Snorkel --- VLDB 8/28/18





Snorkel Rapidly Creating Training Sets to Program Software 2.0

Alex Ratner

Stanford University



Coauthors, Users & Sponsors







(Brown)

(Facebook)

Jason Fries

Sen Wu





Braden Hancock Paroma Varma Chris De Sa (Cornell)

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





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.

R

Snorkel: A System for Rapidly Creating Training Sets



Goal: Bring all sources to bear to program ML systems in a radically faster and easier way

snorkel.stanford.edu



The Snorkel Pipeline



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*













Supervision as Code



Simple LF Example: Pattern Matching



Labeling functions (LFs) are black box UDFs that can express domain expertise



Simple LF Example: Pattern Matching



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



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

We express these all as *labeling functions*



Supported by Simple Jupyter Interface



Edit	View Insert Cell Kernel Widgets Help	hon 2 C
	Applying Labeling Functions	
	First we construct a LabelManager.	
In []:	from snorkel.annotations import LabelManager	
	<pre>label_manager = LabelManager()</pre>	
	Next we run the LabelManager to to apply the labeling functions to the training CandidateSet. We'll start with some of labeling functions:	our
In []:	<pre>spouses = {'wife', 'husband', 'ex-wife', 'ex-husband'} family = {'father', 'mother', 'sister', 'brother', 'son', 'daughter',</pre>	
In []:	<pre>spouses = {'wife', 'husband', 'ex-wife', 'ex-husband'} family = {'father', 'mother', 'sister', 'brother', 'son', 'daughter',</pre>	
In []:	<pre>spouses = {'wife', 'husband', 'ex-wife', 'ex-husband'} family = {'father', 'mother', 'sister', 'brother', 'son', 'daughter',</pre>	else

Key Idea: Supervision as code



Problem: Labeling Functions Can...

- Overlap & conflict
- Have varying, *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 $Y_i \in \{NORMAL, ABNORMAL\}$



Intuition: Learn the accuracies from the overlaps



Modeling Correlated LFs is Crucial

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

def LF_2(X):
 if subfn_A(X) > 0.7:
 return 1



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



Learning the Label Model without Labels

- SGD + Gibbs Sampling over a Factor Graph
 - 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]
- Late breaking: New matrix-approximation approach that is orders of magnitude faster!

These techniques allow us to learn LF accuracies *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 Commoditization 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



Noisy, conflicting labels

Resolve conflicts, re-weight & combine Generalize beyond the labeling functions



Results on Chemical-Disease Relations





Deploying Commodity Models Over *Servable* Features

- def LF_pneumothorax(c):
 if re.search(r'pneumo.*', c.report.text):
 return "ABNORMAL"
- def LF_pleural_effusion(c):
 if "pleural effusion" in c.report.text:
 return "ABNORMAL"
- def LF_normal_report(c, thresh=2):
 if len(NORMAL_TERMS.intersection(c.
 report.words)) > thresh:
 return "NORMAL"

Indication: Chest pain. Findings: Mediastinal contours are within normal limits. Heart size is within normal limits. No focal consolidation, pneumothorax or pleural effusion. Impression: No acute cardiopulmonary abnormality.

Develop LFs over *nonservable* data (e.g. historical text reports)





Receiver Operating Characteristic (ROC) curve





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



3rd Place Score

No machine learning experience Beginner-level Python

How well did these new Snorkel users do?



- 71% New Snorkel users matched or beat 7 hours of hand-labeling
- **2.8** x Faster than hand-labeling data



Average improvement in model performance





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]

- \bigcirc »
- 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 (pip: snorkel-metal)



Big vision: Amortizing labeling cost at organizational-scale + enabling new programming models



Conclusion and Next Steps





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 humanin-the-loop development cycles

