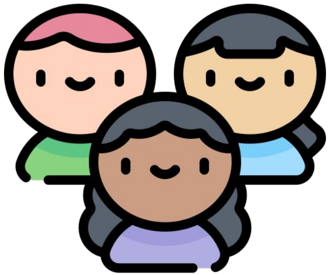
Validating Labeling Functions in Domain Shift

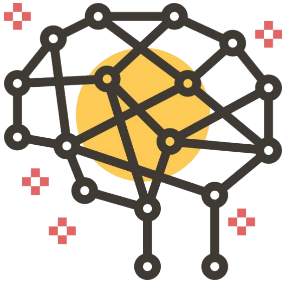
20223137 Yewon Kim
20224560 Seungjoo Lee

To obtain more labeled training data, weak supervision leverages **cheaper & noisy** labels



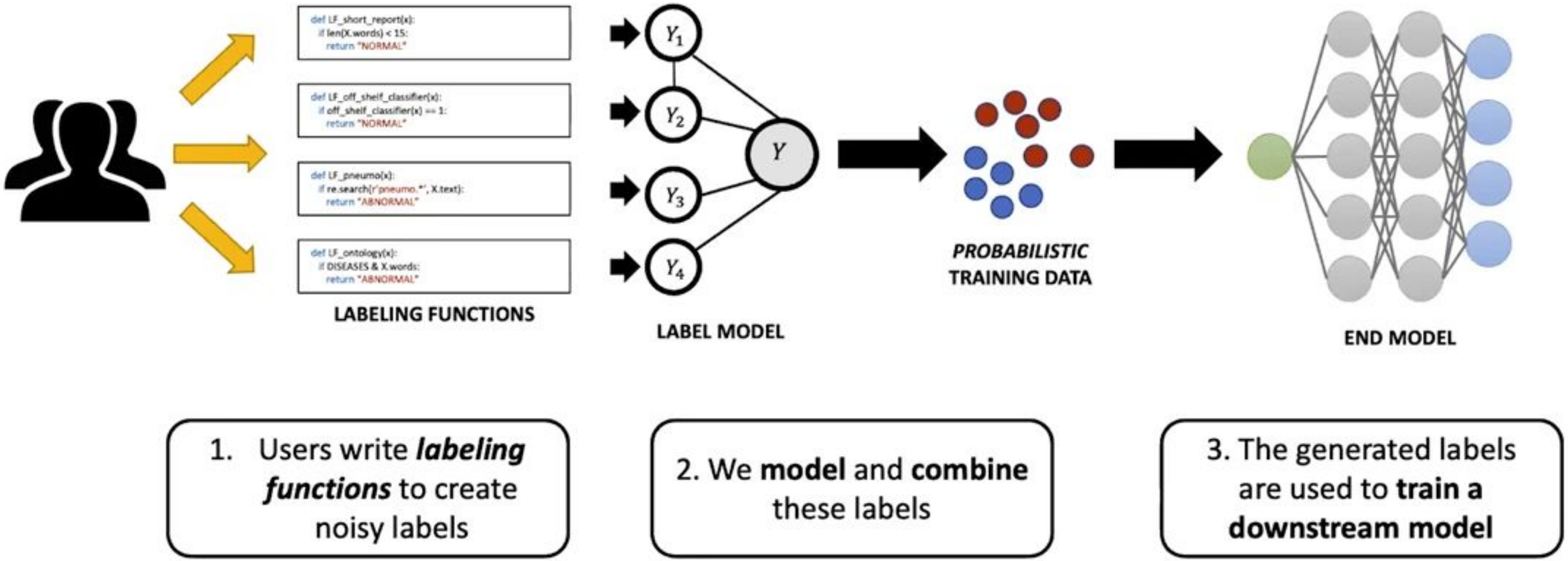
Get cheaper labels from non-experts
e.g., crowdsourcing

Get higher-level supervision from experts
e.g., labeling functions



Get pseudo-labels from pre-trained models
e.g., knowledge distillation

Labeling function (LF) is a lightweight and cost effective way to generate labels in unlabeled data.



Source: https://snorkel.ai/weak-supervision-modeling/

However, what if data distribution shifts?

Developer side

```
^(?=.*\bkid\b)(?=.*\blike\b).*$
```

Data stream

time →

Domain: toy

review/"My kid likes it"
⇒ **positive**

Example scenario:
sentiment analysis from review data

However, what if data distribution shifts?

Developer side

```
^(?=.*\bkid\b)(?=.*\blike\b).*$
```

Data stream

time →

Domain: toy	Domain: book
review/"My kid likes it" ⇒ **positive**	review/"A true love story" ⇒ **???**

LFs are no longer valid;
need to update!

`Example scenario:`
`sentiment analysis from review data`

However, what if data distribution shifts?

Accurately and timely **detecting the data shift** and
prompting engineers to update LFs is crucial
in order to ensure the reliable performance of an end model!

Example scenario:
sentiment analysis from review data

Our idea: Use the outputs from LFs to detect domain shift!

- Previous works: **observe an input x itself** to determine if it is out-of-distribution (OOD)
  - Specifically, define a score function $s(x)$ and classify it as OOD if $s(x) < \delta$ where δ is a predefined threshold.
  - Score functions: e.g., language models
- Instead, we **observe outputs of LFs** to determine OOD
  - Outputs of LFs contain richer information as LFs are specifically designed to identify certain aspects of the data.
  - More efficient and scalable, as it does not necessarily require models to capture important features from raw data.

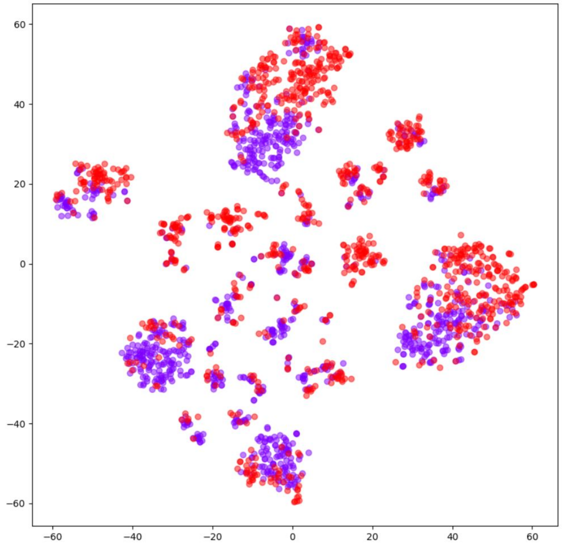
Method: (1) Changing discrete LFs to continuous LFs

Example of **discrete** LF on NLP sentiment analysis:
Keyword-based heuristic function

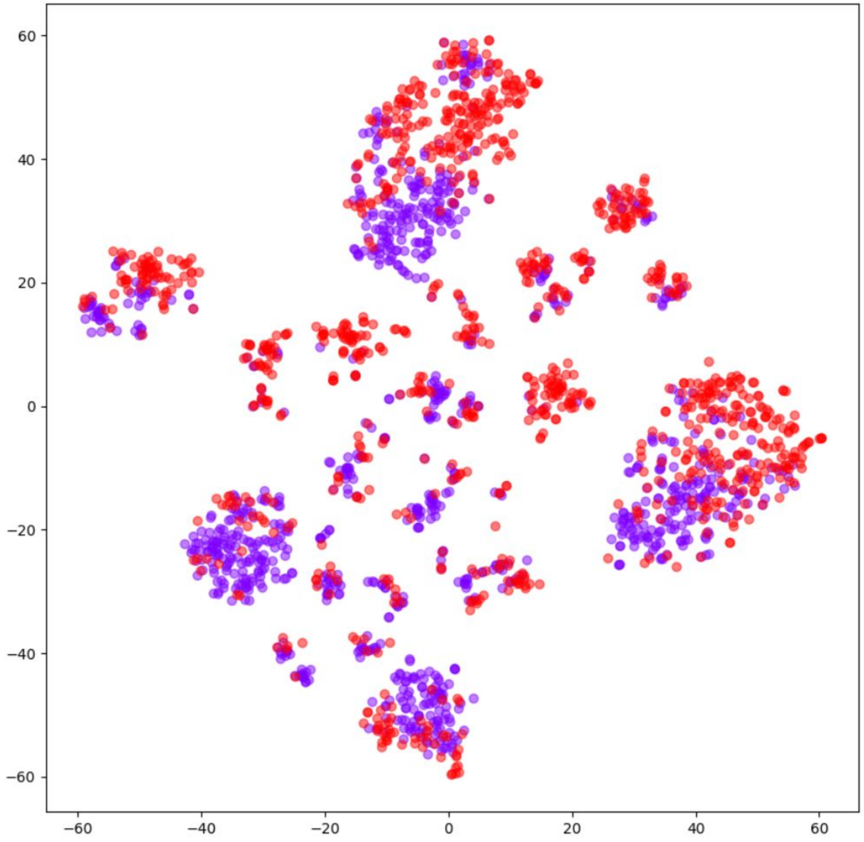
```
@labeling_function
def positive_keyword_lf(text, keyword):
    if keyword in text.lower():
        return POSITIVE
    return ABSTAIN
```

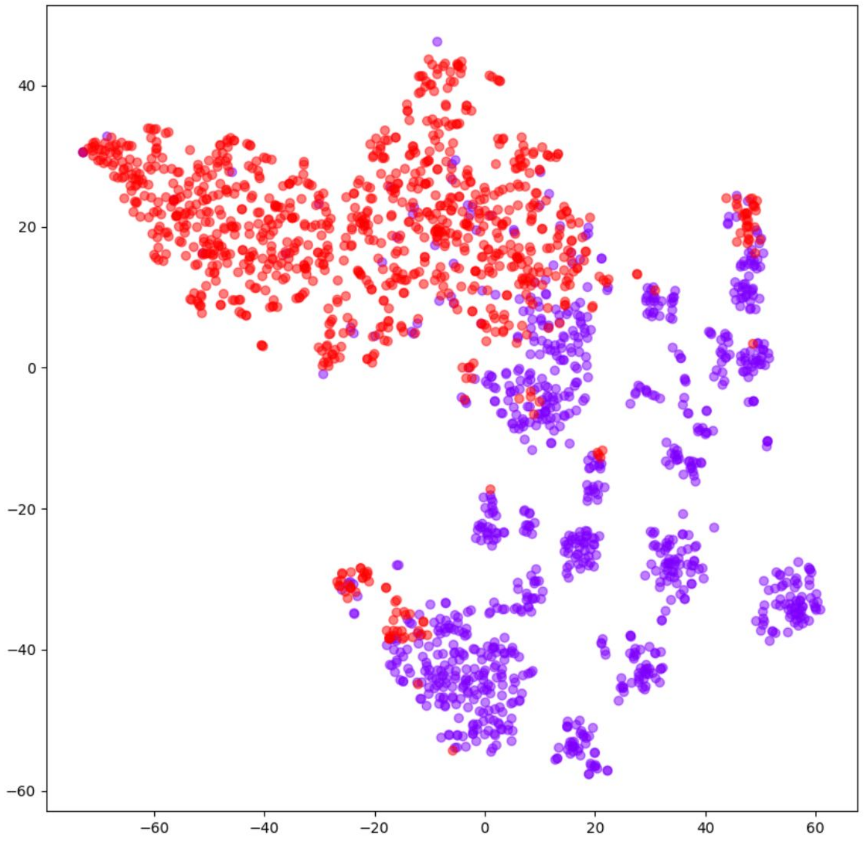
- Outputs **limited** values (POSITIVE, NEGATIVE, ABSTAIN)
- Information from discrete LFs are **not enough** to detect OOD

T-SNE result, 8 discrete LFs



Method: (1) Changing discrete LFs to continuous LFs





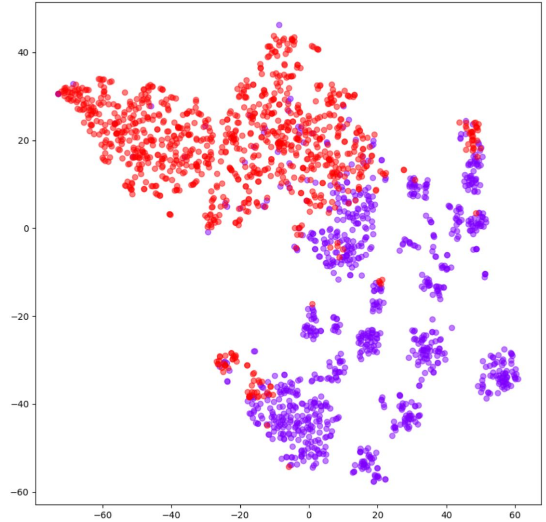
Method: (1) Changing discrete LFs to continuous LFs

Example of **continuous** LF:
Using **cosine similarity** of **GloVe** word embedding

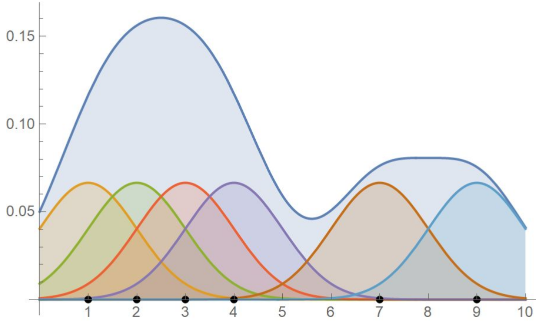
```
@labeling_function
def positive_keyword_lf(text, keyword):
    text_emb = glove(text.lower().split())
    keyword_emb = glove([keyword])

    return get_cosine_similarity(text_emb, keyword_emb)
```

- Cosine similarity between passage and keyword
- Outputs **continuous** values → **dense** information

T-SNE result, 8 continuous LFs



Method: (2) Kernel density estimation



Given an input $x_i \in D_{ID}^{Tr}$ where D_{ID}^{Tr} is in-distribution (source) train data, let f_{x_i} be a vector of labeling function outputs from x_i. (=feature)

Estimate the marginal feature probability density function $\mathrm{p}(f)$ based on Gaussian kernel:

$$p(f) \approx \hat{p}(f) = \frac{1}{|D_{ID}^{Tr}|h} \sum_{j=1}^{|D_{ID}^{Tr}|} \mathcal{K}\left(\frac{f - f_j}{h}\right)$$

where h is a smoothing bandwidth (hyperparameter) and $\mathcal{K}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ is a Gaussian kernel function.

Given $\hat{p}(f)$ and a predefined threshold δ, we can determine whether a new feature f' is OOD at test time.

Overall pipeline : Training phase

Unlabeled training data

→

Discrete LFs

$$lf_1^d(x)$$
$$lf_2^d(x)$$
$$\vdots$$
$$lf_N^d(x)$$

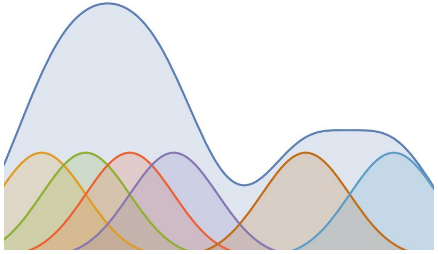
→

Continuous LFs

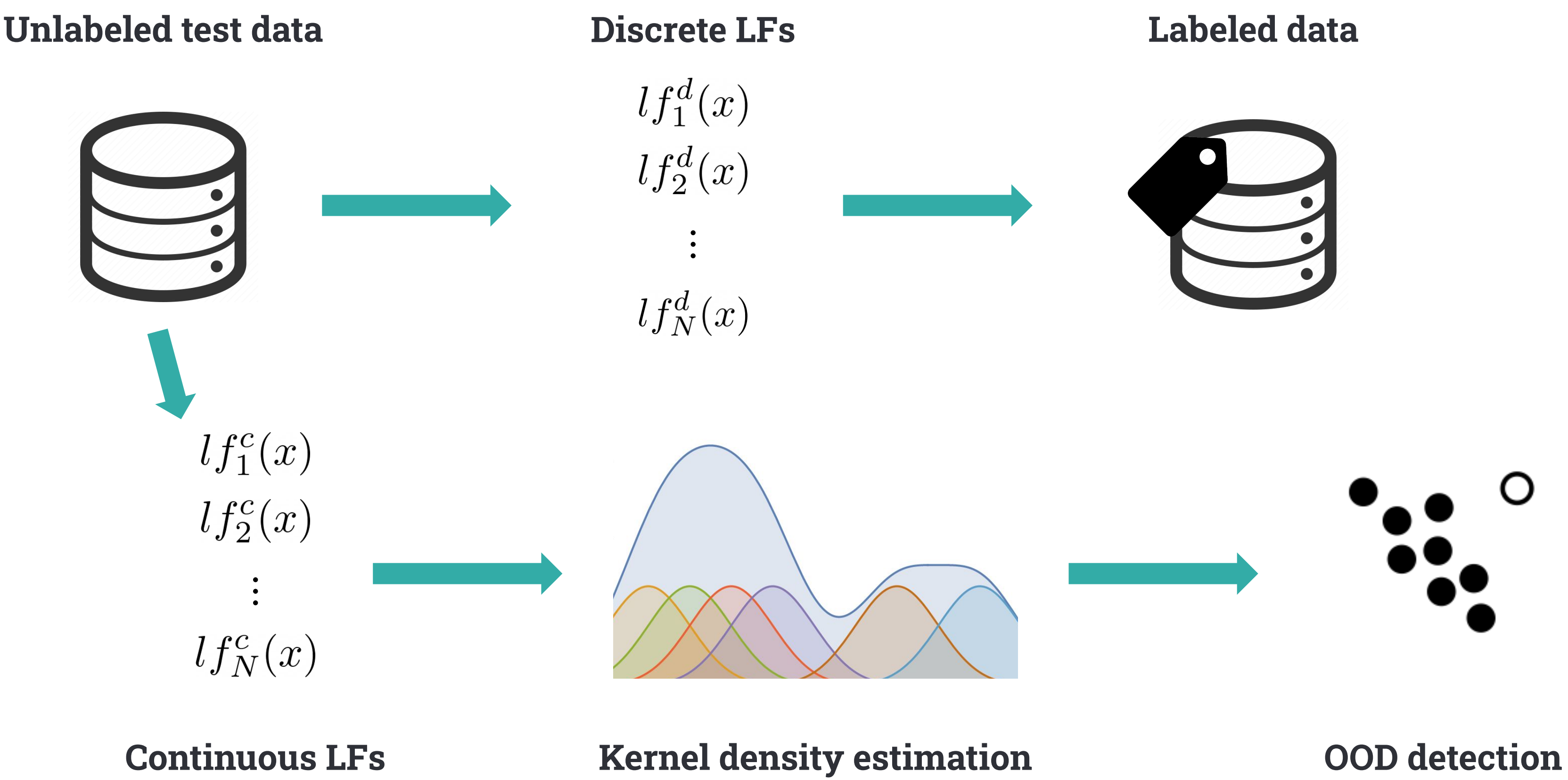
$$lf_1^c(x)$$
$$lf_2^c(x)$$
$$\vdots$$
$$lf_N^c(x)$$

→

Kernel density estimation



Overall pipeline : Testing phase



Evaluation setup: task and dataset

Sentiment analysis task (binary classification); we used IMDB [1], Yelp [2], and Amazon reviews [3].

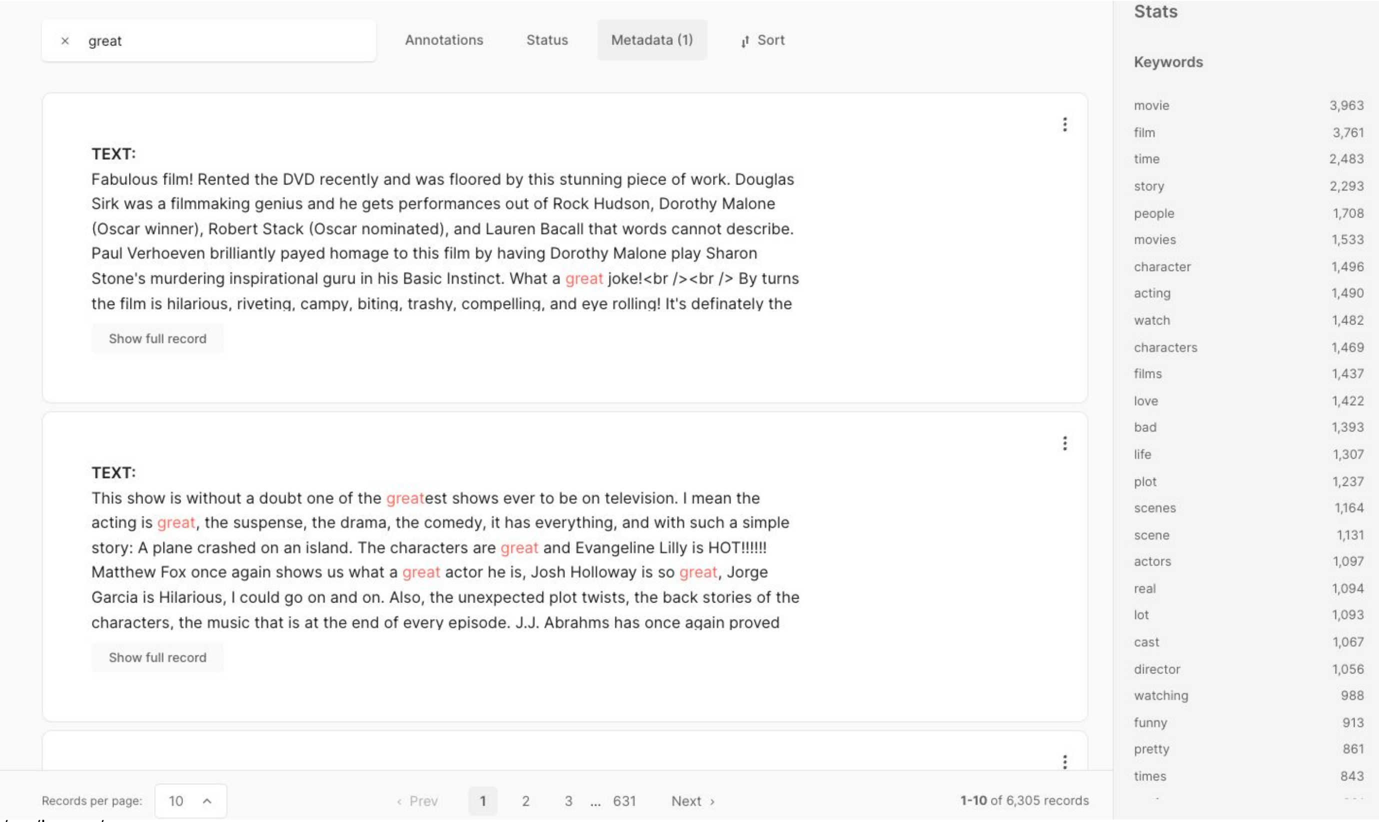
- Train : ID (20000) / Test : ID (5000) + OOD(5000)

In-distribution (ID)	Out-of-distribution (OOD)
IMDB	Yelp
	Amazon-baby
	Amazon-electronics
	Amazon-jewelry
	Amazon-home
	Amazon-sports

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*
[2] Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems 28 (NIPS 2015).
[3] R. He, J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016

Evaluation setup: LF development

- Keyword-based interactive LF generation using Argilla[1]



[1] https://docs.argilla.io/en/latest/

Evaluation setup: LF development

- Keyword-based interactive LF generation using Argilla[1]
    - 12 positive keywords, 20 negative keywords

Label	Keywords
Positive	impress, adorable, enjoy, excellent, beautiful, wonderful, recommend, best, masterpiece, performance * best, performance * good
Negative	terrible, poor, stupid, wrong, disappoint, painful, awful, boring, worse, worst, bad, cliche, killer, unnecessary, waste, least try, nothing * special, nothing * even, performance * worst, acting * bad

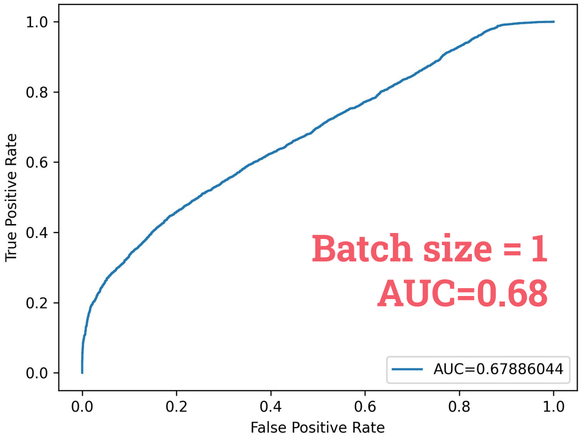
[1] https://docs.argilla.io/en/latest/

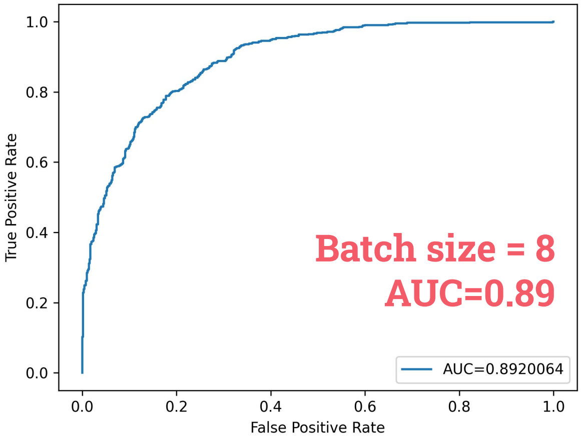
Results: OOD Detection

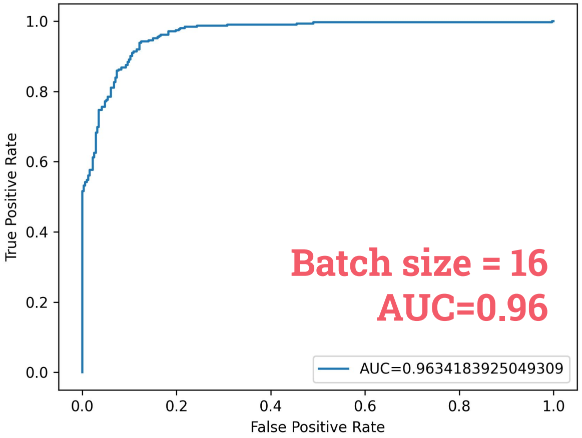
KDE h = 0.05, batch 16 fixed

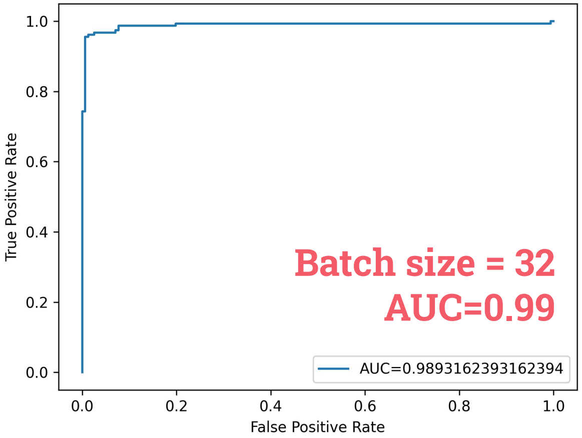
ID	OOD	AUROC	Accuracy	
			OOD (Coverage)	ID (Coverage)
IMDB	Yelp	0.93	0.78 (0.57)	0.74 (0.82)
	Amazon-baby	0.96	0.78 (0.41)	
	Amazon-electronics	0.95	0.75 (0.42)	
	Amazon-jewelry	1.00	0.86 (0.39)	
	Amazon-home	0.98	0.80 (0.39)	
	Amazon-sports	0.99	0.79 (0.33)	

Batch-AUROC Tradeoff









Discussion & Future work

- Providing **explainable** prompts to engineers
  - Train **separate** OOD detector for **each LF**
  - When the **OOD** detected, run **LF OOD detectors** to **find out wrong LF**s
- Other ways to convert discrete LFs to continuous LFs
- Using **coverage** as OOD predictor
  - **Coverage drops significantly** with OOD data
- Experiments on different **shift scenarios** & **domains**
  - Only IMDB is used as source distribution
  - Applying to **other NLP task**s
  - Applying to other domains such as **vision**

OOD	Accuracy
	OOD (Coverage)
Yelp	0.78 (**0.57**)
Amazon-baby	0.78 (**0.41**)
Amazon-electronics	0.75 (**0.42**)
Amazon-jewelry	0.86 (**0.39**)
Amazon-home	0.80 (**0.39**)
Amazon-sports	0.79 (**0.33**)