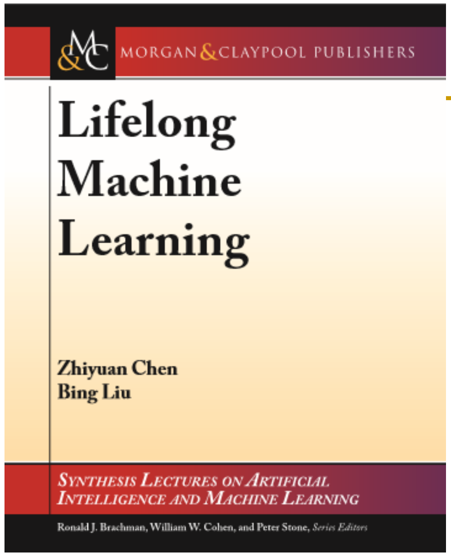
EMNLP-2016 Tutorial, Nov 1, 2016, Austin, Texas, USA

Lifelong Machine Learning for Natural Language Processing


MORGAN & CLAYPOOL PUBLISHERS
Lifelong Machine Learning
Zhiyuan Chen
Bing Liu
Synthesis Lectures on Artificial Intelligence and Machine Learning
Ronald J. Brachman, William W. Cohen, and Peter Stone, Series Editors

Zhiyuan (Brett) Chen, Google

Bing Liu, University of Illinois at Chicago

Introduction

(Chen and Liu, 2016-book)

- Classic Machine Learning (ML) paradigm: isolated single-task learning
  - Given a dataset, run an ML algo. to build a model
  - Without considering the past learned knowledge
- Existing ML algorithms such as
  - SVM, NB, DT, Deep NN, CRF, and topic models
  - Have been very successful in practice
- Let's call this: Machine Learning (ML) 1.0

Introduction: ML 1.0

- Weaknesses of “isolated learning”
  - Knowledge learned is not retained or accumulated
    - Needs a large number of training examples
    - Suitable for well-defined & narrow tasks in restricted env.
- Human beings never learn in isolation
  - We retain knowledge & use it to learn more knowlg.
  - Learn effectively from a few or no examples
    - Our knowledge learned and accumulated in the past
      - which allows us to learn with little data or effort

Introduction: An Example

- Nobody has ever given me 1000 positive and 1000 negative online reviews and ask me
  - to build a classifier to classify Camera reviews
    - In fact, I don’t need any training data
- I have accumulated so much knowledge
    - about how people praise and criticize things
- If I don’t have the accumulated knowledge, NO
  - E.g., I don’t know Arabic and if someone gives me 2000 training reviews in Arabic, I cannot do it.

Introduction: ML 2.0

Thrun, 1996b; Silver et al 2013; Chen and Liu, 2014a, 2016-book

- Statistical ML is getting increasingly mature
- It's time for *Lifelong Machine Learning* (LML)
  - Retain/accumulate learned knowledge in the past & use it to help future learning
    - become more knowledgeable & better at learning
  - Learn by mimicking "human learning"

- Let us call this paradigm Machine Learning 2.0
  - Without LML, it is unlikely we can build a truly intelligent system.

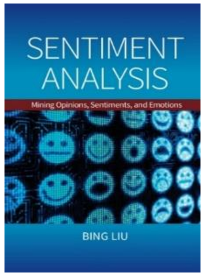
Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

A Motivating Example
(Liu, 2012, 2015)



- My interest in LML stemmed from extensive experiences on *sentiment analysis* in a startup company many years ago.
- Sentiment analysis (SA)
  - Sentiment and target aspect: "*The screen is great, but the voice quality is poor.*"
    - Positive about screen but negative about voice quality
  - Extensive knowledge sharing across tasks/domains
    - Sentiment expressions & aspects

Knowledge Shared Across Domains

- After working on many SA projects for clients, I realized
  - a lot of concept sharing across domains
  - as we see more and more domains, fewer and fewer things are new.
- Easy to see sharing of sentiment words,
  - e.g., *good, bad, poor, terrible*, etc.
- There is also a great deal of aspect sharing
  - product feature sharing

Sharing of Product Features

- Observation: A great deal of product features (or aspects) overlapping across domains
  - Every product review domain has the aspect *price*
  - Most electronic products share the aspect *battery*
  - Many also share the aspect of *screen*.
  - Many also share sound quality
  - ….
- It is rather “silly” not to exploit such sharing in learning or extraction.

What does that Mean for Learning?

- How to systematically exploit such sharing?
  - Retain/accumulate knowledge learned in the past.
  - Leverage the knowledge for new task learning
- I.e., *lifelong machine learning* (LML)
- This leads to our own work
  - Lifelong topic modeling (Chen and Liu 2014a, b)
  - Lifelong sentiment classification (Chen et al 2015)
  - Several others

LML is Suitable for NLP

- Knowledge, easily shared across domains
  - Words and phrases almost have the same meaning in different domains or tasks.
  - Sentences in all domains follow the same syntax
- Knowledge, useful in different types of tasks.
  - NLP problems are closely related to each other
    - POS tagging, coreference resolution, entity recognition, …
- Big data provides a great opportunity for LML
  - Learn a large amount of knowledge to become
    - More and more knowledgeable & better at learning

LML is Useful in General

- LML is suitable for all learning
- It is hard to imagine:
  - We have to learn everything from scratch whenever we encounter a new problem or environment.
- If that were the case,
  - Intelligence is unlikely

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Definition of LML

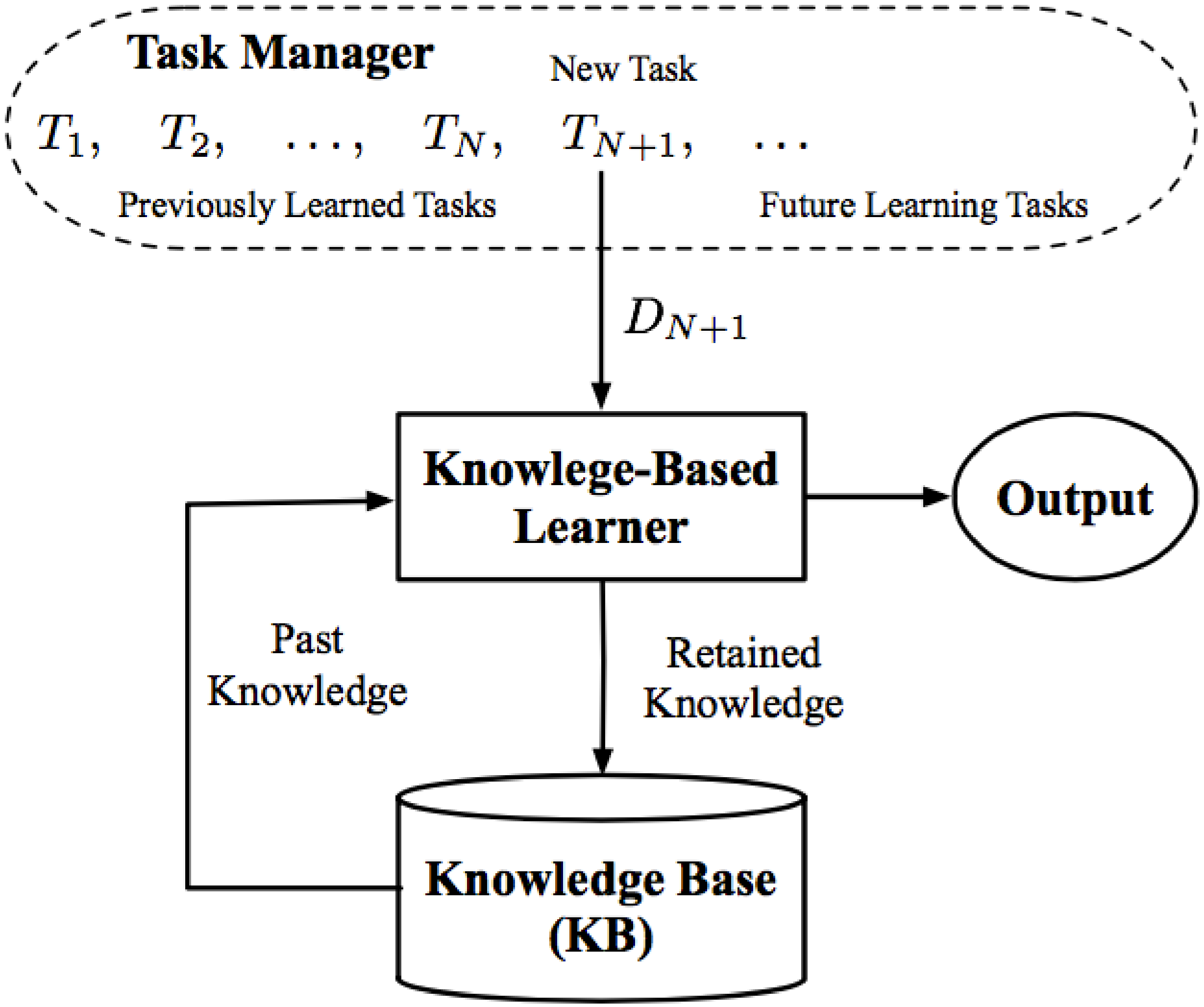
(Thrun 1995, Chen and Liu, 2016 – new book)

- The learner has performed learning on a sequence of tasks, from 1 to *N*.
- When faced with the (*N*+1)th task, it uses the relevant knowledge in its *knowledge base* (KB) to help learning for the (*N*+1)th task.
- After learning (*N*+1)th task, KB is updated with learned results from (*N*+1)th task.

Key Characteristics of LML
(Chen and Liu, 2016 – new book)

- Continuous learning process
- Knowledge accumulation in KB
- Use of past knowledge to help future learning

Lifelong Machine Learning System



Components of LML

- Knowledge Base (KB)
  - Past Information Store (PIS)
    - Data, intermediate and final results
  - Meta-Knowledge Miner (MKM)
    - Meta-mining of PIS and MKS
  - Meta-Knowledge Store (MKS)
    - mined knowledge
  - Knowledge Reasoner (KR)
    - Make inference to generate more knowledge
- Most current systems don't have all these

Components of LML (Contd)

- Knowledge-Based Learner (KBL)
  - Leverage past knowledge in KB in new learning
    - Task Knowledge Miner (TKM): identify/mine knowledge suitable for the task
  - Learner
- Task Manager
  - Receives and manages arriving tasks
- Output
  - Model for the current task

Two Types of Knowledge

- *Global knowledge*: Many existing LML methods assume that there is a *global latent structure* among tasks that are shared by all (Bou Ammar et al., 2014, Ruvolo and Eaton, 2013b, Tanaka and Yamamura, 1997, Thrun, 1996b, Wilson et al., 2007)
  - This global structure can be learned and leveraged in the new task learning.
  - These methods grew out of multi-task learning.

Two Types of Knowledge (Contd)

- *Local knowledge:* Many other methods do not assume such a *global latent structure* among tasks (Chen and Liu, 2014a,b, Chen et al., 2015, Fei et al., 2016, Liu et al., 2016, Shu et al., 2016)
- During the learning of a new task,
  - they select the pieces of prior knowledge to use based on the need of the new task.
- Called *local knowledge* because they are not assumed to form a coherent global structure.

Two Kinds of Tasks

- Independent tasks: each task is independent of other tasks
  - Each task can be learned independently, although using knowledge gained in other tasks may help this task learning
  - Much of the current research assume this.
- Dependent tasks: each task has some dependency on some other tasks, e.g.,
  - Cumulative learning (Fei et al 2016)

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Transfer learning

- Source domain(s): With labeled training data
- Target domain: With little/no labeled training data
- Goal: leverage the information from the source domain(s) to help learning in the target domain
    - Only optimize the target domain/task learning

A Large Body of Literature

- Transfer learning has been a popular research topic and researched in many fields, e.g.,
  - Machine learning
  - Data mining
  - Natural language processing
  - Computer vision
- (Taylor and Stone, 2009, Pan & Yang, 2010). presented excellent surveys with extensive references.

One Transfer Learning Technique

- Structural correspondence learning (SCL) (Blitzer et al., 2006)
- Pivot features
  - Have the same characteristics or behaviors in both domains
  - Non-pivot features which are correlated with many of the same pivot features are assumed to correspond

Choosing Pivot Features

- For different applications, pivot features may be chosen differently, for example,
  - For part-of-speech tagging, frequently-occurring words in both domains are good choices (Blitzer et al., 2006)
  - For sentiment classification, pivot features are words that frequently-occur in both domains and also have high mutual information with the source label (Blitzer et al., 2007).

Finding Feature Correspondence

- Compute the correlations of each pivot feature with non-pivot features in both domains by building binary pivot predictors

$$f_\ell(\mathbf{x}) = \mathrm{sgn}(\hat{\mathbf{w}}_\ell \cdot \mathbf{x}), \quad \ell = 1 \ldots m$$

  - Using unlabeled data (predicting whether the pivot feature *l* occurs in the instance)
  - The weight vector $\hat{\mathbf{w}}_\ell$ encodes the covariance of the non-pivot features with the pivot feature

Finding Feature Correspondence

- Positive values in $\hat{\mathbf{w}}_\ell$:
  - Indicate that those non-pivot features are positively correlated with the pivot feature *l* in the source or the target
- Produce a correlation matrix W

$$W = [\hat{\mathbf{w}}_1 | \ldots | \hat{\mathbf{w}}_m]$$

Computing Low Dim. Approximation

- SVD is employed to compute a low-dimensional linear approximation θ

$$W = UDV^T \quad \theta = U^T_{[1:h,:]}$$

- θ : mapping from original space to new space

- The final set of features used for training and for testing: original features **x** + θ**x**

Multi-Task Learning

- Problem statement: Co-learn multiple related tasks simultaneously:
  - All tasks have labeled data and are treated equally
  - Goal: optimize learning/performance across all tasks through shared knowledge
- Rationale: introduce inductive bias in the joint hypothesis space of all tasks (Caruana, 1997)
  - By exploiting the task relatedness structure, or shared knowledge

One Multi-Task Model: GO-MTL
(Kumar et al., ICML 2012)

- GO-MTL: Grouping and Overlap in Multi-Task Learning
- Does not assume that all tasks are related
- Applicable to classification and regression

GO-MTL Assumptions

- All task models share latent basic model components
- Each task model is a linear combination of shared latent components
- The linear weight is sparse, to use a small number of latent components

Notations

- N tasks in total
- k (< N) latent basis model components
- Each basis task is represented by $\boldsymbol{l}$ (a vector of size d)
- For all latent tasks, $\boldsymbol{L} = (\boldsymbol{l}_{1}, \boldsymbol{l}_{2}, \ldots, \boldsymbol{l}_{k})$
- $\boldsymbol{L}$ is learned from N individual tasks.
  - E.g., weights/parameters of logistic regression or linear regression

The Approach

- $\mathbf{s}^t$ is a linear weight vector and is assumed to be sparse.

$$\boldsymbol{\theta}^t = \mathbf{L}\mathbf{s}^t$$

- Stacking $\mathbf{s}^t$ ($\boldsymbol{\theta}^t$) for all tasks, we get **S** ($\boldsymbol{\Theta}$). **S** captures the task grouping structure.

$$\underset{d \times N}{\boldsymbol{\Theta}} = \underset{d \times k}{\mathbf{L}} \times \underset{k \times N}{\mathbf{S}}$$

Objective Function in GO-MTL

$$\sum_{t=1}^{N}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\mathbf{x}_i^t;\mathbf{L}\mathbf{s}^t),y_i^t\right)+\mu\left\|\mathbf{S}\right\|_1+\lambda\left\|\mathbf{L}\right\|_F^2$$

Optimization Strategy

- Alternating optimization strategy to reach a local minimum.
- For a fixed ***L***, optimize *s*$_t$:

$$\mathbf{s}^t = \underset{\mathbf{s}^t}{\operatorname{argmin}} \sum_{i=1}^{n_t} \mathcal{L}\left(f(\mathbf{x}_i^t; \mathbf{L}\mathbf{s}^t), y_i^t\right) + \mu \left\|\mathbf{s}^t\right\|_1$$

- For a fixed ***S***, optimize ***L***:

$$\underset{\mathbf{L}}{\operatorname{argmin}} \sum_{t=1}^{N} \sum_{i=1}^{n_t} \mathcal{L}\left(f(\mathbf{x}_i^t; \mathbf{L}\mathbf{s}^t), y_i^t\right) + \lambda \left\|\mathbf{L}\right\|_F^2$$

A Large Body of Literature

- Two tutorials on MTL
  - Multi-Task Learning: Theory, Algorithms, and Applications. SDM-2012, by Jiayu Zhou, Jianhui Chen, Jieping Ye
  - Multi-Task Learning Primer. IJCNN'15, by Cong Li and Georgios C. Anagnostopoulos

Transfer, Multitask → Lifelong

- Transfer learning vs. LML
  - Transfer learning is not continuous
  - No retention or accumulation of knowledge
  - Only one directional: help target domain

Transfer, Multitask → Lifelong

- Transfer learning vs. LML
  - Transfer learning is not continuous
  - No retention or accumulation of knowledge
  - Only one directional: help target domain
- Multitask learning vs. LML
  - Multitask learning retains no knowledge except data
  - Hard to re-learn all when tasks are numerous

- Online (incremental) multi-task learning is LML

Online Learning

- The training data points come in a sequential order (online setting)
  - Computationally infeasible to train over the entire dataset
- Different from LML
  - Still performs the same learning task over time
  - LML aims to learn from a sequence of different tasks, retain and accumulate knowledge

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Lifelong Supervised Learning (LSL)

- The learner has performed learning on a sequence of supervised learning tasks, from 1 to *N*.

- When faced with the (*N*+1)th task, it uses the relevant knowledge and labeled training data of the (*N*+1)th task to help learning for the (*N*+1)th task.

Early Work on Lifelong Learning
(Thrun, 1996b)

- Concept learning tasks: The functions are learned over the lifetime of the learner, f_1, f_2, f_3, … $\in F$.
- Each task: learn the function $f: I \rightarrow \{0, 1\}$. $f(x)=1$ means x is a particular concept.
  - For example, $f_{dog}(x)=1$ means x is a dog.
- For *n*th task, we have its training data X
  - Also the training data X_k of k =1 , 2, …, n-1 tasks.

Intuition

- The paper proposed a few approaches based on two learning algorithms,
  - Memory-based, e.g., kNN or shepard's method
  - Neural networks
- Intuition: when we learn $f_{dog}(x)$, we can use functions or knowledge learned from previous tasks, such as $f_{cat}(x)$, $f_{bird}(x)$, $f_{tree}(x)$, etc.
  - Data for $f_{cat}(X)$, $f_{bird}(X)$, $f_{tree}(X)$… are support sets.

Memory based Lifelong Learning

- First method: use the support sets to learn a new representation, or function

  g: *I* → *I'*

  - which maps input vectors to a new space. The new space is the input space for the final *k*NN
  - Adjust *g* to minimize the energy function

$$E := \sum_{k=1}^{n-1} \sum_{\langle x, y=1 \rangle \in X_k} \left(\sum_{\langle x', y'=1 \rangle \in X_k} ||g(x) - g(x')|| - \sum_{\langle x', y'=0 \rangle \in X_k} ||g(x) - g(x')|| \right)$$

  - g is a neural network, trained with Back-Prop. kNN is then applied for the *n*th (new) task

Second Method

- It learns a distance function using support sets

  d: $I \times I \rightarrow [0, 1]$

  - It takes two input vectors x and x' from a pair of examples <x, y>, <x', y'> of the same support set X_k (k = 1, 2, , …, n-1)
  - d is trained with neural network using back-prop, and used as a general distance function
  - Training examples are:

$$\langle (x, x'), 1 \rangle \quad \text{if } y{=}y'{=}1$$
$$\langle (x, x'), 0 \rangle \quad \text{if } (y{=}1 \wedge y'{=}0) \text{ or } (y{=}0 \wedge y'{=}1)$$

Making Decision

- Given the new task training set X_n and a test vector x, for each +ve example, (x', y'=1)$\in X_n$,
  - d(x, x') is the probability that x is a member of the target concept.
- Decision is made by using votes from positive examples, <x_1, 1>, <x_2, 1>, … $\in X_n$ combined with Bayes' rule

$$P(f_n(x)=1) = 1 - \left(1 + \prod_{\langle x', y'=1\rangle \in X_n} \frac{d(x,x')}{1-d(x,x')}\right)^{-1}$$

LML Components in this Case

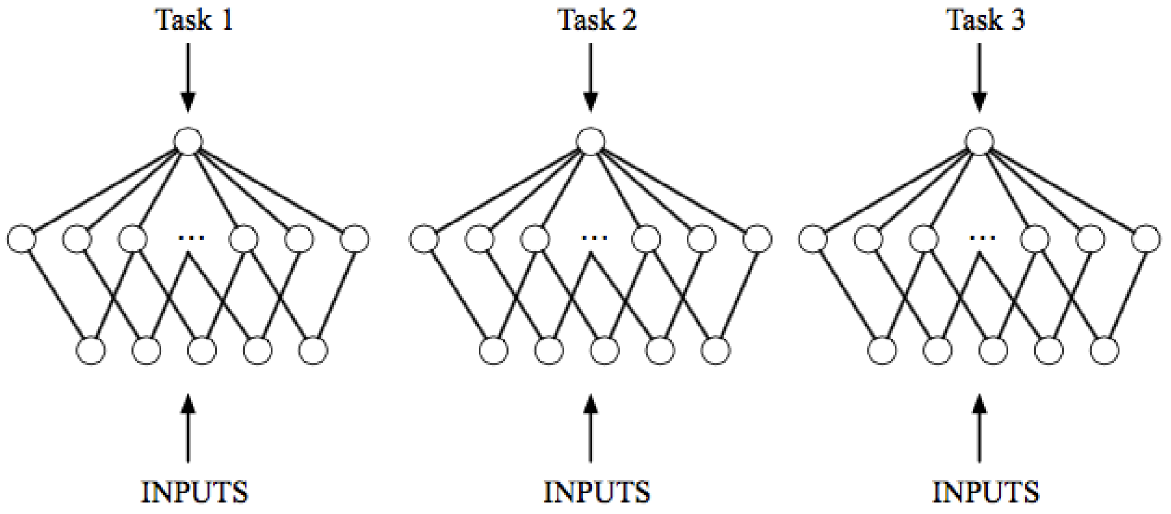
- **KB**
  - Store all the support sets.
  - Distance function *d*(x, x’): the probability of example x and x’ being the same concept.
- **KBL**
  - Voting with Bayes’ rule.

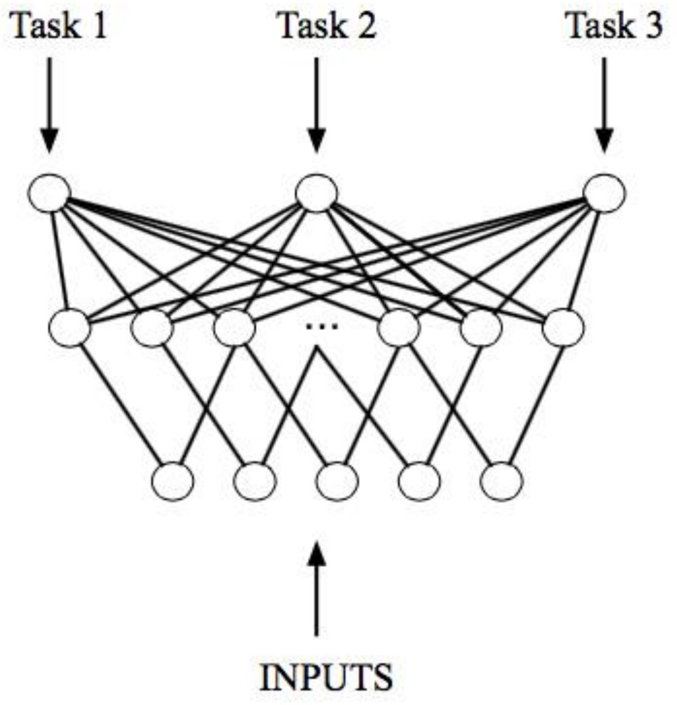
Neural Network approaches

- Approach 1: based on that in (Caruana, 1993, 1997), which is actually a batch multitask learning approach.
  - Simultaneously minimize the error on both the support sets $\{X_k\}$ and the training set X_n

- Approach 2: an *explanation-based neural network (EBNN)*

Neural Network approaches





Task Clustering (TC)
(Thrun and O'Sullivan, 1996)

- In general, not all previous *N*-1 tasks are similar to the *N*th (new) task
- Based on a similar idea to the lifelong memory-based methods in (Thrun, 1996b)
  - It clusters previous tasks into groups or clusters
- When the (new) *N*th task arrives, it first
  - selects the most similar cluster and then
  - uses the distance function of the cluster for classification in the *N*th task

Some Other Early works on LML

- Constructive inductive learning to deal with learning problem when the original representation space is inadequate for the problem at hand (Michalski, 1993)
- Incremental learning primed on a small, incomplete set of primitive concepts (Solomonoff, 1989)
- Explanation-based neural networks MTL (Thrun, 1996a)
- MTL method of functional (parallel) transfer (Silver & Mercer, 1996)
- Lifelong reinforcement learning (Tanaka & Yamamura, 1997)
- Collaborative interface agents (Metral & Maes, 1998)

ELLA

(Ruvolo & Eaton, 2013a)

- ELLA: Efficient Lifelong Learning Algorithm
- It is based on GO-MTL (Kumar et al., 2012)
  - A batch multitask learning method
- ELLA is online multitask learning method
  - ELLA is more efficient and can handle a large number of tasks
  - Becomes a lifelong learning method
    - The model for a new task can be added efficiently.
    - The model for each past task can be updated rapidly.

Inefficiency of GO-MTL

- Since GO-MTL is a batch multitask learning method, the optimization goes through all tasks and their training instances (Kumar et al., 2012).

$$\sum_{t=1}^{T}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^{(t)};\boldsymbol{L}\boldsymbol{s}^{(t)}), y_i^{(t)}\right)+\mu\|\boldsymbol{S}\|_1+\lambda\|\boldsymbol{L}\|_F^2$$

- Very inefficient and impractical for a large number of tasks.
  - It cannot incrementally add a new task efficiently

Initial Objective Function of ELLA

- Objective Function (Average rather than sum)

$$e_T(\mathbf{L}) = \frac{1}{T}\sum_{t=1}^{T}\min_{\mathbf{s}^{(t)}}\left\{\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f\left(\mathbf{x}_i^{(t)};\mathbf{L}\mathbf{s}^{(t)}\right),y_i^{(t)}\right) + \mu\|\mathbf{s}^{(t)}\|_1\right\} + \lambda\|\mathbf{L}\|_{\mathsf{F}}^2\ , \qquad (1)$$

Approximate Equation (1)

- Eliminate the dependence on all of the past training data through inner summation
  - By using the second-order Taylor expansion of around $\theta = \theta^{(t)}$ where
  - $\theta^{(t)}$ is an optimal predictor learned on only the training data on task *t*.

Removing inner summation

$$\frac{1}{N}\sum_{t=1}^{N}\min_{\mathbf{s}^t}\left\{\|\hat{\boldsymbol{\theta}}^t-\mathbf{L}\mathbf{s}^t\|^2_{\boldsymbol{H}^t}+\mu\|\mathbf{s}^t\|_1\right\}+\lambda\|\mathbf{L}\|_F^2$$

$$\boldsymbol{H}^t=\frac{1}{2}\nabla^2_{\boldsymbol{\theta}^t,\boldsymbol{\theta}^t}\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^t;\boldsymbol{\theta}^t),y_i^t\right)\Big|_{\boldsymbol{\theta}^t=\hat{\boldsymbol{\theta}}^t}$$

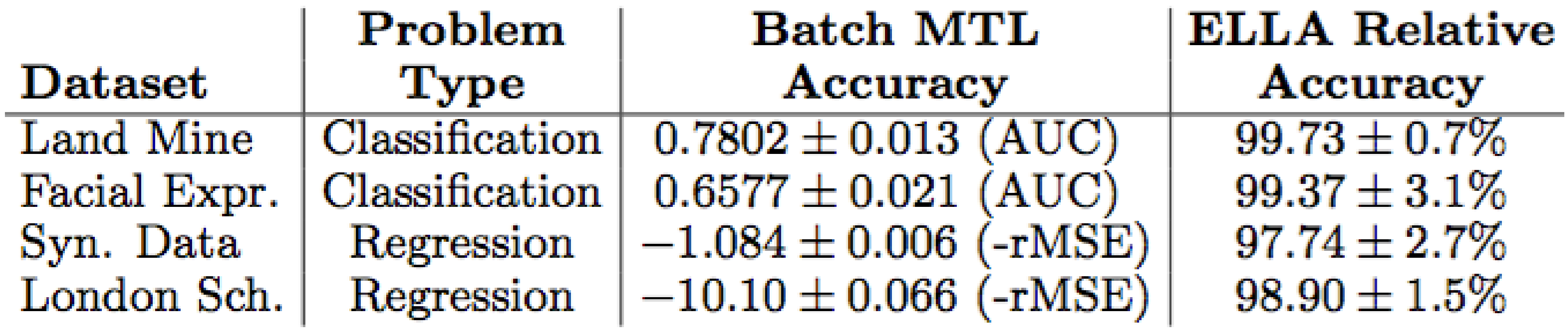
$$\hat{\boldsymbol{\theta}}^t=\underset{\boldsymbol{\theta}^t}{\operatorname{argmin}}\frac{1}{n_t}\sum_{i=1}^{n_t}\mathcal{L}\left(f(\boldsymbol{x}_i^t;\boldsymbol{\theta}^t),y_i^t\right)$$

Simplify optimization

- GO-MTL: when computing a single candidate ***L***, an optimization problem must be solved to re-compute the value of each $s^{(t)}$.
- ELLA: after $s^{(t)}$ is computed given the training data for task *t*, it will not be updated when training on other tasks. Only ***L*** will be changed.
- Note: (Ruvolo and Eaton, 2013b) added the mechanism to actively select the next task to learn.

ELLA Accuracy Result

- ELLA vs. GO-MTL

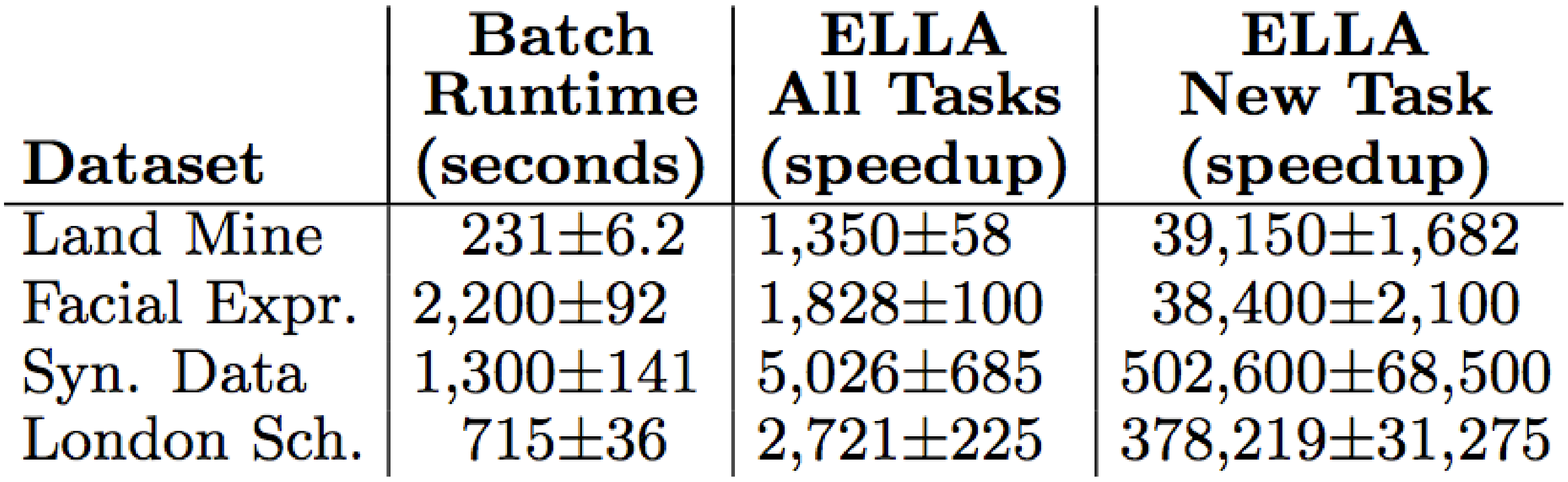


Dataset	Problem Type	Batch MTL Accuracy	ELLA Relative Accuracy
Land Mine	Classification	0.7802 ± 0.013 (AUC)	$99.73 \pm 0.7\%$
Facial Expr.	Classification	0.6577 ± 0.021 (AUC)	$99.37 \pm 3.1\%$
Syn. Data	Regression	-1.084 ± 0.006 (-rMSE)	$97.74 \pm 2.7\%$
London Sch.	Regression	-10.10 ± 0.066 (-rMSE)	$98.90 \pm 1.5\%$

Batch MTL is GO-MTL

ELLA Speed Result

- ELLA vs. GO-MTL



Dataset	Batch Runtime (seconds)	ELLA All Tasks (speedup)	ELLA New Task (speedup)
Land Mine	231±6.2	1,350±58	39,150±1,682
Facial Expr.	2,200±92	1,828±100	38,400±2,100
Syn. Data	1,300±141	5,026±685	502,600±68,500
London Sch.	715±36	2,721±225	378,219±31,275

ELLA is 1K times faster than GO-MTL on all tasks, 30K times on a new task

LML Components of ELLA

- **KB**
  - Stores all the task data
  - Matrix ***L*** for *K* basis tasks and *S*
- **KBL**
  - Each task parameter vector is a linear combination of **KS**, i.e., $\boldsymbol{\theta}^{(t)} = \boldsymbol{L}s^{(t)}$
  - Alternating optimization solving

Lifelong Sentiment Classification
(Chen, Ma, and Liu 2015)

- "*I bought a cellphone a few days ago. It is such a nice phone. The touch screen is really cool. The voice quality is great too. ....*"
- Goal: classify docs or sentences as + or -.
  - Need to manually label a lot of training data for each domain, which is highly labor-intensive
- Can we not label for every domain or at least not label so many docs/sentences?

A Simple Lifelong Learning Method

Assuming we have worked on a *large number of past domains* with all their training data *D*

- Build a classifier using *D*, test on new domain
  - Note - using only one past/source domain as in ***transfer learning*** is not good.
- In many cases – improve accuracy by as much as 19% (= 80%-61%). Why?
- In some others cases – not so good, e.g., it works poorly for toy reviews. Why? “toy”

Lifelong Sentiment Classification
(Chen, Ma and Liu, 2015)

- It adopts a Bayesian optimization framework for LML using stochastic gradient decent

- Lifelong learning uses
  - Word counts from the past data as priors.
  - Penalty terms to deal with domain dependent sentiment words and reliability of knowledge.

Naïve Bayesian Text Classification

- Key parameter

$$P(w|c_j) = \frac{\lambda + N_{c_j,w}}{\lambda|V| + \sum_{v=1}^{|V|} N_{c_j,v}}$$

- Only depends on the count of words in each class

Stored Information

- Probabilities of a word appearing in positive or negative

  $P^{\hat{t}}(w|+)$ and $P^{\hat{t}}(w|-)$

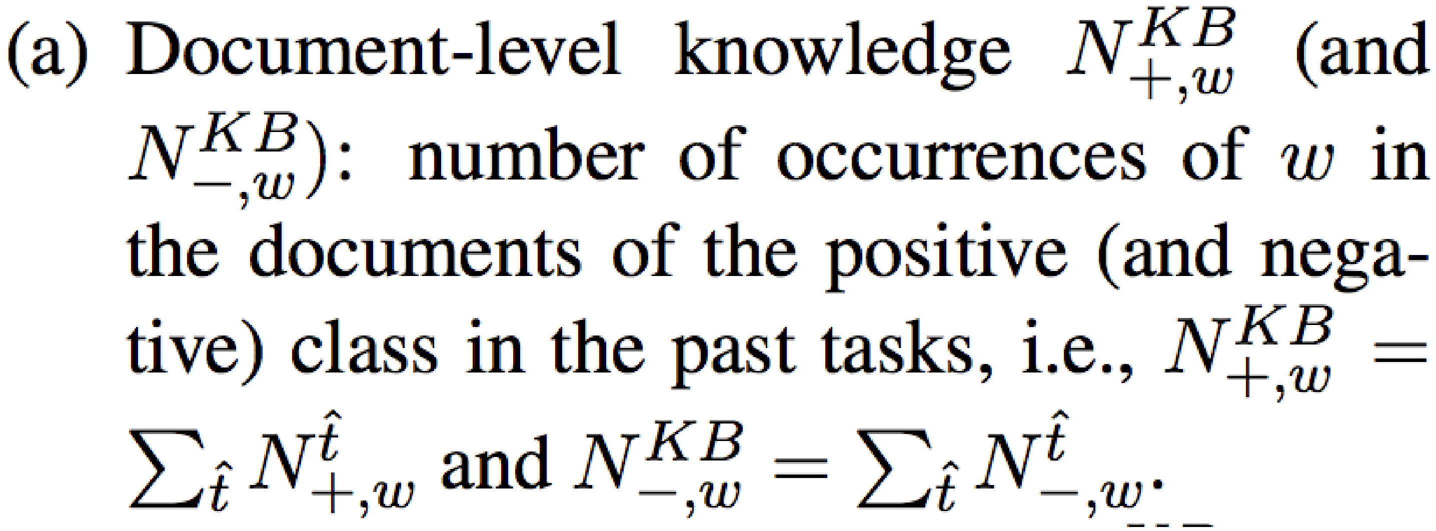
- Word counts
  - Number of times that a word appears in positive class: $N^{\hat{t}}_{+,w}$
  - Number of times that a word appears in negative class: $N^{\hat{t}}_{-,w}$

Knowledge Base

- Two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge

Knowledge Base

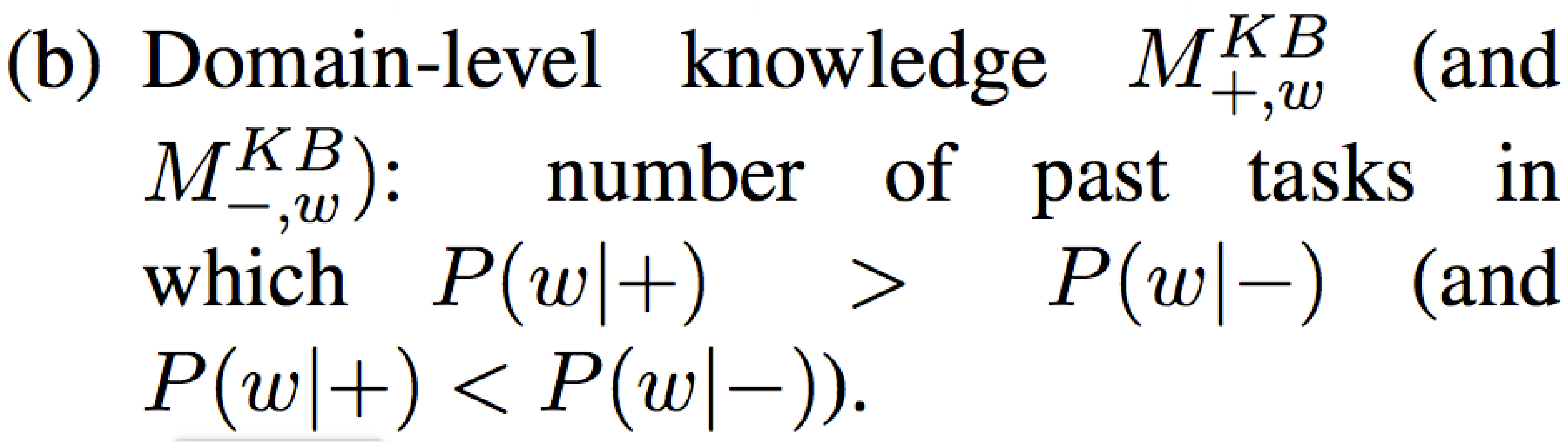
- Two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge



(a) Document-level knowledge $N^{KB}_{+,w}$ (and $N^{KB}_{-,w}$): number of occurrences of w in the documents of the positive (and negative) class in the past tasks, i.e., $N^{KB}_{+,w} = \sum_{\hat{t}} N^{\hat{t}}_{+,w}$ and $N^{KB}_{-,w} = \sum_{\hat{t}} N^{\hat{t}}_{-,w}$.

Knowledge Base

- Two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge



(b) Domain-level knowledge $M^{KB}_{+,w}$ (and $M^{KB}_{-,w}$): number of past tasks in which $P(w|+) > P(w|-)$ (and $P(w|+) < P(w|-)$).

Objective Function

- Maximize the probably difference

$$\sum_{i=1}^{|D^t|} \left(P\left(c_j|d_i\right) - P\left(c_f|d_i\right)\right)$$

- c_j: labeled class in groundtruth
- c_f: all classes other than c_j

Exploiting Knowledge via Penalties

- Penalty terms for two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge

Exploiting Knowledge via Penalties

- Penalty terms for two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge

$$\frac{1}{2}\alpha \sum_{w \in V_T} \left(\left(X_{+,w} - N^t_{+,w} \right)^2 + \left(X_{-,w} - N^t_{-,w} \right)^2 \right)$$

  - *t* is the new task

Exploiting Knowledge via Penalties

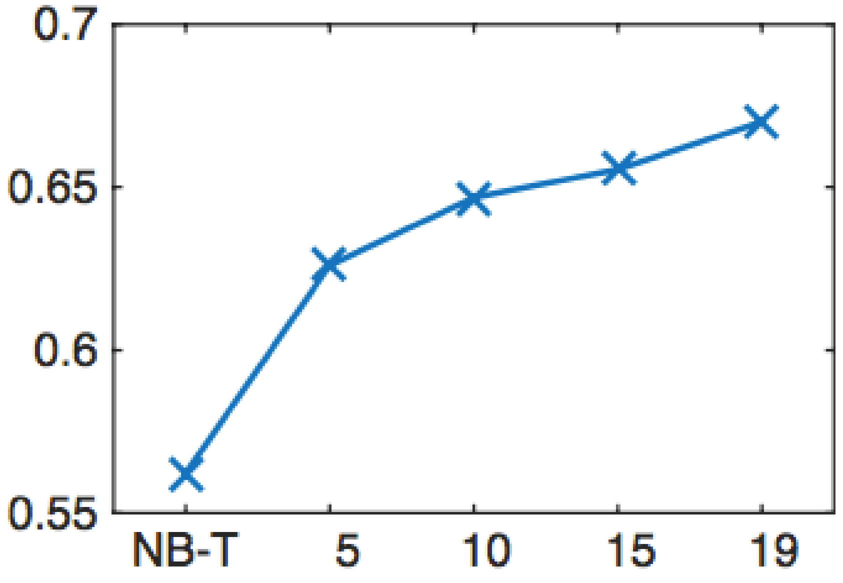
- Penalty terms for two types of knowledge
  - Document-level knowledge
  - Domain-level knowledge

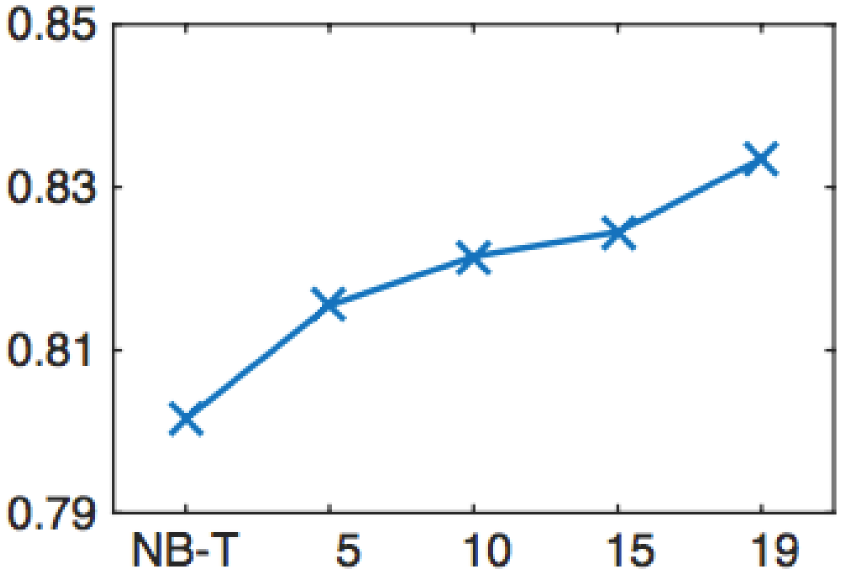
$$\frac{1}{2}\alpha \sum_{w \in V_S} \left(X_{+,w} - R_w \times X^0_{+,w}\right)^2 + \frac{1}{2}\alpha \sum_{w \in V_S} \left(X_{-,w} - (1 - R_w) \times X^0_{-,w}\right)^2$$

  - R_W: ratio of #tasks where *w* is positive / #all tasks
  - $X^0_{+,w} = N^t_{+,w} + N^{KB}_{+,w}$ and $X^0_{-,w} = N^t_{-,w} + N^{KB}_{-,w}$

One Result of LSC model

- Better F1-score (left) and accuracy (right) with more past tasks





LML Components of LSC

- **KB**
  - Word counts from previous tasks
  - Document-level knowledge
  - Domain-level knowledge
- **KBL**
  - LSC algorithm with regularization

Cumulative Learning

(Fei et al., 2016)

- Cumulative learning
  - Incrementally adding a new class without re-training the whole model from scratch
    - Learner becomes more knowledgeable
  - Detecting unseen classes in test data
    - Traditional supervised learning cannot do this
    - It needs *open classification*

- Self-learning: detect unseen/new things and learn them.

Cumulative Learning is LML

- At time point *t*, a *t*-class classifier F_t learned from past datasets $D^t = \{D_1, D_2, \ldots, D_t\}$ of classes $Y^t = \{l_1, l_2, \ldots, l_t\}$.
  - F_t classifies each test instance **x** to either one of the known classes in Y^t or the *unknown class* $\boldsymbol{l_0}$.
    - $y = F_t(\mathbf{x}), y \in \{l_1, l_2, \ldots, l_t, \boldsymbol{l_0}\}$
- At time point t+1, a class l_{t+1} (D_{t+1}) is added, F_t is ***updated*** to a (t+1)-class classifier F_{t+1}
    - $y = F_{t+1}(\mathbf{x}), y \in \{l_1, l_2, \ldots, l_t, l_{t+1}, \boldsymbol{l_0}\}$

Learning cumulatively

- How to incrementally add a class without retraining from scratch?
- "Human learning": uses the past knowledge F_t to help learn the new class l_{t+1}.
  - Find similar classes SC from known classes Y^t. E.g
    - Old classes: Y^t = {movie, cat, politics, soccer}.
    - New class: l_{t+1}= basketball
    - SC = {soccer}
  - Building F_{t+1} by focusing on separating l_{t+1} and SC.

Cumulative Learning Algorithm

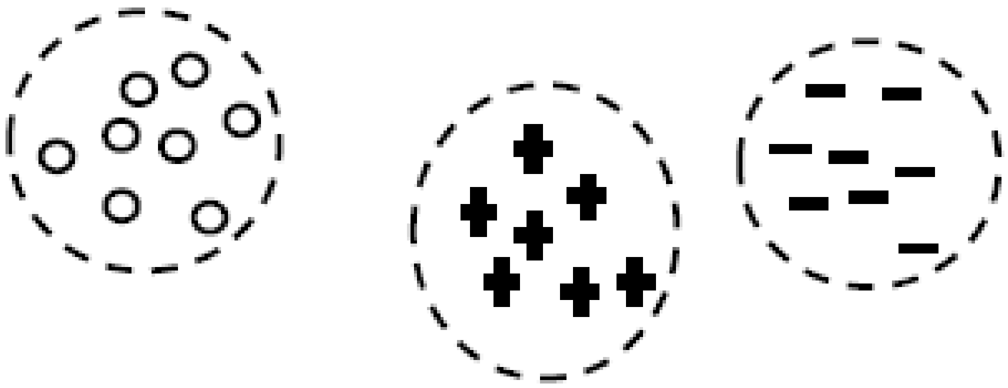
- $F_t = \{f_1, f_2, \ldots, f_t\}$, a set of binary classifiers.
- Identifying a set of similar classes ***SC*** to the new class l_{t+1} by
  - Using each f_i to classify instances in D_{t+1}.
    - SC is the set of classes that accept many from D_{t+1}
- Build f_{t+1} for l_{t+1} using classes in ***SC*** as negative data.
- Update each classifier for classes in SC by adding class l_{t+1} as an extra negative class.

Open Classification
(Fei and Liu, 2016)

- Traditional classification makes the closed world assumption:
  - Classes in testing have been seen in training
  - i.e., no new classes in the test data
- Not true in many real-life environments.
  - New data may contain unseen class documents
- We need *open (world) classification*
  - Detect the unseen class of documents

Open Classification

- *Open space risk* formulation (see Fei & Liu 2016)
  - Don't give each class too much open space
  - SVM is one half space for each class: too much
- Ideally, a "ball" to cover each class l_i
  - Each "ball" is a binary classifier f_i



Open World Learning

- Build a set of 1-vs-rest classifiers, one for each training class l_i.
- The set of 1-vs-rest classifiers $F_t = \{f_1, f_2, \ldots, f_{t+1}\}$ works together to classify
  - Each binary classifier produces a probability $P(y|\mathbf{x})$
  - $\boldsymbol{l_0}$: class of unknown

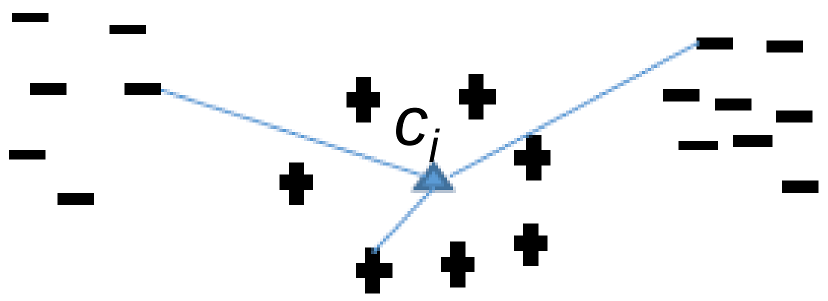
$$y^* = \begin{cases} argmax_{y \in Y^{t+1}} P(y|\mathbf{x}) & if\ P(y|\mathbf{x}) \geq \theta \\ l_0 & otherwise \end{cases}$$

CBS Learning

- To detect unseen classes, Fei and Liu (2016) proposed CBS learning:
  - Center-based similarity (CBS) space learning.
- It performs space transformation
  - Each document vector *d* is transformed to a CBS space vector
    1. Compute centers c_i for the positive class
    2. Compute similarities of each document to c_i.

    - This gives us a new data set in the CSB space.

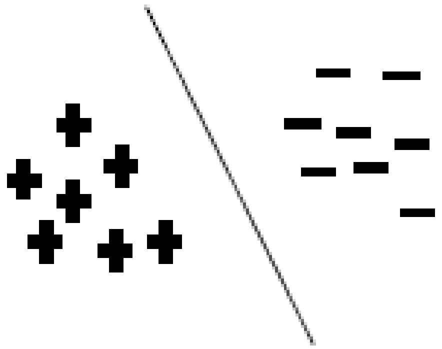
Space Transformation and Learning



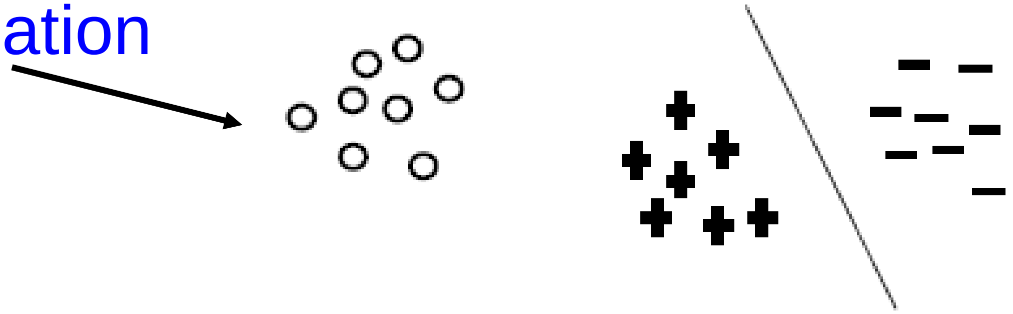
- We can use many similarity measures.
- After space transformation, we can run SVM to build a classification in the CBS space
  - CBS learning basically finds a ball for each class

Why does CBS Learning Work?

- SVM classifier

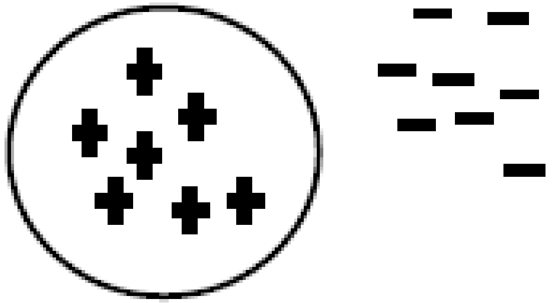


- SVM classification (test)
  - Wrong classification

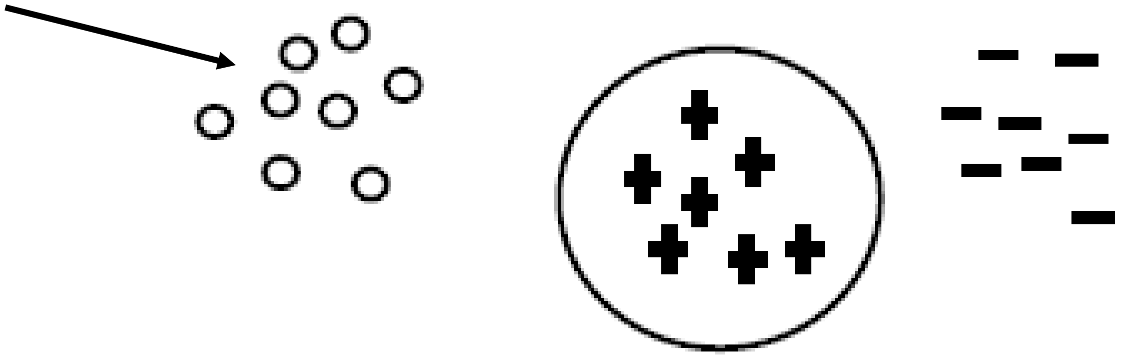


Why does CBS Learning Work?

- CBS classifier



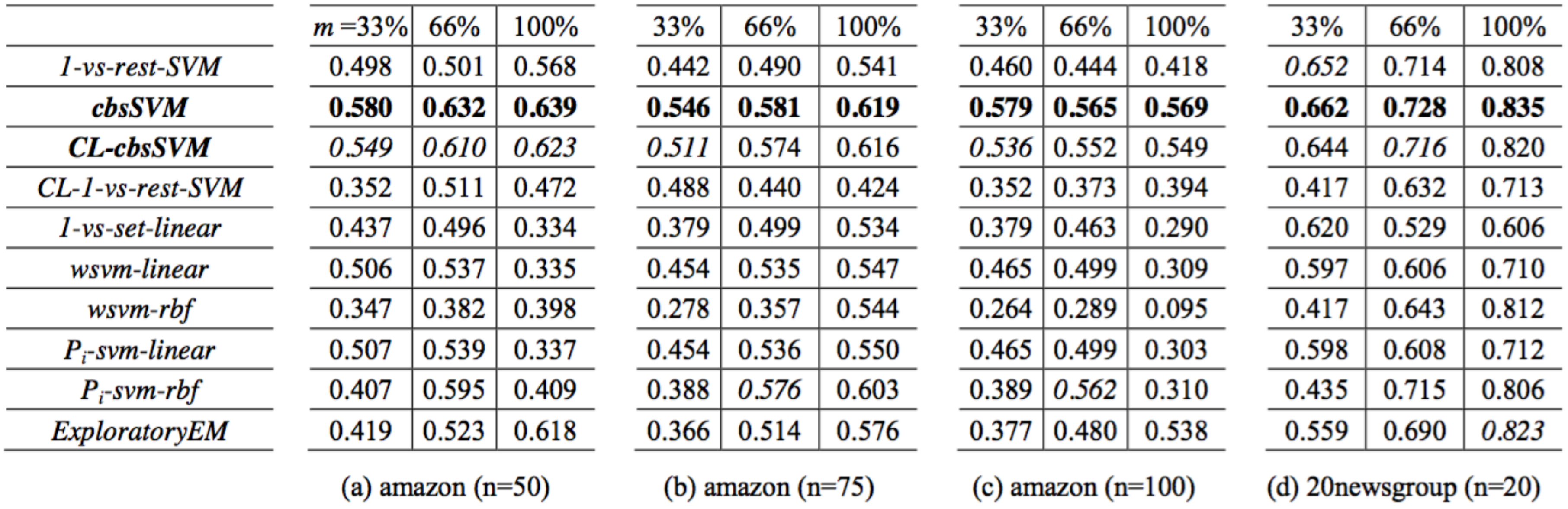
- CBS classification (test)
  - Correct now



Evaluation

Datasets

- Amazon reviews of 100 domains.
- 20 classes in 20newsgroup.



	m =33%	66%	100%	33%	66%	100%	33%	66%	100%	33%	66%	100%
1-vs-rest-SVM	0.498	0.501	0.568	0.442	0.490	0.541	0.460	0.444	0.418	*0.652*	0.714	0.808
cbsSVM	**0.580**	**0.632**	**0.639**	**0.546**	**0.581**	**0.619**	**0.579**	**0.565**	**0.569**	**0.662**	**0.728**	**0.835**
CL-cbsSVM	*0.549*	*0.610*	*0.623*	*0.511*	0.574	0.616	*0.536*	0.552	0.549	0.644	*0.716*	0.820
CL-1-vs-rest-SVM	0.352	0.511	0.472	0.488	0.440	0.424	0.352	0.373	0.394	0.417	0.632	0.713
1-vs-set-linear	0.437	0.496	0.334	0.379	0.499	0.534	0.379	0.463	0.290	0.620	0.529	0.606
wsvm-linear	0.506	0.537	0.335	0.454	0.535	0.547	0.465	0.499	0.309	0.597	0.606	0.710
wsvm-rbf	0.347	0.382	0.398	0.278	0.357	0.544	0.264	0.289	0.095	0.417	0.643	0.812
P_i-svm-linear	0.507	0.539	0.337	0.454	0.536	0.550	0.465	0.499	0.303	0.598	0.608	0.712
P_i-svm-rbf	0.407	0.595	0.409	0.388	*0.576*	0.603	0.389	*0.562*	0.310	0.435	0.715	0.806
ExploratoryEM	0.419	0.523	0.618	0.366	0.514	0.576	0.377	0.480	0.538	0.559	0.690	*0.823*
	(a) amazon (n=50)			(b) amazon (n=75)			(c) amazon (n=100)			(d) 20newsgroup (n=20)		

LML Components in this Case

- **KB**
  - Previous model $F_t = \{f_1, f_2, \ldots, f_t\}$
  - Training data from previous tasks
- **KBL**
  - Cumulative learning algorithm

20 Minutes Break

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

LTM: Lifelong Topic Modeling
(Chen and Liu, ICML-2014)

- Topic modeling (Blei et al 2003) finds topics from a collection of documents.
  - A document is a distribution over topics
  - A topic is a distribution over terms/words, e.g.,
    - {*price*, *cost*, *cheap*, *expensive*, …}

LTM: Lifelong Topic Modeling
(Chen and Liu, ICML-2014)

- Topic modeling (Blei et al 2003) finds topics from a collection of documents.
  - A document is a distribution over topics
  - A topic is a distribution over terms/words, e.g.,
    - {*price*, *cost*, *cheap*, *expensive*, …}
- **Question**: how to find good past knowledge and use it to help new topic modeling tasks?
- **Data**: product reviews in the sentiment analysis context

Sentiment Analysis (SA) Context

- "*The size is great, but pictures are poor*."
  - Aspects (product features): size, picture
- Why lifelong learning can help SA?
  - Online reviews: Excellent data with extensive sharing of aspect/concepts across domains
    - A large volume for all kinds of products
- Why big (and diverse) data?
  - Learn a broad range of reliable knowledge. More knowledge makes future learning easier.

Key Observation in Practice

- A fair amount of aspect overlapping across reviews of different products or domains
  - Every product review domain has the aspect *price*,
  - Most electronic products share the aspect *battery*
  - Many also share the aspect of *screen*.
- This sharing of concepts / knowledge across domains is true in general, not just for SA.
  - It is rather “silly” not to exploit such sharing in learning

Problem setting

- Given a large set of document collections (big data), $D = \{D_1, D_2, \ldots, D_N\}$, learn from each D_i to produce the results S_i. Let $S = \bigcup_i S_i$.
  - S is called *topic base*
- Goal: Given a test/new collection D^t, learn from D^t with the help of S (and possibly D).
  - D^t in D or D^t not in D
  - The results learned this way should be better than those without the guidance of S (and D)

What knowledge?

- Should be in the same aspect/topic

  => Must-Links

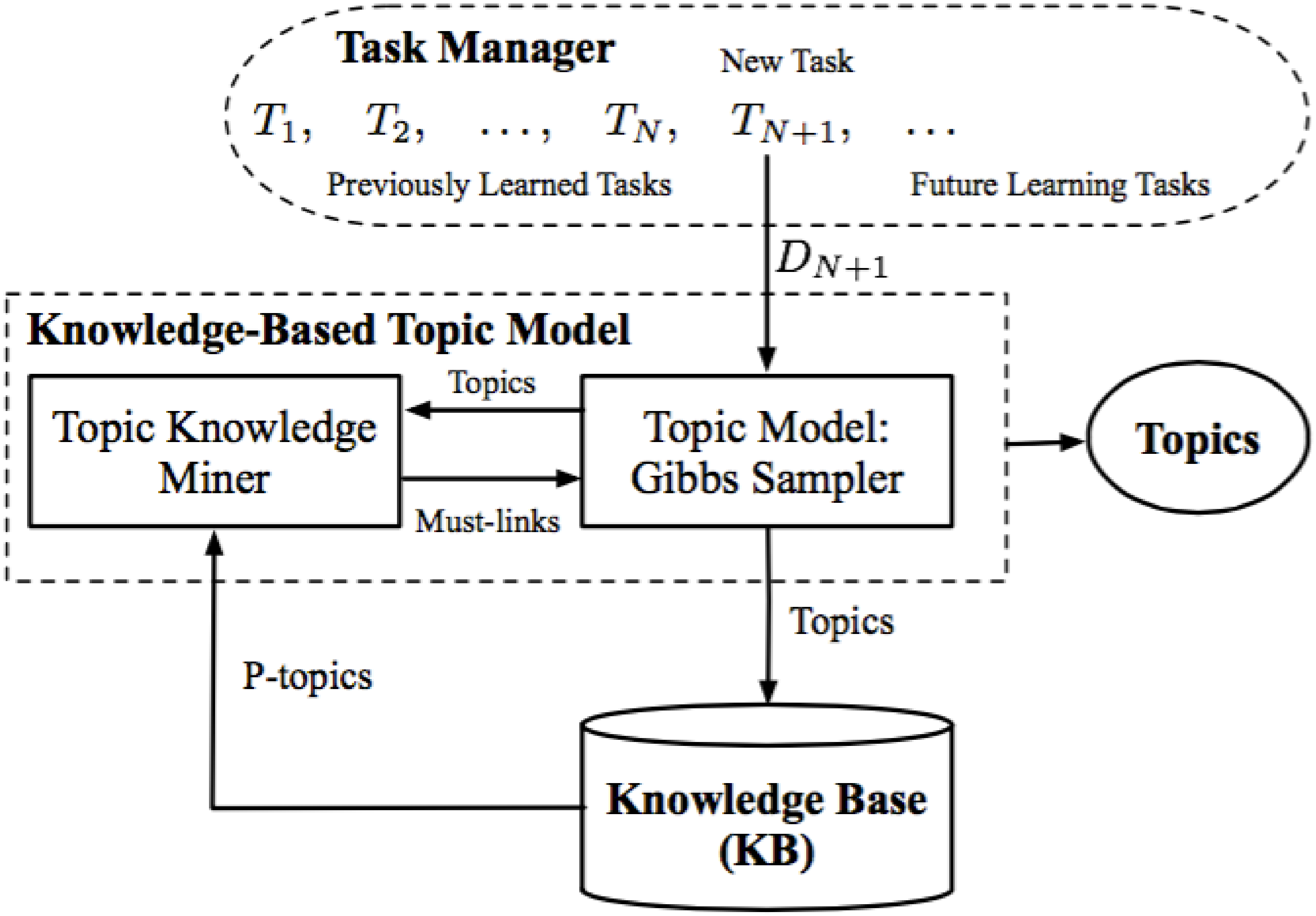
  e.g., {picture, photo}

- Should not be in the same aspect/topic

  => Cannot-Links

  e.g., {battery, picture}

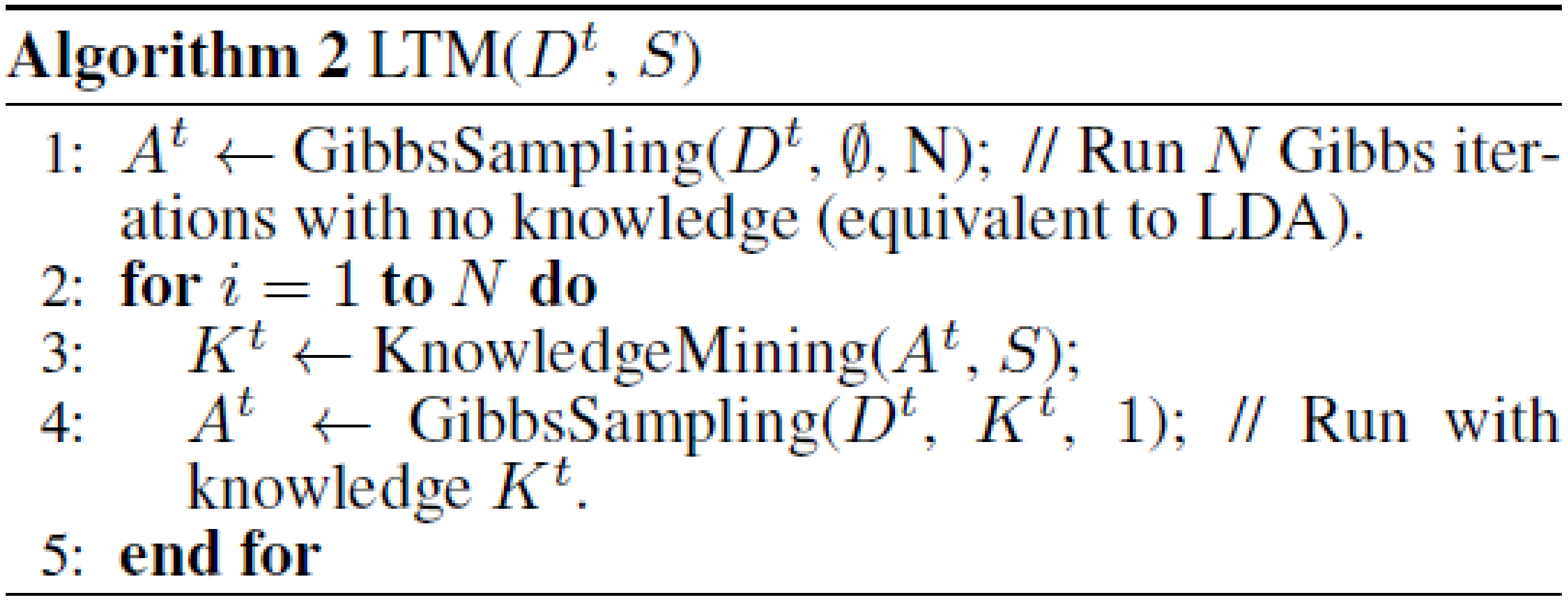
LTM System


Task Manager
New Task
T_1, T_2, …, T_N, T_{N+1}, …
Previously Learned Tasks
Future Learning Tasks
D_{N+1}
Knowledge-Based Topic Model
Topic Knowledge Miner
Topics
Must-links
Topic Model: Gibbs Sampler
Topics
Topics
P-topics
Knowledge Base (KB)

LTM Model

- Step 1: Run a topic model (e.g., LDA) on each domain D_i to produce a set of topics S_i called Topic Base
- Step 2: Mine prior knowledge (must-links) and use knowledge to guide modeling.

LTM Model



Algorithm 2 LTM(D^t, S)

```
1: $A^t \leftarrow$ GibbsSampling($D^t$, $\emptyset$, N); // Run $N$ Gibbs iterations with no knowledge (equivalent to LDA).
2: for $i = 1$ to $N$ do
3:     $K^t \leftarrow$ KnowledgeMining($A^t$, $S$);
4:     $A^t \leftarrow$ GibbsSampling($D^t$, $K^t$, 1); // Run with knowledge $K^t$.
5: end for
```

Knowledge Mining Function

- Topic matching: find similar topics from topic base for each topic in the new domain

- Pattern mining: find frequent itemsets from the matched topics

An Example

- Given a newly discovered topic:

  {*price*, *book*, *cost*, *seller, money*}

  - We find 3 matching topics from topic base *S*
    - Domain 1: {*price*, *color*, *cost*, *life, picture*}
    - Domain 2: {*cost*, *screen*, *price, expensive, voice*}
    - Domain 3: {*price*, *money*, *customer, expensive*}

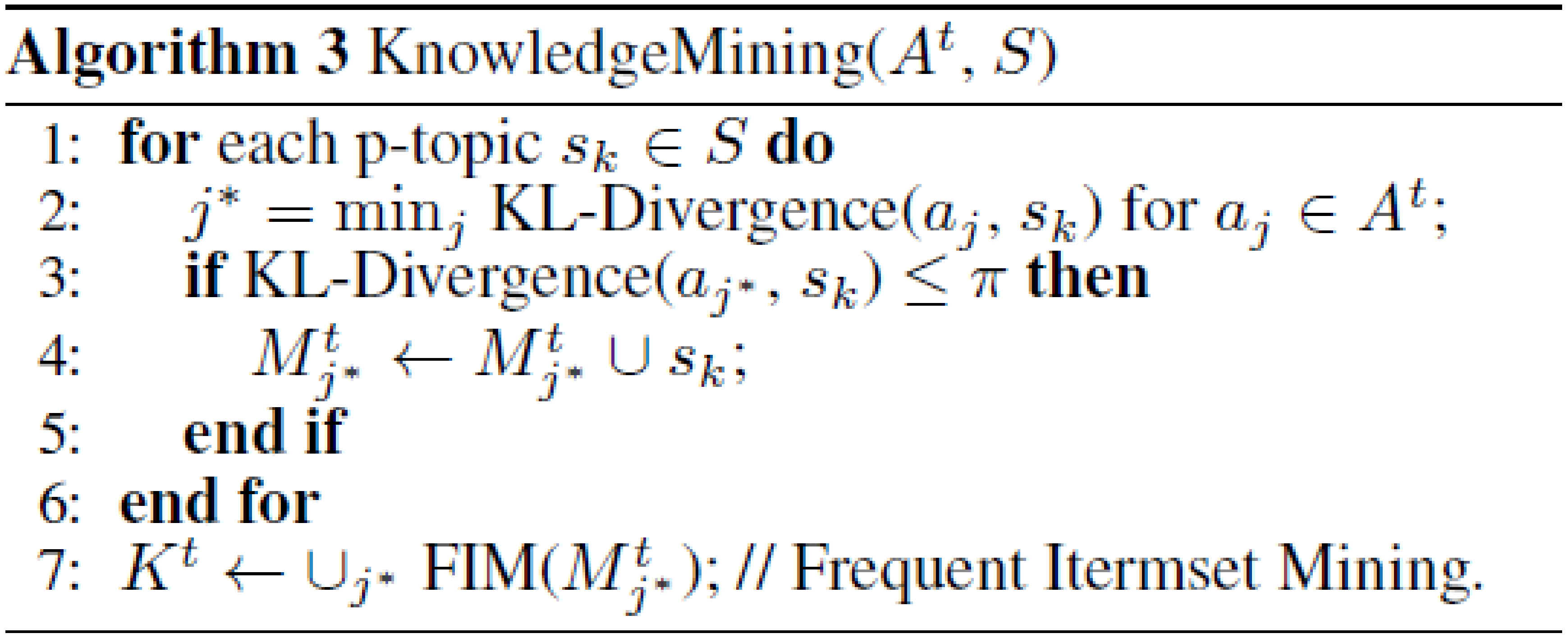
An Example

- Given a newly discovered topic:

  {*price*, *book*, *cost*, *seller, money*}

  - We find 3 matching topics from topic base *S*
    - Domain 1: {*price*, *color*, *cost*, *life, picture*}
    - Domain 2: {*cost*, *screen*, *price, expensive, voice*}
    - Domain 3: {*price*, *money*, *customer, expensive*}
- If we require words to appear in at least two domains, we get two must-links (knowledge):
  - {*price*, *cost*} and {*price*, *expensive*}.
  - Each set is likely to belong to the same aspect/topic.

Knowledge Mining Function

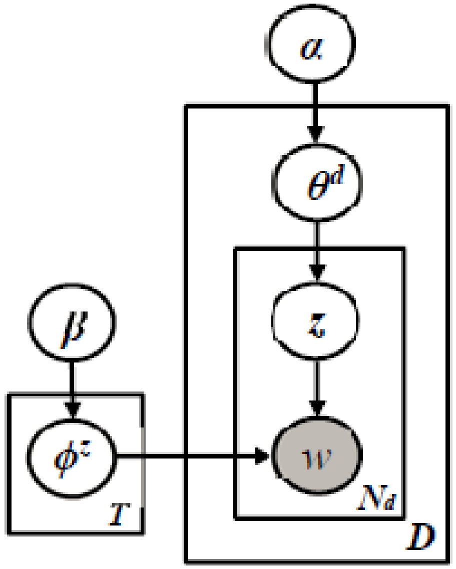


Algorithm 3 KnowledgeMining(A^t, S)

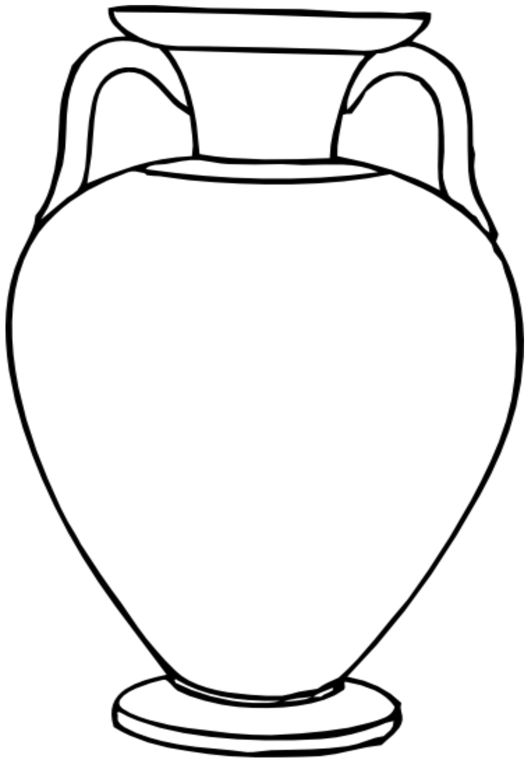
```
1: for each p-topic $s_k \in S$ do
2:     $j^* = \min_j$ KL-Divergence($a_j, s_k$) for $a_j \in A^t$;
3:     if KL-Divergence($a_{j^*}, s_k$) $\leq \pi$ then
4:         $M^t_{j^*} \leftarrow M^t_{j^*} \cup s_k$;
5:     end if
6: end for
7: $K^t \leftarrow \cup_{j^*}$ FIM($M^t_{j^*}$); // Frequent Itermset Mining.
```

Model Inference: Gibbs Sampling

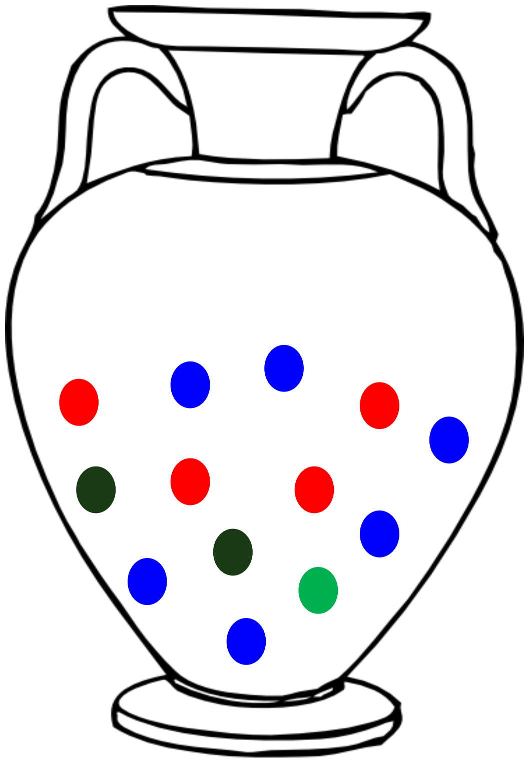
- How to use the *must-links* knowledge?
  - e.g., {*price*, *cost*} & {*price*, *expensive*}
- Graphical model: same as LDA
- But the model inference is very different
  - Generalized Pólya Urn Model (GPU)
- Idea: When assigning a topic *t* to a word *w*, also assign *a fraction of t* to words in must-links sharing with *w*.



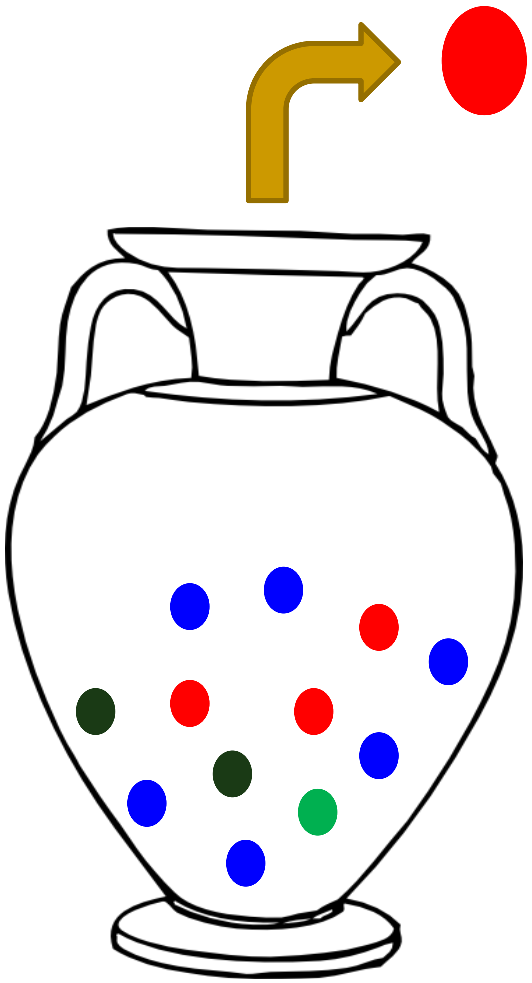
Simple Pólya Urn Model (SPU)



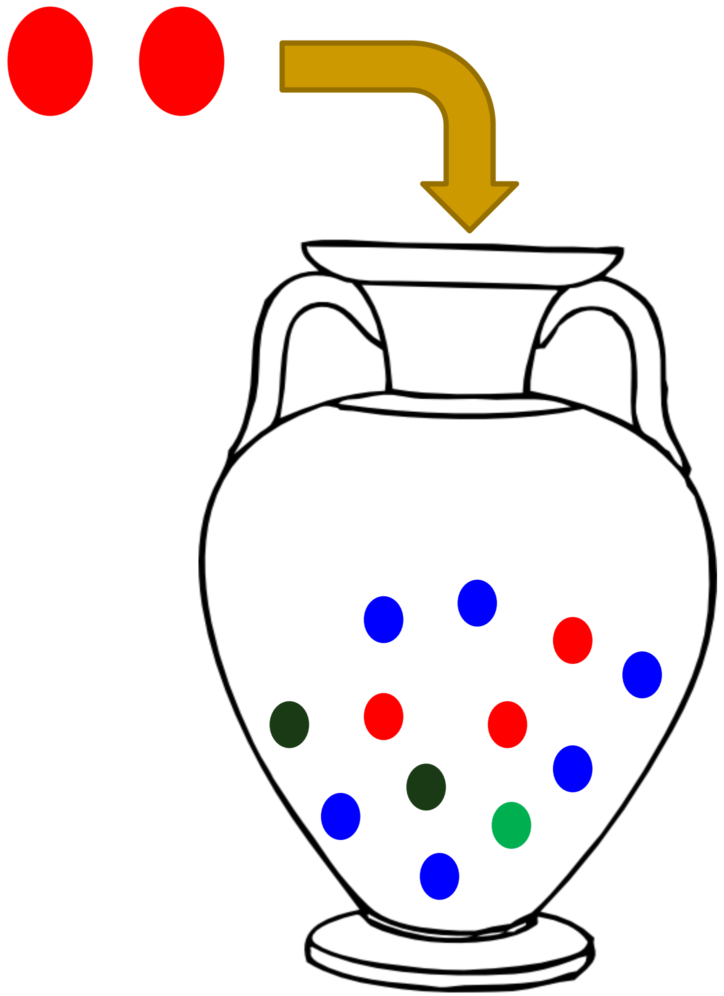
Simple Pólya Urn Model (SPU)



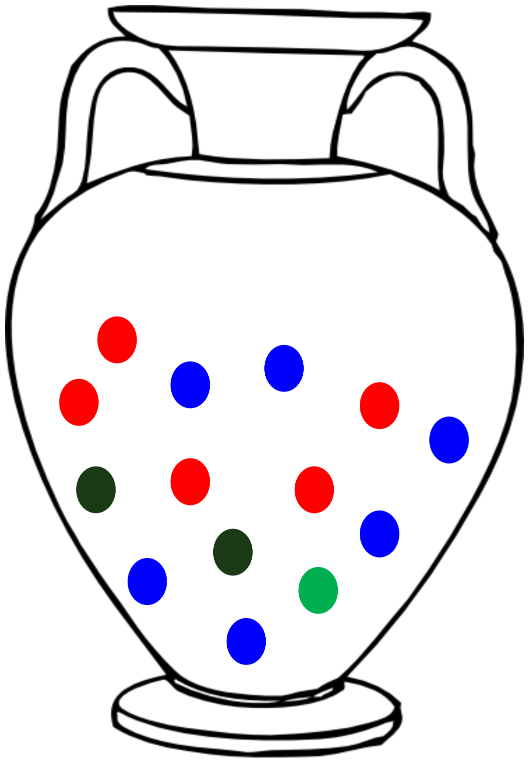
Simple Pólya Urn Model (SPU)



Simple Pólya Urn Model (SPU)

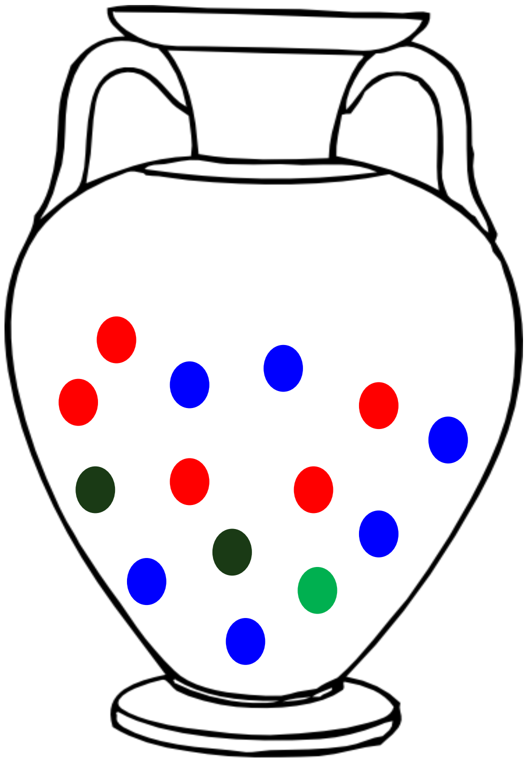


Simple Pólya Urn Model (SPU)



Simple Pólya Urn Model (SPU)

The rich get richer!



Interpreting LDA Under SPU

Drawing word w under a topic t:

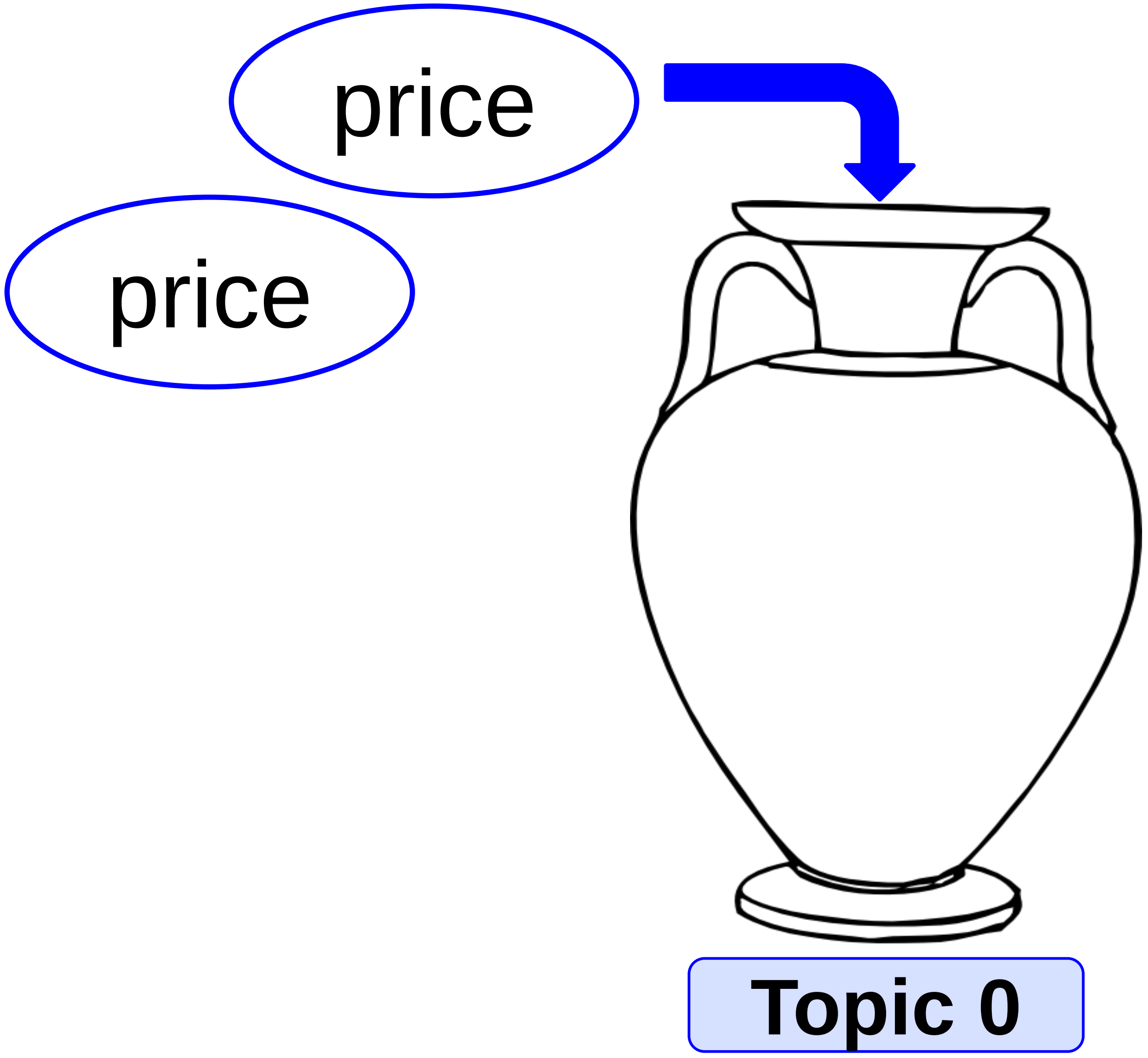
Increase the probability of seeing w under t

Decrease the probability of seeing $w' \neq w$ under t

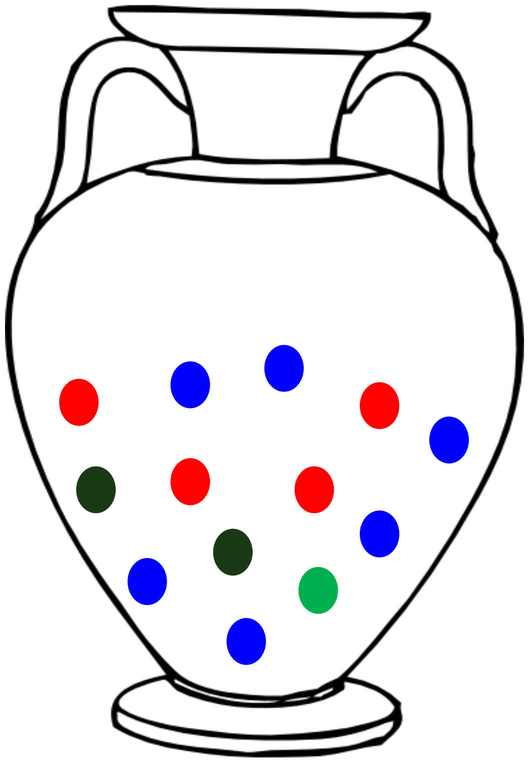
Interpreting LDA Under SPU

price
Topic 0

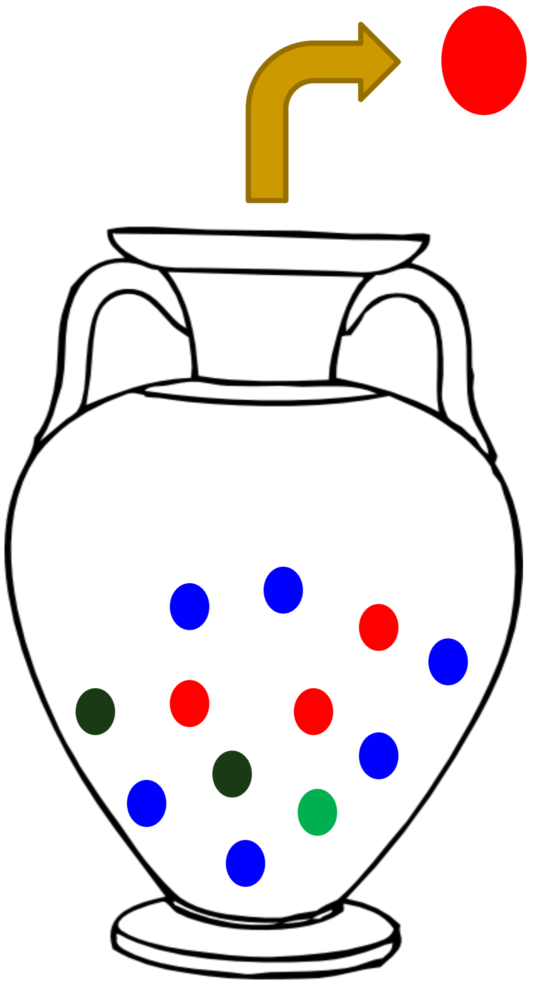
Interpreting LDA Under SPU



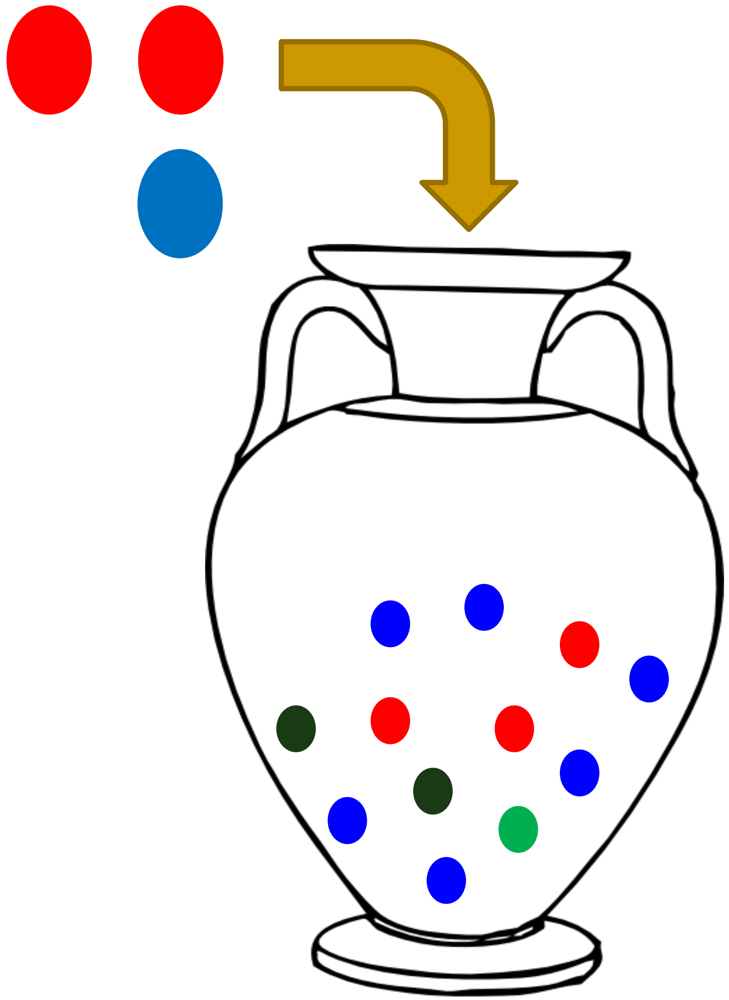
Generalized Pólya Urn Model (GPU)



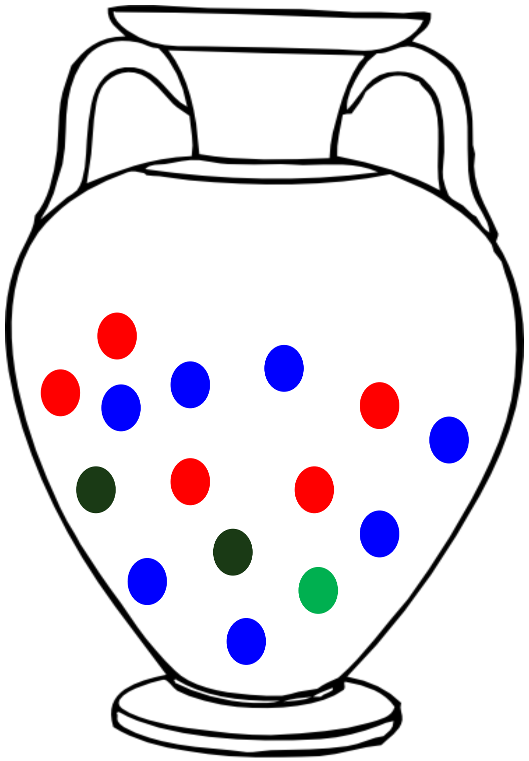
Generalized Pólya Urn Model (GPU)



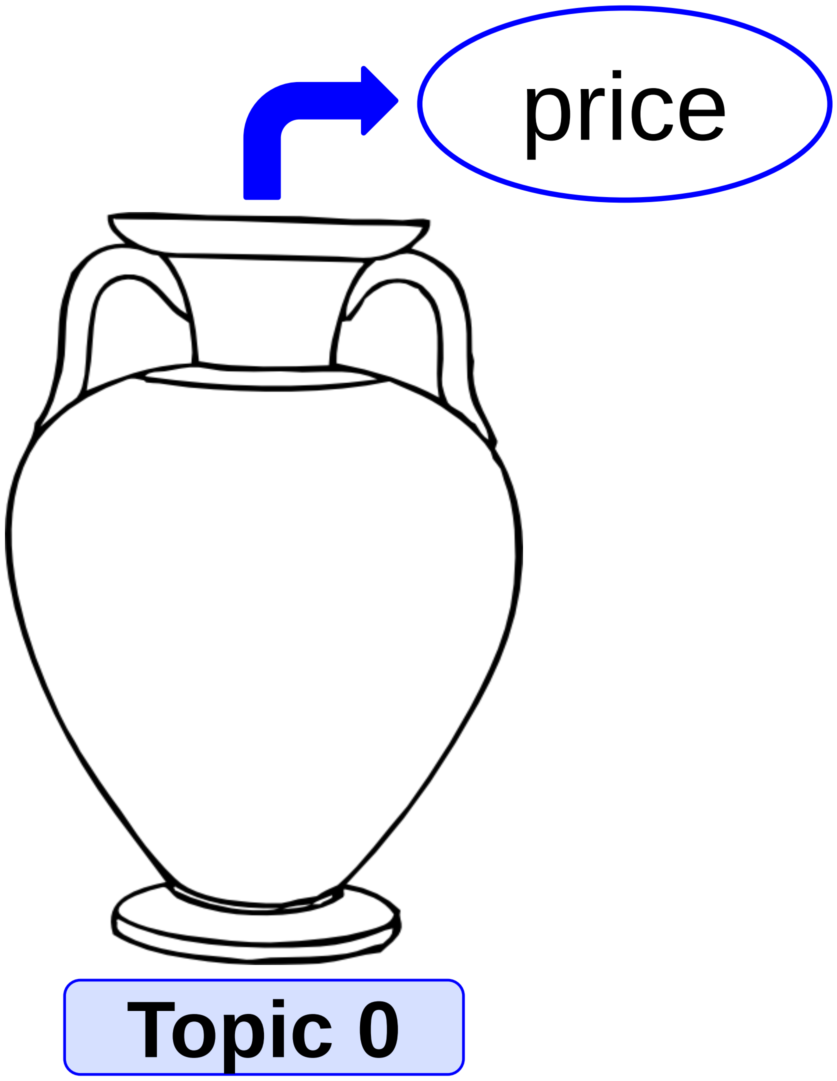
Generalized Pólya Urn Model (GPU)



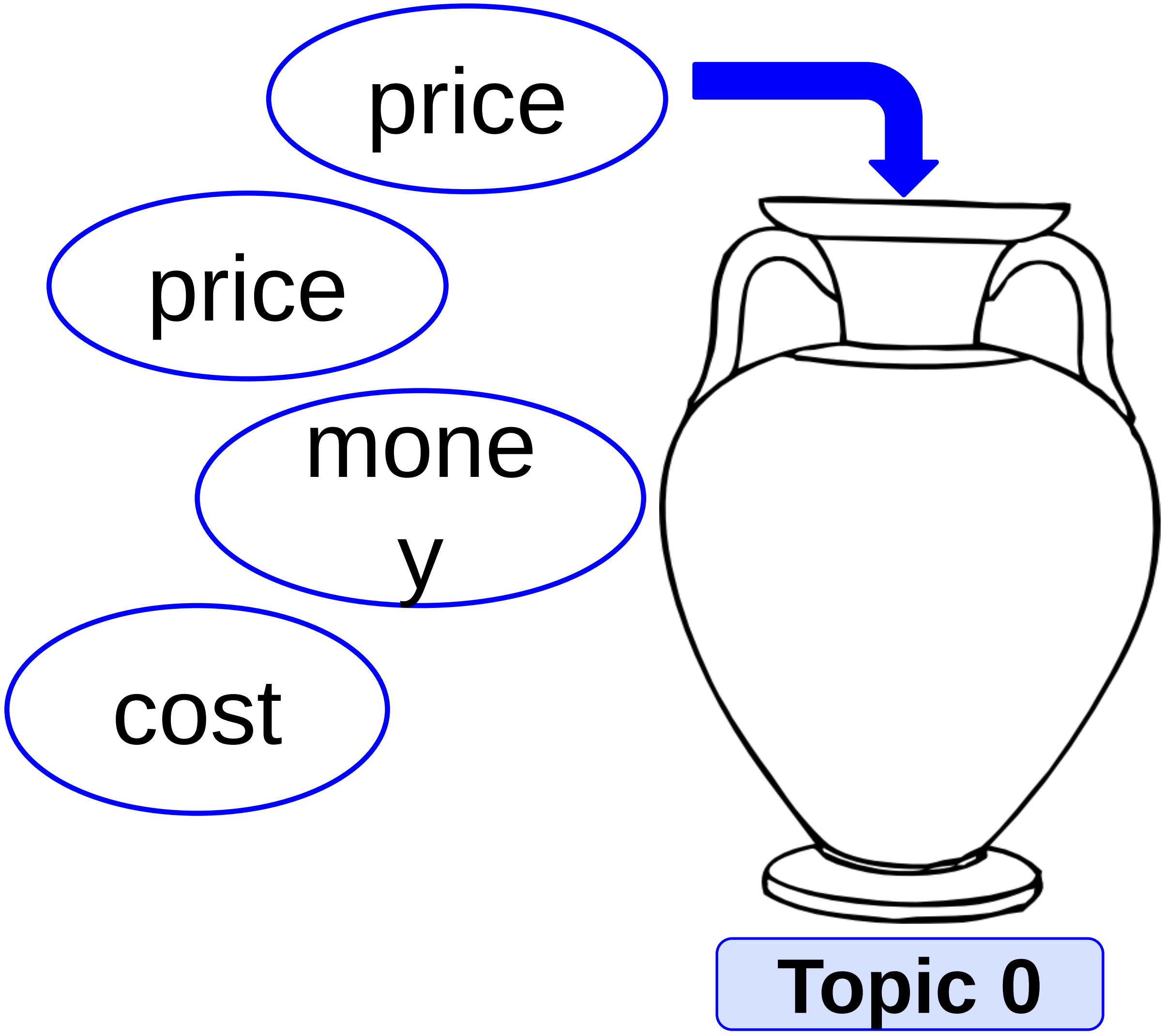
Generalized Pólya Urn Model (GPU)



Applying GPU



Applying GPU



Gibbs Sampling

$$P(z_i = t | \boldsymbol{z}^{-i}, \boldsymbol{w}, \alpha, \beta, \mathbf{A}') \propto$$

$$\frac{n_{d,t}^{-i} + \alpha}{\sum_{t'=1}^{T} (n_{d,t'}^{-i} + \alpha)} \times \frac{\sum_{w'=1}^{V} \mathbf{A}'_{t,w',w_i} \times n_{t,w'}^{-i} + \beta}{\sum_{v=1}^{V} (\sum_{w'=1}^{V} \mathbf{A}'_{t,w',v} \times n_{t,w'}^{-i} + \beta)}$$

Experiment Results

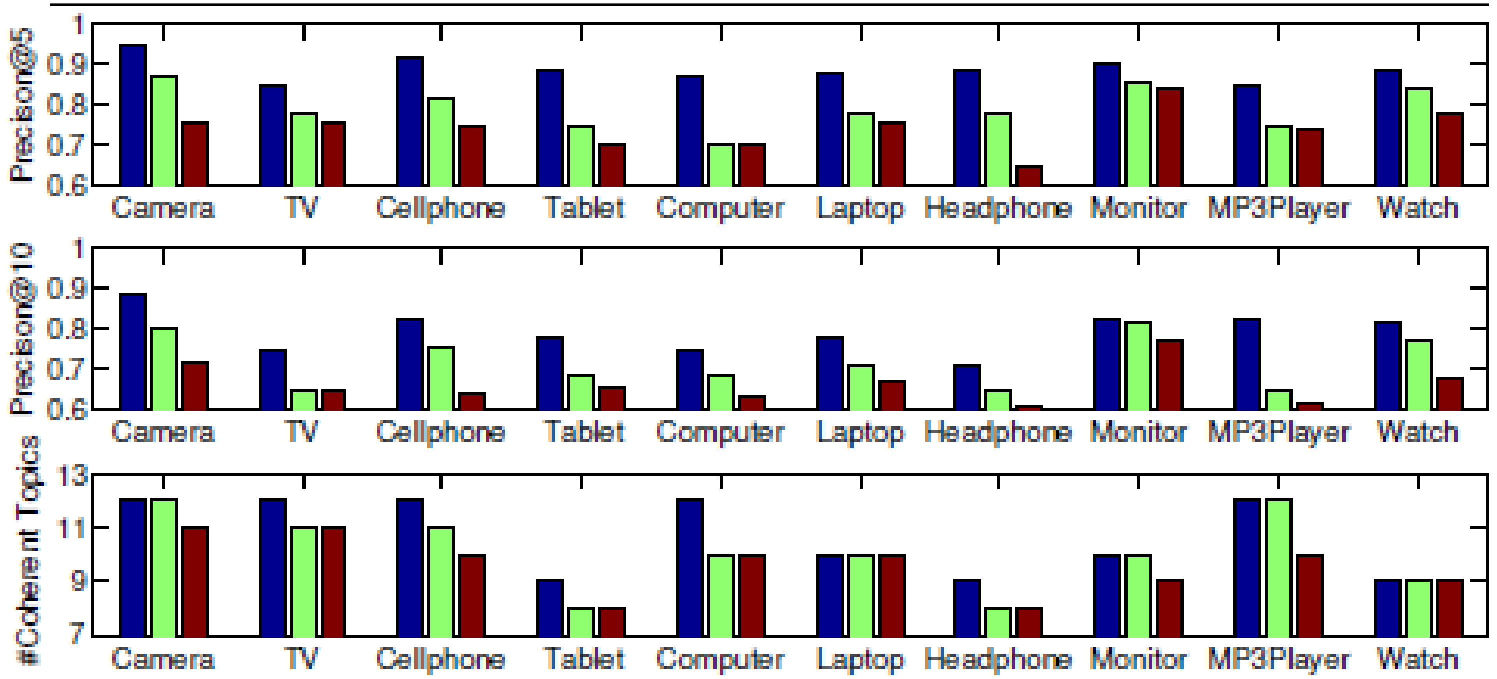


Figure 2. Top & Middle: Topical words *Precision@5* & *Presicion@10* of coherent topics of each model respectively; Bottom: number of coherent (#Coherent) topics discovered by each model. The bars from left to right in each group are for LTM, LDA, and DF-LDA. On average, for *Precision@5* and

LML Components of LTM

- **KB**
  - Stores topics/aspects generated in the past tasks
  - Knowledge: Must-Links
- **KBL**
  - LTM is based on Generalized Pólya Urn Model

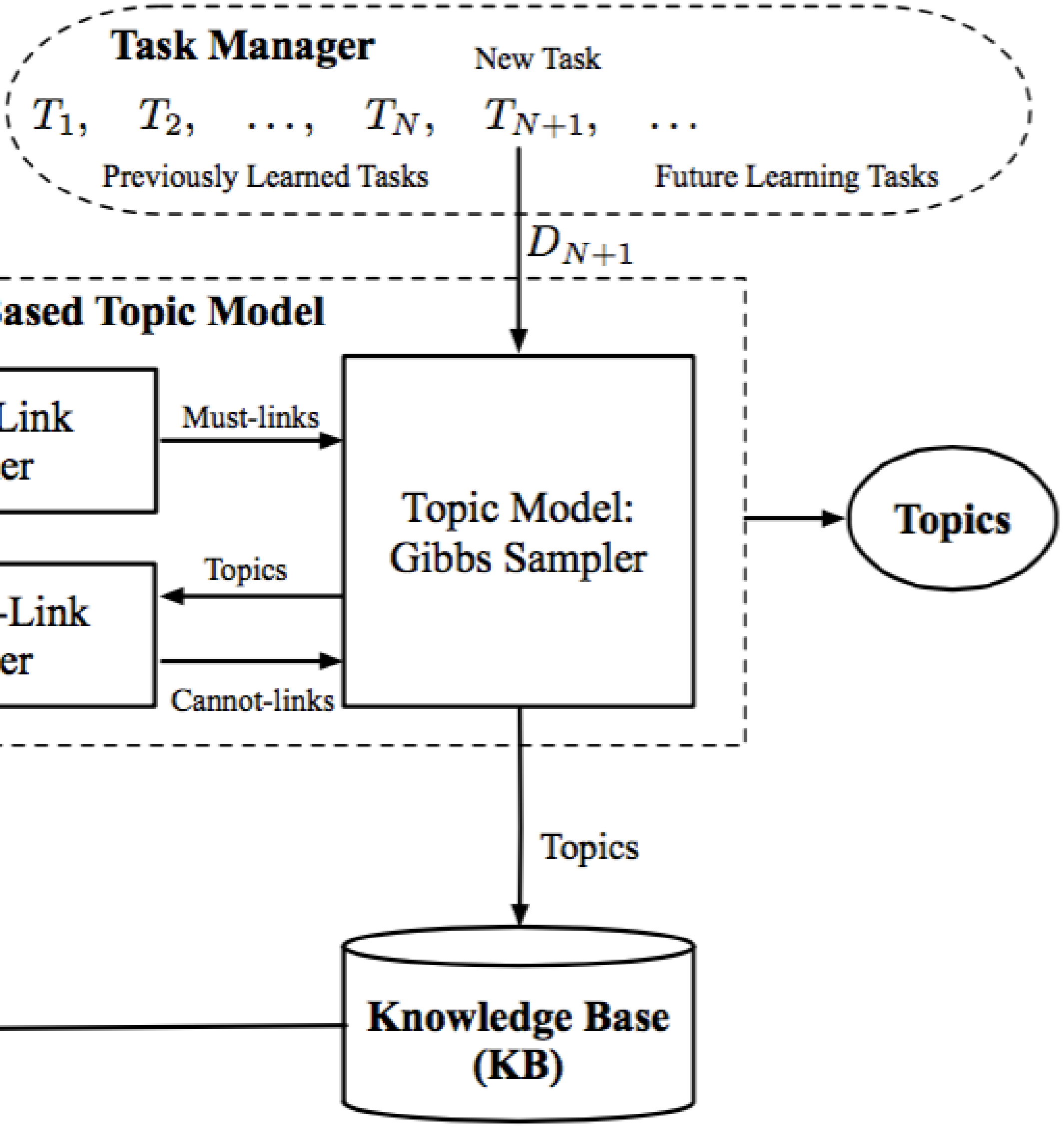
AMC: Modeling with Small Datasets
(Chen and Liu, KDD-2014)

- The LTM model is not sufficient when the data is small for each task because
    - It cannot produce good initial topics for matching to identify relevant past topics.
- AMC mines must-links differently
    - Mine must-links from the PIS without considering the target task/data

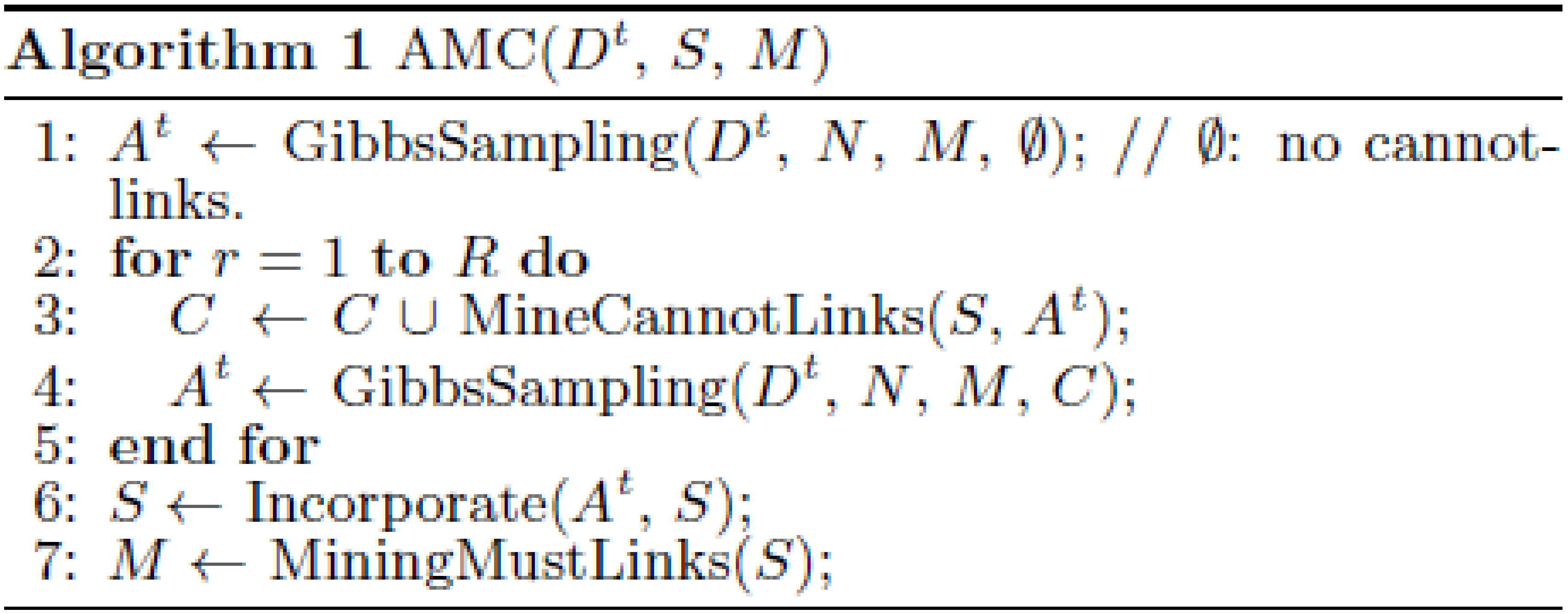
Cannot-Links

- In this case, we need to mine cannot-links, which is tricky because
  - There is a huge number of cannot-links $O(V^2)$
    - V is the vocabulary size
- We thus need to focus on only those terms that are relevant to target data D^t.
  - That is, we need to embed the process of finding cannot-links in the sampling

AMC System


Task Manager
New Task
T_1, T_2, …, T_N, T_{N+1}, …
Previously Learned Tasks
Future Learning Tasks
D_{N+1}
Knowledge-Based Topic Model
Must-Link Miner
Must-links
Topic Model: Gibbs Sampler
Topics
Cannot-Link Miner
Cannot-links
Topics
P-topics
Topics
Knowledge Base (KB)

Overall Algorithm

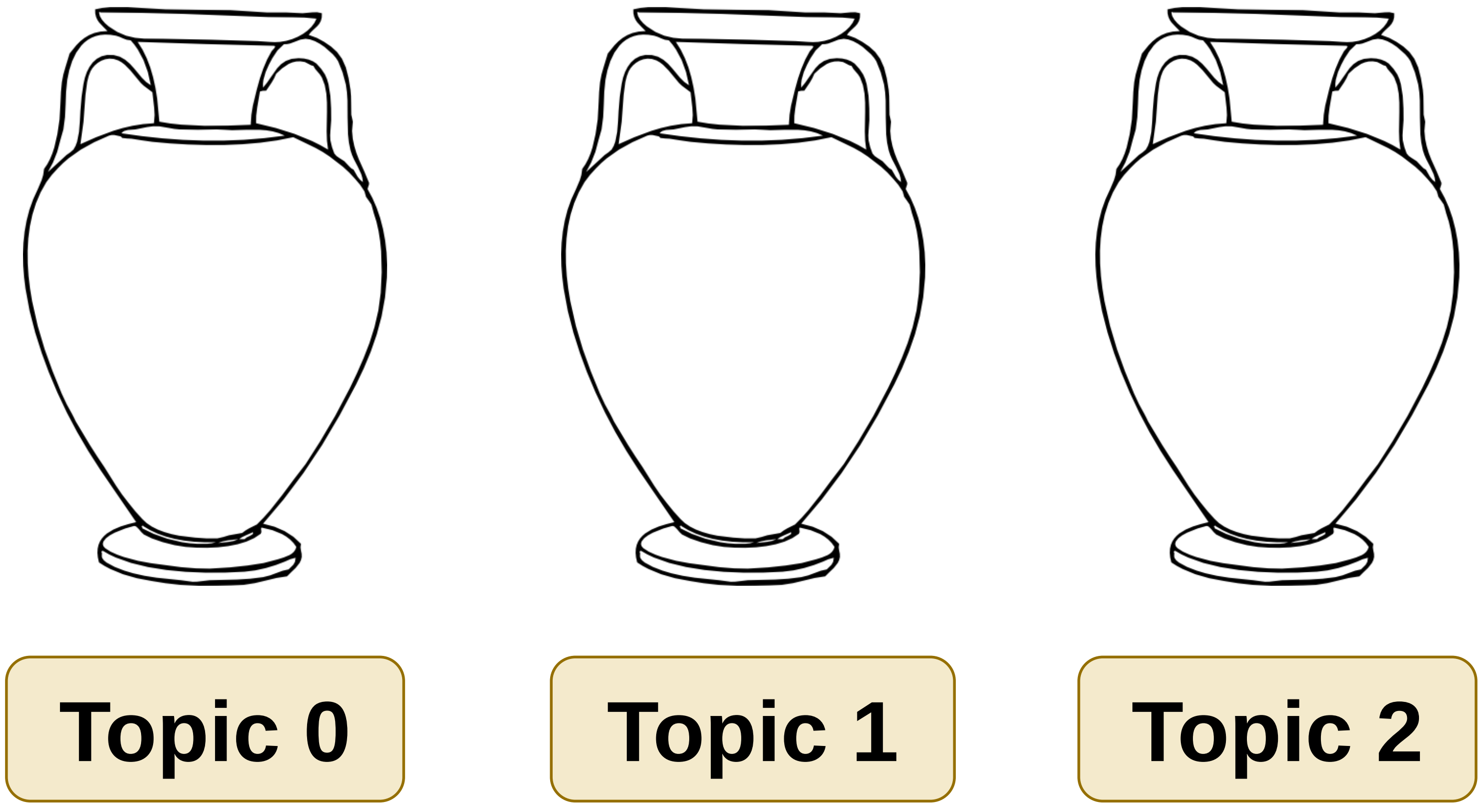


Algorithm 1 AMC(D^t, S, M)

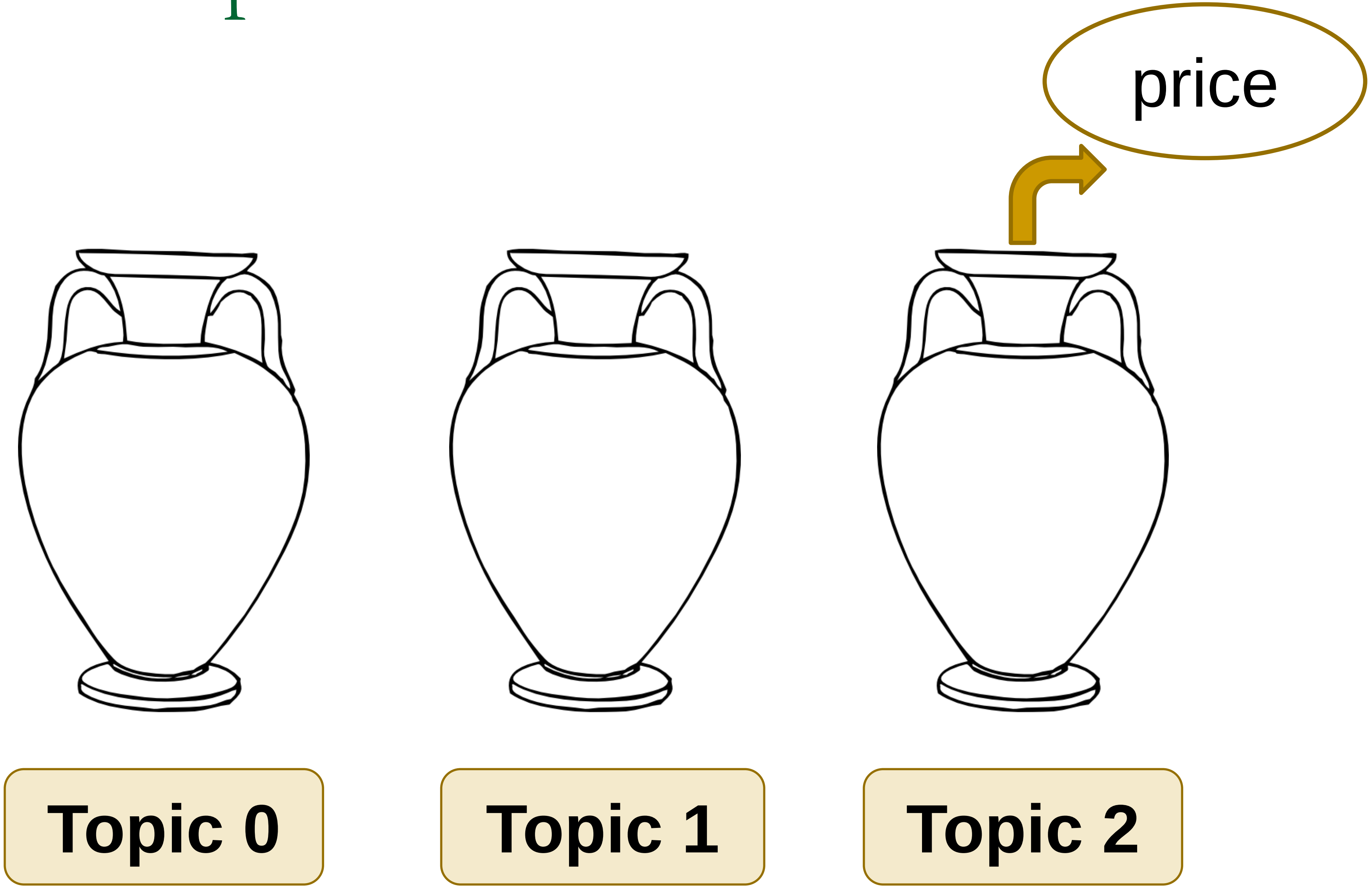
```
1: $A^t \leftarrow$ GibbsSampling($D^t$, $N$, $M$, $\emptyset$); // $\emptyset$: no cannot-links.
2: for $r = 1$ to $R$ do
3:     $C \leftarrow C \cup$ MineCannotLinks($S$, $A^t$);
4:     $A^t \leftarrow$ GibbsSampling($D^t$, $N$, $M$, $C$);
5: end for
6: $S \leftarrow$ Incorporate($A^t$, $S$);
7: $M \leftarrow$ MiningMustLinks($S$);
```

- Sampling becomes much more complex
  - It proposed M-GPU model (multi-generalized Polya urn model)

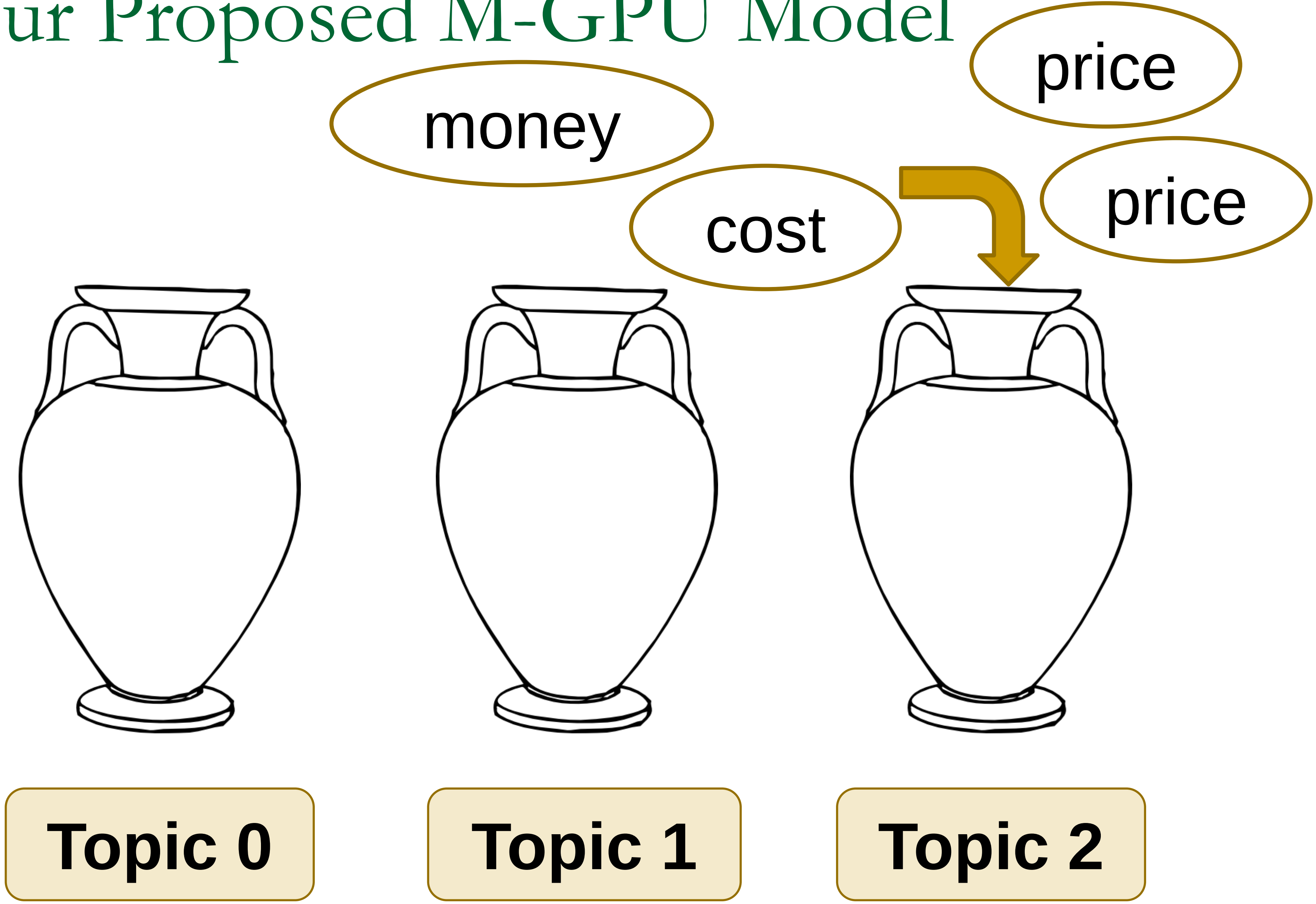
Our Proposed M-GPU Model



Our Proposed M-GPU Model

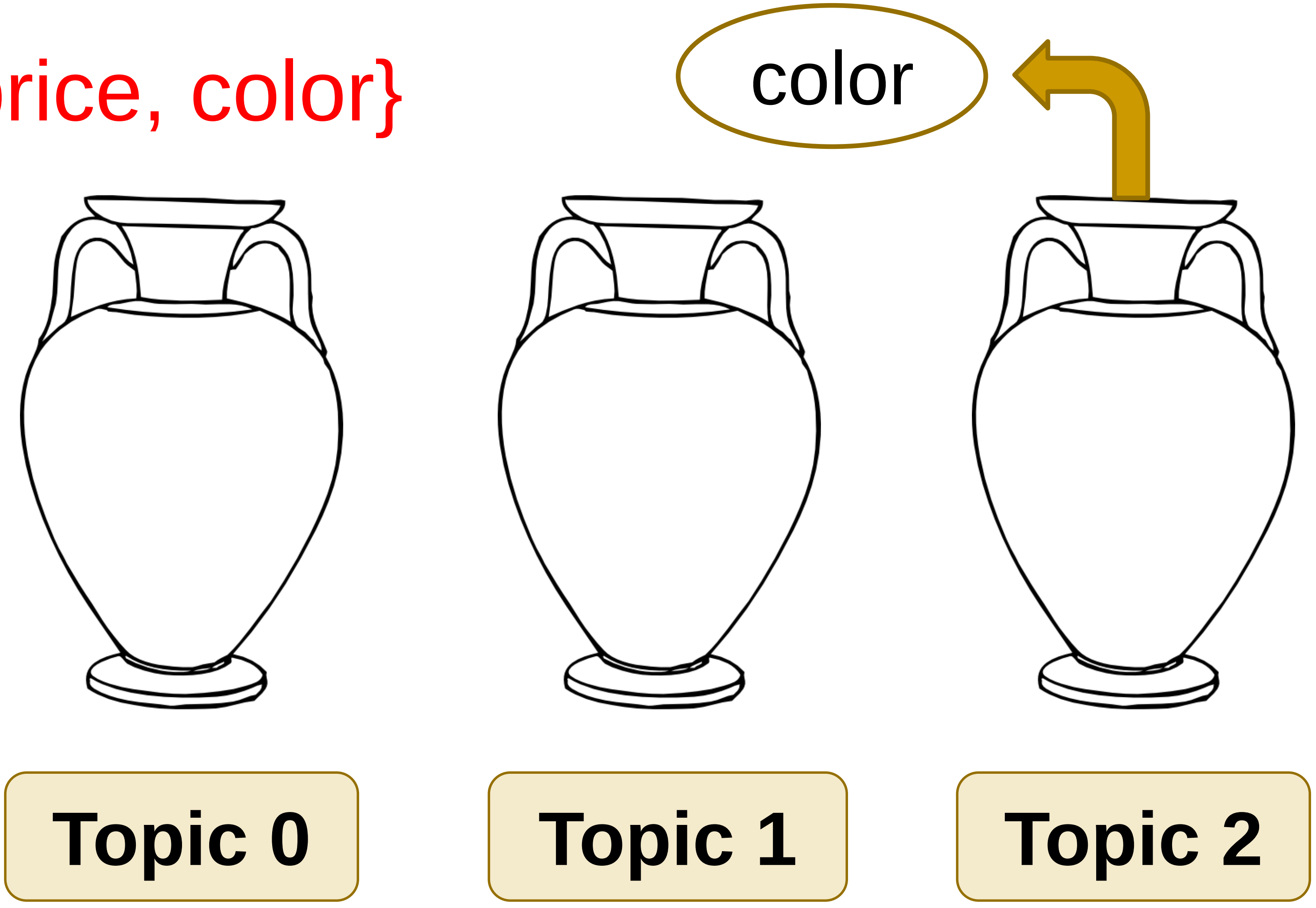


Our Proposed M-GPU Model



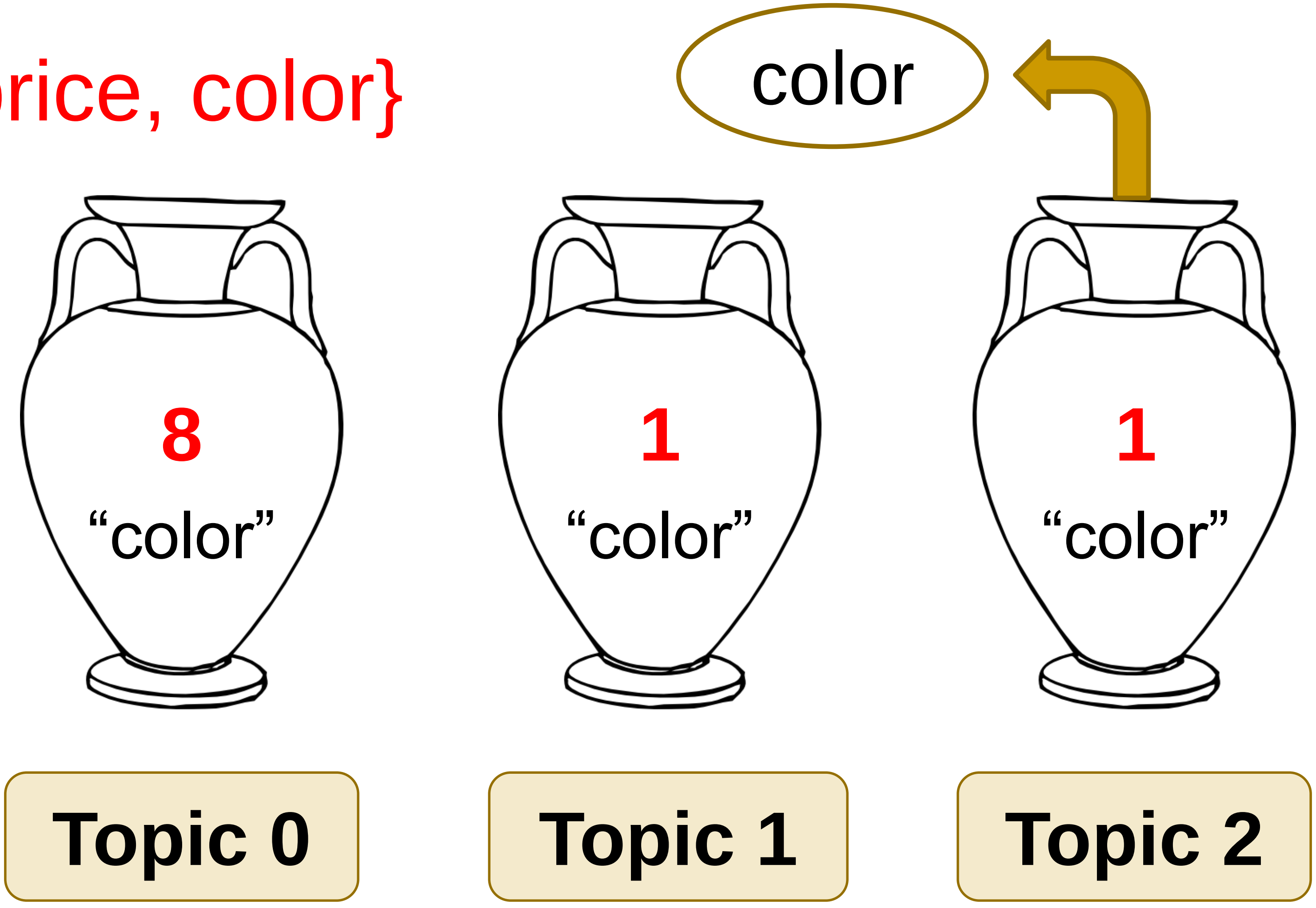
Our Proposed M-GPU Model

- {price, color}

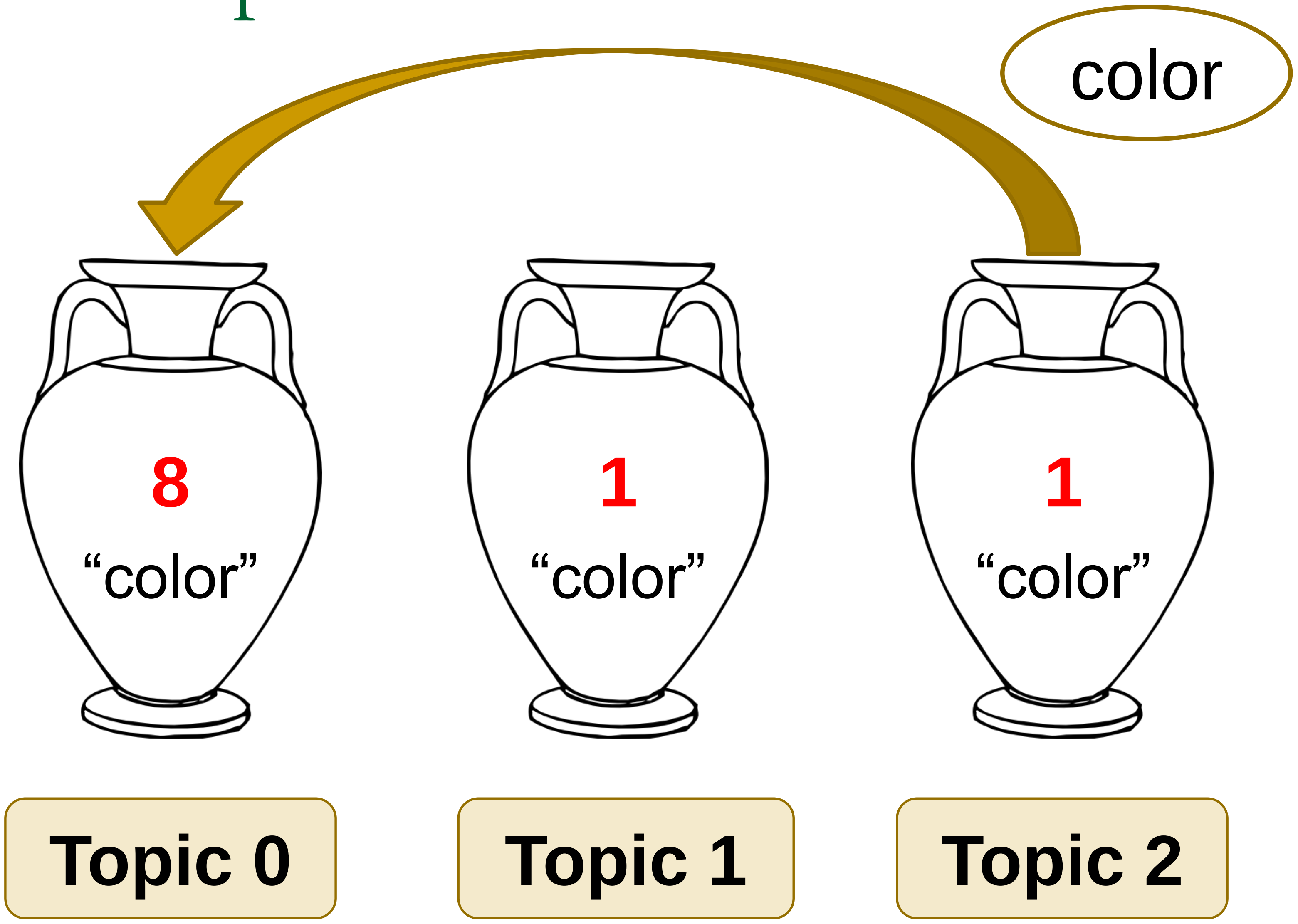


Our Proposed M-GPU Model

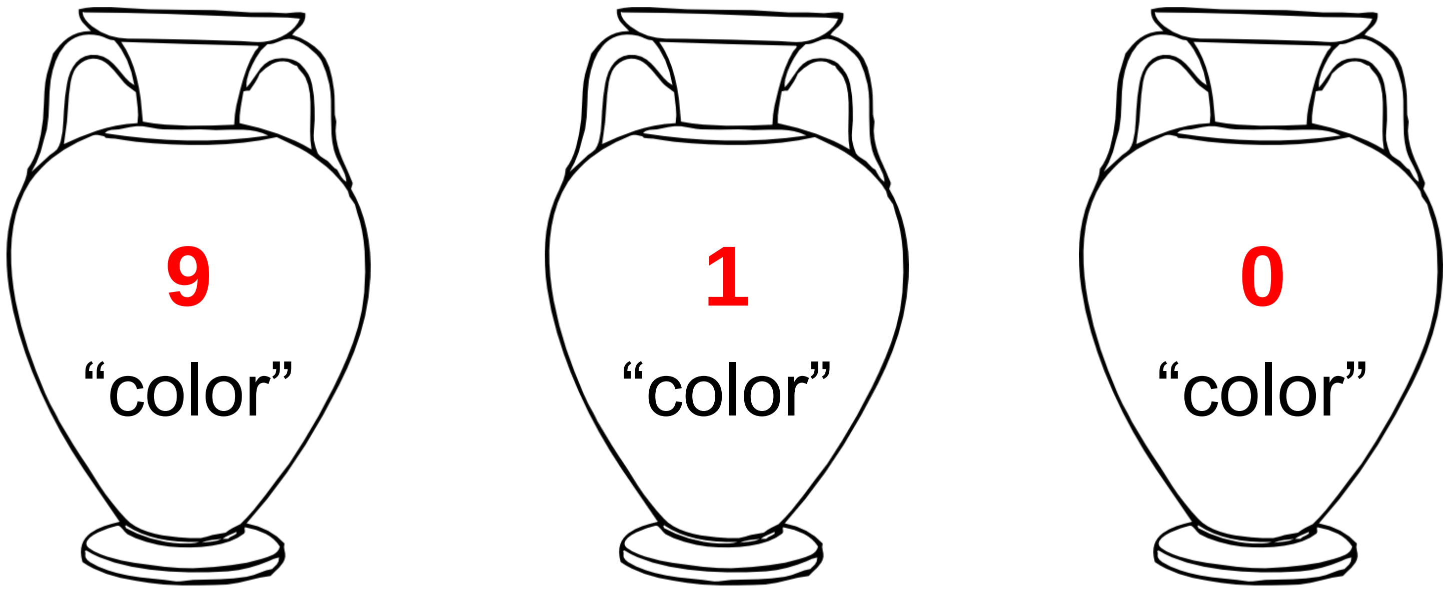
- {price, color}



Our Proposed M-GPU Model



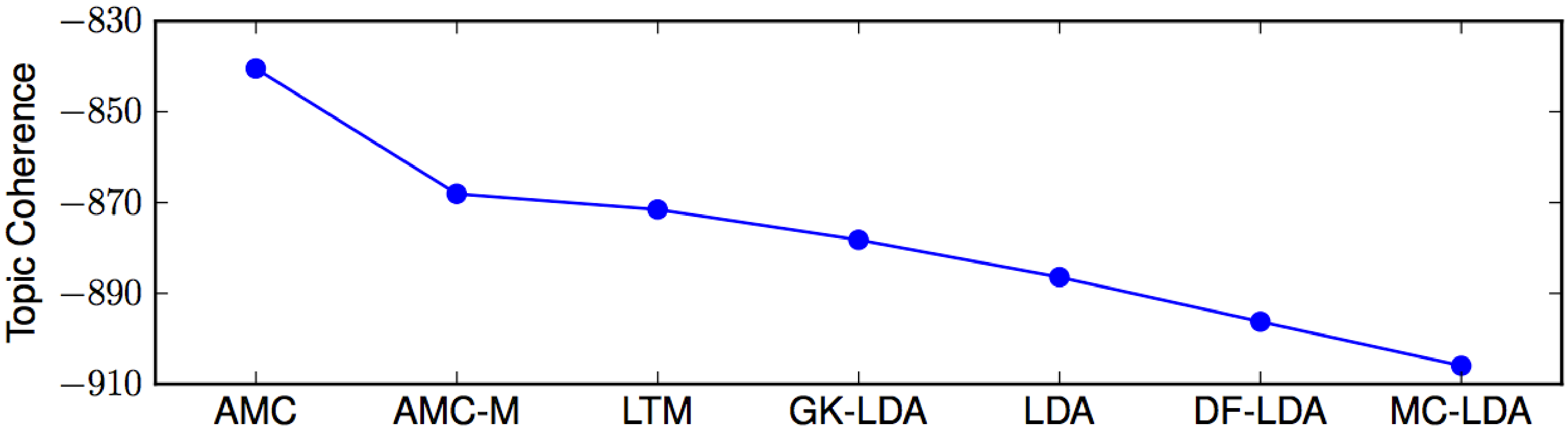
Our Proposed M-GPU Model



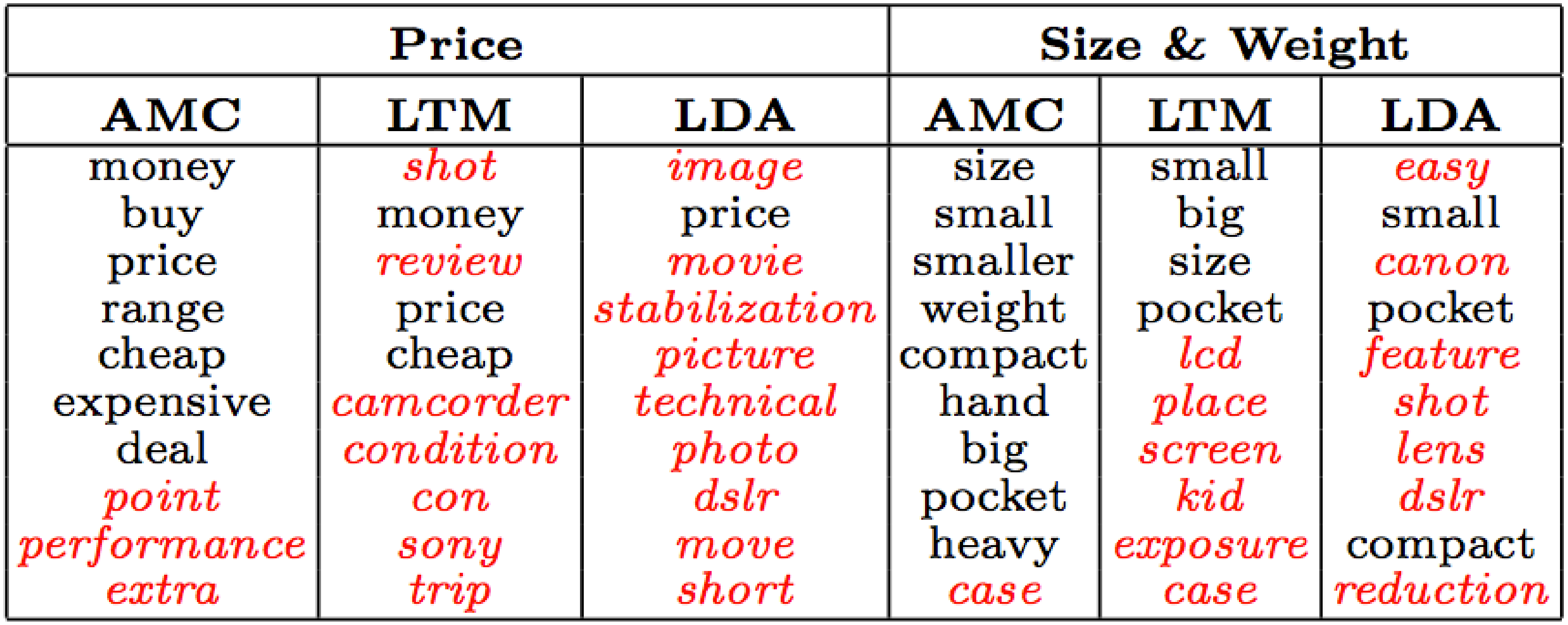
Topic 0 Topic 1 Topic 2

AMC results

Metrics: Topic Coherence (Mimno et al., 2011)



AMC results



Price			Size & Weight		
AMC	**LTM**	**LDA**	**AMC**	**LTM**	**LDA**
money	*shot*	*image*	size	small	*easy*
buy	money	price	small	big	small
price	*review*	*movie*	smaller	size	*canon*
range	price	*stabilization*	weight	pocket	pocket
cheap	cheap	*picture*	compact	*lcd*	*feature*
expensive	*camcorder*	*technical*	hand	*place*	*shot*
deal	*condition*	*photo*	big	*screen*	*lens*
point	*con*	*dslr*	pocket	*kid*	*dslr*
performance	*sony*	*move*	heavy	*exposure*	compact
extra	*trip*	*short*	*case*	*case*	*reduction*

Table 2: Example topics of AMC, LTM and LDA from the Camera domain. Errors are italicized and marked in red.

LML Components of AMC

- **KB**
  - Stores topics/aspects generated in the past tasks
  - Knowledge: Must-Links and Cannot-Links
- **KBL**
  - AMC is based on multi-generalized Pólya Urn Model

LAST Model

- Lifelong aspect-based sentiment topic model (Wang et al., 2016)
- Knowledge
  - Aspect-opinion pair, e.g., {shipping, quick}
  - Aspect-aspect pair, e.g., {shipping, delivery}
  - Opinion-opinion pair, e.g, {quick, fast}

Lifelong Information Extraction
(Liu et al., 2016)

- Specifically: aspect extraction
- "*The size is great, but pictures are poor*."
  - Aspects (product features): size, picture
- An effective approach
  - Double Propagation (DP) (Qiu et al 2011): a syntactic rule-based extraction method
    - Still has a lot of room for improvement.

Problem and Solution

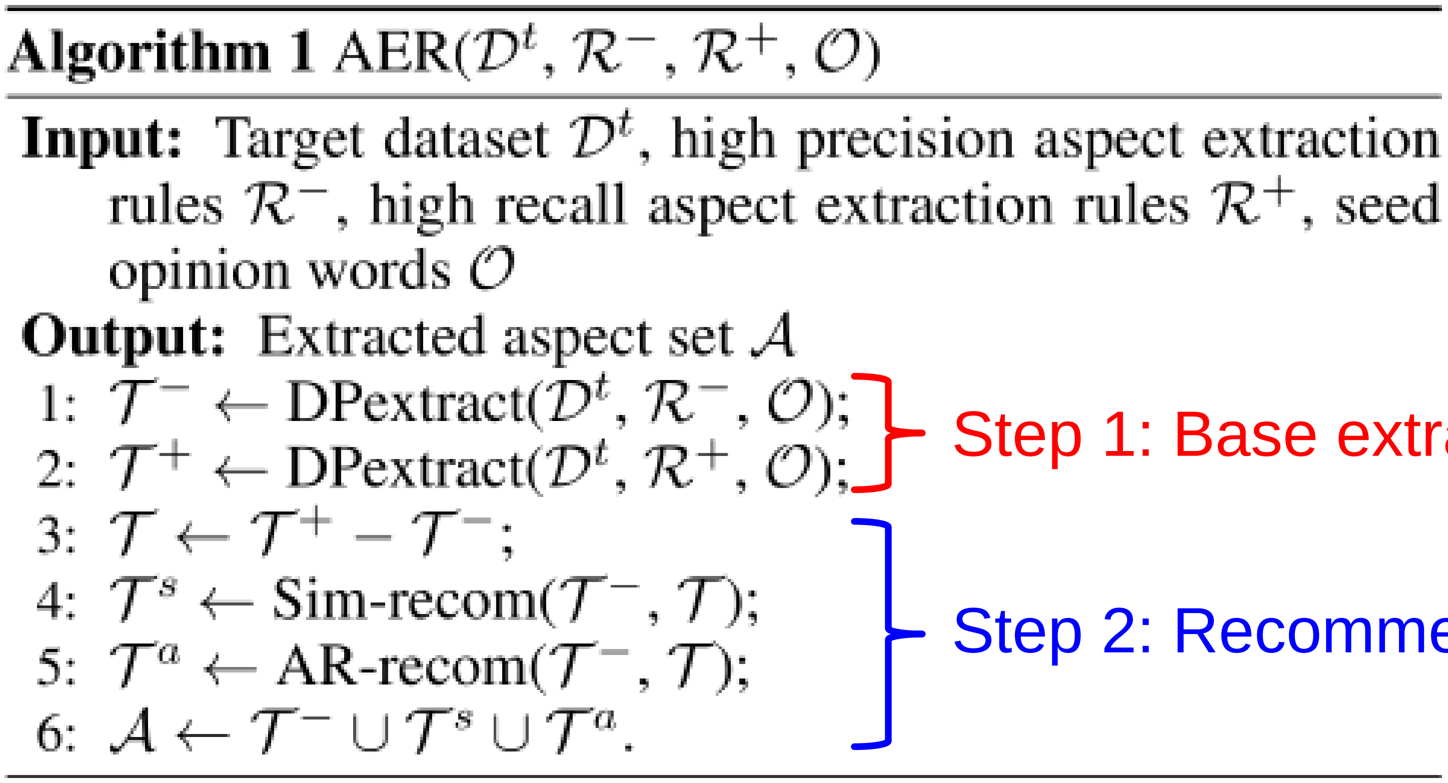
- Problem of syntactic rule-based methods
  - hard to design a set of rules to perform extraction with high precision and recall.
- Possible solution
  - Use prior knowledge mined by exploiting the abundance of reviews for all kinds of products since many products share aspects.
    - e.g., many electronic products have aspect battery.

How to Use Prior Knowledge?

- Use extracted aspects from reviews of a large number of other products to help extract aspects from reviews of the current product.
  - Using recommendation.

- This work uses DP as the base and improve its results dramatically through
  - aspect recommendation.

Overall Algorithm



Algorithm 1 AER($\mathcal{D}^t, \mathcal{R}^-, \mathcal{R}^+, \mathcal{O}$)

Input: Target dataset $\mathcal{D}^t$, high precision aspect extraction rules $\mathcal{R}^-$, high recall aspect extraction rules $\mathcal{R}^+$, seed opinion words $\mathcal{O}$

Output: Extracted aspect set $\mathcal{A}$

```
1: $\mathcal{T}^- \leftarrow$ DPextract($\mathcal{D}^t, \mathcal{R}^-, \mathcal{O}$);
2: $\mathcal{T}^+ \leftarrow$ DPextract($\mathcal{D}^t, \mathcal{R}^+, \mathcal{O}$);
3: $\mathcal{T} \leftarrow \mathcal{T}^+ - \mathcal{T}^-$;
4: $\mathcal{T}^s \leftarrow$ Sim-recom($\mathcal{T}^-, \mathcal{T}$);
5: $\mathcal{T}^a \leftarrow$ AR-recom($\mathcal{T}^-, \mathcal{T}$);
6: $\mathcal{A} \leftarrow \mathcal{T}^- \cup \mathcal{T}^s \cup \mathcal{T}^a$.
```

Step 1: Base extraction

Step 2: Recommendation

- Algorithm AER, short for Aspect Extraction based on Recommendation.

Step 1: Base Extraction

- Use the DP method (DPextract) to extract an initial (or base) set T^- of aspects employing a set R^- of high precision rules.
  - Set $T^-$of extracted aspects has very high precision but low recall.
- Extract a set T^+ of aspects from a larger set R^+ of high recall rules also using DPextract.
  - Set $T^+$of extracted aspects has very high recall but low precision.

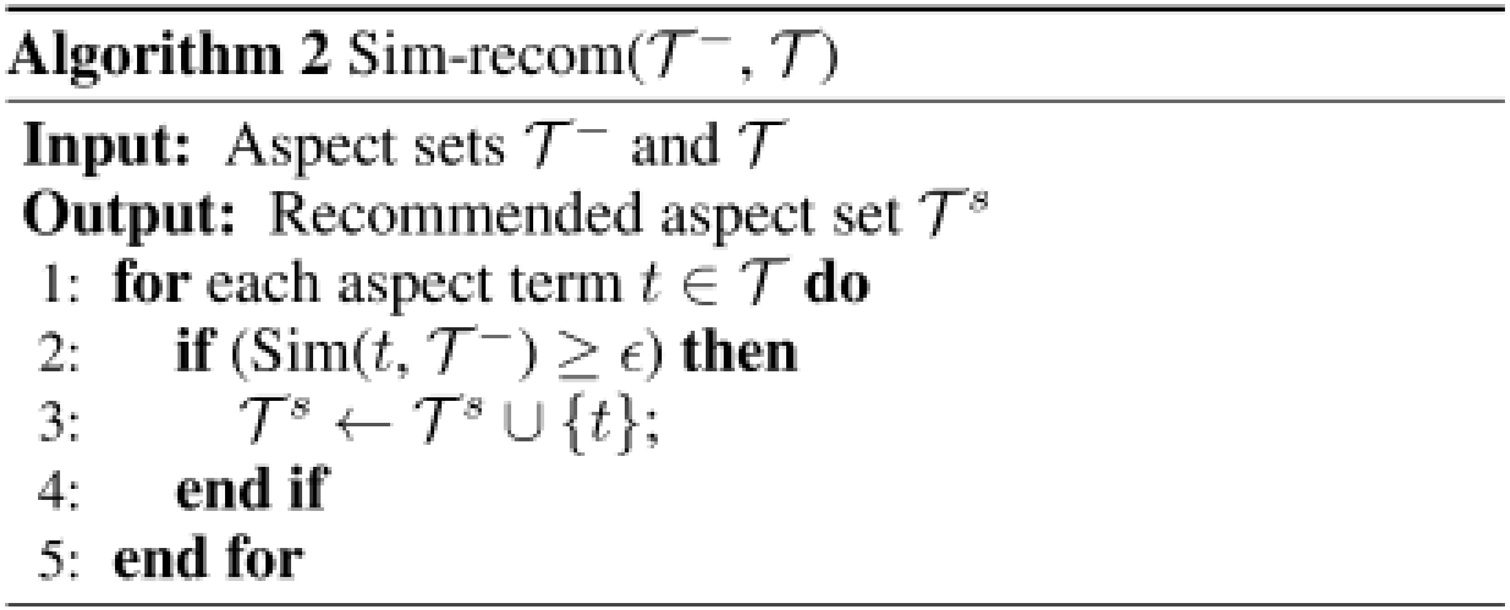
Step 2: Recommendation

- Recommend more aspects using T^{-} as the base to improve the recall. To ensure recommendation quality, AER requires
  - Aspects must be from $T = T^{+} - T^{-}$.
- Two forms of recommendation
  - similarity-based (Sim-recom) and
  - association-based (AR-recom).

Similarity-based Recommendation

- Solve the problem of missing synonymous aspects.
  - e.g., we can recommend “photo” and “image” through “picture” as they are similar in meaning.
- Employ word vectors trained from a large corpus of 5.8 million reviews for similarity comparison.
  - But can also be trained using past data

Algorithm Sim-recom



Algorithm 2 Sim-recom($\mathcal{T}^-$, $\mathcal{T}$)

```
Input: Aspect sets $\mathcal{T}^-$ and $\mathcal{T}$
Output: Recommended aspect set $\mathcal{T}^s$
1: for each aspect term $t \in \mathcal{T}$ do
2:     if (Sim($t$, $\mathcal{T}^-$) $\geq \epsilon$) then
3:         $\mathcal{T}^s \leftarrow \mathcal{T}^s \cup \{t\}$;
4:     end if
5: end for
```

- For each term $t \in T$, if the similarity between t and any term in T^- is at least ϵ, then recommend t as an aspect

Association-based Recommendation

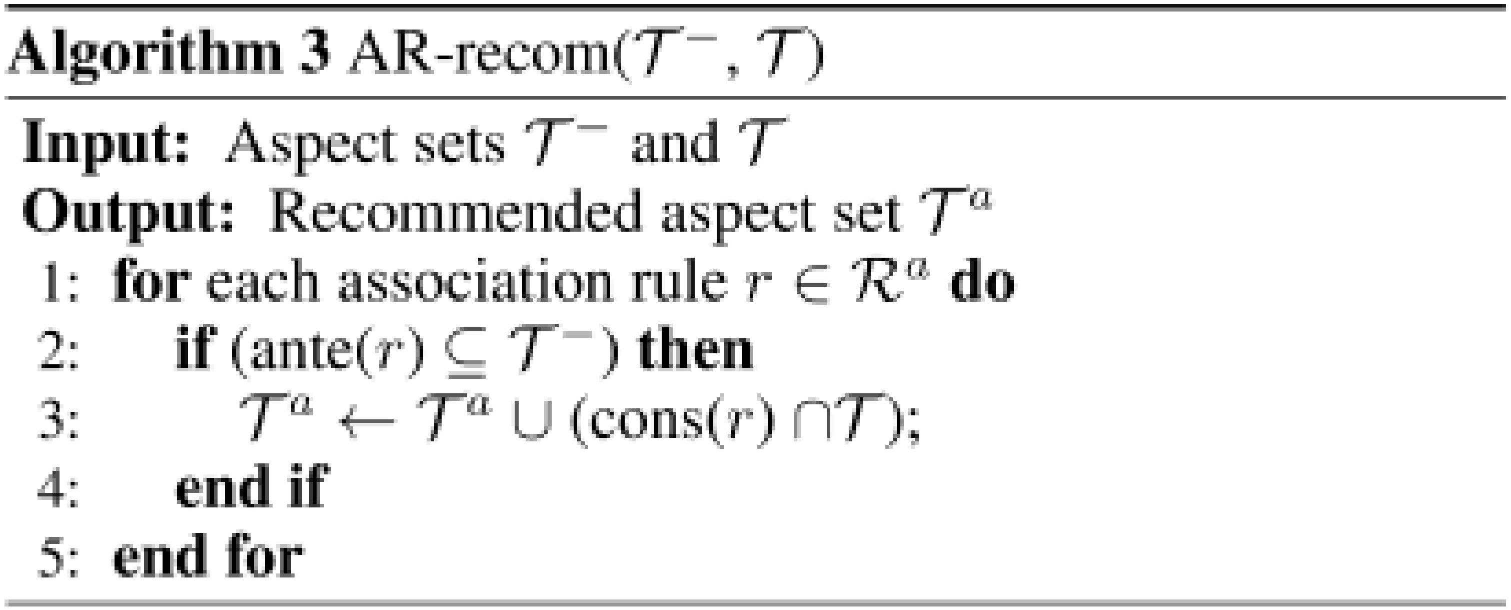
- It aims to solve the problem of missing correlated or co-occurring aspects.
  - e.g., we can recommend "battery" through "picture" as they are highly related -- pictures are taken by digital devices which need batteries.
- To mine aspect associations,
  - apply association rule mining to aspects extracted from reviews of previous products/domains.

Association Rule Generation

- The set of aspects extracted from each domain in the past forms a transaction in *DB*.
- Apply an association rule mining algorithm to *DB* to generate a set of rules.
  - An association rule in could be:

- antecedent $\boxed{\text{picture, display}} \rightarrow \boxed{\text{video, purchase}}$ consequent

Algorithm AR-recom



Algorithm 3 AR-recom($\mathcal{T}^-$, $\mathcal{T}$)

```
Input:  Aspect sets $\mathcal{T}^-$ and $\mathcal{T}$
Output: Recommended aspect set $\mathcal{T}^a$
1: for each association rule $r \in \mathcal{R}^a$ do
2:     if (ante($r$) $\subseteq \mathcal{T}^-$) then
3:         $\mathcal{T}^a \leftarrow \mathcal{T}^a \cup$ (cons($r$) $\cap \mathcal{T}$);
4:     end if
5: end for
```

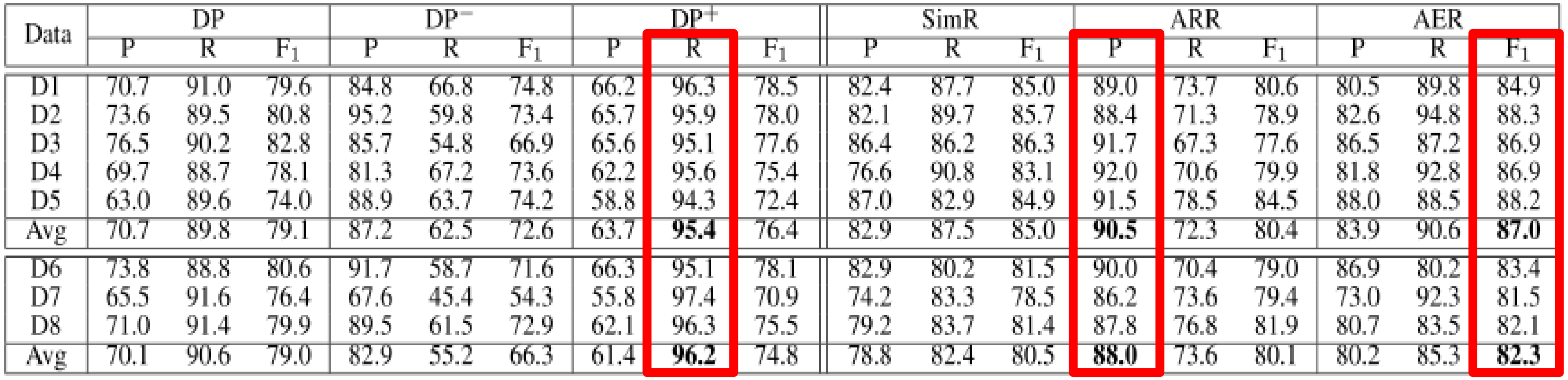
- For each association rule *r* ∈ R^a,
  - if ante(*r*) is a subset of T^-, then recommend the terms in cons(r) ∩ T as aspects.

Evaluation

- Compared Approaches
  - SimR uses only aspect similarities for recommendation.
  - ARR uses only aspect associations for recommendation.
  - AER uses both aspect similarities and associations for recommendation.

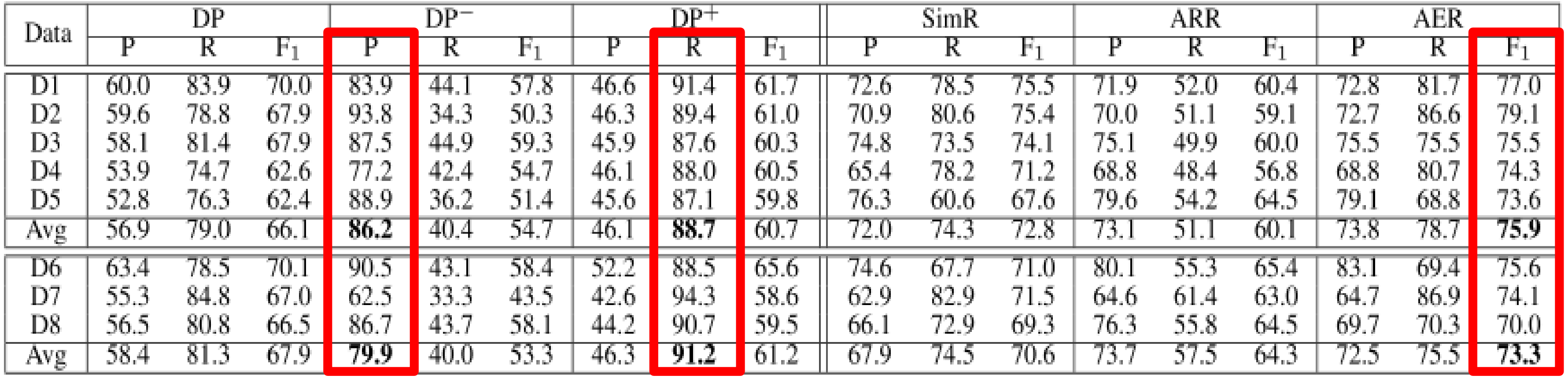
Experimental Results (Overall results)

Table 2: Precision, Recall and F$_1$-score of DP, DP$^-$, DP$^+$, SimR, ARR and AER evaluated based on multiple aspect term occurrences.



Data	DP			DP$^-$			DP$^+$			SimR			ARR			AER		
	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$
D1	70.7	91.0	79.6	84.8	66.8	74.8	66.2	96.3	78.5	82.4	87.7	85.0	89.0	73.7	80.6	80.5	89.8	84.9
D2	73.6	89.5	80.8	95.2	59.8	73.4	65.7	95.9	78.0	82.1	89.7	85.7	88.4	71.3	78.9	82.6	94.8	88.3
D3	76.5	90.2	82.8	85.7	54.8	66.9	65.6	95.1	77.6	86.4	86.2	86.3	91.7	67.3	77.6	86.5	87.2	86.9
D4	69.7	88.7	78.1	81.3	67.2	73.6	62.2	95.6	75.4	76.6	90.8	83.1	92.0	70.6	79.9	81.8	92.8	86.9
D5	63.0	89.6	74.0	88.9	63.7	74.2	58.8	94.3	72.4	87.0	82.9	84.9	91.5	78.5	84.5	88.0	88.5	88.2
Avg	70.7	89.8	79.1	87.2	62.5	72.6	63.7	**95.4**	76.4	82.9	87.5	85.0	**90.5**	72.3	80.4	83.9	90.6	**87.0**
D6	73.8	88.8	80.6	91.7	58.7	71.6	66.3	95.1	78.1	82.9	80.2	81.5	90.0	70.4	79.0	86.9	80.2	83.4
D7	65.5	91.6	76.4	67.6	45.4	54.3	55.8	97.4	70.9	74.2	83.3	78.5	86.2	73.6	79.4	73.0	92.3	81.5
D8	71.0	91.4	79.9	89.5	61.5	72.9	62.1	96.3	75.5	79.2	83.7	81.4	87.8	76.8	81.9	80.7	83.5	82.1
Avg	70.1	90.6	79.0	82.9	55.2	66.3	61.4	**96.2**	74.8	78.8	82.4	80.5	**88.0**	73.6	80.1	80.2	85.3	**82.3**

Table 3: Precision, Recall and F$_1$-score of DP, DP$^-$, DP$^+$, SimR, ARR, and AER evaluated based on distinct aspect terms.



Data	DP			DP$^-$			DP$^+$			SimR			ARR			AER		
	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$	P	R	F$_1$
D1	60.0	83.9	70.0	83.9	44.1	57.8	46.6	91.4	61.7	72.6	78.5	75.5	71.9	52.0	60.4	72.8	81.7	77.0
D2	59.6	78.8	67.9	93.8	34.3	50.3	46.3	89.4	61.0	70.9	80.6	75.4	70.0	51.1	59.1	72.7	86.6	79.1
D3	58.1	81.4	67.9	87.5	44.9	59.3	45.9	87.6	60.3	74.8	73.5	74.1	75.1	49.9	60.0	75.5	75.5	75.5
D4	53.9	74.7	62.6	77.2	42.4	54.7	46.1	88.0	60.5	65.4	78.2	71.2	68.8	48.4	56.8	68.8	80.7	74.3
D5	52.8	76.3	62.4	88.9	36.2	51.4	45.6	87.1	59.8	76.3	60.6	67.6	79.6	54.2	64.5	79.1	68.8	73.6
Avg	56.9	79.0	66.1	**86.2**	40.4	54.7	46.1	**88.7**	60.7	72.0	74.3	72.8	73.1	51.1	60.1	73.8	78.7	**75.9**
D6	63.4	78.5	70.1	90.5	43.1	58.4	52.2	88.5	65.6	74.6	67.7	71.0	80.1	55.3	65.4	83.1	69.4	75.6
D7	55.3	84.8	67.0	62.5	33.3	43.5	42.6	94.3	58.6	62.9	82.9	71.5	64.6	61.4	63.0	64.7	86.9	74.1
D8	56.5	80.8	66.5	86.7	43.7	58.1	44.2	90.7	59.5	66.1	72.9	69.3	76.3	55.8	64.5	69.7	70.3	70.0
Avg	58.4	81.3	67.9	**79.9**	40.0	53.3	46.3	**91.2**	61.2	67.9	74.5	70.6	73.7	57.5	64.3	72.5	75.5	**73.3**

LML Components of AER

- **KB**
  - Word vectors
  - Aspects extracted from previous tasks
  - Learned association rules
- **KBL**
  - DP + Two forms of recommendations

Lifelong Relaxation Labeling
(Shu et al., 2016)

- Relaxation Labeling (RL) is an unsupervised graph-based label propagation algorithm.
  - Unsupervised classification

- It is augmented with lifelong learning (*Lifelong-RL*) to exploit past knowledge learned from previous tasks.

Relaxation Labeling (RL)

- Graph consists of nodes and edges.
  - Node: object to be labeled
  - Edge: a binary relationship between two nodes.
- Each node n_i in the graph is associated with a multinomial distribution $P(L(n_i))$
  - $L(n_i)$ is the label of n_i on a label set Y .
- Each edge has two conditional distributions:
  - $P(L(n_i) \mid L(n_j))$ and $P(L(n_j) \mid L(n_i))$

Relaxation Labeling (contd)

- Neighbors *Ne*(n_i) of a node n_i are associated with a weight distribution w(n_j | n_i)

$$\sum_{n_j \in Ne(n_i)} w(n_j | n_i) = 1.$$

- RL iteratively updates the label distribution of each node until convergence.
- Initially, we have $P^0(L(n_i))$. Let $\Delta P^{r+1}(L(n_i))$ be the change of $P(L(n_i))$ at iteration r + 1.

$$\Delta P^{r+1}(L(n_i)) = \sum_{n_j \in Ne(n_i)} \left(w(n_j|n_i) \times \sum_{y \in Y} P(L(n_i)|L(n_j) = y) \times P^r(L(n_j) = y) \right)$$

Relaxation Labeling (contd)

- Updated label distribution for iteration *r* + 1 is computed as follows:

$$P^{r+1}(L(n_i)) = \frac{P^r(L(n_i)) \times (1 + \Delta P^{r+1}(L(n_i)))}{\sum_{y \in Y} P^r(L(n_i) = y) \times (1 + \Delta P^{r+1}(L(n_i) = y))}$$

- The final label of node n_i is its highest probable label.

$$L(n_i) = \underset{y \in Y}{\operatorname{argmax}}(P(L(n_i) = y))$$

What past knowledge can be used?

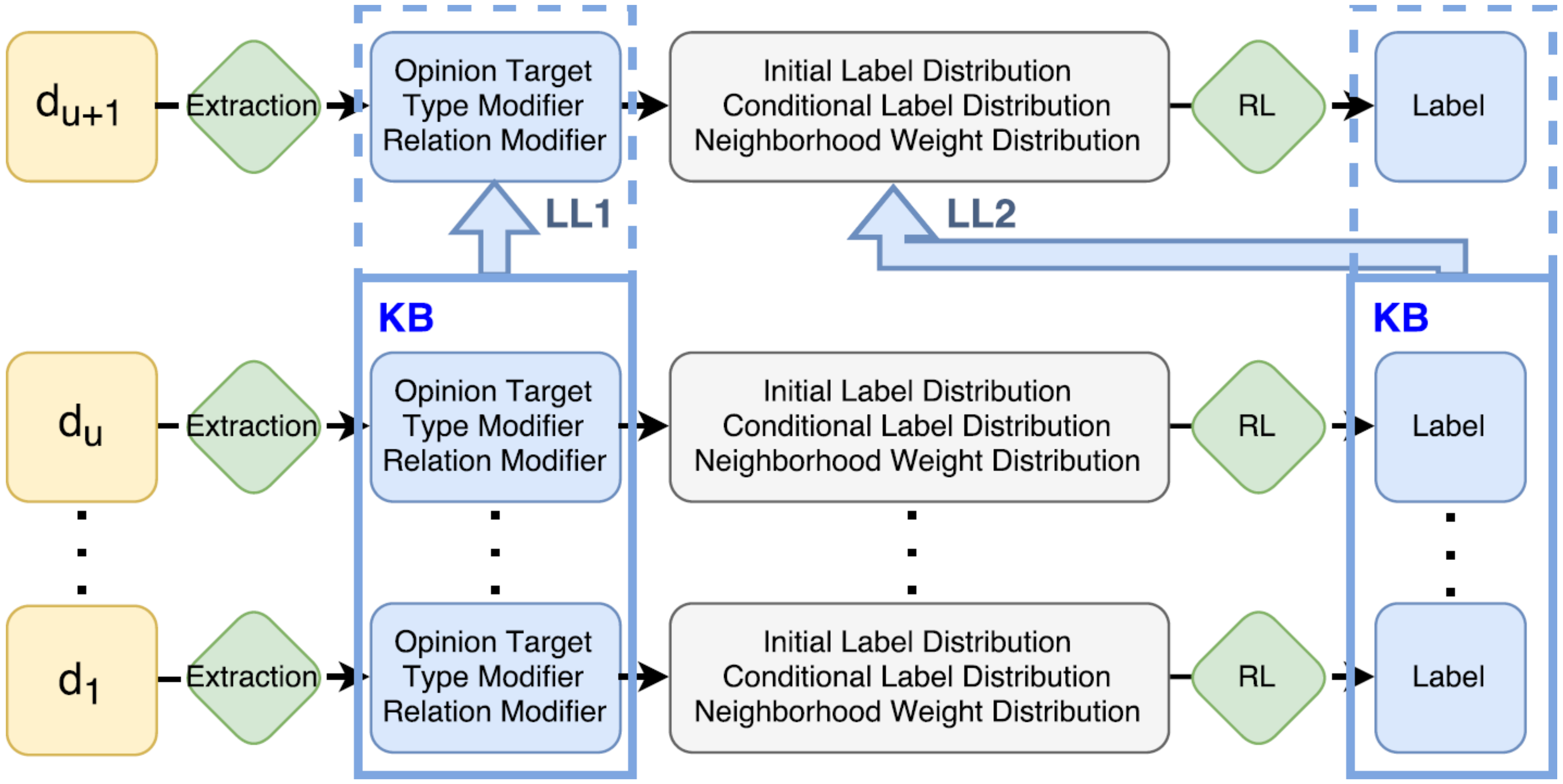
Lifelong-RL uses two forms of knowledge

- Prior edges: graphs are usually not given or fixed but are built based on text data.
  - If the data is small, many edges may be missing
  - But such edges may existing in the graphs of some previous tasks
- Prior labels: initial $P^0(L(n_i))$ is quite hard to set, but results from previous tasks can be used to set it more accurately.

Lifelong-RL for a SA task
(Shu et al., 2016)

- Problem: opinion target labeling
  - Separating entities and aspects
  - Example: "Although the engine is slightly weak, this car is great."
    - Entity: car; Aspect: engine
  - Target extract often cannot distinguish the two
- Suitable for lifelong learning
  - Shared edges, and shared entities and aspects and their labels across domains

Lifelong-RL architecture



d_{u+1}
Extraction
Opinion Target Type Modifier Relation Modifier
Initial Label Distribution Conditional Label Distribution Neighborhood Weight Distribution
RL
Label
LL1
LL2
KB
d_u
Extraction
Opinion Target Type Modifier Relation Modifier
Initial Label Distribution Conditional Label Distribution Neighborhood Weight Distribution
RL
Label
KB
d_1
Extraction
Opinion Target Type Modifier Relation Modifier
Initial Label Distribution Conditional Label Distribution Neighborhood Weight Distribution
RL
Label

- Relation modifiers indicate edges.
- Type modifiers and prior labels help set $P^0(L(n_i))$

LML Components of Lifelong-RL

- **KB**
  - Edges from previous tasks
  - Node labels from previous tasks
- **KBL**
  - Relaxation labeling

Outline

Never Ending Language Learner
(Carlson et al., 2010; Mitchell et al., 2015)

- NELL: Never Ending Language Learner
- Perhaps the only live LML system
  - it has been reading the Web to extract certain types of information (or knowledge)
  - 24/7 since January 2010.
- NELL has accumulated millions of facts with attached confidence weights
  - called beliefs,

Input to NELL

- An ontology defining a set of target categories and relations to be learned,
  - a handful of seed training examples for each, and
  - a set of coupling constraints about categories and relations (Person & Sport are mutually exclusive).
- Webpages crawled from the Web
- Interactions with human trainers to correct some mistakes made by NELL

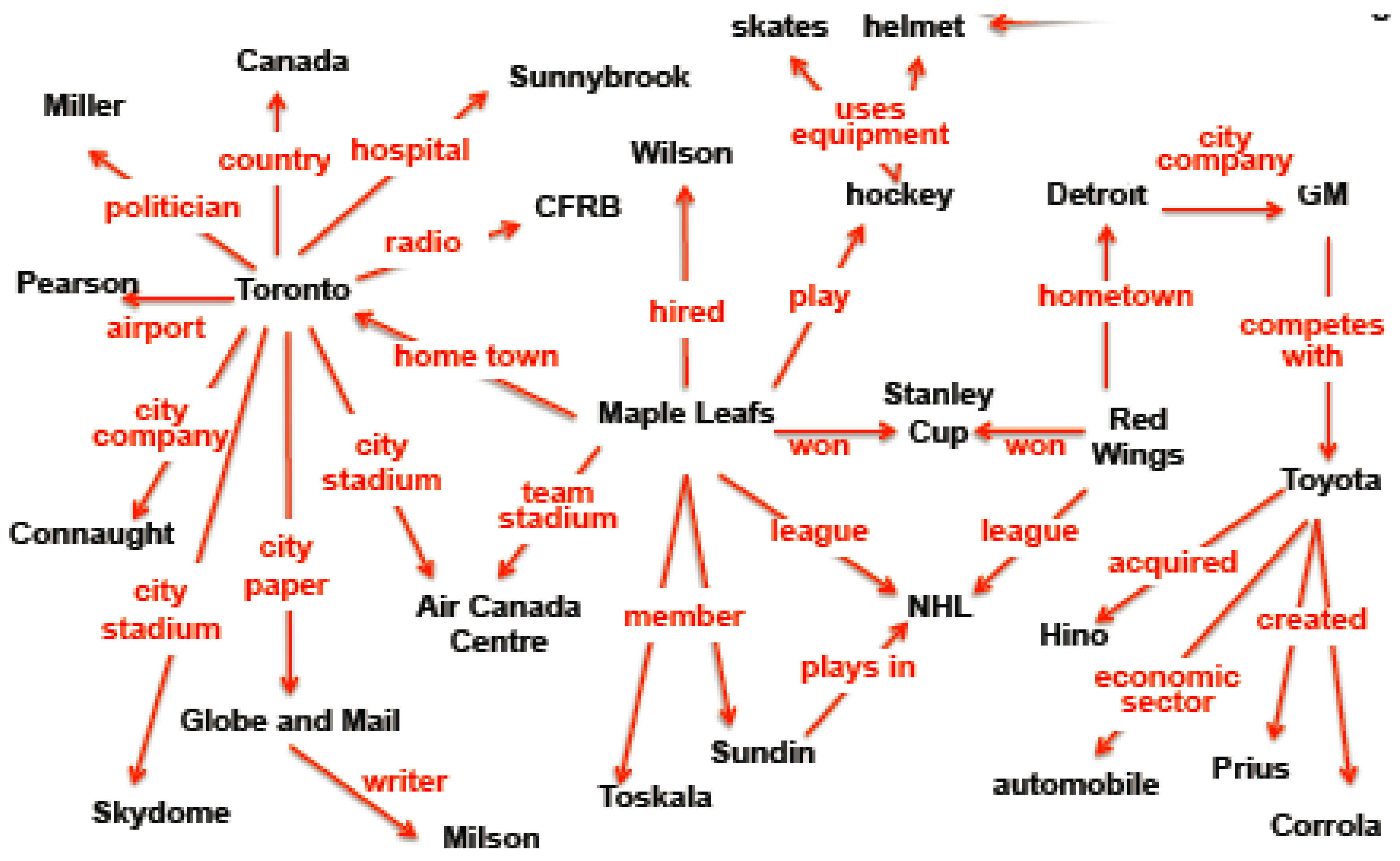
Goal of NELL

- Reading - extract facts from webpages to populate the initial ontology
  - *category* of a noun or noun phrase, e.g., Los Angeles is a *city*
  - *relations* of a pair of noun phrases
    - hasMajor(Stanford, Computer Science)
- Learn to perform the above extraction tasks better each day.

Knowledge Base

- Instance of category: which noun phrases refer to which specified semantic categories
  - For example, *Los Angeles* is in the category *city*.
- Relationship of a pair of noun phrases, e.g., given a name of an organization and the location, check if
  - *hasOfficesIn(<organization>, <location>)*.
- …

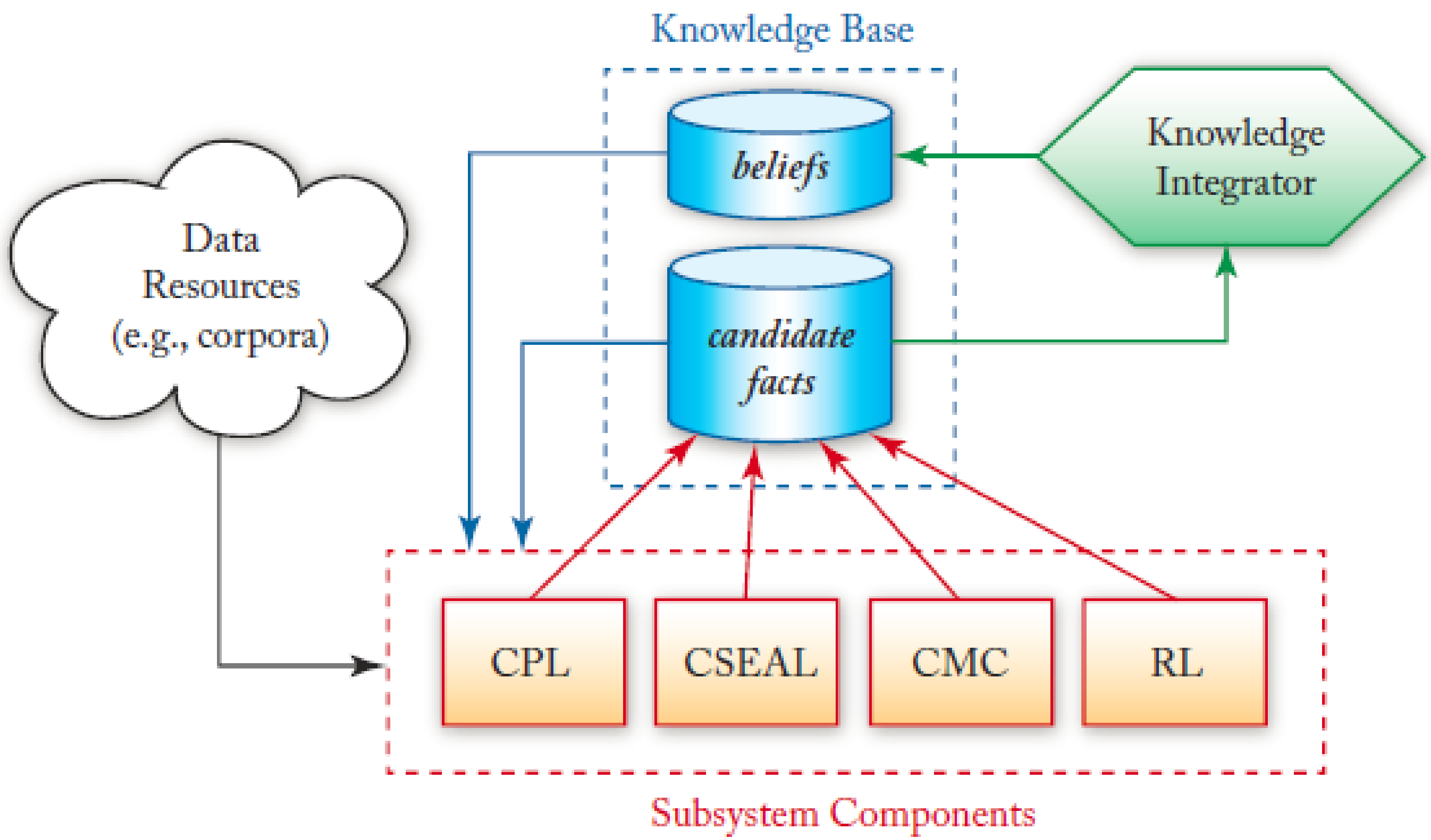
NELL Knowledge Fragment



Semi-supervised Learning

- Training examples
  - human-labeled instances in NELL's ontology
  - labeled examples contributed over time through NELL's crowdsourcing website,
  - a set of NELL self-labeled training examples corresponding to NELL's current knowledge base,
  - a large amount of unlabeled Web text.
- 2nd and 3rd sets of the training examples propel NELL's lifelong learning

NELL Architecture



Coupled Pattern Learner (CPL)

- CPL: extractors extracting both category and relation instances using contextual patterns.
  - Examples
    - Category pattern: “mayor of X” and
    - Relation pattern: “X plays for Y”

- Such patterns can also be learned.
- Mutual exclusion & type-checking constraints
  - filter candidate facts to ensure quality

Coupled SEAL (CSEAL)

- CSEAL: an extraction and learning system that extracts facts from semi-structured webpages using wrapper induction
- Based on set expansion or PU learning
  - Wrapper: html strings specifying the left and right context of an entity.
- Mutual exclusion & type-checking constraints:
  - filtered out likely errors

Coupled Morphological Classifier (CMC)

- CMC: a set of binary classifiers, one for each category,
  - To classify whether the extracted candidate facts/beliefs by other subsystems are indeed of their respective categories.
- Positive training examples:
  - beliefs in the current knowledge base.
- Negative training examples
  - beliefs satisfying mutual exclusion constraints

Rule Learner (RL)

- Its goal is to learn probabilistic Horn clauses
  - to use them to infer new relations from the existing relations in the knowledge base.
- Reasoning capability
  - represents an important advance of NELL
  - It does not exist in most current LML systems.

Coupling Constraints in NELL

- *Multi-view co-training coupling constraint*
  - Agreement: the same category or relation learned from different data sources, or *views*.
- *Subset/superset coupling constraint*
  - When a new category is added to NELL's ontology, its parents (supersets) are also specified.
- *Horn clause coupling constraint*
  - E.g., "X living in Chicago" and "Chicago being a city in U.S." $\rightarrow$ "X lives in U.S."

LML Components of NELL

- **KB**
  - Extracted facts and relations
  - Reasoning capability
- **KBL**
  - All the learners and extractors

ALICE: Lifelong Info. Extraction
(Banko and Etzioni 2007)

- Similar to NELL, Alice performs similar continuous/lifelong information extraction of
  - concepts and their instances,
  - attributes of concepts, and
  - various relationships among them.
  - The knowledge is iteratively updated

- Extraction based on syntactic patterns like
  - (<x> *such as* <y>) and (*fruit such as* <y>),

Lifelong Strategy

- The output knowledge upon completion of a learning task is used in two ways:
    - to update the current domain theory (i.e., domain concept hierarchy and abstraction) and
    - to generate subsequent learning tasks.
- This behavior makes Alice a lifelong agent
    - i.e., Alice uses the knowledge acquired during the *nth* task to specify its future learning agenda.

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Reinforcement Learning

- An agent learns actions through trial and error interactions with a dynamic environment
- The agent gets reward/penalty after each action
- Each action changes the state of the environment
- The agent usually needs a large amount of quality experience (cost is high)

Lifelong Reinforcement Learning (LRL)

- Utilize the experience accumulated from other tasks
- Learn faster in a new task with fewer interactions

Example LRL Works

- Lifelong robot learning with knowledge memorization (Thrun and Mitchell 1995)
- Treating each environment as a task (Tanaka and Yamamura 1997)
- Hierarchical Bayesian approach (Wilson et al., 2007)
- Policy Gradient Efficient Lifelong Learning Algorithm (PG-ELLA) (Bou Ammar et al., 2014)

Outline

- A motivating example
- What is lifelong machine learning?
- Related learning paradigms
- Lifelong supervised learning
- Lifelong unsupervised learning
- Semi-supervised never-ending learning
- Lifelong reinforcement learning
- Summary

Summary

- This tutorial gave an introduction to LML with a focus on NLP applications
- Existing LML research is still in its infancy
  - Understanding of LML is very limited
  - Current research mainly focuses on
    - Only one type of tasks in a system
- LML needs big data – to learn a large amount of reliable knowledge of different types.
  - The more we know the better we can learn

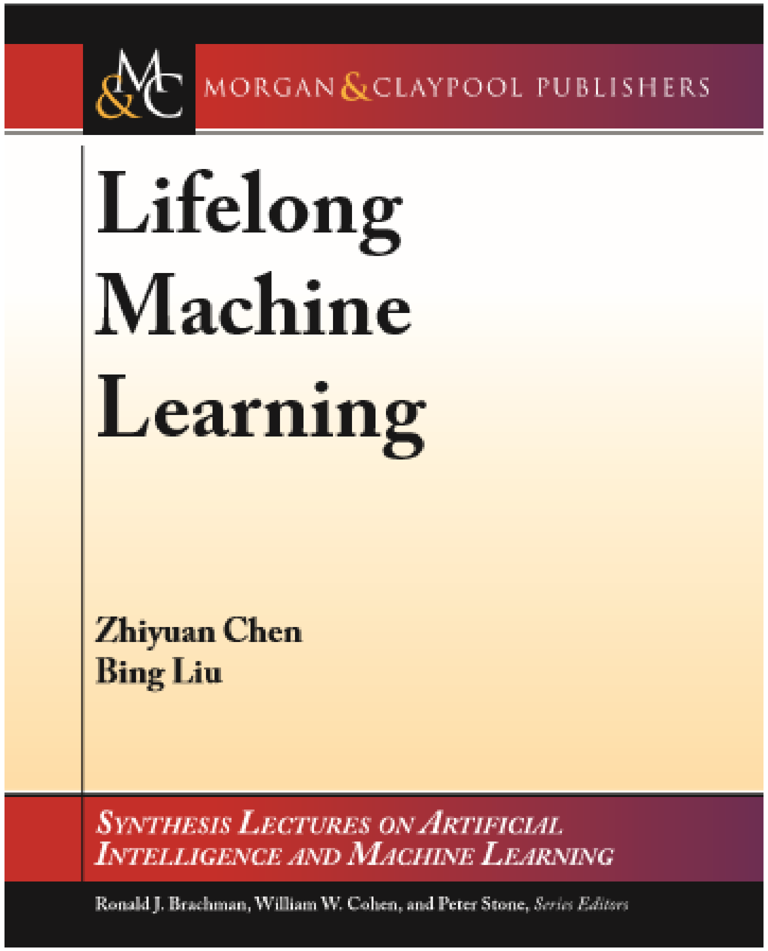
Summary
(Chen and Liu 2016-book)

There are many challenges for LML, e.g.,

- Correctness of knowledge
- Applicability of knowledge
- Knowledge representation and reasoning
- Learn with tasks of multiple types
- Self-motivated learning
- Compositional learning
- Learning in interaction with humans & systems

Coming Soon (Nov 2016)
(Chen and Liu 2016-book)

- Introduction
- Related Learning Paradigms
- Lifelong Supervised Learning
- Lifelong Unsupervised Learning
- Lifelong Semi-supervised Learning for Information Extraction
- Lifelong Reinforcement Learning
- Conclusion


MORGAN & CLAYPOOL PUBLISHERS

Lifelong Machine Learning

Zhiyuan Chen
Bing Liu

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Ronald J. Brachman, William W. Cohen, and Peter Stone, *Series Editors*

Thank You!

Q & A

Reference (1)

Ana Paula Appel and Estevam Rafael Hruschka Junior. 2011. Prophet--a link-predictor to learn new rules on nell. In Proceedings of 2011 IEEE 11th International Conference on Data Mining Workshops, pages 917–924.

G. H. Bakhir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. 2007. Predicting Structured Data. Cambridge, MA, USA: MIT Press.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research, 3, 993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In Proceedings of ACL, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In Proceedings of EMNLP, pages 120–128.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In Proceedings of COLT, pages 92–100.

Reference (2)

Haitham Bou Ammar, Rasul Tutunov, and Eric Eaton. 2015. Safe policy search for lifelong reinforcement learning with sublinear regret. In Proceedings of ICML.

Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2009. Coupling Semi-Supervised Learning of Categories and Relations. In Proceedings of Proc. of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing.

Rich Caruana. 1997. Multitask Learning. Machine learning, 28(1), 41–75.

Zhiyuan Chen and Bing Liu. 2014a. Topic Modeling using Topics from Many Domains, Lifelong Learning and Big Data. In Proceedings of ICML, pages 703–711.

Zhiyuan Chen and Bing Liu. 2014b. Mining Topics in Documents : Standing on the Shoulders of Big Data. In Proceedings of KDD, pages 1116–1125.

Zhiyuan Chen, Nianzu Ma, and Bing Liu. 2015. Lifelong Learning for Sentiment Classification. In Proceedings of ACL, pages 750–756.

Zhiyuan Chen and Bing Liu. 2016. Lifelong Machine Learning. Morgan & Claypool Publishers.

Reference (3)

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect Extraction with Automated Prior Knowledge Learning. In Proceedings of ACL, pages 347–358.

Sanjoy Dasgupta, Michael L. Littman, and David McAllester. 2001. PAC generalization bounds for co-training. Advances in neural information processing systems, 1, 375–382.

Hal Daumé III. 2008. Bayesian multitask learning with latent hierarchies. In Proceedings of Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pages 135–142.

Geli Fei, Shuai Wang, and Bing Liu. 2016. Learning Cumulatively to Become More Knowledgeable. In Proceedings of KDD.

Kuzman Ganchev, João V Graça, John Blitzer, and Ben Taskar. 2008. Multi-view learning over structured and non-identical outputs. In Proceedings of In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI.

Jayant Krishnamurthy and Tom M. Mitchell. 2011. Which Noun Phrases Denote Which Concepts. In Proceedings of Proceedings of the Forty Ninth Annual Meeting of the Association for Computational Linguistics.

Reference (4)

Abhishek Kumar, Hal Daum, and Hal Daume Iii. 2012. Learning Task Grouping and Overlap in Multi-task Learning. In Proceedings of ICML, pages 1383–1390.

Bing Liu. 2015. Sentiment Analysis Mining Opinions, Sentiments, and Emotions. Cambridge University Press.

Bing Liu. 2012. Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 1–167.

Qian Liu, Bing Liu, Yuanlin Zhang, Doo Soon Kim, and Zhiqiang Gao. 2016. Improving Opinion Aspect Extraction using Semantic Similarity and Aspect Associations. In Proceedings of AAAI.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In Proceedings of NAACL.

Y. Lashkari M. Metral and Pattie Maes. 1998. Collaborative interface agents. Readings in agents, 111.

Ryszard S. Michalski. 1993. Learning= inferencing+ memorizing. Foundations of Knowledge Acquisition, pages 1–41. Springer.

Reference (5)

Thahir Mohamed, Estevam Hruschka Jr., and Tom Mitchell. 2011. Discovering Relations between Noun Categories. In Proceedings of Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1447–1455. Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Guillaume Obozinski, Ben Taskar, and Michael I. Jordan. 2010. Joint Covariate Selection and Joint Subspace Selection for Multiple Classification Problems. Statistics and Computing, 20(2), 231–252.

Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. IEEE Trans. Knowl. Data Eng., 22(10), 1345–1359.

Saulo D. S. Pedro and Estevam R. Hruschka Jr. 2012. Collective intelligence as a source for machine learning self-supervision. In Proceedings of Proc. of the 4th International Workshop on Web Intelligence and Communities, pages 5:1–5:9. NY, USA: ACM.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion Word Expansion and Target Extraction through Double Propagation. Computational Linguistics, 37(1), 9–27.

Reference (6)

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, pages 117–120.

Paul Ruvolo and Eric Eaton. 2013. ELLA: An efficient lifelong learning algorithm. In Proceedings of ICML, pages 507–515.

Paul Ruvolo and Eric Eaton. 2013. Active Task Selection for Lifelong Machine Learning. In Proceedings of AAAI, pages 862–868.

Daniel L. Silver and Robert Mercer. 1996. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. Connection Science, 8(2), 277–294.

Daniel L. Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In Proceedings of AAAI Spring Symposium: Lifelong Machine Learning, pages 49–55.

Reference (7)

Lei Shu, Bing Liu, Hu Xu, and Annice Kim. 2016. Separating Entities and Aspects in Opinion Targets using Lifelong Graph Labeling. In Proceedings of EMNLP.

Ray J. Solomonoff. 1989. A system for incremental learning based on algorithmic probability. In Proceedings of Proceedings of the Sixth Israeli Conference on Artificial Intelligence, Computer Vision and Pattern Recognition, pages 515–527.

Karthik Sridharan and Sham M. Kakade. 2008. An Information Theoretic Framework for Multi-view Learning. In Proceedings of COLT, pages 403–414.

Fumihide Tanaka and Masayuki Yamamura. 1997. An approach to lifelong reinforcement learning through multiple environments. In Proceedings of 6th European Workshop on Learning Robots, pages 93–99.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10(Jul), pages 1633–1685, 2009

Sebastian. Thrun. 1996. Explanation-Based Neural Network Learning: A Lifelong Learning Approach. Kluwer Academic Publishers.

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In Proceedings of NIPS, pages 640–646.

Reference (8)

Sebastian Thrun and Joseph O’Sullivan. 1996. Discovering Structure in Multiple Learning Tasks: The TC Algorithm. In Proceedings of ICML, pages 489–497. Morgan Kaufmann.

Shuai Wang, Zhiyuan Chen, and Bing Liu. 2016. Mining Aspect-Specific Opinion using a Holistic Lifelong Topic Model. In Proceedings of WWW.

Wei Wang and Zhi-Hua Zhou. 2010. A new analysis of co-training. In Proceedings of Proceedings of the 27th international conference on machine learning (ICML-10), pages 1135–1142.