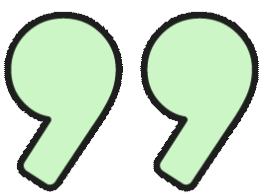
# Recommender Systems with Generative Retrieval

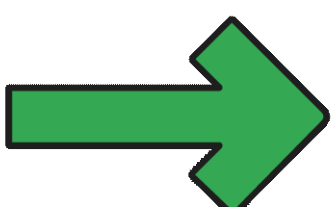
- AI 2팀



Google Developer Group
Editable University Name

# Introduction

AI파트_2팀 방가연



# 추천 시스템이란?

사람들이 관심 있는 콘텐츠를 찾을 수 있도록 도와주는 시스템

- 유튜브에서 내가 좋아할 만한 **비디오 추천**
- 앱스토어에서 내가 자주 사용하는 앱과 비슷한 **앱 추천**
- 쇼핑몰에서 **상품 추천**
- 멜론이나 스포티파이 같은 곳에서 **음악 추천** 등

# 검색 및 랭킹(retrieve-and-rank)

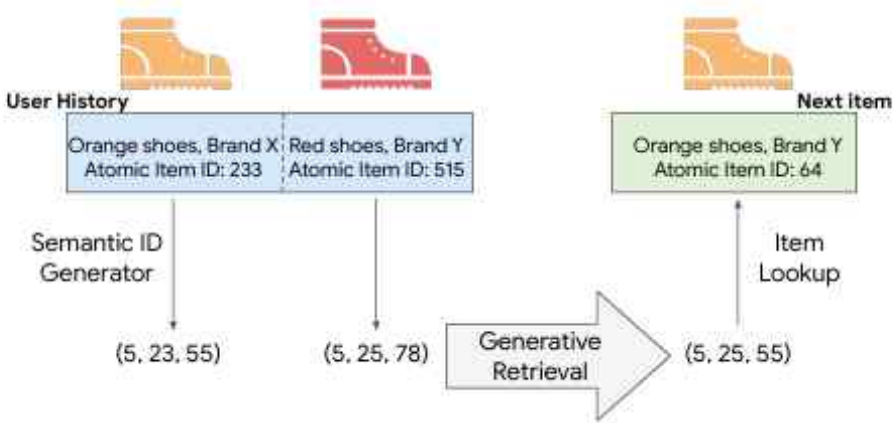
1. 검색 단계(Retrieval): 추천 가능한 여러 아이템 중 후보군을 먼저 뽑기
1. 순위 매기기 단계(Ranking): 후보군 중에서 가장 좋은 것을 선택

즉 검색된 후보들만 보고 판단하기 때문에

검색 단계에서 매우 관련성 높은 후보를 내놓는 것이 중요

검색단계에서 좋은 후보가 나오지 못했다면 무용지물 → 이 검색 단계를 새로운 방식으로 추천

# <TIGER 프레임워크>




사용자 기록(User History): 사용자가 최근에 주황색 신발, 빨간색 신발을 봄.

Semantic ID 생성:
의미 있는 숫자 조합 (5, 23, 55) 으로 변경
예) 'Brand', 'Color', 'Category' 같은 요소로 구성된 번호 조합

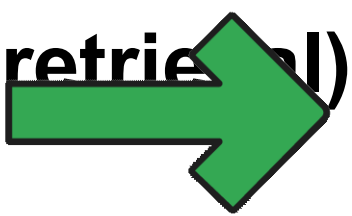
Generative Retrieval (생성 기반 검색)

Transformer 모델이 이 Semantic ID들을 보고 다음에 추천할 아이템의 Semantic ID 직접 생성

추천 결과(new item)

생성된 Semantic ID에 해당하는 아이템(예: 또 다른 주황색 신발)을 찾아서 추천

기존 추천 시스템에서 사용되는 검색(retrieval) 모델들



# ① Matrix Factorization (행렬 분해)

사용자와 아이템을 같은 벡터 공간에 임베딩하여, 두 벡터 사이의 유사도(보통 내적 또는 코사인 유사도)로 선호도를 계산

작동 방식:

- 예: 사용자 A의 벡터 = [0.8, 0.2], 아이템 X의 벡터 = [0.7, 0.3]
- → 두 벡터가 가까우면, 그 아이템을 추천!

과거 데이터를 바탕으로 아이템 간, 사용자 간의 숨겨진 패턴을 학습

# ② Dual-Encoder (듀얼 인코더 구조)

사용자 쿼리와 아이템을 각각 다른 신경망 모델(타워)로 임베딩하고, 두 벡터 간 내적(inner product)으로 유사도 계산

작동 방식:

- 왼쪽 타워: 사용자의 시청/구매 이력 등을 받아서 쿼리 임베딩 생성
- 오른쪽 타워: 아이템의 정보(설명, 장르, 태그 등)로 임베딩 생성
  두 벡터 간 내적 → 점수가 높을수록 추천 우선순위 높음

비선형 관계까지 잘 포착 가능

쿼리와 아이템을 별도로 사전 계산해 저장해두기 쉬움 (ANN과 잘 결합됨)

# ③ Approximate Nearest Neighbor (근사 최근접 이웃, ANN)

엄청나게 많은 아이템 중에서, 가장 비슷한 벡터들만 빠르게 찾는 알고리즘

작동 방식:

- 수만~수백만 개 아이템 중에서, 유사한 것 몇백 개만 먼저 추림

수학 문제집 수천 권 중에서

→ 나랑 취향이 비슷한 애들이 좋아한 문제집 3권"만 빠르게 찾아냄

검색 속도를 극적으로 향상시킴: retrieval 단계의 핵심 기술

# **TIGER** 모델이 기존 추천 시스템과의 차별점

**기존 방식 vs. Semantic ID 방식**

아이템을 무작위 ID나 숫자 기반 임베딩

(한계)

- 유사한 아이템 간 의미 공유 불가능
- 새로운 아이템(콜드스타트)에 대한 일반화가 어려움

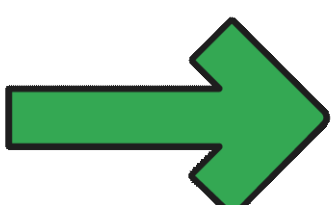
의미 있는 표현인 "Semantic ID":
단순 숫자가 아닌, 아이템의 콘텐츠(텍스트, 설명 등) 기반으로 생성된 의미 있는 토큰들의 조합
(장점)

- 비슷한 아이템 간 지식 공유 가능
- 특정 아이템만 반복 추천하는 문제를 줄일 수 있음
- 콜드스타트(신규 아이템) 대응 가능
- 토큰 조합으로 수십억 개 아이템도 표현 가능

Google Developer Group
Editable University Name

# Related Work

This is where the subtitle can be displayed.

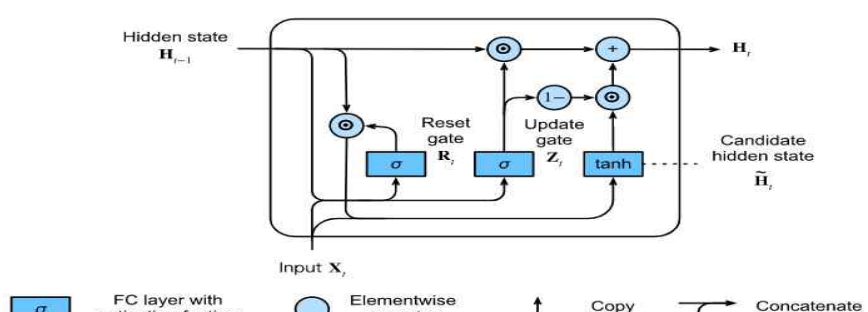


# Sequential Recommenders(순차 추천 시스템)

순차 추천 시스템: 사용자의 과거 행동 순서를 고려하여 다음에 어떤 아이템을 추천할지 예측하는 추천 시스템.

- GRU4ReC: RNN의 기존 단점을 GRU를 통해 보완하여 사용자의 행동 순서를 학습하여 더 나은 추천을 제공
- GRU(Gated Recurrent Unit)란? 리셋 게이트와 업데이트 게이트를 사용하여 RNN의 장기 의존성 문제를 해결함. LSTM과 비슷하지만 구조는 더 단순하기 때문에 데이터의 양이 가볍고 빠름.
- NARM(Neural Attentive Session-based Recommendation): 기존 GRU 모델에 **attention 메커니즘을** 추가하여 이전 행동 뿐만 아니라 사용자가 장기적인 관심까지 반영하여 추천함.
- Attention 메커니즘이란? 가장 중요한 아이템을 강조하는 기법.

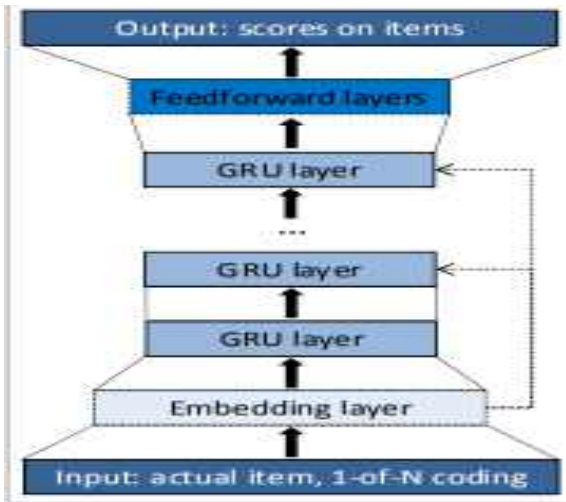
Ex) 만약 사용자가 청바지 → 반팔 티셔츠 → 운동화 → 정장 바지로 검색을 하였을 때, 가장 중요한 클릭인 정장 바지(마지막 클릭)가 추천에 더 영향을 주는 방식.

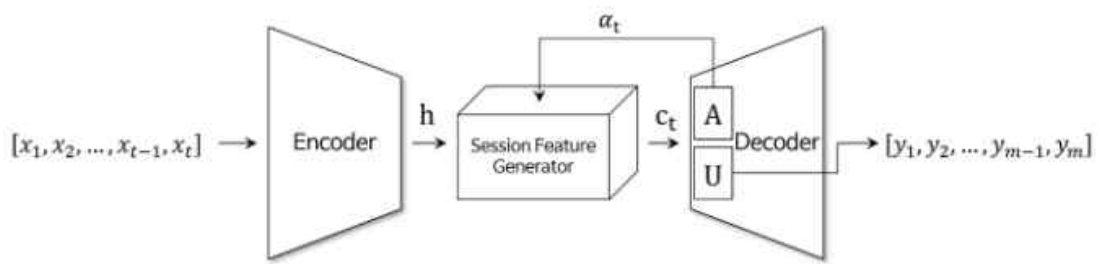


# Sequential Recommenders(순차 추천 시스템)

순차 추천 시스템: 사용자의 과거 행동 순서를 고려하여 다음에 어떤 아이템을 추천할지 예측하는 추천 시스템.

- GRU4ReC: 최초로 RNN을 사용한 추천 시스템. RNN을 이용해 사용자의 행동 순서를 학습하여 더 나은 추천을 제공
- NARM(Neural Attentive Session-based Recommendation): 기존 GRU 모델에 attention 메커니즘을 추가하여 이전 행동 뿐만 아니라 사용자가 장기적인 관심까지 반영하여 추천함.



# Sequential Recommenders(순차 추천 시스템)

순차 추천 시스템: 사용자의 과거 행동 순서를 고려하여 다음에 어떤 아이템을 추천할지 예측하는 추천 시스템.

- AttRec: 사용자의 **현재 세션(intent)을 모델링**하기 위해 **자기 주의 메커니즘(self-attention)** 사용
- SASRec: **디코더 전용 Transformer 모델**을 활용한 **self-attention** 사용
- Self-Attention이란? **입력 데이터 내에서 중요한 요소를 스스로 찾고 집중하는 기법. 주로 Transformer 모델에서 핵심적인 역할을 함.**

**Ex)** "어벤져스"를 봤다면, 이전에 본 "아이언맨"과 "캡틴 아메리카"가 중요할 확률이 높음

**-> 높은 가중치를 받고 학습을 함.**

| 특징 | RNN (기본 순환신경망) | GRU4Rec (GRU 기반 추천 시스템) | Self-Attention 기반 모델 (SASRec, BERT4Rec 등) |
|---|---|---|---|
| 기본 개념 | 시퀀스 데이터를 순차적으로 처리하는 신경망 | GRU를 활용한 세션 기반 추천 시스템 | 자기 주의 메커니즘(Self-Attention)을 활용한 추천 모델 |
| 주요 구조 | 단순한 순환 구조 | GRU(게이트를 활용한 RNN) | Transformer의 Self-Attention 구조 |
| 장기 의존성 문제 | ✅ 해결이 어렵고, 학습이 잘 안 됨 (Vanishing Gradient 문제) | ✅ GRU의 게이트 메커니즘 덕분에 해결됨 | ✅ Attention을 활용하여 장기 의존성을 효과적으로 모델링 |
| 학습 속도 | ⏳ 느림 (길이가 길어질수록 비효율적) | ⏳ RNN보다는 빠르지만, 여전히 병렬 처리가 어려움 | ⚡ 빠름 (병렬 처리 가능) |
| 추천 방식 | 과거 데이터를 기반으로 순차적으로 예측 | 사용자의 세션 데이터를 활용하여 다음 아이템을 추천 | Self-Attention을 이용해 시퀀스 내의 모든 항목 간 관계를 학습하여 추천 |
| 대표 모델 | 기본 RNN, LSTM | GRU4Rec | SASRec, BERT4Rec, P5, S3-Rec |

# Sequential Recommenders(순차 추천 시스템)

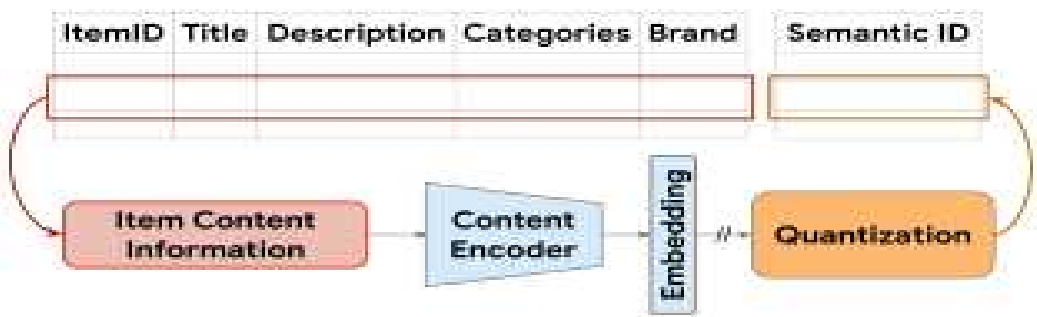
| 특징 | RNN (기본 순환신경망) | GRU4Rec (GRU 기반 추천 시스템) | Self-Attention 기반 모델 (SASRec, BERT4Rec 등) |
|---|---|---|---|
| 기본 개념 | 시퀀스 데이터를 순차적으로 처리하는 신경망 | GRU를 활용한 세션 기반 추천 시스템 | 자기 주의 메커니즘(Self-Attention)을 활용한 추천 모델 |
| 주요 구조 | 단순한 순환 구조 | GRU(게이트를 활용한 RNN) | Transformer의 Self-Attention 구조 |
| 장기 의존성 문제 | ✅ 해결이 어렵고, 학습이 잘 안 됨 (Vanishing Gradient 문제) | ✅ GRU의 게이트 메커니즘 덕분에 해결됨 | ✅ Attention을 활용하여 장기 의존성을 효과적으로 모델링 |
| 학습 속도 | ⏳ 느림 (길이가 길어질수록 비효율적) | ⏳ RNN보다는 빠르지만, 여전히 병렬 처리가 어려움 | ⚡ 빠름 (병렬 처리 가능) |
| 추천 방식 | 과거 데이터를 기반으로 순차적으로 예측 | 사용자의 세션 데이터를 활용하여 다음 아이템을 추천 | Self-Attention을 이용해 시퀀스 내의 모든 항목 간 관계를 학습하여 추천 |
| 대표 모델 | 기본 RNN, LSTM | GRU4Rec | SASRec, BERT4Rec, P5, S3-Rec |

셀프 어텐션 기반 모델은 더 정교한 추천이 가능하고 속도도 빠름!

# Semantic ID

## Semantic ID란? 왜 필요할까?

- 앞서 설명한 기존 추천 시스템에서는 아이템을 ID(숫자)로만 표현하여 비슷한 의미를 가진 아이템을 구별하기 어려움.
- 이를 보완하기 위해 아이템의 의미를 반영한 Semantic ID를 만들어 계층적 특성을 공유하여 추천 시스템에서의 성능을 높임.




(a) Semantic ID generation for items using quantization of content embeddings.

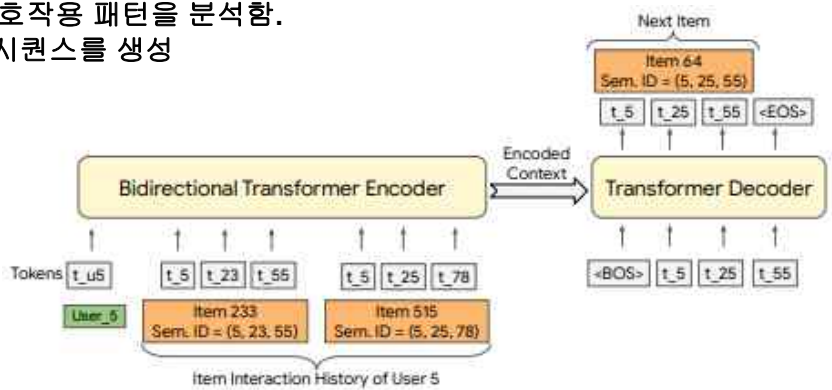
# 생성적 검색

## 인코더와 디코더 모델을 사용한 생성적 검색 수행 과정

- Transfomer 인코더-디코더 아키텍처: 문맥을 잘 반영하여 이전의 입력들이 다음 예측에 영향을 미칠 수 있도록 함.

1. **입력 레이어**: 사용자의 아이템 상호작용 이력이 입력으로 주어짐.
2. **Bidirectional Transformer Encoder:** 입력 시퀀스를 양방향으로 인코딩
3. **Encoded context:** 인코딩된 정보를 디코더로 전달, 이때 사용자의 상호작용 패턴을 분석함.
4. **Transformer Decoder**: 인코딩된 내용과 시작 토큰을 기반으로 다음 시퀀스를 생성

BOS: 디코더에 입력 시퀀스의 시작을 알리기 위한 토큰
EOS: 시퀀스가 끝났음을 의미하는 토큰, 이후 시퀀스 출력 생성 중단




(b) Transformer based encoder-decoder setup for building the sequence-to-sequence model used for generative retrieval.

# **Semantic ID** 생성 및 표현

아이템의 내용 기반(텍스트, 이미지, 메타데이터 등)을 활용하여 각 아이템이 의미적으로 가까운 벡터로 생성된다.

- 아이템 콘텐츠 → 사전 학습된 콘텐츠 임베딩 모델(예: Sentence-T5, BERT 등)을 사용하여 콘텐츠 특성을 표현하는 임베딩 벡터 생성.
- 생성된 벡터를 양자화하여 Semantic ID로 표현.
- 논문은 계층적 양자화 방법으로 RQ-VAE를 소개함.

▶ **결과적으로, 아이템은 랜덤한 ID 대신 의미가 담긴 코드(semantic tokens)들의 조합으로 표현된다.**

➢ Semantic ID) Semantic codeword가 결합된 tuple 형태 $Semantic\ ID = (c_0, c_1, \ldots, c_{m-1})$

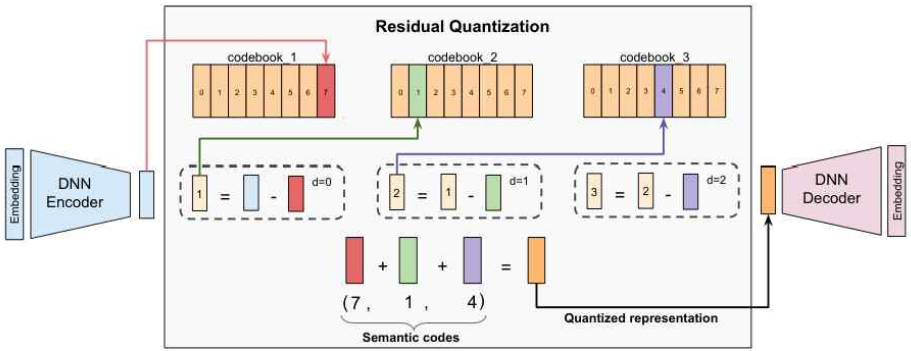
기존 방식 Atomic ID

해리포터 1권 -> Item ID 560

TIGER의 Semantic ID

해리포터 1권 -> [“소설”, “판타지”, “해리포터“, “1권”]

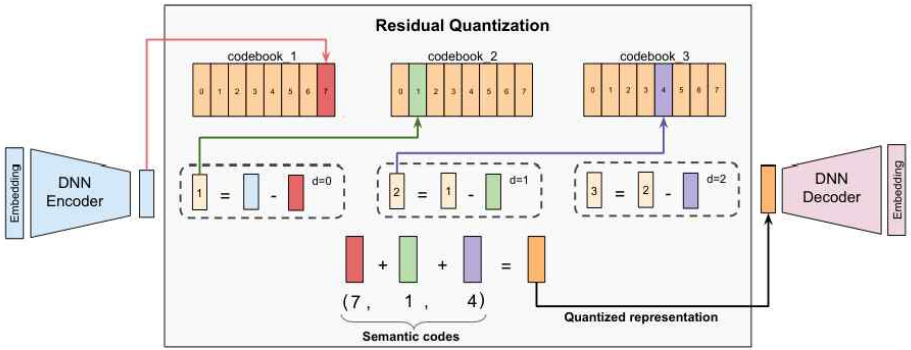
# **Semantic ID** RQ_VAE(Residual Quantization Variational Autoencoder)




- 아이템 콘텐츠 특징으로 얻어진 semantic embedding을 DNN Encoder 모델에 투입.
- 얻어진 embedding vector를 여러 개의 레벨로 구성된 codebook을 사용하여 순차적으로 양자화.
- Residual Quantization(잔차 양자화)는 다음과 같이 동작함.

1. embedding($z$)에 가장 가까운 벡터를 첫번째 수준의 codebook에서 선택. 선택한 codebook 내 색인의 값을 semantic code로 기록.
2. 최초 embedding($z$)과 선택한 closest vector 사이의 차이(잔차, residual)를 계산.
3. 이 residual vector를 다음 레벨 codebook에서 다시 가장 가까운 벡터와 비교하여 이를 반복. 각 레벨에서 결정된 semantic 코드들이 최종 Semantic ID 튜플을 구성.
4. 3개의 codebook을 거쳐 "(7, 1, 4)"가 Semantic ID로 생성됨.

# **Semantic ID** RQ_VAE(Residual Quantization Variational Autoencoder)




Semantic embedding ( $z$ )는 최초 encoder ($\mathcal{E}$)를 통해 얻는 embedding 임:

$$z := \mathcal{E}(x) \quad (encoder\text{에 의해 얻어진 } embedding\text{벡터})$$

레벨별로 잔차($r_d$)가 점진적으로 계산됨:

레벨 d = 0인 초기에는:

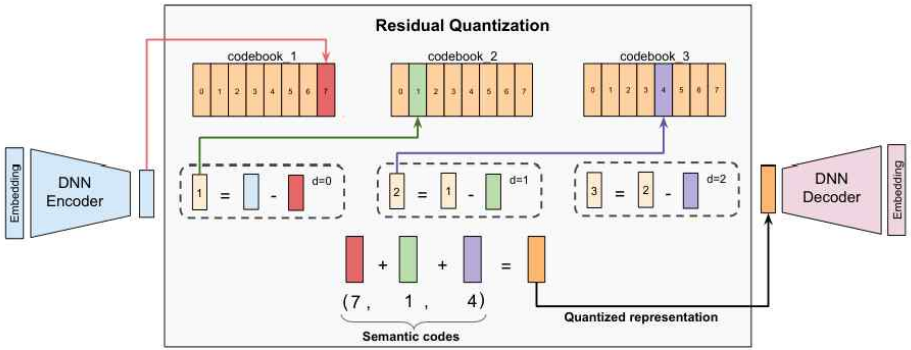
$$r_0 := z$$

레벨 d에서 codebook ($C_d$) 는 다음과 같이 정의됨:

$$C_d := \{e_k\}_{k=1}^{K}$$

($e_k$: $d$번째 $codebook$의 $k$번째 $embedding$ 벡터, $K$: $codebook$의 크기)

# **Semantic ID** RQ_VAE(Residual Quantization Variational Autoencoder)




각 레벨 d에서 residual vector $(r_d)$를 codebook $(C_d)$ 내 가장 가까운 벡터로 양자화함:

$$c_d = \arg\min_k \|r_d - e_k\|$$

다음 잔차(residual)은 현 residual에서 선택한 codebook 벡터를 빼서 계산:

$$r_{d+1} = r_d - e_{c_d}$$

이 과정을 반복하여 총 m개의 코드 $((c_0, c_1, \ldots, c_{m-1}))$를 얻는다.

# **Semantic ID** RQ_VAE(Residual Quantization Variational Autoencoder)

양자화된 최종 벡터 ($\hat{z}$)는 이렇게 정의할 수 있음:

$$\hat{z} := \sum_{d=0}^{m-1} e_{c_d}$$

- 이후 decoder는 이 벡터 ($\hat{z}$)를 받아 원래 아이템 특징 ($x$) 를 복원(재구성).

RQ-VAE의 손실함수(Loss Function):
RQ-VAE는 크게 2개의 항으로 이루어진 손실함수를 사용.

$$\mathcal{L}(x) = \mathcal{L}_{recon}(x) + \mathcal{L}_{rqvae}(x)$$

이때, 각 항의 정의는:
① 재구성 손실(Reconstruction loss, ($\mathcal{L}_{recon}$)
- 원래 입력 ( $x$ )를 decoder가 재구성한 결과(($\hat{x}$))와의 차이.

$$\mathcal{L}_{recon}(x) := \|x - \hat{x}\|^2$$

② RQ-VAE에 사용되는 양자화 손실($\mathcal{L}_{rqvae}$):
- 양자화될 때의 residual vector와 선택된 codebook vector와의 차이를 줄이기 위한 손실.

$$\mathcal{L}_{rqvae}(x) := \sum_{d=0}^{m-1} \left( |sg[r_d] - e_{c_d}|^2 + \beta |sg[e_{c_d}] - r_d|^2 \right)$$

여기서
- ($sg[\cdot]$): stop gradient 연산으로, gradient가 계산될 때 내부 미분을 중단시키는 함수
- ($\beta$): 양자화 항의 비중을 조절하기 위한 hyperparameter

# **Semantic ID** RQ_VAE(Residual Quantization Variational Autoencoder)

코드북 초기화 및 잠재적 문제 해결:

- RQ-VAE 학습 중 코드북(codebook collapse)이 고정된 소수의 코드벡터에 맵핑되는 문제를 방지하기 위해 K-means 클러스터링 기반으로 codebook을 초기화. (초기 값 기준을 K-means를 통해 결정).

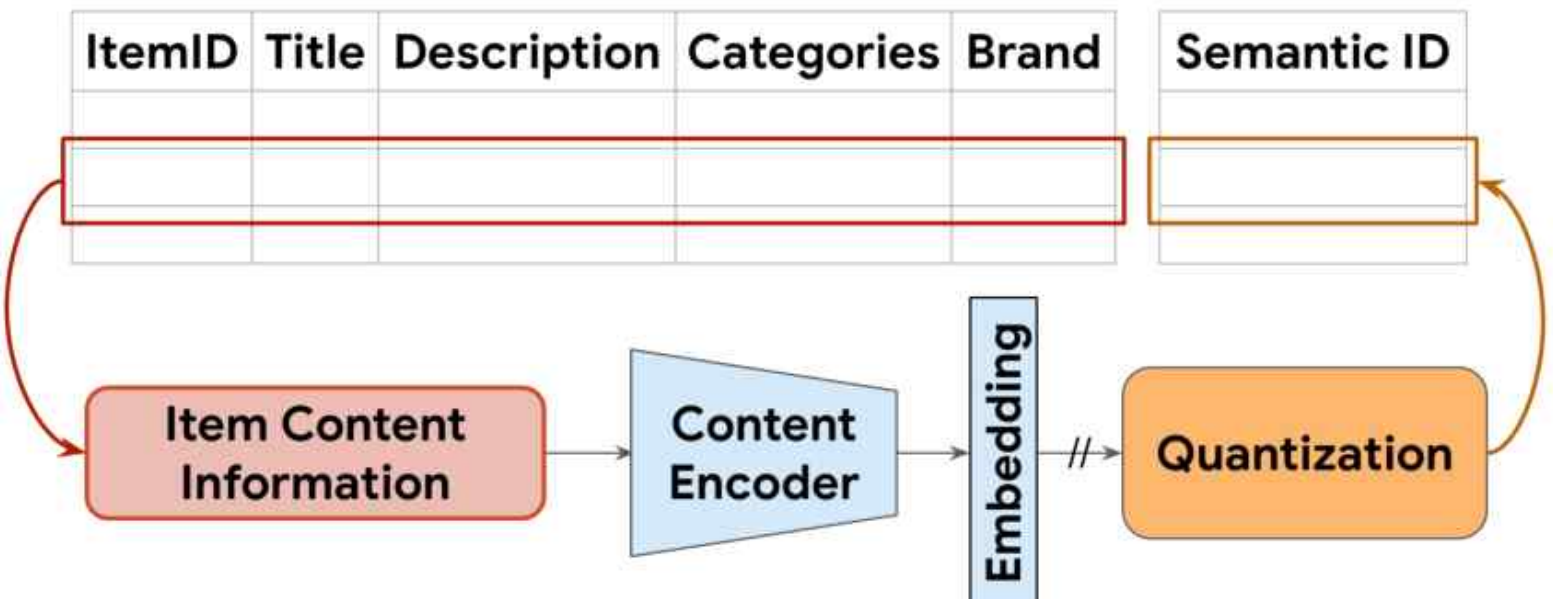
충돌 문제 해결:

- Semantic ID의 충돌 해결 방법은 아래와 같이 표현할 수 있음.
- 기본 Semantic ID 튜플은 길이 m이며, 충돌되는 경우 indexing을 위해 길이가 m+1이 될 수 있도록 새로운 인덱스를 추가적으로 부여:

$$(c_0, c_1, c_2, ..., c_{m-1}, i)$$

(i는 추가적인 충돌해결을 위한 구분용 인덱스 값: 0, 1, 2,... )

# Semantic ID


ItemID
Title
Description
Categories
Brand
Semantic ID
Item Content Information
Content Encoder
Embedding
Quantization

# **Semantic ID** 기반 Generative Retrieval

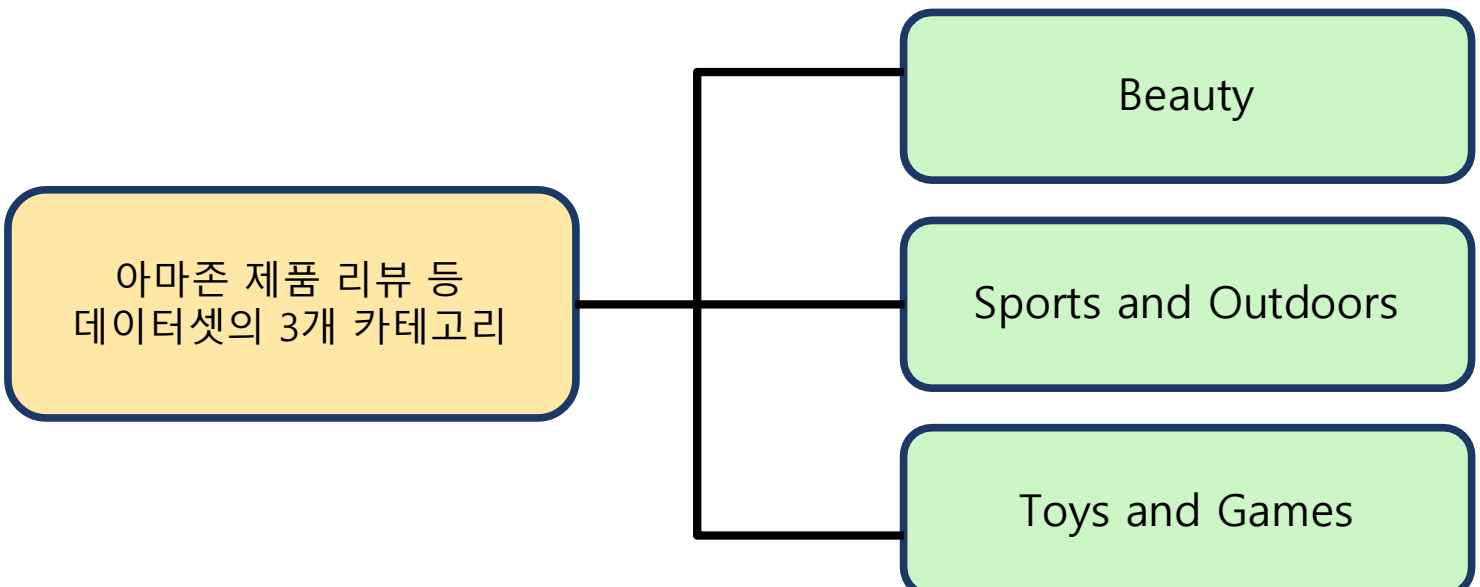
Semantic ID로 데이터 입력 (추천을 위한 입력 데이터로 활용됨.)

사용자가 시간 순서로 아이템과 상호작용한 기록을 Semantic ID들의 시퀀스 형태로 만든 후, 이를 기반으로 Transformer 모델을 이용하여 차기 아이템을 예측

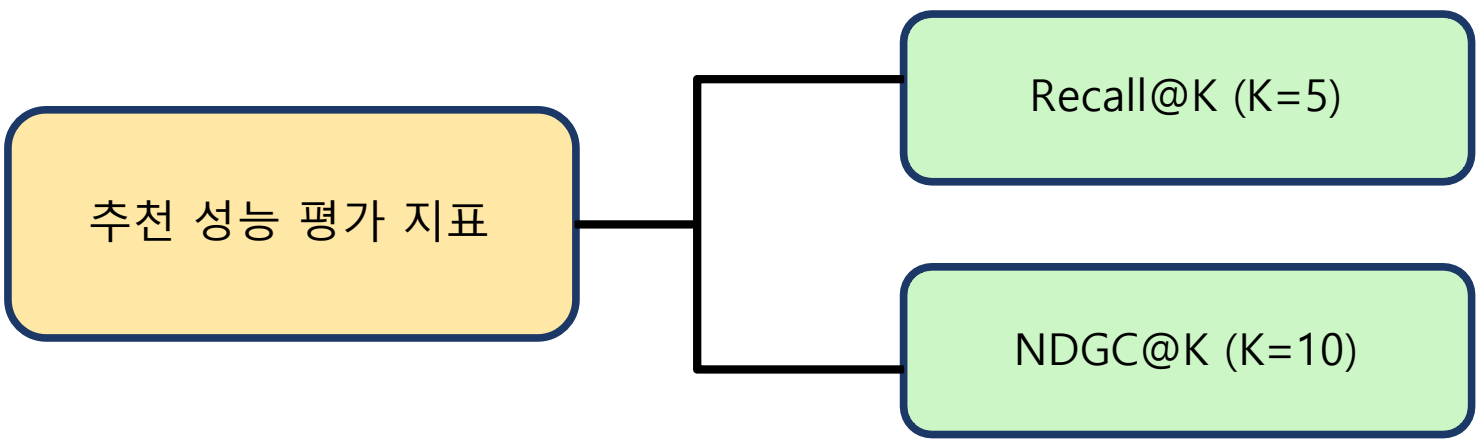
사용자의 아이템 상호작용(Item ID 단위)을 Semantic ID 단위로 변형하여, Transformer를 이용한 generative recommender system 이용 추천

모델은 과거부터 현재까지 사용자의 아이템 ID의 Semantic ID 시퀀스를 입력으로 받아, 다음 아이템의 Semantic ID를 예측하는 방식으로 훈련

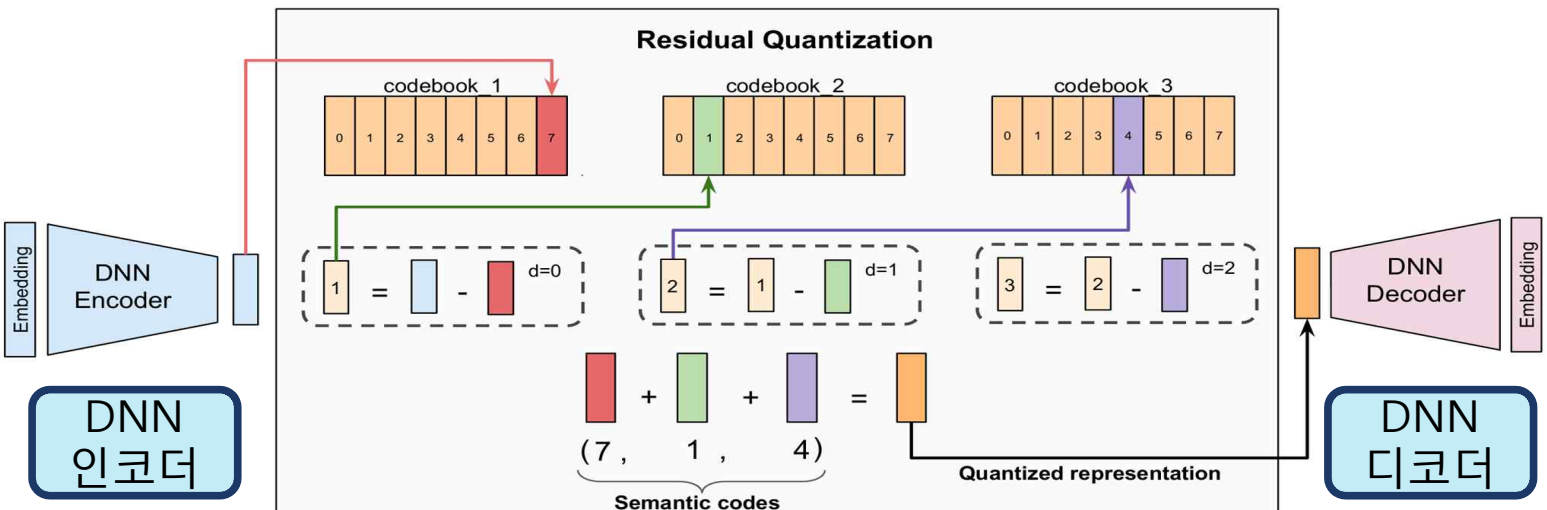
# 4. 실험 (Experiments)



# 4. 실험 (Experiments)



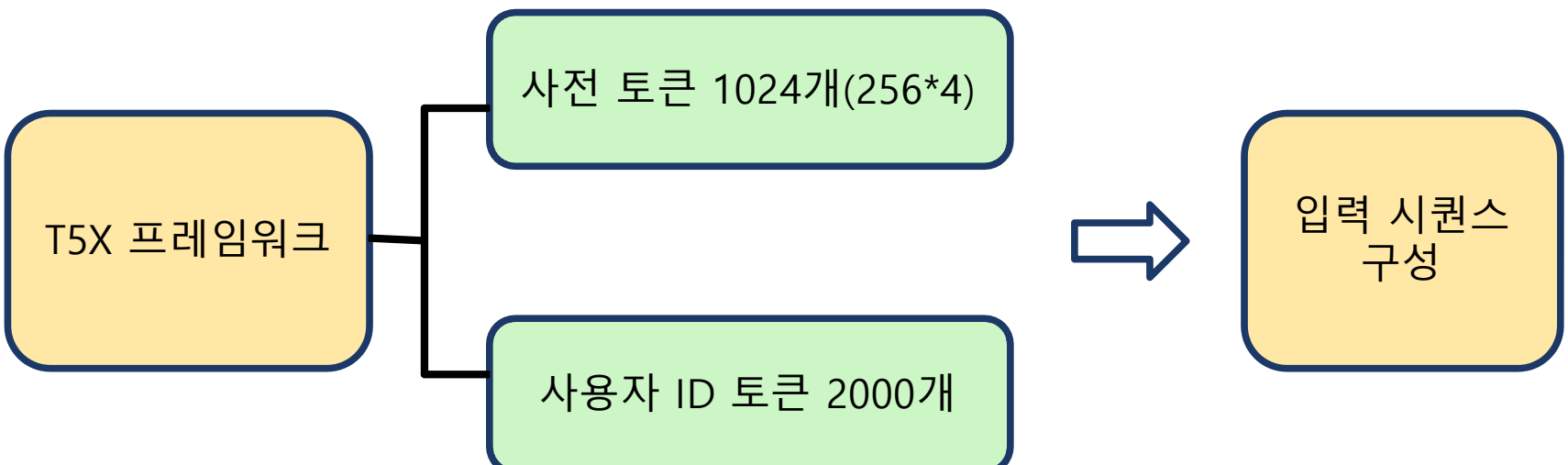
# RQ-VAE 모델 구현




$$L(x) = \|x - \hat{x}\|^2 + \sum_{d=0}^{m-1} \|sg[r_i] - e_{c_i}\|^2 + \beta \|r_i - sg[e_{c_i}]\|^2$$

β = 0.25
20,000 epoch
학습률 0.4
배치 크기 1024
Adagrad 옵티마이저 사용

# 시퀀스-투-시퀀스 모델 구현



# 시퀀스-투-시퀀스 모델 구현

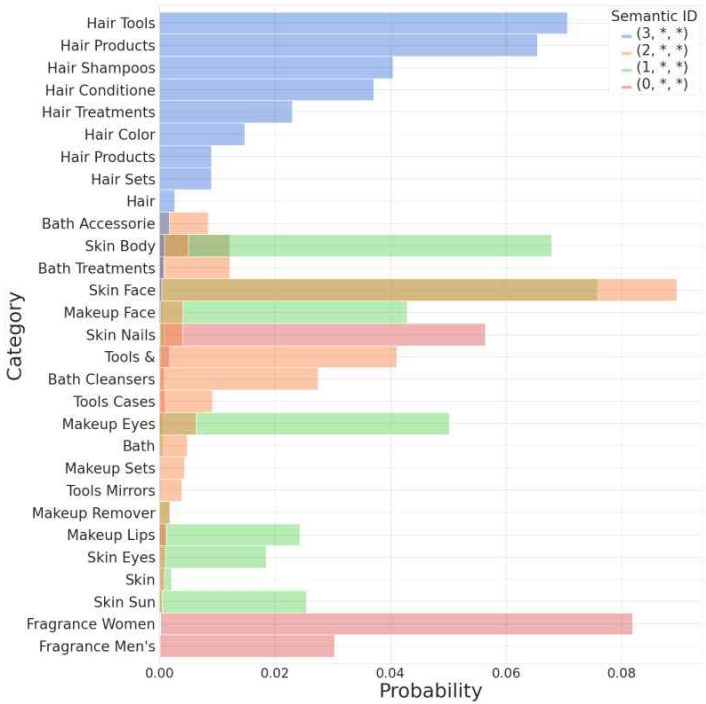
- **Transformer 인코더-디코더 레이어:**각각 4개 레이어
- **Self-attention 헤드:**6개, 차원은 64
- **MLP 차원:**1024, 입력 차원은 128
- **활성화 함수:**ReLU
- **드롭아웃:**0.1
- **모델 파라미터:**약 1,300만 개
- **학습 단계:**"Beauty"와 "Sports and Outdoors" 데이터셋은 200,000단계,
  "Toys and Games"는 100,000단계 학습
- **배치 크기:**256
- **학습률:**처음 10,000단계까지는 0.01, 이후에는 역제곱근 감쇠 스케줄 적용

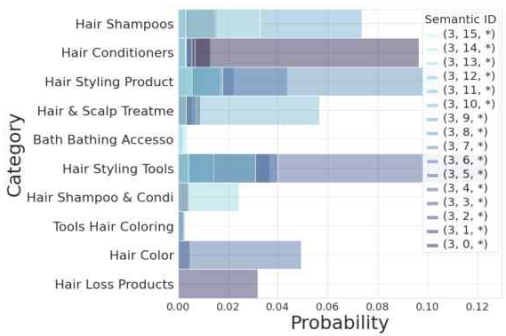
# 4.1 연속적 추천 성능 평가
## (Performance on Sequential Recommendation)

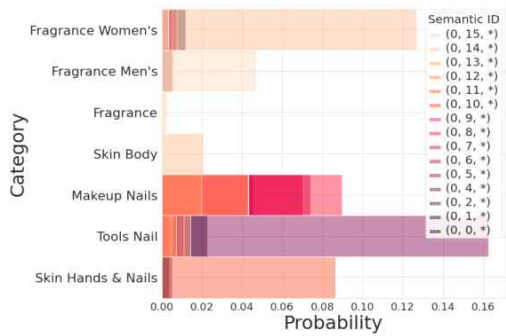
Table 1: Performance comparison on sequential recommendation. The last row depicts % improvement with TIGER relative to the best baseline. Bold (underline) are used to denote the best (second-best) metric.

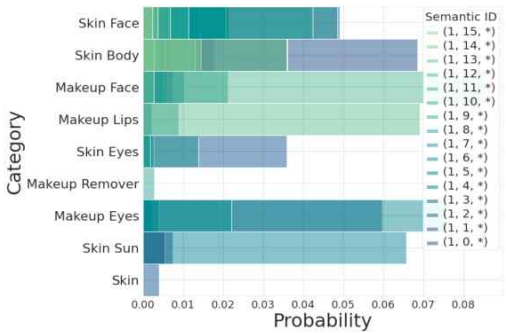
| Methods | Sports and Outdoors | | | | Beauty | | | | Toys and Games | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 |
| P5 [8] | 0.0061 | 0.0041 | 0.0095 | 0.0052 | 0.0163 | 0.0107 | 0.0254 | 0.0136 | 0.0070 | 0.0050 | 0.0121 | 0.0066 |
| Caser [33] | 0.0116 | 0.0072 | 0.0194 | 0.0097 | 0.0205 | 0.0131 | 0.0347 | 0.0176 | 0.0166 | 0.0107 | 0.0270 | 0.0141 |
| HGN [25] | 0.0189 | 0.0120 | 0.0313 | 0.0159 | 0.0325 | 0.0206 | 0.0512 | 0.0266 | 0.0321 | 0.0221 | 0.0497 | 0.0277 |
| GRU4Rec [11] | 0.0129 | 0.0086 | 0.0204 | 0.0110 | 0.0164 | 0.0099 | 0.0283 | 0.0137 | 0.0097 | 0.0059 | 0.0176 | 0.0084 |
| BERT4Rec [32] | 0.0115 | 0.0075 | 0.0191 | 0.0099 | 0.0203 | 0.0124 | 0.0347 | 0.0170 | 0.0116 | 0.0071 | 0.0203 | 0.0099 |
| FDSA [42] | 0.0182 | 0.0122 | 0.0288 | 0.0156 | 0.0267 | 0.0163 | 0.0407 | 0.0208 | 0.0228 | 0.0140 | 0.0381 | 0.0189 |
| SASRec [17] | 0.0233 | 0.0154 | 0.0350 | 0.0192 | 0.0387 | <u>0.0249</u> | 0.0605 | 0.0318 | <u>0.0463</u> | <u>0.0306</u> | 0.0675 | 0.0374 |
| $S^3$-Rec [44] | <u>0.0251</u> | <u>0.0161</u> | <u>0.0385</u> | <u>0.0204</u> | <u>0.0387</u> | 0.0244 | <u>0.0647</u> | <u>0.0327</u> | 0.0443 | 0.0294 | <u>0.0700</u> | <u>0.0376</u> |
| **TIGER [Ours]** | **0.0264** | **0.0181** | **0.0400** | **0.0225** | **0.0454** | **0.0321** | **0.0648** | **0.0384** | **0.0521** | **0.0371** | **0.0712** | **0.0432** |
| | +5.22% | +12.55% | +3.90% | +10.29% | +17.31% | +29.04% | +0.15% | +17.43% | +12.53% | +21.24% | +1.71% | +14.97% |

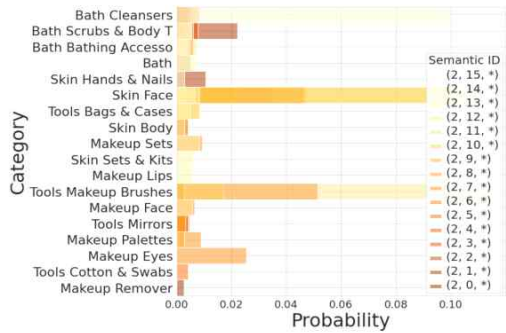
# 4.2 아이템 표현 분석(Item Representation)



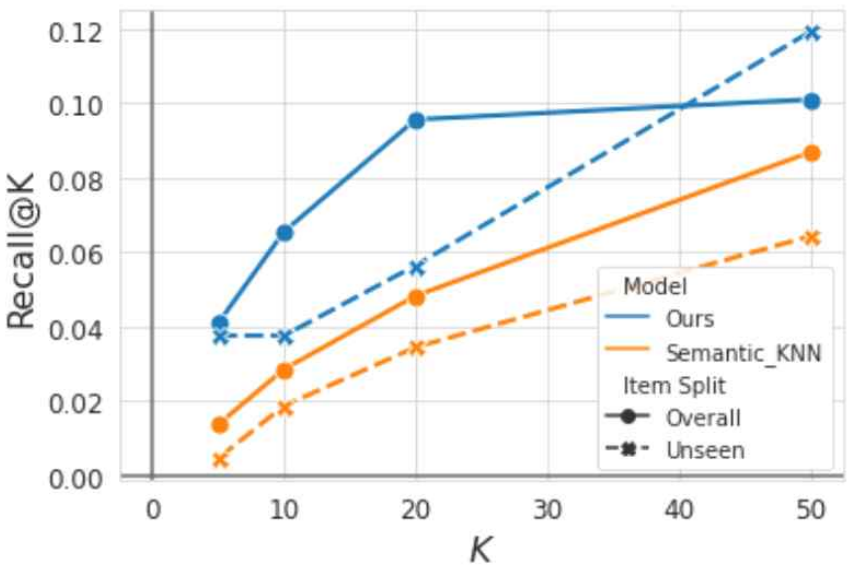



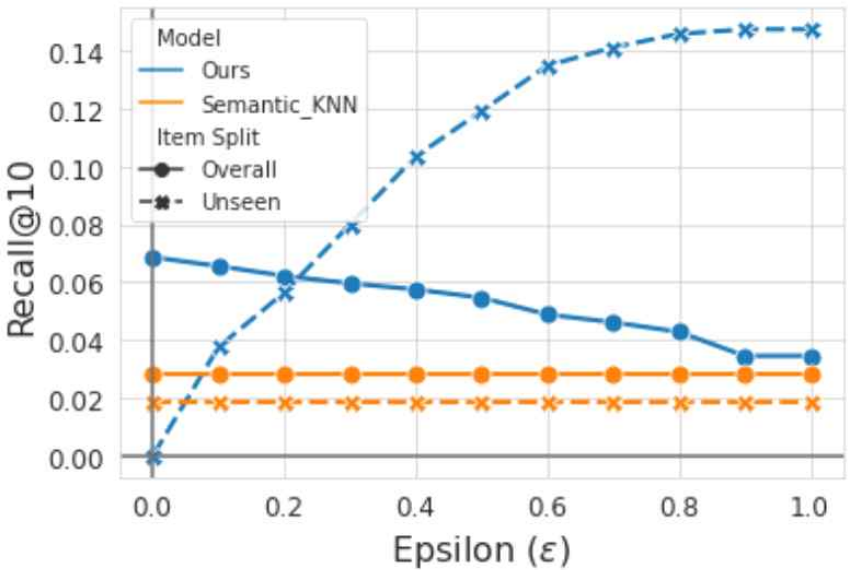
# 4.2 아이템 표현 분석(Item Representation)

| Methods | Sports and Outdoors | | | | Beauty | | | | Toys and Games | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 | Recall @5 | NDCG @5 | Recall @10 | NDCG @10 |
| Random ID | 0.007 | 0.005 | 0.0116 | 0.0063 | 0.0296 | 0.0205 | 0.0434 | 0.0250 | 0.0362 | 0.0270 | 0.0448 | 0.0298 |
| LSH SID | 0.0215 | 0.0146 | 0.0321 | 0.0180 | 0.0379 | 0.0259 | 0.0533 | 0.0309 | 0.0412 | 0.0299 | 0.0566 | 0.0349 |
| RQ-VAE SID | **0.0264** | **0.0181** | **0.0400** | **0.0225** | **0.0454** | **0.0321** | **0.0648** | **0.0384** | **0.0521** | **0.0371** | **0.0712** | **0.0432** |

# 4.3 새로운 기능(New Capabilities)-콜드 스타트 추천




(a) Recall@K vs. K, ($\epsilon = 0.1$).




(b) Recall@10 vs. $\epsilon$.

Figure 5: Performance in the cold-start retrieval setting.

# 4.3 새로운 기능(New Capabilities)-추천 다양성 향상

Table 3: The entropy of the category distribution predicted by the model for the Beauty dataset. A higher entropy corresponds more diverse items predicted by the model.

| Temperature | Entropy@10 | Entropy@20 | Entropy@50 |
|---|---|---|---|
| T = 1.0 | 0.76 | 1.14 | 1.70 |
| T = 1.5 | 1.14 | 1.52 | 2.06 |
| T = 2.0 | 1.38 | 1.76 | 2.28 |

# 4.3 새로운 기능(New Capabilities)-추천 다양성 향상

Table 4: Recommendation diversity with temperature-based decoding.

| Target Category | Most-common Categories for top-10 predicted items | |
|---|---|---|
| | T = 1.0 | T = 2.0 |
| Hair Styling Products | Hair Styling Products | Hair Styling Products, Hair Styling Tools, Skin Face |
| Tools Nail | Tools Nail | Tools Nail, Makeup Nails |
| Makeup Nails | Makeup Nails | Makeup Nails, Skin Hands & Nails, Tools Nail |
| Skin Eyes | Skin Eyes | Hair Relaxers, Skin Face, Hair Styling Products, Skin Eyes |
| Makeup Face | Tools Makeup Brushes,Makeup Face | Tools Makeup Brushes, Makeup Face,Skin Face, Makeup Sets, Hair Styling Tools |
| Hair Loss Products | Hair Loss Products,Skin Face, Skin Body | Skin Face, Hair Loss Products, Hair Shampoos,Hair & Scalp Treatments, Hair Conditioners |

# 4.4 추가 실험(Ablation Study)

Table 5: Recall and NDCG metrics for different number layers.

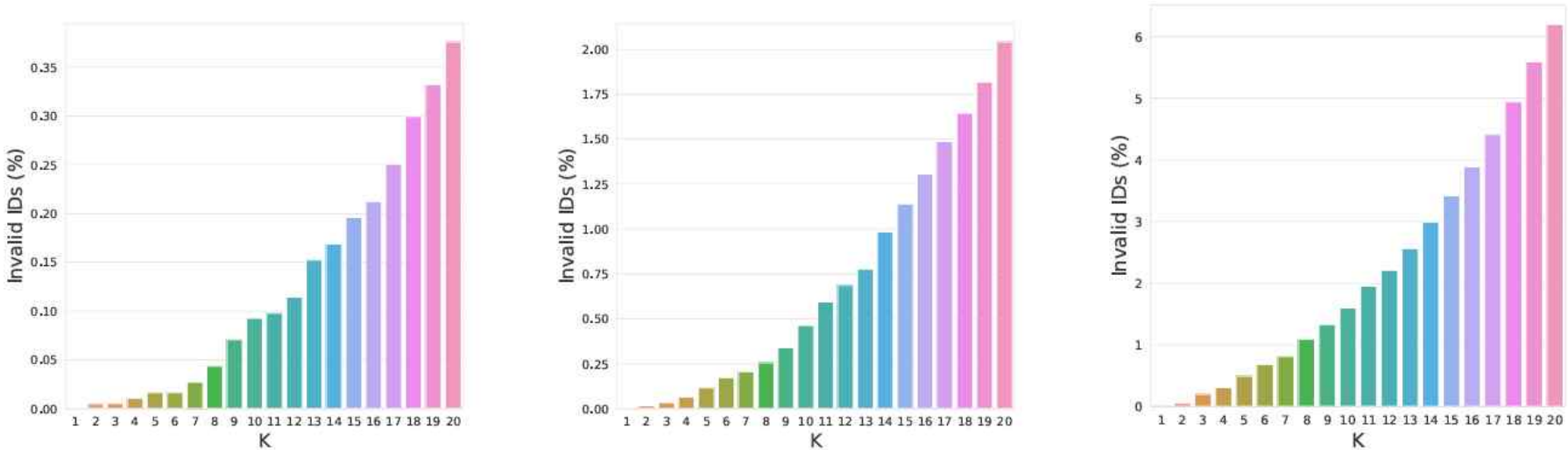
| Number of Layers | Recall@5 | NDCG@5 | Recall@10 | NDCG@10 |
|---|---|---|---|---|
| 3 | 0.04499 | 0.03062 | 0.06699 | 0.03768 |
| 4 | 0.0454 | 0.0321 | 0.0648 | 0.0384 |
| 5 | 0.04633 | 0.03206 | 0.06596 | 0.03834 |

# 4.4 추가 실험(Ablation Study)

Table 8: The effect of providing user information to the recommender system

| Recall@5 | NDCG@5 | Recall@10 | NDCG@10 | |
|---|---|---|---|---|
| No user information | 0.04458 | 0.0302 | 0.06479 | 0.0367 |
| With user id (reported in the paper) | 0.0454 | 0.0321 | 0.0648 | 0.0384 |

# 4.5 잘못된 ID 처리(Invalid ID)




(a) Sports and Outdoors (b) Beauty (c) Toys and Games

Figure 6: Percentage of invalid IDs when generating Semantic IDs using Beam search for various values of $K$. As shown, $\sim 0.3\% - 6\%$ of the IDs are invalid when retrieving the top-20 items.

# 5. 결론(Results)

- TIGER 모델은 생성적 검색을 활용한 새로운 추천 시스템 패러다임을 제시.
- 기존 추천 모델보다 높은 성능을 보이며, 콜드 스타트 문제 해결과 추천 다양성 증가에 기여.
- RQ-VAE의 계층적 양자화 방식이 추천 성능 향상에 중요한 역할을 함.
- 미래 연구로 잘못된 ID 처리 방식 개선, 모델 경량화, 대규모 데이터셋 확장 적용을 계획.