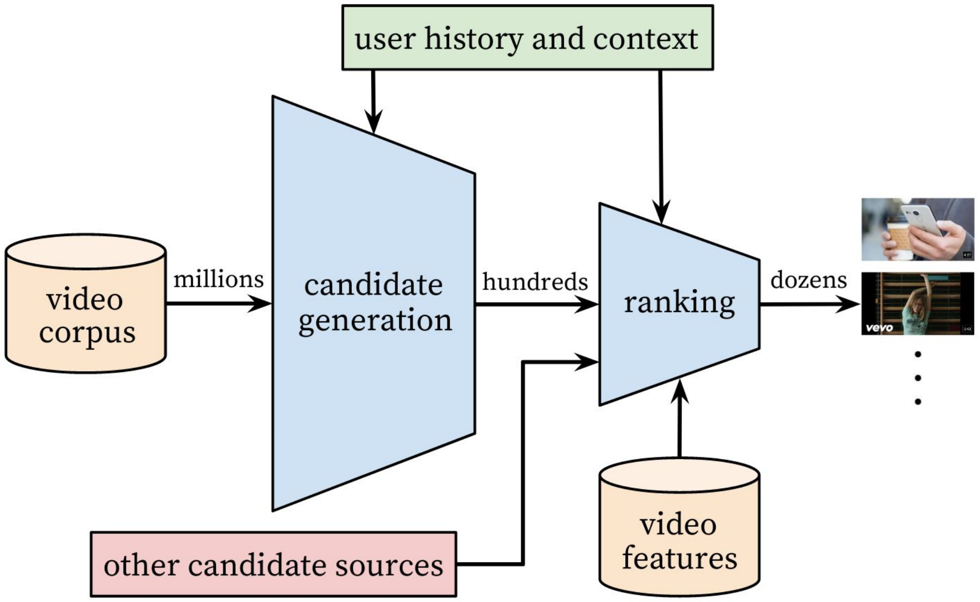
Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems

Deep Candidate Generation



- pre-compute offline the item representations via the item encoder
- use online vector-based kNN service for fast retrieval of top items

[Deep Neural Networks for YouTube Recommendation](#)

MLE

- D - dataset of user clicks: $x_{u,t}$ represents user's clicks prior to the t-th click $y_{u,t}$
- T_u denotes the number of clicks from the user u
- X - set of all possible click sequences
- y - represents a clicked item

$$\mathcal{D} = \{(x_{u,t},\, y_{u,t}) : u = 1, 2, \dots, N,\, t = 1, 2, \dots, T_u\}$$

$$x_{u,t} = \left\{y_{u,1:(t-1)}\right\};\; x \in \mathcal{X};\; y \in \mathcal{Y};\, |\mathcal{Y}| \sim 100M$$

$$\phi_\theta(x,\, y) = \langle f_\theta(x),\, g_\theta(y)\rangle$$

- training data goes from current undergoing systems that can cause bias towards popular items.
- some high-quality items can be under-explored in the training data and an algorithm trained via MLE will continue to under-estimate the relevance of the under-explored items in order to faithfully fit the observed data.
- there is a problem of too many items

$$\arg\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} -\log p_\theta(y|x),\; where$$

$$p_\theta(y|\, x) = \frac{\exp \phi_\theta(x,\, y)}{\sum_{y'\in\mathcal{Y}} \exp \phi_\theta(x,\, y')}$$

Sampling

$$\{y_i\}_{i=1}^{L} \sim p_n(y|\mathrm{x})$$

- Negative Sampling[1]

$$\arg\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,\, y)\, \in \mathcal{D}} \left\{ \log \sigma(\phi_\theta(x,\, y)) + \frac{1}{L} \sum_{i=1}^{L} \log \sigma(-\phi_\theta(x,\, y_i)) \right\}$$

- Sampled Softmax[2]

$$\arg\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,\, y)\, \in \mathcal{D}} -\log \left\{ \frac{exp(\phi_\theta(x, y) - \log p_n(y|x))}{exp(\phi_\theta(x, y) - \log p_n(y|x)) + \sum_{i=1}^{L} exp(\phi_\theta(x, y_i) - \log p_n(y_i|x))} \right\}$$

- Sampled softmax in general outperforms other approximations such as negative sampling when the vocabulary is large

- Most implementations assume $p_n(y \mid x) = p_n(y)$ and set $p_n(y)$ somehow proportional to the popularity of the items to improve convergence.

1: Distributed Representations of Words and Phrases and their Compositionality

2: Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model

Sampled Softmax

$$P(y|x) \;=\; \exp\phi_\theta(x, y) \,/\, Z; \quad Z \;=\; \sum_{i=1}^{|\mathcal{Y}|} \exp\phi_\theta(x, y_i);$$

- Importance Sampling:

$$\Rightarrow \nabla \log P(y|x) \;=\; \nabla\phi_\theta(x,\, y) \;-\; \underbrace{\sum_{i\,=\,1}^{|\mathcal{Y}|} P(y_i|x)\nabla\phi_\theta(x,\, y_i)}_{to\ estimate} \;\simeq\; \nabla\phi_\theta(x,\, y) \;-\; \frac{1}{L}\sum_{i\,=\,1}^{L}\frac{P(y_i|x)}{p_n(y_i|\mathrm{x})}\nabla\phi_\theta(x,\, y_i)$$

- Self-Normalized Importance Sampling (biased):

$$w(y|\mathrm{x}) \;=\; Z \,\cdot\, P(y|\mathrm{x}) \,/\, p_n(y|\mathrm{x}) \;\Rightarrow\; \nabla \log P(y|\mathrm{x}) \;\simeq\; \nabla\phi_\theta(x,\, y) \;-\; \sum_{i=1}^{L}\frac{\nabla\phi_\theta(x,\, y_i) \,\cdot\, w(y_i|\mathrm{x})}{\sum_{j=1}^{L} w(y_j|\mathrm{x})}$$

$$\Rightarrow \nabla \log P(y|\mathrm{x}) \;\simeq\; \nabla\phi_\theta(x,\, y) \;-\; \sum_{i=1}^{L}\frac{\nabla\phi_\theta(x,\, y_i) \,\cdot\, \exp\left(\phi_\theta(x,\, y_i) \;-\; \log p_n(y_i|x)\right)}{\sum_{j=1}^{L}\exp\left(\phi_\theta(y_j|\mathrm{x}) \;-\; \log p_n(y_j|x)\right)}$$

1: [Monte Carlo theory, methods and examples: chapter 9]

2: [Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model]

Multinomial IPW Loss

$$\arg\min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} -\frac{1}{q(y \mid x)} \cdot \log p_{\theta}(y \mid x)$$

Derivation features:

- The previous works consider **bernoulli propensities** related with **users' attention**, i.e. whether a user notices a recommended item or not
- This recommender system is a sequential recommender system where a **user will receive only one recommendation** y when the user's state becomes x (it is easy to verify that conclusions still hold when user state x receives K recommendations)
- **Single user state** x for conciseness
- The **biased dataset Dπ** generation:
  - Policy π makes a recommendation, draws a **one-hot impression vector O** from the **multinomial distribution** $q\pi\ (y \mid x)$
  - user x is associated with a **multi-hot vector C** representing the user's preference regarding all the items, y-th element is drawn from the bernoulli distribution **p (click = 1 | x, y)**

Multinomial IPW Loss: derivation

- Training with unbiased data

$$p_{\pi_{\text{uni}}}(y|x) = \frac{p(\text{click}=1 \mid x, y)}{\sum_{y'} p(\text{click}=1 \mid x, y')}$$

$$R(\theta|x) = -\sum_y p_{\pi_{\text{uni}}}(y|x) \log p_\theta(y|x)$$

- Training with biased data

$$\hat{R}_{\text{naive}}(\theta|\mathcal{D}_\pi) = -\sum_y O_y C_y \log p_\theta(y|x)$$

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_\pi}\left[\hat{R}_{\text{naive}}(\theta|\mathcal{D}_\pi)\right] &= -\sum_y \mathbb{E}[O_y]\mathbb{E}[C_y] \log p_\theta(y \mid x) \\ &= -\sum_y q_\pi(y \mid x) p(\text{click}=1 \mid x, y) \log p_\theta(y \mid x)\end{aligned}$$

- IPW Loss

$$\hat{R}_{\text{IPW}}(\theta|\mathcal{D}_\pi) = -\sum_y \frac{1}{q(y \mid x)} O_y C_y \log p_\theta(y \mid x)$$

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_\pi}\left[\hat{R}_{\text{IPW}}(\theta|\mathcal{D}_\pi)\right] &= -\sum_y \frac{q_\pi(y \mid x)}{q(y \mid x)} p(\text{click}=1 \mid x, y) \log p_\theta(y \mid x) \\ &\propto -\sum_y \frac{q_\pi(y \mid x)}{q(y \mid x)} p_{\pi_{\text{uni}}}(y \mid x) \log p_\theta(y \mid x).\end{aligned}$$

IPW Loss: Multinomial or Bernoulli?

- **Multivariate Bernoulli**: O are drawn from a multivariate bernoulli policy, i.e., Oy follows an independent bernoulli distribution $q\pi$ (recommend = 1 | x, y).
- **Multinomial**: **empirical** results that report **better performance** with a multinomial candidate generation method
- **Multivariate Bernoulli**: formulation **implicitly assumes that the number of recommendations** requested by a user can be modeled as part of the recommendation policy π, the expectation of the number of recommendations is: $\sum_y q_\pi(\text{recommend} = 1|\text{x,y})$

  In many systems the number of recommendations received by a user is decided by the user rather than the system: the user can request more recommendations by scrolling down the page or stop receiving any new recommendation by leaving the page.

- **Multinomial**: $q\ \pi\ (y \mid x)$ only models which one item it should recommend if the **user explicitly requests the system to make one recommendation**.

Contrastive Loss

- doesn't correct the bias introduced by sampling
- the contrastive loss is in principle optimizing the same objective as IPW Loss, and CL-REC is a simple implementation that doesn't require two separate steps and can **avoid the instability brought by the division** of $q(y \mid x)$.

THEOREM 1. *The optimal solutions of the contrastive loss (Eq. 3) and the IPW loss (Eq. 4) both minimize the KL divergence from* $p_\theta(y \mid x)$ *to* $r(y \mid x) = \frac{p_{\text{data}}(y|x)/q(y|x)}{\sum_{y' \in \mathcal{Y}} p_{\text{data}}(y'|x)/q(y'|x)}$*, if* $p_n(y \mid x)$ *is set to be* $q(y \mid x)$. *Here* $p_{\text{data}}(y \mid x)$ *is the data distribution, i.e. what is the frequency of* y *appearing in* $\mathcal{D}$ *given context* x.

$$\boxed{\arg\min_\theta \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} -\log \frac{\exp\left(\phi_\theta(x,y)\right)}{\exp\left(\phi_\theta(x,y)\right) + \sum_{i=1}^{L} \exp\left(\phi_\theta(x,y_i)\right)}}$$

Proof: IPW

- proof for a single training user's state *x*

$$
-\sum_{y:(x,y)\in\mathcal{D}} \frac{\log p_\theta(y \mid x)}{q(y \mid x)} \propto -\sum_{y\in\mathcal{Y}} \frac{p_{\mathrm{data}}(y \mid x)}{q(y \mid x)} \log p_\theta(y \mid x)
$$

$$
\propto -\sum_{y\in\mathcal{Y}} r(y \mid x) \log p_\theta(y \mid x) = D_{\mathrm{KL}}(r \| p_\theta) + \mathrm{const.w.r.t.}\ \theta.
$$

Proof: Contrastive Loss

- proof for a single training user's state *x*

- Single sample (x, y). Let $C = \{y\} \cup \{y_i\}^{L}_{i=1}$ be the multi-set: L negative samples drawn from $q(y \mid x)$

- $q(C \mid x, y) = \prod_{i=1}^{L} q(y_i \mid x)$, if y in C, else q(C | x, y) = 0, then contrastive loss:

$$
\begin{aligned}
&- \sum_{y:(x,y)\in\mathcal{D}} \sum_{C} q(C \mid x, y) \log \frac{\exp\left(\phi_\theta(x, y)\right)}{\sum_{y' \in C} \exp\left(\phi_\theta(x, y')\right)} \\
\propto &- \sum_{y\in\mathcal{Y}} \sum_{C} q(C \mid x, y) p_{data}(y \mid x) \log \frac{\exp\left(\phi_\theta(x, y)\right)}{\sum_{y' \in C} \exp\left(\phi_\theta(x, y')\right)}
\end{aligned}
$$

Proof: Contrastive Loss

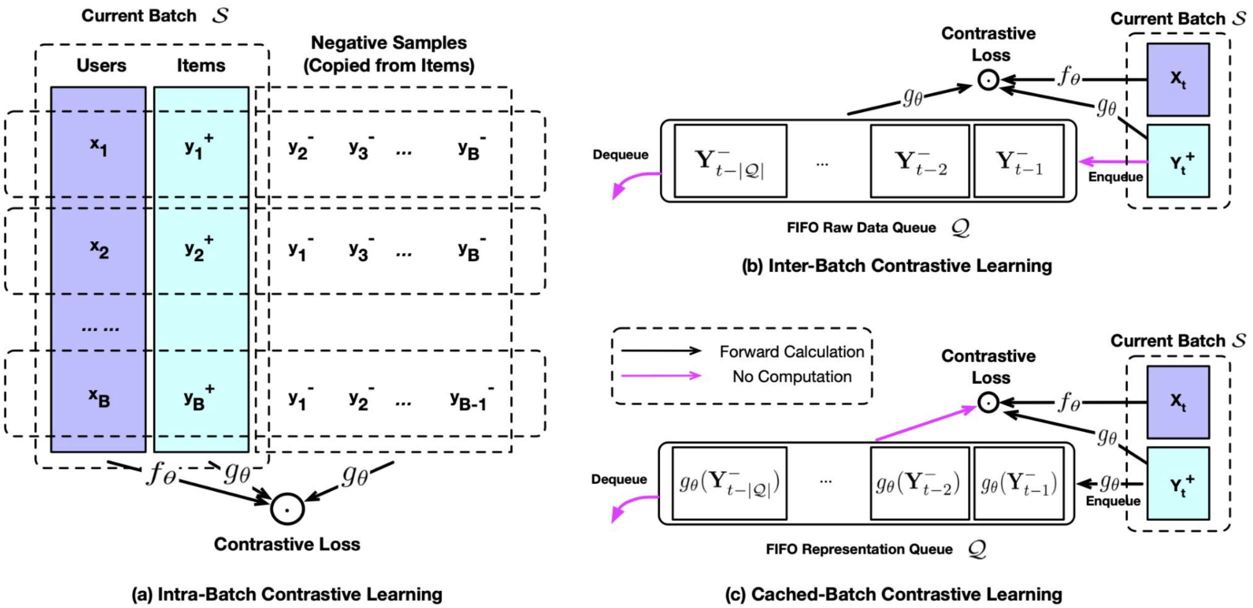
- Let $q(C \mid x) = \prod_{y' \in C} q(y' \mid x)$, then $q(C|\,\mathrm{x,y}) = \frac{q(C|\mathrm{x})}{q(y|\mathrm{x})}$ if C includes y and loss is proportional to:

$$
\begin{aligned}
& -\sum_{y \in \mathcal{Y}} \sum_{C: y \in C} \frac{q(C \mid x)}{q(y \mid x)} p_{data}(y \mid x) \log \frac{\exp\left(\phi_\theta(x, y)\right)}{\sum_{y' \in C} \exp\left(\phi_\theta(x, y')\right)} \\
=& \mathbb{E}_{q(C|x)}\left[-\sum_{y \in C} \frac{p_{data}(y \mid x)}{q(y \mid x)} \log \frac{\exp\left(\phi_\theta(x, y)\right)}{\sum_{y' \in C} \exp\left(\phi_\theta(x, y')\right)}\right] \\
=& \mathbb{E}_{q(C|x)}\left[D_{\mathrm{KL}}(r^C \| p_\theta^C)\right] + \text{const.w.r.t. } \theta.
\end{aligned}
$$

- distributions are based on C, but since we are minimizing the KL divergence under all possible C **the global optima** will be the ones that make $p\theta\ (y \mid x)$ equal to $r\ (y \mid x)$ for all y

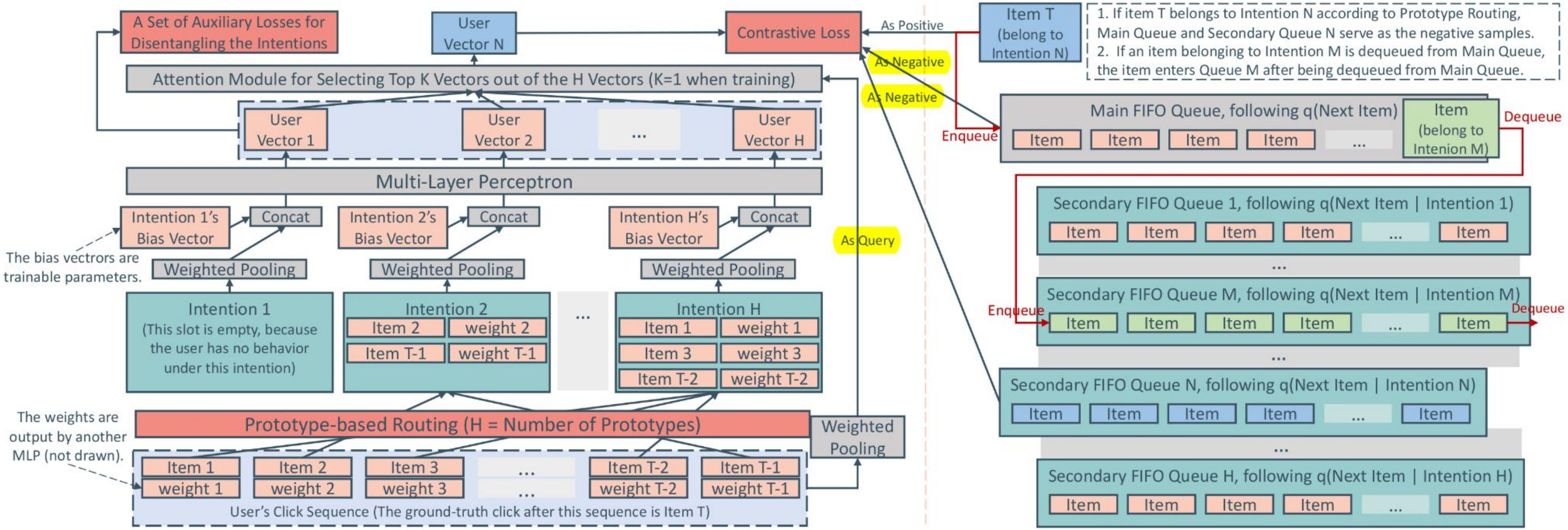
$$
r^C(y|x) = \frac{p_{\mathrm{data}}(y|x)/q(y|x)}{\sum_{y' \in C} p_{\mathrm{data}}(y'|x)/q(y'|x)} \text{ and } p_\theta^C(y|x) = \frac{\exp(\phi_\theta(x,y))}{\sum_{y' \in C} \exp(\phi_\theta(x,y'))}
$$

CL-REC



- The exact $q(y \mid x)$ is hard to estimate: there are many complicated modules
- In CL-REC: $\boldsymbol{q(y \mid x) \approx q(y)}$, where $q(y)$ probability that **item y is recommended to someone**
- **$q(y)$ has a high correlation with $p_{\text{data}}(y)$**, (probability that item y is being recommended and clicked by someone), as the existing system is already highly optimized.
- In CL-REC: $\boldsymbol{q(y) \approx p_{\text{data}}(y)}$
- **Cached version: no longer back-propagate** through the negative examples from **the previous batches,** only through examples from the present batch

Multi CL-REC



- **Improve** upon CLRec more accurate propensity score **$q(y$ | user x's intent)**: intent is a cluster of categories user can be interested in.
- Multi-CLRec uses H queues $\{Q_h\}_{h=1}$ corresponding to intentions. Here H = 64
- There is implicit bias reduction based on a **propensity score $\propto q(y$ | user x's intention is $h) + \alpha \cdot q(y)$;** $\alpha > 0$, is the smoothing factor
- Negative samples go from a **secondary queue Q_h** and the **main queue Q_0**.

Multi CL-REC: architecture

- **Item Encoder**

$$
\begin{aligned}
g_\theta(y_t) = \texttt{MLP}_1([\ & \text{embedding of item } y_t\text{'s unique ID;} \\
& \text{embedding of item } y_t\text{'s category ID, i.e., } c_t; \\
& \text{embedding of item } y_t\text{'s seller ID;} \\
& \text{embeddings of } y_t\text{'s tags;} \quad \ldots]),
\end{aligned}
$$

- **μ_h**- trainable vectors to represent intention prototypes, needed for routing each item to intention with some probability

$$
p_{h|t} = \frac{\exp(p'_{h|t})}{\sum_{h'=1}^{H} \exp(p'_{h'|t})}, \quad \text{where } p'_{h|t} = \frac{\langle \boldsymbol{\mu}_h, c_t \rangle}{\rho \cdot \|\boldsymbol{\mu}_h\| \cdot \|c_t\|}
$$

- MLP to model the importance of each clicked item in the user's **history** (for user encoder):

$$
p_t = \frac{\exp(p'_t)}{\sum_{t'=1}^{T-1} \exp(p'_{t'})}, \quad \text{where} \quad
\begin{aligned}
p'_t = \texttt{MLP}_2([\ & \text{item embedding } g_\theta(y_t)\text{; category embedding } c_t; \\
& \text{time gap between clicking item } y_t \text{ and item } y_T; \\
& \text{user's dwell time on item } y_t; \quad \ldots]) \text{ and } p'_t \in \mathbb{R}.
\end{aligned}
$$

Multi CL-REC: architecture

- **MLP of Weighted Pooling and bias**, to obtain H intention vectors for user. Each element of history is routed.

$$\mathrm{z}_h = \mathtt{MLP}_3([\mathrm{z}'_h; \boldsymbol{\beta}_h]), \quad \mathrm{z}'_h = \sum_{t=1}^{T-1} p_{h|t} \cdot p_t \cdot \mathrm{g}_\theta(y_t)$$

- **Attention Module**: m (user history) as Query

$$\mathrm{f}_\theta(x) = \mathrm{z}_{h^*}, \quad \text{where } h^* = \underset{h \in \{1,2,\ldots,H\}}{\arg\max}\ w_h, \quad w_h = \frac{\exp(\tilde{w}_h)}{\sum_{h'=1}^{H} \exp(\tilde{w}_{h'})},$$

$$\tilde{w}_h = \frac{\langle \mathrm{m}, \mathrm{z}_h \rangle}{\rho \cdot \|\mathrm{m}\| \cdot \|\mathrm{z}_h\|}, \quad \mathrm{m} = \mathtt{MLP}_4\left(\sum_{t=1}^{T-1} p_t \cdot \mathrm{g}_\theta(y_t)\right).$$

- **Straight through argmax approach** (instead of h^*):

$$\ddot{w}_h = \text{stop\_gradient}(I[h = h^*] - w_h) + w_h$$

Multi CL-REC: losses

- Contrastive: positive is enqueued in main queue; the object deued from it routed to intention queue by $p_{h|y}$

$$\mathcal{L}_{cl} = -\log \sum_{h=1}^{H} \ddot{w}_h \cdot \frac{\exp\{\cos(z_h, g_\theta(y_T))/\rho\}}{\sum_{h'}^{H} \sum_{y \in Q_0 \cup Q_{h^+}} \exp\{\cos(z_{h'}, g_\theta(y'))/\rho\}},$$

$$\text{where } h^+ = \arg\max_h p_{h|\mathrm{T}}, \quad p_{h|y^-} = \arg\max_h \frac{\langle \mu_h, c^- \rangle}{\rho \cdot \|\mu_h\| \cdot \|c^-\|}$$

- Auxiliary
  - to make routing polarized $p_{h^+|T} \to 1$ and $p_{h'|T} \to 0$ for $h' \neq h^+$:

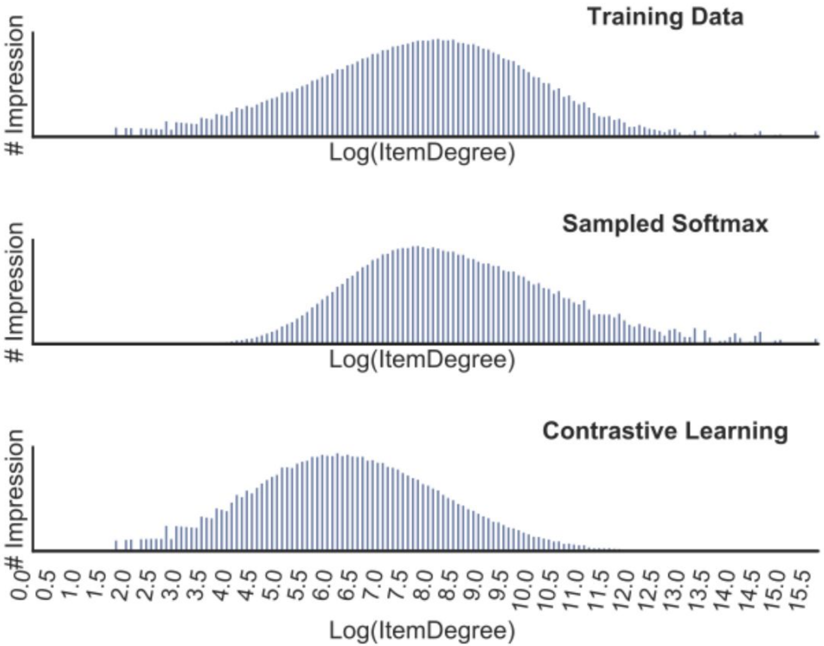
$$\boxed{\mathcal{L}_{\mathrm{aux},1} = -\log p_{h^+|T}, \quad \text{where } h^+ = \arg\max_{h \in \{1,2,\ldots,H\}} p_{h|T}}$$

  - to push the correct intention vector closer to the target item:

$$\boxed{\mathcal{L}_{\mathrm{aux},2} = -\log \frac{\exp(s_{h^+,T})}{\sum_{h'=1}^{H} \exp(s_{h',T})}, \quad s_{h,T} = \frac{\langle \mathrm{z}_h, \mathrm{g}_\theta(y_T) \rangle}{\rho \cdot \|\mathrm{z}_h\| \cdot \|\mathrm{g}_\theta(y_T)\|}}$$

  - to balance intentions by number of items : $\boxed{\mathcal{L}_{\mathrm{aux},3} = \sum_{h=1}^{H} \frac{1}{H} \cdot (\log \frac{1}{H} - \log \pi_h), \quad \pi_h = \mathbb{E}_{\mathcal{B}}[p_{h|t}]}$

Experiments: diversity



	Aggregated Diversity
sampled-softmax	10,780,111
CLRec	21,905,318

- **ItemDegree (popularity)**: the rightmost bar is not the highest because the **number of the extremely popular items is small**, even though each item in the bucket has a very high degree.
- **Diversity:** the number of **distinct items** recommended to a randomly sampled subset of users

Experiments

Method	HR@50	CTR(online)
negative sampling	7.1%	outdated
shared negative sampling	6.4%	-
sampled-softmax	17.6%	3.32%
CLRec	17.8%	3.85%

Method	CTR	Average Dwell Time	Popularity Index
MIND	5.87%	-	0.658
CLRec	6.30%	+11.9%	0.224

Method	Train on Weekdays' Data	
	Test on Weekdays	Test on Weekends
CLRec	17.18%	17.16%
Multi-CLRec	17.25%	17.68%

- **HR@50** (offline) - the percentage of times clicked item appeared in 50 kNN generated candidates for sampled user sequences
- **CTR** (online) - the percentage of recommendations made by the algorithm that are finally clicked by the users
- **Dwell Time** (online) - is the average time spent by the users on reading the details of the items clicked by them
- **Popularity Index** - how much an algorithm prefers recommending the items that are already popular:

$$\frac{\mathbb{E}_A[p_{\text{data}}(y)]}{\max_{A'} \mathbb{E}_{A'}[p_{\text{data}}(y)]}$$

$$\mathbb{E}_A[p_{\text{data}}(y)] = \sum_{y \in \mathcal{Y}} p_{\text{data}}(y) \cdot p(A \text{ recommends item y})$$

user encoder can select and output top-K vectors at serving time

(0) The user's latest eight clicks.

(1) Top items retrived by vector 1.

(2) Top items retrived by vector 2.

(3) Top items retrived by vector 3.

(4) Top items retrived by vector 4.

(5) Top items retrived by vector 5.

(6) Top items retrived by vector 6.

(7) Top items retrived by vector 7.

Thank you for your attention