Snorkel: Accelerating Machine Learning with Training Data Management

Alex Ratner
UW / Stanford

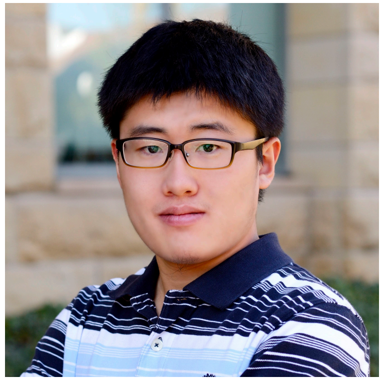
Snorkel Team @ Stanford


Braden Hancock


Ines Chami


Vincent Chen


Clara McCreery


Sen Wu

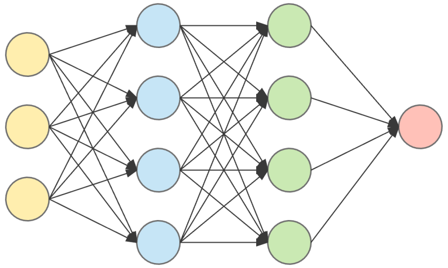

Chris Ré

And many more!

snorkel.org

ML Application =

Model + Data + Hardware




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

The line `import resnet as model` is struck through.

?

```
aws ec2 run-instances \
        --instance-type p3.2xlarge
        --instance-type p3.16xlarge
```

The line `--instance-type p3.2xlarge` is struck through.

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

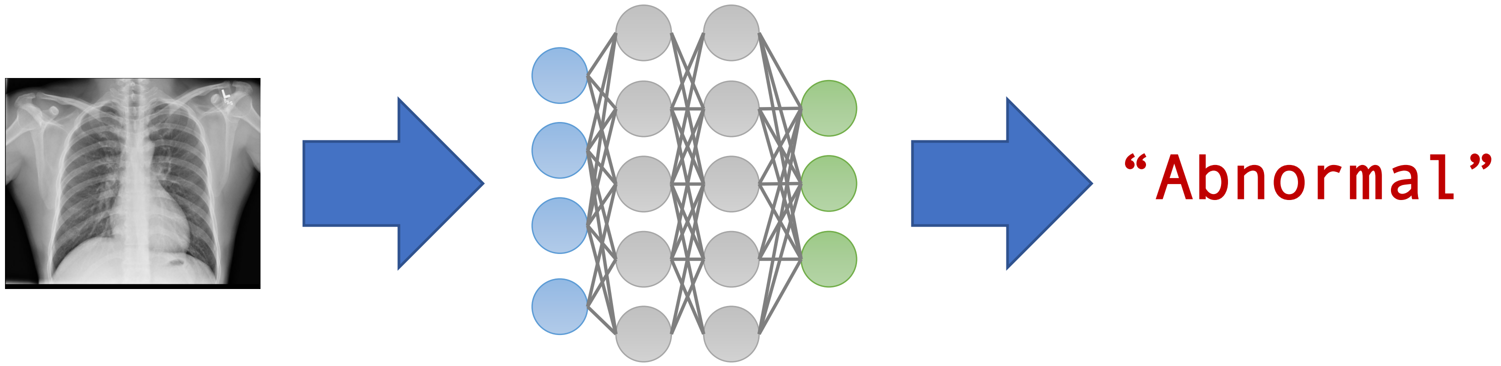
Training data is not

Training data is the key ingredient in ML

But it's created and managed in *manual, ad hoc* ways

Stanford MEDICINE

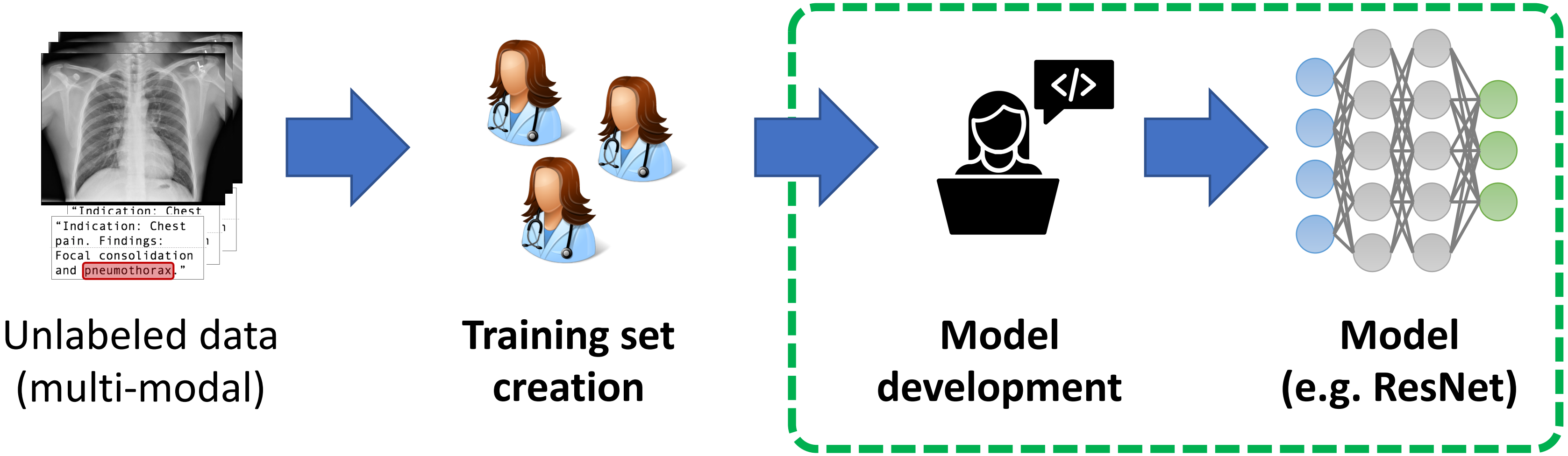
Example: Chest X-Ray Triage



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



2-3 days

$\pm$ **1 point** due to model choice

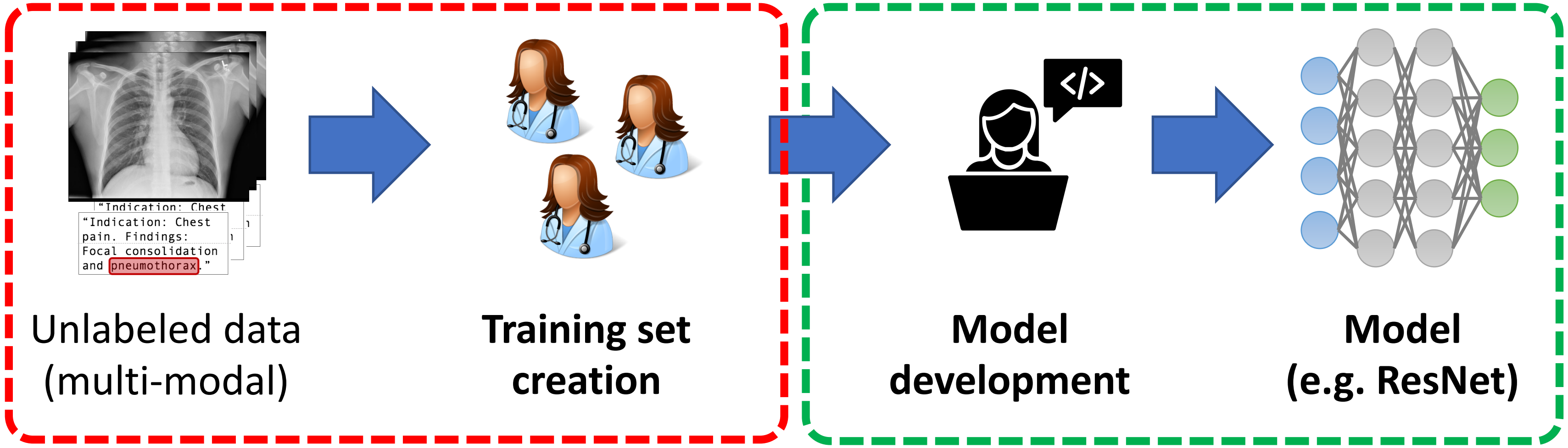
Model dev is often radically easier today!

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

(All scores: ROC AUC)

Stanford MEDICINE

Example: Chest X-Ray Triage



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

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

(All scores: ROC AUC)

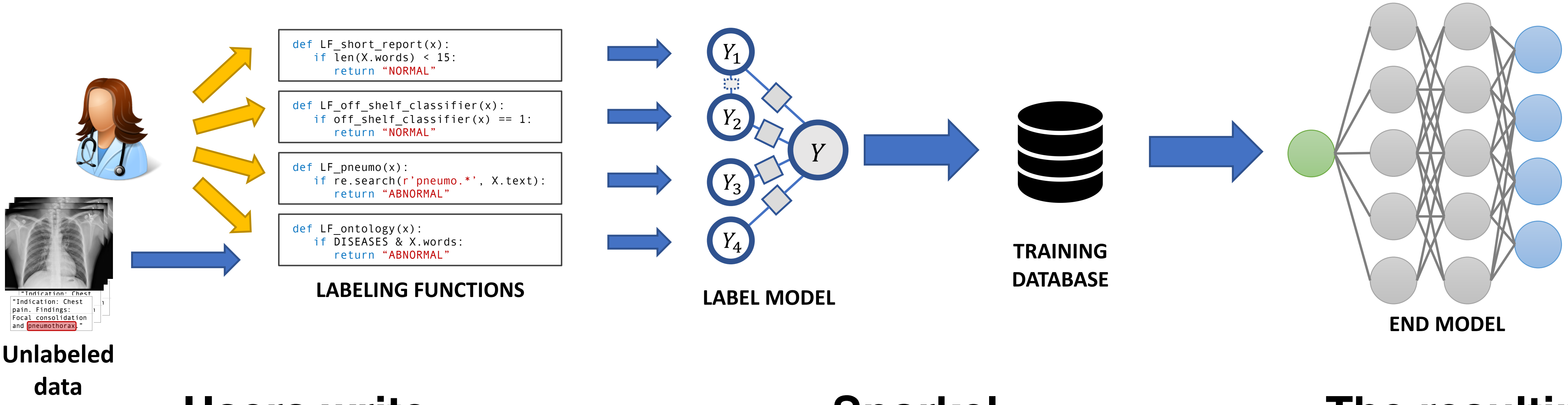
KEY IDEA:

Let users **build and manage training datasets** programmatically, then clean & integrate it for them

snorkel

The Snorkel Pipeline

snorkel.org



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

Gather Raw Data **Label Training Data** **Train Model**

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

[Bach et. al., SIGMOD Industry 2019]

Snorkel: Real-World Deployments

snorkel

snorkel.org

Greene Lab
Stanford MEDICINE
Stanford Center for Genomics and Personalized Medicine
Helix Group
Stanford University
Shah Lab
Stanford MEDICINE
Radiology
IBIIS
INTEGRATIVE BIOMEDICAL IMAGING INFORMATICS | STANFORD
VA | U.S. Department of Veterans Affairs

Science & Medicine

intel
Google
Alibaba.com
Microsoft
IBM
蚂蚁金服
ANT FINANCIAL
TOSHIBA
Chegg
HITACHI
vmware
BASF
We create chemistry
NEC
Infosys
accenture

Industry

FDA
DARPA MEMEX
NASA JPL
DARPA

Government

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 **L**anguage **U**nderstanding **E**valuation

snorkel

ML² NYU

Microsoft

salesforce

Alibaba.com

Google AI

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

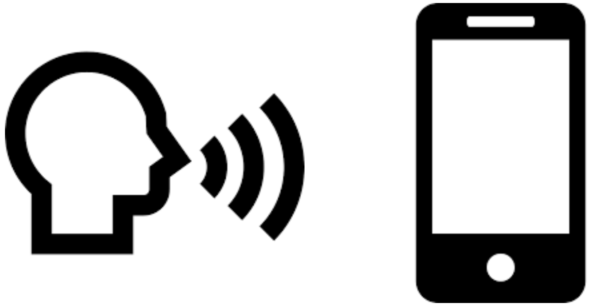
~1M total examples

SuperGLUE Example

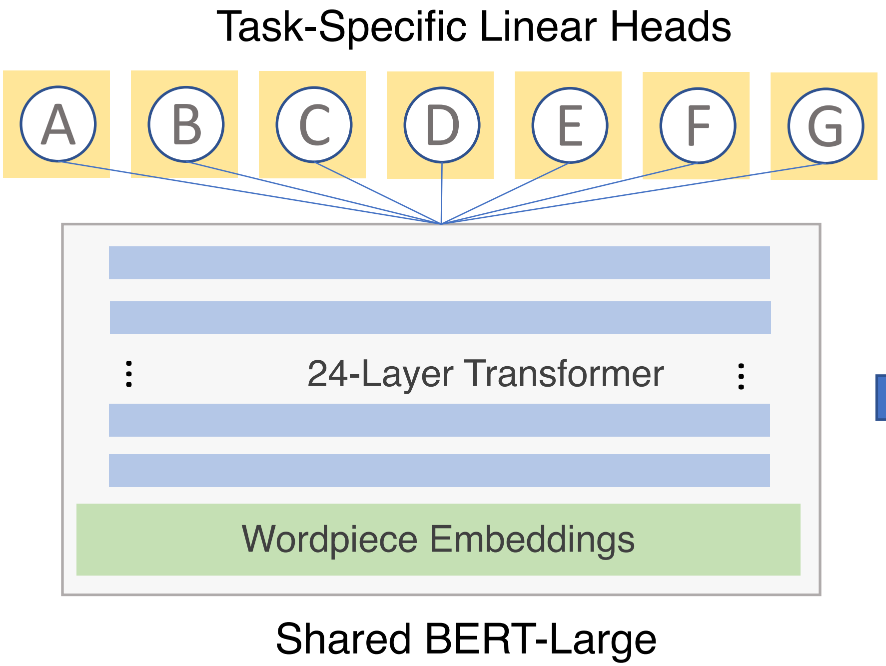
WiC task: Is the **target** word being used in the same way in both sentences?

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?



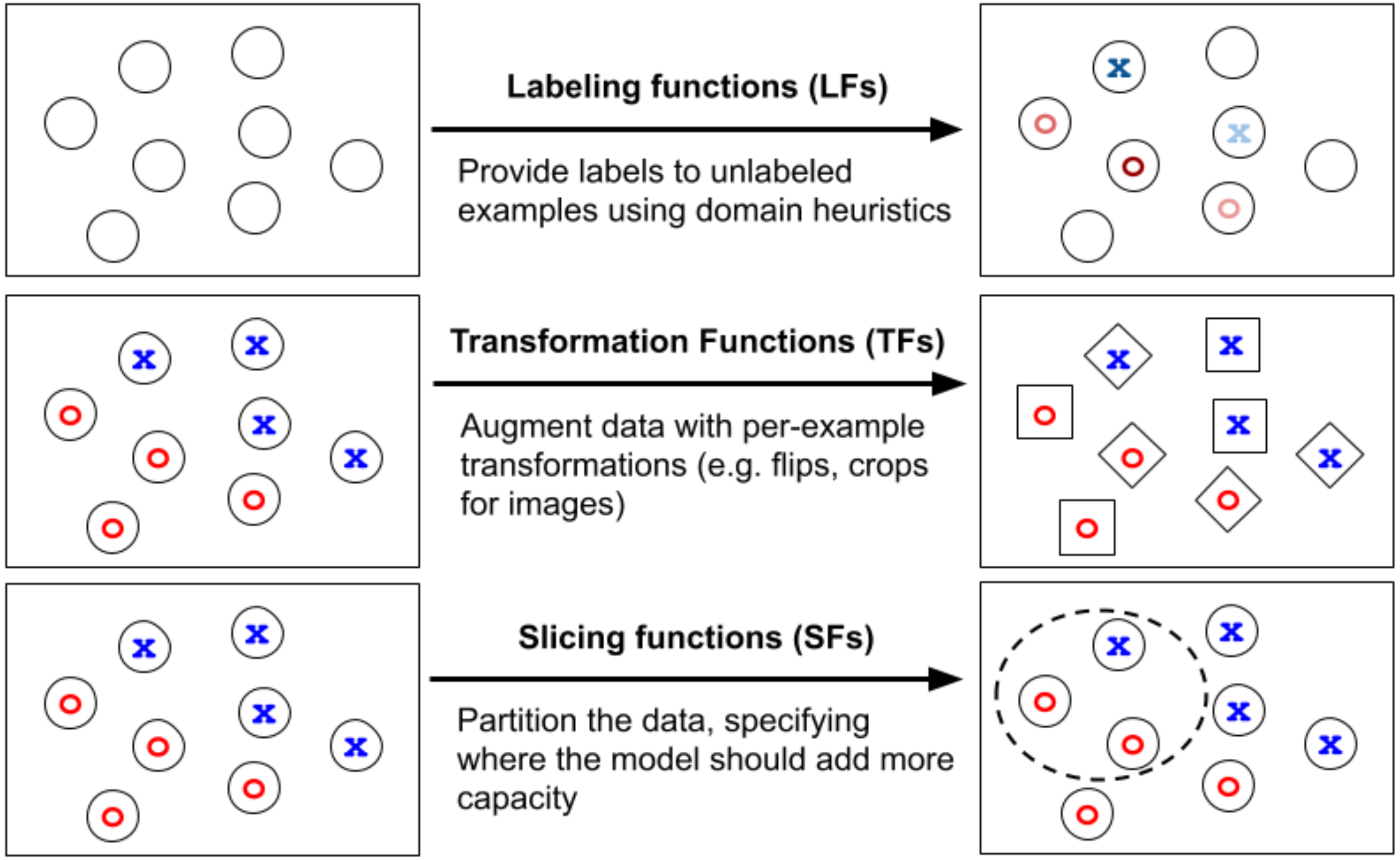
+

= SOTA?

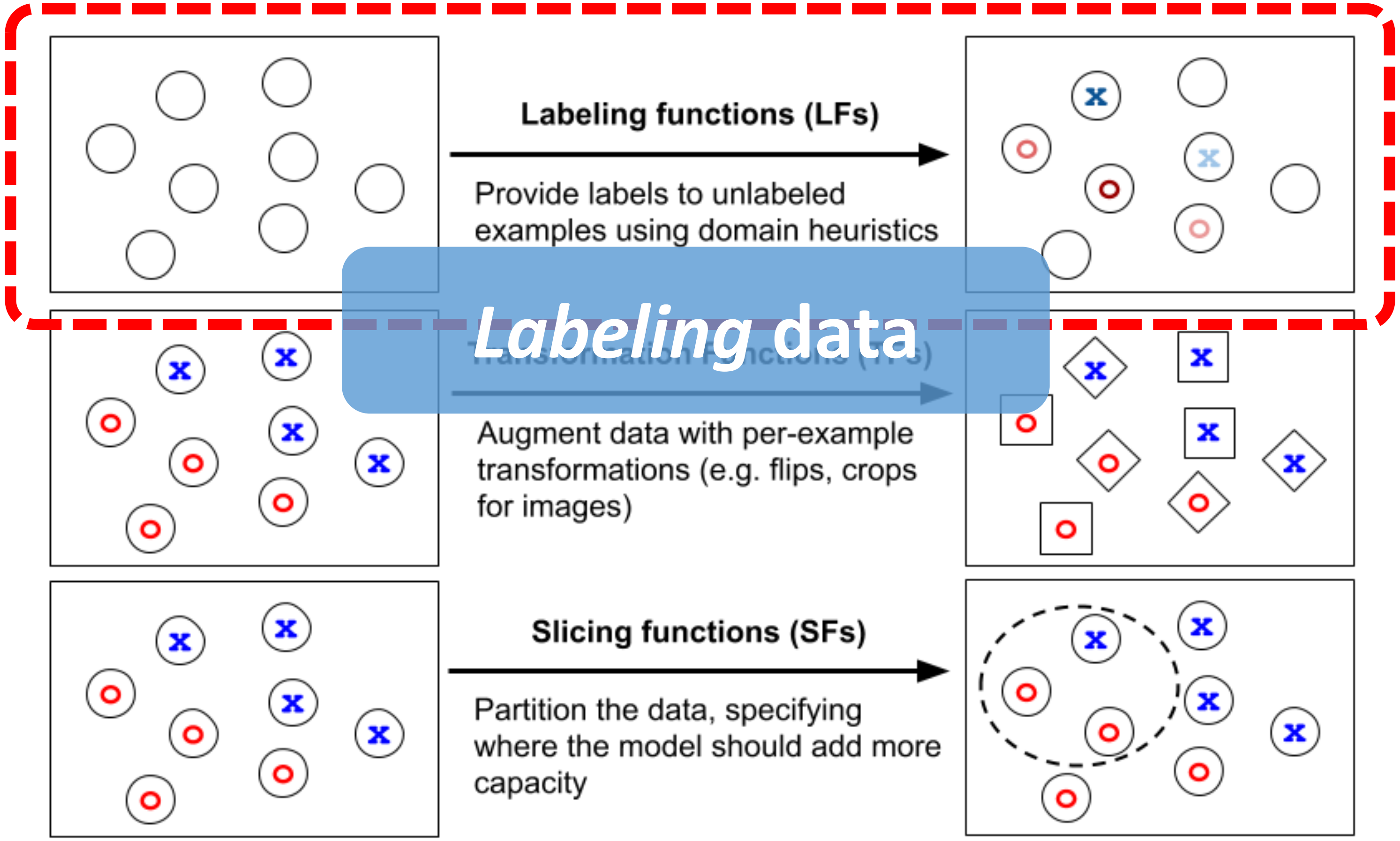
Rank	Name	Model	URL	Score
1	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.6
2	Stanford Hazy Research	Snorkel Metal		74.5
3	SuperGLUE Baselines	BERT++		70.5
		BERT		68.0
		CBOW		48.6
		Most Frequent Class		46.9
		Outside Best		-

New SOTA score!

Three Key Training Data Operations



Three Key Training Data Operations



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
```

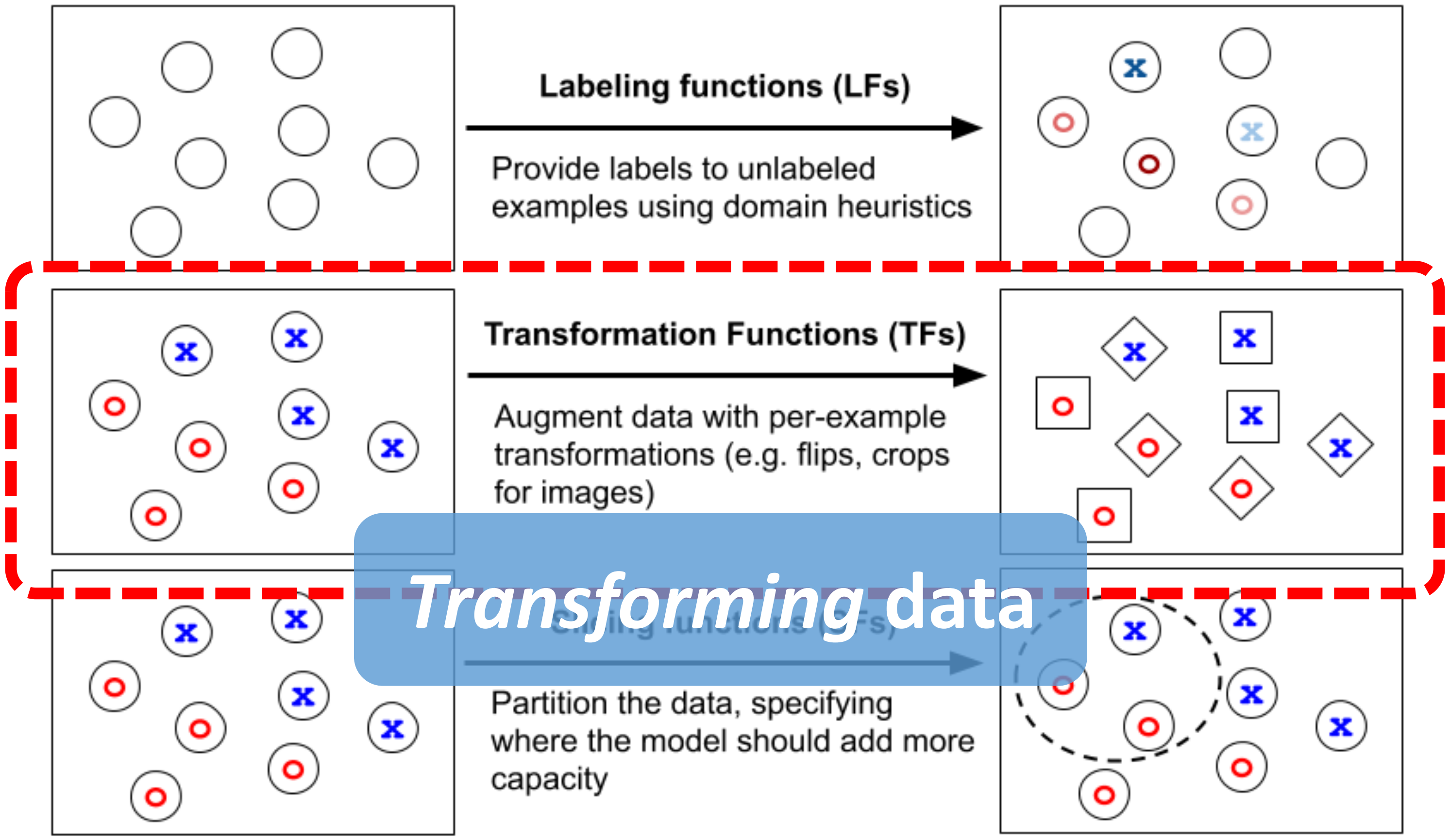
id: x1
Sentence 0: Can <u>I **invite** you</u> for dinner on Sunday night?
Sentence 1: The <u>organizers **invite** submissions</u> of papers.
Label: FALSE

`lf_matching_trigrams(x1) == ABSTAIN`

id: x2
Sentence 0: He felt <u>a **stream** of</u> air .
Sentence 1: The hose ejected <u>a **stream** of</u> water .
Label: TRUE

`lf_matching_trigrams(x2) == TRUE`

Three Key Training Data Operations



SuperGLUE Transformation Function (TF)

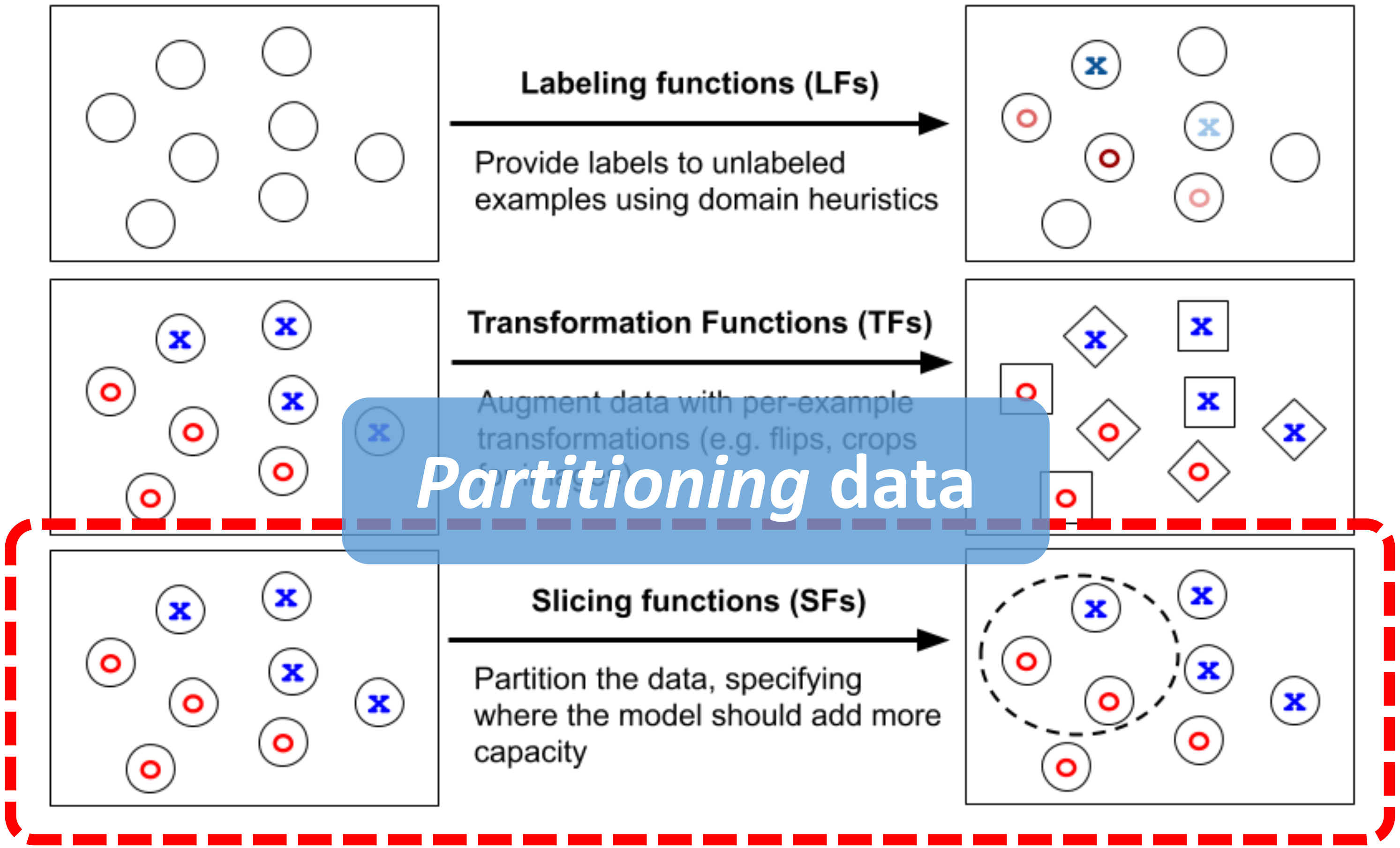
```
def tf_days_of_the_week(x):
    yield x
    for DAY in DAYS_OF_WEEK:
        yield replace_with_synonym(x, word=DAY, synonyms=DAYS_OF_WEEK)
```

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

`tf_days_of_the_week(x1)` ⇒

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



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?
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 .

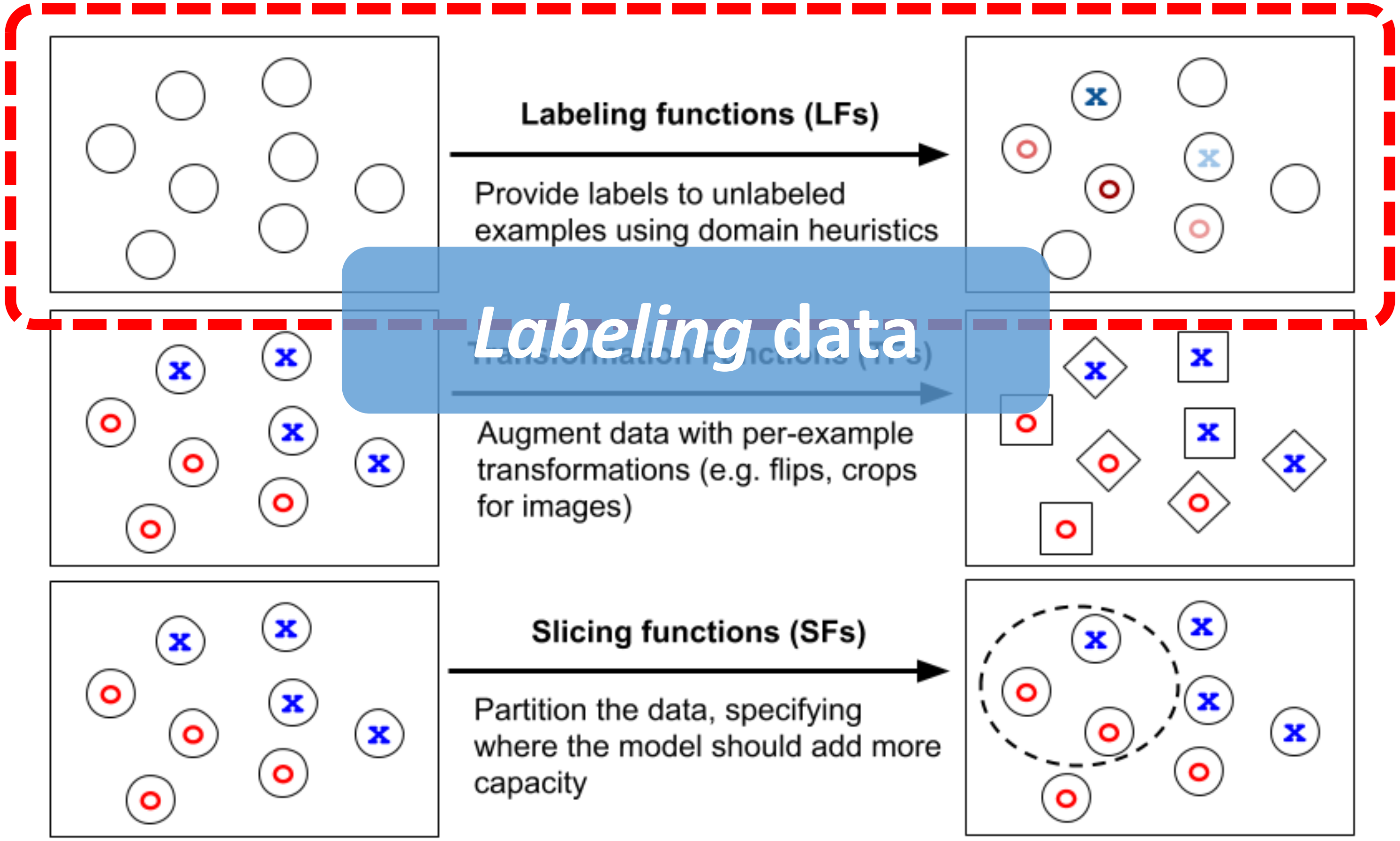
`sf_target_is_noun(x2) == NOUN_SLICE`

snorkel

Key Idea: Let users spend their time **building and modifying the training data**

snorkel.org

Three Key Training Data Operations

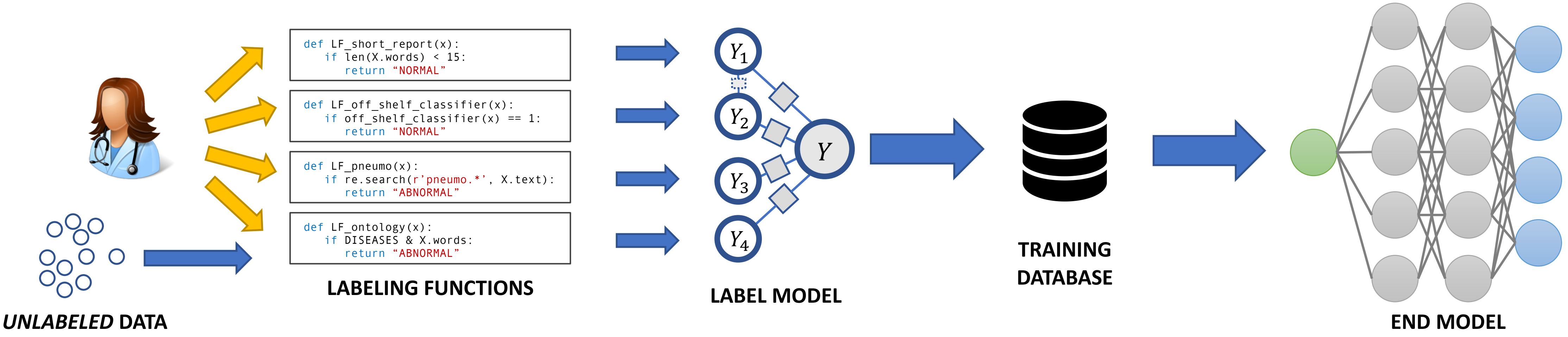


Problem: Hand-labeling is slow, expensive, & static

Idea: Enable users to label training data *programmatically*

The Snorkel Pipeline

snorkel.org



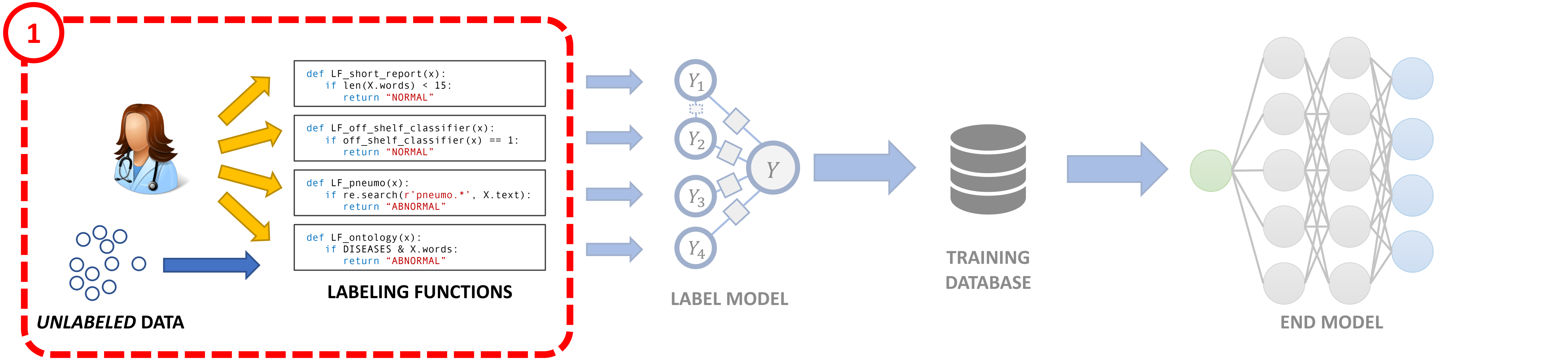
Users write *labeling functions* to heuristically label data

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

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
```

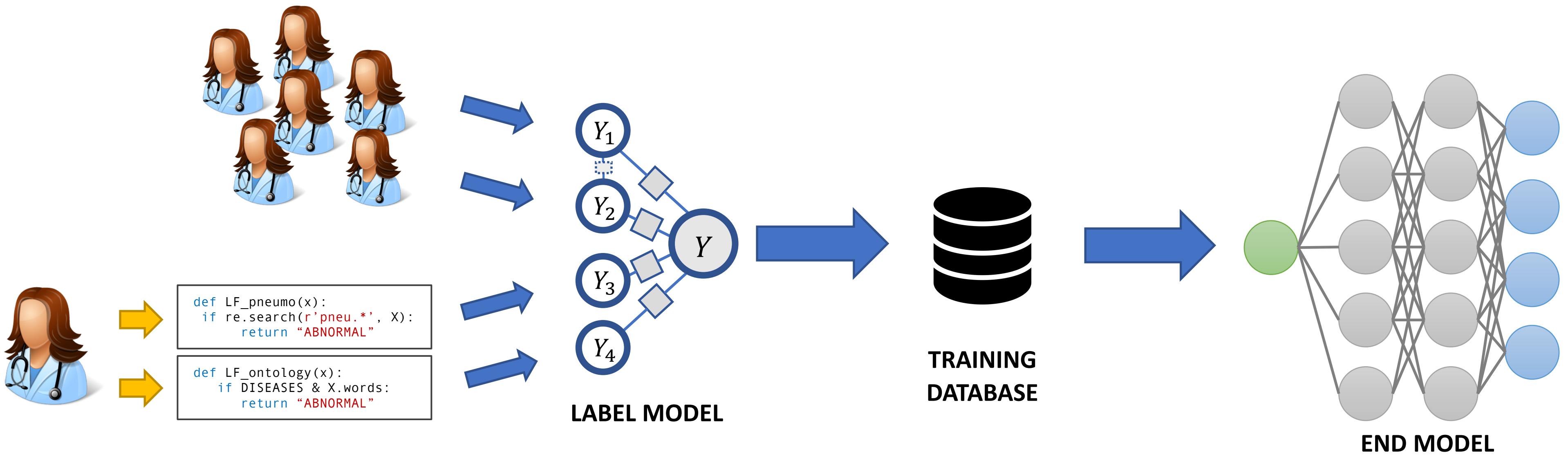
id: x1
Sentence 0: Can I **invite** you for dinner on Sunday night?
Sentence 1: The organizers **invite** submissions of papers.
Label: FALSE

`lf_matching_trigrams(x1) == ABSTAIN`

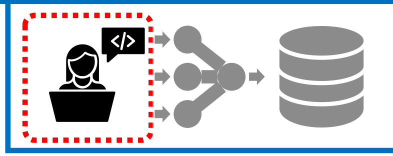
id: x2
Sentence 0: He felt a **stream** of air .
Sentence 1: The hose ejected a **stream** of water .
Label: TRUE

`lf_matching_trigrams(x2) == TRUE`

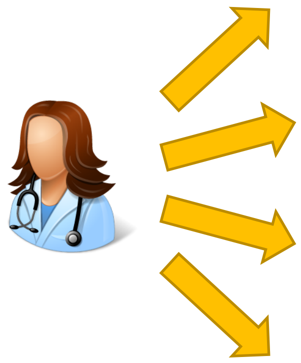
Hybrid Crowd + Programmatic Labeling



Snorkel as a management layer for human (e.g. internal crowd) + programmatic labeling



Result:
Supervision as Code



```
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”
```

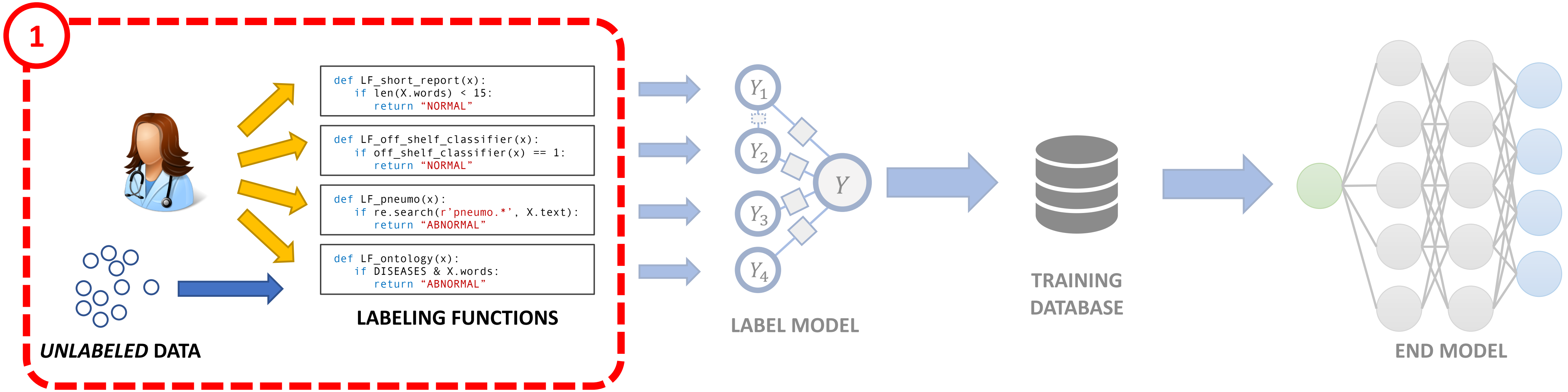
```
def LF_pneumo(x):
    if re.search(r’pneumo.*’, X.text):
        return “ABNORMAL”
```

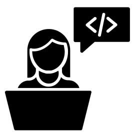
```
def LF_ontology(x):
    if DISEASES & X.words:
        return “ABNORMAL”
```

LABELING FUNCTIONS

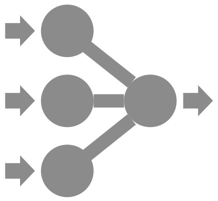
But, very messy supervision…

(1) Writing Labeling Functions





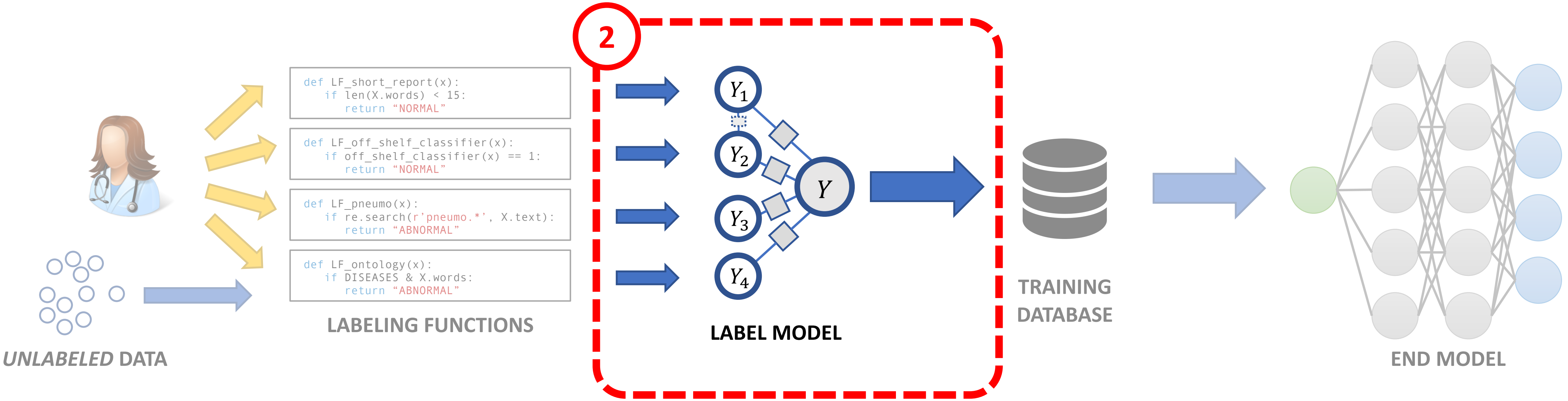
Users write *labeling functions* to heuristically label data



Snorkel ***cleans and combines*** the LF labels

The resulting training database used to train an ML model

(2) Clean & integrate noisy labels



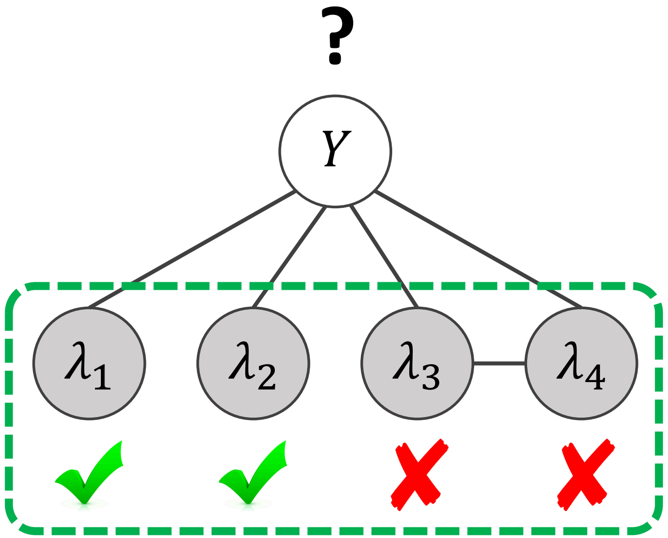
Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

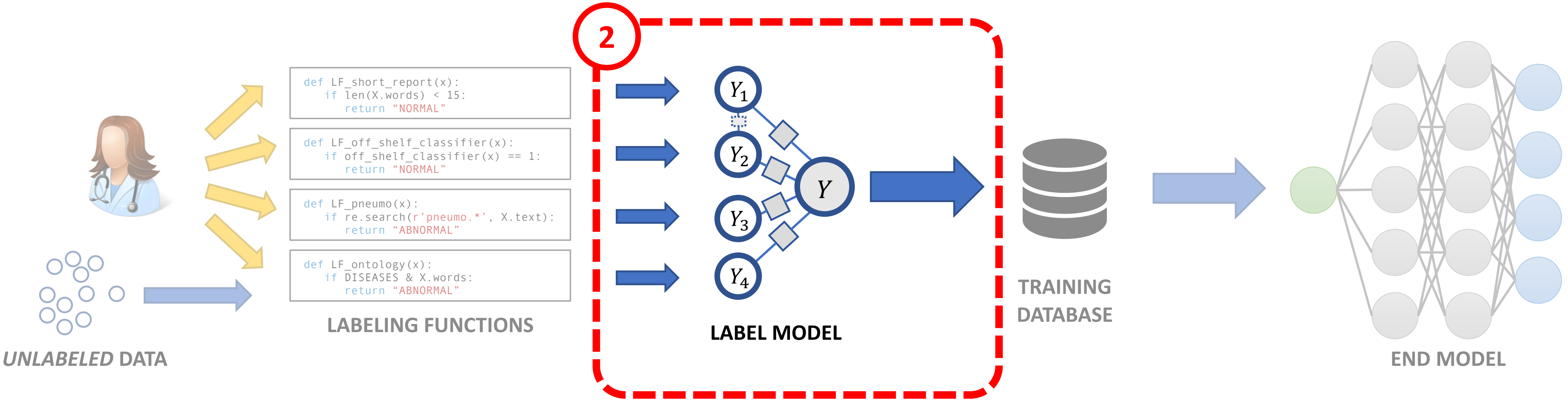
How can we do this without ground-truth labels?

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

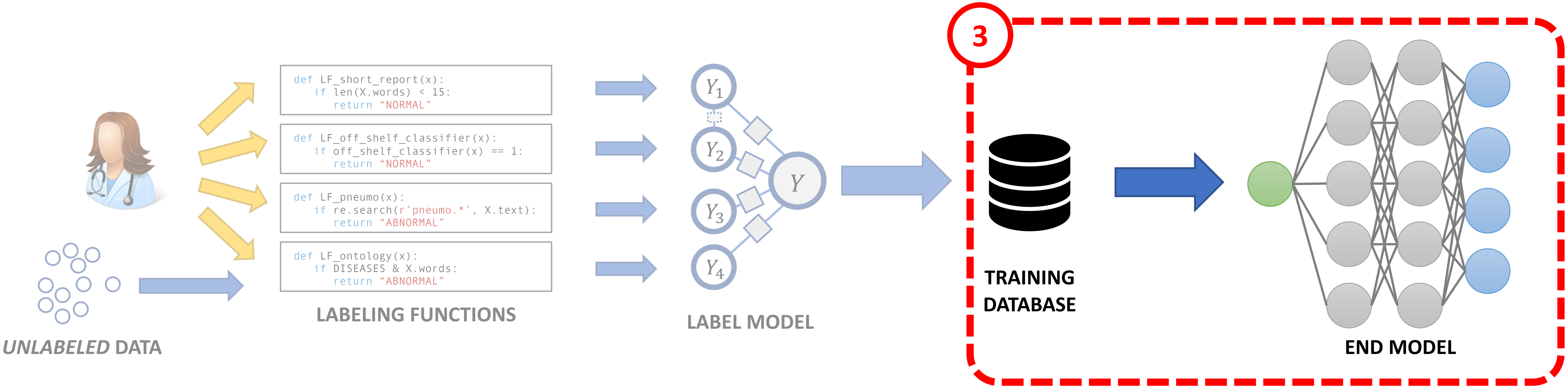


Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

(3) Train end model w/ training DB



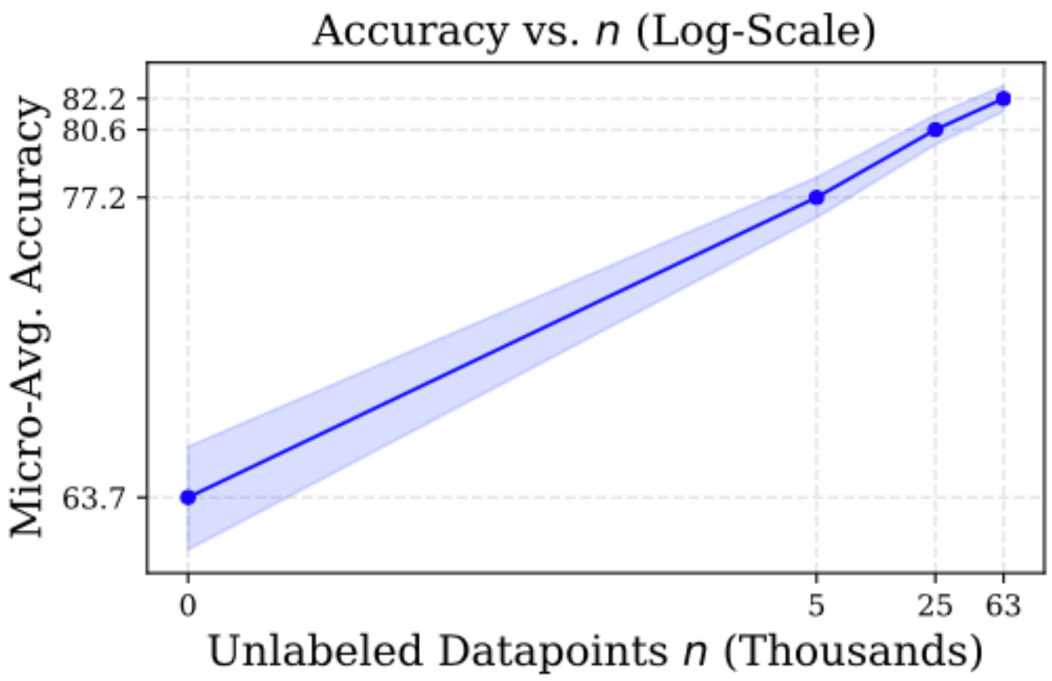
Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

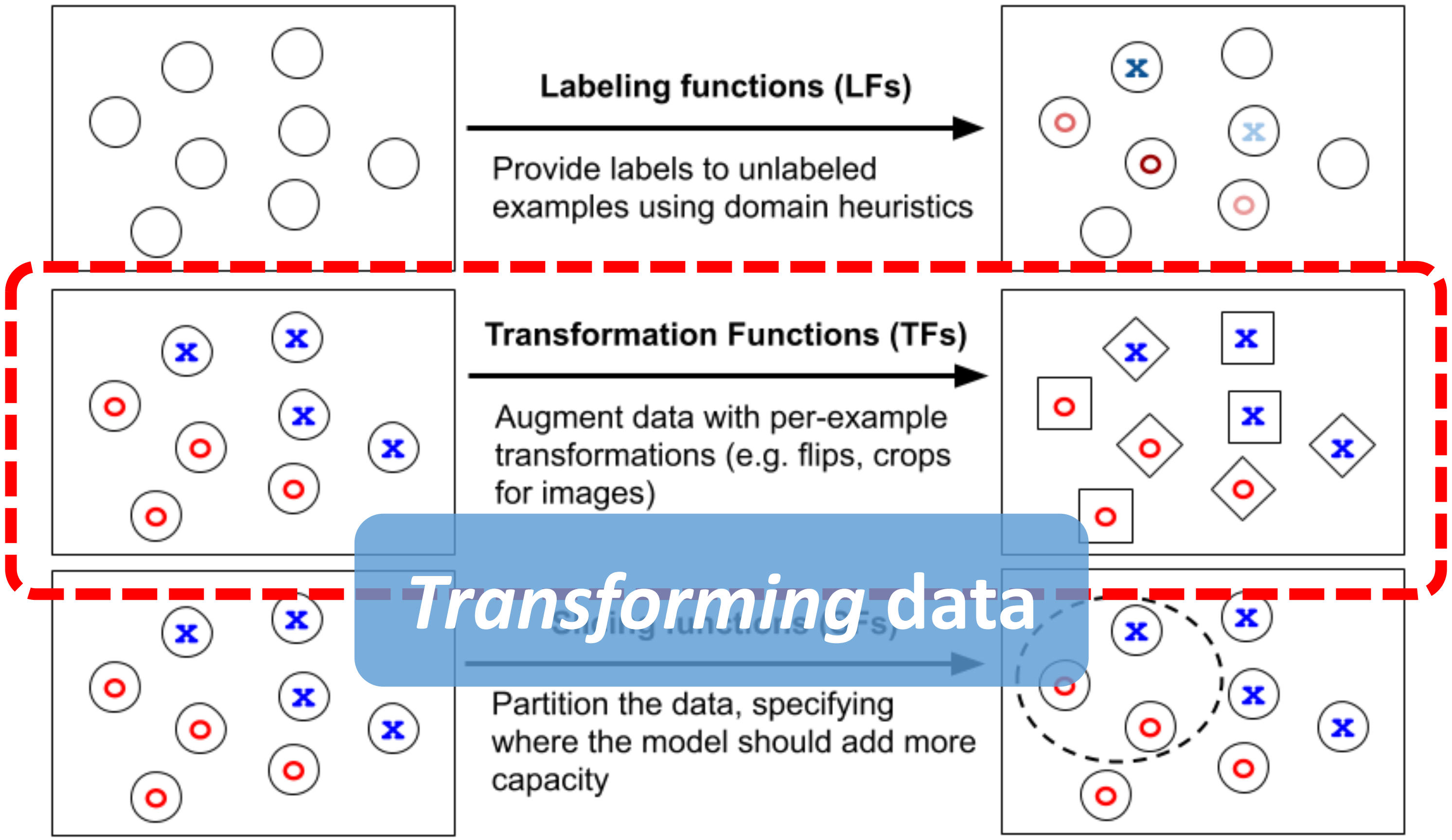
Key question: How do we communicate the lineage (quality) of the training labels?

Highlight: Scaling with *unlabeled* data

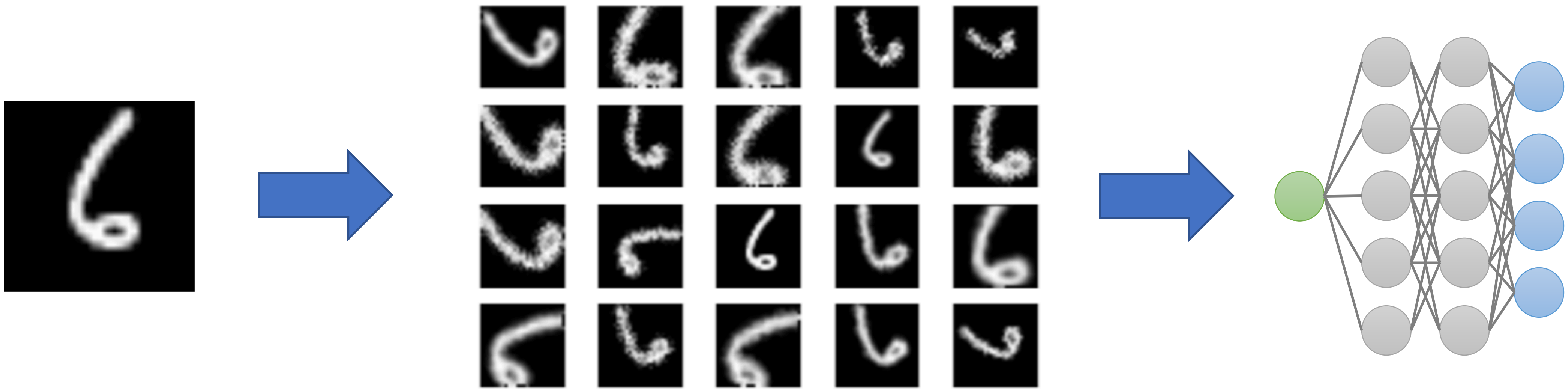


Takeaway: Add more *unlabeled* data---without changing the LFs---and get better end performance!

Three Key Training Data Operations



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)

```
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.

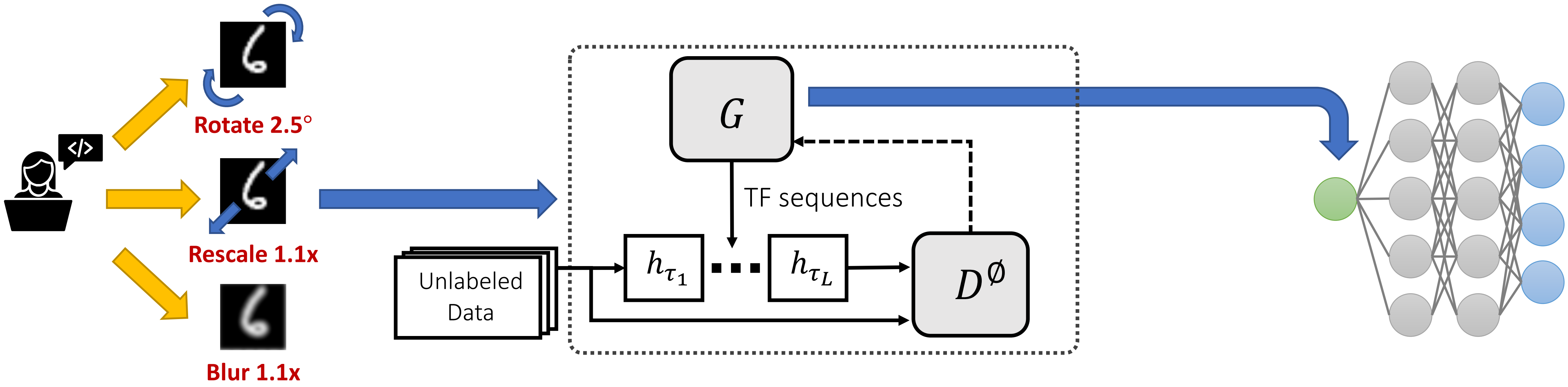
`tf_days_of_the_week(x1)` ⇒

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?

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

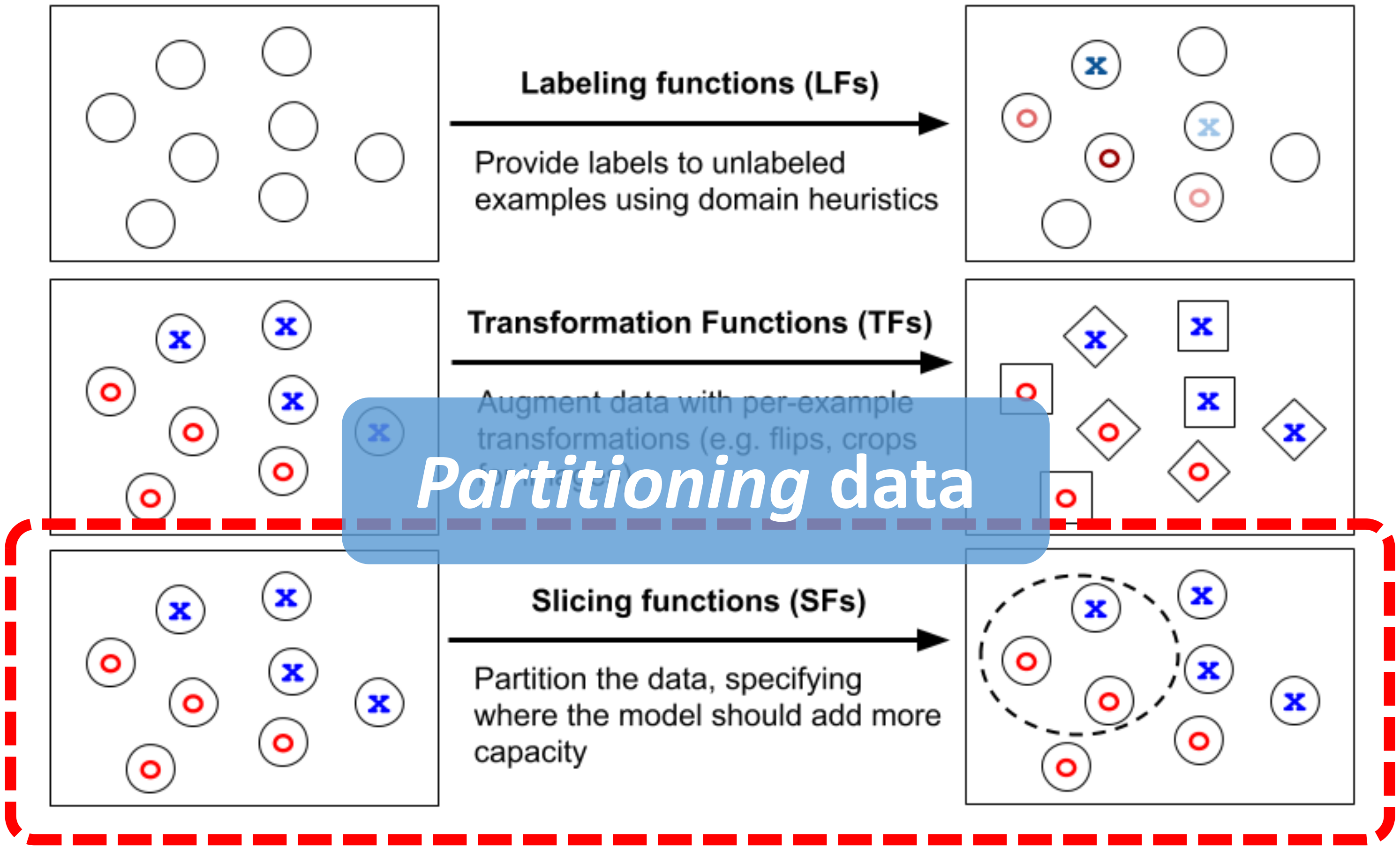


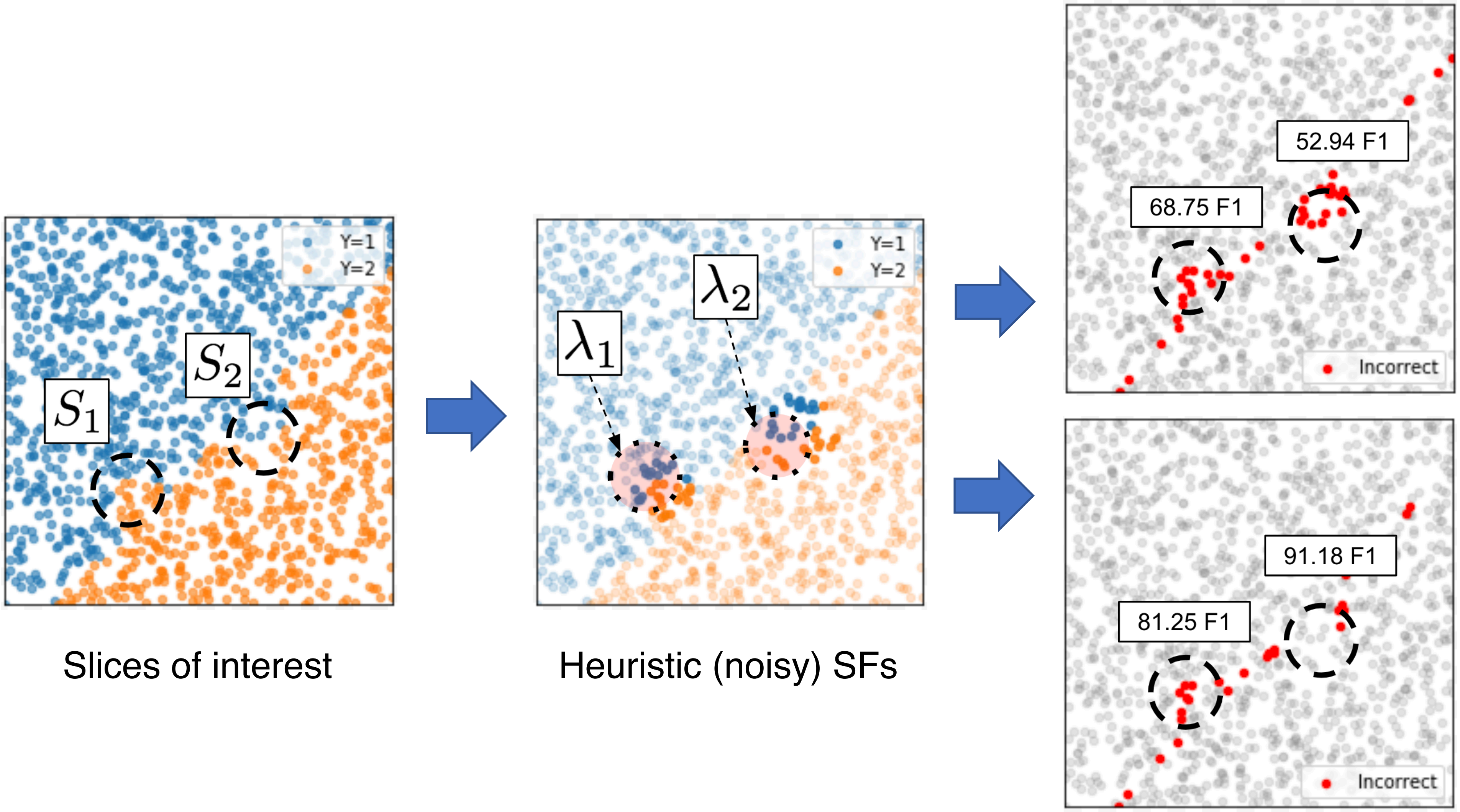
Users write ***transformation functions (TFs)***

We **learn a generative model** to tune & compose the TFs

The **learned data augmentation policy** used for training the end model

Three Key Training Data Operations





Slices of interest

Heuristic (noisy) SFs

Without Slices

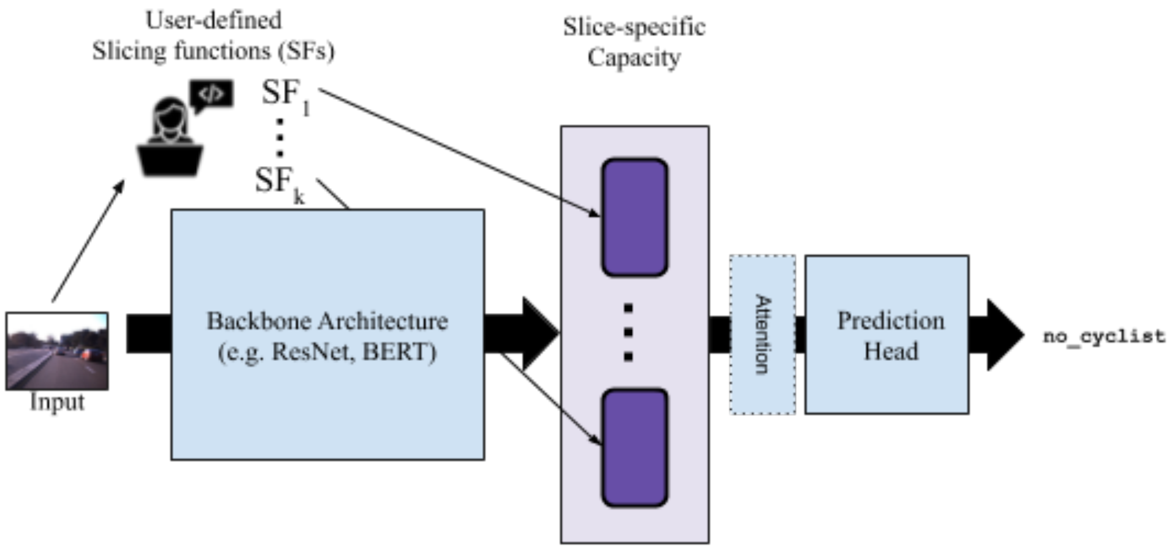
With Slices

+25 pts

Slicing Functions (SFs) specify where the model should add more capacity

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)

```
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