NeurIPS 2019

Irene

27,Jan, 2020

Outlines

Interesting Stats: about conference

NLP Ideas

Graph Neural Net Ideas

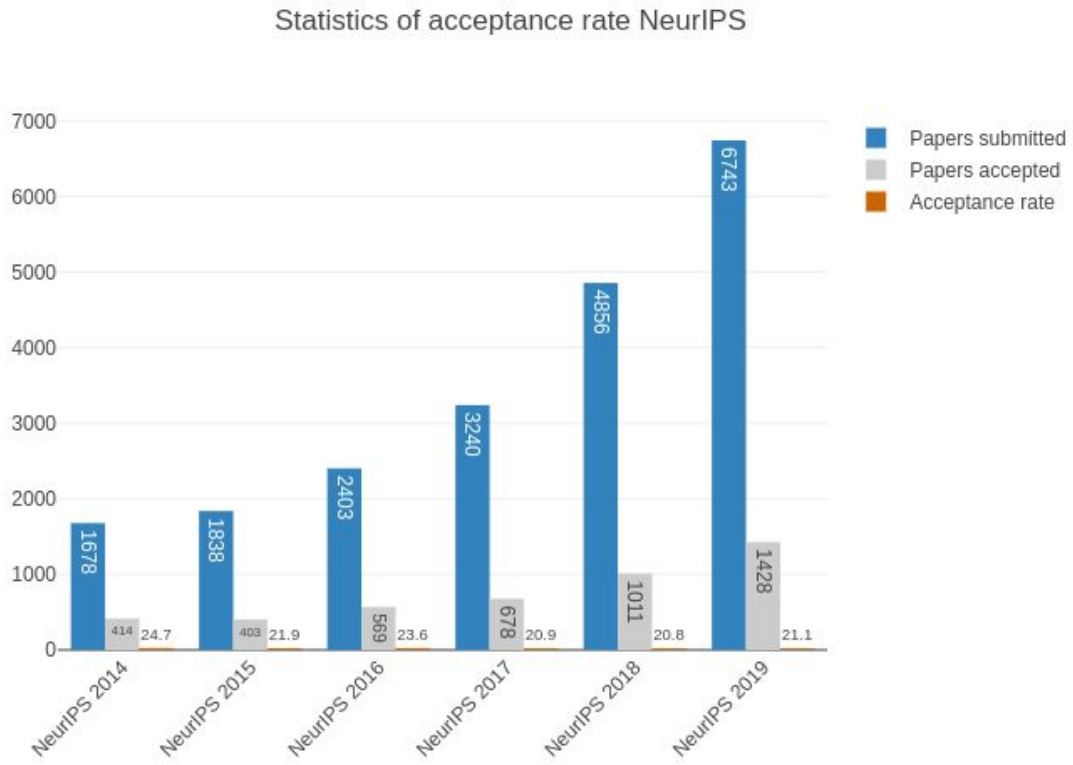
Special Track in Medical Science, Climate Change

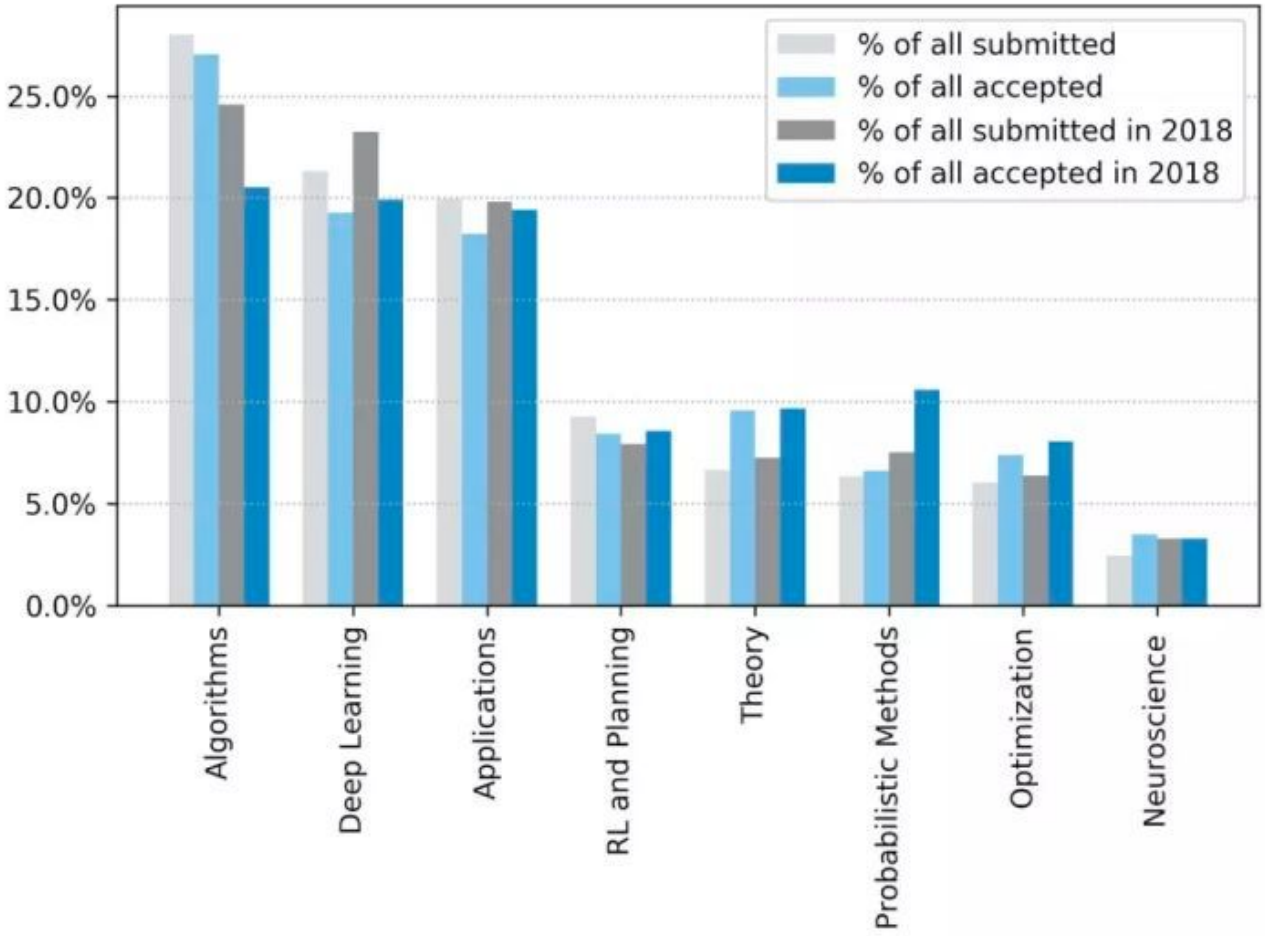
Other Fun Facts

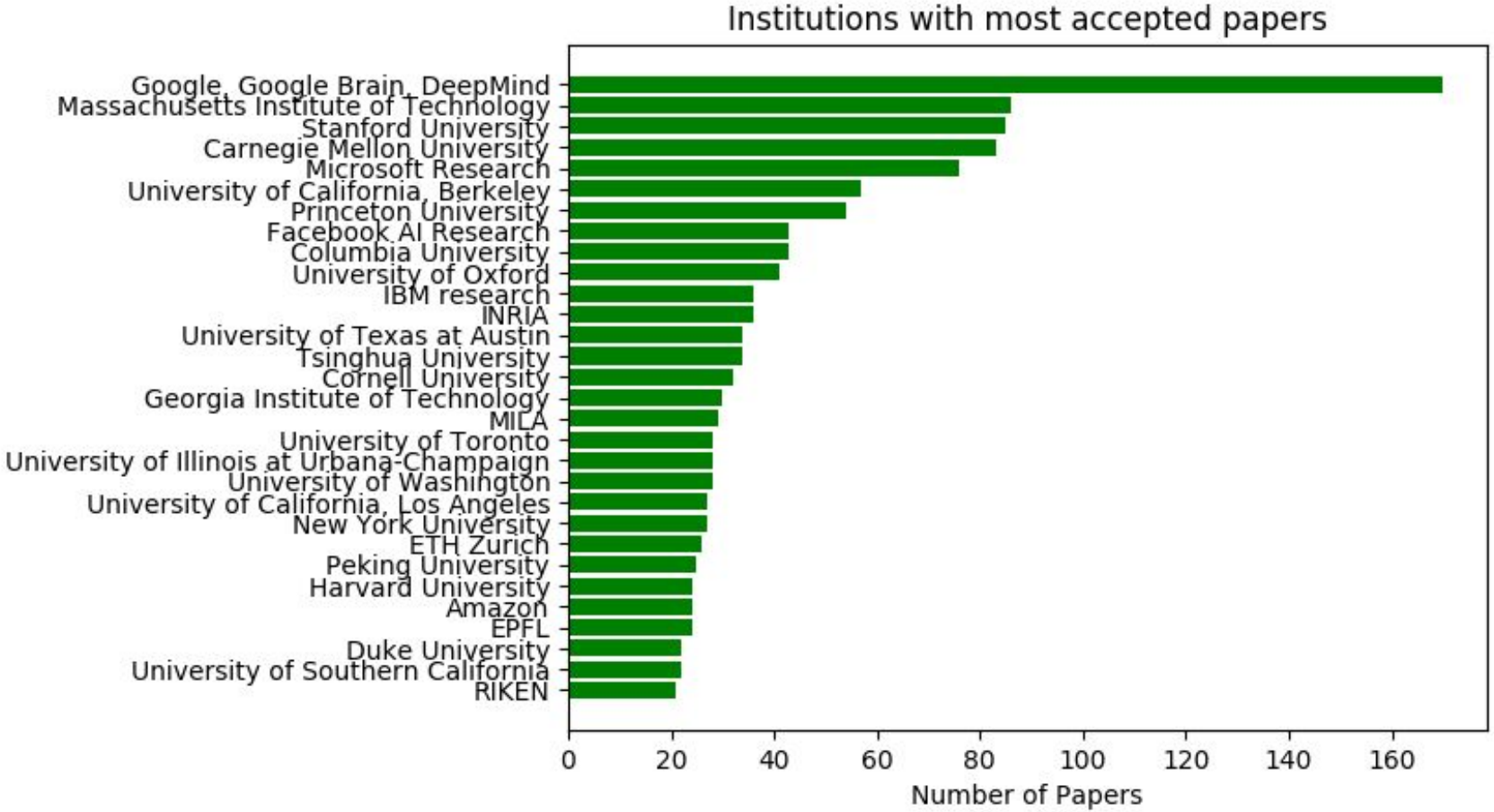
Stats

Facts about NeurIPS 2019: link

They received a record-breaking 6743 submissions this year, of which 1428 were accepted (including 36 orals and 164 spotlights)



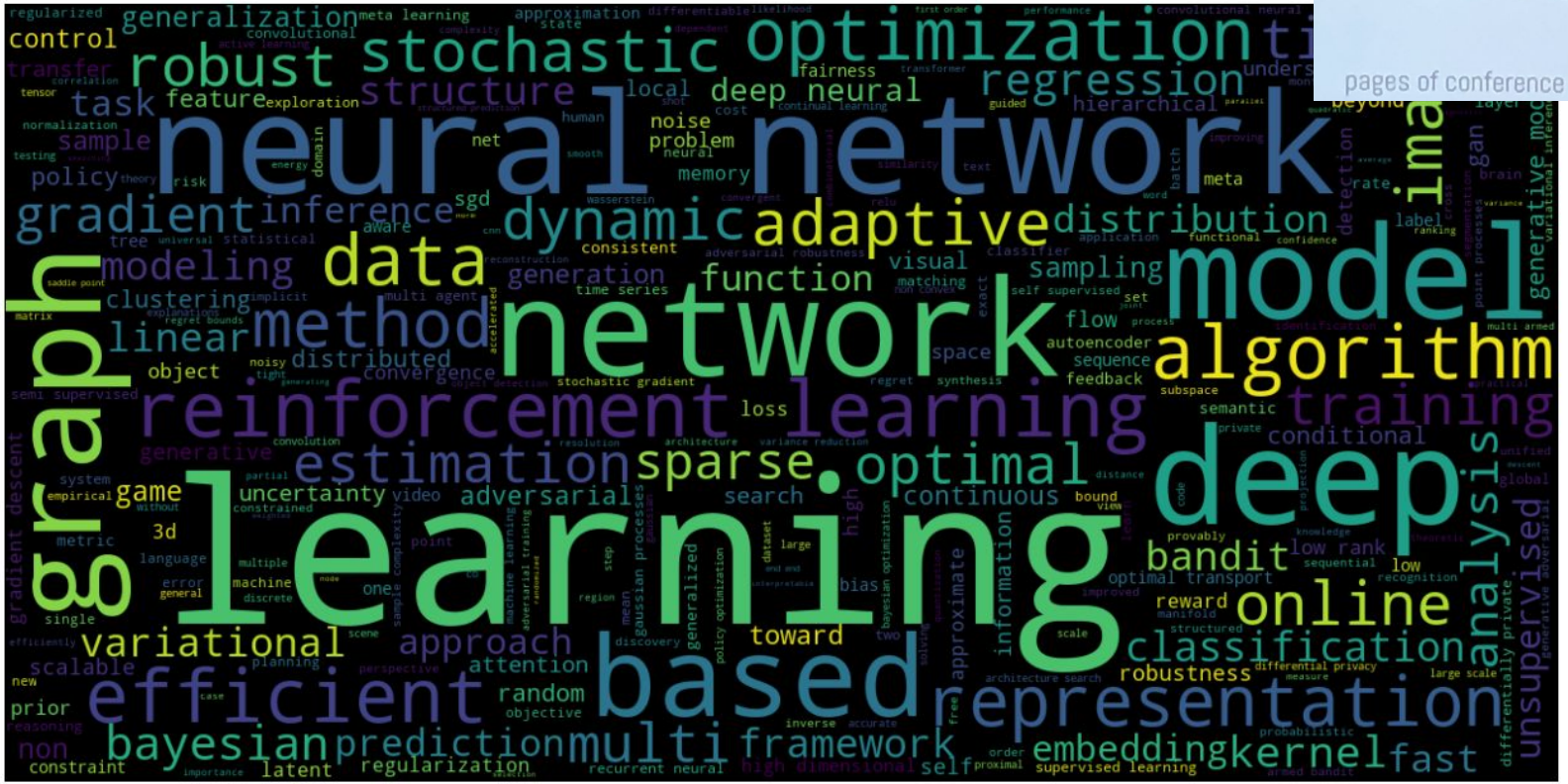

% of all submitted
% of all accepted
% of all submitted in 2018
% of all accepted in 2018
25.0%
20.0%
15.0%
10.0%
5.0%
0.0%
Algorithms
Deep Learning
Applications
RL and Planning
Theory
Probabilistic Methods
Optimization
Neuroscience


Institutions with most accepted papers
Google, Google Brain, DeepMind
Massachusetts Institute of Technology
Stanford University
Carnegie Mellon University
Microsoft Research
University of California, Berkeley
Princeton University
Facebook AI Research
Columbia University
University of Oxford
IBM research
INRIA
University of Texas at Austin
Tsinghua University
Cornell University
Georgia Institute of Technology
MILA
University of Toronto
University of Illinois at Urbana-Champaign
University of Washington
University of California, Los Angeles
New York University
ETH Zurich
Peking University
Harvard University
Amazon
EPFL
Duke University
University of Southern California
RIKEN
0
20
40
60
80
100
120
140
160
Number of Papers

Words from accepted papers' titles

~16k

pages of conference proceedings



Outstanding New Directions Paper Award

Uniform convergence may be unable to explain generalization in deep learning

Vaishnavh Nagarajan
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA
vaishnavh@cs.cmu.edu

J. Zico Kolter
Department of Computer Science
Carnegie Mellon University &
Bosch Center for Artificial Intelligence
Pittsburgh, PA
zkolter@cs.cmu.edu

Best Paper Award More

Distribution-Independent PAC Learning of Halfspaces with Massart Noise

Ilias Diakonikolas
University of Wisconsin-Madison
ilias@cs.wisc.edu

Themis Gouleakis
Max Planck Institute for Informatics
tgouleak@mpi-inf.mpg.de

Christos Tzamos
University of Wisconsin-Madison
tzamos@wisc.edu

Test of Time Award Videos

Dual Averaging Method for Regularized Stochastic Learning and Online Optimization

Lin Xiao
Microsoft Research, Redmond, WA 98052
lin.xiao@microsoft.com

Abstract

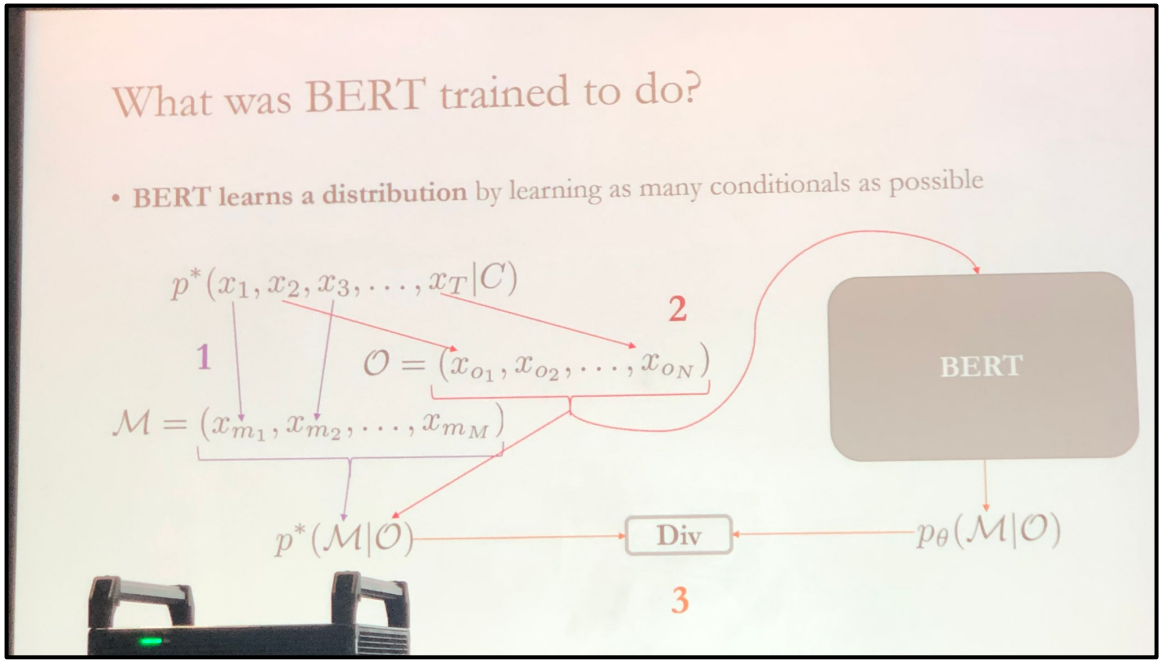
We consider regularized stochastic learning and online optimization problems, where the objective function is the sum of two convex terms: one is the loss function of the learning task, and the other is a simple regularization term such as ℓ_1-norm for promoting sparsity. We develop a new online algorithm, the *regularized dual averaging* (RDA) method, that can explicitly exploit the regularization structure in an online setting. In particular, at each iteration, the learning variables are adjusted by solving a simple optimization problem that involves the running average of all past subgradients of the loss functions and the whole regularization term, not just its subgradient. Computational experiments show that the RDA method can be very effective for sparse online learning with ℓ_1-regularization.

NLP Ideas

Tutorial: Language Models

Predict location and symbol separately;

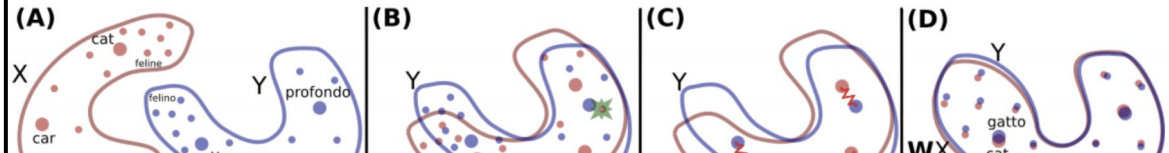
Formulating LM as a sequential decision making process



X-lingual: Comparing Unsupervised Word Translation Methods Step by Step

Cross-lingual word vector space alignment is the task of mapping the vocabularies of two languages into a shared semantic space

An evaluation on various cross-lingual word embedding methods



$$W^{\star} = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}}$$

Adversarial way to learn W, then fine-tune W.

Word translation without parallel data (2018)

Comparing Unsupervised Word Translation Methods Step by Step

- Learn word-word mapping: **distribution matching** and refinement.
- Showed that vanilla GANs are the best in precision and robustness.
- Unsupervised way: inti with a seed dictionary
    - unsupervised dictionary induction (UBDI) using GANs: learn a **linear** transformation to minimize the divergence between a target distribution (say French word embeddings) and a source distribution (the English word embeddings projected into the French space) -> instabilities in other languages.
- Learning seed (bilingual) dictionary: proposed a simple criterion based on cosine similarities between nearest neighbors in the learned alignment.

Method: GAN-initialized UBDI

- Use GANs:
  - English words E, French words F: generator learns a mapping Ω, such that ΩE is very close to F; discriminator is used to tell if a word is French or English.
  - My own trial! From word-level to sentence-level:
    - I utilized similar way into a sentence level, where I replaced the generator with a pre-trained BERT model.
    - Fails totally!
    - GANs are hard to train; BERT may be too complicated, it won't no longer be a "linear" mapping then.
    - My blog post about GANs+pytorch, another post about cross-lingual papers.

Evaluation

- Compare the mappings on Estonian (et), Farsi (fa), Finnish (fi), Latvian (lv), Turkish (tr), and Vietnamese (vi). Pre-trained FastText, MUSE dictionaries and VecMap System.

	PROCRUSTES	STOCHASTIC DICTIONARY INDUCTION	
	C-MUSE	C-MUSE	Artetxe et al. (2018)
et-en	27.5	47.6	47.6
fa-en	40.9	**41.5**	40.2
fi-en	58.9	62.5	**63.6**
lv-en	33.2	**44.1**	41.6
tr-en	60.6	**62.8**	60.6
vi-en	51.3	**54.3**	0.3
average	45.4	**52.1**	42.3

Table 3: Comparison of MUSE with cosine-based model selection over 10 random restarts (C-MUSE) with and without stochastic dictionary induction (with suggested hyper-parameters from Artetxe et al. (2018)), against state of the art. Using vanilla GANs is better than Gromov-Wasserstein on average and better on 4/6 language pairs.

AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification

Highlights:

A tree-based solution for Extreme multi-label text classification (XMTC) ;

Probabilistic label tree (PLT), which allows to handle millions of labels;

Uses BiLSTMs to capture long-distance dependency among words and a multi-label attention to capture the most relevant parts of texts to each label;

Evaluated over 6 benchmarks datasets including Amazon-3M with around 3 million labels and 2 millions samples, achieve sota scores with competitive costs on time and space.

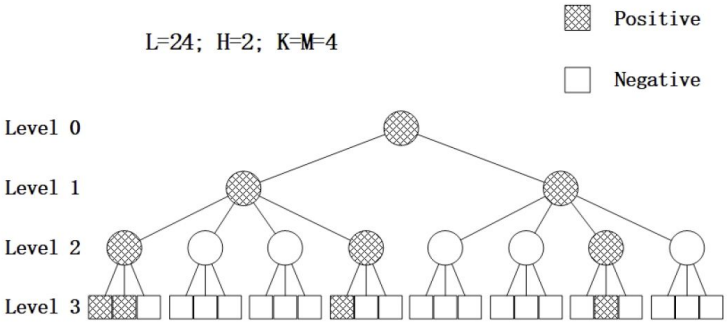
XMTC (paper link) Related Methods

- Four categories: 1-vs-All, Embedding-based, Instance or label tree-based and Deep learning-based methods.
- Recent methods:
  - XML-CNN based method, but cannot capture the most relevant parts of the input text to each label.
  - Seq2seq method: MLC2Seq, etc (recurrent neural network (RNN) to encode a given raw text and an attentive RNN as a decoder to generate predicted labels sequentially.)

AttentionXML Model: PLT

- A shallow PLT: small H, small M (applied clustering method to build the tree)
- K-means to create a tree for labels: leaf nodes are true labels.

$$P(z_n = 1|x) = \prod_{i \in Path(n)} P(z_i = 1|z_{Pa(i)} = 1, x)$$



(a)

T_0: 1 2 4 8 16 32 64 128 256 512 1024 8000
T_1: 1 2 4 8 16 32 64 128 1024 8000
T_2: 1 2 4 8 16 128 1024 8000
T_3: 1 16 128 1024 8000

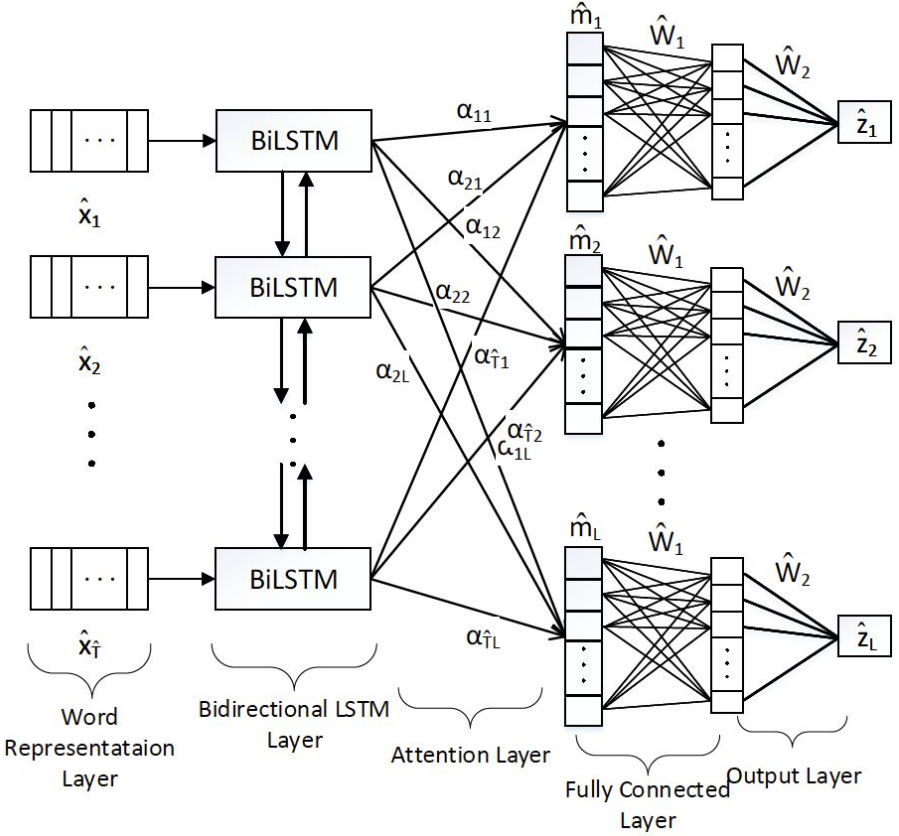
(b)

AttentionXML Model

- 300-dimensional GloVe
- Multi-Label Attention

$$\hat{\mathbf{m}}_j = \sum_{i=1}^{\hat{T}} \alpha_{ij} \hat{\mathbf{h}}_i, \qquad \alpha_{ij} = \frac{e^{\hat{\mathbf{h}}_i \hat{\mathbf{w}}_j}}{\sum_{t=1}^{\hat{T}} e^{\hat{\mathbf{h}}_t \hat{\mathbf{w}}_j}},$$

- Cross-entropy loss function



AttentionXML Datasets

Table 1: Datasets we used in our experiments.

Dataset	N_{train}	N_{test}	D	L	$\overline{L}$	$\hat{L}$	$\overline{W}_{train}$	$\overline{W}_{test}$
EUR-Lex	15,449	3,865	186,104	3,956	5.30	20.79	1248.58	1230.40
Wiki10-31K	14,146	6,616	101,938	30,938	18.64	8.52	2484.30	2425.45
AmazonCat-13K	1,186,239	306,782	203,882	13,330	5.04	448.57	246.61	245.98
Amazon-670K	490,449	153,025	135,909	670,091	5.45	3.99	247.33	241.22
Wiki-500K	1,779,881	769,421	2,381,304	501,008	4.75	16.86	808.66	808.56
Amazon-3M	1,717,899	742,507	337,067	2,812,281	36.04	22.02	104.08	104.18

N_{train}: #training instances, N_{test}: #test instances, D: #features, L: #labels, $\overline{L}$: average #labels per instance, $\hat{L}$: the average #instances per label, $\overline{W}_{train}$: the average #words per training instance and $\overline{W}_{test}$: the average #words per test instance. The partition of training and test is from the data source.

AttentionXML Evaluation (part)

Table 3: Performance comparisons of AttentionXML and other competing methods over six benchmarks. The results with the stars are from **Extreme Classification Repository** directly.

Methods	P@1=N@1	P@3	P@5	Methods	P@1=N@1	P@3	P@5
EUR-Lex				Amazon-670K			
AnnexML	79.66	64.94	53.52	AnnexML	42.09	36.61	32.75
DiSMEC	83.21	70.39	58.73	DiSMEC	44.78	39.72	36.17
PfastreXML	73.14	60.16	50.54	PfastreXML*	36.84	34.23	32.09
Parabel	82.12	68.91	57.89	Parabel	44.91	39.77	35.98
XT	79.17	66.80	56.09	XT	42.54	37.93	34.63
Bonsai	82.30	69.55	58.35	Bonsai	45.58	40.39	36.60
MLC2Seq	62.77	59.06	51.32	MCL2Seq	-	-	-
XML-CNN	75.32	60.14	49.21	XML-CNN	33.41	30.00	27.42
AttentionXML-1	*85.49*	*73.08*	*61.10*	AttentionXML-1	*45.66*	*40.67*	*36.94*
AttentionXML	**87.12**	**73.99**	**61.92**	AttentionXML	**47.58**	**42.61**	**38.92**

Ouroboros: On Accelerating Training of Transformer-Based Language Models

- The first model-parallel algorithm that speeds the training of Transformer-based language models;

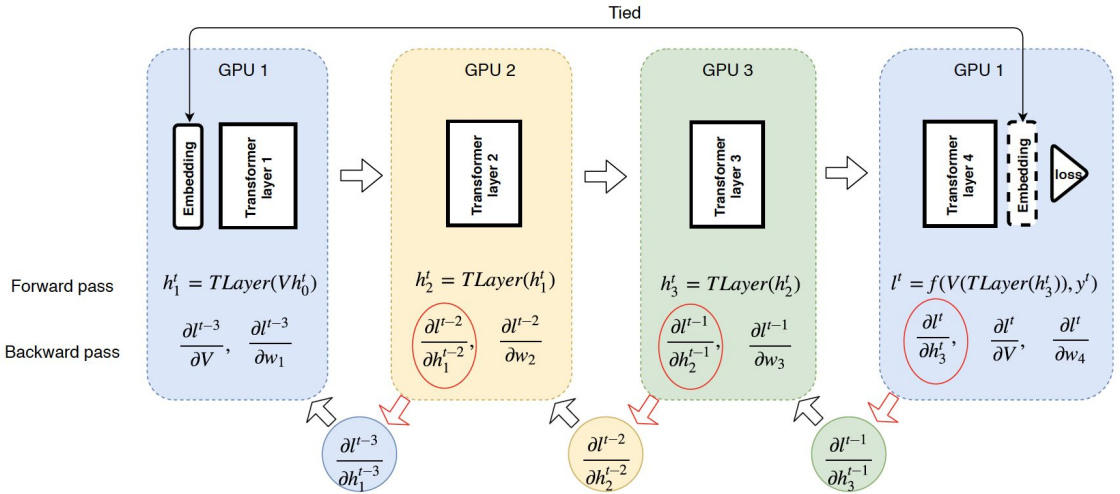


Figure 2: Forward and backward computation of the proposed algorithm. We split a Transformer-based language model into four modules and allocate them into three GPUs, where the first and the last module are placed on the same GPU. In the figure, h denotes activations, w denotes weights, and V represents embedding layers. $TLayler$ represents Transformer layer. The input embedding and output projection are tied together.

Evaluation

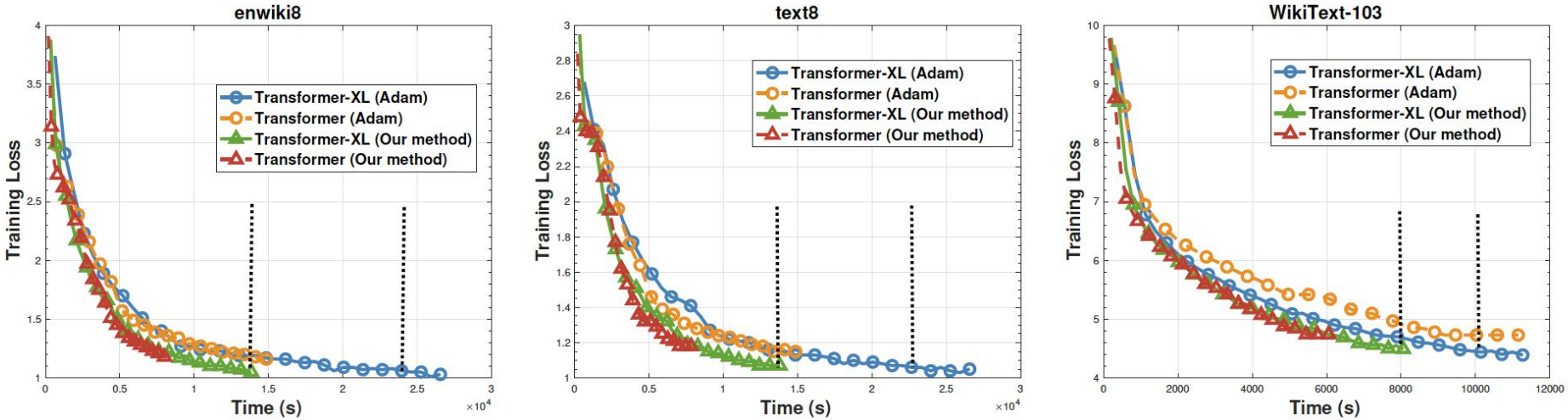


Figure 3: Convergence of the methods, regarding steps and computational time. We evaluate our algorithm on both Transformer and Transformer-XL language models.

Kernelized Bayesian Softmax for Text Generation

Motivation: a word may have multiple senses according to different context, some of which might be distinct;

KerBS, better softmax for text generation:

a) it employs a Bayesian composition of embeddings for words with multiple senses;

b) it is adaptive to semantic variances of words and **robust to rare sentence context** by imposing learned kernels to capture the closeness of words (senses) in the embedding space.

GNN Papers

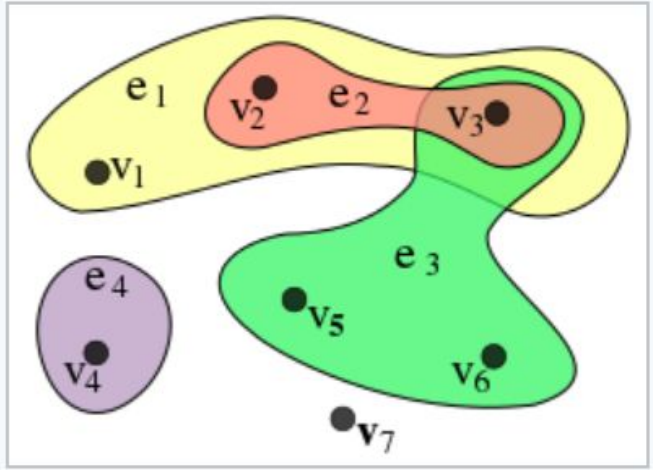
HyperGCN: A New Method of Training Graph Convolutional Networks on Hypergraphs

A **hypergraph** is a generalization of a graph in which **an edge can join any number of vertices**.

hypergraphs: relationships are complex and go beyond pairwise associations

a novel GCN for SSL on attributed hypergraphs.

(source wiki)



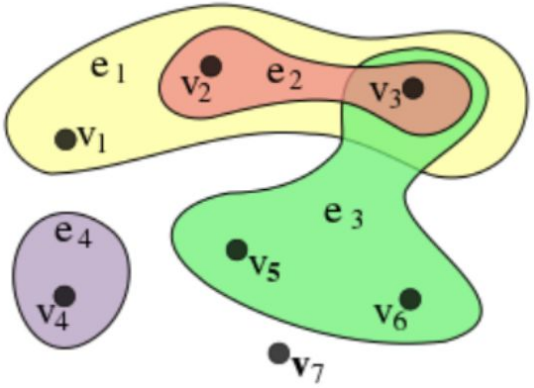
An example of a hypergraph, with $X = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and $E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$. This hypergraph has order 7 and size 4. Here, edges do not just connect two vertices but several, and are represented by colors.

Intuition

Problem definition:

$$\mathcal{H} = (V, E) \qquad |V| = n, |E| = m \text{ and a small set } V_L \text{ of labelled hypernodes.}$$

The task is to assign labels to each hypernode.



Share the same edge, (very likely) have the same label.

$$\sum_{e \in E} \max_{i,j \in e} ||h_i - h_j||^2$$

Define a Hypergraph Laplacian:

generalization from pairwise A to hypergraph

Method: Hypergraph Laplacian; 1-HyperGCN

Symmetrically normalized hypergraph Laplacian, to get A for the next later.

$$\mathbb{L}(S) := (I - D^{-\frac{1}{2}} A_S D^{-\frac{1}{2}})S \qquad S \in \mathbb{R}^n$$

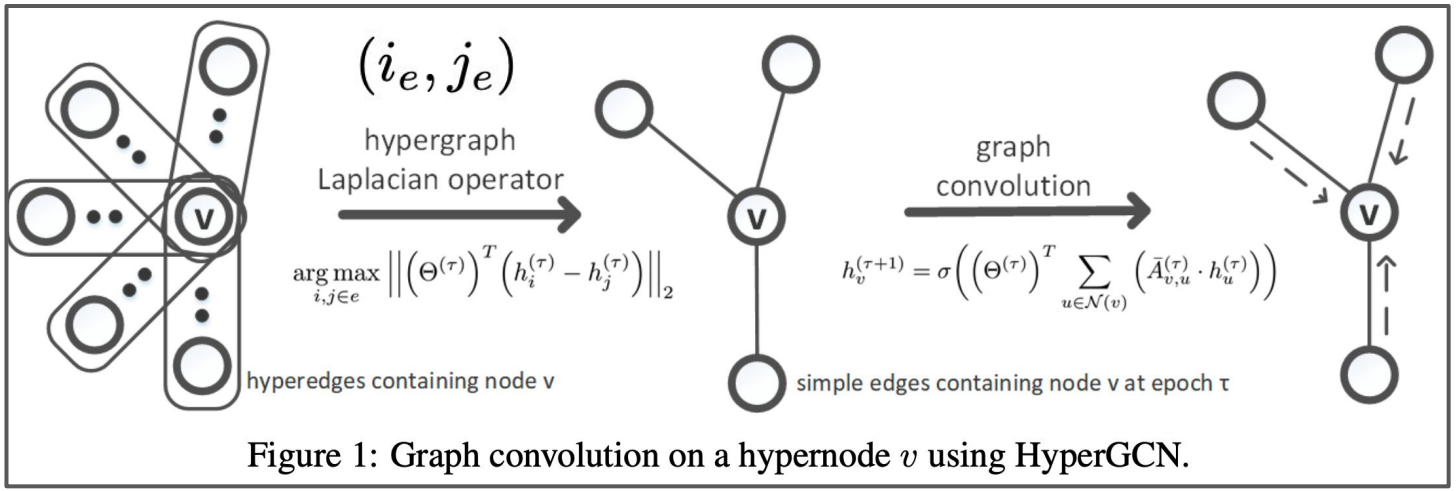


Figure 1: Graph convolution on a hypernode v using HyperGCN.

Datasets

Table 3: Real-world hypergraph datasets used in our work. Distribution of hyperedge sizes is not symmetric either side of the mean and has a strong positive skewness.

	DBLP (co-authorship)	**Pubmed** (co-citation)	**Cora** (co-authorship)	**Cora** (co-citation)	**Citeseer** (co-citation)
# hypernodes, $\lvert V\rvert$	43413	19717	2708	2708	3312
# hyperedges, $\lvert E\rvert$	22535	7963	1072	1579	1079
avg.hyperedge size	$\mathbf{4.7 \pm 6.1}$	$\mathbf{4.3 \pm 5.7}$	$\mathbf{4.2 \pm 4.1}$	$\mathbf{3.0 \pm 1.1}$	$\mathbf{3.2 \pm 2.0}$
# features, d	1425	500	1433	1433	3703
# classes, q	6	3	7	7	6
label rate, $\lvert V_L\rvert/\lvert V\rvert$	0.040	0.008	0.052	0.052	0.042

Results (part)

Table 4: Results of SSL experiments. We report mean test error ± standard deviation (lower is better) over 100 train-test splits. Please refer to section 5 for details.

Data	Method	**DBLP** co-authorship	**Pubmed** co-citation	**Cora** co-authorship	**Cora** co-citation	**Citeseer** co-citation
$\mathcal{H}$	CI	54.81 ± 0.9	52.96 ± 0.8	55.45 ± 0.6	64.40 ± 0.8	70.37 ± 0.3
$\mathbf{X}$	MLP	37.77 ± 2.0	30.70 ± 1.6	41.25 ± 1.9	42.14 ± 1.8	41.12 ± 1.7
$\mathcal{H}, \mathbf{X}$	MLP + HLR	30.42 ± 2.1	30.18 ± 1.5	34.87 ± 1.8	36.98 ± 1.8	37.75 ± 1.6
$\mathcal{H}, \mathbf{X}$	HGNN	25.65 ± 2.1	29.41 ± 1.5	31.90 ± 1.9	$\mathbf{32.41 \pm 1.8}$	$\mathbf{37.40 \pm 1.6}$
$\mathcal{H}, \mathbf{X}$	1-HyperGCN	33.87 ± 2.4	30.08 ± 1.5	36.22 ± 2.2	34.45 ± 2.1	38.87 ± 1.9
$\mathcal{H}, \mathbf{X}$	FastHyperGCN	27.34 ± 2.1	29.48 ± 1.6	32.54 ± 1.8	$\mathbf{32.43 \pm 1.8}$	$\mathbf{37.42 \pm 1.7}$
$\mathcal{H}, \mathbf{X}$	HyperGCN	$\mathbf{24.09 \pm 2.0}$	$\mathbf{25.56 \pm 1.6}$	$\mathbf{30.08 \pm 1.8}$	$\mathbf{32.37 \pm 1.7}$	$\mathbf{37.35 \pm 1.6}$

Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks

Motivation: Original GCNs on large graphs: high computation and memory costs; sampling-based methods to train GCNs on a subset of nodes.

Sampling based on:

- node-wise
- layer-wise

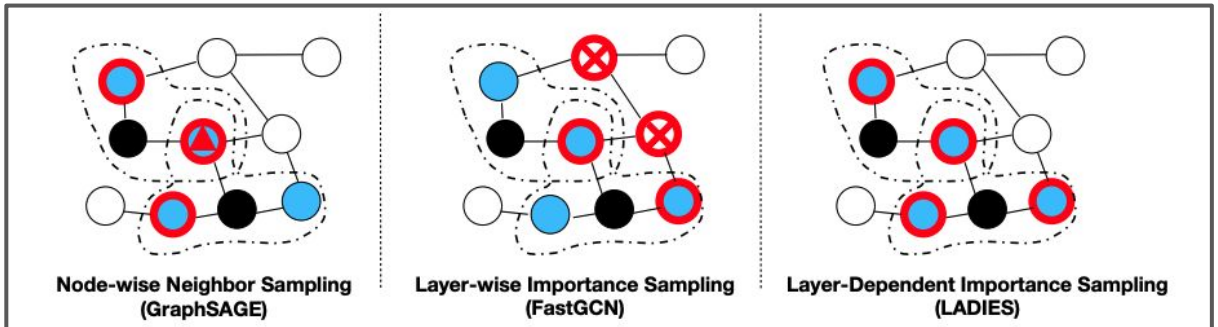


Figure 1: An illustration of the sampling process of GraphSage, FastGCN, and our proposed LADIES. Black nodes denote the nodes in the upper layer, blue nodes in the dashed circle are their neighbors, and node with the red frame is the sampled nodes. As is shown in the figure, GraphSAGE will redundantly sample a neighboring node twice, denoted by the red triangle, while FastGCN will sample nodes outside of the neighborhood. Our proposed LADIES can avoid these two problems.

LADIES

layer-wise

thus the neighbor nodes can be taken into account together to calculate next layers' embeddings without redundancy

neighbor-dependent

thus the sampled adjacency matrix is dense without losing much information for training

importance

sampling method should be adopted to reduce the sampling variance and accelerate convergence.

Results (part)

Dataset	Sample Method	F1-Score(%)	Total Time(s)	Mem(MB)	Batch Time(ms)	Batch Num
Cora (2708)	Full-Batch	76.5 ± 1.4	1.19 ± 0.82	30.72	15.75 ± 0.52	80.8 ± 51.7
	GraphSage (5)	75.2 ± 1.5	6.77 ± 4.94	471.39	78.42 ± 0.87	65.2 ± 52.1
	FastGCN (64)	25.1 ± 8.4	0.55 ± 0.65	3.13	9.22 ± 0.20	63.2 ± 71.2
	FastGCN (512)	78.0 ± 2.1	4.70 ± 1.35	7.33	10.08 ± 0.29	487 ± 147
	LADIES (64)	77.6 ± 1.4	4.19 ± 1.16	**3.13**	9.68 ± 0.48	436 ± 118.4
	LADIES (512)	**78.3 ±1.6**	**0.72 ± 0.39**	7.35	9.77 ± 0.28	75.6 ± 37.0
Citeseer (3327)	Full-Batch	62.3 ± 3.1	0.61 ± 0.70	68.13	15.77 ± 0.58	40.6 ± 22.8
	GraphSage (5)	59.4 ± 0.9	4.51 ± 3.68	595.71	53.14 ± 1.90	57.2 ± 42.1
	FastGCN (64)	19.2 ± 2.7	0.53 ± 0.48	5.89	8.88 ± 0.40	64.0 ± 57.0
	FastGCN (512)	44.6 ± 10.8	4.34 ± 1.73	13.97	10.41 ± 0.51	386 ± 167
	FastGCN (1024)	63.5 ± 1.8	2.24 ± 1.01	23.24	10.54 ± 0.27	223 ± 98.6
	LADIES (64)	**65.0 ± .1.4**	2.17 ± 0.65	**5.89**	9.60 ± 0.39	232 ± 66.8
	LADIES (512)	64.3 ± 2.4	**0.41 ± 0.22**	13.92	10.32 ± 0.23	37.6 ± 11.9

Keep It Simple: Graph Autoencoders Without Graph Convolutional Networks

Replace GCN **encoder** with a linear model wrt the adjacency matrix of the graph and a unique weight matrix.

$$Z = \tilde{A}W \quad \text{then} \quad \hat{A} = \sigma(ZZ^T).$$

Takeaways:

Dense datasets: Blogs, Google pages, etc. GCN encoder performance increases with the size of the graph.

Nature of the dataset is crucial: in citation graphs, if a reference A in an article B cited by some authors is relevant to their work, authors will likely also cite this reference A (creating a first order link)

Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks

-social interactions between humans and their physical interactions with the scene

-a graph-based generative adversarial network: graph attention network + Bicycle-GAN

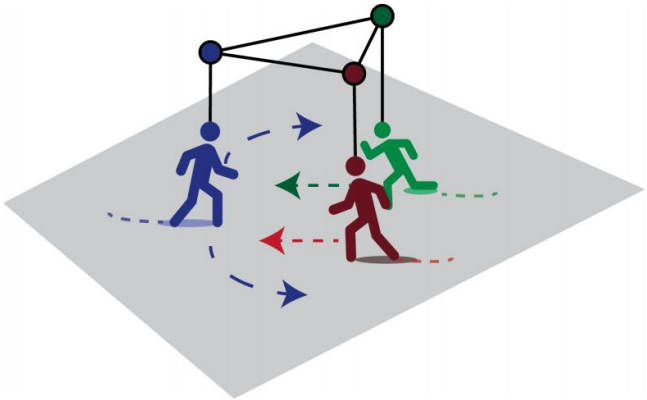


Figure 1: We show multimodal behavior for the blue pedestrian, who must make a decision about which direction they will take to avoid the red-green pedestrian group.

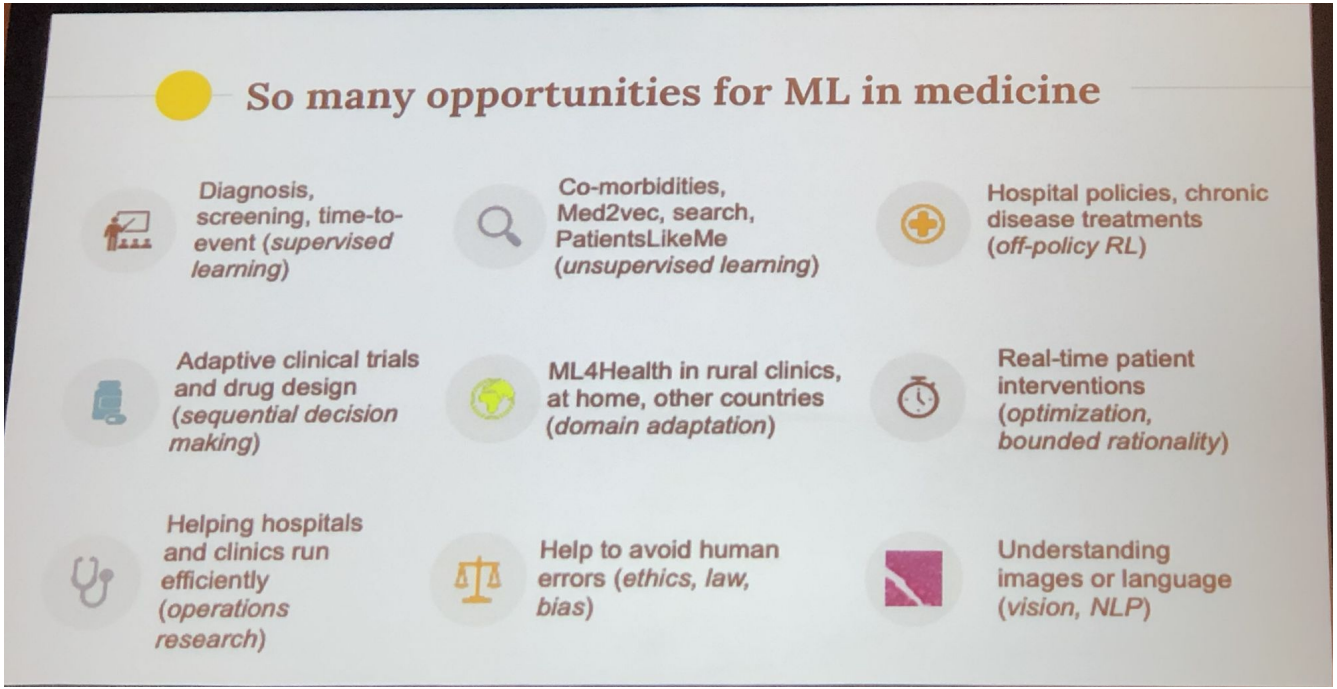
DL+Healthcare

ML/DL + Healthcare

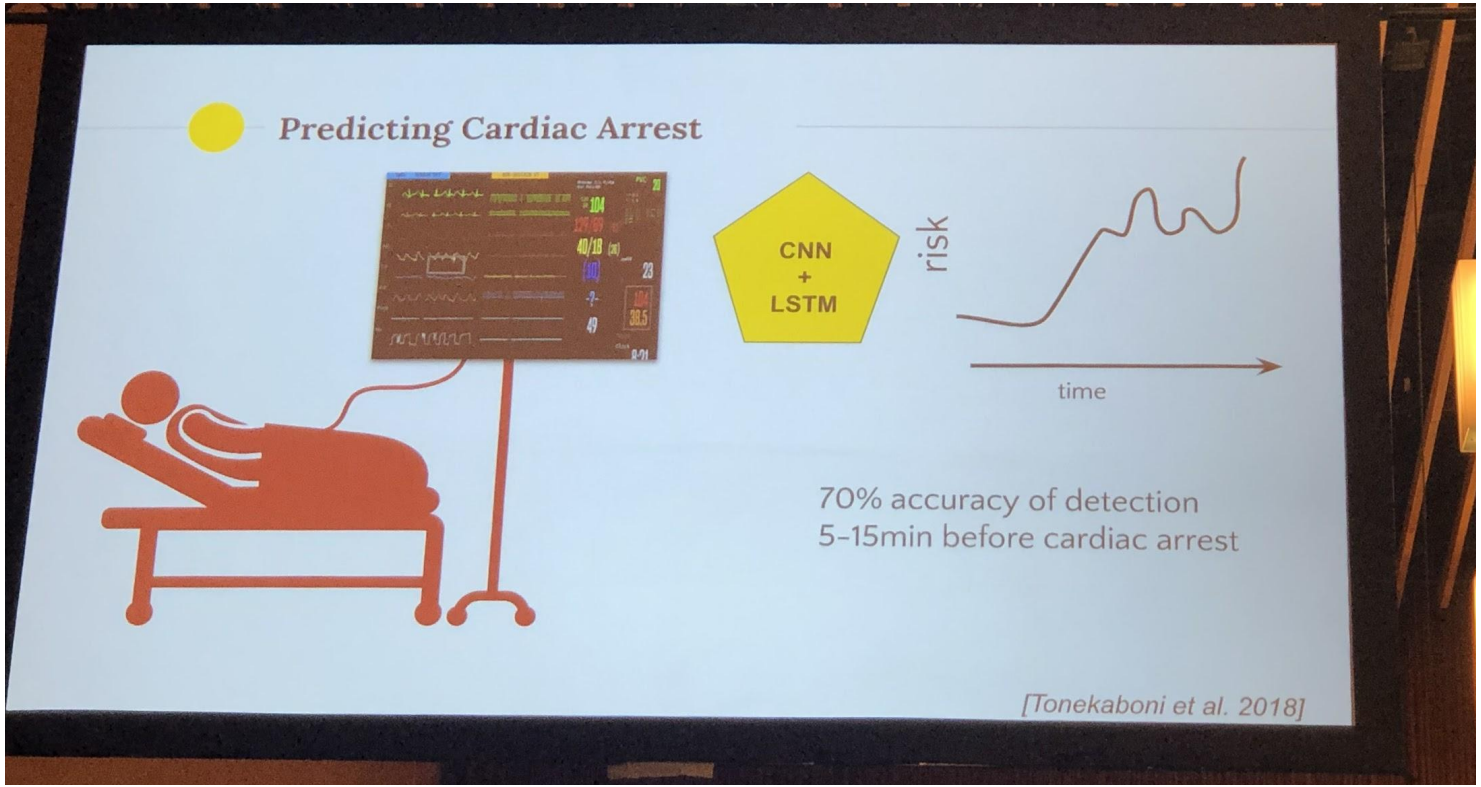
ML for computational biology and health;

Challenges in data;

Biomedical data can drive innovation in ML.



When Clinicians/Doctors try to integrate ML/DL...



Transfusion: Understanding Transfer Learning for Medical Imaging

VGG, Bert; Pretraining, fine tuning

TL for medical imaging: not quite similar

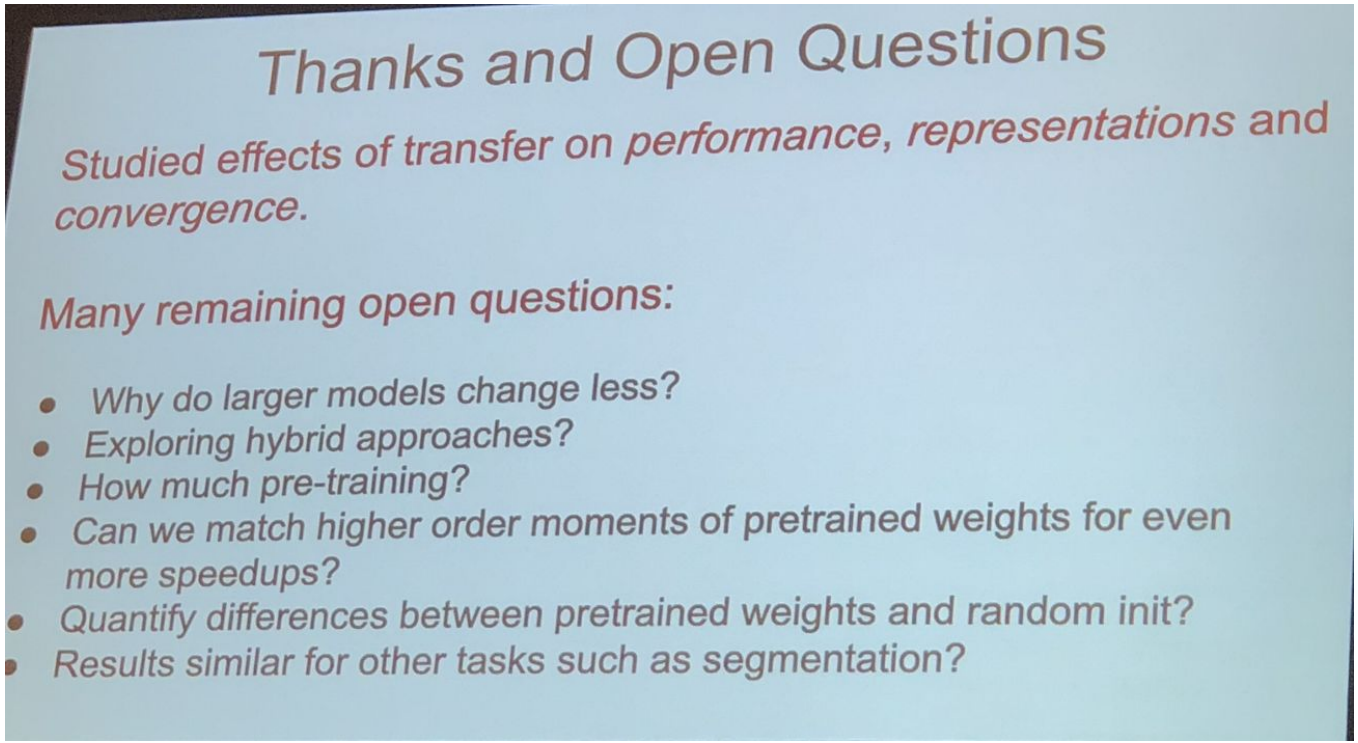
Takeaway:

TL and random init perform the same; small models can work like larger ones.

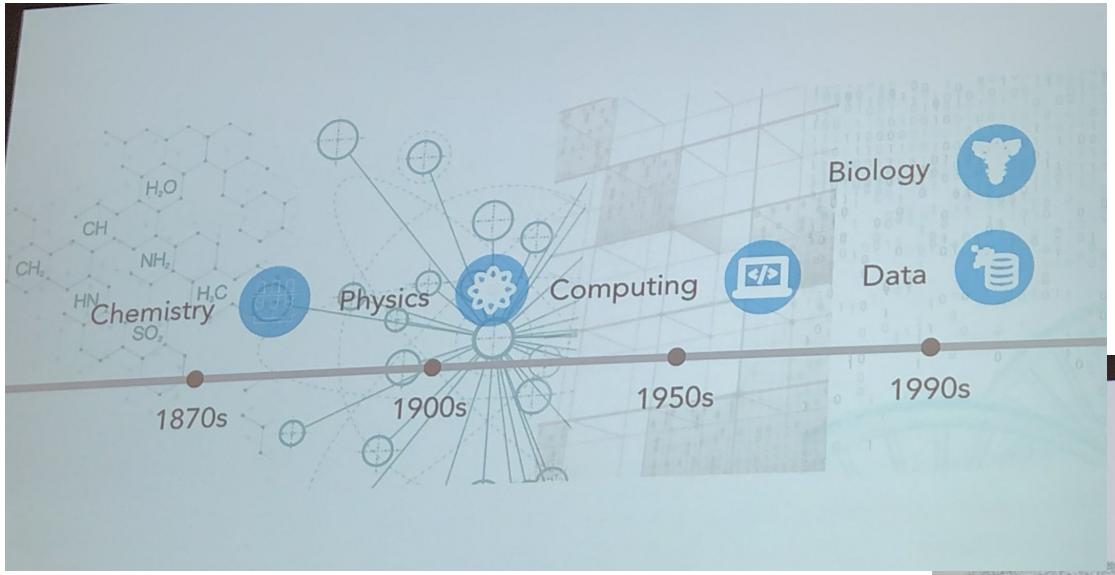
How does TL help/effect?

-visualization; convergence speed

Transfer learning and other thoughts

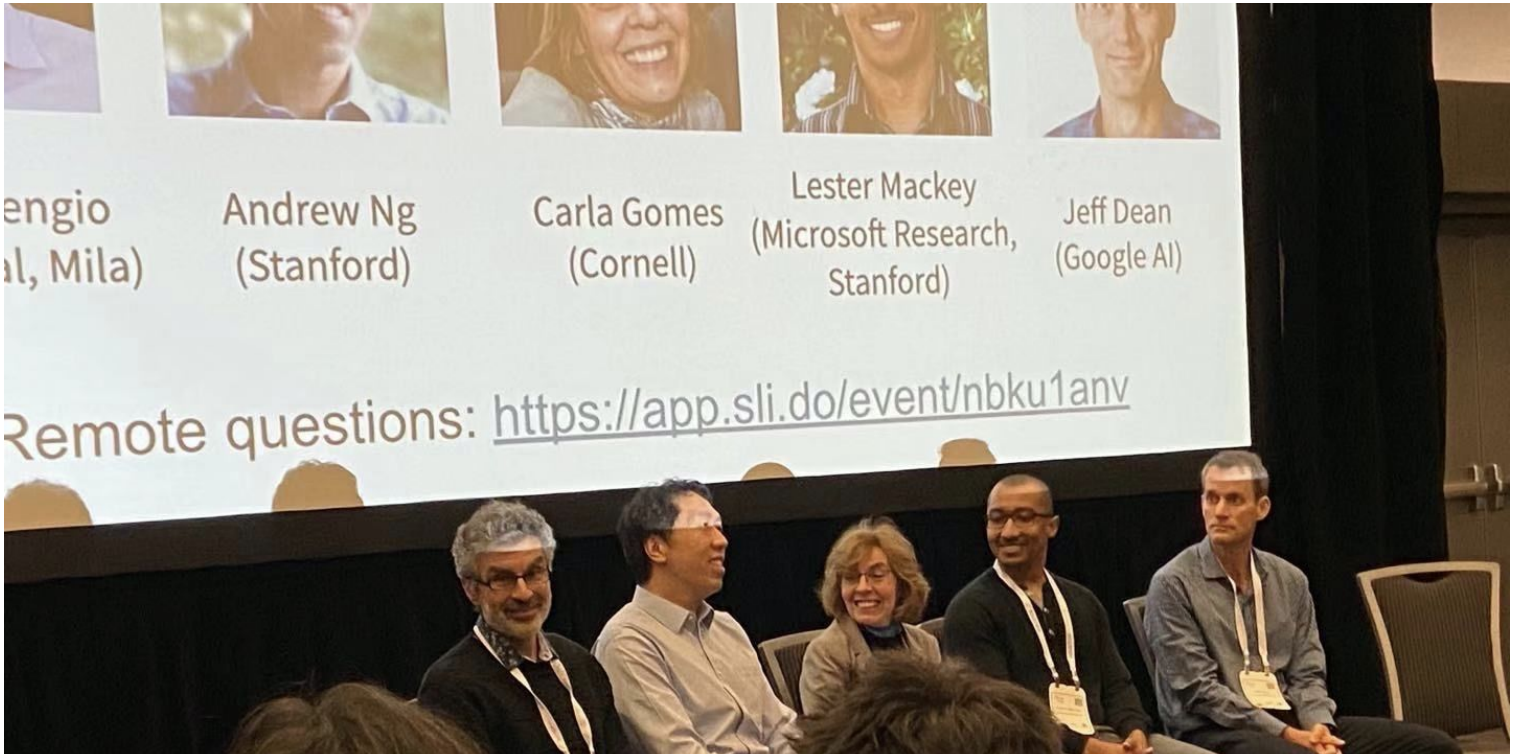


(**Daphne Koller**) ML: a new approach to drug



A Special Panel

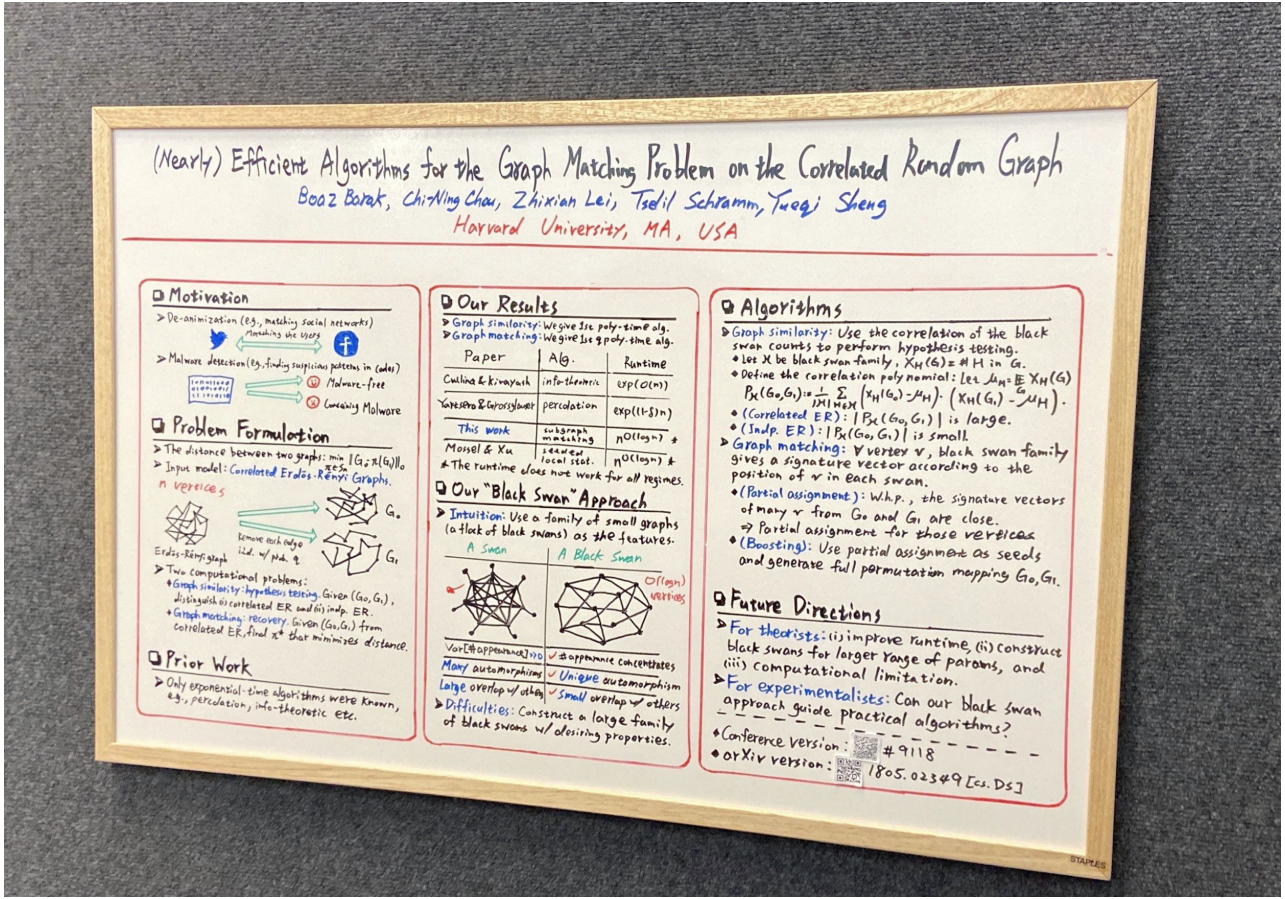
Panel in Climate Change + ML(1)



Panel in Climate Change + ML(2)

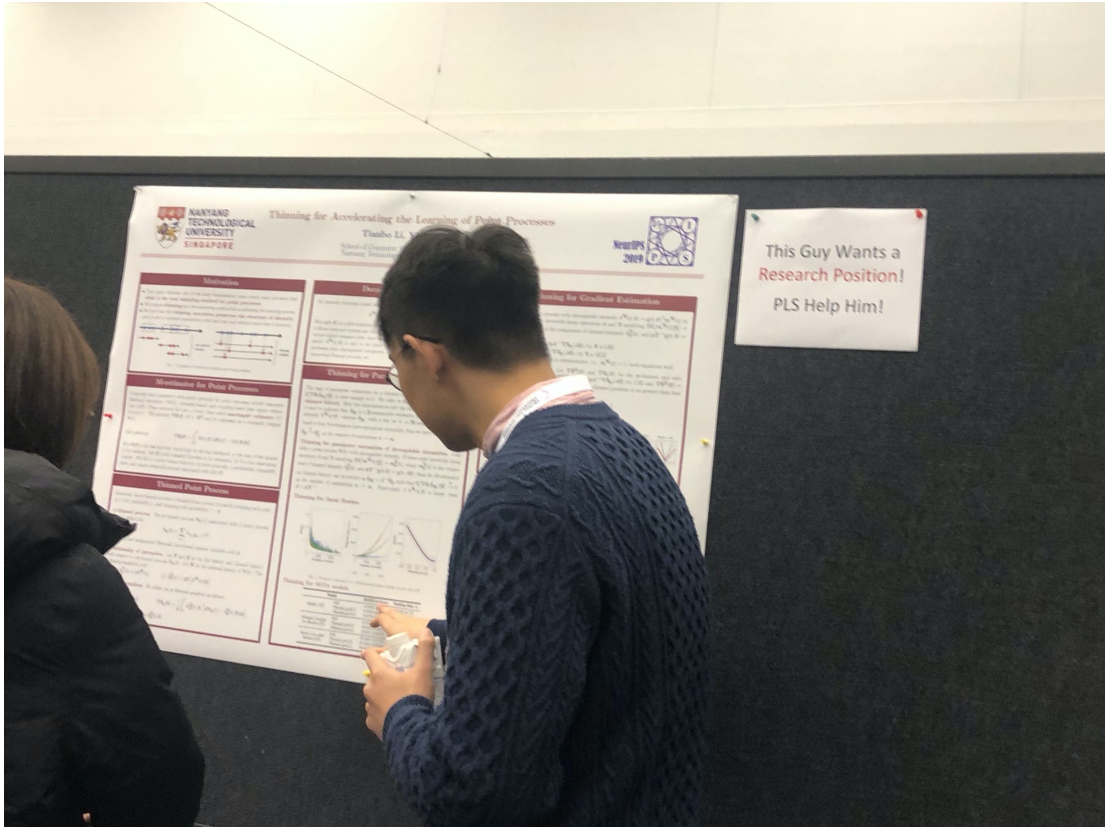
1. work for small and raw dataset: **self-supervised** learning, very exciting and applicable
2. train large multi-task models, **transfer learning** to get good results on new task where only a few samples
3. combine prior knowledge, scientific reasoning, unsupervised learning, to get meaningful solutions
4. New directions like RL: energy consumption, etc
5. **Q&A**:
   change management problem
   work with domain experts
   ML is not always very helpful (ethics)

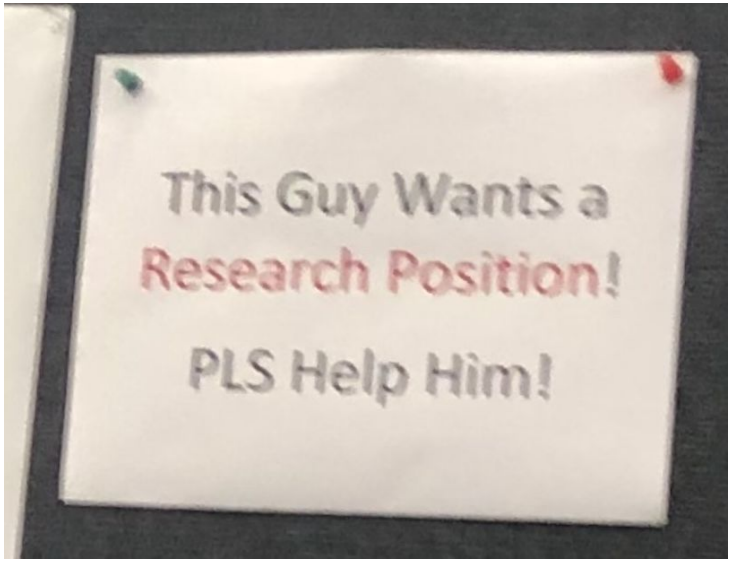
Other Fun Facts



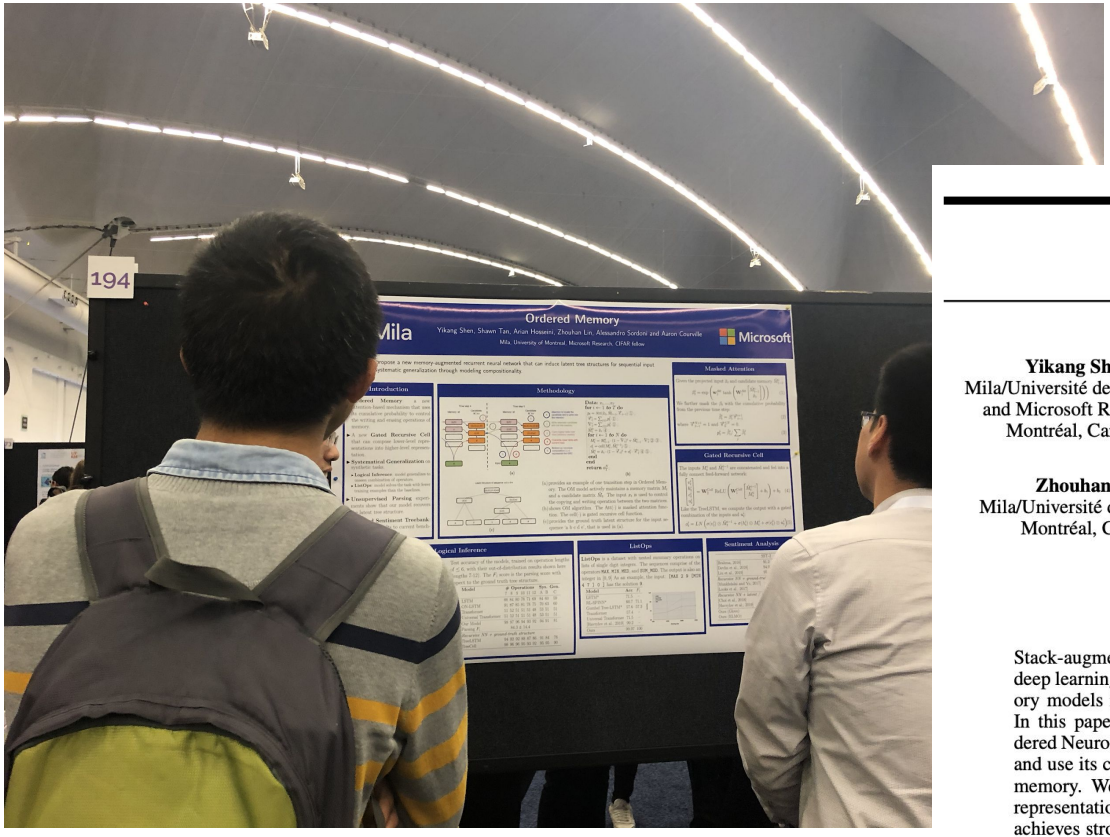
A poster in the main poster session.

Large Posters!





How to find a job during poster session?



A Very Short Title…. link

Ordered Memory

Yikang Shen* Mila/Université de Montréal and Microsoft Research Montréal, Canada

Shawn Tan* Mila/Université de Montréal Montréal, Canada

Arian Hosseini* Mila/Université de Montréal and Microsoft Research Montréal, Canada

Zhouhan Lin Mila/Université de Montréal Montréal, Canada

Alessandro Sordoni Microsoft Research Montréal, Canada

Aaron Courville Mila/Université de Montréal Montréal, Canada

Abstract

Stack-augmented recurrent neural networks (RNNs) have been of interest to the deep learning community for some time. However, the difficulty of training memory models remains a problem obstructing the widespread use of such models. In this paper, we propose the Ordered Memory architecture. Inspired by Ordered Neurons (Shen et al., 2018), we introduce a new attention-based mechanism and use its cumulative probability to control the writing and erasing operation of memory. We also introduce a new Gated Recursive Cell to compose lower level representations into higher level representation. We demonstrate that our model achieves strong performance on the logical inference task (Bowman et al., 2015) and the ListOps (Nangia and Bowman, 2018) task. We can also interpret the model to retrieve the induced tree structure, and find that these induced structures align with the ground truth. Finally, we evaluate our model on the Stanford Sentiment Treebank tasks (Socher et al., 2013), and find that it performs comparatively with the state-of-the-art methods in the literature[2].

13k Registered

Thanks

Q&A

ireneli.eu

Other GCN links

https://tkipf.github.io/graph-convolutional-networks/ (GCN tutorial)

https://www.dgl.ai/ (DGL library)

Graph-BERT paper: https://arxiv.org/pdf/2001.05140v1.pdf

GCN to text classification: https://arxiv.org/pdf/1809.05679.pdf