(2025.11.26) OneRec: Unifying Retrieve and Rank with Generative Recommender and Preference Alignment

연구 배경

"Retrieval" → "Ranking" N 단계로 이루어진 기존의 추천 시스템(cascade ranking)의 구조적 한계

1. 단계 별 목표의 충돌: 기존 추천 시스템은 여러 단계로 나뉘어 있으며, 각 단계는 다른 구조와 크기의 모델을 사용합니다. 이러한 구조는 유사한 목표를 모델링할 때조차 충돌을 야기할 수 있다고 합니다.
   예를 들어, 검색(retrieval) 단계의 효율성은 랭킹(ranking) 모델의 제약에 의해 제한될 수 있으며, 랭킹 모델 또한 최적화 되지 않은 업스트림 결과에 영향을 받을 수 있습니다.
2. 낮은 계산 효율성: 자체 연구에 따르면 50% 이상의 리소스가 실질적인 계산이 아닌 저장과 모델간의 통신에 낭비되고 있음을 확인했습니다. GPU는 계산이 아닌 데이터 로딩과 네트워크 병목에 취약 합니다.
3. AI 트렌드와 멀어짐: 1의 목표 충돌이 기존 추천시스템의 LLM 도입 및 활용도를 제한하고 있다고 주장합니다.

Generative Recommendation (GRs)의 발전

GENRE, DSI, NCI, CGR, TIGER 등과 같은 GR 모델들의 발전

선행 연구

GRs

기존의 추천 시스템 방식이 주어진 쿼리(query)에 대해 해당 쿼리의 임베딩(embedding)을 얻은 다음, ANN 등의 유사도 알고리즘을 사용하여 아이템을 검색(retrieval)하는 방식이라면, GR은 주로 트랜스포머 구조를 사용해 e2e로 후보 아이템의 ID를 직접 생성해 내는 방식입니다.
관련 선행연구를 정리합니다.

1.GENRE

트랜스포머(Transformer) 기반 아키텍처를 사용하여 주어진 쿼리에서 참조하는 엔티티의 이름을 토큰 단위로 반환하는 방식으로 엔티티 검색 구조를 제안했습니다.

2.DSI

최초로 각 문서에 Semantic ID 를 할당한 시스템을 도입하고 역시 최초로 검색 애플리케이션에서 트랜스포머 기반 엔드투엔드(end-to-end) 방식을 사용했습니다.

3.TIGER

"Retrieval" 을 위해 오토인코더(RQ-VAE)를 사용하여 생성된 Semantic ID(generative Semantic IDs)를 활용한 최초의 연구 입니다.

논문 핵심

OneRec은 E2E GR 시스템으로 사용자의 과거 행동 데이터를 인코더로 학습하여 관심사를 파악하고, 대규모 MoE 기반 디코더를 통해 정확한 짧은 동영상 추천을 생성하는 추천 시스템 입니다.
그리고 모델 출력을 보상 함수(reward function)와 정렬(align)시켜 퀄리티를 높이기 위한 강화 학습 프레임워크를 제안합니다.

사용된 이론

RQ-Kmeans

Quantization에 필요한 Codebook을 잔차(residual)에 K-means clustering 하는 방식으로 생성하는 알고리즘 입니다.

$$\mathcal{R}^{(1)} = \left\{ \tilde{\mathbf{M}}_i \in \mathbb{R}^{N_{\tilde{M}} \times d_t} \mid \forall \text{ video } i \right\}$$

$$C^{(l)} = \text{K-means}(\mathcal{R}^{(l)}, N_t)$$

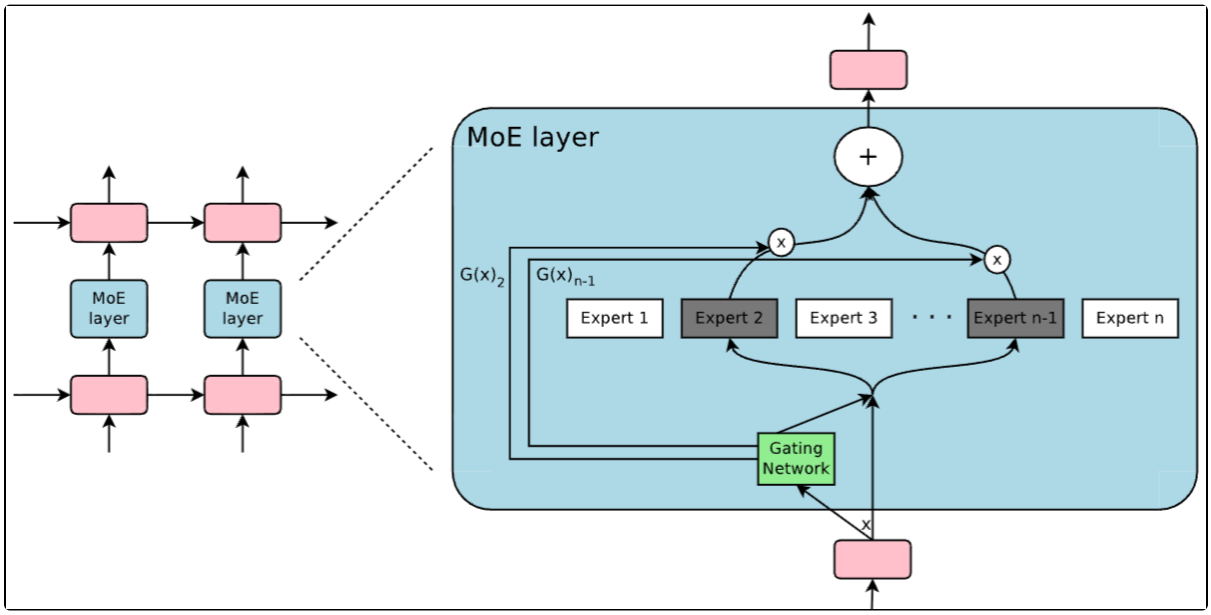
$$s_i^l = \arg\min_k \left\| \mathcal{R}_i^{(l)} - \boldsymbol{c}_k^{(l)} \right\|$$

$$\mathcal{R}_i^{(l+1)} = \mathcal{R}_i^{(l)} - \mathbf{c}_{s_i^l}^{(l)}$$

1. 최초 레이어의 잔차 R^(1)은 일반적인 임베딩입니다.
2. 해당 레이어의 코드북 C는 이 잔차 임베딩에 K-means 알고리즘을 적용, 찾아낸 N_t개의 centroid 들로 이루어 집니다.
3. 아이템 i에 대해 가장 가까운 centroid 의 index 를 찾습니다.
4. 아이템 i의 다음 레이어 잔차는 현재 레이어의 잔차에서 가장 가까운 centroid를 뺀 값으로 업데이트 됩니다. (이 수식은 집합에 속한 모든 아이템 원소에 대해 개별적으로 적용됩니다.)

RQ-Kmeans는 RQ-VAE에 비해 원본 입력 정보의 보존 능력, 코드북 활용률이 뛰어나고 토큰 분포가 더 균일하다고 합니다.

Mixture-of-Experts (MoEs)



MoE는 크게 두 가지 주요 요소로 구성됩니다.

1.Sparse MoE layers

Sparse MoE 레이어는 일반적으로 사용되는 dense한 FFN 레이어 대신 사용됩니다.
MoE 레이어는 특정 수의 "전문가(experts)"를 가지며, 각 전문가는 하나의 신경망입니다. 실제로는 이 전문가들이 FFN 이거나 심지어 MoE 자체가 될 수도 있어 계층적인 MoE를 형성할 수도 있습니다.

2.Gate Network (Router)

$$y = \sum_{i=1}^{n} G(x)_i \, E_i(x), \quad G_\sigma(x) = \text{Softmax}(x \cdot W_g)$$

어떤 토큰(입력 데이터 단위)을 어떤 전문가에게 보낼지 결정하는 역할을 합니다.

GRPO

GRPO는 PPO의 변형으로 그룹 단위로, PPO가 이전 policy 대비 신규 policy가 과도하게 벗어나지 않도록 제약 하여 안정적으로 개선할 수 있게 한 것을 적용하여, GRPO는 policy model의 G개 출력을 그룹으로 사용합니다.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{ \min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon \right)\hat{A}_{i,t} \right] \right\} - \beta\mathbb{D}_{\text{KL}}\left[\pi_\theta||\pi_{\text{ref}}\right] \right]$$

$$\mathbb{D}_{KL}[\pi_\theta||\pi_{ref}] = \frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - \log\frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - 1$$

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

1번째 수식

- O: policy model의 출력 시퀀스
- G: policy model의 응답 그룹
- Min[~]: PPO의 objective로 새로운 policy가 이전 policy보다 얼마나 달라졌는지, 달라짐 정도가 허용범위를 넘지 않도록 clip

2번째 수식

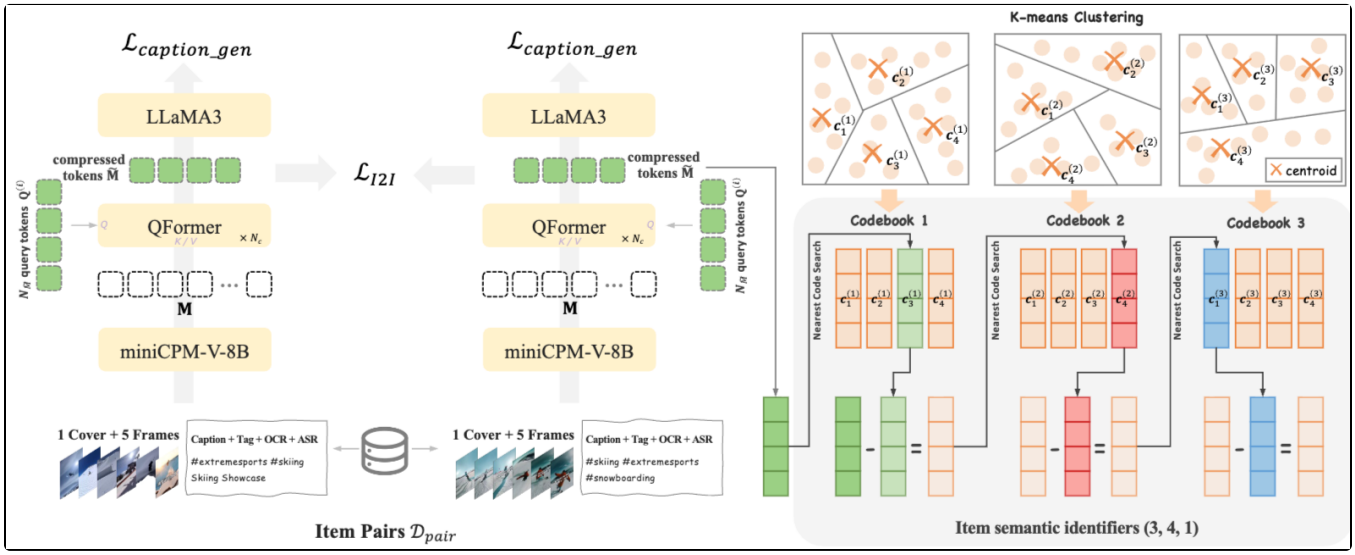
policy가 reference model 에서 많이 벗어나지 않도록 하는 penalty term → RL 과정에서 모델이 지나치게 reward aggressive 해지는 걸 방지합니다.

3번째 수식

Reward 모델이 policy model의 출력을 scoring 한 보상값 r_i 을 이용해서 바탕으로 정규화된 보상 = 어드밴티지

방법론

1. Tokenizer



왼쪽 (멀티모달 표현 생성)

$$\mathbf{Q}^{(i+1)} = \text{CrossAttn}(\mathbf{Q}^{(i)}, \mathbf{M}, \mathbf{M}),$$

$$\mathbf{Q}^{(i+1)} = \text{FFN}(\text{RMSNorm}(\mathbf{Q}^{(i+1)})), \text{ for } i \in \{1, 2, \ldots, N_c\}.$$

1 Cover + 5 Frames+ Caption + Tag + OCR(Image-to-Text) + ASR(Speech-to-Text) 로 이루어진 멀티모달 데이터 인풋을 sMLLM (miniCOM-V-8B)를 이용해 멀티모달 임베딩으로 만든 후 크로스 어텐션 + FFN(QFormer) 을 거쳐 보다 압축된 임베딩으로 만듭니다.

$$\mathcal{L}_{I2I} = -\frac{1}{|\mathcal{B}|}\sum_{(i,j)\in\mathcal{B}}\log\frac{\exp\left(\text{sim}\left(\tilde{\mathbf{M}}_i, \tilde{\mathbf{M}}_j\right)/\tau\right)}{\sum_{(i',j')\in\mathcal{B}}\exp\left(\text{sim}\left(\tilde{\mathbf{M}}_i, \tilde{\mathbf{M}}_{j'}\right)/\tau\right)}$$

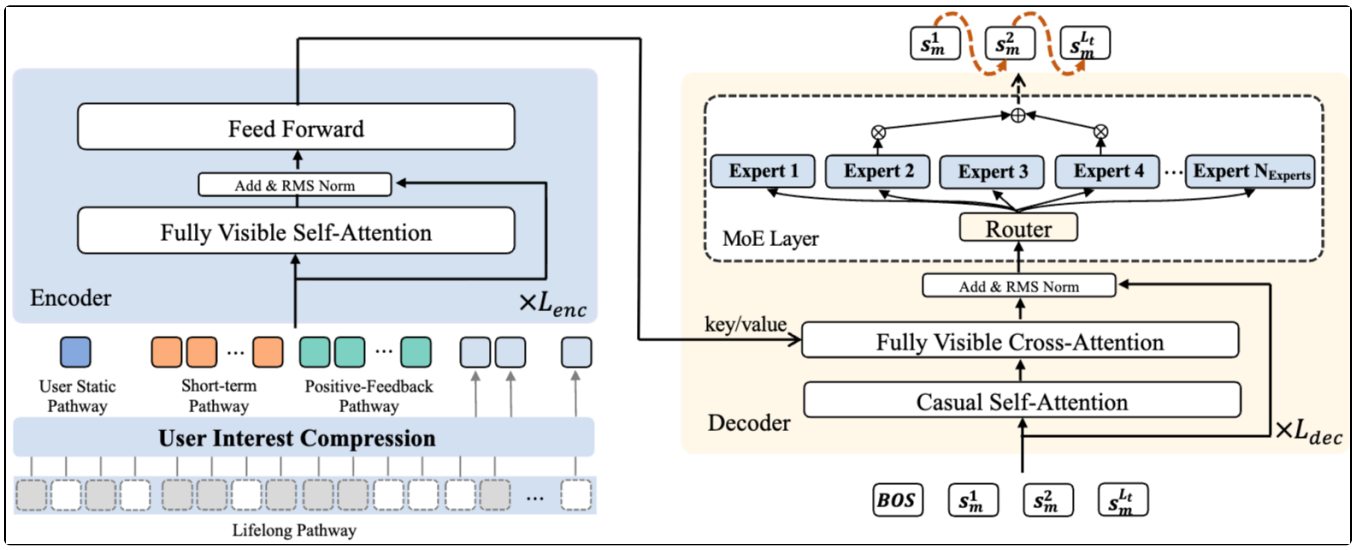
$$\mathcal{L}_{\text{caption\_gen}} = -\sum_{k} \log P(t^{k+1} | [t^1, t^2, \ldots, t^k])$$

압축된 임베딩은 대조학습을 통해 유사한 비디오 끼리 비슷한 임베딩으로 정렬되고, LLM(LLaMA3)를 통해 다음 비디오 캡션 토큰 예측을 하므로써 임베딩이 비디오에 대한 이해를 높이도록 유도됩니다.
조금 더 자세히 설명하면, 대조학습에서 사용하는 방식은 softmax와 유사하게 분자는 현재 아이템 기준 긍정으로 분류되는 아이템, 분모는 전체 아이템 셋으로 마찬가지로 negative log likelihood 을 취해 분자의 값을 키우거나 분모를 줄이는 방식으로 손실을 최소화 하도록 설계되어 있습니다.

오른쪽 (토큰화)

왼쪽의 과정으로 생성된 비디오 임베딩은 앞서 설명한 RQ-Kmeans 를 통해 Semantic ID로 변환됩니다.

2.Encoder & Decoder



Encoder

인코더에서는 4종류의 유저 행동 데이터를 인풋으로 표현 벡터(임베딩)를 산출합니다.

1. 정적 데이터: uuid, 성별, 나이 등의 정적인 데이터
2. 단기 흔적: 최근 interaction (20개) 비디오의 메타데이터 및 semantic id
3. 긍정 피드백 데이터: 유저가 긍정적으로 interaction한 비디오의 메타데이터 및 semantic id
4. 장기 흔적: 100,000개 이상의 초장기 시점에서 interaction 비디오들을 K-means로 클러스터링 한 후, 가장 가까운 클러스터의 대표 비디오의 메타데이터 및 semantic id
   장기 흔적 데이터의 경우 예외적으로 임베딩 이후 QFormer 과정을 거쳐 임베딩을 압축합니다.

$$\mathbf{f}_l = [\mathbf{e}^l_{\text{vid}}; \mathbf{e}^l_{\text{aid}}; \mathbf{e}^l_{\text{tag}}; \mathbf{e}^l_{\text{ts}}; \mathbf{e}^l_{\text{playtime}}; \mathbf{e}^l_{\text{dur}}; \mathbf{e}^l_{\text{label}}]$$

$$\mathbf{v}_l = \text{Dense}(\text{LeakyReLU}(\text{Dense}(\mathbf{f}_l)))$$

(정적 데이터, 단기 흔적, 긍정 피드백 데이터, 장기 흔적)

$$\mathbf{h}_l^{(i+1)} = \textbf{\{CrossAttn\}}(\mathbf{h}_l^{(i)}, \mathbf{v}_l, \mathbf{v}_l),$$

$$\mathbf{h}_l^{(i+1)} = \mathbf{FFN}(\mathbf{RMSNorm}(\mathbf{h}_l^{(i+1)}))$$

(장기 흔적)

$$\mathbf{z}^{(1)} = [\mathbf{h}_u; \mathbf{h}_s; \mathbf{h}_p; \mathbf{h}_l] + e_{\text{pos}}$$

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \text{SelfAttn}(\text{RMSNorm}(\mathbf{z}^{(i)}))$$

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i+1)} + \text{FFN}\left(\text{RMSNorm}\left(\mathbf{z}^{(i+1)}\right)\right)$$

4종류의 임베딩은 concat 후 self-attention + FFN을 거쳐 최종 인코더의 산출물이 됩니다.

Decoder

Decoder는 Encoder의 output을 이용해 사용자가 선호할 만한 비디오를 생성하는 역할을 합니다.

$$S_m = \left\{ s_{[\text{BOS}]}, s_m^1, s_m^2, \ldots, s_m^{L_l} \right\}$$

후보 비디오 m의 Semantic ID 시퀀스로 codebook의 element의 index 로 이루어져 있습니다. (e.g., (3,4,1))

$$\mathbf{d}_m^{(0)} = \text{Emb\_lookup}(S_m)$$

다음은 모델 내부에 저장된 임베딩 테이블에서 해당 인덱스에 매핑된 임베딩 벡터를 가져옵니다. 이것이 decoder의 첫 레이어가 됩니다.

$$\mathbf{d}_m^{(i+1)} = \mathbf{d}_m^{(i)} + \textbf{CausalSelfAttn}(\mathbf{d}_m^{(i)})$$

$$\mathbf{d}_m^{(i+1)} = \mathbf{d}_m^{(i+1)} + \text{CrossAttn}(\mathbf{d}_m^{(i+1)}, \mathbf{Z}_{\text{enc}}, \mathbf{Z}_{\text{enc}})$$

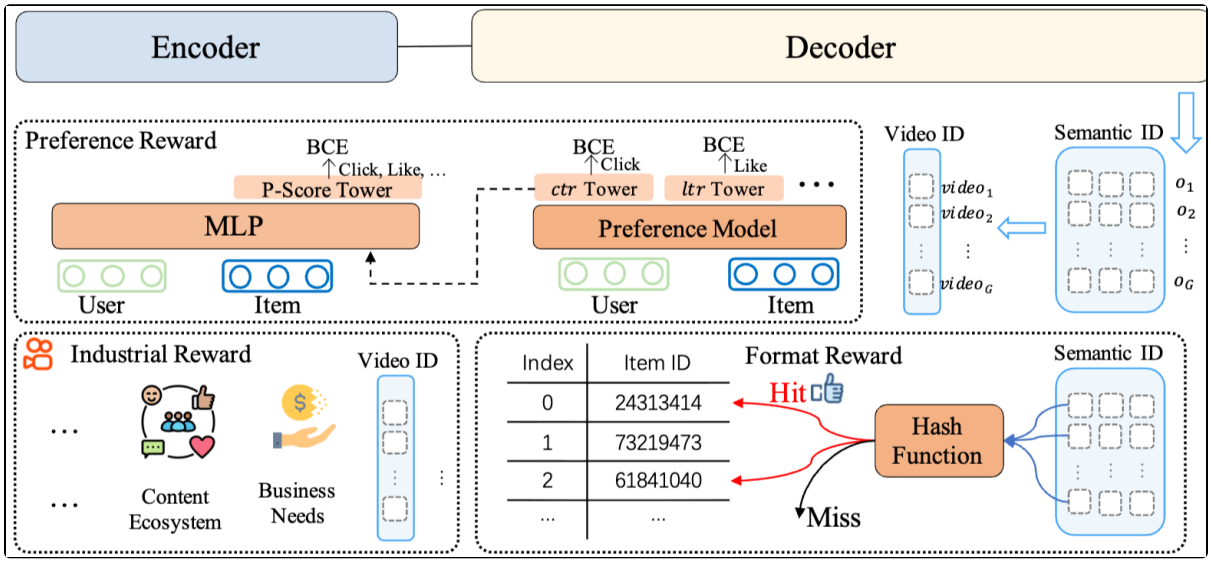
$$\mathbf{d}_m^{(i+1)} = \mathbf{d}_m^{(i+1)} + \text{MoE}(\text{RMSNorm}(\mathbf{d}_m^{(i+1)}))$$

Masked self-attention 후 encoder의 output을 key/value로 cross attention 합니다. 그리고 FFN으로 이루어진 MoE 레이어를 통과하게 됩니다.

$$\mathcal{L}_{\text{NTP}} = -\sum_{j=0}^{L_l - 1} \log P\left(s_m^{j+1} \mid \left[s_{[\text{BOS}]}, s_m^1, s_m^2, \ldots, s_m^j\right]\right)$$

expert의 산출물의 가중합으로 target 비디오의 semantic id를 예측합니다.

Reward Model



2의 Encoder-Decoder 모델은 지도학습 만으로 훈련되어 과거 데이터에 의존한 후보 예측만 가능한 상태입니다. 이 한계를 타파하기 위해 변형된 GRPO (=ECPO)를 프레임워크로 RL을 수행합니다.
ECPO 에서 사용하는 어드벤티지 (A)를 3가지 유형으로 바꿔가며 다양한 목표에 맞춰 정책 모델(2의 Encoder-Decoder 모델) 을 학습 시키게 됩니다.

그리고 **RL과 2번의 SL은 훈련 과정에서 연결**되어 일어난다고 합니다.

$$\mathcal{J}_{\text{ECPO}}(\theta) = \mathbb{E}_{u\sim P(U),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|u)}{\pi'_{\theta_{\text{old}}}(o_i|u)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|u)}{\pi'_{\theta_{\text{old}}}(o_i|u)}, 1-\epsilon, 1+\epsilon \right) A_i \right) \right]$$

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})}$$

$$\pi'_{\theta_{old}}(o_i|u) = \max\left(\frac{\text{sg}(\pi_\theta(o_i|u))}{1+\epsilon+\delta}, \pi_{\theta_{old}}(o_i|u) \right), \delta > 0$$

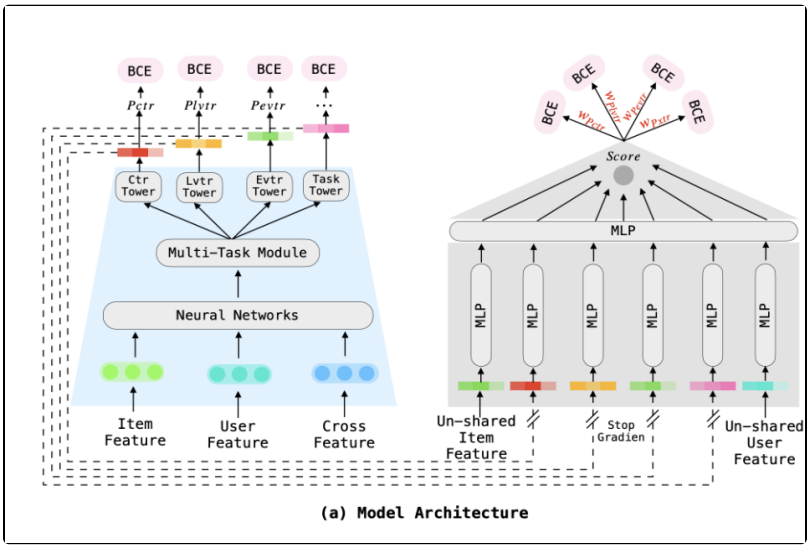
위 ECPO 수식을 살펴보면 1,2번 수식은 GRPO와 거의 동일합니다. 단지 신규 모델이 지나치게 reward aggressive 하는걸 막아주던 KL divergence term이 없어졌습니다.
이유는 RL학습 과 SL 학습이 연결되어 일어나기 때문에 SL의 NTP loss가 이 penalty term을 대체해 주기 때문이라고 하네요.

세번 째 수식을 보면 old policy를 old policy 와 new policy의 조정 값 중 더 큰 값으로 강제합니다. 이를 통해 ECPO의 PPO term 부분 (Min[~]) 의 분모가 지나치게 작아지는 걸 막아서, Gradient Explosion을 방지합니다.

그리고 이 프레임워크에서 사용하는 어드벤티지 A는 앞서 설명한 대로 총 3가지 유형입니다.

유저 선호도 (Preference Reward)

아이템에 대한 유저의 선호도, 즉 "좋은 추천" 이라는 걸 판명 하는 것은 어려운 일로 기존에는 좋아요, 코멘트, 클릭 등의 지표를 합산한 예측값 (xtr)을 점수로 사용했습니다.
다만 이 많은 지표들의 '합산'은 손으로 이루어지게 되고 이는 xtr의 정확도를 낮추고 있었다고 합니다. 때문에 유저선호도 어드벤티지 에서 사용하는 Pantheon 모델은 NN을 이용해서 다양한 목표를 잘 균형잡힌 하나의 스코어로 융합하는 역할을 합니다.



위 그림은 P-score 모델을 생성에 사용된 Pantheon 모델의 아키텍쳐 입니다. 왼쪽의 사이드 타워들은 랭킹 모델로 아이템, 유저, 아이템+유저 임베딩(Cross Feature)을 입력 받아 ctr, lvtr 등 각각 다른 목표로 학습되어 각자 다른 representation을 산출합니다. 그리고 오른쪽은 실질적인 'Pantheon' 모델로 ranking model 옆에 붙어서 이 타워들의 고차원 representation을 가져다 쓰는 부가 모듈로 볼 수 있습니다.
각각의 다른 표현을 concat 한 다음에 MLP 레이어들을 거친 후 sigmoid로 0~1 분포의 스코어를 만드는 간단한 구조이지만, 핵심은 각 표현의 합을 구성할때 어떤 weight를 선택할 지 입니다.

Pantheon

원래 Pantheon은 weight selection 을 policy로 보고 각 weight 조합을 action 으로 보고 RL을 수행하며 최적의 weight 를 찾아냅니다.
Weight 조합을 샘플링 한 후 A/B 테스트를 통해 reference model 보다 좋은 아웃풋을 가져왔는지를 기준으로 (T/F) Reward를 주는 방식입니다.
(만약 reference model을 뛰어넘었다면 reference model을 교체하거나, weight를 조정합니다.)

OneRec

$$\mathcal{L}_{\text{P-Score}} = \sum_{xtr \in S_o} w^{\text{xtr}} \mathcal{L}^{\text{xtr}}_{\text{P-Score}}$$

$$\mathcal{L}^{\text{xtr}}_{\text{P-Score}} = -(y^{\text{xtr}} \log p + (1 - y^{\text{xtr}}) \log(1 - p))$$

$$S_o = \{\text{ctr}, \text{lvtr}, \text{ltr}, \text{vtr}, \ldots\}$$

반면에 OneRec에서는 y^xtr 을 이용해서 지도학습을 통해 weight를 조정해 나가는 방식을 취하고 있습니다. y^xtr은 weighted fusion 되지 않은 raw label 인 것 같네요. (y^ctr = 0/1, y^ltr = 0/1 , ...)

최종적으로 산출된 P-score를 reward r_i로 Advantage에 사용해서 RL을 진행하게 됩니다.

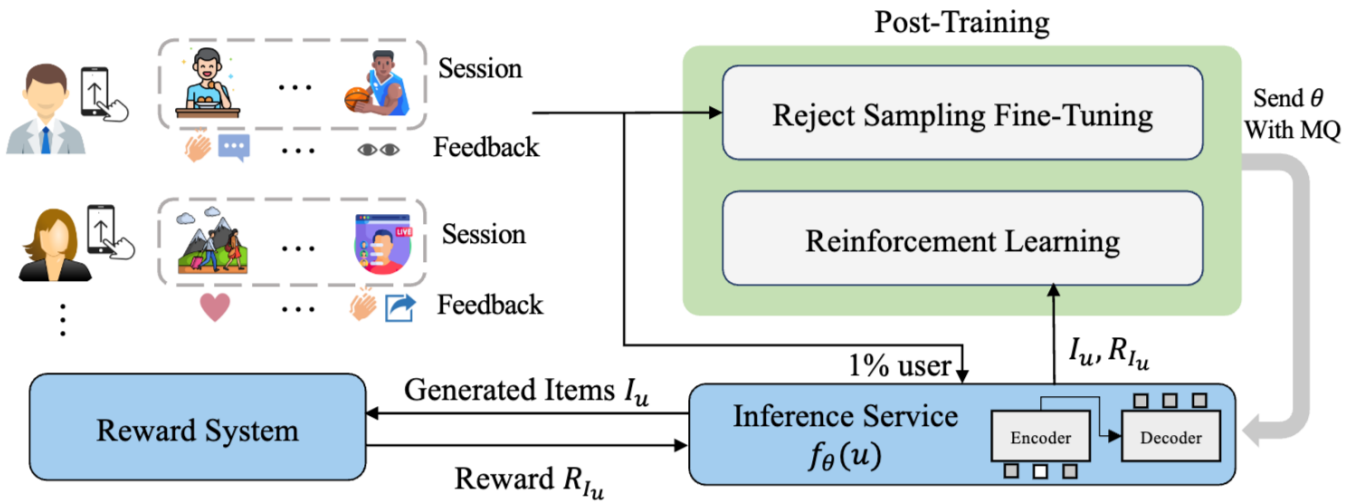
Format Reward

RL을 하면 음의 A (A<0) 에 민감하게 반응해 합법적인 semantic ID(실제 아이템과 맵핑 가능)와 불법적인 semantic ID(맵핑되는 아이템이 존재X) 구분이 모호해 지는 squeezing effect가 발생합니다.
이를 해결하기 위해 랜덤하게 샘플링을 해서 실제로 존재하는 합법 semantic ID인지 확인해서 보상(1/0)을 주는 방식으로 어드밴티지 A 를 구성해서 위 ECPO에 대입해 다시 학습을 진행합니다.

Industrial Reward

보상 함수를 특정 산업 시나리오의 요구사항에 맞춰 그때그때 다르게 구성하거나 조절하는 영역으로 보여집니다. 비디오 커뮤니티 생태계의 건강, 상업적 목표 달성, 바이럴 방지 등의 목적으로 A 함수를 구성할 수 있음을 예로 듭니다.

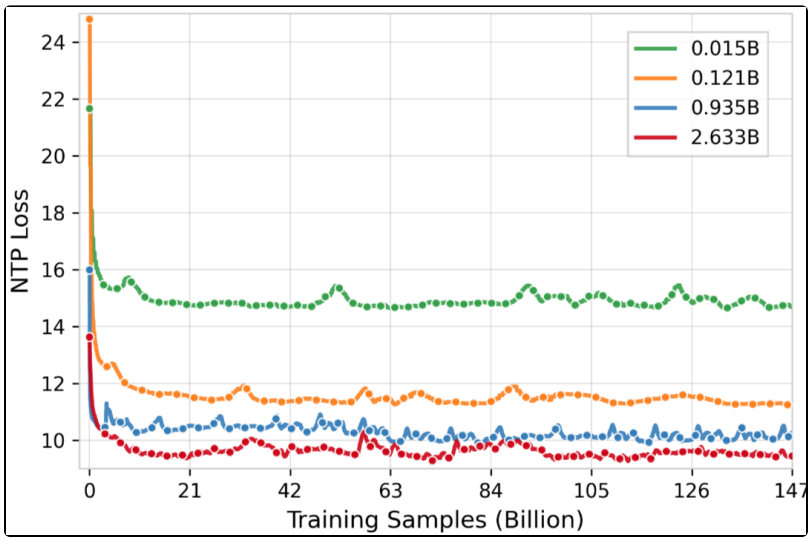
학습



실시간으로 예측에 대한 유저의 피드백을 받고 이를 바탕으로 NTP 손실을 최소화 하는 방향으로 모델을 학습(RSFT)시키면서 동시에 1%정도의 피드백을 샘플로 리워드를 추정해 RL도 같이 진행합니다.
그리고 RSFT와 RL을 통해 학습된 최신 모델 파라미터로 업데이트하여 실시간 추론 서비스에 반영됩니다.

실험 결과 및 평가

Scaling



학습 데이터를 늘릴 수록 NTP Loss가 줄어드는데 0.935B 모델 에서 최적의 효율을 보이는 걸로 나타났습니다.

RL

	Method	vtr	Watch time	App Stay Time	Video View[1]
Pass@32	OneRec w/o RL	0.1978	+1.62%	-0.10%	-4.18%
	OneRec w/ RL	0.2138	+3.17%	+0.39%	-9.87%
	Relative Impr.	+8.08%	+1.55%	+0.49%↑↑↑	-3.69%
Pass@128	OneRec w/o RL	0.2239	+4.61%	+1.11%	-12.75%
	OneRec w/ RL	0.2387	+5.22%	+1.49%	-15.06%
	Relative Impr.	+6.61%	+1.53%	+0.38%↑↑	-2.65%
Pass@512	OneRec w/o RL	0.2444	+6.32%	+1.66%	-15.54%
	OneRec w/ RL	0.2494	+5.88%	+1.75%	-13.88%
	Relative Impr.	+2.05%	-0.41%	+0.09%↑	+1.97%

RL이 모델 퍼포먼스에 큰 영향을 준 것으로 나타났습니다. 한편, 최적의 vtr, Watch time, App Stay Time을 찾기위해 RL을 진행하였기에 Video View와 같은 다른 metric 들은 오히려 더 낮은 퍼포먼스를 보여 줍니다.

RQ-Kmeans

		RQ-VAE	RQ-Kmeans
Reconstruction Loss ↓		0.0548	**0.0410**
Codebook Utilization ↑	layer 1	**1.0000**	**1.0000**
	layer 2	0.9963	**1.0000**
	layer 3	0.9958	**1.0000**
Token Distribution Entropy ↑	layer 1	8.3892	**8.9191**
	layer 2	8.4805	**8.7770**
	layer 3	8.6037	**8.7276**

RQ-VAE에 비해 RQ-Kmeans가 정보 손실, codebook 활용도, 토큰의 분포 측면 모두에서 앞서는 성능을 보여주는 것으로 확인되었습니다.

한계

1. 현재는 LLM은 임베딩을 생성하는 도구로서만 기능하고 있으므로, LLM의 semantic(의미론적) 정보 처리 능력을 활용하여 시스템 전반의 수준을 높이는 시도가 필요
2. 보상 모델에만 의존하는 강화학습의 한계
3. 인코더-디코더 아키텍처에서 비효율적인 연산 자원 배분 문제. 전체 자원의 97.66%가 생성(generation)이 아니라 입력 시퀀스 인코딩(컨텍스트 인코딩)에 소비되고 있음

2번과 3번은 OneRec v2에서 개선 된 것으로 보입니다.

레퍼런스

1. https://arxiv.org/pdf/2502.18965 (오리지널 논문)
2. https://arxiv.org/pdf/2506.13695 (1에서 수정된 테크니컬 리포트)
3. https://arxiv.org/pdf/2508.20900 (v2 논문 - OneRec의 한계점 지적 + 개선점 제시)
4. https://papers.neurips.cc/paper_files/paper/2023/file/20dcab0f14046a5c6b02b61da9f13229-Paper-Conference.pdf (TIGER)
5. https://arxiv.org/pdf/1707.06347 (PPO)
6. https://arxiv.org/pdf/2402.03300 (DeepSeekMath - GRPO)
7. https://arxiv.org/pdf/2411.11739 (RQ-Kmeans)
8. https://arxiv.org/pdf/1701.06538 (MoE 참고 1)
9. https://arxiv.org/pdf/2101.03961 (MoE 참고 2)
10. https://arxiv.org/pdf/2505.13894 (Pantheon)