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>$P_{tot}$</td>
<td>330</td>
<td>mV</td>
</tr>
<tr>
<td>$T_S \leq 71{^\circ}C$</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>$T_S \leq 115{^\circ}C$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Junction temperature</td>
<td>$T_J$</td>
<td>150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>$T_{stg}$</td>
<td>-65 ... 150</td>
<td></td>
</tr>
</tbody>
</table>

Fonduer automatically parses the richly formatted data into the data model that:
- Preserves structure/semantics across modalities
- Unifies a diverse variety of formats and styles
- Serves as the formal representation in KBC
We want to detect and repair errors in a dataset

University of Chicago, *Chicago*, IL

Where does data cleaning come up? All analytics!

- Data feeds
- Investment
- Urban data
### Chicago’s food inspection dataset

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td><strong>Johnnyo's</strong></td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
</tbody>
</table>

**Detect and repair** errors in a structured dataset.
Constraints and minimality

Functional dependencies

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

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t4</td>
<td>Johnnyo’s</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

Bohannon et al., 2005, 2007; Kolahi and Lakshmanan, 2005; Bertossi et al., 2011; Chu et al., 2013; 2015 Fagin et al., 2015
Functional dependencies

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>John Veliotis Sr.</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>Johnnyo's</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

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

Action: Fewer erroneous than correct cells; perform minimum number of changes to satisfy all constraints
## Constraints and minimality

**Functional dependencies**

- $c_1$: DBAName $\rightarrow$ Zip
- $c_2$: Zip $\rightarrow$ City, State
- $c_3$: City, State, Address $\rightarrow$ Zip

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t4</td>
<td>Johnnyo's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

*Error; correct zip code is 60608*

**Does not fix errors and introduces new ones.**
### 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

### External list of addresses

<table>
<thead>
<tr>
<th>Ext_Address</th>
<th>Ext.City</th>
<th>Ext.State</th>
<th>Ext.Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>1208 N Wells ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60610</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKANName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t4</td>
<td><strong>Johnnyo's</strong></td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

---

Fan et al., 2009; Bertossi et al., 2010; Chu et al., 2015
## 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  

## External list of addresses

<table>
<thead>
<tr>
<th>Ext_Address</th>
<th>Ext_City</th>
<th>Ext_State</th>
<th>Ext_Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>1208 N Wells ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60610</td>
</tr>
</tbody>
</table>

### Action: Map external information to input dataset using matching dependencies and repair disagreements
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

External list of addresses

<table>
<thead>
<tr>
<th>Ext_Address</th>
<th>Ext.City</th>
<th>Ext.State</th>
<th>Ext_Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>1208 N Wells ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60610</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t4</td>
<td><strong>Johnyo’s</strong></td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

External dictionaries may have limited coverage or not exist altogether
Quantitative statistics

Reason about co-occurrence of values across cells in a tuple

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td><strong>Chicago</strong></td>
<td>IL</td>
<td><strong>60608</strong></td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td><strong>60609</strong></td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td><strong>60609</strong></td>
</tr>
<tr>
<td>t4</td>
<td><strong>Johnyo’s</strong></td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td><strong>60608</strong></td>
</tr>
</tbody>
</table>

Example: Chicago co-occurs with IL

Hellerstein, 2008; Mayfield et al., 2010; Yakout et al., 2013
## Quantitative statistics

Reason about co-occurrence of values across cells in a tuple

Estimate the distribution governing each attribute

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t4</td>
<td>John Veliotis Sr.</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

Again, fails to repair the wrong zip code
Let’s combine everything

<table>
<thead>
<tr>
<th>Constraints and minimality</th>
<th>External data</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBAName</td>
<td>AKAName</td>
</tr>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
</tr>
<tr>
<td>t4</td>
<td>John Veliotis Sr.</td>
</tr>
</tbody>
</table>

**Quantitative statistics**

<table>
<thead>
<tr>
<th>DBAName</th>
<th>AKAName</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>John Veliotis Sr.</td>
<td>Johnny's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>t2</td>
<td>John Veliotis Sr.</td>
<td>Johnny's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>t3</td>
<td>John Veliotis Sr.</td>
<td>Johnny's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>t4</td>
<td>John Veliotis Sr.</td>
<td>Johnny's</td>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
</tr>
</tbody>
</table>

Different solutions suggest different repairs
Probabilistic data repairs

HoloClean
[VLDB’17]
Probabilistic data repairs

HoloClean
[VLDB’17]
Error detection in HoloClean

HoloClean focuses on repairing. Error detection is treated as black-box.

Error detection splits input into **correct and potentially erroneous cells**.

---

Input

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

Output

Correct cells:

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td><strong>Chicago</strong></td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td><strong>Chicago</strong></td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td><strong>Chicago</strong></td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td><strong>Chicago</strong></td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

Potentially erroneous cells:

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>1208 N Wells ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60610</td>
</tr>
</tbody>
</table>

---

Error Detection Example:

**Zip → City**

**External:**

<table>
<thead>
<tr>
<th>Ext_Address</th>
<th>Ext.City</th>
<th>Ext.State</th>
<th>Ext.Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60606</td>
</tr>
<tr>
<td>1208 N Wells ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60610</td>
</tr>
</tbody>
</table>
Probabilistic data repairs

HoloClean
[VLDB’17]
HoloClean’s model for data repairs

- Each cell is a random variable
- Value co-occurrences capture data statistics
- Constraints introduce correlations
- \( c_1: \text{Zip} \rightarrow \text{City} \)

```
<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>
```

- Unknown (to be inferred) RV
- Factor (encodes correlations)
Probabilistic data repairs

HoloClean [VLDB’17]
HoloClean’s model

Factor Graph

\[ \begin{align*}
\text{t1.City} & \quad \text{t1.Zip} \\
\text{w1} & \quad \text{w2} \\
\text{t4.City} & \quad \text{t4.Zip} \\
\text{w1} & \quad \text{w2}
\end{align*} \]

Exponential family (canonical form)

\[ w = (w_1, w_2, \ldots, w_s)^T \]

\[ P(x|w) = \exp \left( \sum_{i=1}^{s} w_i T_i(x) - A(w) \right) \]

HoloClean automatically generates a factor graph that captures:
- Co-occurrences
- Correlations due to constraints
- Evidence due to external data

Repairing is a learning and inference problem:
Learn parameters \( w \) (use SGD) and infer the marginal distribution for unknown variables (use Gibbs sampling)
Probabilistic data repairs

HoloClean is a compiler for automatically generating probabilistic programs for data cleaning
HoloClean in practice

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

State-of-the-art does not scale or performs no correct 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

<table>
<thead>
<tr>
<th>Tuple ID</th>
<th>University</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>U of Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>t2</td>
<td>U of Chicago</td>
<td>IL</td>
</tr>
<tr>
<td>t3</td>
<td>U of Chicago</td>
<td>CA</td>
</tr>
</tbody>
</table>

Functional dependency: University $\rightarrow$ State

“The same University must be in the same State”

FDs correspond to constraints over random variables (RVs)

Example:


Naive globally consistent model: It introduces correlations over four random variables.

We have $P^4$ possible worlds for such correlations.

D: domain of random variables
## Relaxing constraints

**Tuple ID** | **University** | **State**  
---|---|---
**t1** | U of Chicago | IL  
**t2** | U of Chicago | IL  
**t3** | U of Chicago | CA

**Functional dependency:**

University $\rightarrow$ State

“**The same University must be in the same State**”

**Relax constraints to features over independent RVs**  
(\textit{corresponds to a voting model})

**Example:**

\[
\begin{align*}
t1.\text{University} &= \text{U of Chicago} & \Rightarrow & & \text{IL} = \text{CA} \\
\text{U of Chicago} &= t3.\text{University} & \Rightarrow & & \text{IL} = \text{CA} \\
\text{U of Chicago} &= \text{U of Chicago} & \Rightarrow & & t1.\text{State} = \text{CA} \\
\text{U of Chicago} &= \text{U of Chicago} & \Rightarrow & & \text{IL} = t3.\text{State}
\end{align*}
\]

Only 4$D$ possible worlds considered

HoloCleans’ locally consistent model introduces features over independent random variables.
Relaxing constraints

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

“Address=3465 S Morgan St”

“Zip -> City”

“Address=3465 S Morgan St”
Relaxing constraints

<table>
<thead>
<tr>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1 3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
<tr>
<td>t2 3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t3 3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60609</td>
</tr>
<tr>
<td>t4 3465 S Morgan ST</td>
<td>Chicago</td>
<td>IL</td>
<td>60608</td>
</tr>
</tbody>
</table>

“Assignment Chicago violates Zip -> City due to t4”

“Address = 3465 S Morgan St”

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

“Assignment Chicago violates Zip -> City due to t1”
We have one relaxed factor for each value in the domain of the RV.
Faster compilation, learning, and inference when we do not prune the RV domain.
Increased robustness (more accurate repairs) when RV domain is ill-specified (no heavy pruning used)
Data cleaning is a ML problem

The HoloClean Framework

1. Error Detection Module
   - Use integrity constraints
   - Leverage external data
   - Detect outliers
   - Identify possible repairs

2. Compilation Module
   - Automatic Featurization
   - Statistical analysis and candidate repair generation
   - Compilation to factors/tensors

3. Repair Module
   - Ground probabilistic model
   - Statistical learning (weights)
   - Probabilistic inference

1. Combine disparate signals to perform accurate data repairs

2. Data cleaning is a statistical learning and inference problem
   - Transition from logic to probability

3. Connections to data vs knowledge tradeoffs in structured prediction
A quest for rigor

1. HoloClean provided empirical evidence the probabilistic methods work better.

2. The ad-hoc relaxations for efficiency give more accurate data repairs.

Why did logic fail us? and Why does relaxing constraints work?
1. In 1969, Edgar F. Codd introduced the relational data model.

2. In 2007, C.J. Date wrote that logic and databases are “inextricably intertwined”.

Two main uses of logic in databases

1. Logic is used as a database query language to express questions asked against databases.

2. Logic is used as a specification language to express integrity constraints in databases.
Noise models in DB theory

Coping with Inconsistent Databases

Two different approaches:

- **Data Cleaning**: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying (adding, deleting, updating) tuples in relations.
  - This is the main approach in industry (e.g., IBM InfoSphere Quality Stage, Microsoft DQS).
  - More engineering than science as quite often arbitrary choices have to be made.

- **Database Repairs**: A framework for coping with inconsistent databases in a principled way and without "cleaning" dirty data first.

Slide by Phokion Kolaitis [SAT 2016]
Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)

$\Sigma$ a set of integrity constraints and $I$ an inconsistent database.
A database $J$ is a repair of $I$ w.r.t. $\Sigma$ 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]
Noise models outside DB

original word

noisy channel

noisy word
Noise models outside DB

Noisy Channel

1. We see an observation $x$ in the noisy world

2. Find the correct world $w$

$$\hat{w} = \arg \max_{w \in W} P(w|x)$$

Applications

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

*Let’s try new foundations for data cleaning!*  
…and see how they relate to logic.*
Probabilistic Unclean Databases

Intentional Data Model

Step 1: Tuples are generated independently

\[
P(R^D) \stackrel{\text{def}}{=} \prod_{i \in \text{ids}_R(D)} (p_R \cdot \mathbf{T}_{G_R(D[i])}) \times \prod_{i \in \text{ids}_R(D)} (1 - p_R).
\]

- Probability that tuple index was included in the world
- Probability obtaining a certain value

Step 2: Logical constraints ensure consistency

\[
M(D) \stackrel{\text{def}}{=} \frac{1}{Z} \times P(D) \times \prod_{\varphi \in \Phi} e^{-w(\varphi) \cdot |V(D, \varphi)|}
\]

- Log-linear model penalizing invalid "possible worlds"

[Work under submission, 2018]
Probabilistic Unclean Databases

(A) Schema, Attribute Domain, and Constraint Specification

Tuple ID

Business Listing

Integrity Constraints
PK: Business ID
FD: Zip Code $\rightarrow$ City, State

<table>
<thead>
<tr>
<th>Tuple ID</th>
<th>Business ID</th>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Porter</td>
<td>Madison</td>
<td>WI</td>
<td>53703</td>
</tr>
<tr>
<td>2</td>
<td>Graft</td>
<td>Madison</td>
<td>WI</td>
<td>53703</td>
</tr>
<tr>
<td>3</td>
<td>EVP Coffee</td>
<td>Madison</td>
<td>WI</td>
<td>53703</td>
</tr>
</tbody>
</table>

(B) The Two-Actor Generation Process

Tuple Identifiers

Constraints $\Phi$

Intentional Data Model $\mathcal{I}$

Sample of clean intended data $\mathcal{I}$

<table>
<thead>
<tr>
<th>Business ID</th>
<th>City</th>
<th>State</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Porter</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>2</td>
<td>Graft</td>
<td>Veronica</td>
<td>WI</td>
</tr>
<tr>
<td>3</td>
<td>EVP Coffee</td>
<td>Madison</td>
<td>WI</td>
</tr>
<tr>
<td>4</td>
<td>Graft</td>
<td>Chicago</td>
<td>IL</td>
</tr>
</tbody>
</table>

Dirty data instance $J^*$ observed after applying the Realizer

Realizer Model

$\mathcal{R}_\mathcal{I}(I, J) \overset{\text{def}}{=} \mathcal{I}(I) \cdot \mathcal{R}_\mathcal{I}(J)$

Probability assigned to an intended instance $I$

Conditional prob. of getting $J$ given $I$

We consider two models
1. Insert unintended tuples (subset)
2. Update values of existing tuples

These models capture the data errors considered in prior works
Probabilistic Unclean Databases

(A) Schema, Attribute Domain, and Constraint Specification

(B) The Two-Actor Generation Process

Computational problems

1. Cleaning: Find most probable $I$

2. Probabilistic query answering (PQA): evaluate a query directly on $J$

3. Learning Intentional and Realizer models
Probabilistic Unclean Databases

Preliminary Results

1. Cleaning: Connections to minimum repairs
2. Cleaning is in P-time for key constraints
3. Connections to consistent query answering
4. Learning from one noisy database without training data
Theorem

For a *subset realizer* with *low noise* probabilistic repairs and minimal subset repairs are equivalent.

**Subset realizer**: Noisy channel that introduces new tuples

**Low noise**: probability of insertion from realizer lower than probability of insertion from intentional model

No assumptions on *tuple independence or attribute value independence*. 
Probabilistic Cleaning vs. Minimal Repairs

Theorem
For an update realizer with low noise probabilistic repairs and cardinality minimal subset repairs are equivalent when (1) tuples are independent and (2) tuple attribute assignments are independent!

Update realizer: Noisy channel that permutes the values of cells (tuple attributes)
Low noise: probability of update less than 0.5

Strong assumptions that violate the relational model!
1. HoloClean provided empirical evidence the probabilistic methods work better.

2. The ad-hoc relaxations for efficiency give more accurate data repairs.

*Why did logic fail us? and Why does relaxing constraints work?*
How hard is structured prediction?

Cleaning is a structured prediction problem

Our relaxation corresponds to an approximation for structured prediction.

Recent work is targeting hardness of structured prediction.

Globerson et al., ICML 2015
Foster et al., AISTATS 2018

Setup: (with noise)
- known graph $G=(V,E)$
- unknown labeling $X:V \to \{0,1\}$
- given noisy parity of each edge
  - flipped with probability $p$

Goal: (approximately) recover $X$.

Formally: want algorithm $A: \{+,-\}^E \to \{0,1\}^V$ that minimizes worst-case expected Hamming error:

$$\max_X \{E_{L \sim D(X)}[\text{error}(A(L), X)]\}$$

We are working on extensions to categorical variables and hypergraphs.