Snorkel:
Accelerating Machine Learning with Training Data Management

Alex Ratner
UW / Stanford
Snorkel Team @ Stanford

And many more!

snorkel.org
ML Application =

Model + Data + Hardware

State-of-the-art models and hardware are commodities

Training data is not

from tensorflow.models \ import resnet as model
import resnet2 as model

aws ec2 run-instances \ --instance-type p3.2xlarge
--instance-type p3.16xlarge
Training data is the key ingredient in ML

But it’s created and managed in manual, ad hoc ways
Example: Chest X-Ray Triage

Motivation: Case prioritization for e.g. low-resource hospitals

[Dunnmon et. al., *Radiology* 2018; Dunnmon & Ratner et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]
Example: Chest X-Ray Triage

Model dev is often radically easier today!

[Example image of a chest X-Ray triage process with labels for unlabeled data (multi-modal), training set creation, model development, and a specific model (e.g., ResNet).]

2-3 days

± 1 point due to model choice

[Dunnmon et. al., *Radiology* 2018; Dunnmon & Ratner et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]

(All scores: ROC AUC)
Example: Chest X-Ray Triage

Unlabeled data (multi-modal) → Training set creation → Model development → Model (e.g. ResNet)

- 8 months
- ± 9 points due to training set size
- ± 8 points due to training set quality
- 2-3 days
- ± 1 point due to model choice

Training data is often the key differentiator

[Ref: Dunnmon et. al., Radiology 2018; Dunnmon & Ratner et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]

(All scores: ROC AUC)
KEY IDEA:
Let users build and manage training datasets programmatically, then clean & integrate it for them.
Users write labeling functions to heuristically label data

Snorkel cleans and combines the LF labels

The resulting training database used to train an ML model

Radiology Example: ~8 hours writing LFs
Example: Fraud Detection

Goal: Be able to *rapidly adapt* training sets under changing conditions using *programmatic* labeling

[Bach et. al., SIGMOD Industry 2019]
Snorkel: Real-World Deployments

In many cases: From **person-months** of hand-labeling to **hours**
Where is weak supervision most helpful?

- **Private data** (can’t ship to crowd workers)

- **High-expertise data** (need specially-trained domain experts)

- **High rate-of-change tasks** (constant need to re-label)

High unit annotation cost integrated over time
How well does focusing on training data management work?
The (Super)GLUE Benchmark

General Language Understanding Evaluation

9 language understanding tasks
(NL inference, sentiment, etc.)

~1M total examples
SuperGLUE Example

**WiC task:** Is the target word being used in the same way in both sentences?

id: x1
Sentence 1: Call my **bank**.
Sentence 2: Find picnic spot near the river **bank**.
Label: **FALSE**

id: x2
Sentence 1: Play **Taylor Swift**.
Sentence 2: Text “hi!” to **Taylor Swift**.
Label: **TRUE**
Q: SOTA by specifying training data?

Task-Specific Linear Heads

Shared BERT-Large

Wordpiece Embeddings

24-Layer Transformer

= SOTA?
<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Model</th>
<th>URL</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SuperGLUE Human Baselines</td>
<td>SuperGLUE Human Baselines</td>
<td><img src="url" alt="Link" /></td>
<td>89.6</td>
</tr>
<tr>
<td>2</td>
<td>Stanford Hazy Research</td>
<td>Snorkel Metal</td>
<td><img src="url" alt="Link" /></td>
<td>74.5</td>
</tr>
<tr>
<td>3</td>
<td>SuperGLUE Baselines</td>
<td>BERT++</td>
<td><img src="url" alt="Link" /></td>
<td>70.5</td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td><img src="url" alt="Link" /></td>
<td>68.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td><img src="url" alt="Link" /></td>
<td>48.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Most Frequent Class</td>
<td><img src="url" alt="Link" /></td>
<td>46.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outside Best</td>
<td><img src="url" alt="Link" /></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

New SOTA score!
Three Key Training Data Operations

- **Labeling functions (LFs)**
  - Provide labels to unlabeled examples using domain heuristics

- **Transformation Functions (TFs)**
  - Augment data with per-example transformations (e.g. flips, crops for images)

- **Slicing functions (SFs)**
  - Partition the data, specifying where the model should add more capacity
Three Key Training Data Operations

Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics

Labeling data
Augment data with per-example transformations (e.g. flips, crops for images)

Slicing functions (SFs)
Partition the data, specifying where the model should add more capacity
SuperGLUE Labeling Function (LF)

```python
def lf_matching_trigrams(x):
    if trigram(x.sentences[0].target) == trigram(x.sentences[1].target):
        return TRUE
    else:
        return ABSTAIN
```

id: x1
Sentence 0: Can I \textit{invite} you for dinner on Sunday night?  
Sentence 1: The organizers \textit{invite} submissions of papers.  
Label: FALSE

id: x2
Sentence 0: He felt a \textit{stream} of air.  
Sentence 1: The hose ejected a \textit{stream} of water.  
Label: TRUE
Three Key Training Data Operations

Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics

Transformation Functions (TFs)
Augment data with per-example transformations (e.g. flips, crops for images)

Transforming data
Partition the data, specifying where the model should add more capacity
SuperGLUE Transformation Function (TF)

```python
def tf_days_of_the_week(x):
    yield x
    for DAY in DAYS_OF_WEEK:
        yield replace_with_synonym(x, word=DAY, synonyms=DAYS_OF_WEEK)
```

id: x1
Sentence 1: Can I invite you for dinner on Sunday night?
Sentence 2: The organizers invite submissions of papers.

Sentence 1: Can I invite you for dinner on Sunday night?
Sentence 1: Can I invite you for dinner on Monday night?
Sentence 1: Can I invite you for dinner on Tuesday night?
Sentence 1: Can I invite you for dinner on Wednesday night?
Sentence 1: Can I invite you for dinner on Thursday night?
Sentence 1: Can I invite you for dinner on Friday night?
Sentence 1: Can I invite you for dinner on Saturday night?
Three Key Training Data Operations

Partitioning data

Labelling functions (LFs)
Provide labels to unlabeled examples using domain heuristics

Transformation Functions (TFs)
Augment data with per-example transformations (e.g., flips, crops)

Slicing functions (SFs)
Partition the data, specifying where the model should add more capacity
SuperGLUE Slicing Function (SF)

def sf_target_is_noun(x):
    if x.sentences[0].target.pos == NOUN and x.sentences[1].target.pos == NOUN
        return NOUN_SLICE
    else:
        return ABSTAIN

id: x1
Sentence 0: Can I invite you for dinner on Sunday night?  sf_target_is_noun(x1) == ABSTAIN
Sentence 1: The organizers invite submissions of papers.

id: x2
Sentence 0: He felt a stream of air.  sf_target_is_noun(x2) == NOUN_SLICE
Sentence 1: The hose ejected a stream of water.
Key Idea: Let users spend their time building and modifying the training data.

snorkel.org
Three Key Training Data Operations

Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics

Augment data with per-example transformations (e.g. flips, crops for images)

Slicing functions (SFs)
Partition the data, specifying where the model should add more capacity
Problem: Hand-labeling is slow, expensive, & static

Idea: Enable users to label training data programmatically
The Snorkel Pipeline

Users write *labeling functions* to heuristically label data.

Snorkel *cleans and combines* the LF labels.

The resulting training database *used to train an ML model*.

*Note: No hand-labeled training data!*
(1) Writing Labeling Functions

Users write **labeling functions** to heuristically label data

**Snorkel cleans and combines the LF labels**

The resulting training database used to train an ML model

```
def LF_pneumo(x):
    if re.search(r'pneumo.*', X.text):
        return "ABNORMAL"

def LF_short_report(x):
    if len(X.words) < 5:
        return "NORMAL"

def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"

def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"
```
SuperGLUE Labeling Function (LF)

```
def lf_matching_trigrams(x):
    if trigram(x.sentences[0].target) == trigram(x.sentences[1].target):
        return TRUE
    else:
        return ABSTAIN
```

id: x1
Sentence 0: Can I *invite you* for dinner on Sunday night?
Sentence 1: The organizers *invite* submissions of papers.
Label: FALSE

id: x2
Sentence 0: He felt a *stream* of air.
Sentence 1: The hose ejected a *stream* of water.
Label: TRUE
Snorkel as a management layer for human (e.g. internal crowd) + programmatic labeling
Labeling Functions

def LF_pneumo(x):
    if re.search(r'pneumo\.*', X.text):
        return "ABNORMAL"

def LF_short_report(x):
    if len(X.words) < 15:
        return "NORMAL"

def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"

def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"

def LF_ontology(x):
    if DISEASES & X.words:
        return "ABNORMAL"

Result:
Supervision as Code

But, very messy supervision...
(1) Writing Labeling Functions

Users write labeling functions to heuristically label data

Snorkel cleans and combines the LF labels

The resulting training database used to train an ML model

```
def LF_short_report(x):
    if len(x.words) < 5:
        return "NORMAL"

def LF_off_shelf_classifier(x):
    if off_shelf_classifier(x) == 1:
        return "NORMAL"

def LF_pneumo(x):
    if re.search("pneumo.*", x.text):
        return "ABNORMAL"

def LF_ontology(x):
    if DISEASES & x.words:
        return "ABNORMAL"
```
(2) Clean & integrate noisy labels

Users write 

labeling functions
to heuristically label data

Snorkel cleans and combines the LF labels

The resulting training database used to train an ML model

How can we do this without ground-truth labels?
Key idea: Learn from the agreements & disagreements between the LFs

[Ratner et. al., AAAI ’19]
[Ratner et. al., NeurIPS ‘16]
(2) Clean & integrate noisy labels

Users write labeling functions to heuristically label data

Snorkel cleans and combines the LF labels

The resulting training database used to train an ML model
(3) Train end model w/ training DB

Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

**Key question:** How do we communicate the lineage (quality) of the training labels?
Highlight: Scaling with *unlabeled* data

Takeaway: Add more *unlabeled* data---without changing the LFs---and get better end performance!
Three Key Training Data Operations

Transforming data

Labeling functions (LFs)
Provide labels to unlabeled examples using domain heuristics

Transformation Functions (TFs)
Augment data with per-example transformations (e.g. flips, crops for images)

Partition the data, specifying where the model should add more capacity
One Critical Tool: Data Augmentation

Ex: 13.4 pt. avg. accuracy gain from data augmentation across top ten CIFAR-100 models
### SuperGLUE Transformation Function (TF)

```python
def tf_days_of_the_week(x):
    yield x
    for DAY in DAYS_OF_WEEK:
        yield replace_with_synonym(x, word=DAY, synonyms=DAYS_OF_WEEK)
```

**id**: x1

Sentence 1: Can I **invite** you for dinner on **Sunday** night?
Sentence 2: The organizers **invite** submissions of papers.

```latex
\text{tf_days_of_the_week}(x1) \quad \rightarrow \quad \begin{aligned}
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Sunday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Monday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Tuesday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Wednesday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Thursday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Friday** night?} \\
\text{Sentence 1: Can I } & \text{**invite** you for dinner on **Saturday** night?}
\end{aligned}
```
Problem: Data augmentation is critical, but hard to hand-tune

Idea: Users provide transformations which we automatically tune and compose
Automatic Data Augmentation from User-Specified Invariances

Users write transformation functions (TFs) to the system. The system learns a generative model to tune and compose the TFs. The learned data augmentation policy is used for training the end model.
Three Key Training Data Operations

- **Labeling functions (LFs)**: Provide labels to unlabeled examples using domain heuristics.
- **Transformation Functions (TFs)**: Augment data with per-example transformations (e.g., flips, crops).
- **Partitioning data**: Partition the data, specifying where the model should add more capacity.
Slicing Functions (SFs) specify where the model should add more capacity.

- Without Slices: 52.94 F1
- With Slices: 91.18 F1

Slices of interest:
- S1
- S2

Heuristic (noisy) SFs:
- $\lambda_1$
- $\lambda_2$
Slicing Functions (SFs)

- The model learns to predict which slices each data point belongs to.
- An **attention mechanism** learns how to combine the representations learned for each slice to make its final prediction.
SuperGLUE Slicing Function (SF)

```python
def sf_target_is_noun(x):
    if x.sentences[0].target.pos == NOUN and x.sentences[1].target.pos == NOUN
        return NOUN_SLICE
    else:
        return ABSTAIN
```

id: x1
Sentence 0: Can I invite you for dinner on Sunday night?
Sentence 1: The organizers invite submissions of papers.

```
sf_target_is_noun(x1) == ABSTAIN
```

id: x2
Sentence 0: He felt a stream of air.
Sentence 1: The hose ejected a stream of water.

```
sf_target_is_noun(x2) == NOUN_SLICE
```
Conclusion

• Key idea: Build MTL models by **programmatically building & modifying the training dataset**

• Three core operations to manipulate training data:
  • Labeling (LFs)
  • Transforming (TFs)
  • Partitioning / ”slicing” (SFs)

• Full code using Snorkel posted soon (by 6/24)!

[Snorkel.Stanford.edu](http://Snorkel.Stanford.edu)