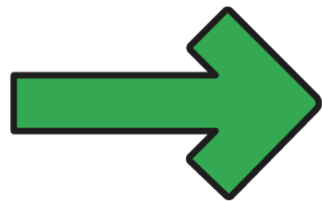




Google Developer Group

LLaRA: Large Language- Recommendation Assistant

대규모 언어 모델 추천 보조 시스템



ABSTRACT

초록



WHY?

- LLM을 이용한 기존의 연구 = ID 인덱스, 텍스트 메타데이터 사용 → 사용자 패턴 포착 x, LLM의 방대한 지식 최대한으로 사용 못함
- LLM(텍스트 데이터): 세계 지식 + 추론 능력
- 전통적인 추천 시스템(사용자 행동 데이터): 사용자 행동 패턴 포착 능력
- **LLM + 전통시스템 = LLaRA**

LLaRA: 전통적인 추천 시스템과 LLM의 장점을 동시에 살리기 위한 방법



연구 목적 및 핵심 아이디어

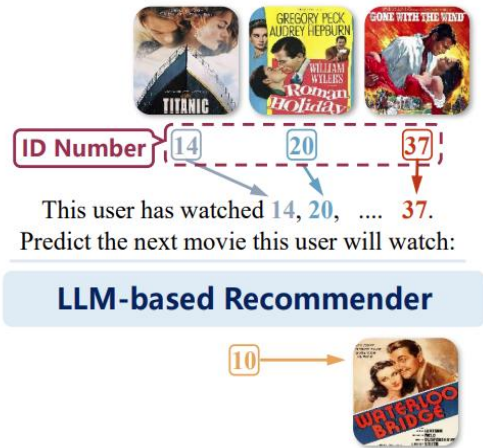
- 전통 추천 모델(Collaborative Filtering/CF) : ID 기반 아이템 임베딩(ex SASRec, Caser, GRU4Rec 등 등)
- LLM: 텍스트 기반 아이템 특징
- CF 모델의 임베딩 + 텍스트 임베딩 -> 하이브리드 프롬프트 방식
- ++커리큘럼 학습 전략 : 초기에는 텍스트 기반 -> 후기에 하이브리드 방식으로 점진적 훈련

INTRODUCTION

서론

✗ 기존의 연구

- 사용자의 행동 데이터 -> 텍스트 프롬프트로 변환: "이 사용자는 [item1], [item2], ..., [itemn]을 시청했습니다. 이 사용자가 다음에 시청할 영화를 예측하십시오."
- [itemk] -> ID기반표현: 텍스트적 특성X -> LLM의 지식 활용X
- [itemk] -> 텍스트기반표현: 사용자의 순차적 패턴 분석 미흡
- ⚠ ID 기반 또는 텍스트 기반만 사용하는 것은 LLM의 능력과 패턴을 동시에 반영하기 어려움



(c) Text Metadata



LLaRA 프레임워크

- CF 데이터: **사용자의 행동 패턴**과 아이템들의 특성을 간접적으로 학습
- CF 데이터 + 텍스트 데이터 -> 두 데이터를 결합한 **하이브리드 프롬프트** 구성
- **프로젝터(SR2LLM)**: CF모델의 임베딩값(50~200차원) -> LLM(768차원)으로 호환 가능하게 맵핑

Titanic = 텍스트 데이터

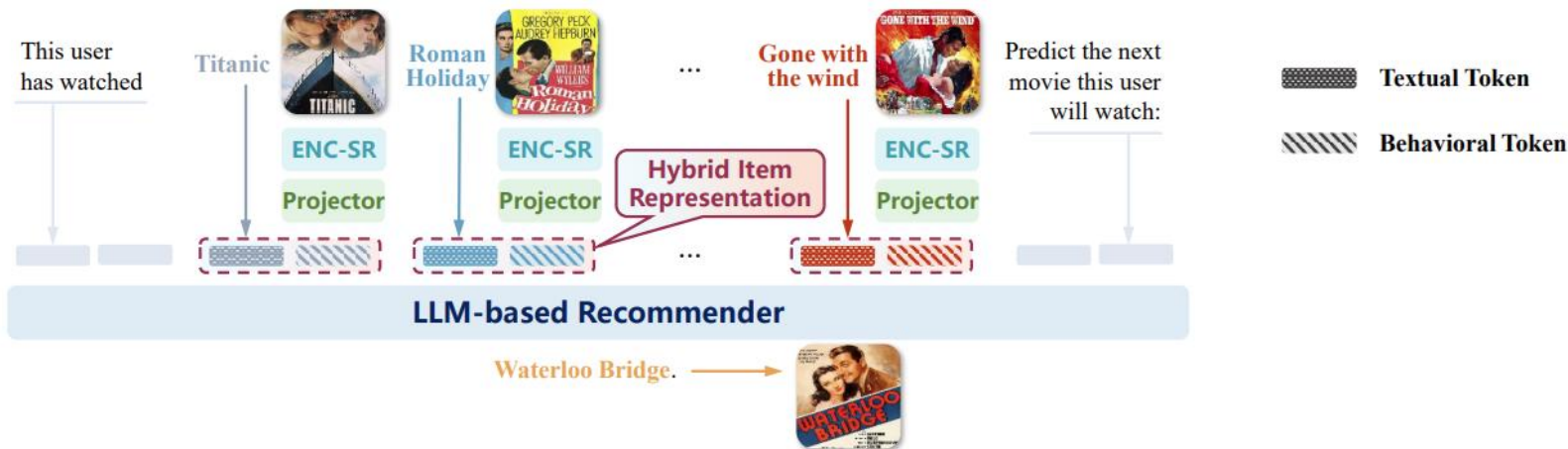
[14] = CF모델의 임베딩값



Item Representation Example

📌 하이브리드 프롬프트 설계

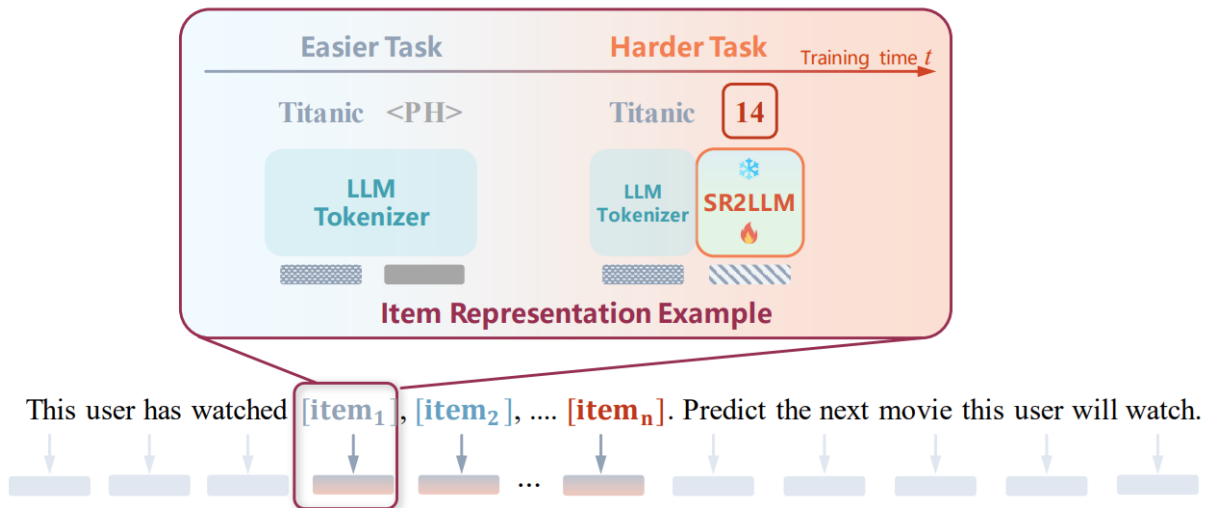
- 행동 패턴 정보 + 텍스트 표현 = 하이브리드 프롬프트
- 기존의 추천기의 CF 데이터 -> 프로젝트 -> 호환 가능한 행동 토큰(Behavioral Token)
- 이러한 방식으로 LLM이 사용자 행동 데이터를 이해하기 쉽도록 만듦





커리큘럼 프롬프트 튜닝

- 커리큘럼 프롬프트 튜닝: 텍스트 기반 프롬프트 -> 하이브리드 프롬프트로 진행되는 점진적 학습
- Easier Task: 단순 텍스트 데이터만 사용 -> LLM을 워밍업
- Harder Task: 텍스트 데이터 + 사용자 행동 데이터까지 사용 -> 순차 추천 능력 향상





실험 결과

- 데이터셋 : MovieLens(영화), Steam(비디오 게임), LastFM(음악)
- HitRatio@1 지표: LLaRA > 모든 기준 모델
- 하이브리드 프롬프트 + 커리큘럼 프롬프트 튜닝의 중요성 또한 검증됨

Item Representation

아이템 표현

Item Representation

- 추천시스템이나 LLM 모두 아이템(예: 영화, 책, 상품 등)을 입력으로 써야한다.
- 아이템을 어떻게 표현할지(벡터화 할지)가 중요하다.

1. Textual Token Representation

2. Behavioral Token Representation

Textual Token Representation

$$\langle emb \rangle_t^i = LLM - TKZ(txt_i)$$

txt_i : i번째 아이템의 텍스트
(예: 상품명, 설명)

$LLM - TKZ$: LLM의 토큰나이저 + 워드 임베딩 레이어.
(LLM에 맞게 텍스트 → 벡터 변환)

"apple" -> (tokenizer) -> id=845 -> (embedding layer) -> [0.123, -0.382, ..., 0.876]

⇒ 텍스트로부터 얻은 벡터는 언어모델의 언어 공간에 자연스럽게 들어갈 수 있으므로 LLM에 잘 맞는다.

Textual Token Representation

- Tokenize the data

Fine Tuning is fun for all!

[34389, 13932, 278, 318, 1257, 329, 477, 0]

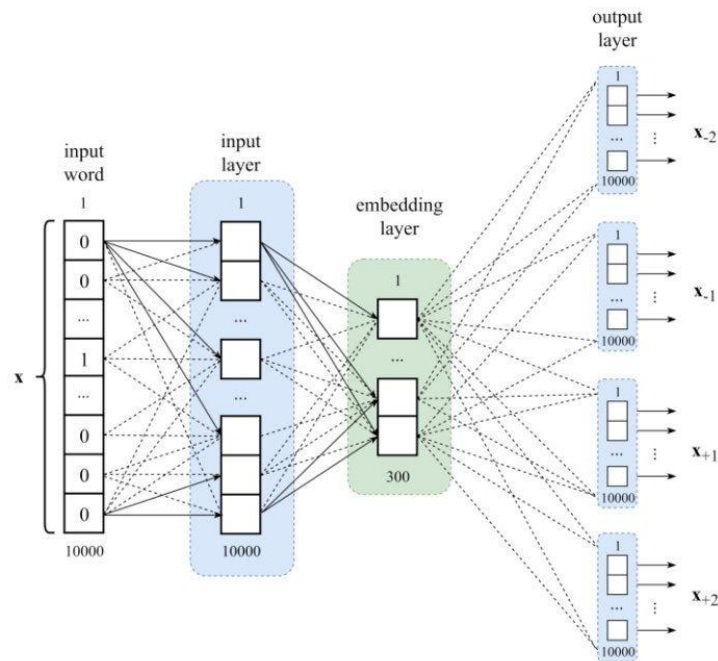
Fine Tuning is fun for all!

Encoding

Decoding

There are multiple popular tokenizers:

- Use the tokenizer associated with your model!



Behavioral Token Representation

$$e_s^i = SR - EMB(i; \Theta_e)$$

$SR - EMB$: 시퀀셜 추천모델(예: GRU4Rec, Caser, SASRec 등)이 만들어낸 아이템 임베딩 생성 함수

θ_e : 추천 모델의 파라미터

e_s^i : ID 기반(행동 기반) 임베딩, 차원 d 인 벡터

→ 추천시스템은 사용자-아이템 상호작용 데이터로부터 각 아이템의 행동 패턴을 임베딩에 녹여둔다. 하지만 이 임베딩은 LLM이 곧장 이해하기 어렵다.

→ ID 기반 임베딩은 언어모델이 쓰는 입력 값과 성격이 달라서 Modality Gap이 존재한다.

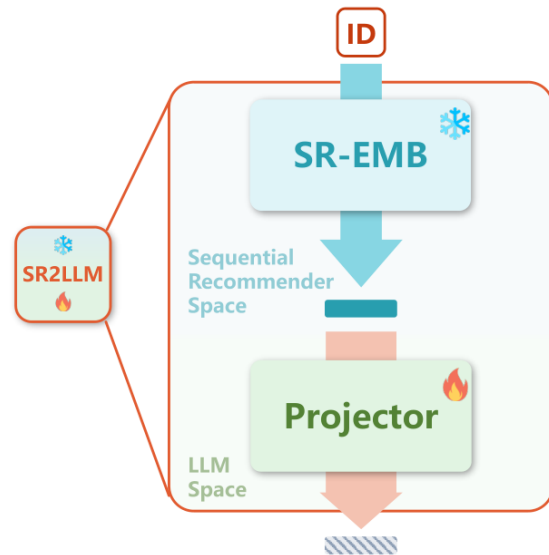
Behavioral Token Representation

- 행동 임베딩을 LLM에 바로 넣을 수 없다.
- SR - EMB를 통해 언어 공간에 맞게 변환(Projector 사용)

$$\langle emb_s^i \rangle = \mathbf{Proj}(e_s^i; \Theta_p)$$

$Proj$: 프로젝트 함수(예: 2-layer MLP 등),
 θ_p : 위 함수의 파라미터

- ID 기반 임베딩(행동 임베딩)을 LLM 언어 공간으로 mapping
- 이러면 LLM이 해석하기 용이하다.



(b) Architecture of SR2LLM.

Hybrid Token Representation

$$\langle emb_t^i \rangle + \langle emb_s^i \rangle \Rightarrow \langle emb_t^i \rangle$$

$$\langle emb_c^i \rangle = \mathbf{Concat}(\langle emb_t^i \rangle, \langle emb_s^i \rangle)$$

→ 텍스트 임베딩과 행동 임베딩을 합쳐서 하나의 임베딩 벡터로 만든다.

- 텍스트 정보와 행동 정보를 모두 가진 임베딩 생성
- 이렇게 하면 아이템의 언어적 의미와 행동적 특성을 동시에 반영할 수 있음.
- LLM이 추천을 더 똑똑하게 할 수 있게(둘 다 이해하게) 할 수 있다!

Hybrid Prompt Design

하이브리드 프롬프트 설계

Hybrid Prompt Design

Text-Only Prompting

Hybrid Prompting

과거 상호작용

선택할 후보군 (정답
+ 네거티브)

정답
(훈련/테스트)

(a) Text-only prompting method.

Input: This user has watched Titanic [PH], Roman Holiday [PH], ... Gone with the wind [PH] in the previous. Please predict the next movie this user will watch. The movie title candidates are The Wizard of Oz [PH], Braveheart [PH],..., Waterloo Bridge [PH],... Batman & Robin [PH]. Choose only one movie from the candidates. The answer is

Output: Waterloo Bridge.

아이템 이름 옆에 텍스트 토큰만 붙음
[PH]_placeholder

(b) Hybrid prompting method.

Input: This user has watched Titanic [emb_s^{14}], Roman Holiday [emb_s^{20}], ... Gone with the wind [emb_s^{37}] in the previous. Please predict the next movie this user will watch. The movie title candidates are The Wizard of Oz [emb_s^5], Braveheart [emb_s^{42}],..., Waterloo Bridge [emb_s^{20}],... Batman & Robin [emb_s^{19}]. Choose only one movie from the candidates. The answer is

Output: Waterloo Bridge.

+행동 임베딩 토큰
< emb_s^i >

•Text-only/Hybrid:

뒤에 붙이는 토큰의 종류에 따라 둘로 나뉨

Curriculum Prompt Tuning

Curriculum Prompt Tuning

Complexity Assessment

Scheduler Formulation

Training Execution

Curriculum Prompt Tuning

기존 LLM

텍스트 데이터 위주 학습
행동 임베딩은 학습에 어렵다.
(텍스트 + 임베딩) → 추가 복잡성



단계적으로 복잡도를 높이는 학습 방법 설계

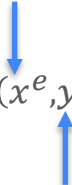
1. 먼저 text-only 방식(쉬운 문제)으로 시작해서
2. 이후 hybrid 방식(복잡한 프롬프트)으로 점진적으로 전환

⇒ 이런 순차적 접근법이 모델이 정보를 온전히 받아들이도록 도움을 준다.

Complexity Assessment

쉬운 과제(Text-only)의 손실

텍스트 프롬프트(“쉬운 문제“용 프롬프트, 영화 이름만
쭉 나열된 것)



A blue arrow points from the text prompt above to the x^e variable in the equation. Another blue arrow points from the text below to the y^e variable in the equation.

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

그 프롬프트에 대한 정답(예: “이 중에 어떤 영화를 추천하는게
맞는가?”)

Complexity Assessment

쉬운 과제(Text-only)의 손실

$|y^e| \rightarrow$ 정답 시퀀스의 길이

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\theta)}(y_t^e | x^e, y_{<t}^e))$$

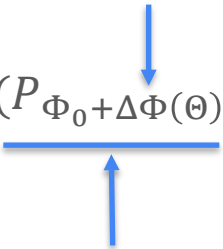
$t \rightarrow$ 정답 시퀀스의 각 단계, $|y^e| \rightarrow$ 정답 시퀀스의 길이

y^e 의 처음부터 끝까지 각 시점마다 예측 에러를
모두 계산해 더한다.

Complexity Assessment

쉬운 과제(Text-only)의 손실

$\Delta\Phi(\theta)$: LoRA(혹은 추가 학습 등으로 새로
조정된 파라미터 변화분)

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\theta)}(y_t^e | x^e, y_{<t}^e))$$


$P_{\Phi_0 + \Delta\Phi(\theta)}$: P : 모델이 예측한 확률분포 함수, Φ_0 : 사전학습 LLM의 기본
파라미터

Complexity Assessment

쉬운 과제(Text-only)의 손실

입력 프롬프트(text-only, 즉 과거 상호작용과 후보군 정보)

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

t번째 정답 아이템(or 토큰)

t번째 이전까지의 정답 시퀀스
(즉, 이미 예측된 출력 값들)

Complexity Assessment

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

- 시퀀스(정답 시퀀스)의 각 단계에서, 정답을 맞출 확률의 로그를 다 합쳐서 마이너스를 붙여 Loss로 표현
- 모델이 정답을 잘 예측할수록 확률이 1에 가까워지고, log값이 0에 가까워져서 Loss가 작아짐(좋은)
- 반대로 못 맞추면 확률이 낮아져 Log Loss가 커짐(나쁨)

Complexity Assessment

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

x^e : "이 유저의 기록(Titanic, Roman Holiday)이 있음. 후보(the Wizard of Oz, Braveheart, Batman & Robin)."

y^e : 실제로 다음에 본 영화(정답) = ["The Wizard of Oz"]

$|y^e| = 1$

수식은 그냥 $t=1$ 한 번만 동작한다.

이때

$$L_{\{easy\}}(x^e, y^e) = -\log\left(P_{\{\Phi\}}("The Wizard of Oz" | x^e)\right)$$

즉, 모델이 "The Wizard of Oz"를 정답으로 뽑을 확률의 로그를 취해서 Loss 차감.

Complexity Assessment

y^e : The Wizard of Oz

$$L_{easy}(x^e, y^e) = -\sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

User has watched: Titanic, Roman Holiday.
Next? Candidates: The Wizard of Oz,
Braveheart, Batman & Robin

가정

"The Wizard of Oz" → 0.84

"Braveheart" → 0.11

"Batman & Robin" → 0.05

그러면,

$$L_{easy}(x^e, y^e) = -\log(0.84) \approx 0.174$$

만약, 모델이 잘못 예측해서 "The Wizard of Oz" 확률이 0.2밖에 안 되면?

$$L_{easy}(x^e, y^e) = -\log(0.2) \approx 1.609$$

→ Loss가 훨씬 크다.

→ 모델 입장에선 손실이 크니까, 파라미터 업데이트를 통해 좀 더 'The Wizard of Oz' 쪽으로 확률이 커지게 학습됨.

Complexity Assessment

복잡도 평가

쉬운 과제(Text-only)의 손실

$$L_{easy}(x^e, y^e) = \sum_{t=1}^{|y^e|} \log(P_{\Phi_0 + \Delta\Phi(\Theta)}(y_t^e | x^e, y_{<t}^e))$$

- (x^e, y^e) Text-Only Prompt를 활용한 학습 데이터

어려운 과제(Text + Behavioral emb)의 손실

$$L_{hard}(x^h, y^h) = \sum_{t=1}^{|y^h|} \log(P_{\Phi_0 + \Delta\Phi(\Theta) + \Theta_P + \Theta_e}(y_t^h | x^h, y_{<t}^h))$$

- Θ_p, Θ_e 프로젝터와 임베딩 층 파라미터

- (x^h, y^h) Hybrid Prompt를 활용한 학습 데이터

Scheduler Formulation

스케줄러 설계

학습 진행 시간 τ , 난이도 높은 과제의 학습 비중을 점차 늘린다.

$$p(\tau) = \frac{\tau}{T}, 0 \leq \tau \leq T$$

$p(\tau)$ 어려운 과제를 학습할 확률
 T 전체 학습기간

초기($\tau = 0$): $p(0)=0 \rightarrow$ 거의 쉬운 문제만!
중간($\tau = \frac{T}{2}$): $p(\frac{T}{2})=0.5 \rightarrow$ 쉬운/어려운 문제 비슷하게!
후기($\tau = T$): $p(T)=1 \rightarrow$ 거의 어려운 문제만!

\Rightarrow “지금 이 시점엔 쉬운 문제와 어려운 문제 중 뭘 더 많이 보여줄까?” 를 결정한다.
훈련이 진행될수록 쉬운 문제 비중은 줄이고, 어려운 문제 비중은 점점 더 늘린다.

Training Execution

$$\mathbb{I}(\tau) = \begin{cases} 1, & \text{learning hard task (w.p. } p(\tau)) \\ 0, & \text{learning easy task (w.p. } 1 - p(\tau)) \end{cases}.$$

→ 쉬운(text-only) 문제의 로스(loss)를 계산해 더함.
→ 단, $\mathbb{I}(\tau) = 0$ 일 때만(즉, 쉬운 문제일 때) 포함됨.

$$\min_{\theta, \theta_p} \sum_{(x,y) \in \mathcal{Z}} ((1 - \mathbb{I}(\tau))L_{easy}(x, y) + \mathbb{I}(\tau)L_{hard}(x, y))$$

학습해야 할 파라미터들
LLM 자체와 임베딩층(projector)의 파라미터

→ 어려운(hybrid) 문제의 로스(loss)를 계산해 더함.
→ 단, $\mathbb{I}(\tau) = 1$ 일 때만(즉, 어려운 문제일 때) 포함됨.

EXPERIMENTS AND RESULTS

- RQ1: LLaRA는 기존의 순차 추천 모델들과 대형 언어 모델(LLM)을 기반으로 한 추천 방법들과 비교하여 어떤 성능을 보이는가?
- RQ2: 하이브리드 프롬프트 방식은 다른 아이템 표현 방식들과 비교하여 어떤가?
- RQ3: 우리 커리큘럼 학습 전략은 다른 방식의 모달리티 통합 방식과 비교해서 어떤가?

Statistics of Datasets

Table 1: Statistics of Datasets.

Dataset	MovieLens	Steam	LastFM
# Sequence	943	11,938	1,220
# Item	1,682	3,581	4,606
# Interaction	100,000	274,726	73,510

Sequence: 사용자별 행동 이력

Item: 아이템 수

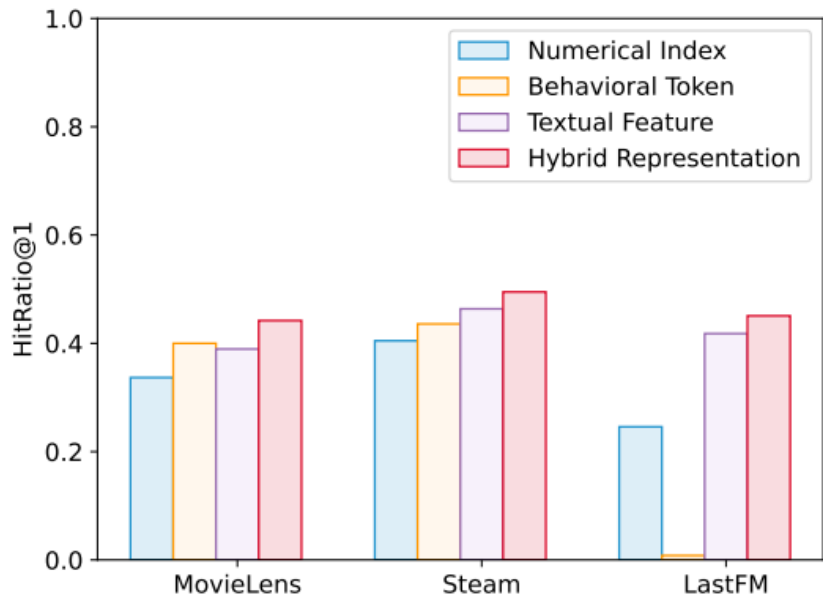
Interaction: 사용자-아이템 간 상호작용의 수

MovieLens: 영화 추천 데이터셋

Steam: 게임 추천 데이터셋

LastFM: 음악 추천 데이터셋

항목별 추천 성능



HitRatio@1: 추천 정확도를 측정하는 지표

Behavioral Token: 추천 모델 학습 → 토큰

Textual Feature: 텍스트 정보만 사용

Numerical Index: 아이템을 숫자 ID로만 표현

Hybrid Representation(LLaRA):

행동 임베딩 + 텍스트 정보를 결합한 표현

하이브리드 표현 방식의 성능이 좋다!

기존 모델과 성능 비교

		MovieLens*		Steam*		LastFM	
		ValidRatio	HitRatio@1	ValidRatio	HitRatio@1	ValidRatio	HitRatio@1
Traditional	GRU4Rec	1.0000	0.3750	1.0000	0.4168	1.0000	0.2616
	Caser	1.0000	0.3861	1.0000	0.4368	1.0000	0.2233
	SASRec	1.0000	0.3444	1.0000	0.4010	1.0000	0.2233
LLM-based	Llama2	0.4421	0.0421	0.1653	0.0135	0.3443	0.0246
	GPT-4	0.9895	0.2000	0.9798	0.3626	1.0000	0.3770
	MoRec	1.0000	0.2822	1.0000	0.3911	1.0000	0.1652
	TALLRec	0.9263	0.3895	0.9840	0.4637	0.9836	0.4180
Ours	LLaRA (GRU4Rec)	0.9684	<u>0.4421</u>	0.9975	<u>0.4924</u>	0.9836	<u>0.4344</u>
	LLaRA (Caser)	0.9684	0.4737	0.9966	0.4874	0.9918	<u>0.4344</u>
	LLaRA (SASRec)	0.9684	<u>0.4421</u>	0.9975	0.4949	1.0000	0.4508

•ValidRatio: 유효한 추천 응답을 생성한 비율. 즉, 말이 되는 정답을 고른 확률

•HitRatio@1: 가장 높은 확률로 예측한 아이템이 실제 선택과 일치하는 비율.

학습 전략별 성능 비교

	MovieLens	Steam	LastFM
Direct	0.4211	0.4899	0.4508
Two-stage	0.4316	0.4840	0.4344
LLaRA (CL)	0.4421	0.4949	0.4508

- Direct: 바로 하이브리드 프롬프트(난이도 높음)
- Two-stage: 텍스트 기반 프롬프트로만 먼저 학습 → 하이브리드로 전환
(단계 구분은 있지만 별도 처리)
- LLaRA (CL): 텍스트 only → 하이브리드 비중 높임

모델별 예측 비교

The watching history of a user is as follows. Please recommend a next movie for this user to watch.



SASRec
Mr. Smith Goes to Washington ✗

TALLRec
The Great Escape ✓

LLaRA
The Great Escape ✓

(a)

(a) 전쟁, 첩보, 모험

The watching history of a user is as follows. Please recommend a next movie for this user to watch.



SASRec
Batman & Robin ✓

TALLRec
The Devil's Own ✗

LLaRA
Batman & Robin ✓

(b)

(b) 액션, SF, 재난

모델별 예측 비교



하이브리드 프롬프트의
LLaRA의 강점을 보여줌!

(a) 왼쪽 사례: "The Great Escape(전쟁+모험)" 정답

SASRec(추천 모델)은 **사용자 행동 패턴만 보고** 정치 드라마를 추천 → 장르 선호 파악 실패

(b) 오른쪽 사례: "Batman & Robin(히어로, 액션)" 정답

TALLRec(LLM)은 단어 의미에 기반해 *The Devil's Own*(**액션**)을 추천했지만, **사용자 행동의 흐름**과는 다소 맞지 않음.

CONCLUSION

- LLaRA는 LLM+전통 추천 시스템의 장점을 융합
- 하이브리드 프롬프트를 활용하여, 추천 정확도 향상
- 커리큘럼 학습 전략을 통한 학습으로 성능 향상에 기여

→ 추천 시스템의 새로운 가능성을 제시한 강력한 프레임워크임.

Q&A