Active Learning and Augmentation

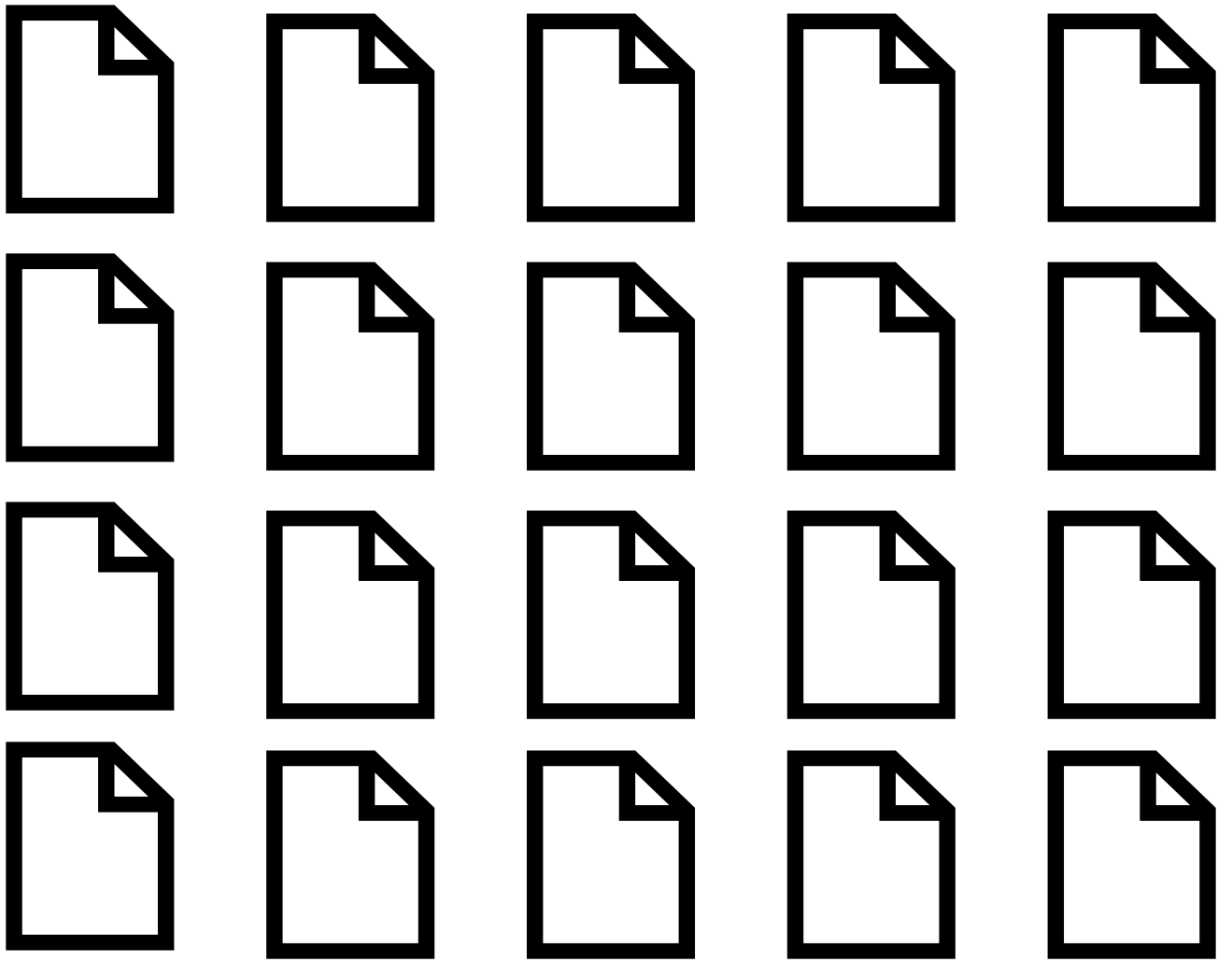
Overview

Machine Learning requires a lot of labeled data

There’s a lot of data

There’s not always very much *labeled* data

What can we do?



Active Learning

Active learning's approach:

We have a limited budget – we can only have so much labeled data

So which data do we choose to label?

If we label these documents how good will our model be?

Active Learning

Active learning's approach:

We have a limited budget – we can only have so much labeled data

So which data do we choose to label?

If we label these documents how good will out model be?

Will it be better if we label these ones instead?

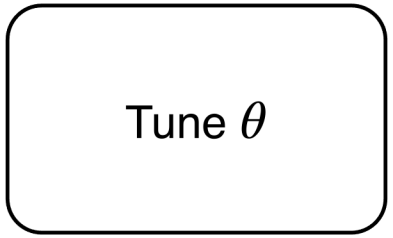
Pool-based AL

We have:

- Pool of unlabeled data *P*
- Model parameterized by θ
- A sorting heuristic *h*

Randomly select a set of seed data *L*

Train the model *L*



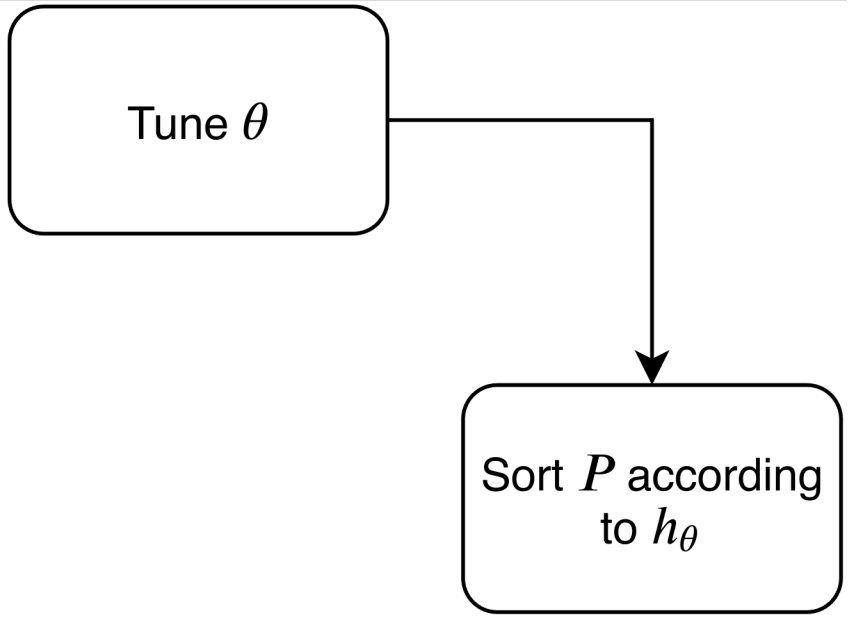
Pool-based AL

We have:

- Pool of unlabeled data *P*
- Model parameterized by θ
- A sorting heuristic *h*

Sort *P* using *h*

The most informative examples should be at the top

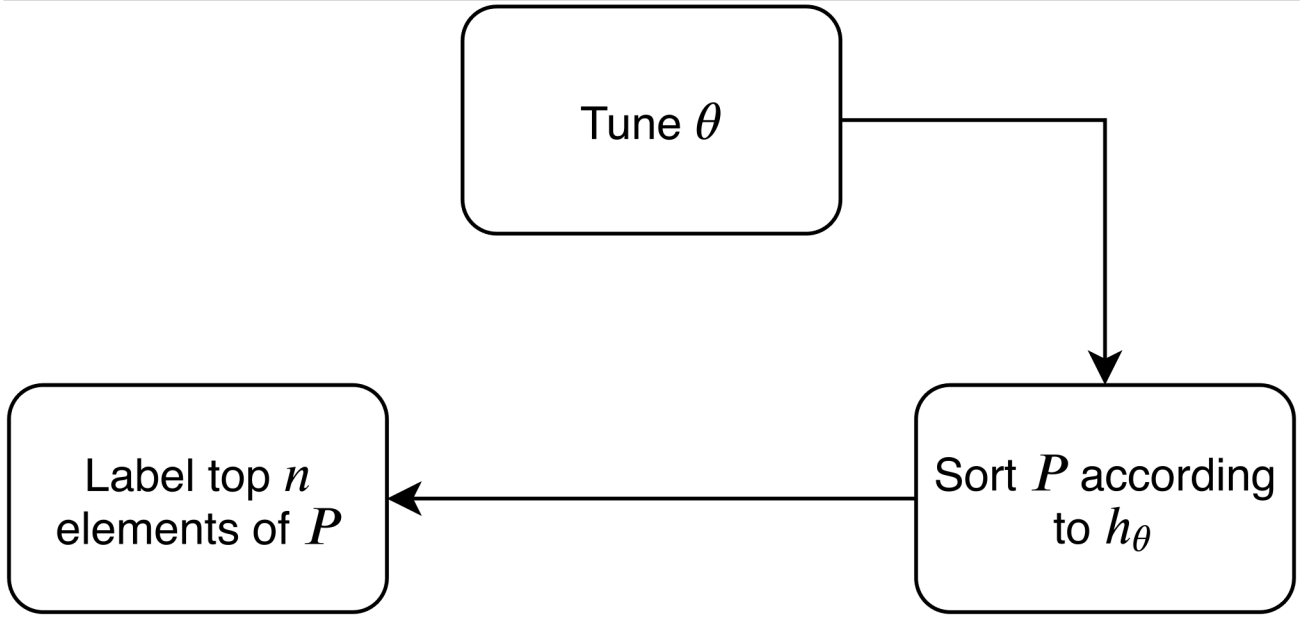


Pool-based AL

We have:

- Pool of unlabeled data *P*
- Model parameterized by θ
- A sorting heuristic *h*

Label the top *n* elements and add them to *L*



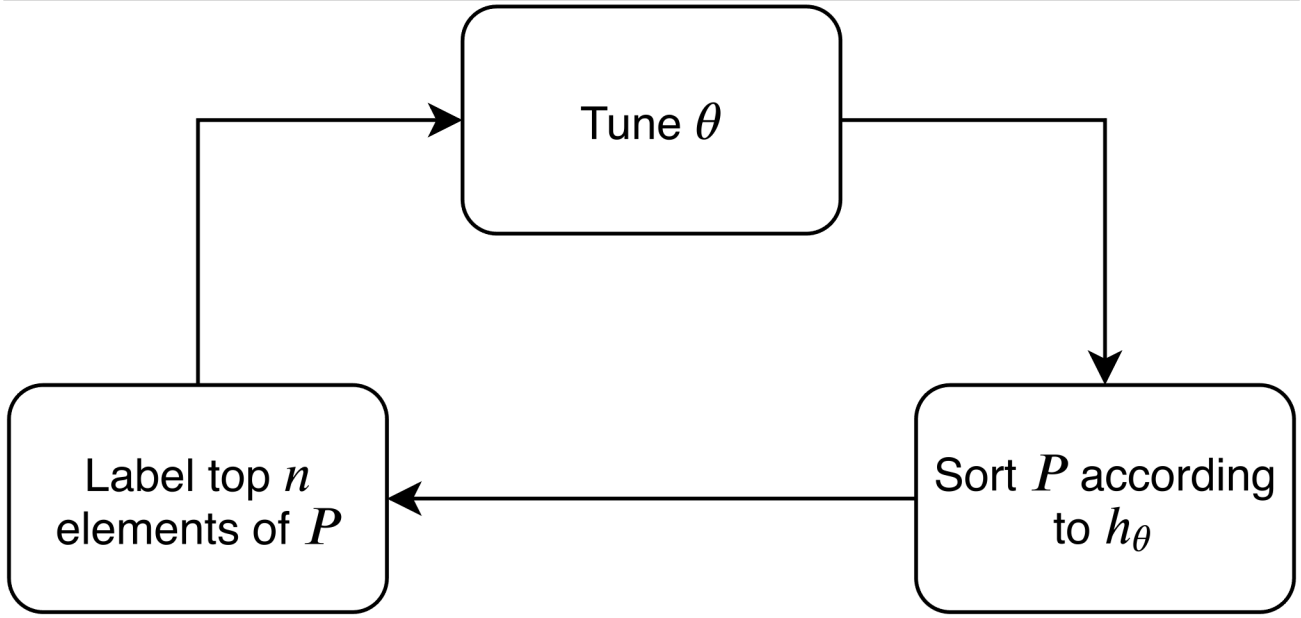
Pool-based AL

We have:

- Pool of unlabeled data *P*
- Model parameterized by θ
- A sorting heuristic *h*

Retrain on the expanded *L*

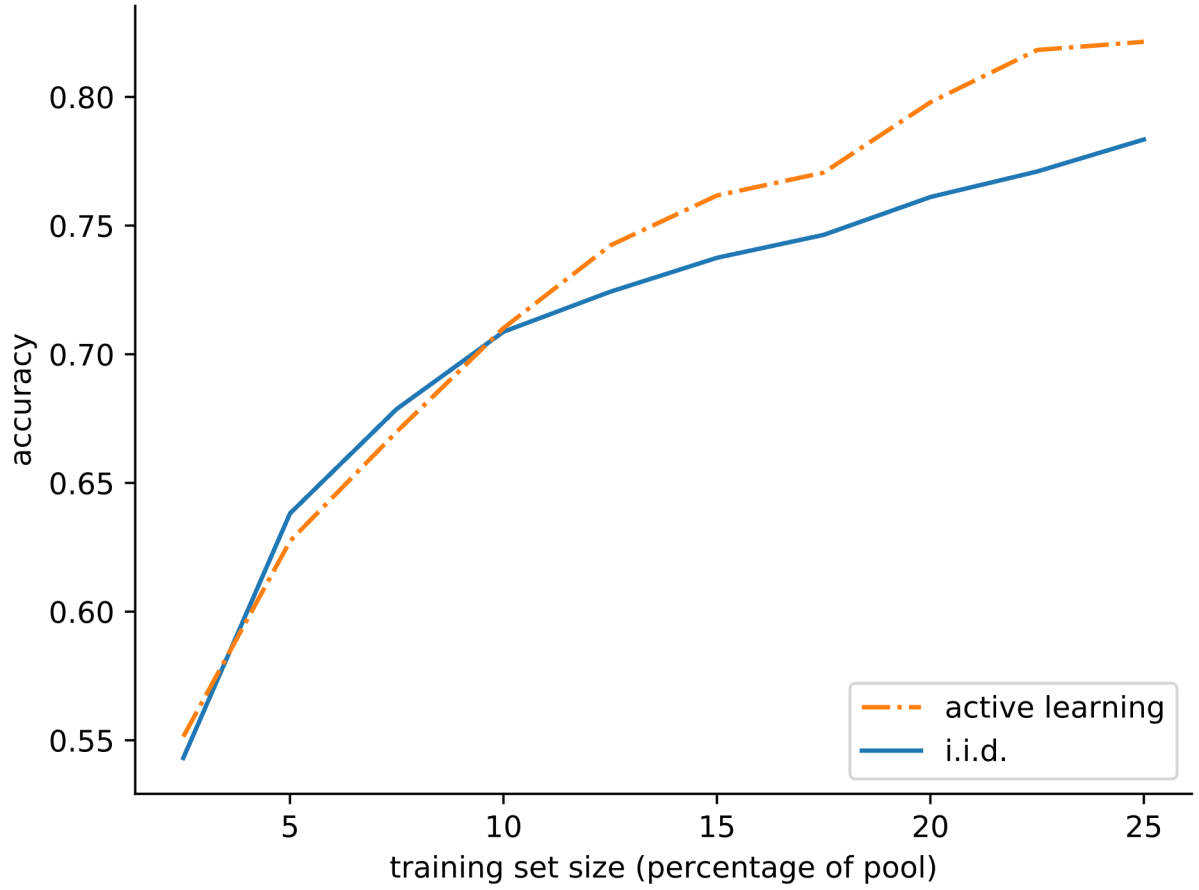
Repeat the cycle until your budget is gone



Learning Curves

The model will improve as the training set *L* grows

If our heuristic is sorting examples well, our model should improve faster



What is this heuristic?

Defining these sorting heuristics is a genre of research

Here’s some examples:

Entropy

Refresher: entropy is a measure of the uncertainty in a probability distribution

With base 2 log it is the number of bits required to represent an event

$$-\sum_{j} P(x_j) \log P(x_j)$$

Uncertainty

$$\underset{\mathbf{x} \in \mathcal{U}}{\operatorname{argmax}} - \sum_j P(y_j|\mathbf{x}) \log P(y_j|\mathbf{x})$$

Measure the entropy of the predicted label distributions

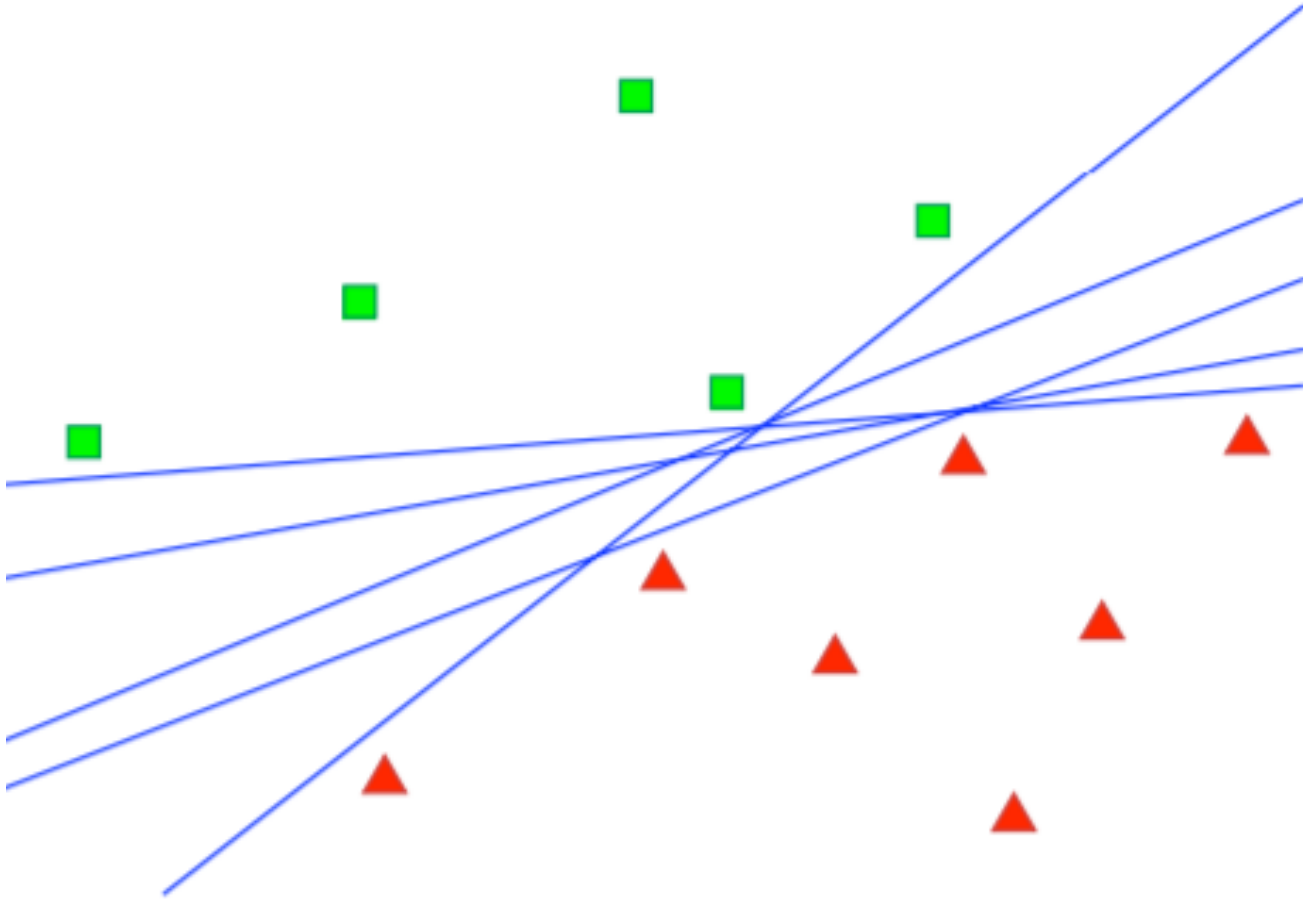
Sort documents from highest to lowest entropy

In binary classification, this is equivalent to selecting documents with predictions closest to 0.5

Query by Committee

Train *n* Different models – this is the committee

Select the documents which maximize the committee's disagreement



Query by Committee

How do we measure the committee's disagreement?

One way is to use KL-Divergence

This is a metric of the difference between two probability distributions, P and Q

$$\sum_j P(x_j) \log \frac{P(x_j)}{Q(x_j)}$$

Query by Committee

$$\underset{\mathbf{x}\in\mathcal{U}}{\operatorname{argmax}} \frac{1}{C}\sum_{c=1}^{C}\sum_{j} P_c(y_j|\mathbf{x}) \log \frac{P_c(y_j|\mathbf{x})}{P_C(y_j|\mathbf{x})}$$

However, we are comparing more than just two distributions

Our metric becomes the average KL divergence between committee members and the committee *consensus*

We define committee consensus as the average of all committee members

Bayesian Active Learning by Disagreement

First: a refresher on dropout

Dropout is a regularization technique in which a layer's input has randomly selected elements set to zero

Different elements are selected at each pass

Typically done only during training

```
>>> t = torch.rand(5,5)
>>> t
tensor([[0.4014, 0.5627, 0.0018, 0.3367, 0.8000],
        [0.4040, 0.9351, 0.6271, 0.1884, 0.7905],
        [0.7535, 0.2752, 0.7701, 0.1413, 0.3814],
        [0.5954, 0.0344, 0.8064, 0.4236, 0.7991],
        [0.6920, 0.8397, 0.9116, 0.3893, 0.2038]])
>>> t = torch.nn.functional.dropout(t, p=0.5)
>>> t
tensor([[0.0000, 1.1254, 0.0000, 0.6734, 0.0000],
        [0.8079, 1.8702, 0.0000, 0.3769, 0.0000],
        [1.5070, 0.5503, 1.5401, 0.0000, 0.7628],
        [0.0000, 0.0000, 0.0000, 0.8472, 1.5983],
        [0.0000, 0.0000, 1.8232, 0.7785, 0.4077]])
```

Bayesian Active Learning by Disagreement

$$\underset{\mathbf{x} \in \mathcal{U}}{\operatorname{argmax}}\left(1 - \frac{\operatorname{count}(\operatorname{mode}(y_{\mathbf{x}}^{1}, \ldots, y_{\mathbf{x}}^{T}))}{T}\right)$$

Apply dropout at test time

Interpret disagreement between model passes as an indicator of uncertainty

Some Caveats

Active learning can be effective

However, it comes with some challenges

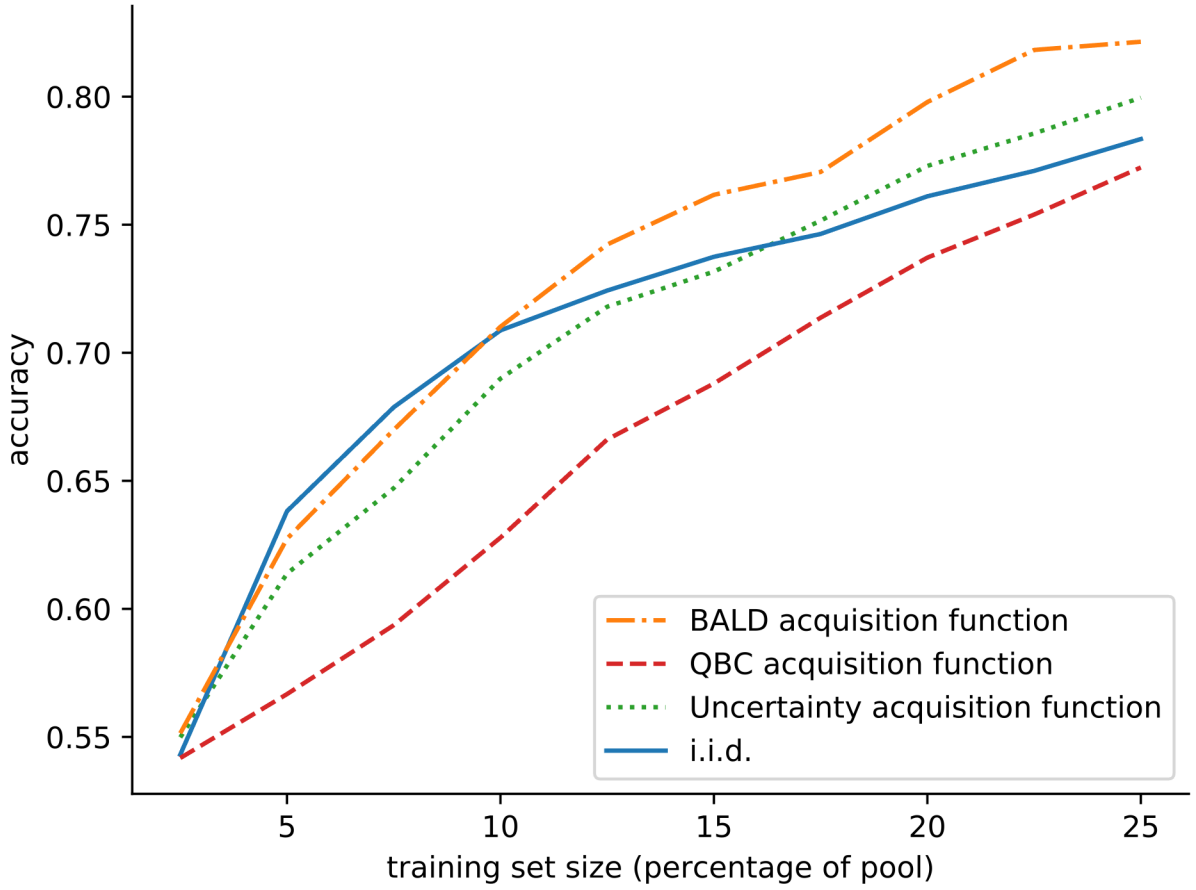
Self plug: https://arxiv.org/pdf/1807.04801.pdf

Which Heuristic?

I gave three examples, but there are more

There are also more variations on the heuristics I showed

Depending on your model and task, the choice can be important

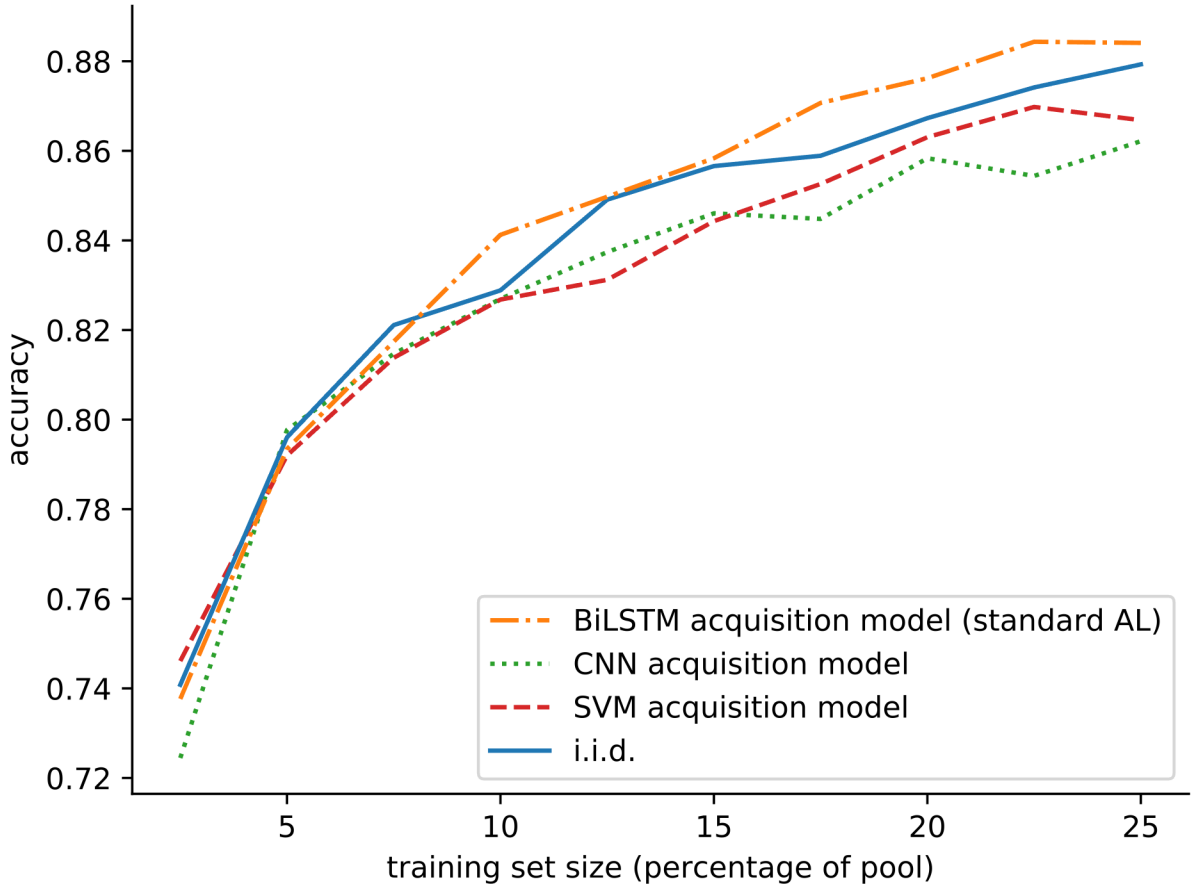


Model-Dataset Coupling

All the heuristics we looked at are conditioned on the model being trained

Datasets tend to outlive models

Will an actively learned dataset still be good if you change your model?



Questions?

Data Augmentation

Can we generate new data points from existing data points?

Define a transformation function *q*

$$(x', y') \leftarrow (q(x), y)$$

Generate new data points by applying *q* to existing data points

The goal is to find augmentations that do not change the correct labeling of *x*

Image Augmentation - Rotation

Image Augmentation - Rotation

$$p'[x', y'] \leftarrow p[x, y]$$

$$x' = \cos\theta(x) + \sin\theta(y)$$

$$y' = -\sin\theta(x) + \cos\theta(y)$$

Image Augmentation - Translation

Image Augmentation - Translation

$$p'[x, y] \leftarrow \begin{cases} p[x - t, y] & \text{if } x - t \geq 0 \\ 128 & \text{if } x - t < 0 \end{cases}$$

Image Augmentation - Cutout

Image Augmentation - Cutout

define a random region with lower bounds x_l, y_l
and upper bounds x_u, y_u

$$p'[x, y] \leftarrow \begin{cases} 128 & \text{if } x_l < x < x_u \text{ and } y_l < y < y_u \\ p[x, y] & \text{otherwise.} \end{cases}$$

Image Augmentation - Brightness

Image Augmentation - Brightness

$$p'[x, y] \leftarrow p[x, y] \times b$$

Image Augmentation - Solarize

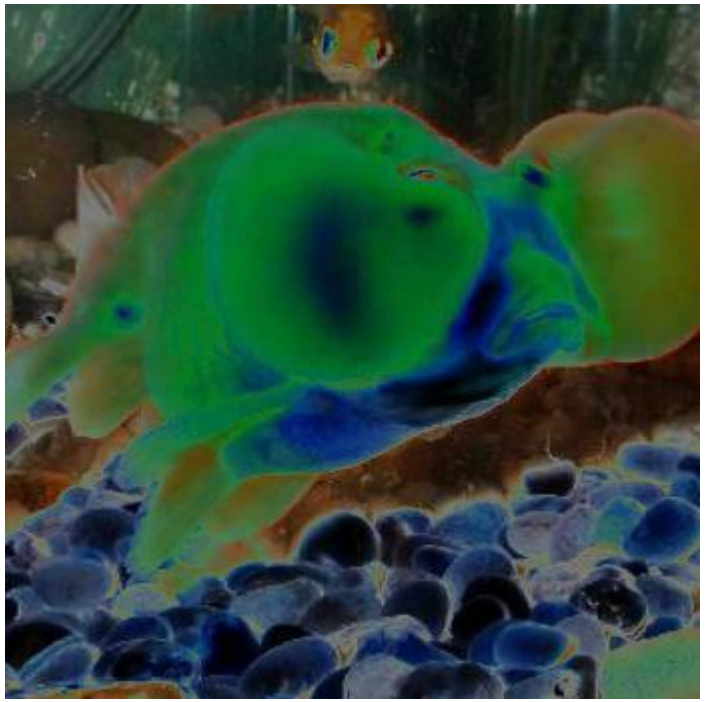


Image Augmentation - Solarize

$$p'[x, y] \leftarrow \begin{cases} p[x, y] & \text{if } p[x, y] < t \\ 255 - p[x, y] & \text{if } p[x, y] \geq t \end{cases}$$

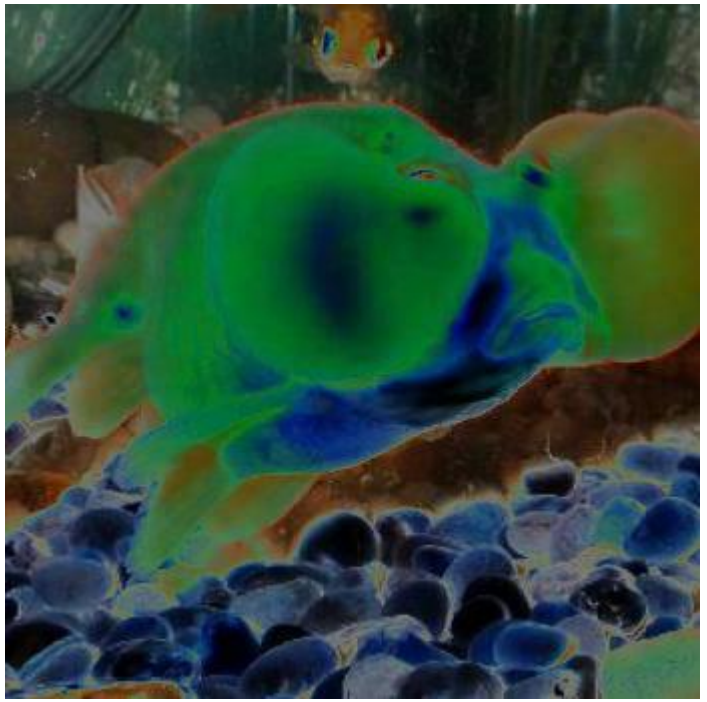
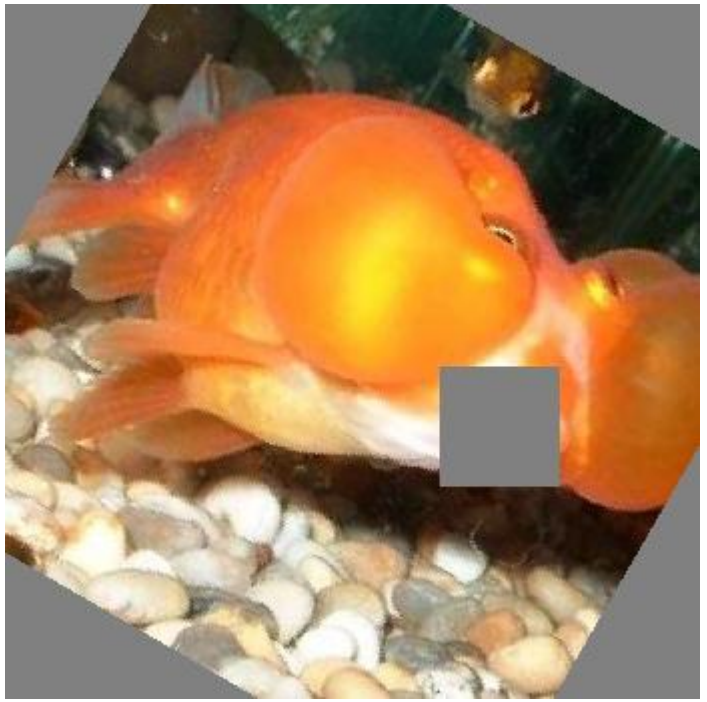
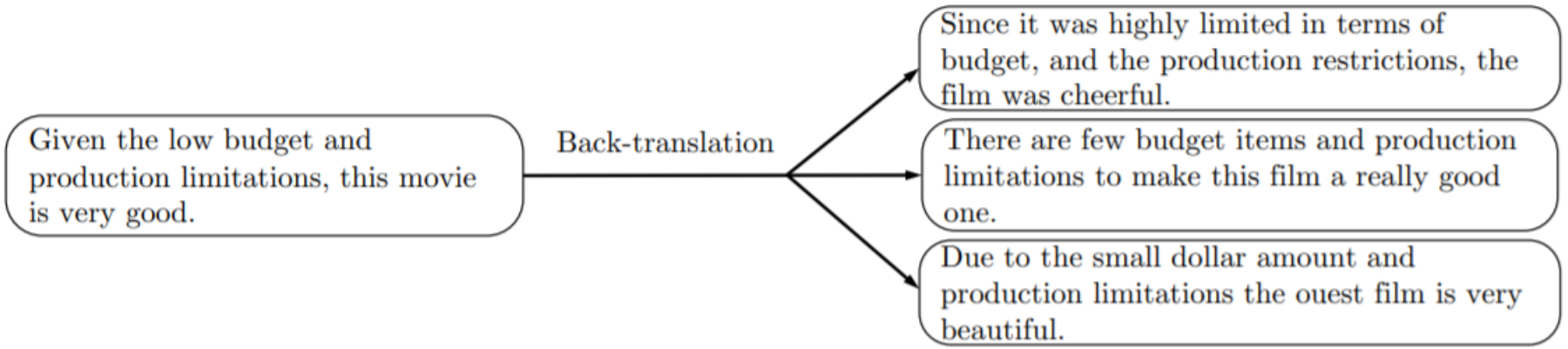


Image Augmentation – Random Combination



Text Augmentation - Backtranslation

Use machine translation to transform English text into a pivot language, then back



Text Augmentation – Word Replacement

Consider a sequence task, let's say we want to predict whether each word in a sentence is a Person, Organization, Location, or none of the above

For each word in a sentence, with probability *p* replace it with some other word

	ORG		PER	PER		PER	
	aberdeen	manager	roy	aitken	said: "it's unfortunate for us that	antoine	cannot play...
BERT	rangers		glen	mc		he	
Rand	delegates		cancer	peripheral		51,000	

BERT refresher

BERT is a language model pretrained on unsupervised text

It is pretrained as a masked language model

Here, we mask out words we have chosen to replace and use BERT's predictions to form a probability distribution over candidates

How to use Augmentations

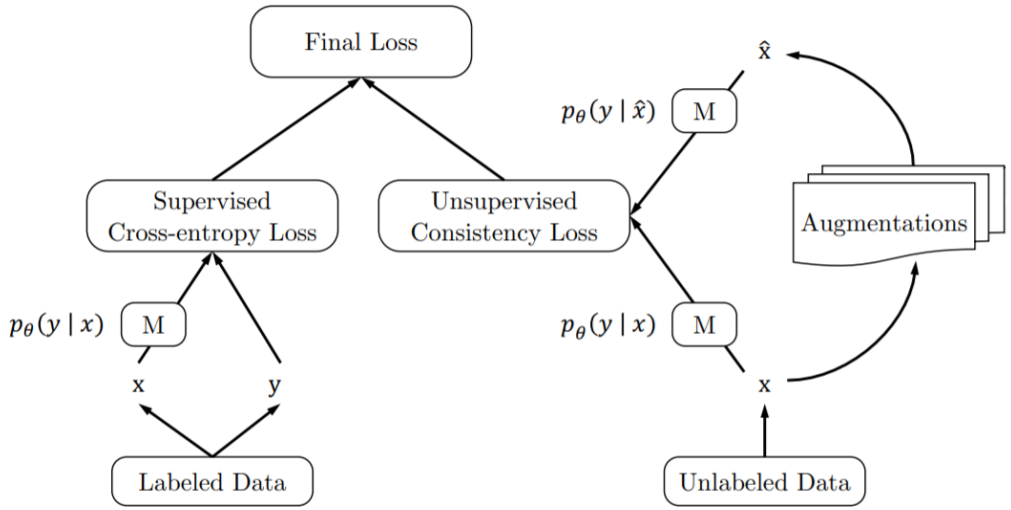
We can simply use our generated examples as additional training date

Or we can incorporate augmentation as a semi-supervision mechanism

Unsupervised Data Augmentation

We use labeled data as normal

But we also train our model to make similar predictions for *x* and *q(x)*



Unsupervised Data Augmentation

We use labeled data as normal

But we also train our model to make similar predictions for *x* and *q(x)*

Formally this means incorporating the KL loss between the two predicted distributions into our loss

$$\sum_{x,y \in L} [-\log(\hat{p}(y|x))] + \lambda \sum_{x \in \mathcal{U}} [\mathcal{L}(x)]$$

$$\mathcal{L}(x) = \mathcal{D}_{\mathrm{KL}}\{\hat{p}(y|x)||\hat{p}(y|q(x))\}$$

Unsupervised Data Augmentation

This can be extended to sequence tasks as well

$$\sum_{x,y \in L} [-\log(\hat{p}(y|x))] + \lambda \sum_{x \in \mathcal{U}} [\mathcal{L}(x)]$$

For example, let's say we want to predict whether each word in a sentence is a Person, Organization, Location, or none of the above

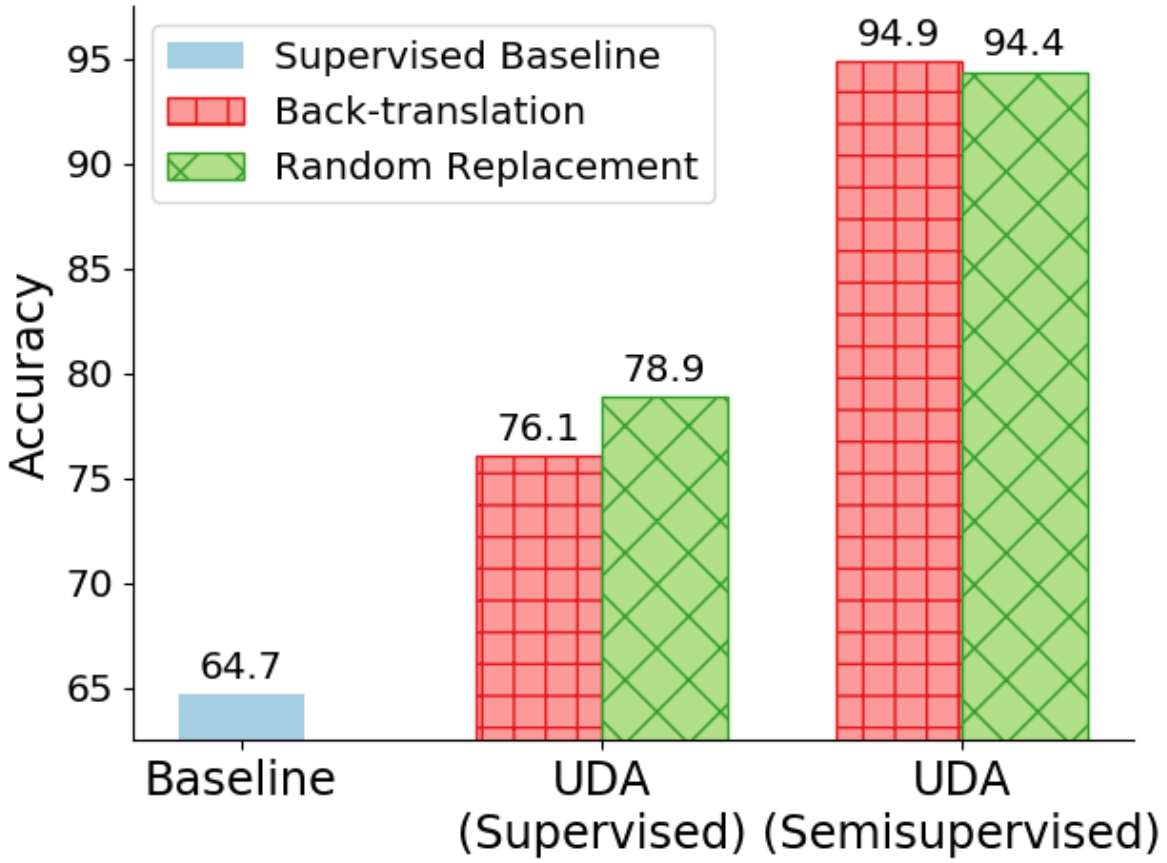
$$\mathcal{L}(x) = \frac{\sum_{j=1}^{n} \mathcal{D}_{\mathrm{KL}}(\hat{p}(y_j|x)||\hat{p}(y_j|q(x)))}{n}$$

Unsupervised Data Augmentation

This technique can lead to very large gains

Here the task is to predict whether amazon reviews are positive or negative

The labeled dataset consists of just 20 reviews

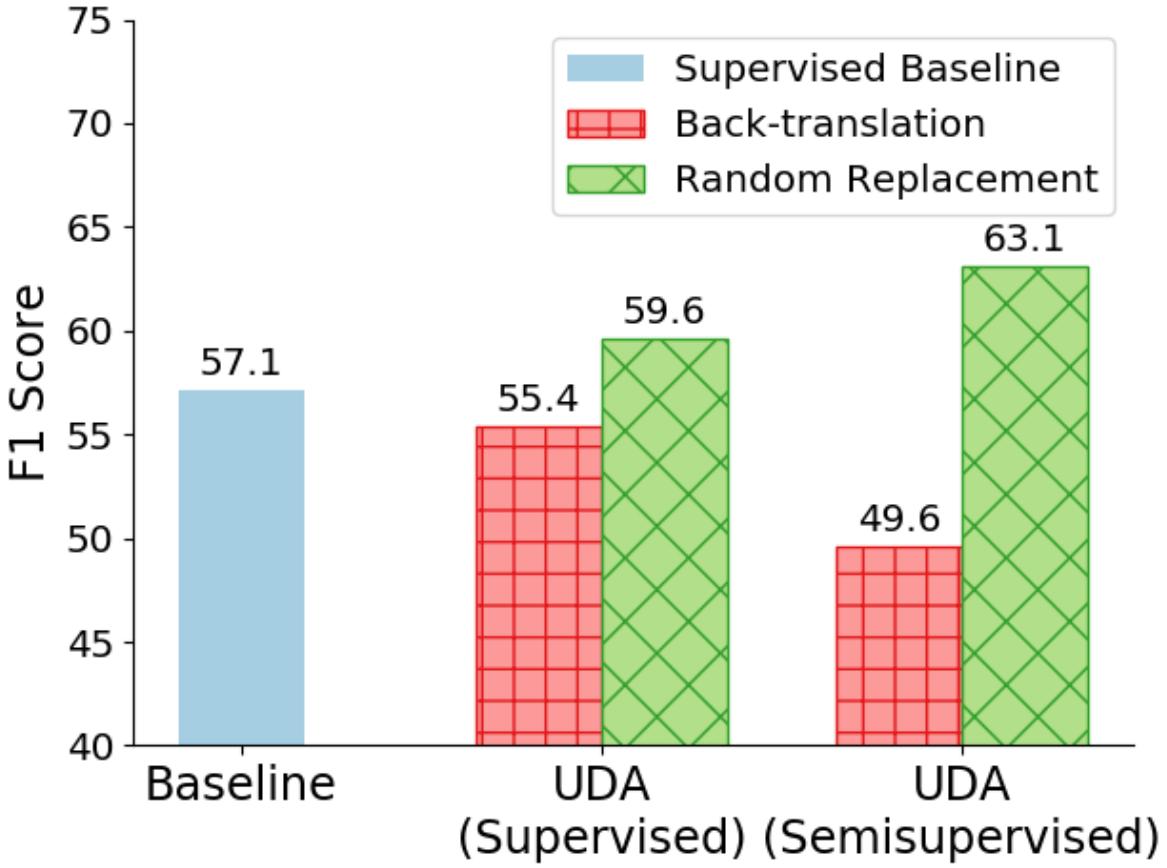


Unsupervised Data Augmentation

Random word replacement is generally competitive with back-translation

Back translation can also stumble when the task differs significantly from the translation models training data

Here the task is to infer from article text whether a medical intervention led to an increase, decrease, or no change in a patient outcome

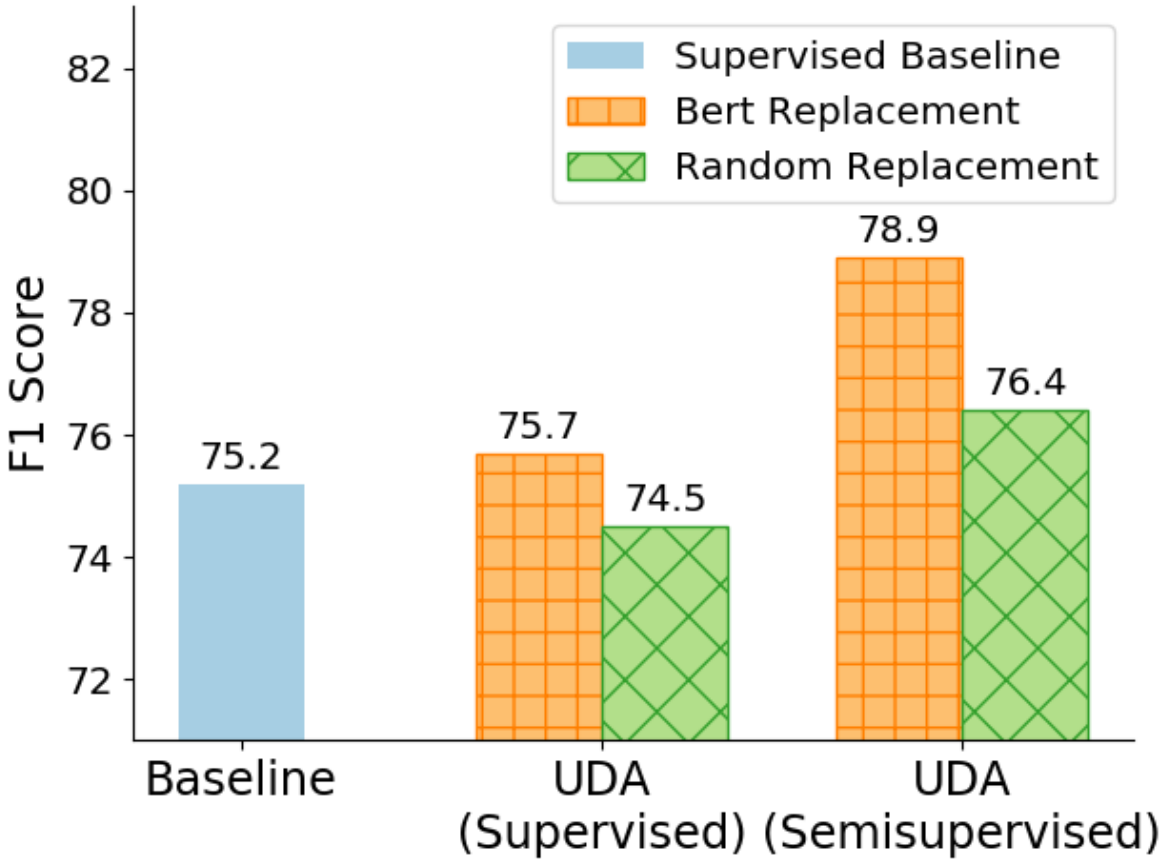


Unsupervised Data Augmentation

Gains also occur on sequence tasks

Here the task is to predict which words are people, locations, organizations, miscellaneous entities, or none of the above

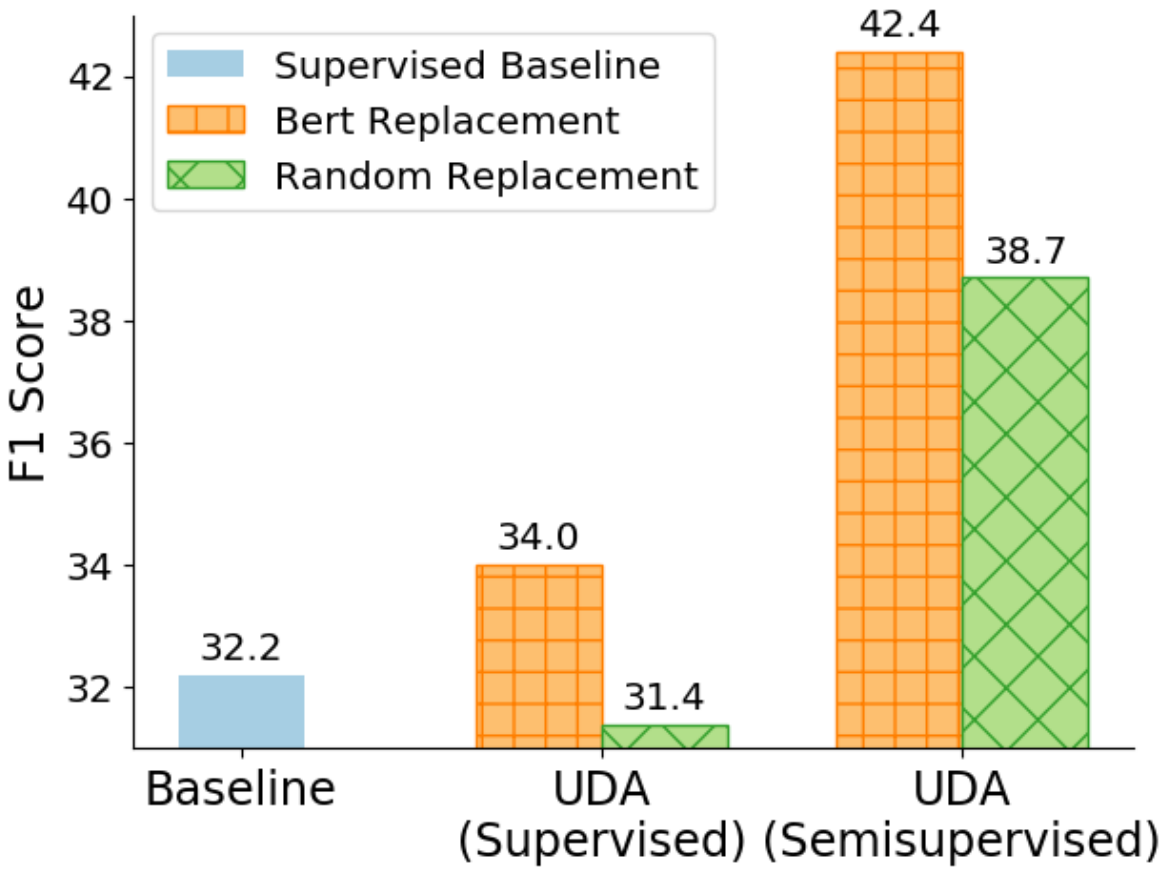
The labeled dataset consists of 200 sentences



Unsupervised Data Augmentation

The benefits for sequence tagging can be significant as well

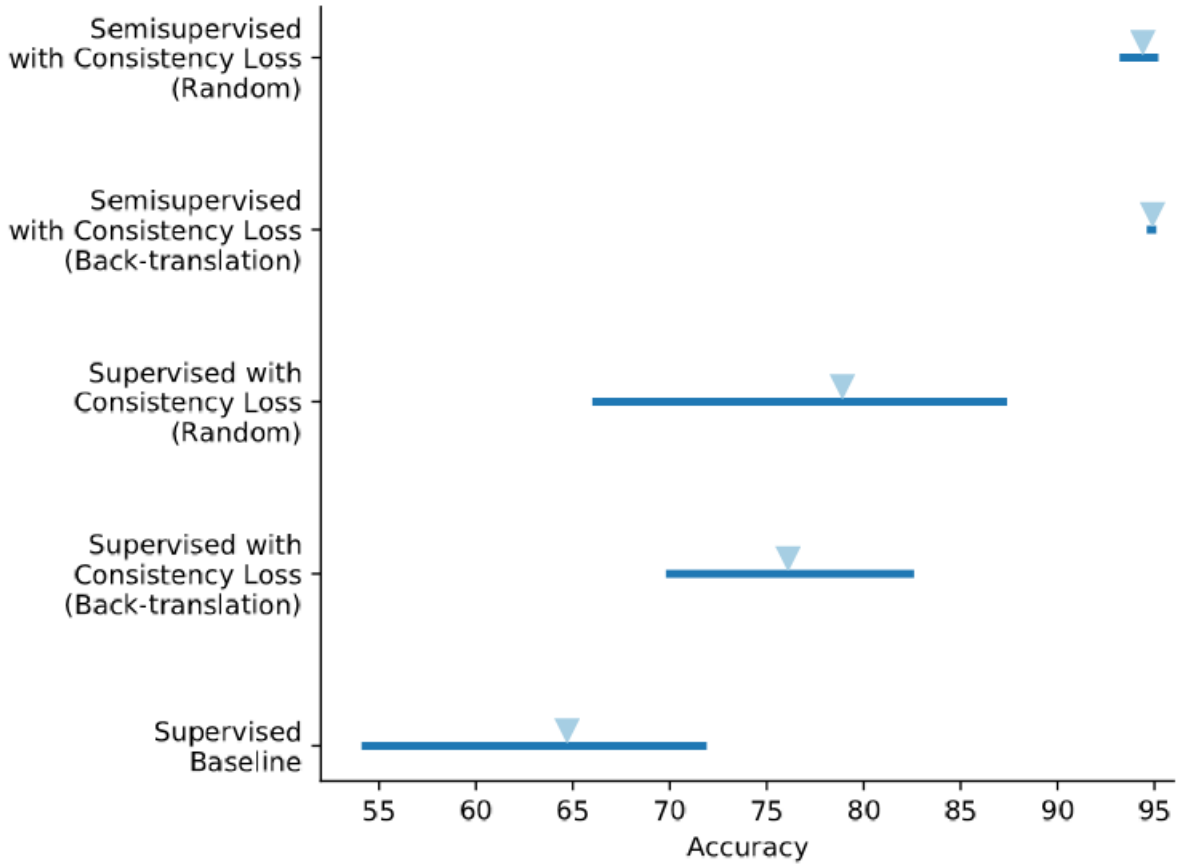
Here the task is to predict which words in a medical article describe test articles (treatments) and end points (measurable results)



Unsupervised Data Augmentation

An additional benefit of UDA is increased consistency of performance

When labeled data sets are very small, the model's performance may vary significantly, depending on which data is chosen for labeling



Recap

- UDA can lead to significant gains, even with very small labeled datasets
- Word replacement tends to equal or outperform back-translation
- Performance often increases meaningfully even without using a large pool of unlabeled data

Questions?

- 75.0%: there exists a heuristic that outperforms i.i.d.
- 60.9%: a specific heuristic outperforms i.i.d.
- 37.5%: transfer of actively acquired data outperforms i.i.d.