The Many Benefits of Weak Supervision

Asaf Valadarsky

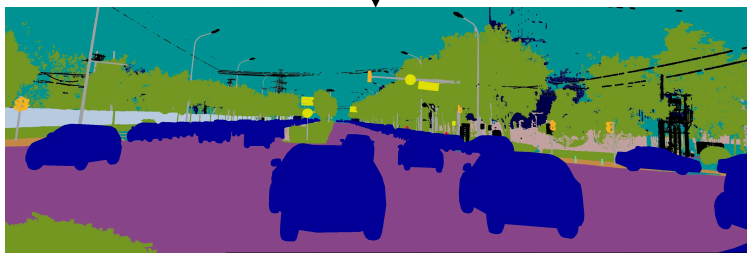
What we'll cover today

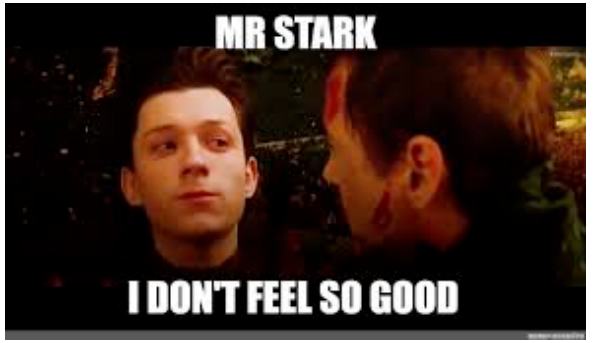
- A super-duper quick re-visitation of supervised learning
- How we (usually) deal with the cost of tagging?
- A new paradigm – data programming / weak supervision
- Why should you care?

Supervised learning (in a nutshell)

- A supervised learning task is defined by an object domain X and an output domain Y
  - Input –$\{(x_i, y_i)\}^N \in (X, Y)$
  - Output – a mapping $M: X \to Y$

Dog





Thanos

What do we learn in SL?

- Thinking in probabilistic terms, we are trying to learn $P(y|x)$
- We could learn a good enough mapping if we have "enough" data
  - But <u>how much</u> is enough?
- MNIST: $|\mathrm{X}_{\mathbf{train}}| = 60k$
  - ~99.8% accuracy using 60k samples
  - ~92% accuracy with only 100 samples (+tricks)
  - Which is better?

The cost of supervision

- Creating tags is (usually) a VERY expensive procedure
  - Manual tagging by experts
  - Tag validation (and revalidation, and revalidation, …)
  - Possible multiple tags (e.g., cats and dogs in the same image)

- Luckily, there are a few others ML approaches we can take
  - Have a partial tagged dataset, and a large untagged one (semi-supervised)
  - Have a partial tagged dataset for some of the classes, and try and infer for a larger amount of classes (Zero/one/few shot learning)
  - They still required tagged data…

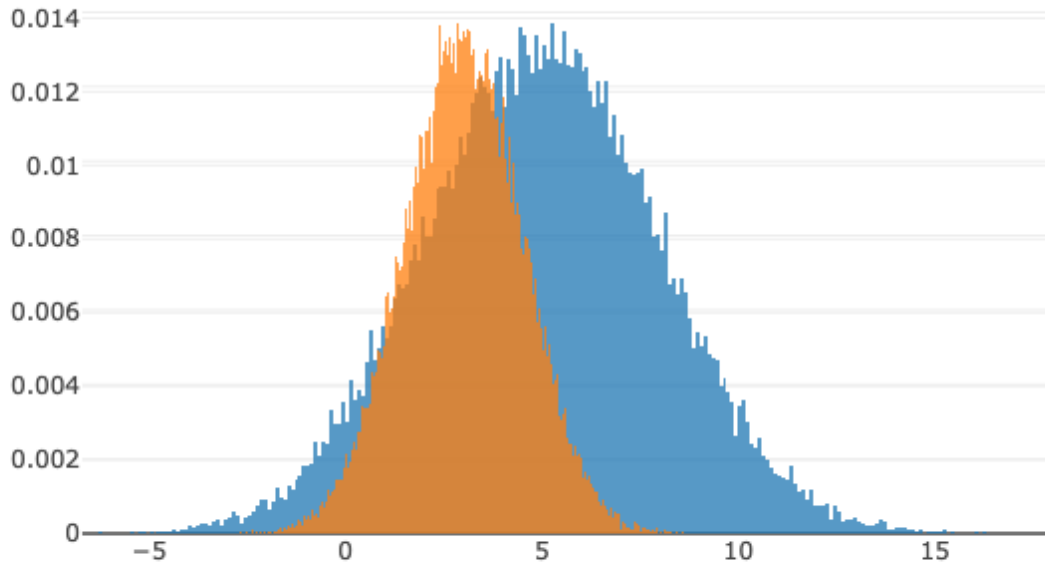
But what about self-supervision?

- Use a proxy task to create supervised signals
  - Hide words in a given sentence and learn a completion model
  - Rotate an image and learn its rotation
  - Select frames from videos and learn which came first/where

- But…
  - You won't always have such a proxy task
  - Might not learn what you hope to learn
  - Usually used to create representations for future supervised tasks

How can we get more labeled data?

- Get more experts to tag your data?
- Make structural assumptions?
- Use models trained on existing tasks?



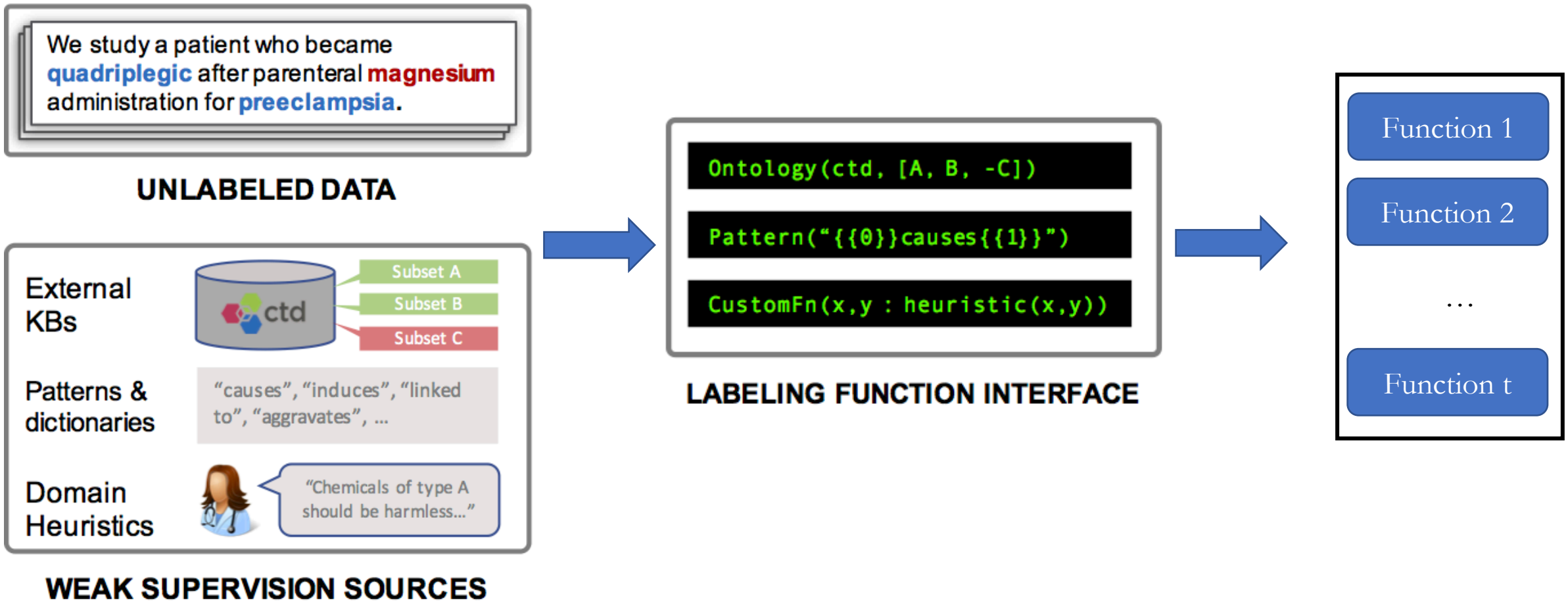
Data programming

External KBs

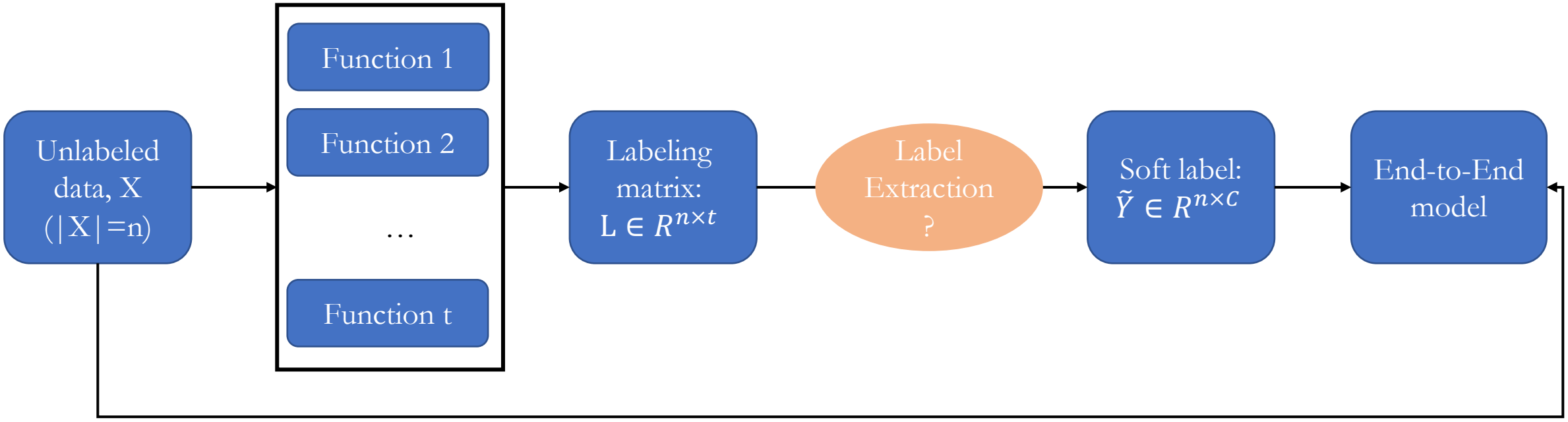
Patterns & Dictionaries

Domain Heuristics

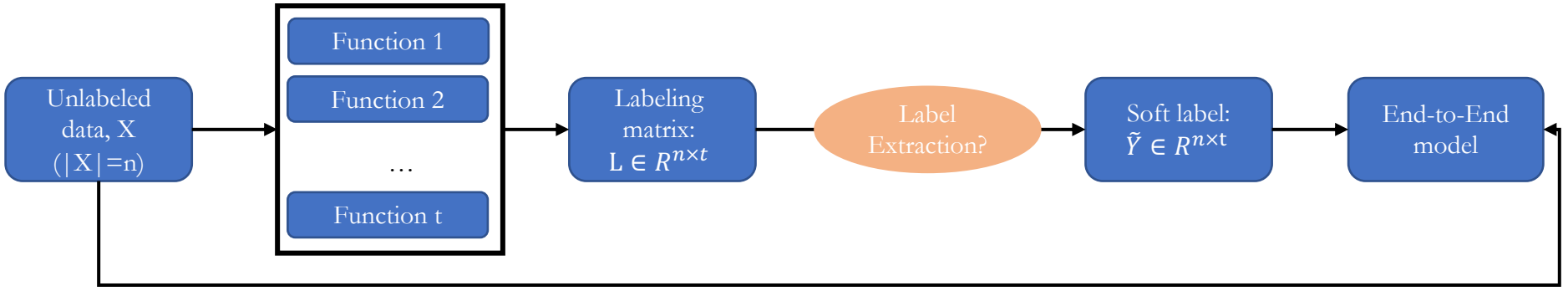
Data programming



Weak supervision (idea)



Weak supervision – label extraction



- Use a majority rule!
  - Could possibly lead to bias in label selection (think correlated functions)

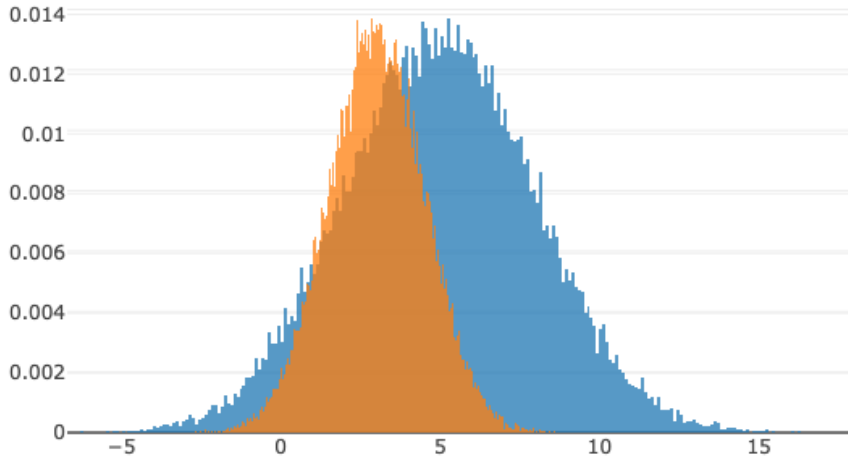
- Let's **learn** the importance of each function!
  - Assign a weight to each function
  - Will use a probabilistic model to answer how these labels might have been generated

Lets be practical for a second

- Q1: How many labeling functions do we need?
- Q2: How do we deal with dependencies between labeling functions?
- Q3: Does this scale?
- Q4: Is this open source?
- Q5: How could I measure my generated models?

Why should we care?

- Sample one distribution (orange/blue) at random
  - From each distribution sample 5 numbers
  - Resulted vector is a single sample
  - Toy dataset is 10K such vectors



- Using only 2 labeling functions (mean & std)
  - Majority: on these labels ~78% accuracy
  - Weak supervision: results with a ~87% accuracy
  - Fully supervised (yet simple) logistic regression ~92% accuracy

Generative vs. Discriminative

- **<u>Reminder:</u>** A discriminative model learns P(Y | X)
- In a **generative** model we learn the joint distribution P(X,Y)
- *Example:* Suppose our data is (1,0), (1,0), (2,0), (2, 1)

	Y=0	Y=1
X=1		
X=2		

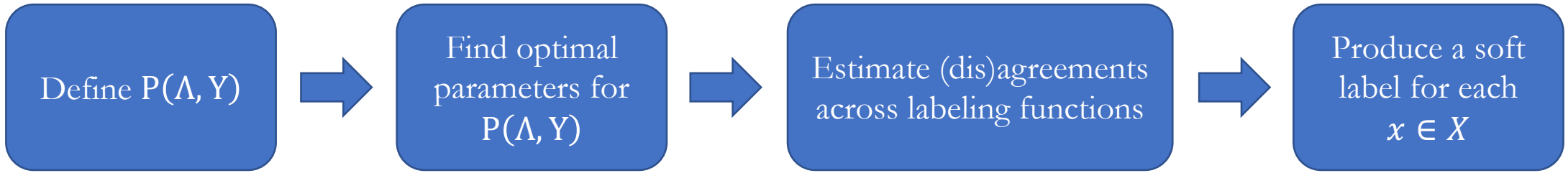
Generative Model

	Y=0	Y=1
X=1	1	0
X=2	1/2	1/2

Discriminative Model

Generative vs. Discriminative

- In our case we don't know the label…
  - Solution: "brute force" your way through!



Generative process for logistic regression

- Given a distribution π over $(x, y) \in \mathrm{X} \times \{-1,1\}$, minimize the logistic loss under a linear model:

$$\mathrm{E}_{(x,y)\in\pi}[\ln(1 + \exp(-w^T f(x)y))]$$

- The user specifies a collection of $\mathbf{m}$ labeling functions.
  - A labeling function is a mapping $\lambda_i : \mathrm{X} \to \{-1,0,1\}$
  - Assume each function has probability β_i to label an object, and α_i probability of being correct

Generative process for logistic regression

- The user specifies a collection of $\mathbf{m}$ labeling functions.
  - A labeling function is a mapping $\lambda_i : \mathrm{X} \to \{-1,0,1\}$
  - Assume each function has probability β_i to label an object, and α_i probability of being correct

$$\mu_{(\alpha,\beta)}(\Lambda, \mathrm{Y}) = \mathrm{P}(\mathrm{Y}) \cdot P(\Lambda|Y) = \frac{1}{2} P(\Lambda|Y = 1) + \frac{1}{2} P(\Lambda|Y = -1)$$

$$= \frac{1}{2} \prod_{i=1}^{m} \left(\beta_i \alpha_i 1_{\{\Lambda_i = Y\}} + \beta_i (1 - \alpha_i) 1_{\{\Lambda_i = -Y\}} + (1 - \beta_i) 1_{\{\Lambda_i = 0\}} \right)$$

Letting both α and β vary, is what transforms this into a family of generative models

Generative process for logistic regression

- Now we need to learn which (α, β) are most consistent with our observations

$$(\hat{\alpha}, \hat{\beta}) = \arg\max_{\alpha,\beta} \sum_{x \in S} \log P_{(\Lambda,Y)\sim\mu_{(\alpha,\beta)}} \left(\Lambda = \lambda(x)\right)$$

$$= \arg\max_{\alpha,\beta} \sum_{x \in S} \log \left(\sum_{y \in \{-1,1\}} \mu_{(\alpha,\beta)}(\lambda(x), y) \right)$$

The resulted parameters maximize the probability that the labels we produce on our training examples occur under our generative model

Wait… what just happened?

- The easy part:
  - Step 1 – define our functions (λ) and their evaluation/success rate (β, α)
  - Step 2 – allow β, α to be reconfigurable across some range
    - E.g., $0.3 \leq \beta_i, \leq 0.5$ and $0.8 \leq \alpha_i \leq 0.9$

- Step 3 – find the optimal values of β, α which best "describe" our dataset

- Now we can sample from the model!
  - Next up: how to optimize a linear model given this framework ☺

Linear model optimization for logistic regression

- For a single sample $x \in S$ we can compute its expected loss, for some linear model w, as:

$$\mathrm{L(x)} = E_{(\Lambda,Y)\sim\mu_{(\hat{\alpha},\hat{\beta})}}\left[\log\left(1 + e^{-w^T xY}\right)\middle|\Lambda = \lambda(x)\right]$$

- So our best linear model can be found by optimizing the (noise-aware) empirical risk:

$$\hat{w} = \arg\min_{w} \frac{1}{|S|}\sum_{x\in S} \mathrm{L(x)} + \rho\|w\|^2$$

Show me the money!

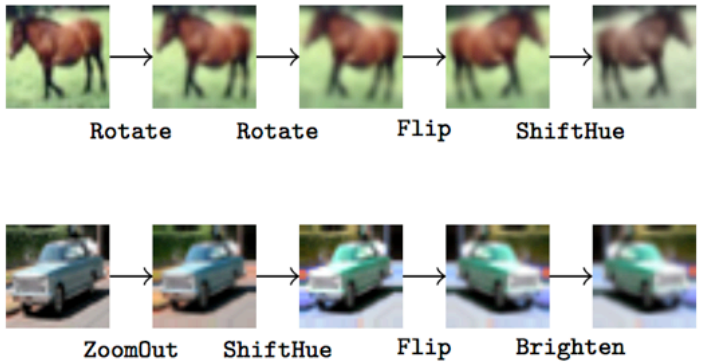
@Google scale

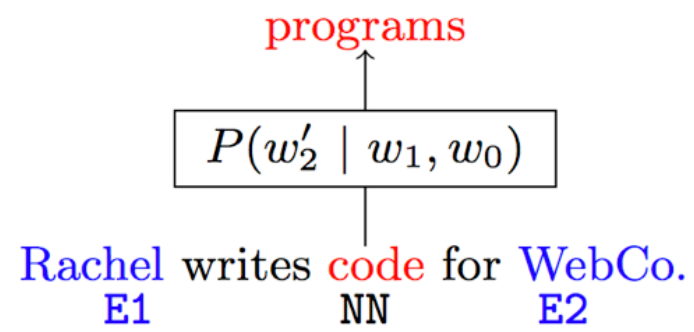
- Content topic classification
  - Using 10 labeling functions leveraging internal tools and domain knowledge
  - Baseline was a supervised approach on ~15K samples

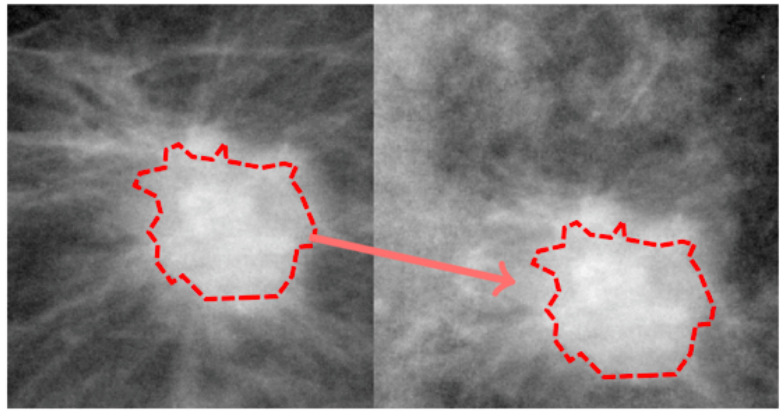
@Google scale – key takeaways

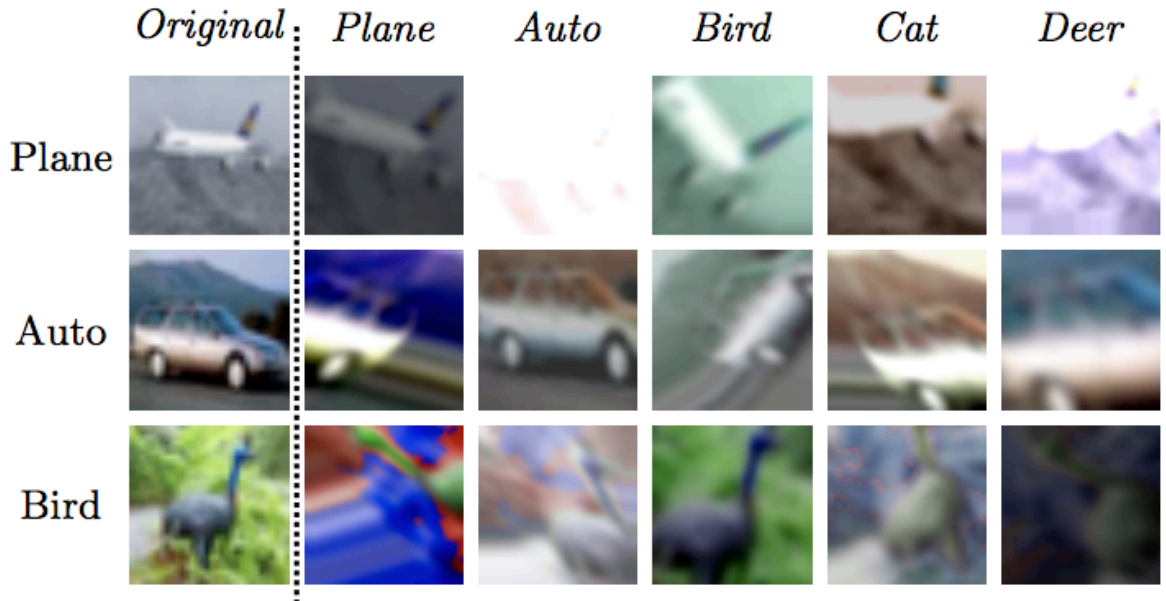
- Scale incurs additional “costs”
- Greatly improved ability to “re-label” existing data
- Can evaluate the signal from each source, allowing rapid development
- Allows for easy sharing across teams and domains

Data augmentation





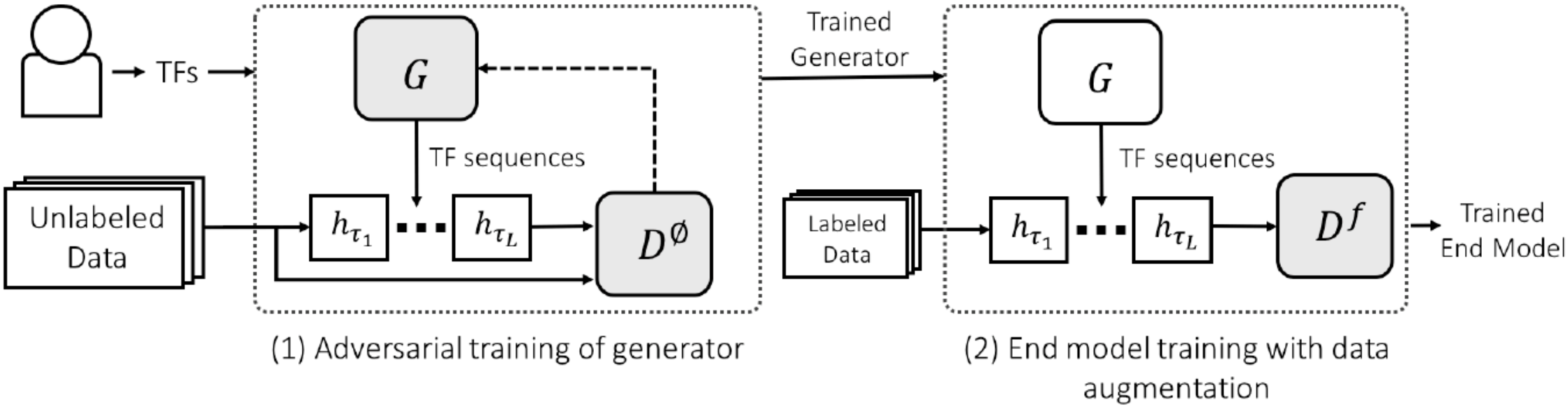




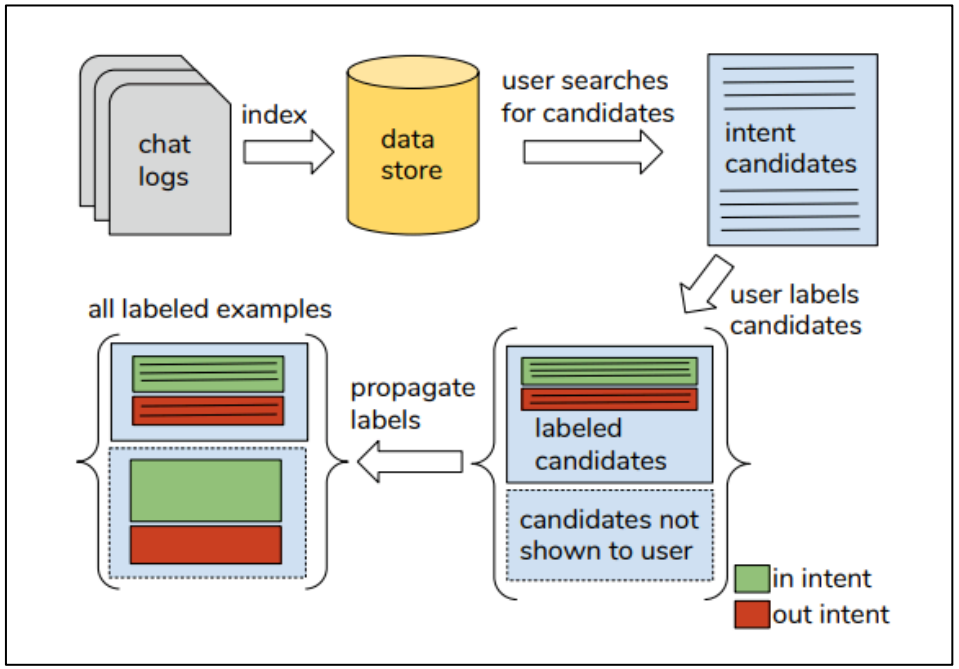
Data augmentation

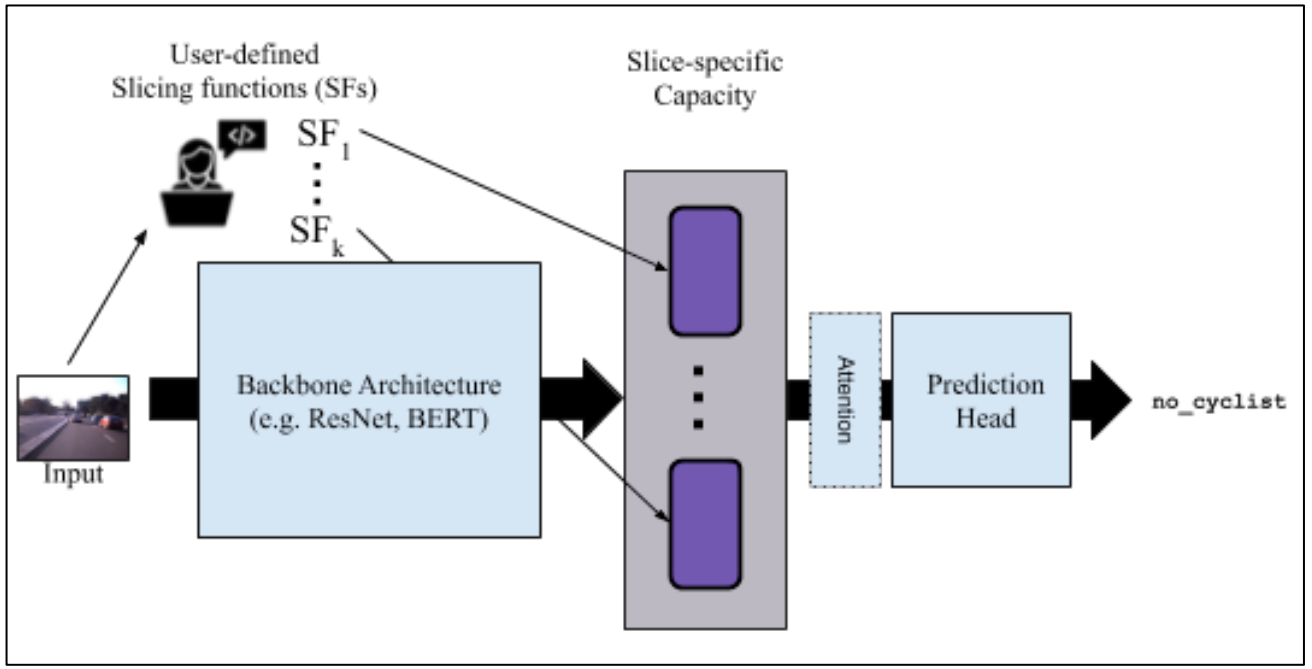
- Represent augmentation as a sequence of incremental "black-box" operations
  - Imaging: rotation, skew, brightness, scaling, contrast, …
  - Text: synonyms, replace entities, change order, …
  - Voice: add background noise, frequency masking, delay, …
- Learn a generative process over these sequences
- Use the generator in E2E pipeline

Data augmentation



Additional use-cases





Summary

- Weak supervision seems like a promising approach for multiple domains where tagged data is non-existing (or severely lacking)
  - And you have access to plenty(ish) amount of untagged data
- Can take multiple sources for heuristics (KBs, existing models, etc.)
- An awesome approach, but requires care:
  - Need to change basic classifiers (e.g., logistic regression) or basic training loop in DL cases
  - Needs adaptation at scale – work in progress by the original developers

Thanks!

Questions?

asaf.valadarsky@gmail.com

Papers

- Main Repo: https://hazyresearch.github.io/snorkel/
  - Original paper: https://arxiv.org/pdf/1605.07723.pdf
  - Data augmentations: https://arxiv.org/pdf/1709.01643.pdf
- @Google scale: https://arxiv.org/pdf/1812.00417.pdf
- Bootstrapping chatbots: https://arxiv.org/pdf/1812.06176.pdf
- Slicing: https://hazyresearch.github.io/snorkel/blog/superglue.html