Learning to Teach && Generating Sentences by Editing Prototypes

Yong Jiang

Tencent AI Lab && ShanghaiTech University

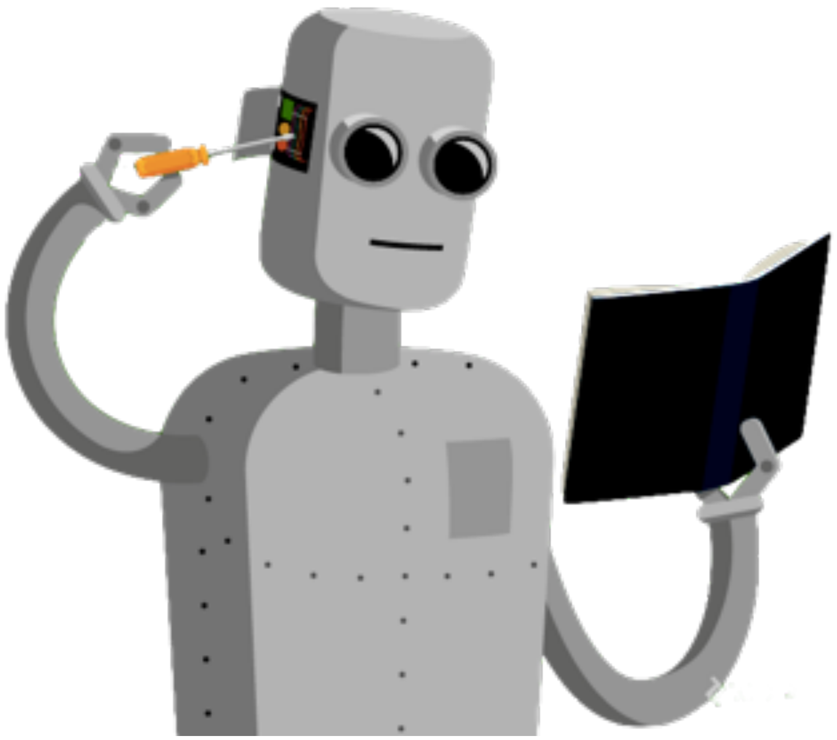
Paper 1

LEARNING TO TEACH

Yang Fan[†]*, Fei Tian[‡]*, Tao Qin[‡], Xiang-Yang Li[†] & Tie-Yan Liu[‡]
[†]School of Computer Science and Technology, University of Science and Technology of China
fyabc@mail.ustc.edu.cn, xiangyangli@ustc.edu.cn
[‡]Microsoft Research
{fetia,taoqin,tyliu}@microsoft.com

Motivation

Motivation



Supervised Learning Strategies

- From easy to hard
  - Curriculum Learning
  - Self-paced Learning
- From convex to non-convex
  - Graduated Optimization
- Need task specific heuristic rules!!!
- How to adaptively adopt different teaching strategies?
  - Design loss function for current mini-batch?
  - Design appro-data for current mini-batch?
  - Design hypothesis space for current mini-batch?

The Student Model (Model S)

- Typically, can be seen as our familiar models in sup learn.
  - predictor f with param w, in function class \Omega
  - metric M for evaluation the gap (l0 loss/cross-entropy)
  - training data D
- Minimum risk:

$$R(\omega) = \int \mathcal{M}(y, f_\omega(x))\, \mathrm{d}P(x, y)$$

- In real application:

$$\omega^* = \arg\min_{\omega \in \Omega} \sum_{(x,y)\in D} L(y, f_\omega(x)) \triangleq \underset{\uparrow}{\mu(D, L, \Omega)}$$

learning algorithm of student model

The Teacher Model (Model T)

- Input: info from Model S and past teaching history
- Output: provide inputs to the student model:
  - Training data D
  - Loss function L
  - Hypothesis space \Omega
- Minimum risk: $\min_{D,L,\Omega} \mathcal{M}(\mu(D, L, \Omega), D_{test})$
- In this paper, the authors use "data teaching".
- They point out that they are working on NMT applications with "loss function teaching" in the author response.

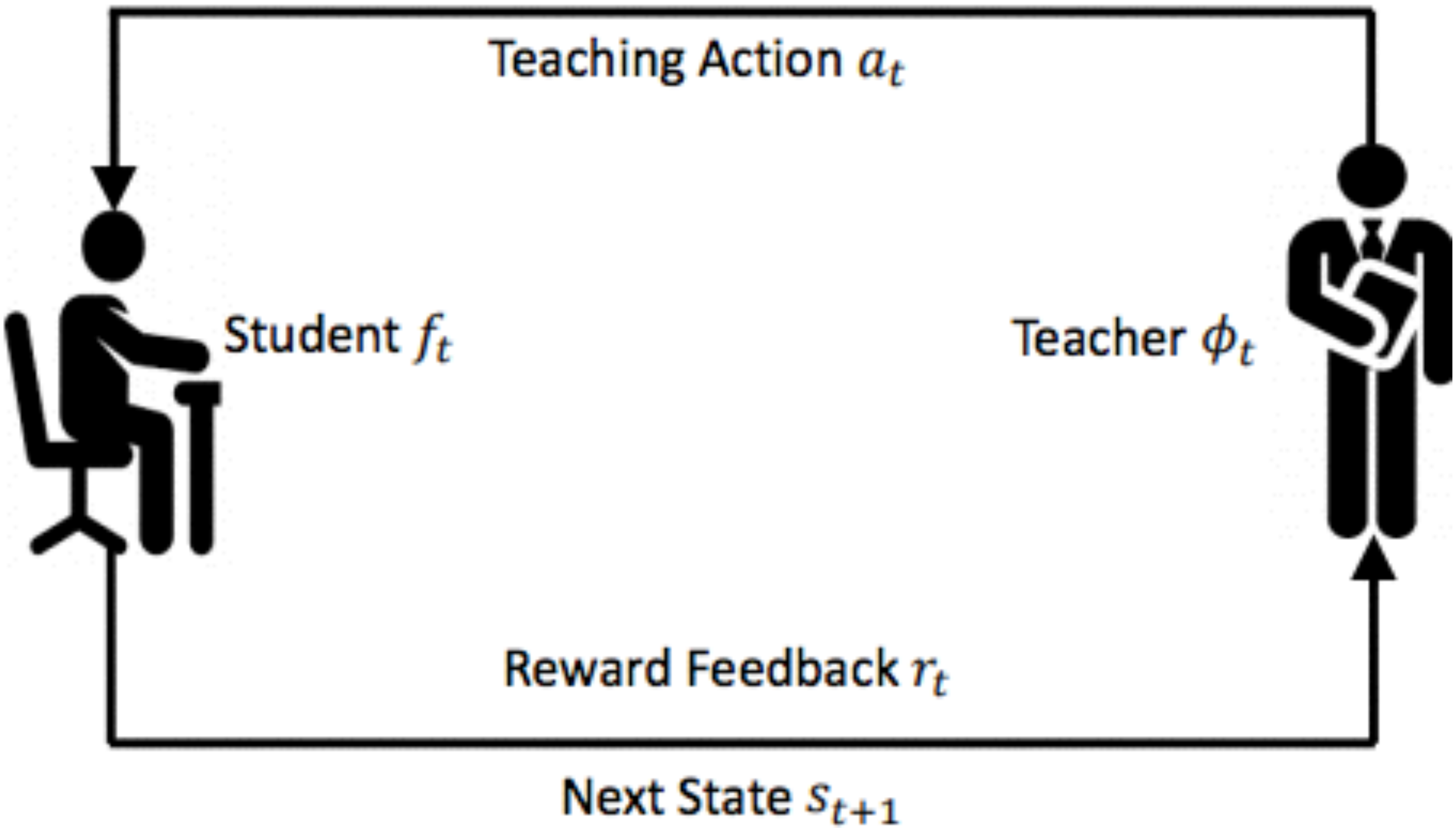
How to Teach?

- Modelling as a sequential decision process
  - state s_t \in S, the input for the Model T.
    - s_t is constructed from Model S f_{t-1}
    - and past teaching history
  - action a_t \in A: output of Model T
  - policy: S -> A paramed by \theta (a.k.a: Model T)
  - reward r_t: how good current Model S is.
- RL for learning:

$$\max_{\theta} \sum_{t} r_t = \max_{\theta} \sum_{t} r(f_t) = \max_{\theta} \sum_{t} r(\mu(\phi_\theta(s_t), L, \Omega))$$

How to Teach?

- Modelling as a sequential decision process



-

Data Teaching: State Features

- Data features (used in curriculum learning)
  - info for data instance, label category, length of sent.
  - linguistic features for text segments.
  - gradients histogram features.
- Features in Model T
  - passed mini-batch number
  - average historical training loss
  - historical validation accuracy
- Combination of both

Data Teaching: Optimization

- Objective function:

$$J(\theta) = E_{\phi_\theta(a|s)}[R(s,a)]$$

- Policy gradient:

$$\nabla_\theta = \sum_{t=1}^{T} E_{\phi_\theta(a_t|s_t)}[\nabla_\theta \log \phi_\theta(a_t|s_t) R(s_t,a_t)]$$
$$\nabla_\theta \approx \sum_{t=1}^{T} \nabla_\theta \log \phi_\theta(a_t|s_t) r_T$$

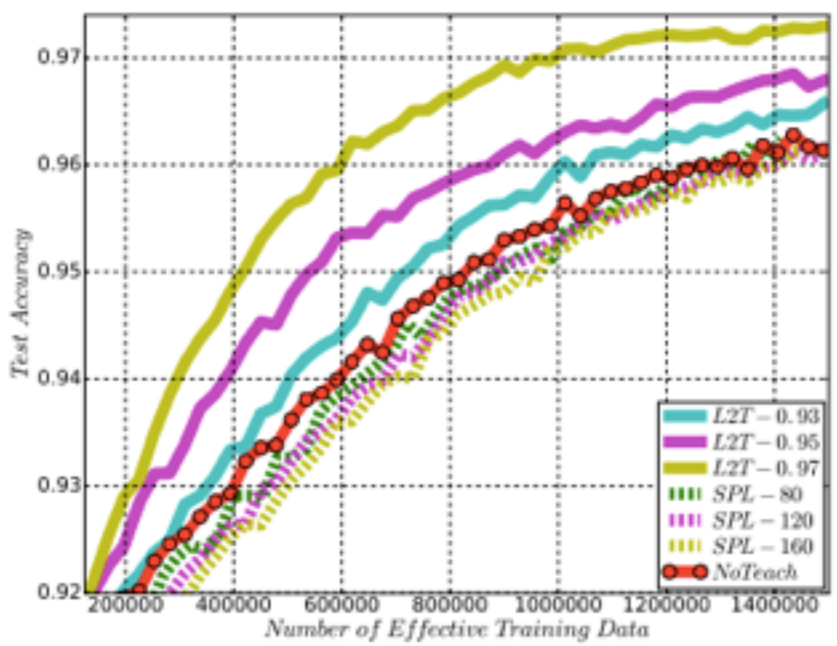
Terminal Reward

- Terminal reward is set by the first mini-batch index in which the accuracy on dev set exceeds τ

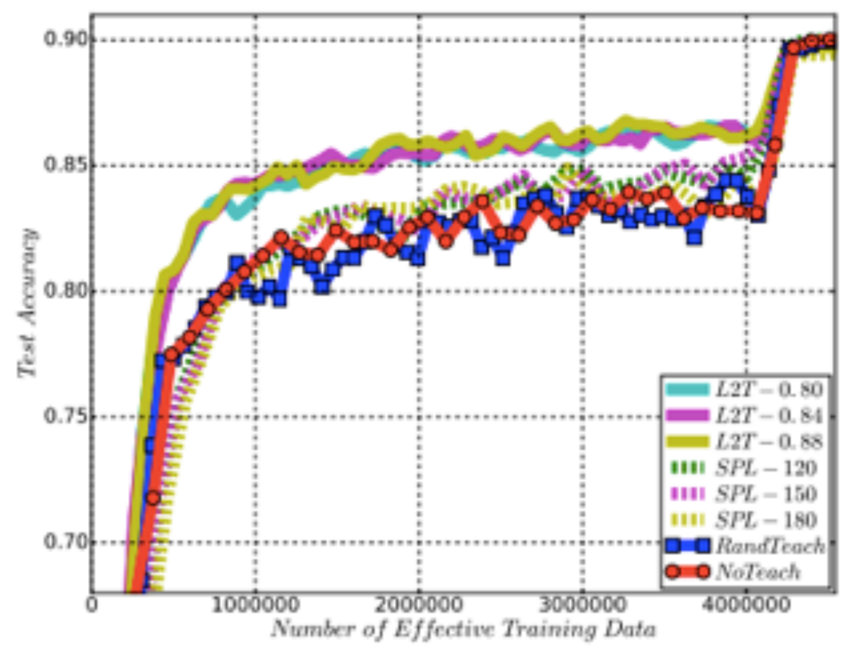
$$r_T = -\log(i_\tau / T')$$

Experiments

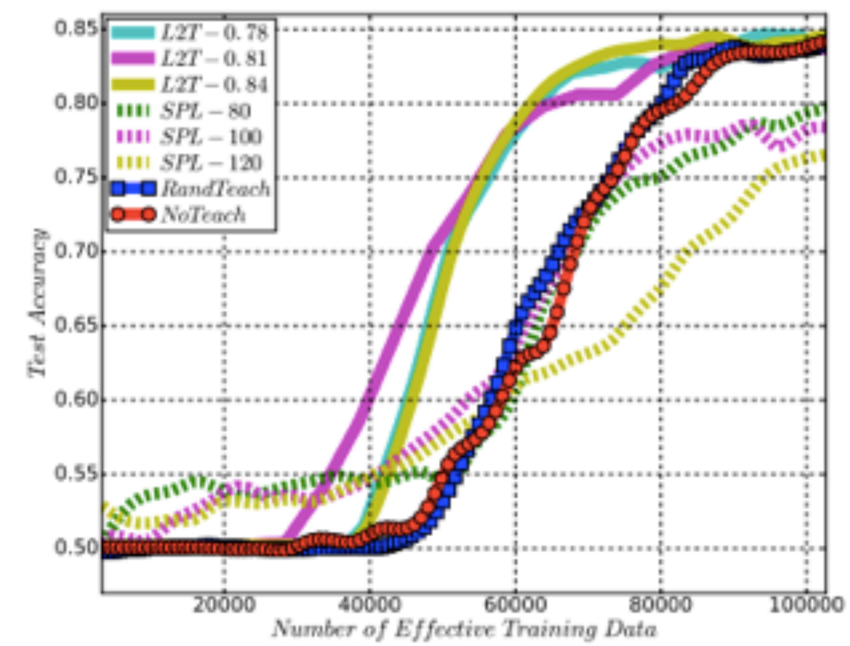
- Teaching a new student with the same mode architecture



(a) MNIST



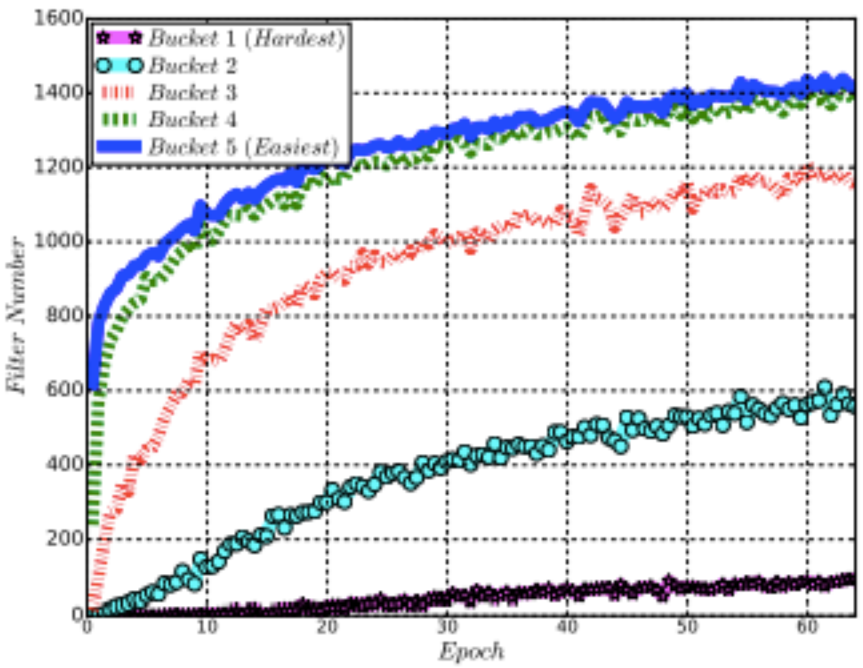
(b) CIFAR-10



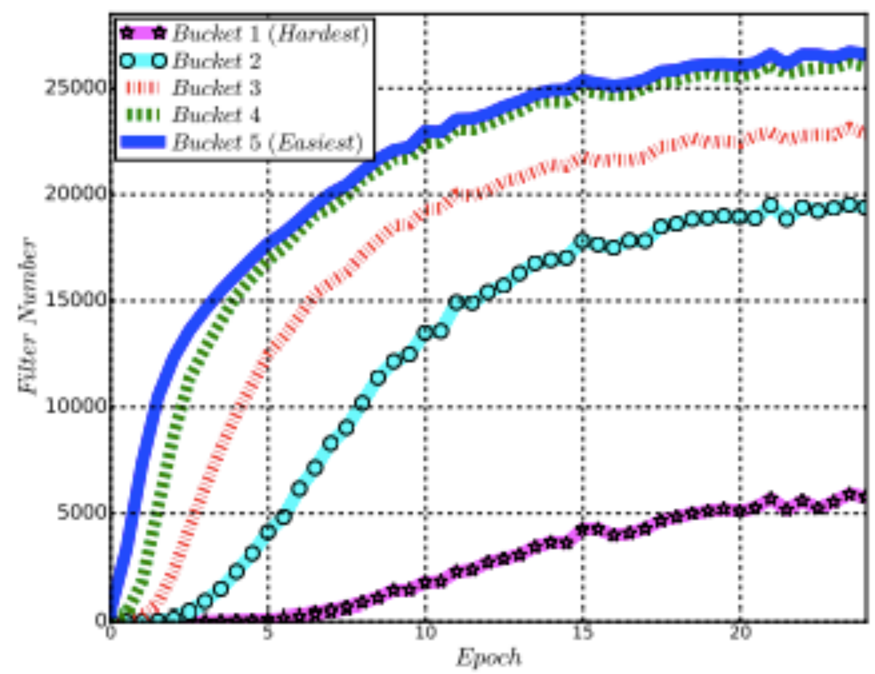
(c) IMDB

Experiments

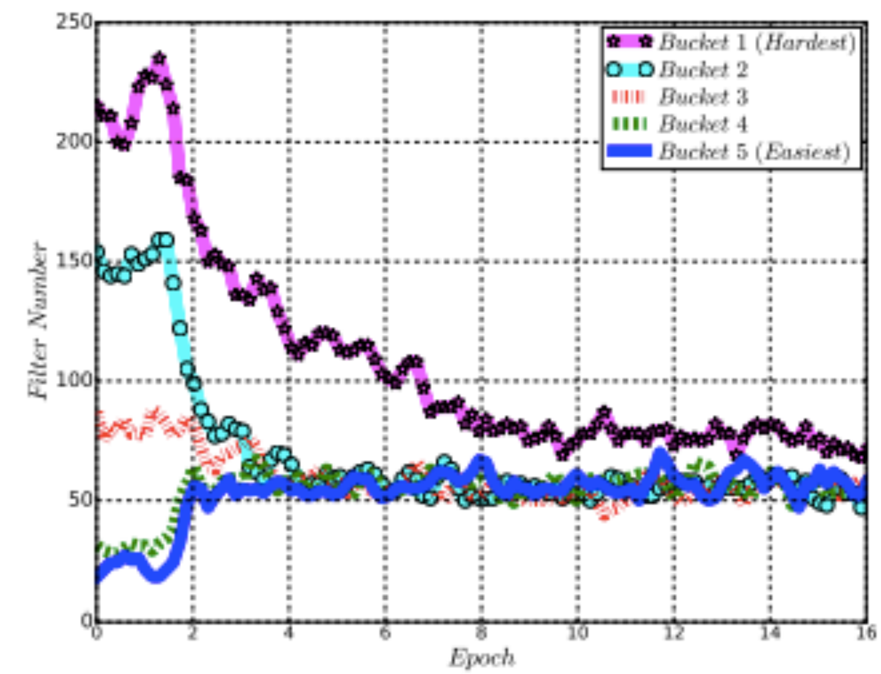
- Filtration number analysis



(a) MNIST



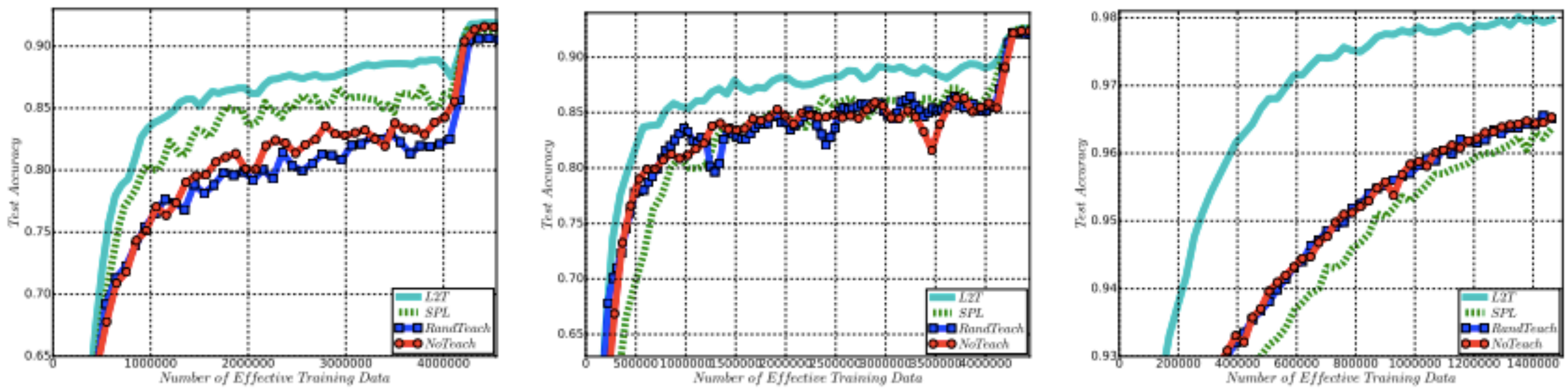
(b) CIFAR-10



(c) IMDB

Experiments

- Teaching a new student with different architecture



(a) ResNet32 → ResNet110 (b) MNIST → CIFAR-10 (c) CIFAR10→MNIST

Experiments

- How about accuracy?

Teaching Policy	NoTeach	SPL	L2T
Accuracy	88.54%	88.80%	**89.46%**

Paper 2

Generating Sentences by Editing Prototypes

Kelvin Guu*[2] Tatsunori B. Hashimoto*[1,2] Yonatan Oren[1] Percy Liang[1,2]
(* equal contribution)
[1]Department of Computer Science [2]Department of Statistics
Stanford University
{kguu,thashim,yonatano}@stanford.edu pliang@cs.stanford.edu

Motivation

- Generation is HARD!!
  - favoring generic utterances such as “I don’t know”
  - the reason is due to starting from SCRATCH?
- Why prototype-then-edit?
  - High quality corpus in training data.

Approach

- Math is fancy! But it can be simplified!
  - the reason is due to starting from SCRATCH?

$$\mathcal{L}_{\text{Lex}} = \sum_{x \in \mathcal{X}} \sum_{x' \in \mathcal{N}(x)} \log p(x \mid x')$$

- Conditional generation process

$$\begin{aligned}
\log p(x \mid x') &= \log \int_z p_{\text{edit}}(x \mid x', z) p(z) dz \\
&\geq \ell(x, x') = \mathbb{E}_{z \sim q(z \mid x, x')} [\log p_{\text{edit}}(x \mid x', z)] \\
&\qquad - D_{\text{KL}}(q(z \mid x, x') \| p(z)).
\end{aligned}$$

Model Components

- Neural Editor, a.k.a: decoder in VAE, P(x|x',z)
  - encoder-decoder structure
  - at each time step, vector z is concatted to predict x.
- Edit Prior, a.k.a: prior in VAE, P(z)
  - In vanilla VAE, a gaussian
- Approximate edit posterior,
- a.k.a: Encoder in vanilla VAE, Q(z|x,x')
  - In vanilla VAE, a gaussian

Model Components

- Edit Prior, a.k.a: prior in VAE
  - two steps: $z_{\text{norm}} \sim \text{Unif}(0, 10)$ $z_{\text{dir}} \sim \text{vMF}(0)$
  - result: $z = z_{\text{norm}} z_{\text{dir}}$
- Approximate edit posterior, a.k.a: Encoder in vanilla VAE
  - Generalization of word vectors $f(x, x') = \sum_{w \in I} \Phi(w) \oplus \sum_{w \in D} \Phi(w)$
  - Stochastic!
    - perturb the norm of f by adding uniform noise
    - perturb direction of f by adding von-Mises Fisher noise

$$
\begin{aligned}
q(z_{\text{dir}} \mid x, x') &= \text{vMF}\left(z_{\text{dir}}; f_{\text{dir}}, \kappa\right) \\
&\propto \exp(\kappa z_{\text{dir}}^{\top} f_{\text{dir}})
\end{aligned}
$$

$$q\left(z_{\text{norm}} \mid x, x'\right) = \text{Unif}(z_{\text{norm}}; [\tilde{f}_{\text{norm}}, \tilde{f}_{\text{norm}} + \epsilon])$$

$$\Rightarrow \quad z = z_{\text{norm}} z_{\text{dir}}$$

Learning

- KL term:

$$D_{\mathrm{KL}}(\mathrm{vMF}(\kappa)\|\mathrm{vMF}(0)) = \kappa \frac{I_{d/2+1}(\kappa) + I_{d/2}(\kappa)\frac{d}{2\kappa}}{I_{d/2}(\kappa) - \frac{d}{2\kappa}} + \frac{d}{2}\log(\kappa/2) - \log(I_{d/2}(\kappa)\Gamma(d/2+1)),$$

- Reconstruction Term:
  - Reparameterization Trick
  - Reject Sampling of Wood (1994)

Experiments: Perplexity

Model	Perplexity (Yelp)	Perplexity (BILLIONWORD)
KN5	56.546	78.361
KN5+MEMORIZATION	55.180	73.468
NLM	40.174	55.146
NLM+MEMORIZATION	38.980	50.969
NLM+KN5	38.149	**47.472**
NEURALEDITOR($\kappa = 0$)	27.600	48.755
NEURALEDITOR($\kappa = 25$)	**27.480**	48.921

Table 1: Perplexity of the NEURALEDITOR with the two VAE parameters κ outperform all methods on YELP and all non-ensemble methods on BILLIONWORD.

Experiments: Examples

Prototype x'	**Revision** x
i had the fried whitefish taco which was decent, but i've had much better.	i had the <unk> and the fried carnitas tacos, it was pretty tasty, but i've had better.
"hash browns" are unseasoned, frozen potato shreds burnt to a crisp on the outside and mushy on the inside.	the hash browns were crispy on the outside, but still the taste was missing.
i'm not sure what is preventing me from giving it <cardinal> stars, but i probably should.	i'm currently giving <cardinal> stars for the service alone.
quick place to grab light and tasty teriyaki.	this place is good and a quick place to grab a tasty sandwich.
sad part is we've been there before and its been good.	i've been here several times and always have a good time.

Table 2: Edited generations are substantially different from the sampled prototypes.

Experiments: Comp with SVAE

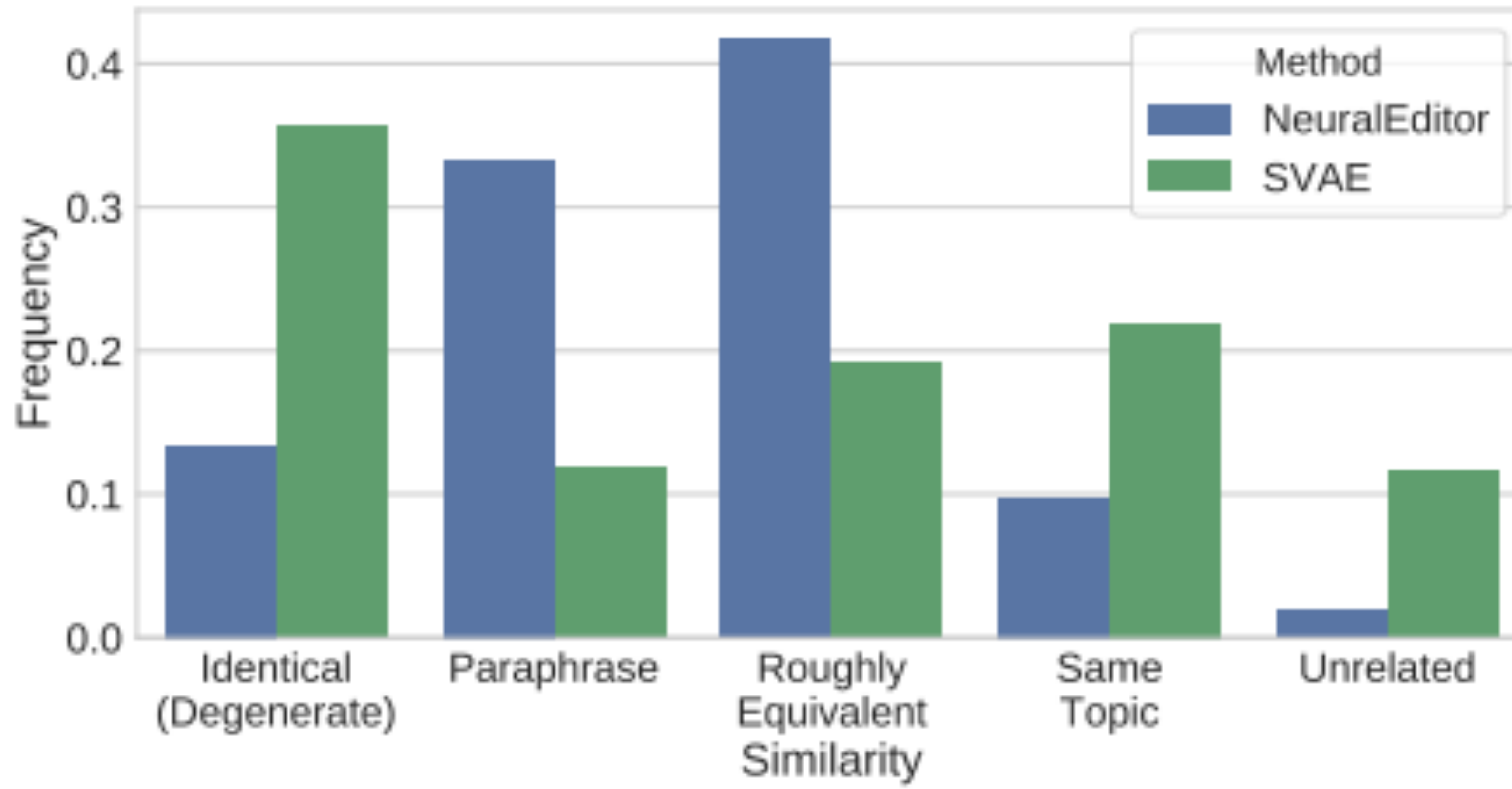


Figure 4: The neural editor frequently generates paraphrases and similar sentences while avoiding unrelated and degenerate ones. In contrast, the SVAE frequently generates identical and unrelated sentences and rarely generates paraphrases.[10]

Ideas

- Learning to reweight training sample in noise situations
- Learning to provide searching space && loss in generation
- Specialised the edit vector
  - sentiment
  - topic
  - subject
  - …