Lifelong Representation Learning for NLP Applications

Hu Xu

Committee Members:
Prof. Philip S. Yu, Chair and Advisor
Prof. Bing Liu, Co-advisor
Prof. Piotr Gmytrasiewicz
Prof. Natalie Parde
Prof. Sihong Xie (Lehigh University)

Motivation

- **Representation learning** lives at the heart of deep learning for NLP: such as in supervised classification and self-supervised (or unsupervised) embedding learning.

- Most existing methods assume a static world and aim to learn representations for the existing world.

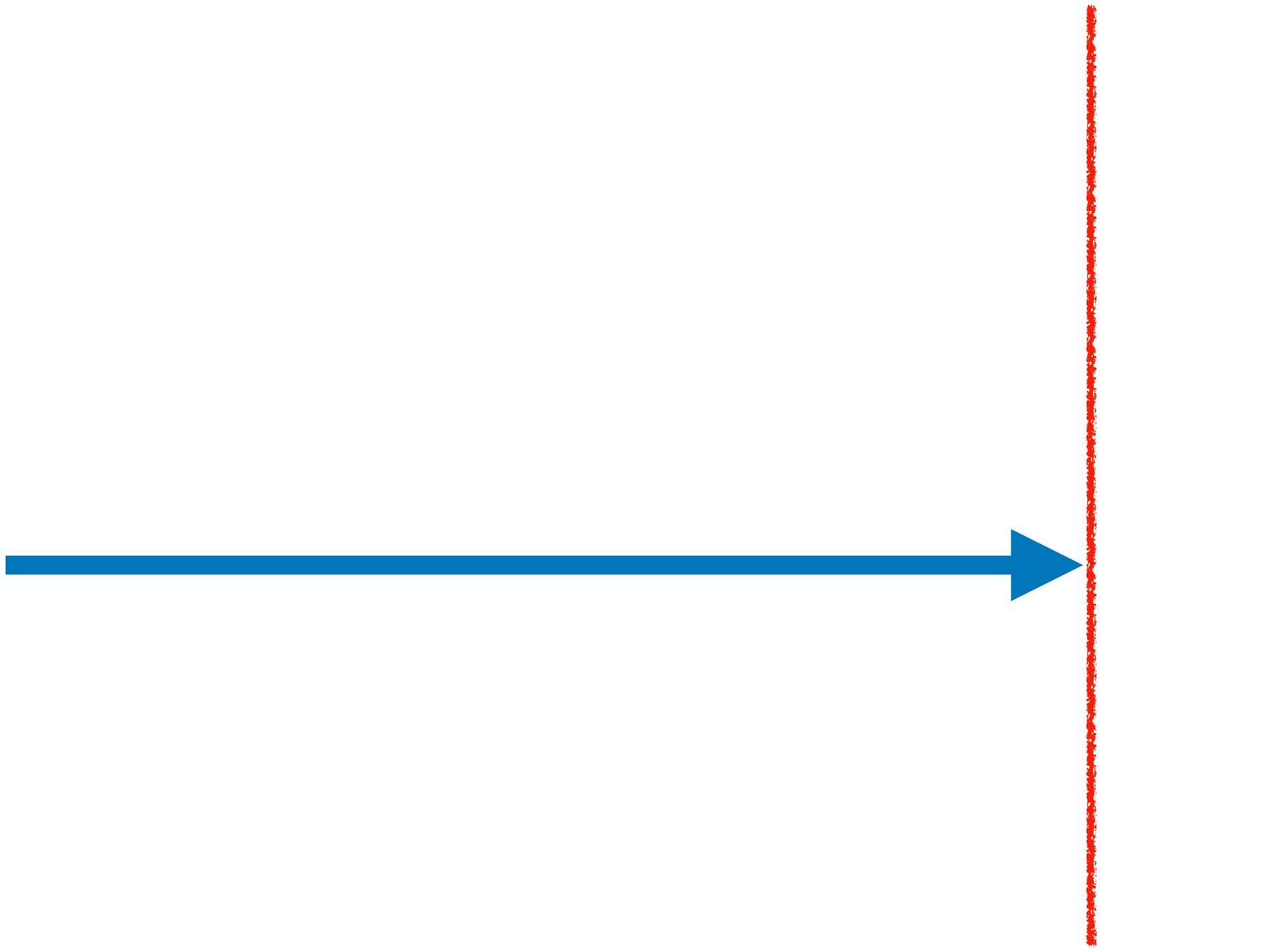
- However, the world keeps evolving and challenging existing learned representation.

Motivation

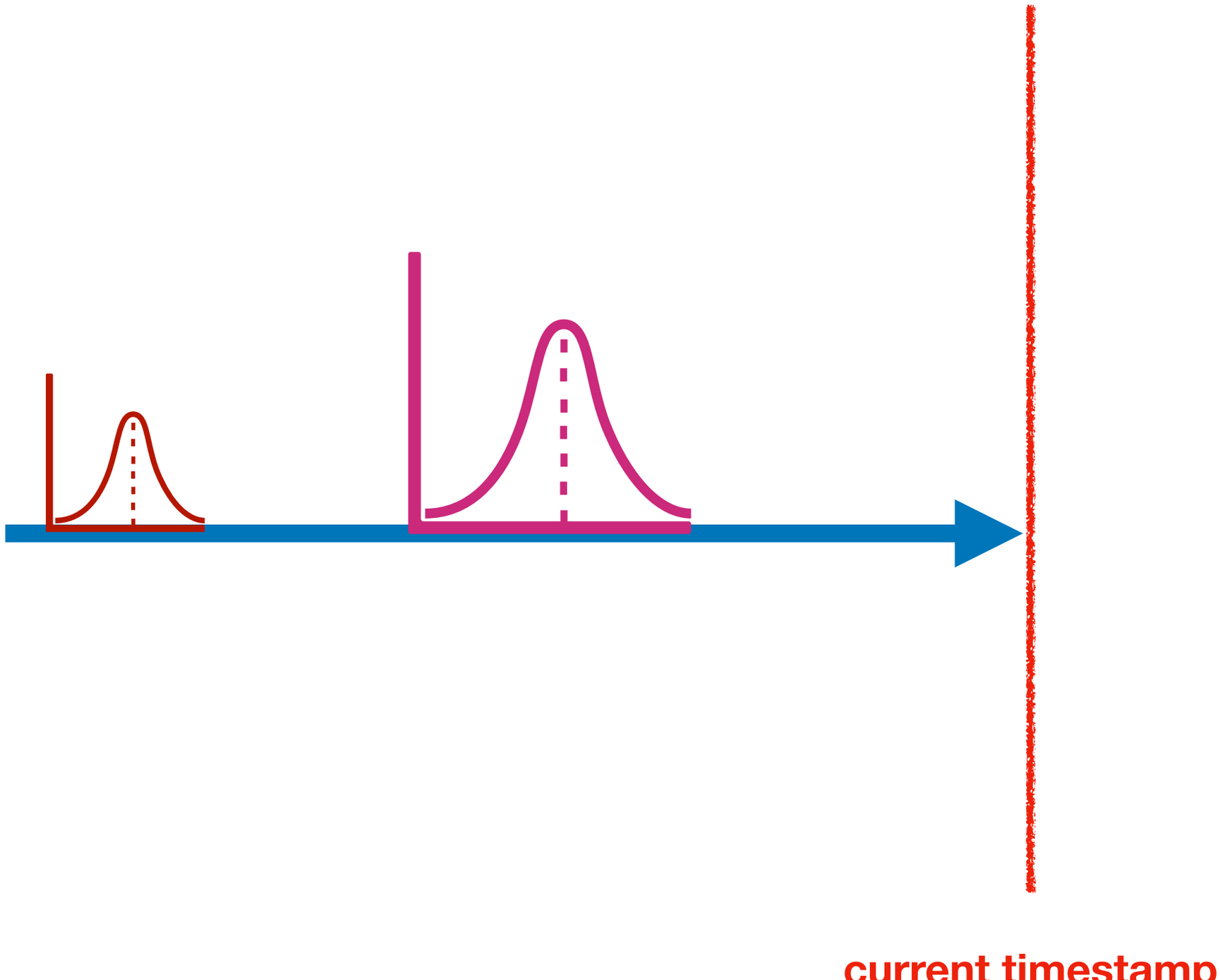
Motivation

The world evolves till now ...

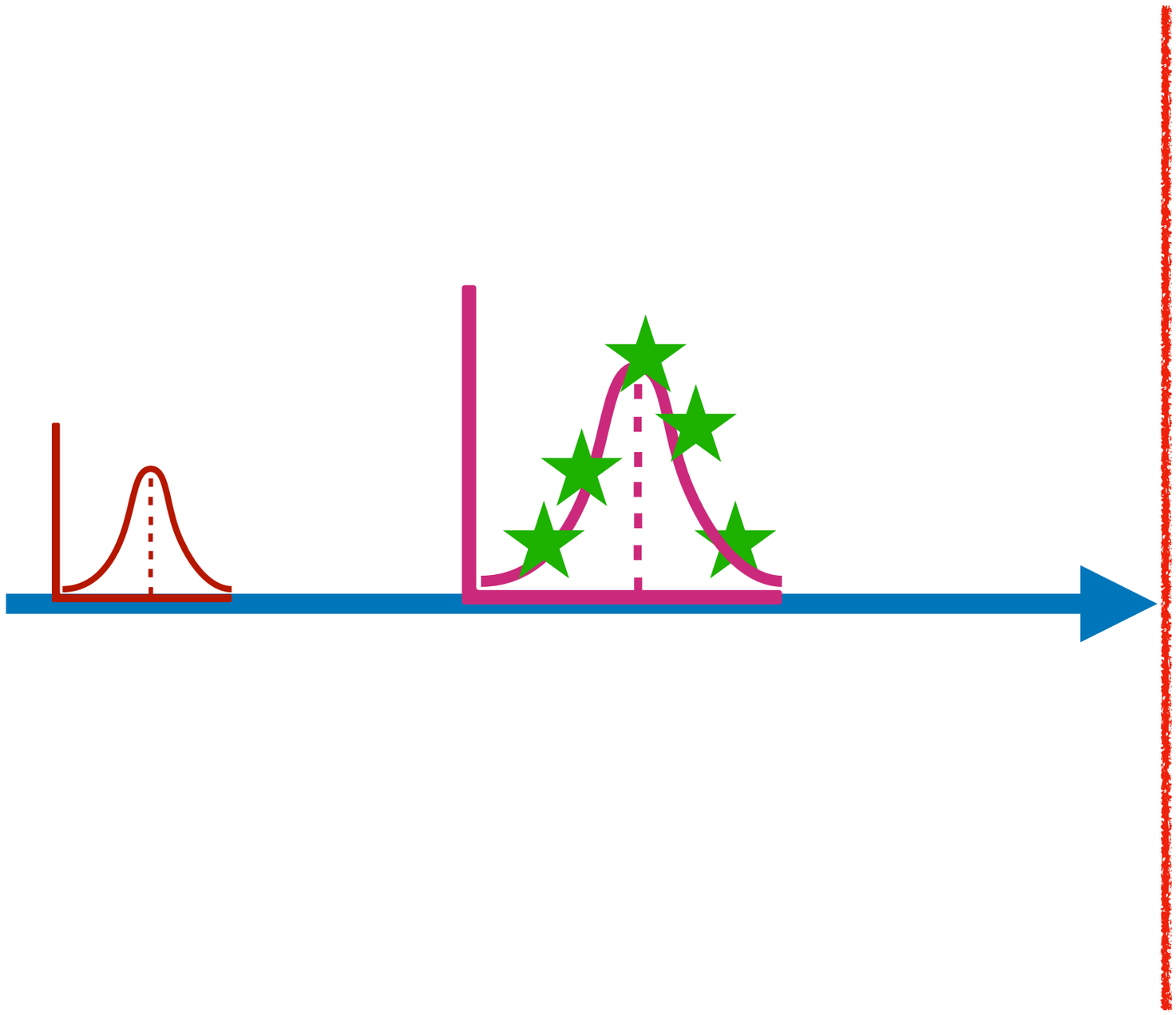
Motivation



Motivation

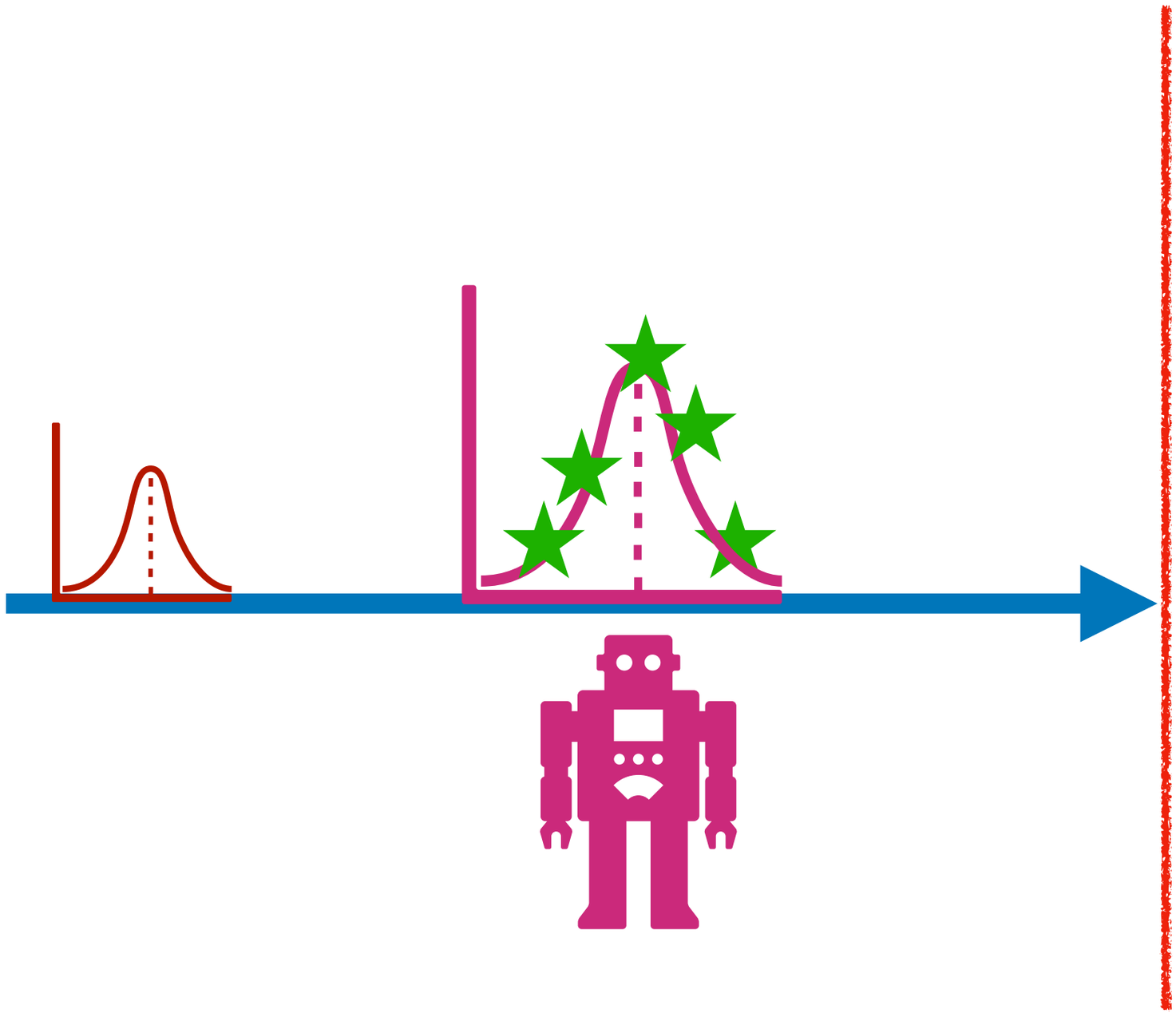


Motivation



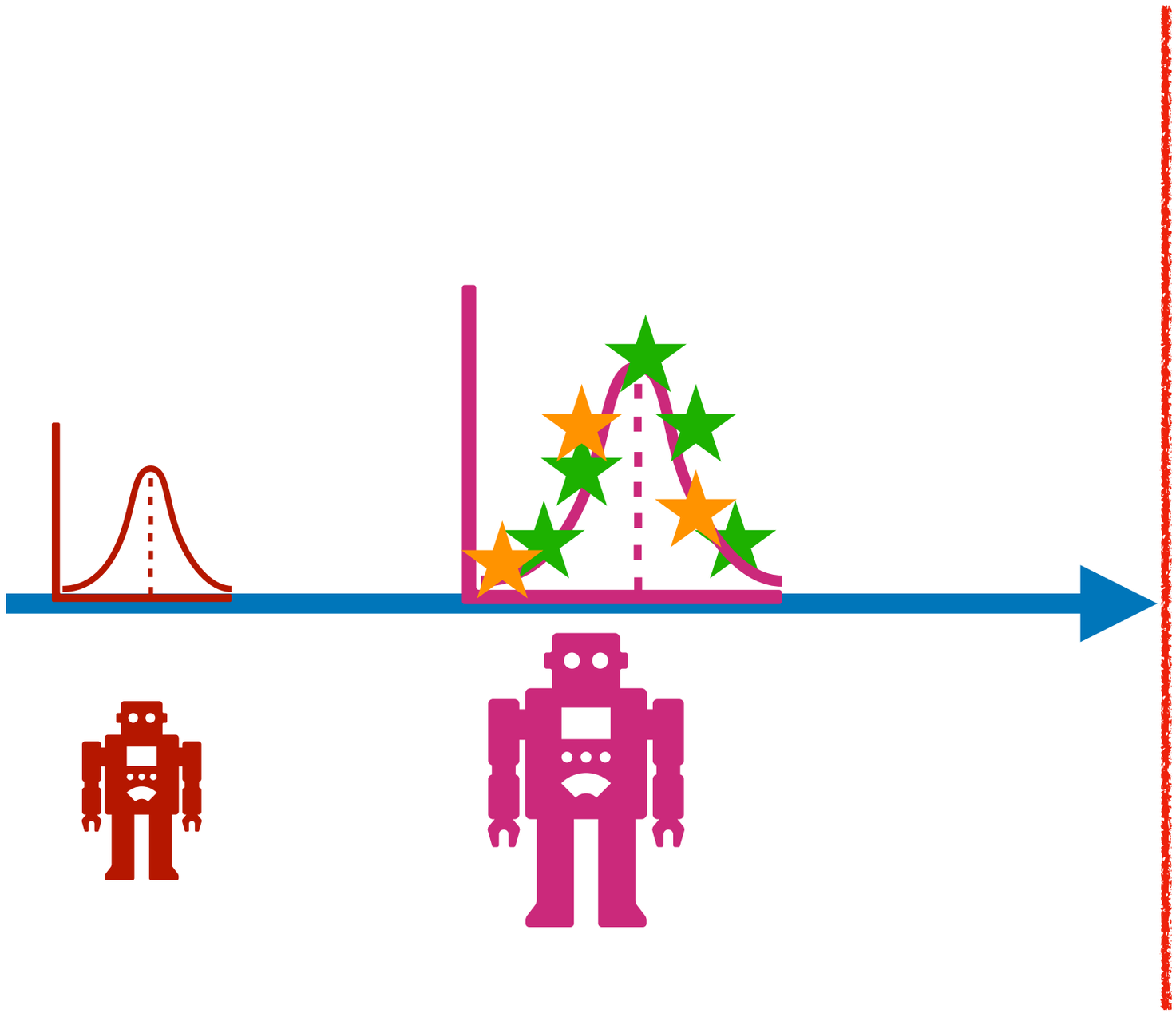
current timestamp

Motivation

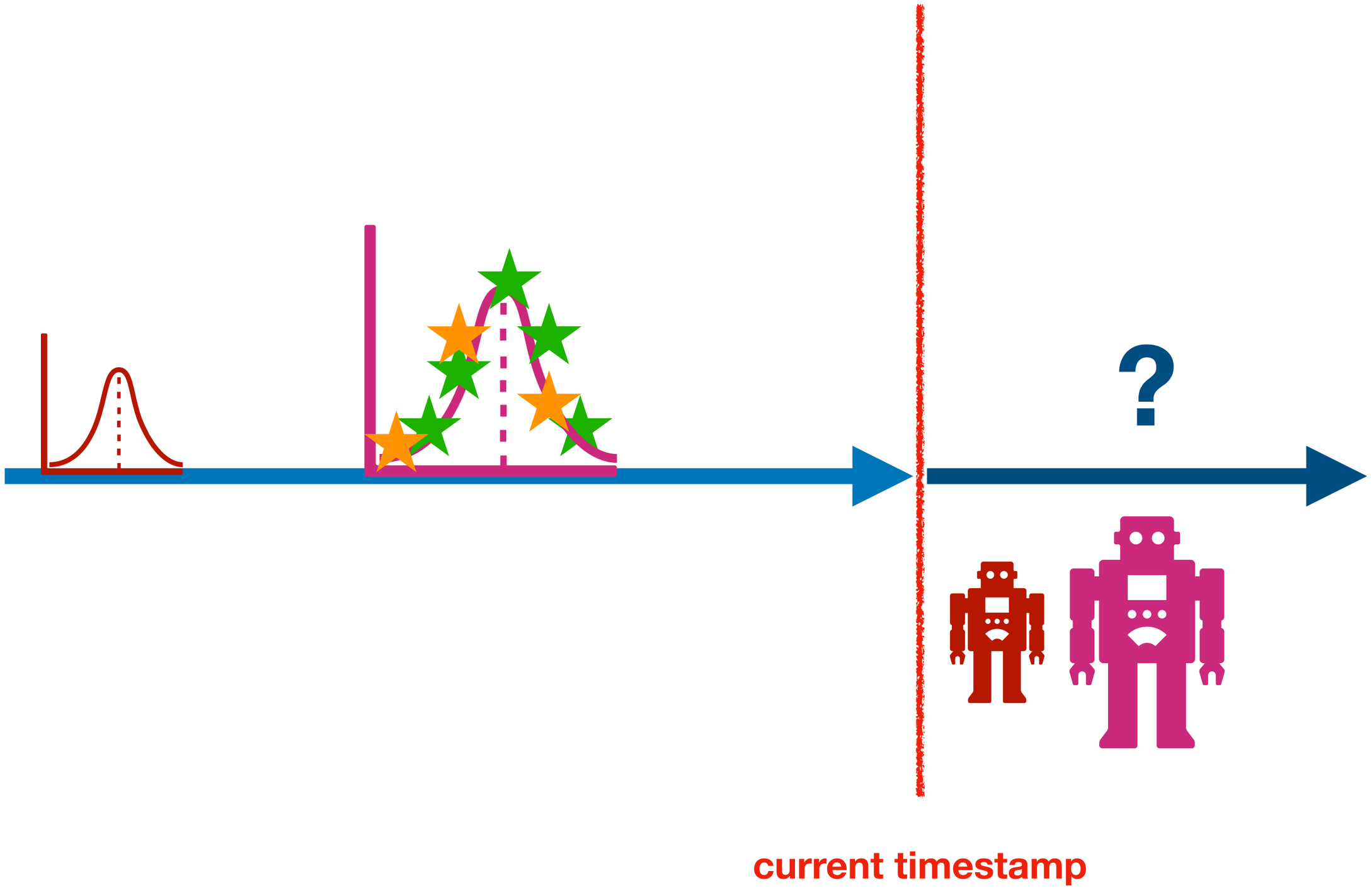


current timestamp

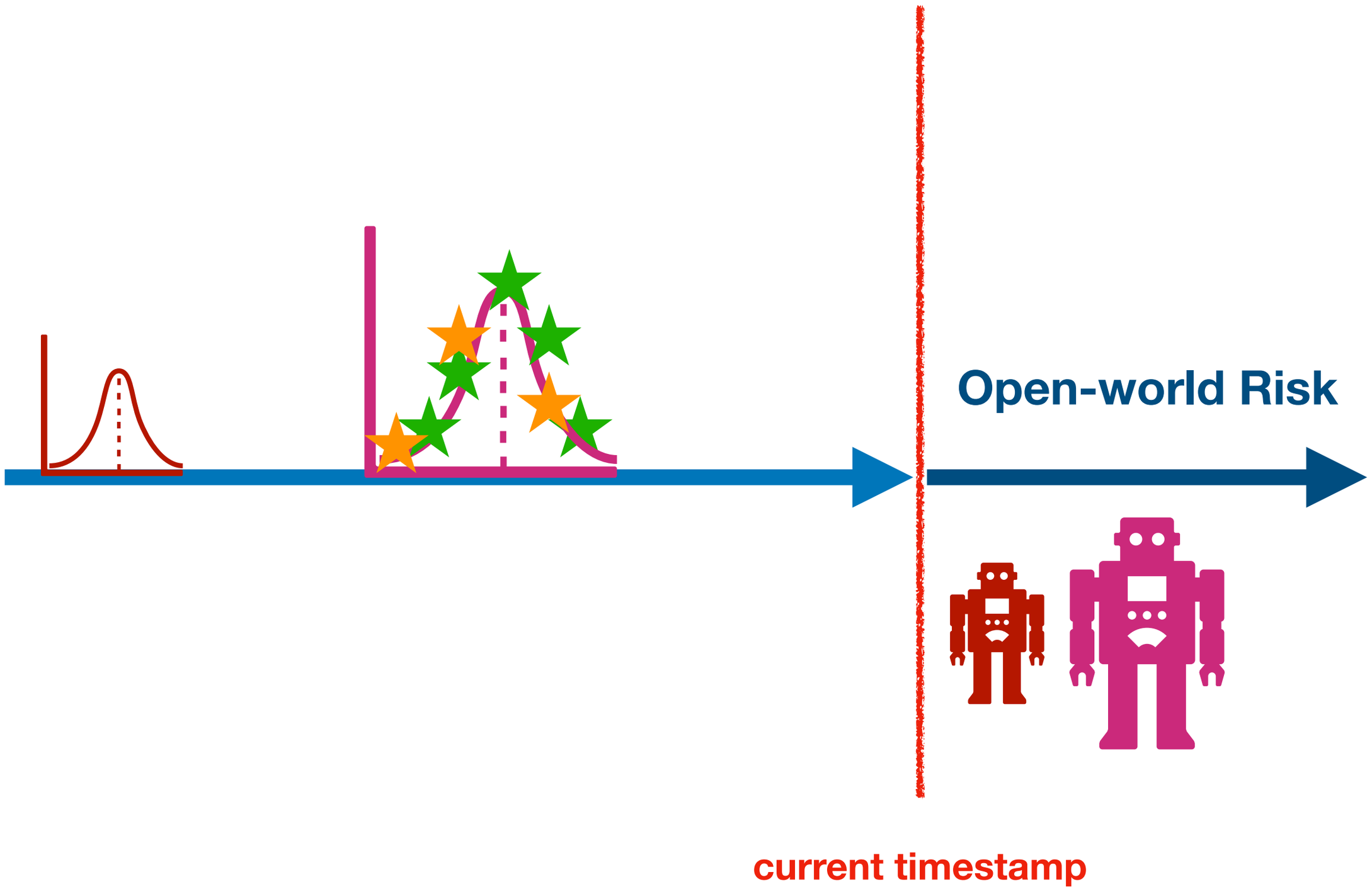
Motivation



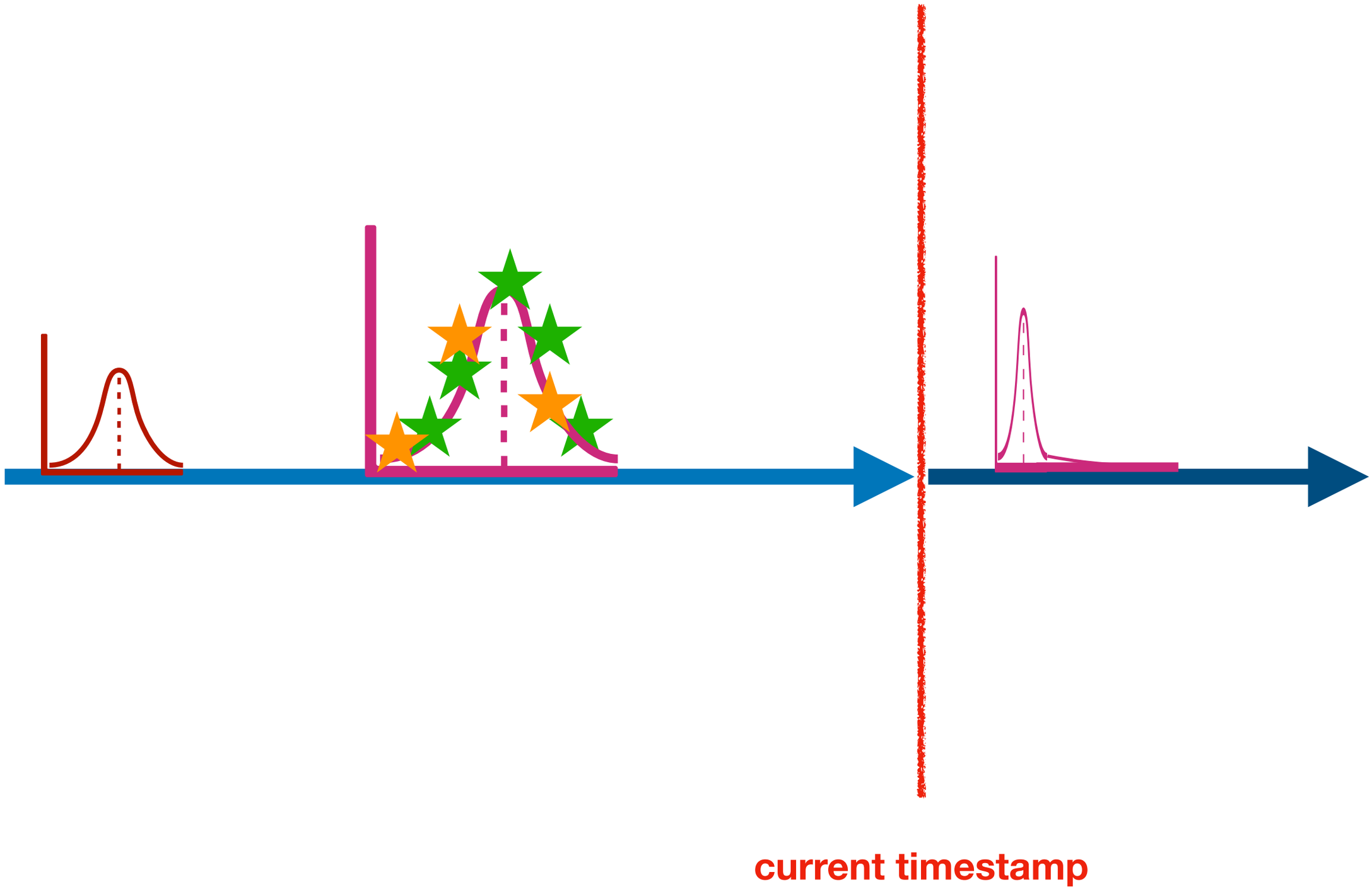
Motivation



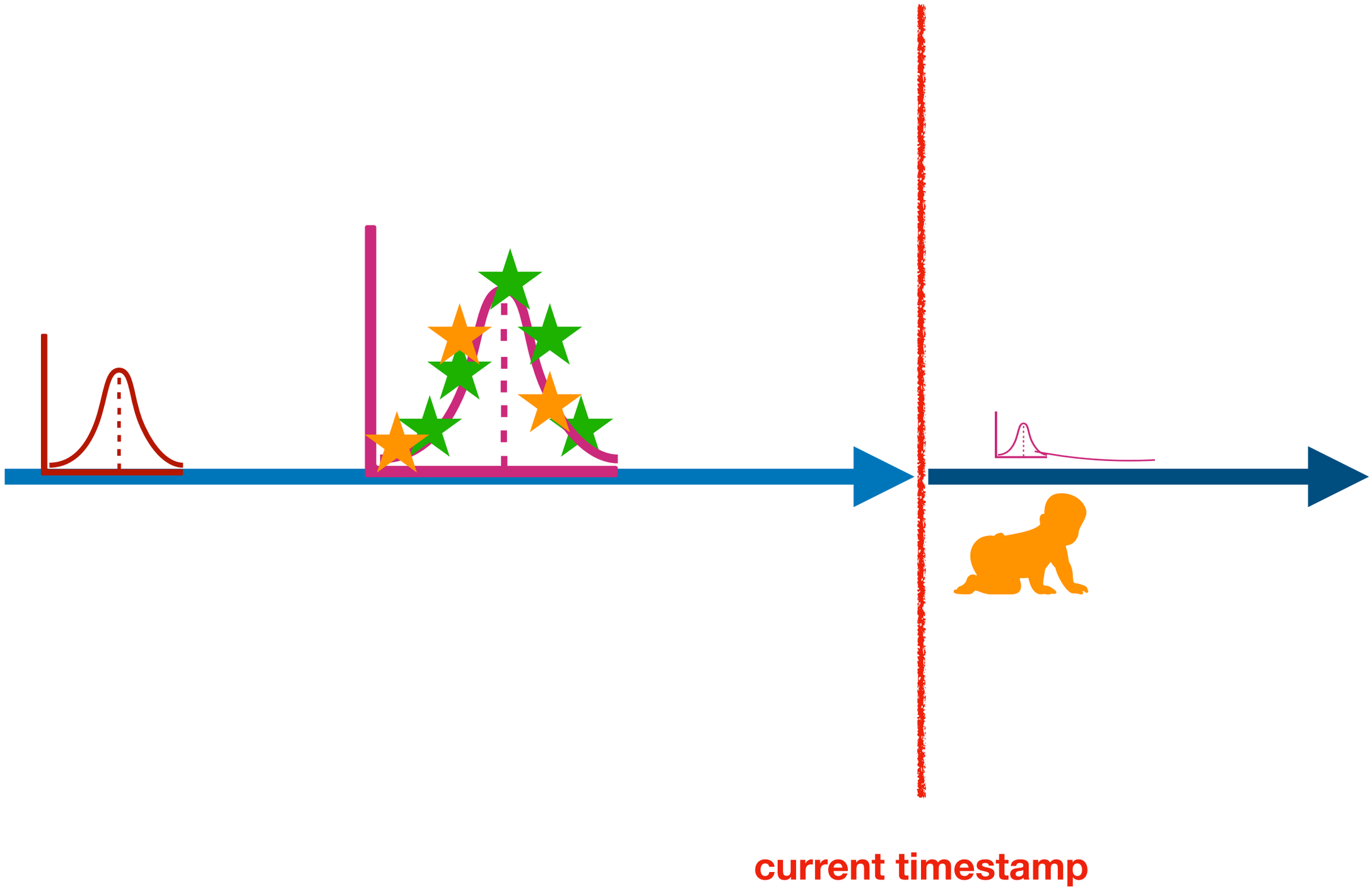
Motivation



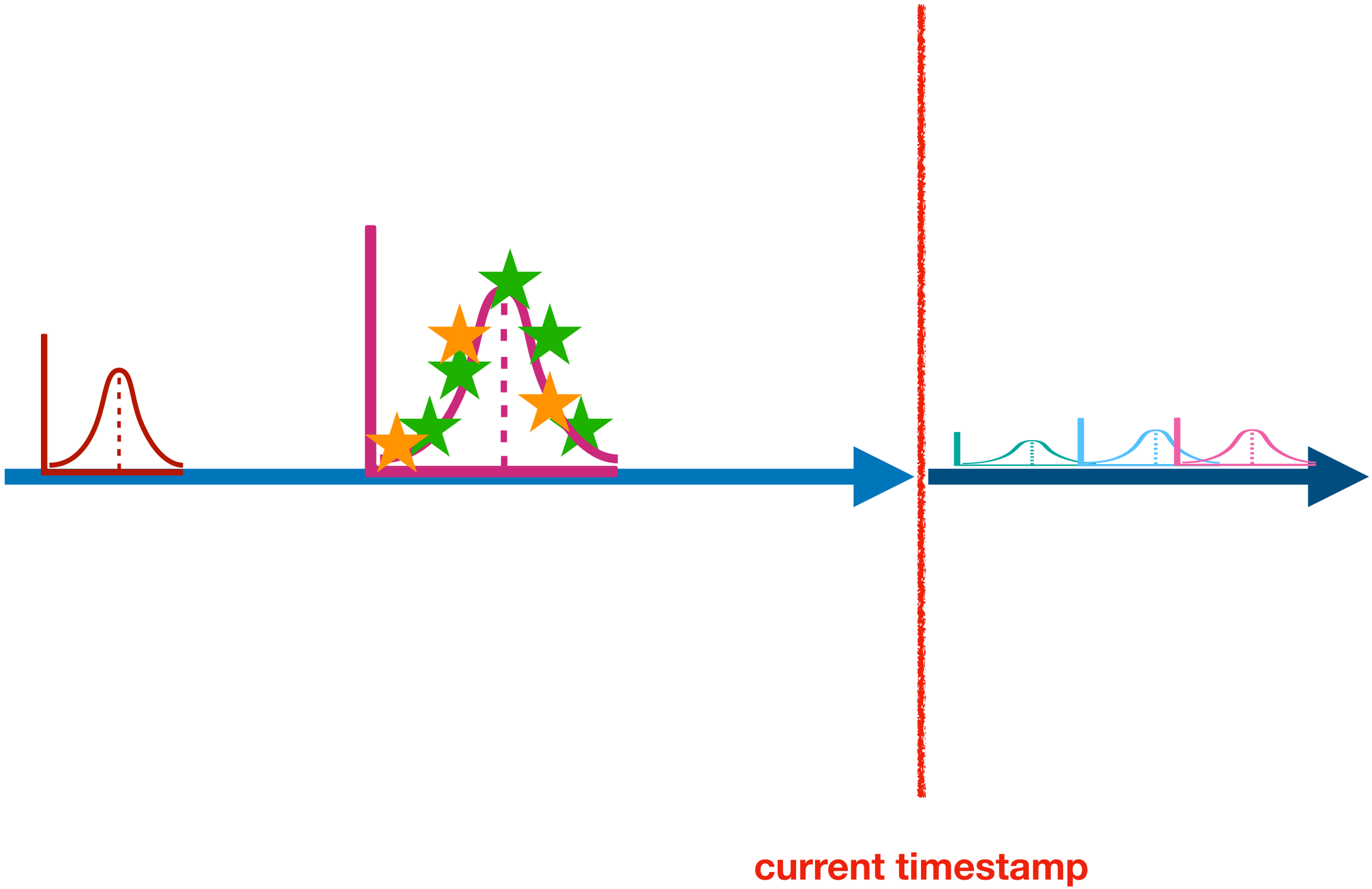
Motivation



Motivation



Motivation



Observation

- The world keeps evolving.

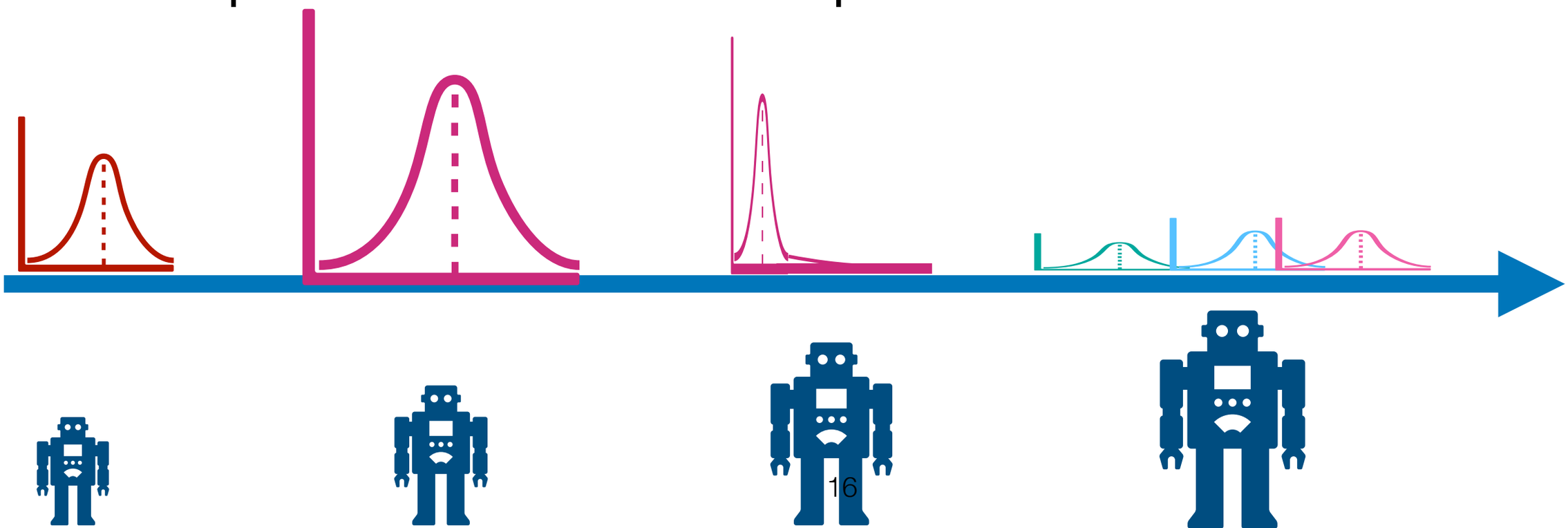
- Existing methods typically favor the majority examples from the existing world.

- But the changes of the world typically end with long-tailed **minority examples**, which greatly challenge existing models in future.

- Can we keep learning from the changes of the world?

Lifelong Representation Learning

- Lifelong learning (Thrun 1995, Chen and Liu 2016/2018) is a problem that aims to learn from a sequence of tasks.
  - online multi-task learning
- Lifelong representation learning aims to improve representations from a sequence of tasks.



Roadmap

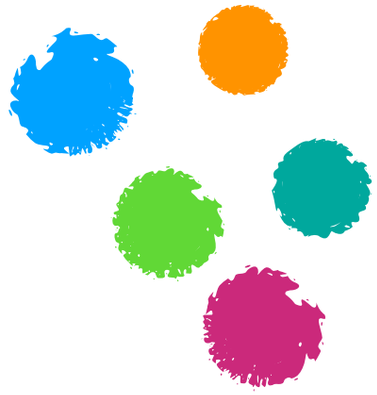
- Motivation
- Lifelong Supervised Learning
  - **Open-world Classification (WWW 2019)**
- Lifelong Self-supervised Learning
  - Domain Word Embedding (IJCAI 2018, ACL 2018)
  - Contextualized Domain Representation Learning

Roadmap

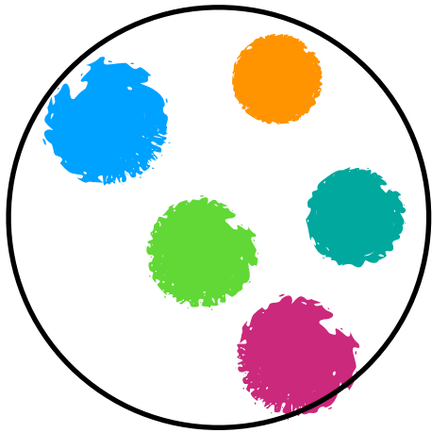
- Lifelong Supervised Learning

  - Open-world Learning and Application to Product Classification (WWW 2019)

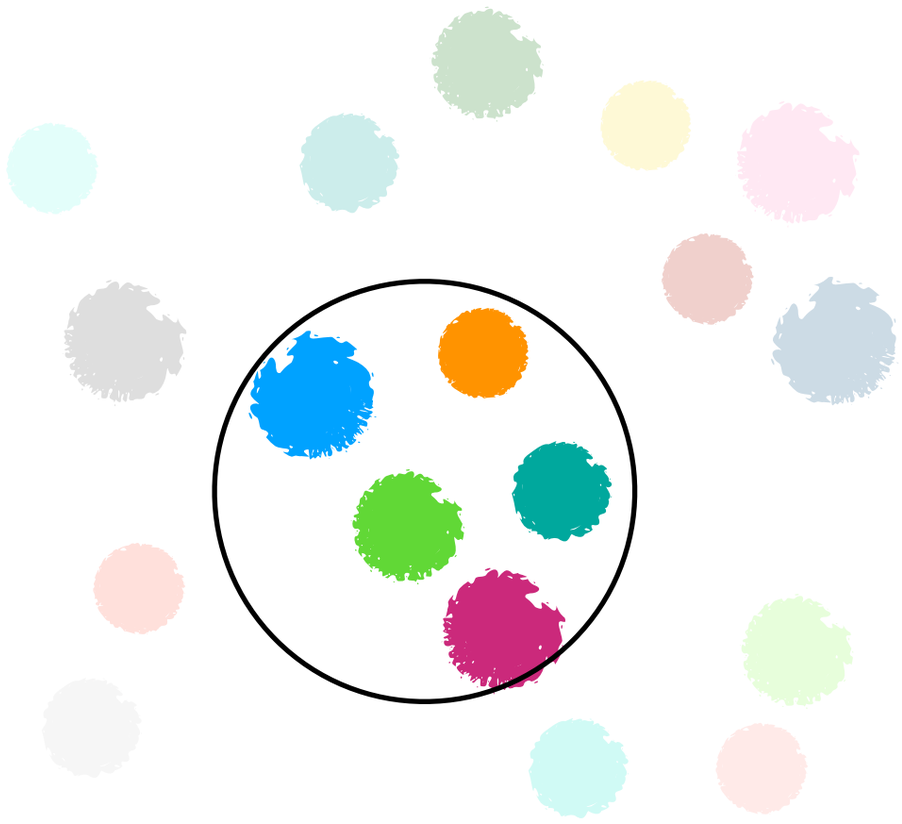
Motivation



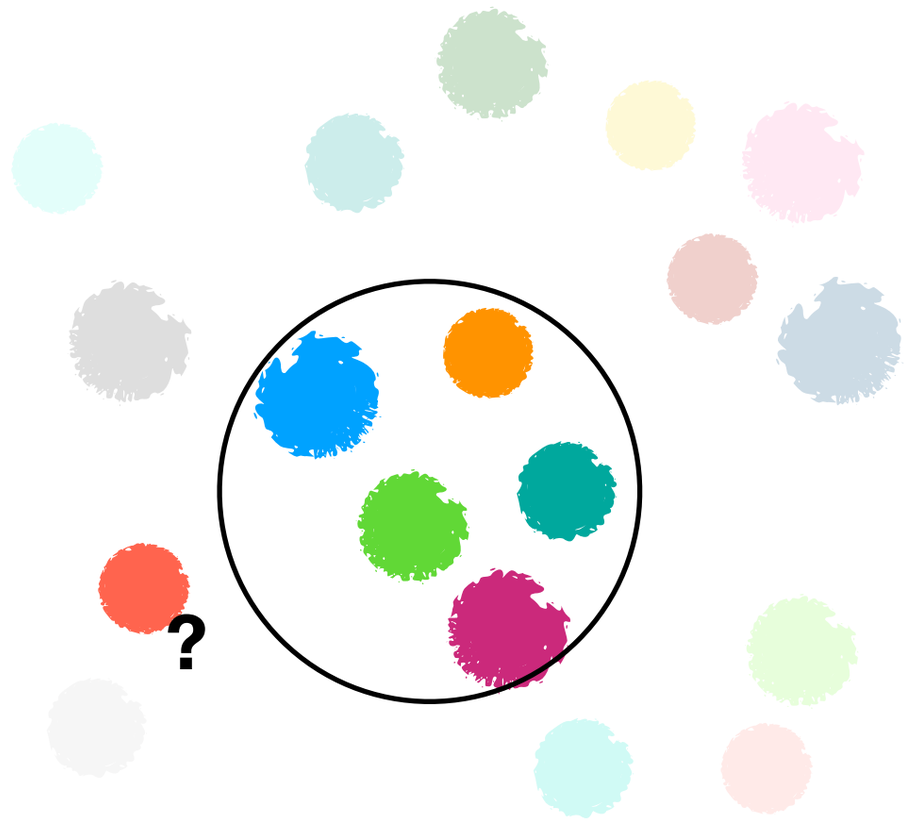
Motivation



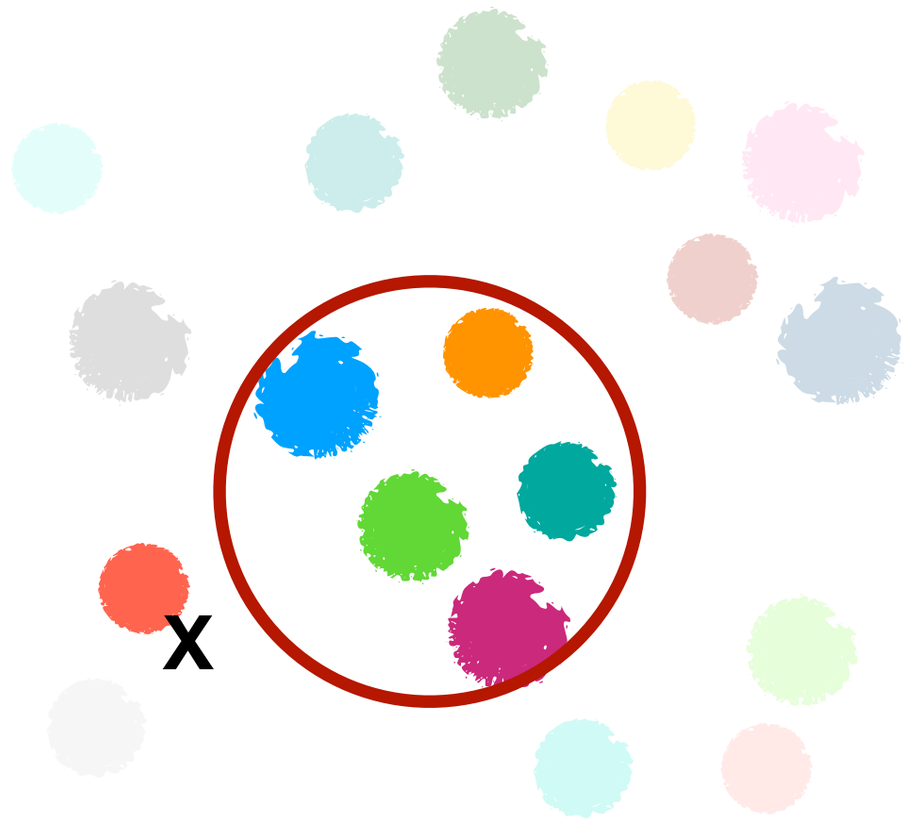
- Traditional classification tasks assume a **closed-world** setting:
    - human specified a set of seen classes that is static and never changed.



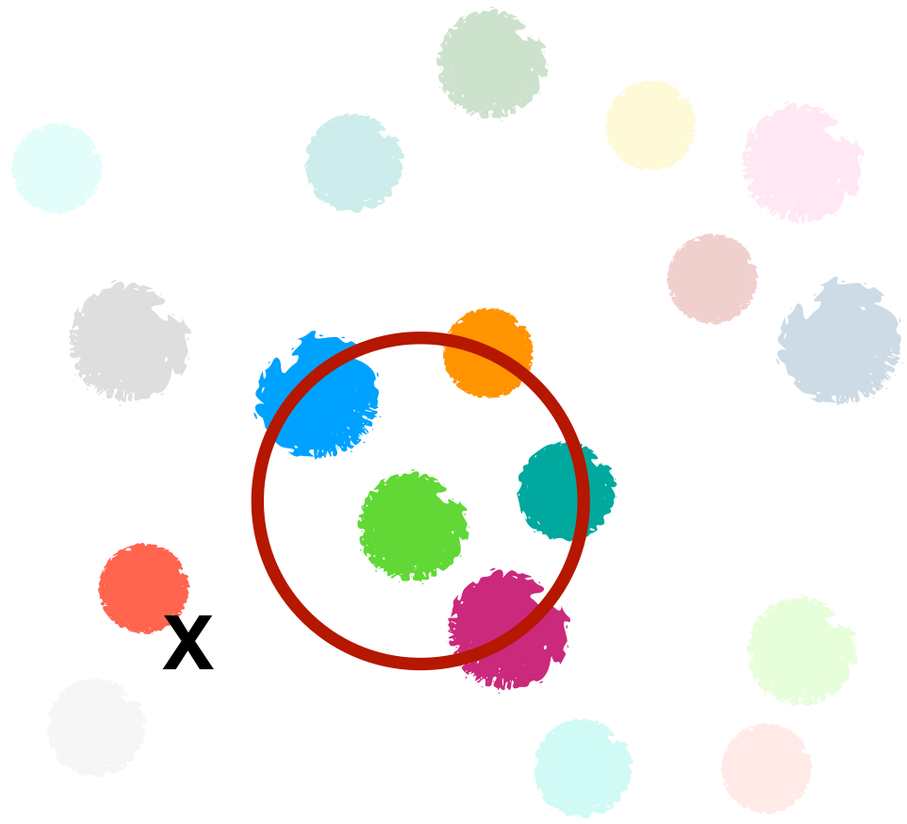
- When some new classes come in, ...



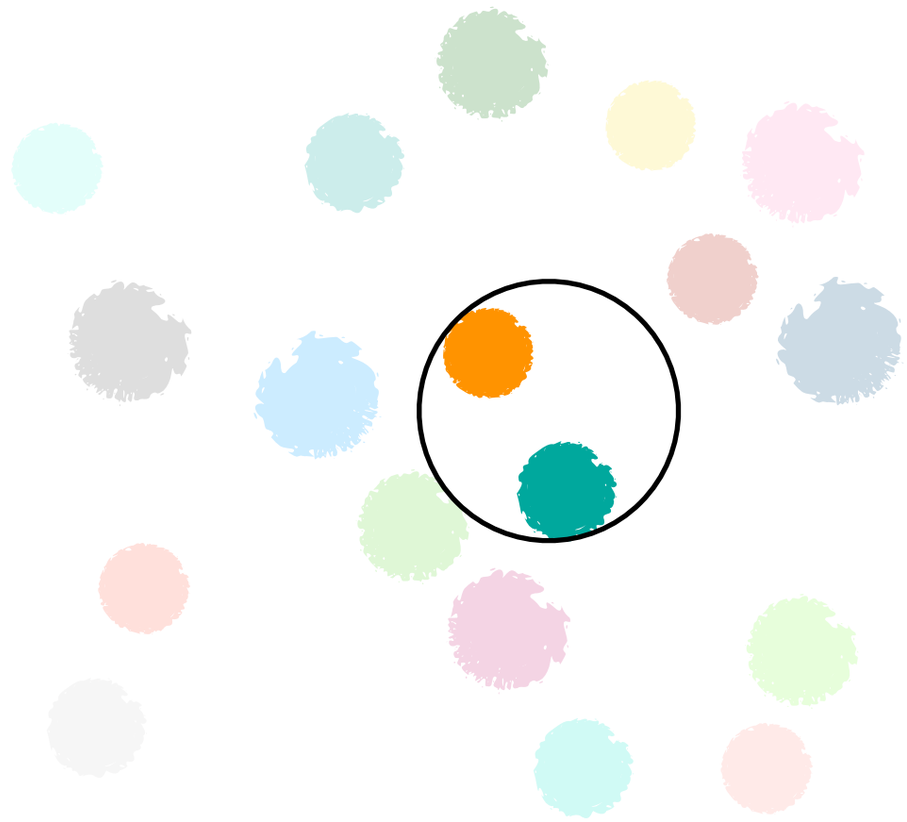
- When some new (unseen) classes come in:
  - It cannot handle it.
  - e.g., the default behavior of a softmax-based classifier forces an example from an unseen class to be one of seen classes.



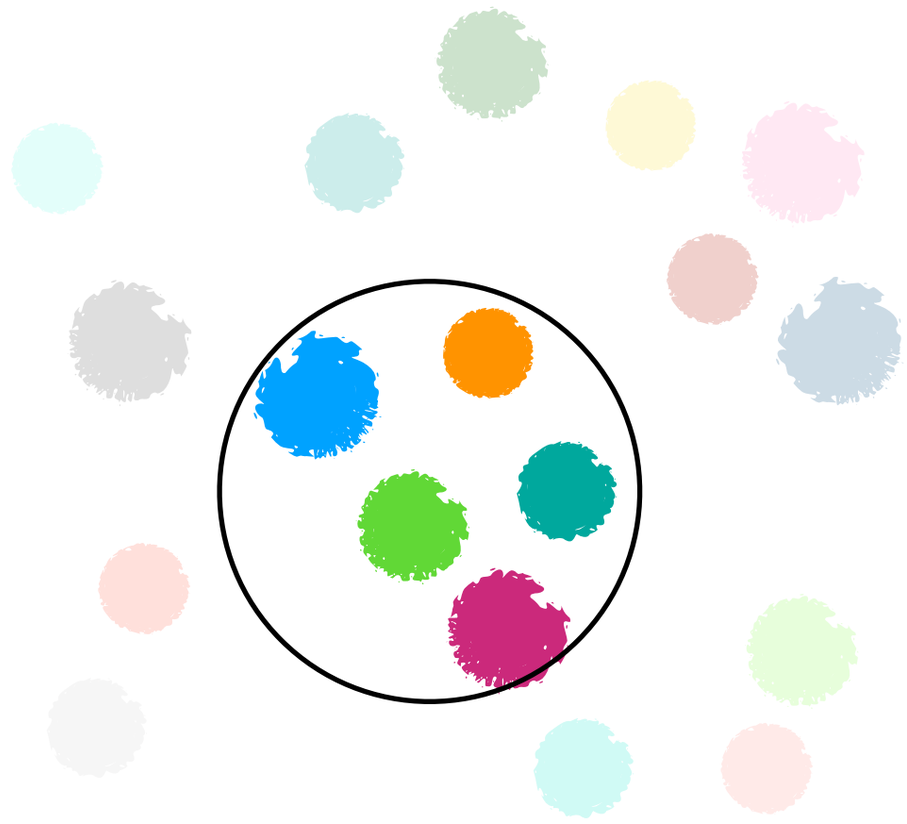
- First, the classifier is expected to be functionally correct on the seen set (Shu et al., 2017):
  - reject the examples from unseen classes.
  -



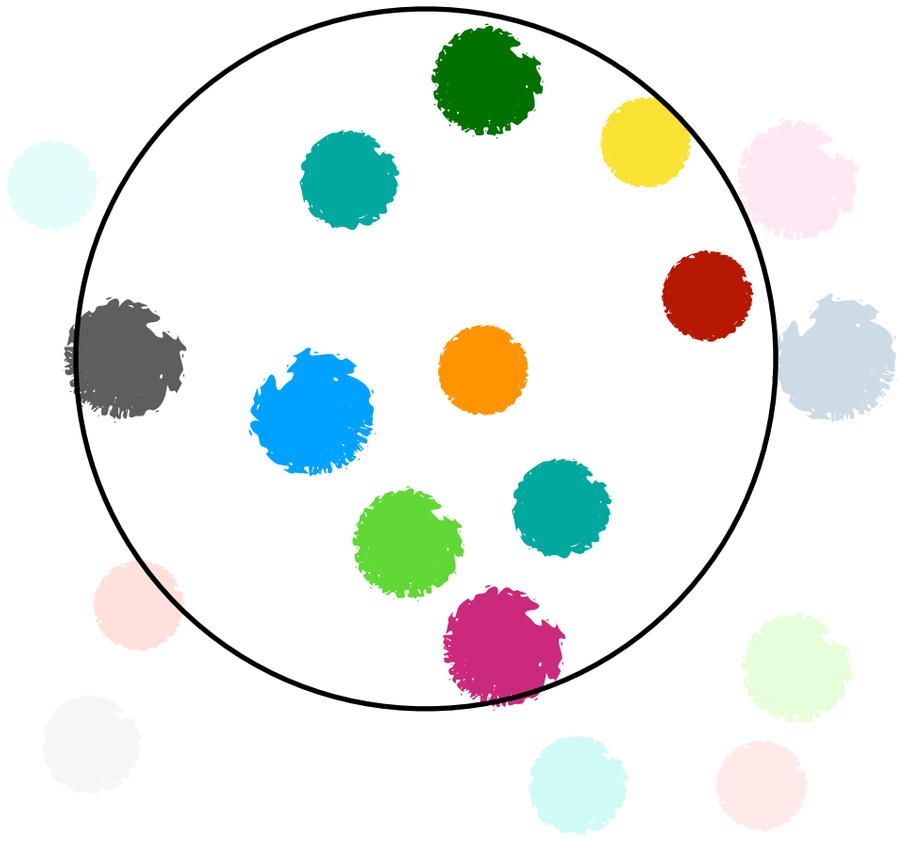
- First, the classifier is expected to be functionally correct on the seen set (Shu et al., 2017):
  - reject the examples from unseen classes.
  - but not on examples from a seen classes.



- Second, just making an old classifier functionally correct is not very useful for future.



- We may want the set of seen classes keeps growing / shrinking without retraining from scratch.
  - **but we still want examples from all unseen classes to be rejected to make the classifier functionally correct.**



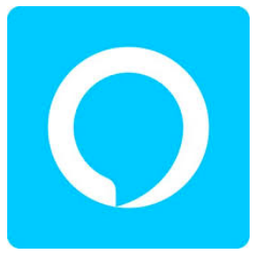
- Open-world learning (OWL) aims detect / reject examples of unseen classes and incrementally learn/accept (or remove) new classes.

Closed-world vs Open-world Classification

(Fei et al, 2016; Shu et al., 2017; Chen and Liu, 2016/2018 LML book)

- Traditional learning makes the **closed world assumption**:
  - Classes in testing have all been seen in training, no new class in testing.

- However, this classic learning paradigm is difficult to function in the real world open environment: new classes may come any time.



Problem Statement

Problem Statement: At any point in time, the learning system is aware of a set of seen classes $S = \{c_1, \ldots, c_m\}$ and has an OWL model/classifier for S but is unaware of a set of unseen classes $U = \{c_{m+1}, \ldots\}$ (any class not in S can be in U) that the model may encounter. The goal of an OWL model is two-fold: (1) classifying examples from classes in S and reject examples from classes in U, and (2) when a new class c_{m+1} (without loss of generality) is removed from U (now $U = \{c_{m+2}, \ldots\}$) and added to S (now $S = \{c_1, \ldots, c_m, c_{m+1}\}$, still being able to perform (1) without re-training the model.

Problem Statement

Problem Statement: At any point in time, the learning system is aware of a set of seen classes $S = \{c_1, \ldots, c_m\}$ and has an OWL model/classifier for S but is unaware of a set of unseen classes $U = \{c_{m+1}, \ldots\}$ (any class not in S can be in U) that the model may encounter. The goal of an OWL model is two-fold: (1) classifying examples from classes in S and reject examples from classes in U, and (2) when a new class c_{m+1} (without loss of generality) is removed from U (now $U = \{c_{m+2}, \ldots\}$) and added to S (now $S = \{c_1, \ldots, c_m, c_{m+1}\}$, still being able to perform (1) without re-training the model.

Problem Statement

Problem Statement: At any point in time, the learning system is aware of a set of seen classes $S = \{c_1, \ldots, c_m\}$ and has an OWL model/classifier for S but is unaware of a set of unseen classes $U = \{c_{m+1}, \ldots\}$ (any class not in S can be in U) that the model may encounter. The goal of an OWL model is two-fold: (1) classifying examples from classes in S and reject examples from classes in U, and (2) when a new class c_{m+1} (without loss of generality) is removed from U (now $U = \{c_{m+2}, \ldots\}$) and added to S (now $S = \{c_1, \ldots, c_m, c_{m+1}\}$, still being able to perform (1) without re-training the model.

Problem Statement

Problem Statement: At any point in time, the learning system is aware of a set of seen classes $S = \{c_1, \ldots, c_m\}$ and has an OWL model/classifier for S but is unaware of a set of unseen classes $U = \{c_{m+1}, \ldots\}$ (any class not in S can be in U) that the model may encounter. The goal of an OWL model is two-fold: (1) classifying examples from classes in S and reject examples from classes in U, and (2) when a new class c_{m+1} (without loss of generality) is removed from U (now $U = \{c_{m+2}, \ldots\}$) and added to S (now $S = \{c_1, \ldots, c_m, c_{m+1}\}$, still being able to perform (1) without re-training the model.

Main Contribution

- Open-world learning (OWL) problem (classification) over a dynamic set of classes.

- A meta-learning framework that aims to learn cross-class (cross-task) representations to solve OWL.

Classification as Humans

- When working on a dynamic set of classes (e.g., # classes > 10), we humans perform comparison-based classification.
  - without learning new classes ahead, we can do classification on examples from new unseen classes.
  - we essentially have a kNN classifier in brain: train a general comparator for any class (Meta-Learning), apply it to concrete data of new classes on the fly.
- Meta-learning could be one way to solve OWL.

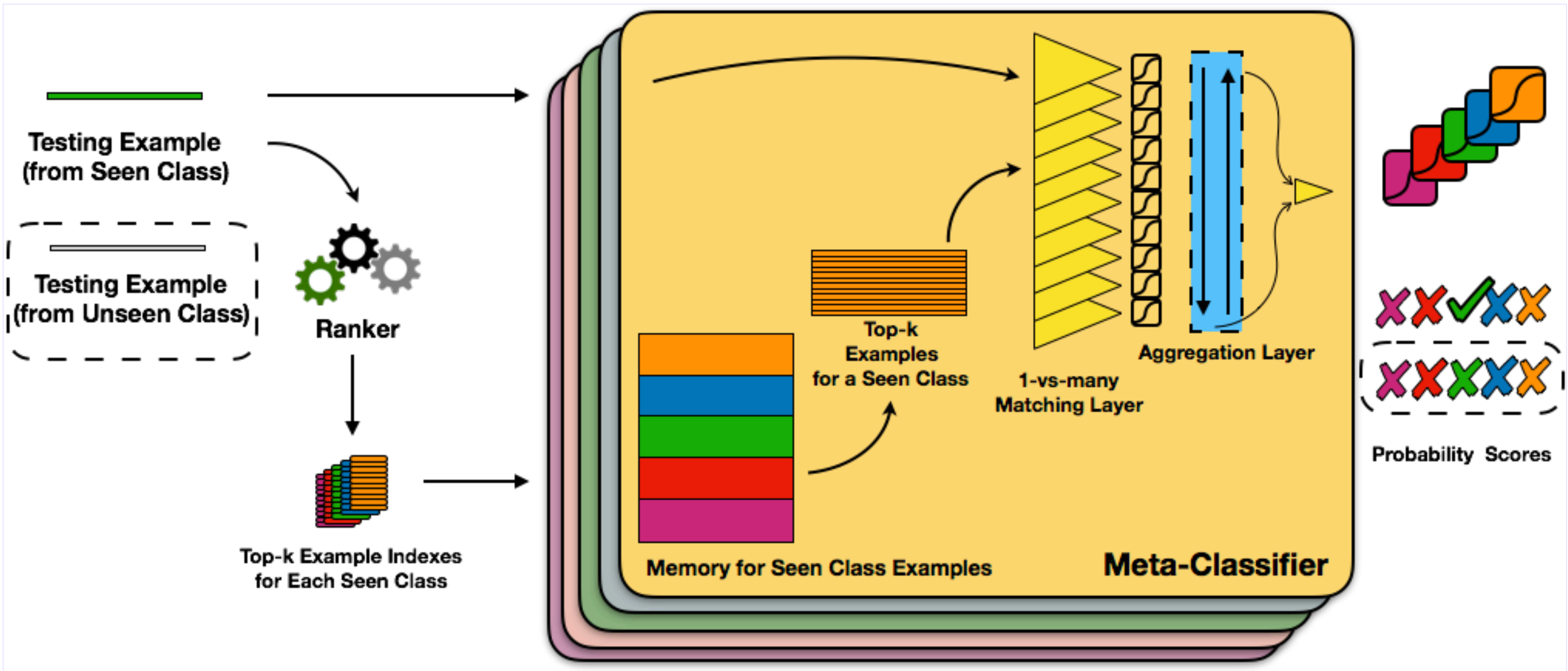
Learning to Accept new Classes (L2AC Framework)

- It maintains a dynamic set S of seen classes that allow new classes to be added / deleted with no model re-training.

- Each class is represented by a small set of training examples.

- In testing, the meta-classifier uses only the examples of the maintained seen classes so far on-the-fly for classification and rejection.

Framework


Testing Example
(from Seen Class)
Testing Example
(from Unseen Class)
Ranker
Top-k Example Indexes
for Each Seen Class
Top-k
Examples
for a Seen Class
1-vs-many
Matching Layer
Aggregation Layer
Memory for Seen Class Examples
Meta-Classifier
Probability Scores

Meta Classifier

- We train a meta classifier:
  - Assume we have an embedding for each document.
    - We first compute relevance score between a pair of embeddings from two documents:

$$r_i = f(x_t, x_{a_i}) = \sigma\Big(W_2 \cdot \mathrm{Relu}\big(W_1 \cdot f_{\mathrm{sim}}(x_t, x_{a_i}) + b_1\big) + b_2\Big).$$

    - We adopt two similarity functions to aggregate the pair of embeddings.

$$f_{\mathrm{sim}}(x_t, x_{a_i}) = f_{\mathrm{abssub}}(x_t, x_{a_i}) \oplus f_{\mathrm{sum}}(x_t, x_{a_i})$$

Meta Classifier

- Then we aggregate all those relevant scores for k examples for each class:

$$p(c|x_t, x_{a_{1:k}}) = \sigma\big(W \cdot \mathrm{BiLSTM}(r_{1:k}) + b\big).$$

- This is essentially a parametric kNN with parametric (trainable) voting mechanism.

Open-world Classification

$$\hat{y} = \begin{cases} \textit{reject}, \text{ if } \max_{c \in S} p(c|x_t, x_{a_{1:k}}) \leq 0.5; \\ \arg\max_{c \in S} p(c|x_t, x_{a_{1:k}}), \text{ otherwise.} \end{cases}$$

- We balance the weights of positive / negative examples during meta training.

Meta Training

- We perform meta training on a holdout set of classes.
  - **Positive examples**: sample one example xq and its kNN examples from the same class c. $(x_q, x_{a_{1:k}} | x_q, c)$
- We choose a set **n** (|M| - 1) NN negative classes for each xq (via cosine similarity to class center).
  - **Negative examples**: xq with kNN examples from a negative class c'.

$$(x_q, x_{a_{1:k}} | x_q, c'), \quad c' \in M \backslash c$$

Experiment

- Dataset: Amazon Dataset (Julian et al., 2016).
- We formulate a task of product classification on description.
-

Meta Training / Testing over Different Set of Classes

Classes for Document Encoder: 1000

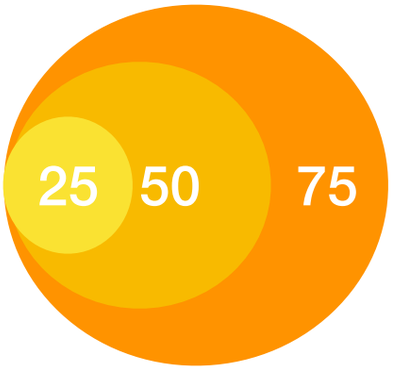
Classes for Meta-Training: 900

Classes for Meta Validation: 100

Classes Testing: 100

Examples for Baseline Training/L2AC Reading: 50%

Examples for Testing: 50%



Experiment

- Dataset: Amazon Dataset (Julian et al., 2016).
  - We formulate a task of product classification on description.

Evaluation Metrics

- Evaluation Metrics:
- Macro F1.
- Weighted F1. $$\sum_{c \in S \cup \{c_{\text{rej}}\}} \frac{N_c}{\sum_{c \in S \cup \{c_{\text{rej}}\}} N_c} \cdot \text{F1}_c,$$

Compared Methods

- DOC (Shu et al., 2017) and variants: retrain DOC for different sets of seen classes.
  - DOC-CNN: DOC (t=0.5) in original DOC paper.
  - DOC-LSTM: DOC with LSTM
  - DOC-Enc: use our pre-trained encoder
  - DOC-*-Gaus: above 3 baselines with gaussian fitting:
    - DOC-CNN-Gaus is DOC in original DOC paper.

Compared Methods

- L2AC and variants:
  - L2AC-n9-NoVote: k=1, n=9.
  - L2AC-n9-Vote3: use kNN manual voting on k=3.
  - L2AC-k5-n9-AbsSub/Sum: ablation study on relevance function.
  - L2AC-k5-n9/14/19: k=5, n=9, 14, 19.

Hyper-parameters on Meta Validation Set

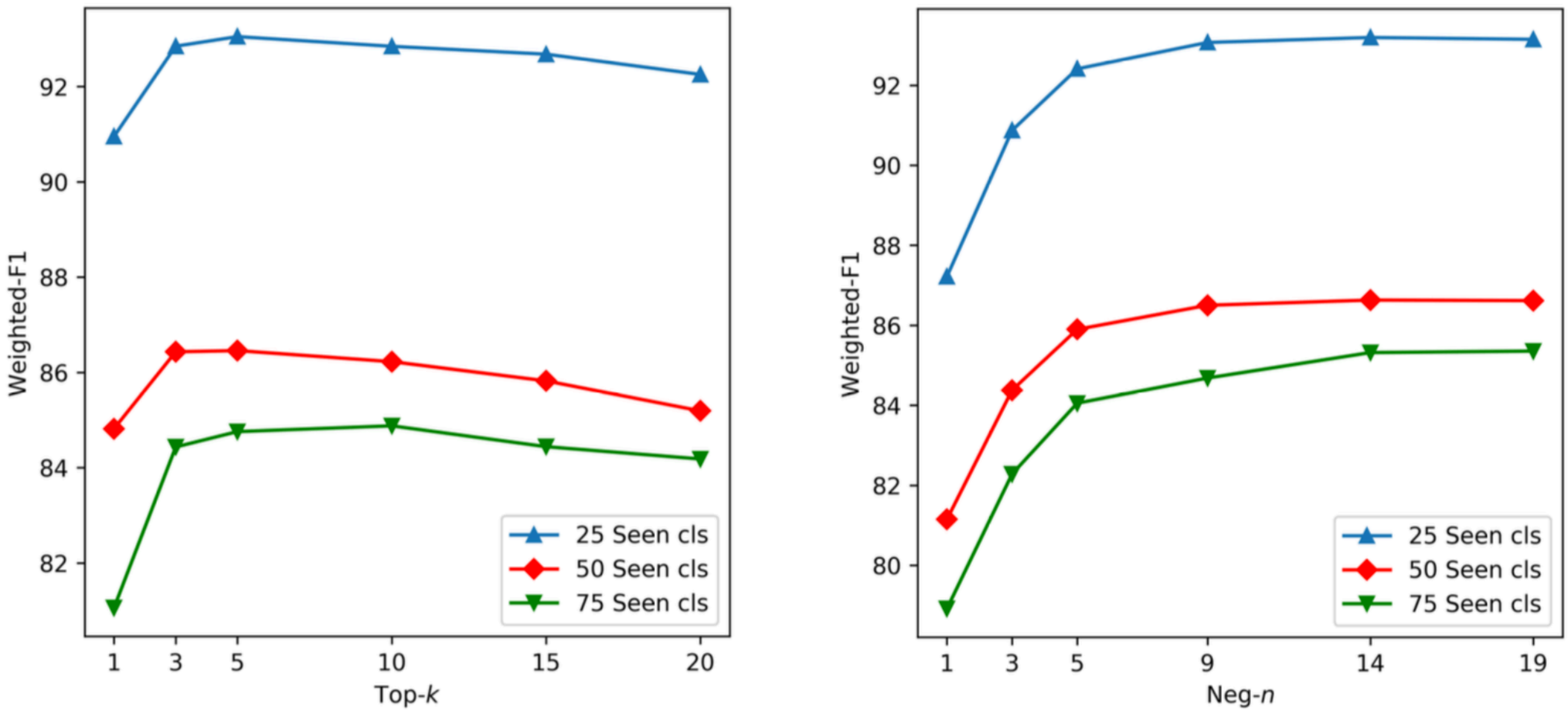


Figure 2: Weighted F1 scores for different k's ($n = 9$) and different n's ($k = 5$).

Results on 10 runs

Methods	$\|S\| = 25$ (WF1)	$\|S\| = 25$ (MF1)	$\|S\| = 50$ (WF1)	$\|S\| = 50$ (MF1)	$\|S\| = 75$ (WF1)	$\|S\| = 75$ (MF1)
DOC-CNN	53.25(1.0)	55.04(0.39)	70.57(0.46)	76.91(0.27)	81.16(0.47)	86.96(0.2)
DOC-LSTM	57.87(1.26)	57.6(1.18)	69.49(1.58)	75.68(0.78)	77.74(0.48)	84.48(0.33)
DOC-Enc	82.92(0.37)	75.09(0.33)	82.53(0.25)	84.34(0.23)	83.84(0.36)	88.33(0.19)
DOC-CNN-Gaus	85.72(0.43)	76.79(0.41)	83.33(0.31)	83.75(0.26)	84.21(0.12)	87.86(0.21)
DOC-LSTM-Gaus	80.31(1.73)	70.49(1.55)	77.49(0.74)	79.45(0.59)	80.65(0.51)	85.46(0.25)
DOC-Enc-Gaus	88.54(0.22)	80.77(0.22)	84.75(0.21)	85.26(0.2)	83.85(0.37)	87.92(0.22)
L2AC-n9-NoVote	91.1(0.17)	82.51(0.39)	84.91(0.16)	83.71(0.29)	81.41(0.54)	85.03(0.62)
L2AC-n9-Vote3	91.54(0.55)	82.42(1.29)	84.57(0.61)	82.7(0.95)	80.18(1.03)	83.52(1.14)
L2AC-k5-n9-AbsSub	92.37(0.28)	84.8(0.54)	85.61(0.36)	84.54(0.42)	83.18(0.38)	86.38(0.36)
L2AC-k5-n9-Sum	83.95(0.52)	70.85(0.91)	76.09(0.36)	75.25(0.42)	74.12(0.51)	78.75(0.57)
L2AC-k**5**-n**9**	<u>93.07</u>(0.33)	86.48(0.54)	<u>86.5</u>(0.46)	85.99(0.33)	<u>84.68</u>(0.27)	88.05(0.18)
L2AC-k5-n14	**93.19**(0.19)	86.91(0.33)	**86.63**(0.28)	86.42(0.2)	85.32(0.35)	88.72(0.23)
L2AC-k5-n19	93.15(0.24)	86.9(0.45)	86.62(0.49)	86.48(0.43)	**85.36**(0.66)	88.79(0.52)

le 1: Weighted F1 (WF1) and macro F1 (MF1) scores on a test set with 100 classes with 3 settings: 25, 50, and 75 seen clas set of seen classes are incrementally expanded from 25 to 75 classes (or gradually shrunk from 75 to 25 classes). The res the averages over 10 runs with standard deviations in parenthesis.

Conclusion

- We propose a challenging problem: open-world learning (classification), which performs classification / rejection on a dynamic set of classes.

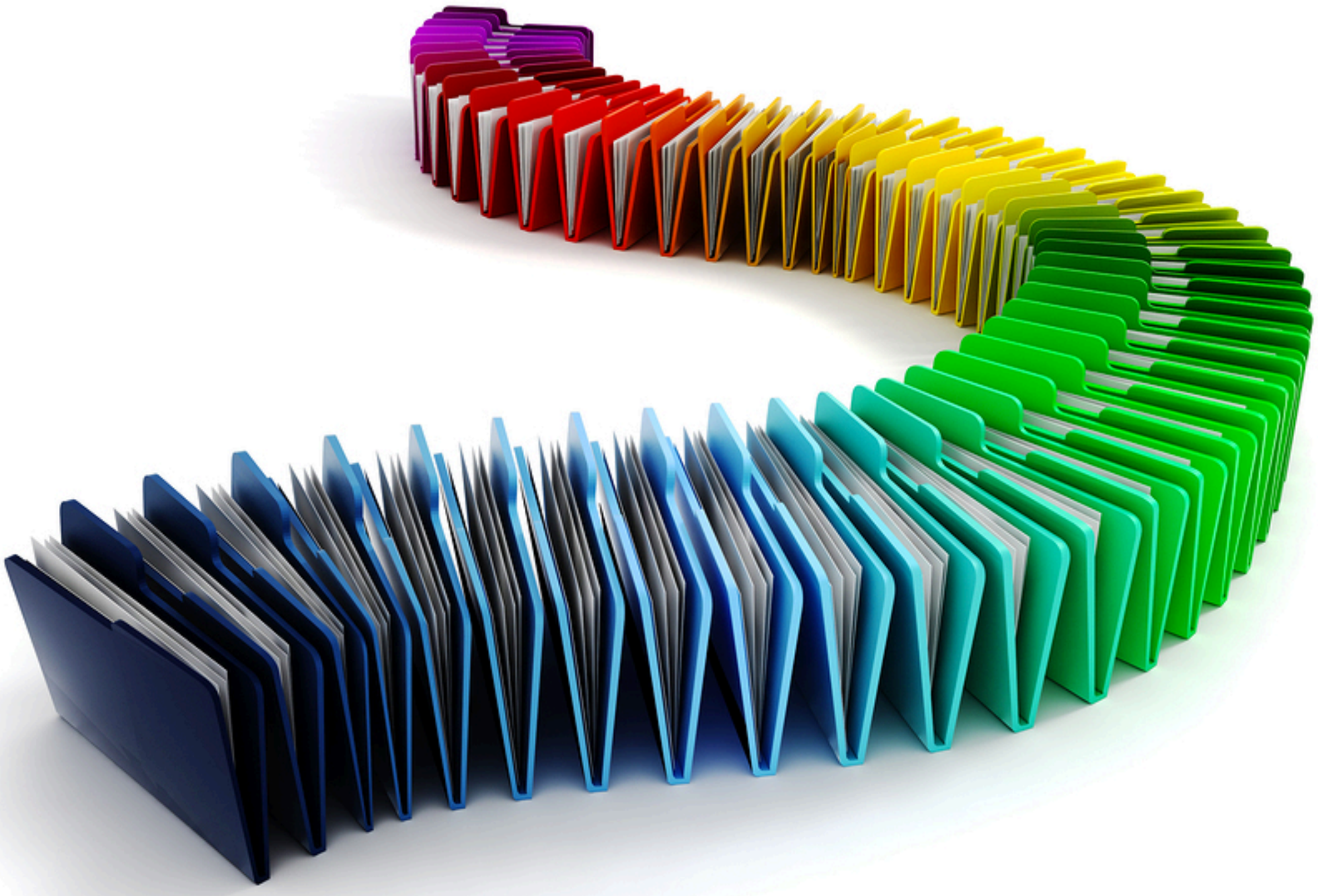
- We propose a meta-learning based method to learn cross-class representations and metrics for detecting intra / inter class examples.

- The method requires no training during testing on a dynamic pool of classes.

Roadmap

- Motivation
- Lifelong Supervised Learning
  - Open-world Classification (WWW 2019)
- Lifelong Self-supervised Learning
  - **Domain Word Embedding (IJCAI 2018, ACL 2018)**
  - Contextualized Domain Representation Learning

Motivation



Motivation

- General-purpose embeddings (fastText, GloVe) are trained on large-scale corpora, which is assumed to cover all domains.
- They lack domain-specific knowledge, especially **new domains with small corpora**.
    - They tend to mix the knowledge of one word from different domains: a small domain will be biased.
- Training domain-specific embeddings typically lacks large in-domain corpora.

Problem

- Lifelong Learning for Domain Word Embedding:

Problem statement: We assume that the learning system has seen n domain corpora in the past: $D_{1:n} = \{D_1, \ldots, D_n\}$, when a new domain corpus D_{n+1} comes with a certain task, the system automatically generates word embeddings for the $(n+1)$-th domain by leveraging some useful information or knowledge from the past n domains.

- We focus on lifelong data augmentation for the (n+1)-th domain by leveraging the past n domains corpora.

Main Contribution

- Lifelong Learning problem for domain word embeddings.

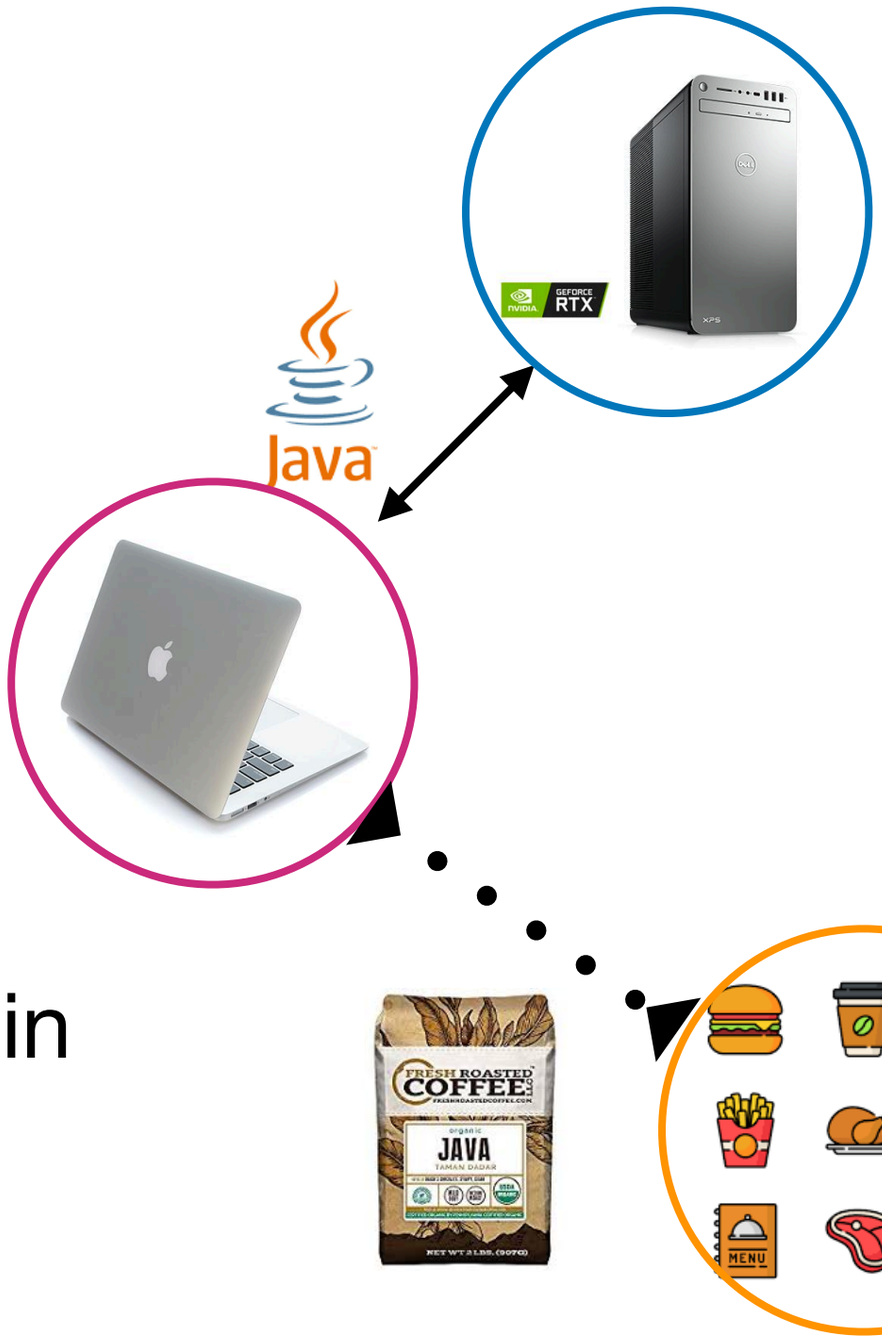
- A meta-learner to learn cross-domain context similarity for a word.

Ideas

Domains are not totally isolated.

- A word in one domain could be similar (or dissimilar) to the word in another domain.

Ideas

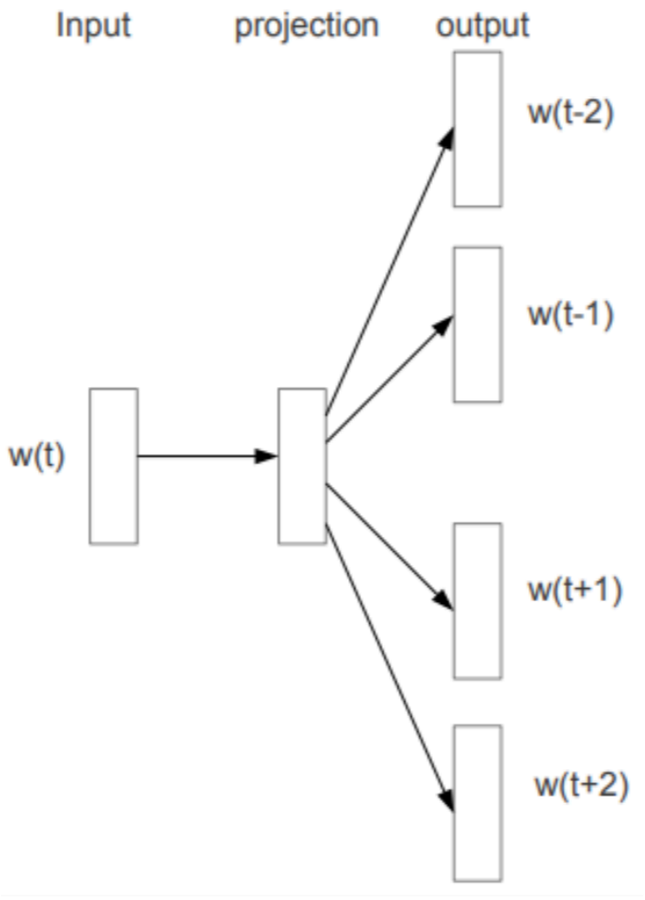


Domains are not totally isolated.

- A word in one domain could be similar (or dissimilar) to the word in another domain.

Ideas

- In the skip-gram model:
  - a positive training example is a **word** (input) and its contexts (output).



- For a laptop domain,
  - we may like examples from "It's excellent for **java** programming" (desktop),
  - but NOT "The **java** coffee from Starbucks is good" (food).

Challenge

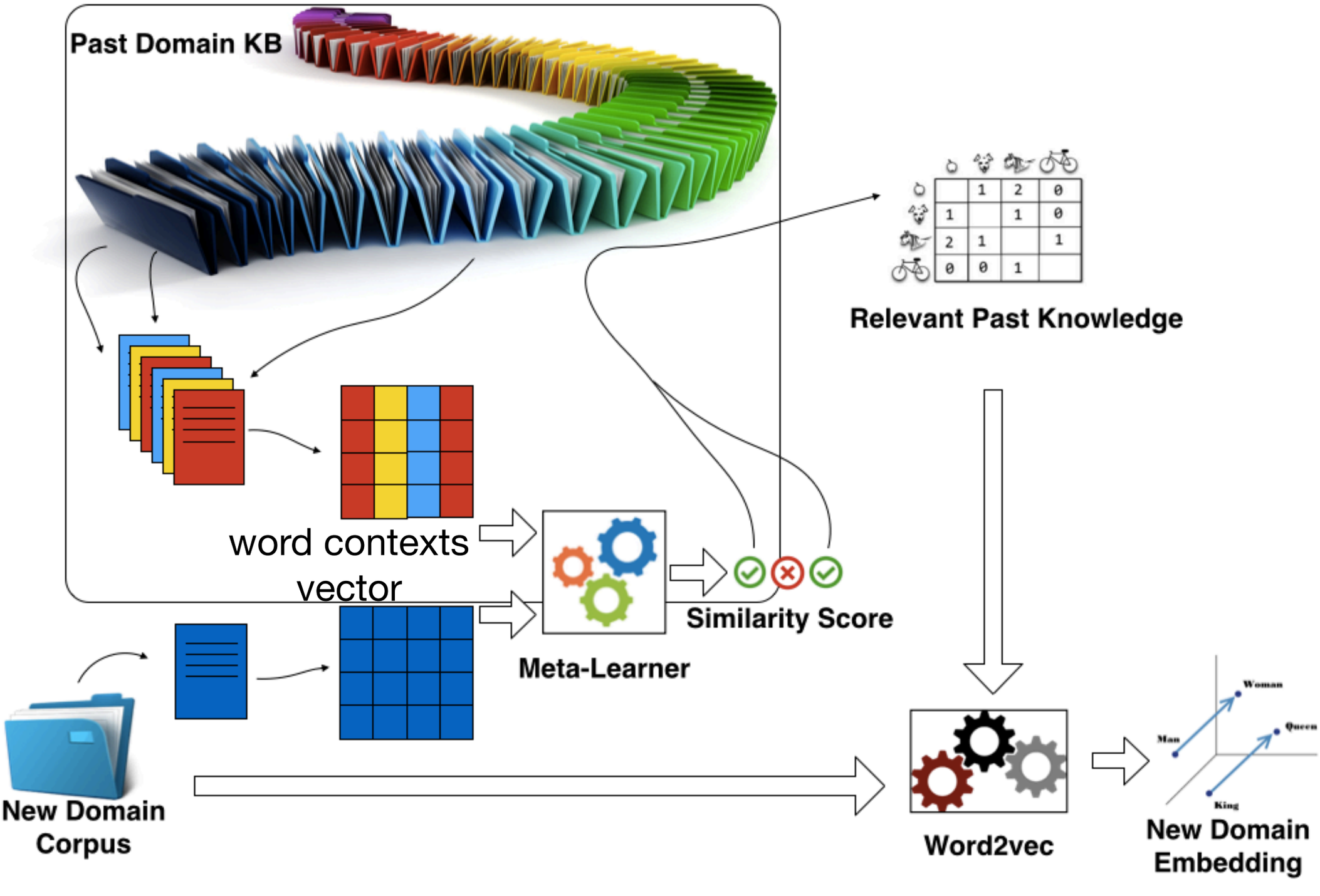
- How to identify **relevant knowledge or contexts** (from a past domain corpus) for the current domain?

- How to **automatically** do that (without human annotation)?

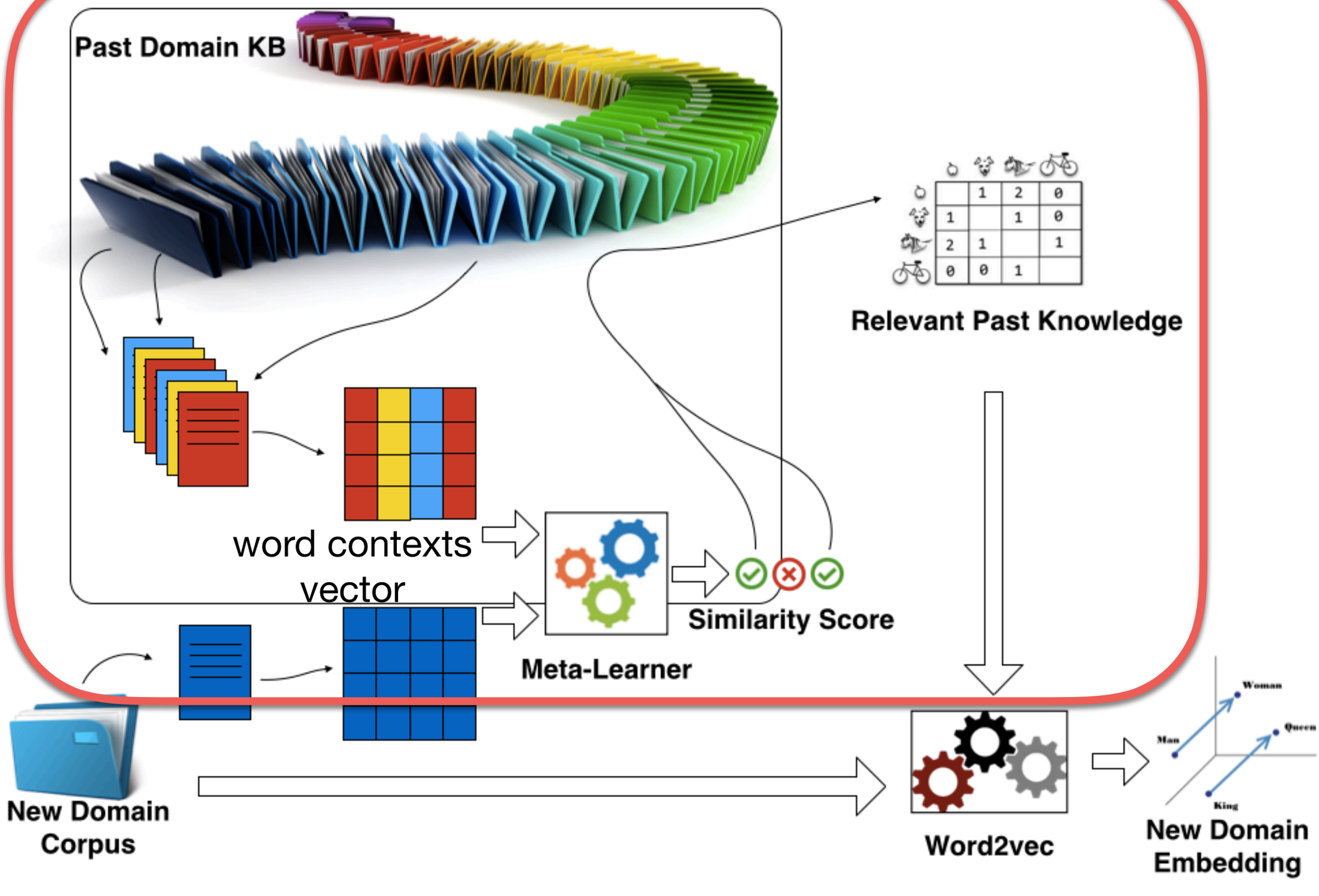
L-DEM

- We propose a meta-learning based framework:
  - L-DEM (Lifelong Domain Embedding via Meta-learning):
- A meta-learner:
  - predict and carry relevant past knowledge (a word and its contexts) to augment the new domain corpus.

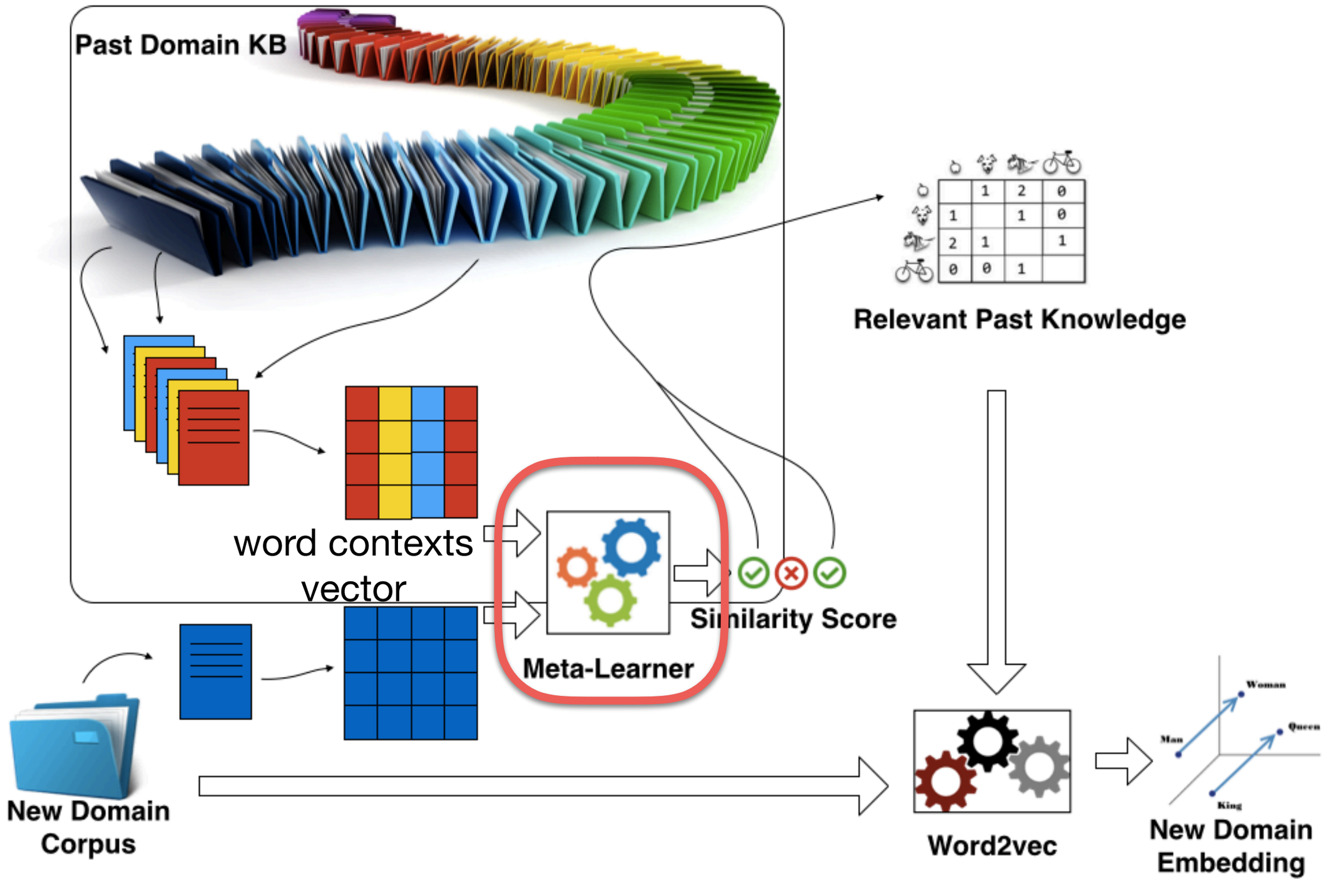
L-DEM


Past Domain KB
Relevant Past Knowledge
word contexts vector
Similarity Score
Meta-Learner
New Domain Corpus
Word2vec
Woman
Man
Queen
King
New Domain Embedding

L-DEM


Past Domain KB
Relevant Past Knowledge
word contexts vector
Meta-Learner
Similarity Score
New Domain Corpus
Word2vec
Woman
Man
Queen
King
New Domain Embedding

L-DEM


Past Domain KB
Relevant Past Knowledge
word contexts vector
Meta-Learner
Similarity Score
New Domain Corpus
Word2vec
Woman
Man
Queen
King
New Domain Embedding

Meta-Learner

- A **word-level cross-domain** meta-learner to identify relevant past knowledge.
  - Input: a pair of word contexts vectors (for the same word);
  - Output: whether they are from similar domains or not.
- Relevant Past Knowledge: a **word** and its contexts in a past domain corpus.

Word Contexts Vector

- a TF vector built from all contexts of a word in a (sub-sampled) domain corpus.
  - contexts: words within a 10 words sliding window.

$$\mathbf{x}_{w_{i,j,k}}$$

- The word indexed by j in the k-th sub-sampled corpus of the i-th domain.

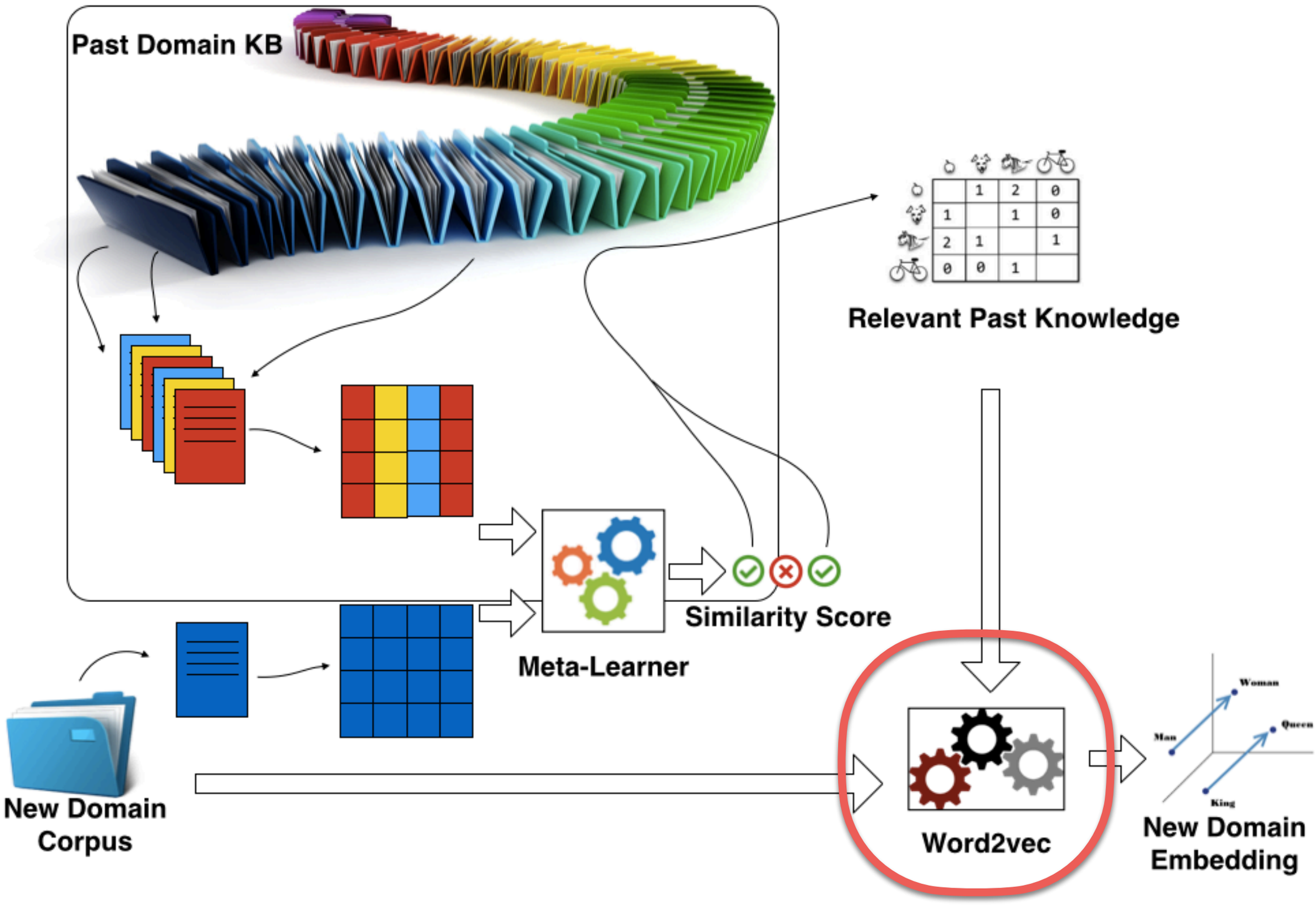
Meta-Learner

- Architecture: a siamese network.

$$\sigma\big(\boldsymbol{W}_2 \cdot \mathrm{abs}\big((\boldsymbol{W}_1 \cdot \frac{\mathbf{x}_{w_{i,j,k}}}{|\mathbf{x}_{w_{i,j,k}}|_1}) - (\boldsymbol{W}_1 \cdot \frac{\mathbf{x}_{w_{i,j',k'}}}{|\mathbf{x}_{w_{i,j',k'}}|_1})\big) + b_2\big), \quad (1)$$

- This network is trained from a holdout m domains.
  - positive examples: two sub-exampled corpora from the same domain;
  - negative examples: two sub-exampled corpora from different domains.

L-DEM


Past Domain KB
Relevant Past Knowledge
Similarity Score
Meta-Learner
New Domain Corpus
Word2vec
Woman
Queen
Man
King
New Domain Embedding

Augmented Embedding Training

- A modified skip-gram that takes two sources:
  - the current domain corpus and relevant past knowledge ***A*** (word contexts).
  -

$$\mathcal{L}'_{D_{n+1}} = \mathcal{L}_{D_{n+1}} + \mathcal{L}_{\mathcal{A}}. \quad (4)$$

Experiments

- Datasets:
  - Amazon Review datasets (He and McAuley, 2016), which is a collection of multiple-domain corpora.

    - We take the 2nd level category as a domain.

Different Sets of Domains

Domains for Meta-Learner Train/Valid/Test: 39/5/12

Past Domains: 50/100/200

Domains for End tasks: 3

Domains for End Tasks

- Computer Components (CC): 13 classes.
- Kitchen Storage and Organization (KSO): 17 classes.
- Cats Supply (CS): 11 classes.
- Each domain is sampled with 2 sizes of in-domain corpus: 10 MB and 30 MB.

Training of Meta-Learner

- We randomly select **2000**, **500** and **1000** words from each **training**, **validation**, and **testing** domain of meta-learner.

Evaluation of Meta-Learner

- F1-score of the proposed base meta-learner model is **0.81**.

- We further fine-tune meta-learner for a new domain:

	CC	KSO	CS
10MB	0.832	0.841	0.856
30MB	0.847	0.859	0.876

Table 1: F1-score of positive predictions of the adapted meta-learner on 3 new domains: Computer Components (CC), Kitchen Storage and Organization (KSO) and Cats Supply (CS).

-

End Task Model

- We use a simple embedding layer + Bi-LSTM model + dense + softmax to evaluate the performance of different embeddings.

- The input size of Bi-LSTM is the same as the embedding and the output size is 128.

- We apply dropout rate of 0.5 on all layers except the last layer and use Adam as the optimizer.

Baselines of Pre-trained Embedding

- No Embedding (NE): random initialization of embedding layer.
- fastText (Wiki.en), GoogleNews, GloVe.Twitter.27B, GloVe.6B, GloVe.840B
- New Domain 10M (ND 10M), New Domain 30M (ND 30M)
- 200 Domains + New Domain 30M (200D + ND 30M)
- L-DENP 200D: replace the meta-learner with a cosine function.

Results

	CC(13)	KSO(17)	CS(11)
NE	0.596	0.653	0.696
fastText	0.705	0.717	0.809
GoogleNews	0.76	0.722	0.814
GloVe.Twitter.27B	0.696	0.707	0.80
GloVe.6B	0.701	0.725	0.823
GloVe.840B	0.803	0.758	0.855
ND 10M	0.77	0.749	0.85
ND 30M	0.794	0.766	0.87
200D + ND 30M	0.795	0.765	0.859
L-DENP 200D + ND 30M	0.806	0.762	0.870
L-DEM 200D + ND 10M	0.791	0.761	0.872
L-DEM 50D + ND 30M	0.795	0.768	0.868
L-DEM 100D + ND 30M	0.803	0.773	0.874
L-DEM 200D + ND 30M	**0.809**	**0.775**	**0.883**

Table 2: Accuracy of different embeddings on classification tasks for 3 new domains (numbers in parenthesis: the number of classes)

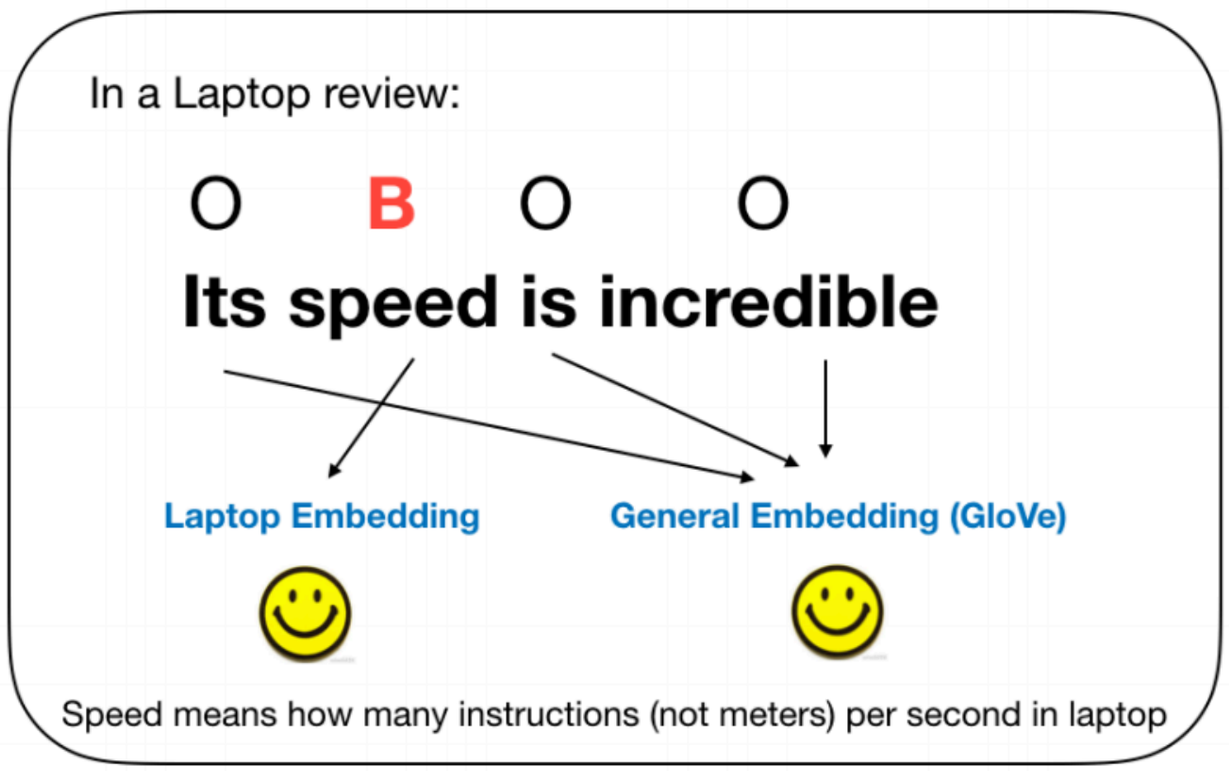
Conclusions

- We formulate a problem of learning domain word embedding from small corpus.
- Given many previous domains and a small new domain corpus, the proposed method can leverage relevant contexts from in the past domain to augment the current domain, via a meta-learner.
- Experimental results show that our method is promising.

Application to Aspect Extraction

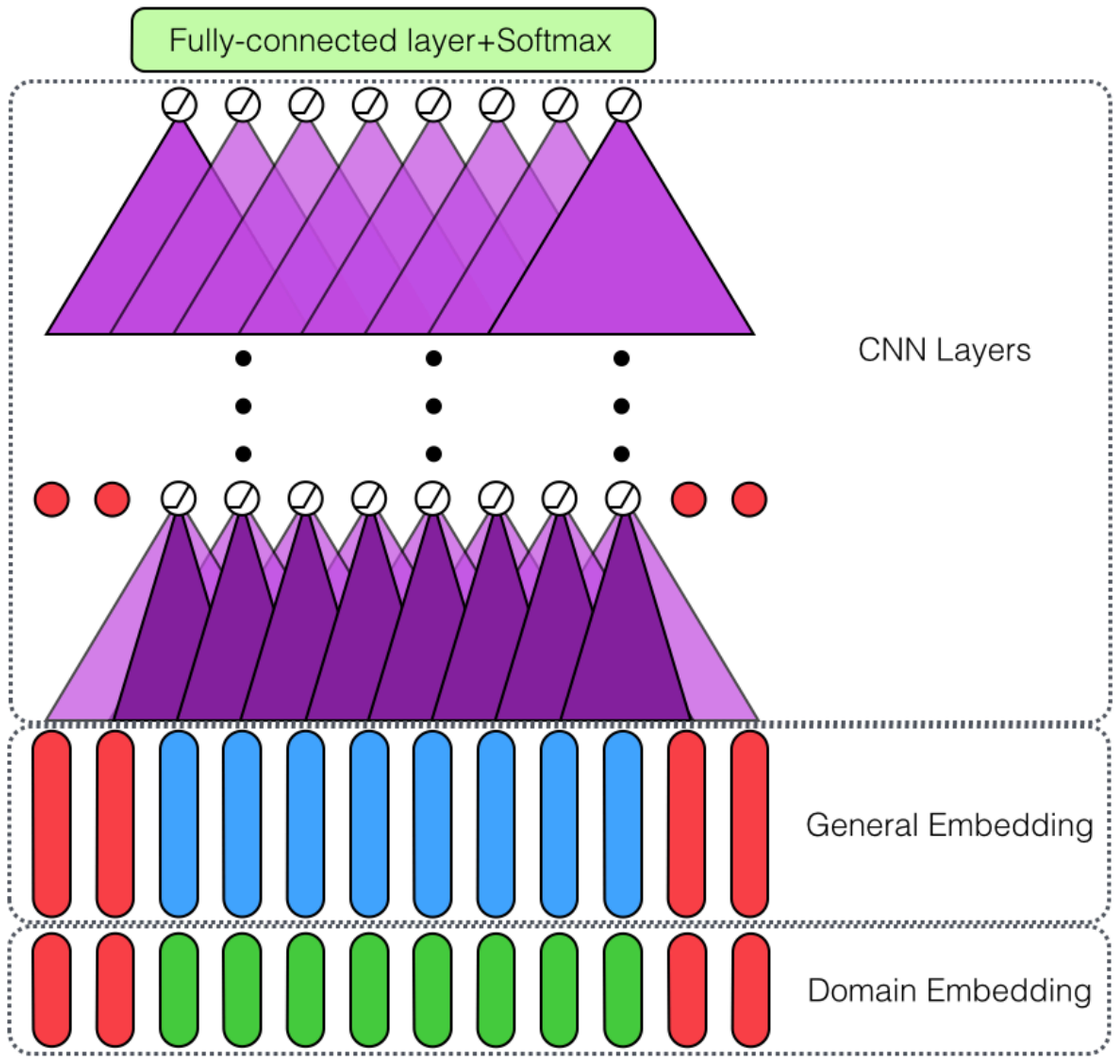
- Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction (ACL 2018)

  - Both general-purpose word embeddings and domain-specific word embeddings have their own merits for different types of words.

- 

Extension

- Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction (ACL 2018)



Model	Laptop	Restaurant
CRF	74.01	69.56
IHS_RD	74.55	-
NLANGP	-	72.34
WDEmb	75.16	-
LSTM	75.25	71.26
BiLSTM-CNN-CRF	77.8	72.5
RNCRF	78.42	-
CMLA	77.80	-
MIN	77.58	73.44
GloVe-CNN	77.67	72.08
Domain-CNN	78.12	71.75
MaxPool-DE-CNN	77.45	71.12
DE-LSTM	78.73	72.94
DE-OOD-CNN	80.21	74.2
DE-Google-CNN	78.8	72.1
DE-CNN-CRF	80.8	74.1
DE-CNN	**81.59**	**74.37**

Table 2: Comparison results in F_1 score: numbers in the third group are averaged scores of 5 runs. *

Roadmap

- Motivation
- Lifelong Supervised Learning
  - Open-world Classification (WWW 2019)
- Lifelong Self-supervised Learning
  - Domain Word Embedding (IJCAI 2018, ACL 2018)
  - **Contextualized Domain Representation Learning**

Future Work

- Contextualized Domain Representation Learning
  - Pre-trained language models, such as ELMo and BERT, significantly boost the performance of many NLP tasks.
- However, they are not perfect on
  - new domains.
  - end tasks with less training data.

Future Work

- Contextualized Domain Representation Learning
  - keep learning better representations from general-purpose language models to domain-specific language models for better fine-tuning of end tasks.
  - keep more relevant tasks before fine-tuning on end tasks with less training data.

Reference

Hu Xu, Bing Liu, Lei Shu, P. Yu, Open-world Learning and Application to Product Classification, The Web Conference (WWW 2019)

Hu Xu, Bing Liu, Lei Shu, Philip S. Yu, BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis, NAACL 2019

Lei Shu, **Hu Xu**, Bing Liu and Piero Molino, Modeling Multi-Action Policy for Task-Oriented Dialogues, EMNLP 2019

Lei Shu, Piero Molino, Mahdi Namazifar, Bing Liu, **Hu Xu**, Huaixiu Zheng and Gokhan Tur, Flexibly-Structured Model for Task-Oriented Dialogues, SIGDIAL 2019,

Hu Xu, Bing Liu, Lei Shu, Philip S. Yu, Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction, ACL 2018

Hu Xu, Bing Liu, Lei Shu, Philip S. Yu, Lifelong Domain Word Embedding via Meta-Learning, IJCAI 2018

Reference Cont.

Hu Xu, Sihong Xie, Lei Shu, Philip S. Yu, Dual Attention Network for Product Compatibility and Function Satisfiability Analysis, AAAI 2018

Hu Xu, Sihong Xie, Lei Shu, Philip S. Yu, Product Function Need Recognition via Semi-supervised Attention Network, IEEE Bigdata 2017

Lei Shu, **Hu Xu**, Bing Liu, DOC: Deep Open Classification of Text Documents, EMNLP 2017

Lei Shu, **Hu Xu**, Bing Liu, Lifelong Learning CRF for Supervised Aspect Extraction, ACL 2017

Hu Xu, Sihong Xie, Lei Shu, Philip S. Yu, CER: Complementary Entity Recognition via Knowledge Expansion on Large Unlabeled Product Reviews, IEEE Bigdata 2016

Lei Shu, Bing Liu, **Hu Xu**, and Annice Kim, Lifelong-RL: Lifelong Relaxation Labeling for Separating Entities and Aspects in Opinion Targets, EMLNP 2016

QAs

- Thanks for coming.

Additional Slides

Roadmap

- Motivation
- Lifelong Supervised Learning
  - Open-world Classification (WWW 2019)
  - **Graph Representation Learning**
- Lifelong Self-supervised Learning
  - Domain Word Embedding (IJCAI 2018, ACL 2018)
  - Contextualized Domain Representation Learning

Future Work

- Lifelong Graph Representation Learning
- learning the changes of a knowledge graph (KB) and the updated representation for reasoning, policy learning and future use.

Retrieve Relevant Past Knowledge

Algorithm 1: Identifying Context Words from the Past

Input : a knowledge base $\mathcal{K}$ containing a vocabulary $\mathcal{K}.V_{wf}$, a base meta-learner $\mathcal{K}.M$, and domain knowledge $\mathcal{K}_{m+1:n}$;
a new domain corpus D_{n+1}.

Output: relevant past knowledge $\mathcal{A}$, where each element is a key-value pair $(w_t, \mathcal{C}_{w_t})$ and $\mathcal{C}_{w_t}$ is a list of context words from all similar domain contexts for w_t.

$(V_{m+1:n}, C_{m+1:n}, E_{m+1:n}) \leftarrow \mathcal{K}_{m+1:n}$
$V_{n+1} \leftarrow \text{BuildVocab}(D_{n+1})$
$C_{n+1} \leftarrow \text{ScanContextWord}(D_{n+1}, V_{n+1})$
$E_{n+1} \leftarrow \text{BuildFeatureVector}(D_{n+1}, \mathcal{K}.V_{wf})$
$M_{n+1} \leftarrow \text{AdaptMeta-learner}(\mathcal{K}.M, E_{m+1:n}, E_{n+1})$
$\mathcal{A} \leftarrow \emptyset$
for $(V_j, C_j, E_j) \in (V_{m+1:n}, C_{m+1:n}, E_{m+1:n})$ **do**
  $O \leftarrow V_j \cap V_{n+1}$
  $F \leftarrow \{(\mathbf{x}_{o,j,1}, \mathbf{x}_{o,n+1,1}) |$
    $o \in O$ and $\mathbf{x}_{o,j,1} \in E_j$ and $\mathbf{x}_{o,n+1,1} \in E_{n+1}\}$
  $S \leftarrow M_{n+1}.\text{inference}(F)$
  $O \leftarrow \{o | o \in O \text{ and } S[o] \geq \delta\}$
  for $o \in O$ **do**
    $\mathcal{A}[o].\text{append}(C_j[o])$
  end
end
$\mathcal{K}_{n+1} \leftarrow (V_{n+1}, C_{n+1}, E_{n+1})$
return $\mathcal{A}$

Augmented Embedding Training

$$\mathcal{L}_{D_{n+1}} = \sum_{t=1}^{T} \Big(\sum_{w_c \in \mathcal{W}_{w_t}} \big(\log \sigma(\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_c}) + \sum_{w_{c'} \in \mathcal{N}_{w_t}} \log \sigma(-\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_{c'}}) \big) \Big), \tag{2}$$

$$\mathcal{L}_{\mathcal{A}} = \sum_{(w_t, \mathcal{C}_{w_t}) \in \mathcal{A}} \Big(\sum_{w_c \in \mathcal{C}_{w_t}} \big(\log \sigma(\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_c}) + \sum_{w_{c'} \in \mathcal{N}_{w_t}} \log \sigma(-\boldsymbol{u}_{w_t}^T \cdot \boldsymbol{v}_{w_{c'}}) \big) \Big). \tag{3}$$

Fine-tune Meta-Learner

- Sample 3000 words from each domain of end tasks and select 3500 paired examples for training, 500 examples for validation and 2000 examples for testing.

Evaluation of End Task

- We randomly draw 1500 reviews from each class to make up the experiment data, from which we keep **10000** reviews for testing and split the rest 7:1 for training and validation, respectively.

- We train and evaluate each task on each system 10 times (with different initializations) and average the results.