Snorkel + Data Programming: **Beyond Hand-Labeled Training Data**

Alex Ratner

Stanford University, InfoLab

MOTIVATION:

In practice, training data is often:

- *The* **bottleneck**
- *The* practical injection point for domain knowledge

KEY IDEA:

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*

My Amazing Collaborators

Stephen Bach

Henry Ehrenberg
(Facebook)

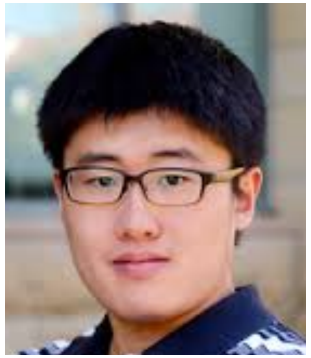
Chris De Sa
(Cornell)

Jason Fries

On the market!

Bryan He

Paroma Varma



Sen Wu

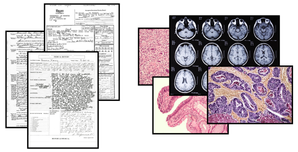
Braden Hancock

Chris Ré

And many more at Stanford & Beyond…

The ML Pipeline Pre-Deep Learning

Collection

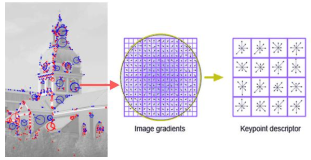


Labeling

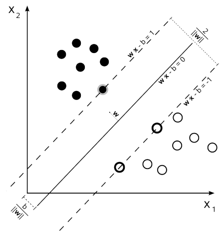
True

False

Feature Engineering

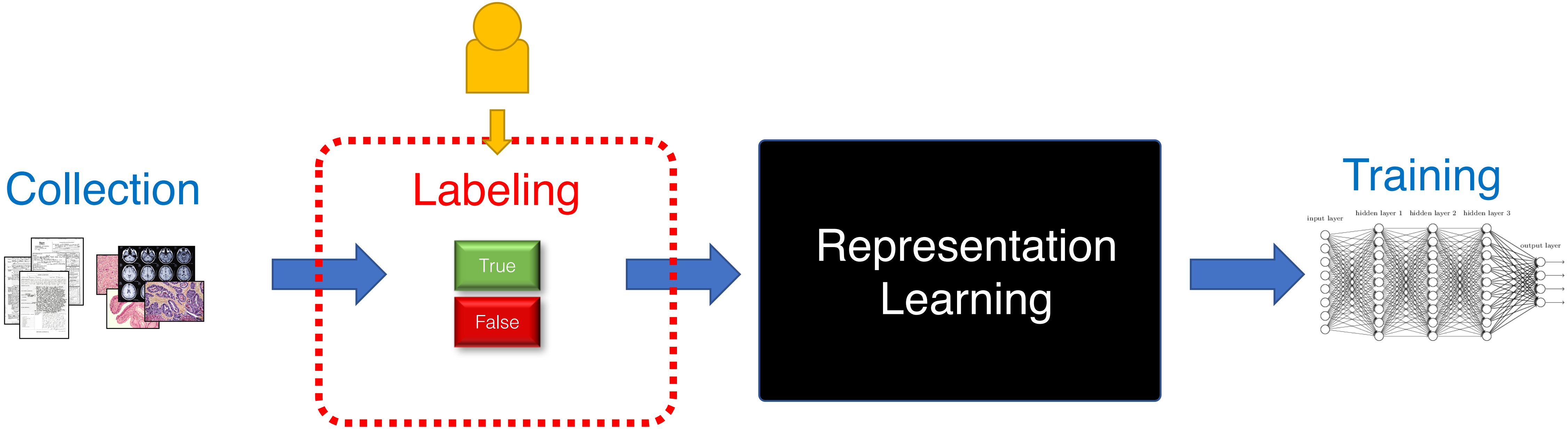


Training



Feature engineering *used to* be the bottleneck…

The ML Pipeline Today



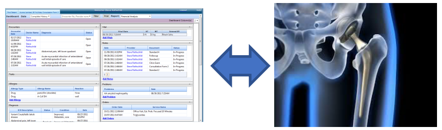
New pain point, new injection point

Training Data: Challenges & Opportunities

- Expensive & Slow:
  - *Especially when domain expertise needed*

Stanford MEDICINE

- 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.*

Shah Lab

U.S. Department of Veterans Affairs

Helix Group

FDA

How do we get—*and use*—training data more effectively?

Data Programming + Snorkel

A Framework + System for Creating Training Data with Weak Supervision

NIPS 2016

SIGMOD (Demo) 2017

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*

Generative Model

Noise-Aware Discriminative Model

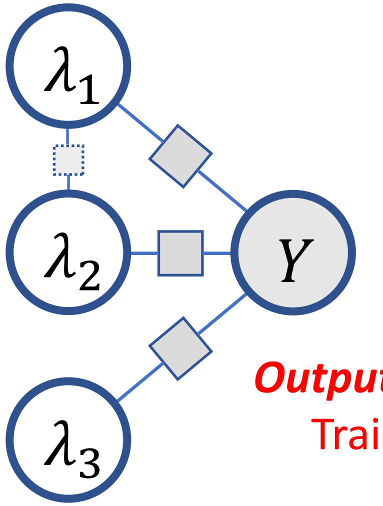
Ex. Application: Knowledge Base Creation (KBC)

DOMAIN EXPERT

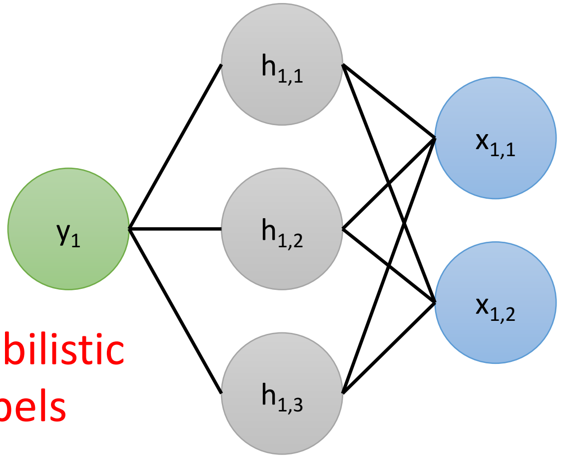
```
def lf1(x):
  cid = (x.chemical_id,
x.disease_id)
  return 1 if cid in KB else 0
```

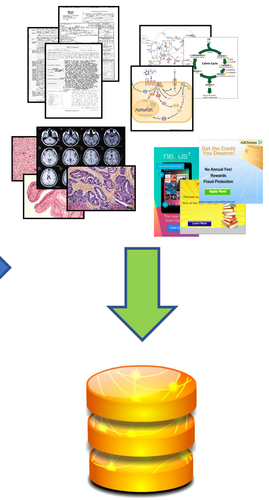
```
def lf2(x):
  m = re.search(r'.*cause.*',
x.between)
  return 1 if m else 0
```

```
def lf3(x):
  m = re.search(r'.*not
cause.*', x.between)
  return 1 if m else 0
```



Output: Probabilistic Training Labels

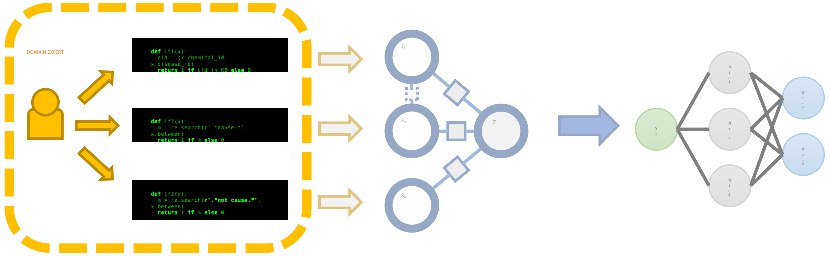




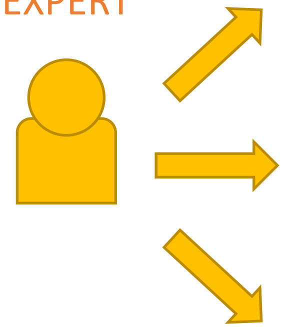
1. Users write *labeling functions* to generate noisy labels
2. We model the labeling functions' behavior to de-noise them
3. We use the resulting prob. labels to train a model

Surprising Point:

No hand-labeled training data!



DOMAIN EXPERT



```
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
```

Step 1: Writing Labeling Functions

A Unifying Framework for Expressing *Weak Supervision*

Example: Chemical-Disease Relation Extraction from Text

Helix Group

FDA

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*

ID	Chemical	Disease	Prob.
00	magnesium	Myasthenia gravis	0.84
01	magnesium	quadriplegic	0.73
02	magnesium	paralysis	0.96

KNOWLEDGE BASE (KB)

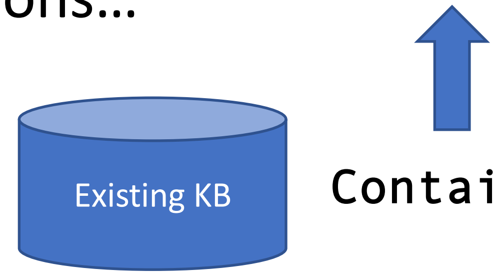
Labeling Functions

- Traditional "distant supervision" rule relying on external KB

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

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

Label = TRUE



Contains (A,B)

This is likely to be true… *but*

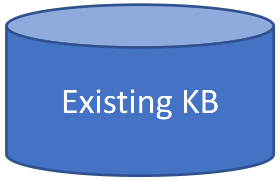
Labeling Functions

- Traditional "distant supervision" rule relying on external KB

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

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

`Label = TRUE`



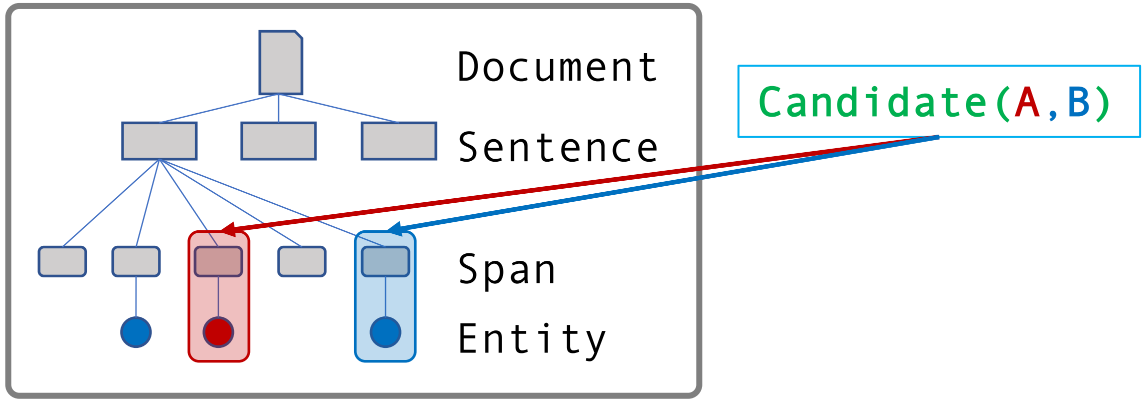
`Contains (A,B)`

...can be false!

We will learn the accuracy of each LF (next)

Writing Labeling Functions in Snorkel

- Labeling functions take in `Candidate` objects:



CONTEXT HIERARCHY

Key Point: *Supervision as code*

- Three levels of abstraction for writing LFs in Snorkel:
  - Python code

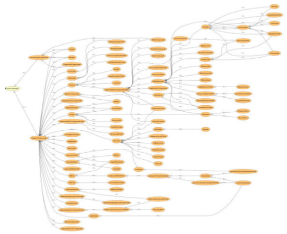
```
def lf1(x):
  cid =(x.chemical_id,x.disease_id)
  return 1 if cid in KB else 0
```

  - LF templates

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

  - LF generators

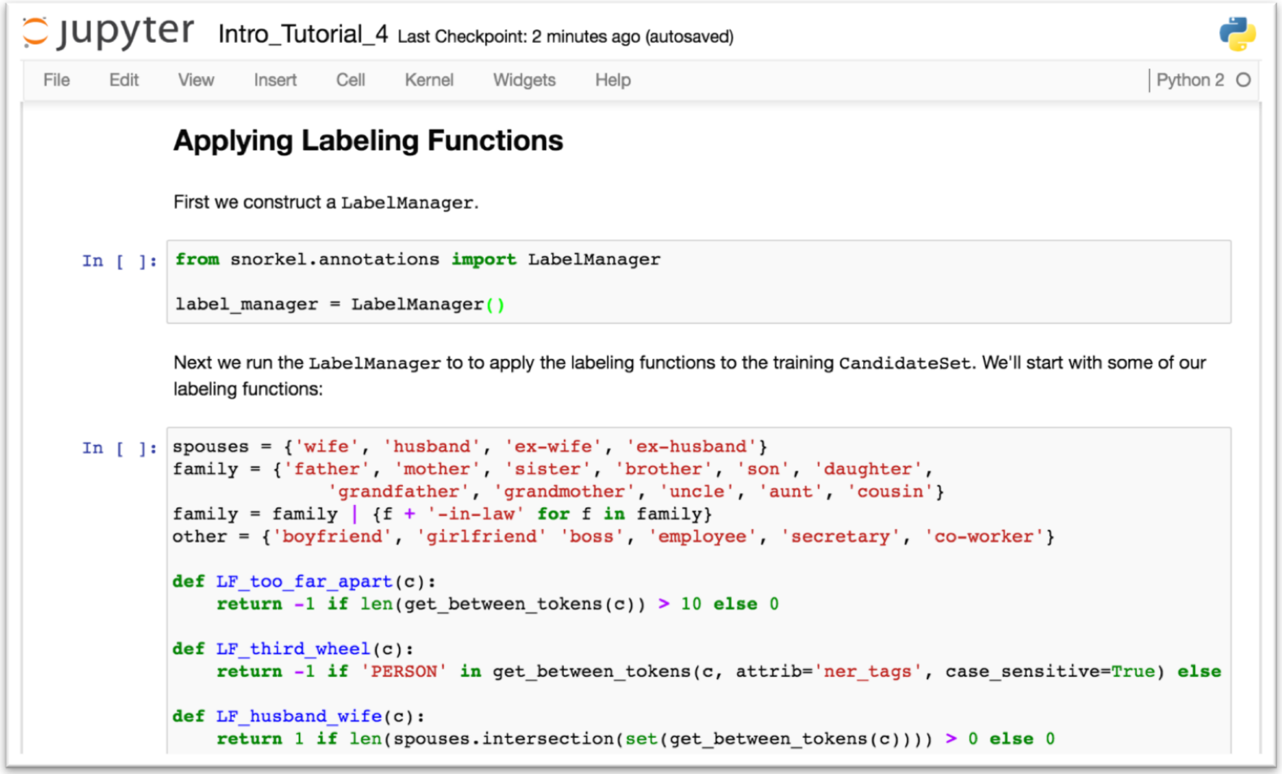
```
for lf in LF_DS_hier(KB, cut_level=2):
    yield lf
```



A knowledge base (KB) with hierarchy

Supported by Simple Jupyter Interface

snorkel



jupyter Intro_Tutorial_4 Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Python 2

Applying Labeling Functions

First we construct a `LabelManager`.

In [ ]:

```
from snorkel.annotations import LabelManager

label_manager = LabelManager()
```

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

In [ ]:

```
spouses = {'wife', 'husband', 'ex-wife', 'ex-husband'}
family = {'father', 'mother', 'sister', 'brother', 'son', 'daughter',
              'grandfather', 'grandmother', 'uncle', 'aunt', 'cousin'}
family = family | {f + '-in-law' for f in family}
other = {'boyfriend', 'girlfriend' 'boss', 'employee', 'secretary', 'co-worker'}

def LF_too_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

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

snorkel.stanford.edu

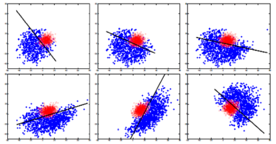
Broader Perspective:

A Template for Weak Supervision

A Unifying Method for Weak Supervision

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





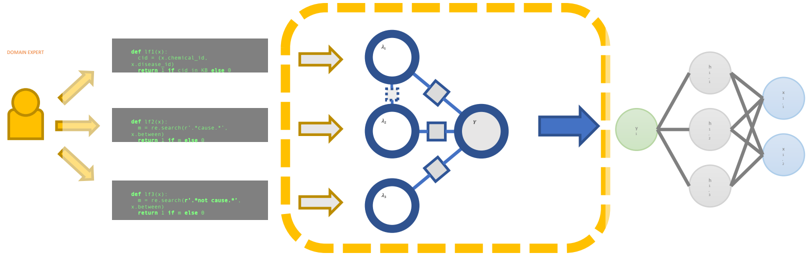

$$\lambda : X \mapsto Y \cup \{\emptyset\}$$

Related Work in Weak Supervision

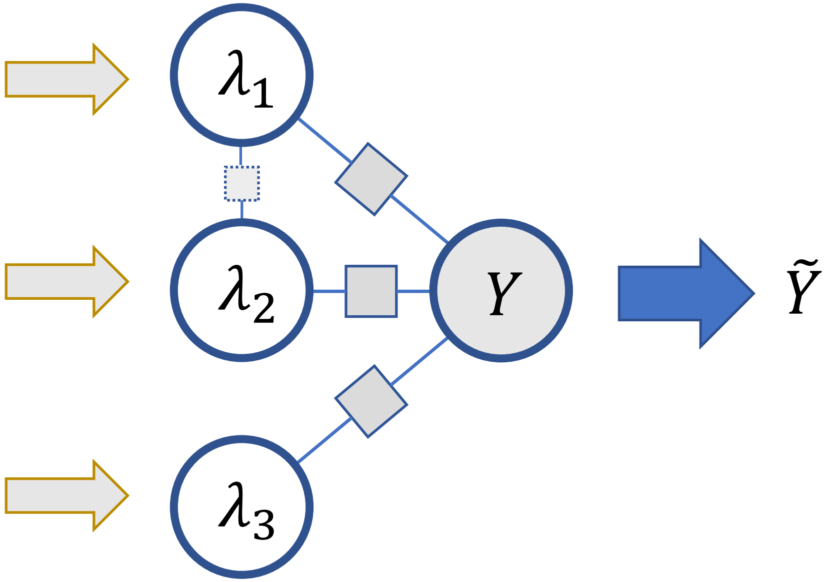
- ***Distant Supervision:*** Mintz et. al. 2009, Alfonesca et. al. 2012, Takamatsu et. al. 2012, Roth & Klakow 2013, Augenstein et. al. 2015, etc.
- ***Crowdsourcing:*** Dawid & Skene 1979, Karger et. al. 2011, Dalvi et. al. 2013, Ruvolo et. al. 2013, Zhang et. al. 2014, Berend & Kontorovich 2014, etc.
- ***Co-Training:*** Blum & Mitchell 1998
- ***Noisy Learning:*** Bootkrajang et. al. 2012, Mnih & Hinton 2012, Xiao et. al. 2015, etc.
- ***Indirect Supervision:*** Clarke et. al. 2010, Guu et. Al. et. al. 2017, etc.
- ***Feature and Class-distribution Supervision:*** Zaidan & Eisner 2008, Druck et. al. 2009, Liang et. al. 2009, Mann & McCallum 2010, etc.
- ***Boosting & Ensembling:*** Schapire & Freund, Platanios et. al. 2016, etc.
- ***Constraint-Based Supervision:*** Bilenko et. al. 2004, Koestinger et. al. 2012, Stewart & Ermon 2017, etc.

Check out our full list @ snorkel.stanford.edu/blog/ws_blog_post.html – *we love suggested additions or other feedback!*

How to handle such a diversity of weak supervision sources?

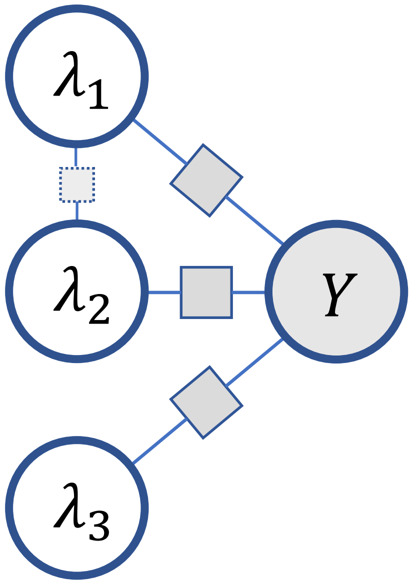


Step 2: Modeling Weak Supervision



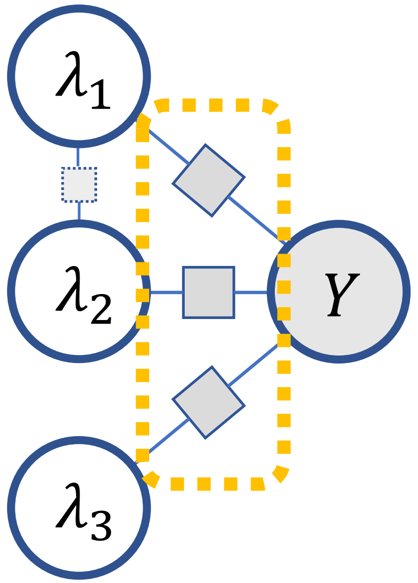
Weak Supervision: Core Challenges

- Unified input format
- Modeling
  - Accuracies of sources
  - Correlations between sources
  - Expertise of sources
- Using to train a wide range of models



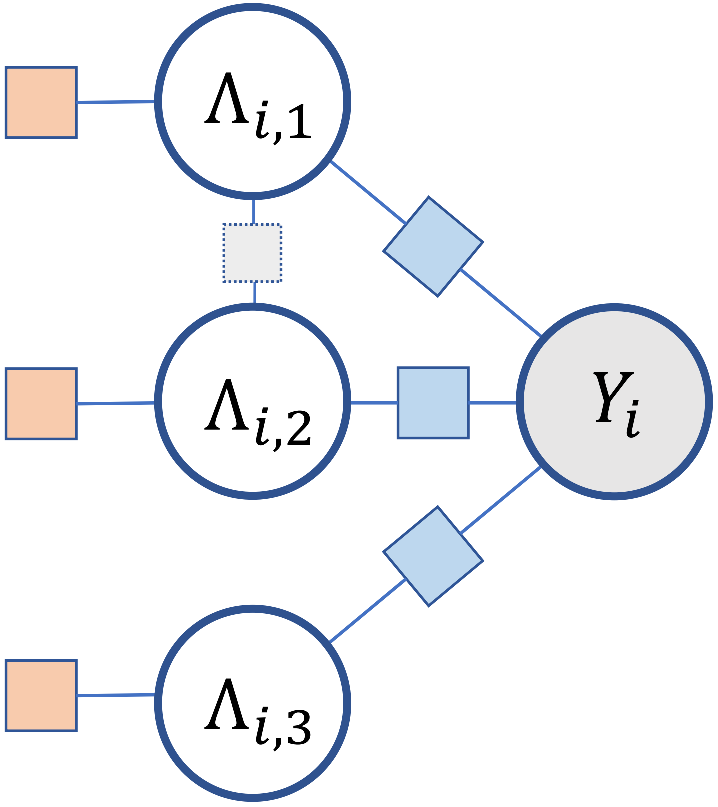
Weak Supervision: Core Challenges

- **Unified input format**
- Modeling
  - **Accuracies of sources** NIPS 2016
  - 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:
$\beta_j = p_\theta(\Lambda_{i,j} \neq \emptyset)$

$$f_j^{lab}(\Lambda_i, Y_i) = \exp(\theta_j^{lab} \Lambda_{i,j}^2)$$

Accuracy:
$\alpha_j = p_\theta\left(\Lambda_{i,j} = Y_i \mid Y_i, \Lambda_{i,j} \neq \emptyset\right)$

$$f_j^{acc}(\Lambda_i, Y_i) = \exp(\theta_j^{acc} \Lambda_{i,j} Y_i)$$

Correlations ICML 2017

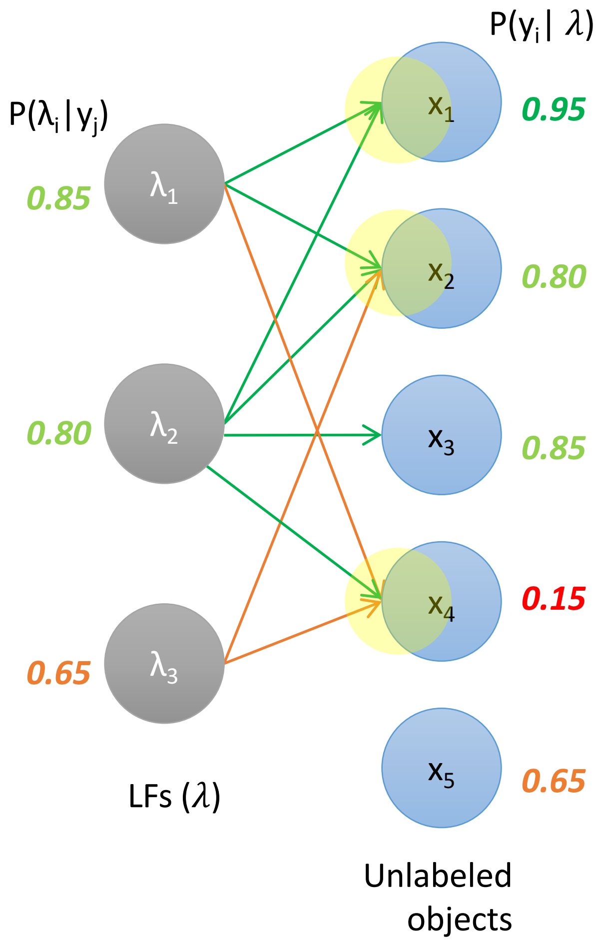
Intuition: Learning from Disagreements

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:

$$\hat{w} = \text{argmin}_w \frac{1}{N} \sum_{i=1}^{N} l(w, x^{(i)}, y^{(i)}) \qquad T = \{(x_1, 0), (x_2, 1), (x_3, 0), \dots\}$$

Here, we learn from the *noisy* labels:

$$\hat{w} = \text{argmin}_w \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{(\boldsymbol{y},\boldsymbol{\Lambda})\sim\boldsymbol{\pi}}[l(w, x^{(i)}, y^{(i)} = y)] \qquad T = \{(x_1, 0.1), (x_2, 0.6), (x_3, 0.3), \dots\}$$

Only requires simple tweak to loss function works over ***many models*** including Logistic Regression, SVMs and LSTMs.

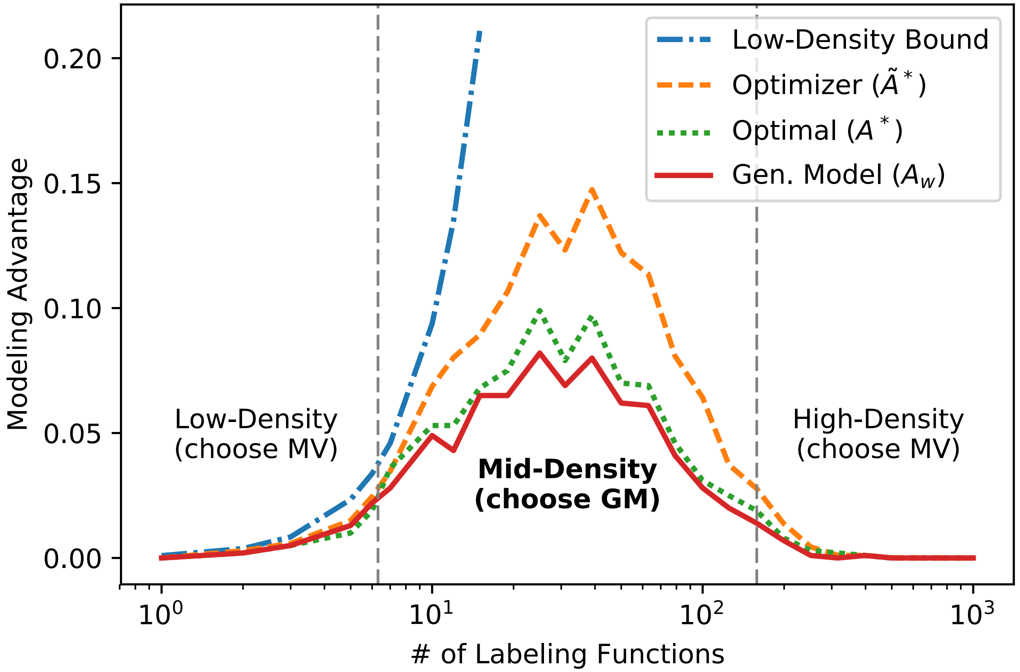
Theory: Scaling with *Unlabeled* Data

- We show that with:
  - $O(1)$ labeling functions of sufficient quality / expressiveness
  - $\tilde{O}(\epsilon^{-2})$ ***unlabeled*** training data points
  - → We get $O(\epsilon)$ generalization risk

This is the same asymptotic scaling as in supervised methods!

When is modeling the noise worthwhile?

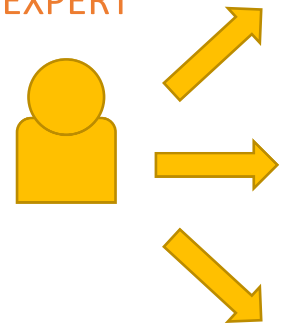
- Can look at *label density:*
  - Low: Too sparse to beat MV
  - High: MV approaches optimal
  - Medium: Just right!
- Can use conditional decision rule to safely skip gen. modeling stage
  - E.g. during early LF dev cycles



Putting it All Back Together

Input: Labeling Functions, *Unlabeled data*

DOMAIN EXPERT

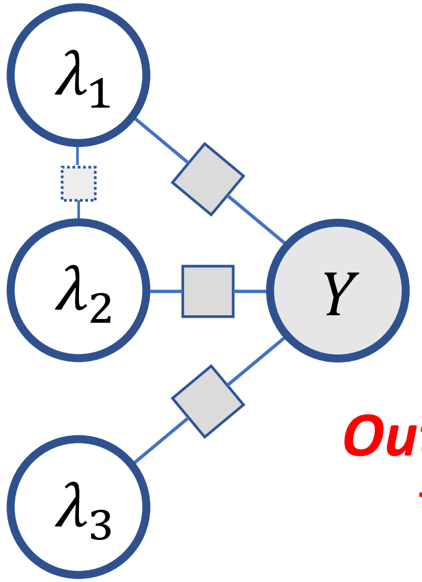


```
def lf1(x):
  cid = (x.chemical_id,
x.disease_id)
  return 1 if cid in KB else 0
```

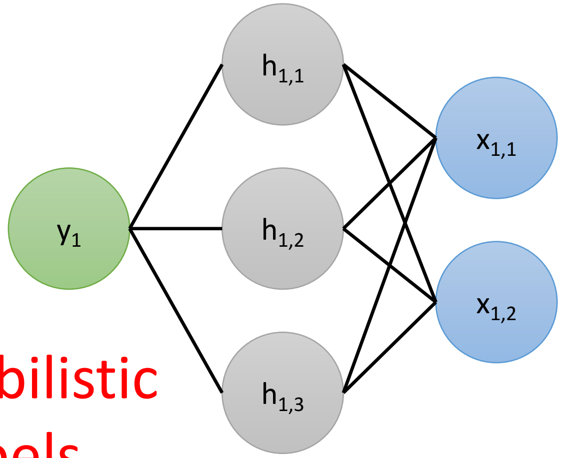
```
def lf2(x):
  m = re.search(r'.*cause.*',
x.between)
  return 1 if m else 0
```

```
def lf3(x):
  m = re.search(r'.*not
cause.*', x.between)
  return 1 if m else 0
```

Generative Model



Noise-Aware Discriminative Model



Output: Probabilistic Training Labels

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

How well does this work in practice?

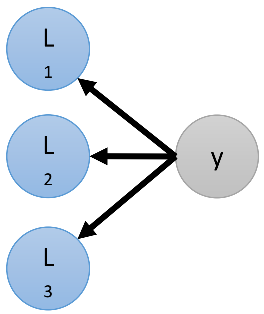
Empirical Results

Results on Chemical-Disease Relations

Precision: 25.5
Recall: 34.8
F1: **29.4**

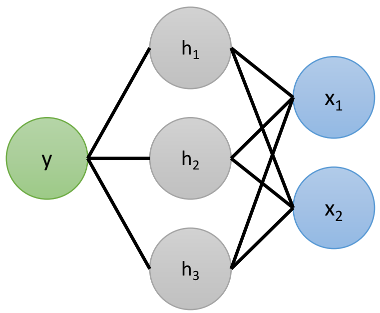
Distant Supervision

Precision: 52.3
Recall: 30.4
F1: **38.5**
+ 9.1



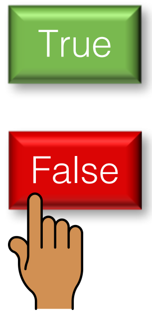
Generative Model

Precision: 38.8
Recall: 54.3
F1: **45.3**
+ 6.8



Discriminative Model

Precision: 39.9
Recall: 58.1
F1: **47.3**
+ 2.0



Hand Supervision

Snorkel is Powering Real Applications

Stanford MEDICINE

Shah Lab

Helix Group

FDA

ESOC Empirical Studies of Conflict

DARPA MEMEX

CISAC STANFORD

Greene Lab

Jet Propulsion Laboratory California Institute of Technology

I.R.D.S.

mobilize

NEC

accenture

mallicklab

DISTRICT ATTORNEY NEW YORK COUNTY

Alibaba Group

TOSHIBA

SING Stanford Information Networks Group

CANARY CENTER AT STANFORD

How easy is this to use in practice?

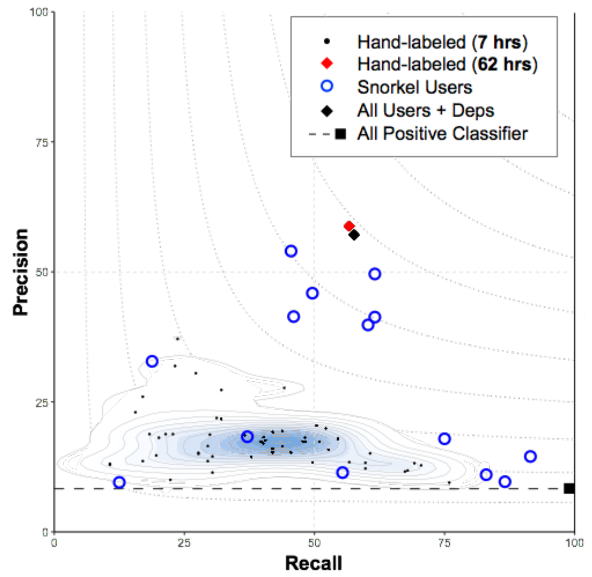
User Study

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

Marta Gaia Zanchi
@medinnovo
Following

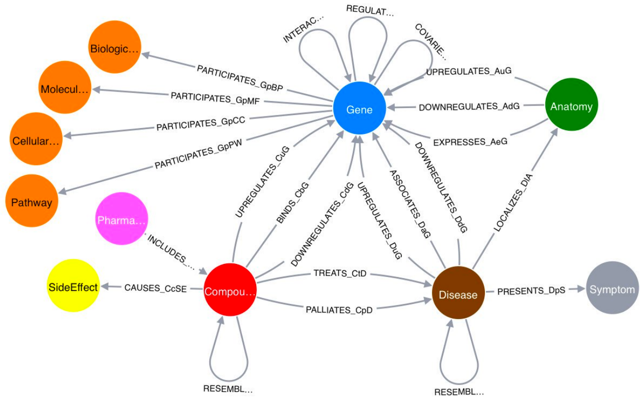
For a newbie, I write pretty darn good #Snorkel #MachineLearning labeling functions. Thanks @MobilizeCenter @jasonafries @stevebach :)

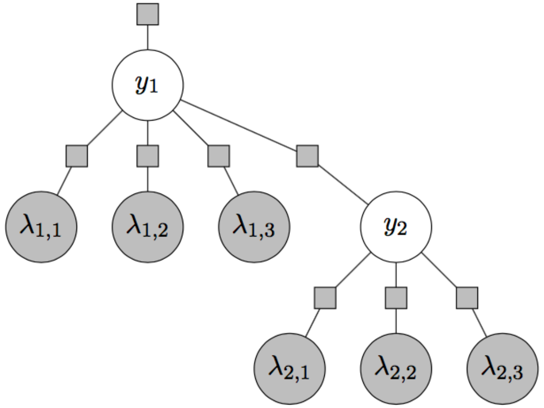
3rd Place Score

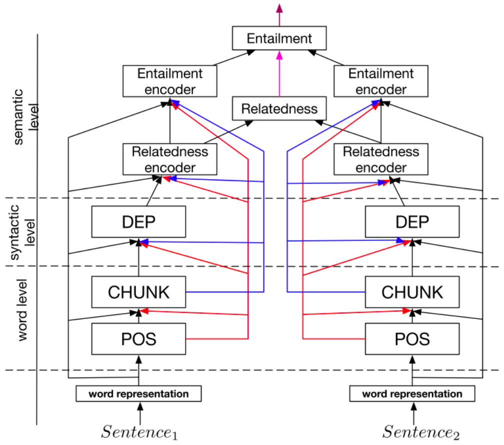
No machine learning experience
Beginner-level Python

What's Next: MTL?

- Hierarchical LFs as weakly-supervised MTL







- And more, see snorkel.stanford.edu

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

snorkel.stanford.edu