Snorkel
Rapidly Creating Training Sets to Program Software 2.0

Alex Ratner
Stanford University
Coauthors, Users & Sponsors

Stephen Bach (Brown)  Henry Ehrenberg (Facebook)  Jason Fries  Sen Wu

Chris De Sa (Cornell)  Braden Hancock  Paroma Varma  Chris Ré

And many more at Stanford & Beyond…
“Software 2.0” is eating Software 1.0

• Not just translation & computer vision anymore…
  • ETL & data cleaning (Holoclean), DB Tuning (Peloton), Video Streaming (Pensieve), Learned Indexes (Kraska et. al.)
A Brief Note on Reasons Why

• **Development Speed:** Google’s translation application went from 500K to 500 lines of code

• **Adaptability:** Commodity models get good performance “out of the box” on a broad range of tasks

• **Deployment:** NN frameworks as a new JVM

However…
The Training Data Bottleneck

Training data is:
• Expensive (need domain experts)
• Static
• Increasingly *the* critical ingredient

**KEY IDEA:**
We can use *noisy sources of signal*, specified at *higher-levels of abstraction*, to rapidly generate training sets.
Snorkel: A System for Rapidly Creating Training Sets

Goal: Bring all sources to bear to \textit{program} ML systems in a radically \textit{faster} and \textit{easier} way
The Snorkel Pipeline

1. Users write *labeling functions* to generate noisy labels

2. Snorkel models and combine these labels

3. We use the resulting *probabilistic* training labels to train a model

Key point: Input is *labeling functions*—No hand-labeled training sets
Key Ideas & Outline

Labeling Functions:

*Supervision as Code*

Label Model:

*Modeling the Noisy Labeling Process*

End Model:

*Easily Leveraging Comoditization Models*
Labeling Functions (LFs)

Supervision as Code
Simple LF Example: Pattern Matching

"Indication: Chest pain. Findings: Focal consolidation and pneumothorax."

```python
def LF_pneumothorax(c):
    if re.search(r'pneumo.*', c.report.text):
        return "ABNORMAL"
```

Label = ABNORMAL

*Labeling functions (LFs)* are black box UDFs that can express domain expertise
Simple LF Example: Pattern Matching

“Indication: Chest pain. Findings: No focal consolidation or pneumothorax…”

def LF_pneumothorax(c):
    if re.search(r'pneumo.*', c.report.text):
        return "ABNORMAL"

Label = ABNORMAL

However, LFs can be noisy! We can estimate their accuracies to handle this (next section)
A Unifying Method for Weak Supervision

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

We express these all as *labeling functions*
Supported by Simple Jupyter Interface

Key Idea: 
**Supervision as code**

```
from snorkel.annotations import LabelManager
label_manager = LabelManager()
```

```
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 0

def LF_husband_wife(c):
    return 1 if len(spouses.intersection(set(get_between_tokens(c)))) > 0 else 0
```
Problem: Labeling Functions Can…

• Overlap & conflict

• Have varying, \textit{unknown} accuracies

• Be correlated with each other

How to formalize & handle this messy input?
Modeling the Noise

De-noising and Combining Weak Supervision with Generative Modeling
A Generative Model of the Training Data Labeling Process

Core Technical Challenge: How to estimate these parameters without any ground-truth labels?
A Generative Model of the Training Data Labeling Process

$Y_i \in \{\text{NORMAL, ABNORMAL}\}$

Learned Accuracies

- 90% LF 1
- 80% LF 2
- 60% LF 3

Probabilistic Training Labels

Intuition: Learn the accuracies from the overlaps
Modeling Correlated LFs is Crucial

We can handle correlations using statistical (or static analysis) techniques

```python
def LF_1(X):
    if subfn_A(X) > 0.5:
        return 1

def LF_2(X):
    if subfn_A(X) > 0.7:
        return 1
```
Learning the Label Model \textit{without} Labels

- SGD + Gibbs Sampling over a Factor Graph
  - \textbf{Learn accs. that best explain the observed LF agreement/disagreement pattern} [NIPS 2016]
  - Estimate correlations automatically [ICML 2017] or based on static analysis of the LFs [NIPS 2017]

- \textbf{Late breaking:} New matrix-approximation approach that is orders of magnitude faster!

These techniques allow us to learn LF accuracies \textit{without} ground truth labels!
Recap: The Snorkel Pipeline

Users write *labeling functions* to generate noisy labels

Snorkel models and combine these labels

We use the resulting *probabilistic* training labels to train a model
Training the End Model
Leveraging the Commodityization of Models with Weak Supervision
We can use Snorkel + commodity ML models to:

- Improve the *recall (coverage)* automatically
- Deploy over different *servable* features or modalities
- Scale with unlabeled data
Goal: Training End Model to Generalize

**Input:** Labeling Functions, *Unlabeled data*

Noisy, conflicting labels

**Label Model**

Resolve conflicts, re-weight & combine

**End Model**

Generalize beyond the labeling functions
Results on Chemical-Disease Relations

Distant Supervision

Generative Model

Discriminative Model

Hand Supervision

Precision: 25.5
Recall: 34.8
F1: 29.4

Precision: 52.3
Recall: 30.4
F1: 38.5 + 9.1

Precision: 38.8
Recall: 54.3
F1: 45.3 + 6.8

Precision: 39.9
Recall: 58.1
F1: 47.3 + 2.0
Deploying Commodity Models Over Servable Features

1. Develop LFs over non-servable data (e.g. historical text reports)
2. Deploy models over servable data (e.g. real-time imaging)
3. We can train commodity models within pts. of multi-year hand-labeling efforts!
Scaling with Unlabeled Data

Empirical results confirm theory: Snorkel scales with the amount of *unlabeled* training data points
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

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

Marta Gaia Zanchi
@medimovo

3rd R4P $50
F1=44.3

Jason Fries, Stephen Bach, Alex Ratner, Joy Ku, Christopher Ré

Snorkel User Study

NIH NLM

mobilize Center for Mobility Data Integration to Insight

Cleveland Clinic Caltech
W2O Group
Stanford Hospital & Clinics
Wisconsin Caltech
Allen Institute for Artificial Intelligence

Hand-labeled (7 hrs)
Hand-labeled (62 hrs)
Snorkel Users
All Users + Depa
All Positive Classifier
What’s Next
Vision: Higher-level, declarative interfaces to ML

Users can provide supervision via higher-level interfaces such as:
- Writing LFs over unsupervised features [NIPS’17]
- Saying natural language explanations [ACL’18]
- Providing observational signals (e.g. eye trackers)

Goal: Make ML radically easier to program
Massively Multi-Task Learning

- Initial prototype: Snorkel MeTaL
  - [https://github.com/HazyResearch/metal](https://github.com/HazyResearch/metal) (pip: snorkel-metal)

Big vision: Amortizing labeling cost at organizational-scale + enabling new programming models
Conclusion and Next Steps

- *Supervision* as the declarative interface to Software 2.0

- Modeling the noise allows us to handle *higher-level, more diverse* supervision

- We can flexibly deploy *servable, commodity* models

**Code, tutorials, articles and more @ snorkel.stanford.edu. Feedback welcome!**
Modeling Supervision Induces New Tradeoff Spaces

- When to model at all?
- When and how much structure to model?

Key Idea: Accelerate initial human-in-the-loop development cycles