Semantic IDs for Recommender Systems via Vector Quantization and Generative Models

벡터 양자화와 생성 모델을 통한 추천시스템의 Semantic ID

Hae-Yoon Koo (구해윤)

Supervisor: Byungkook Oh

Graph & Language Intelligence Laboratory
Department of Computer Science and Engineering
Konkuk University

2026.02.12

CONTENTS

1. Background

Background

추천시스템의 발전

- 전통적인 추천 시스템
  - ✓ 사용자가 아이템과 상호작용 했던 이력을 기반으로
  - ✓ 추천할 후보 아이템의 집합을 판별하거나 순위를 매기는 방식

  - ✓ Sequential Recommendation
    - 사용자가 아이템과 상호작용한 sequence를 통해 다음에 소비할 아이템을 예측
    - *한계점 (by multi-stage filtering)*
      - Cold start
      - Cascading bottlenecks

- LLM의 등장 → Generative Recommendation의 출현
  - ✓ 다음에 소비할 아이템의 ID(identifiers)를 autoregressive하게 생성하여 추천을 직접 수행

- ***효과적이고 효율적인 추천을 위해서는?*** Item ID의 표현이 핵심 역할

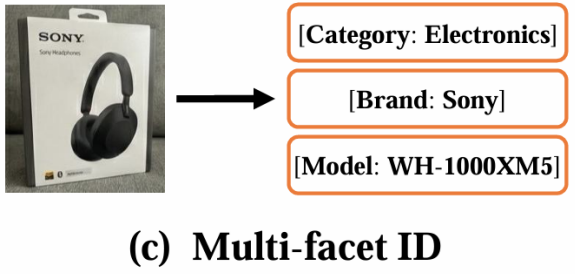
Background

Item ID의 유형

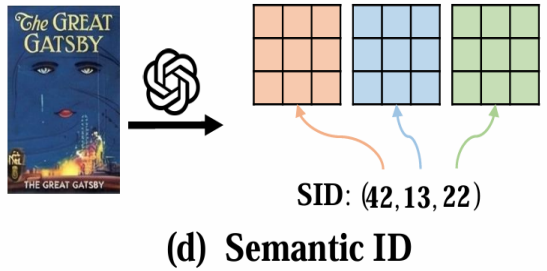
- Numeric ID
  - ✓ 아이템을 숫자 형태의 식별자로 표현

- Textual ID
  - ✓ 아이템이 가지는 텍스트 메타데이터(ex. Title, description 등)를 활용해 식별자로 표현

- Multi-facet ID
  - ✓ 아이템의 여러 속성 조합을 가지는 식별자



- Semantic ID
  - ✓ 아이템이 가진 실제 의미 정보를 압축한 짧은 "의미 코드 묶음"



[REF] A Survey of Item Identifiers in Generative Recommendation: Construction, Alignment, and Generation. *TechRxiv.* January 26, 2026

Background

Semantic ID

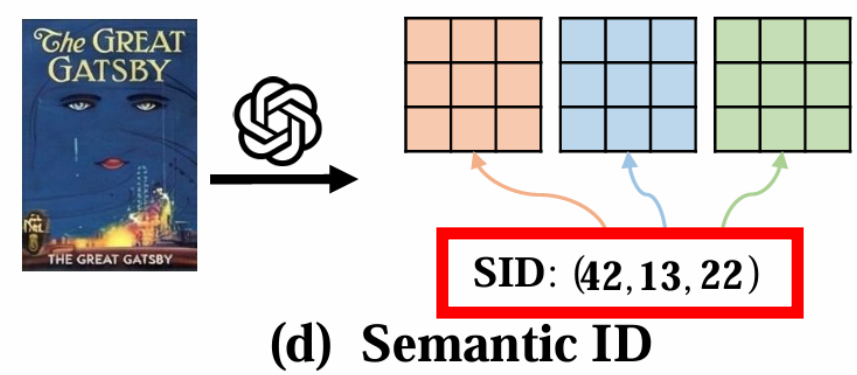
- **기존 Item ID들의 한계점을 극복하기 위해 등장**
  - ✓ *Numeric ID* → **의미가 없는 숫자 형태에 불과해 LLM이 이를 이해할 수 없음**
  - ✓ *Textual ID* → **실제 카탈로그 내 아이템과 일치하지 않을 수 있음**
    - 아이템의 제목(제품명, 책 이름, 뉴스 헤드라인 등)이 주로 활용
      - 같거나 비슷한 제목을 가진 아이템이 존재할 때
      - 아이템의 핵심 속성(브랜드, 모델, 기능 , 색상 등)을 담지 않을 때
      - 아이템의 텍스트 메타데이터가 길거나 비표준 어휘(약어, 오타, 광고문구 등)가 많을 때
  - ✓ *Multi-facet ID* → **여러 속성의 결합으로 입력 길이 및 토큰 비용 증가**

- **사용되는 데이터 양식**
  - ✓ 사용자-아이템의 상호작용 데이터
    - 사용자의 구매 이력, 평점, 좋아요 등
  - ✓ 텍스트 데이터
    - 제목, 설명, 리뷰, 카테고리(브랜드, 기능, 색상, …) 등
  - ✓ 멀티모달 데이터
    - 텍스트 외의 아이템의 이미지, 오디오, 비디오 등

Background

Semantic ID

- 기본 형태
  - ✓ 아이템이 가진 실제 의미 정보를 압축한 짧은 "의미 코드 묶음"
  - ✓ 정수 tuple 형태 → **순서가 있고, 값이 변하지 않음**



(d) Semantic ID

- Q. Numeric ID랑 어떤 차이점이 있을까?
  - ✓ *Numeric ID*는 아무 의미 없는 숫자
    - 은행, 카페 등의 대기표
  - ✓ *Semantic ID*는 "의미를 담은" 숫자
    - 주민등록번호

Background

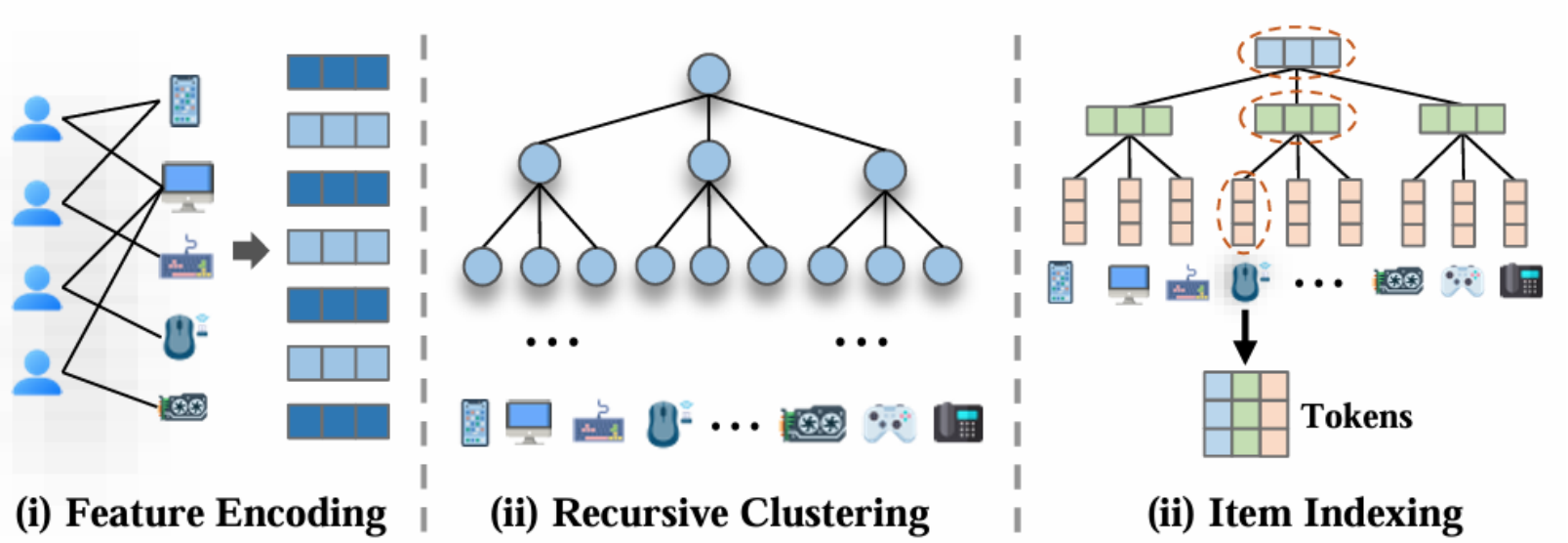
Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 1) Cluster 기반
    - 아이템을 상위 → 하위 구조로 클러스터링 해 트리(또는 계층)을 만들고,
    - 그 경로상의 노드 ID들을 semantic ID로 쓰는 방법
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드들을 골라 semantic ID를 만듦

Background

Semantic ID

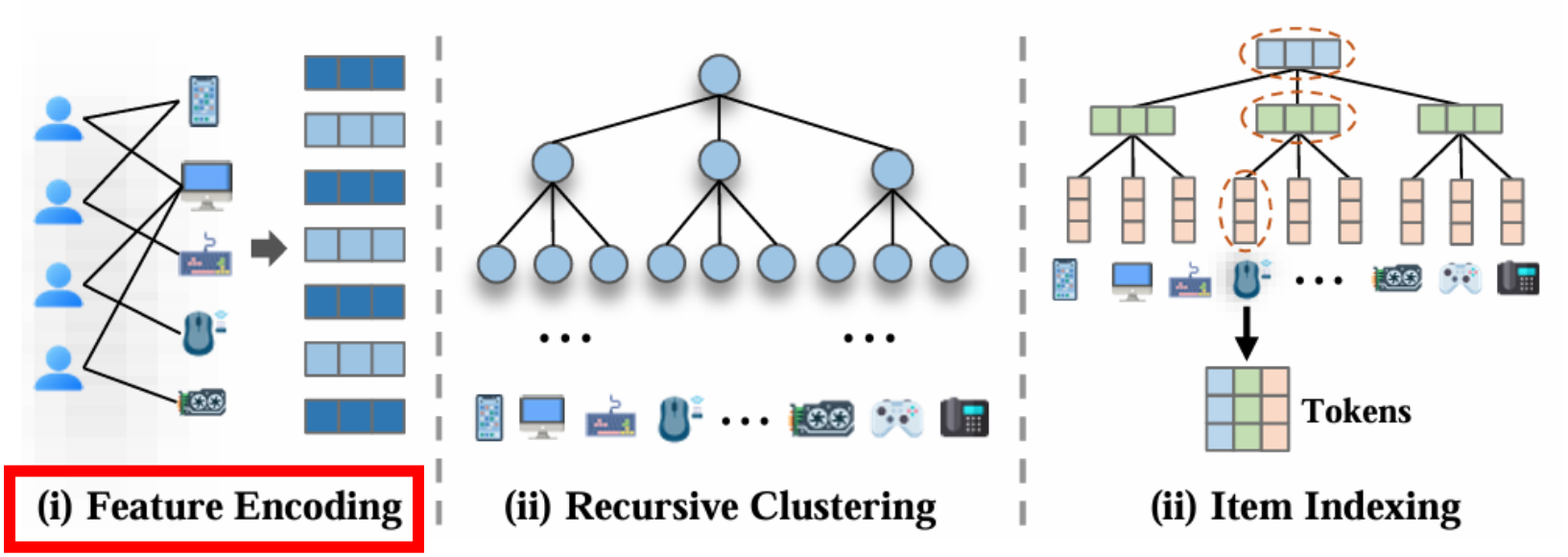
- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 1) Cluster 기반
    - 아이템을 상위 → 하위 구조로 클러스터링 해 트리(또는 계층)을 만들고,
    - 그 경로상의 노드 ID들을 semantic ID로 쓰는 방법



Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 1) Cluster 기반
    - 아이템을 상위 → 하위 구조로 클러스터링 해 트리(또는 계층)을 만들고,
    - 그 경로상의 노드 ID들을 semantic ID로 쓰는 방법

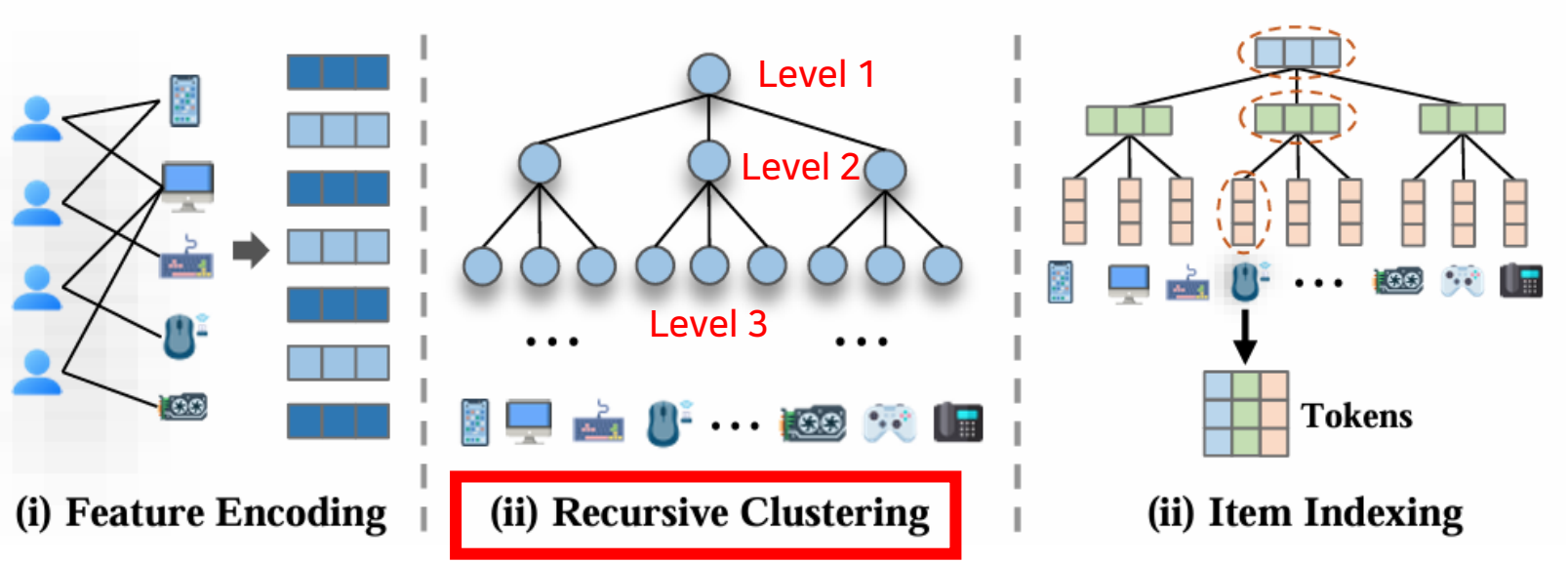


각 아이템에 대해
인코딩 과정을 거쳐
아이템 임베딩 생성

Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 1) Cluster 기반
    - 아이템을 상위 → 하위 구조로 클러스터링 해 트리(또는 계층)을 만들고,
    - 그 경로상의 노드 ID들을 semantic ID로 쓰는 방법

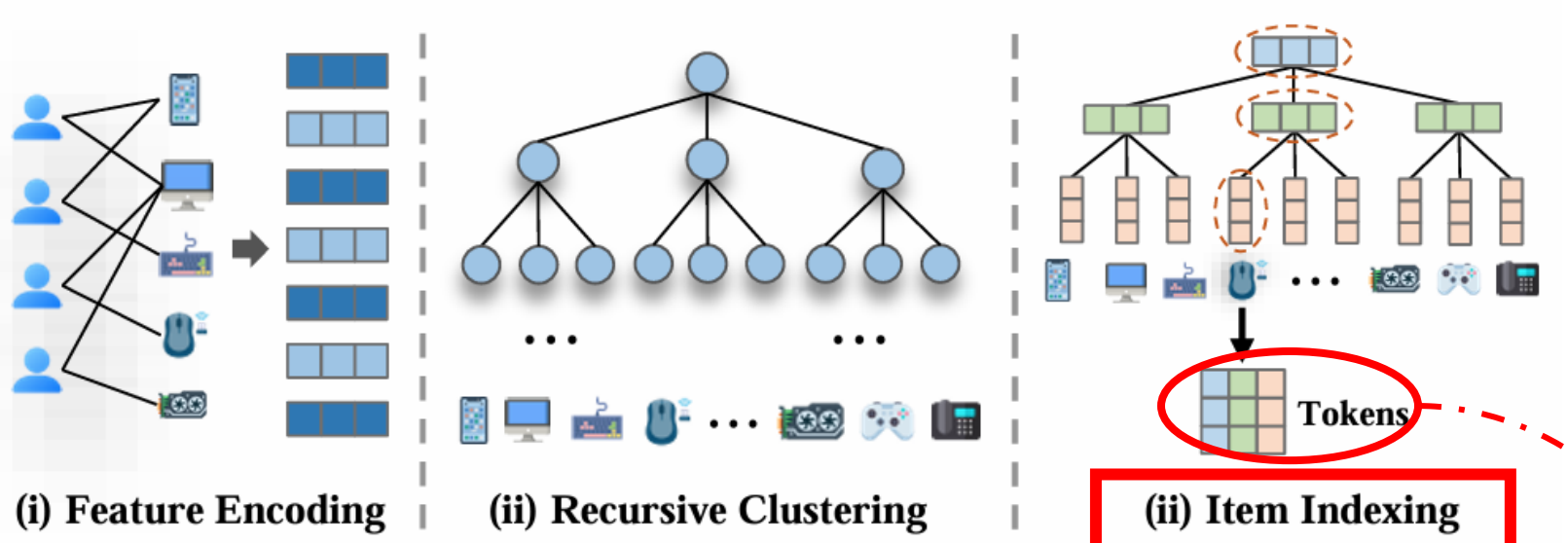


생성한 아이템 임베딩들을
재귀적으로 클러스터링해
레벨별 클러스터 집합을 생성

Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 1) Cluster 기반
    - 아이템을 상위 → 하위 구조로 클러스터링 해 트리(또는 계층)을 만들고,
    - 그 경로상의 노드 ID들을 semantic ID로 쓰는 방법

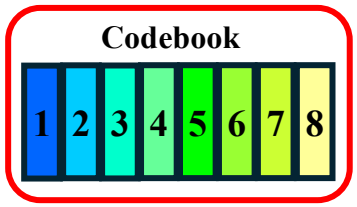


Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드워드를 골라 semantic ID를 만듦

  - ✓ 코드북이란?
    - 의미 벡터들의 사전
      - Ex) 단어장



    - [주의] 코드북은 하나가 될 수도, 여러 개가 될 수도 있음! (= 단어장이 하나 혹은 여러 개인 셈)

  - ✓ 코드워드(Codeword)란?
    - 사전의 개별 단어
      - Ex) 단어장 안의 단어 하나 = apple

Background

Semantic ID

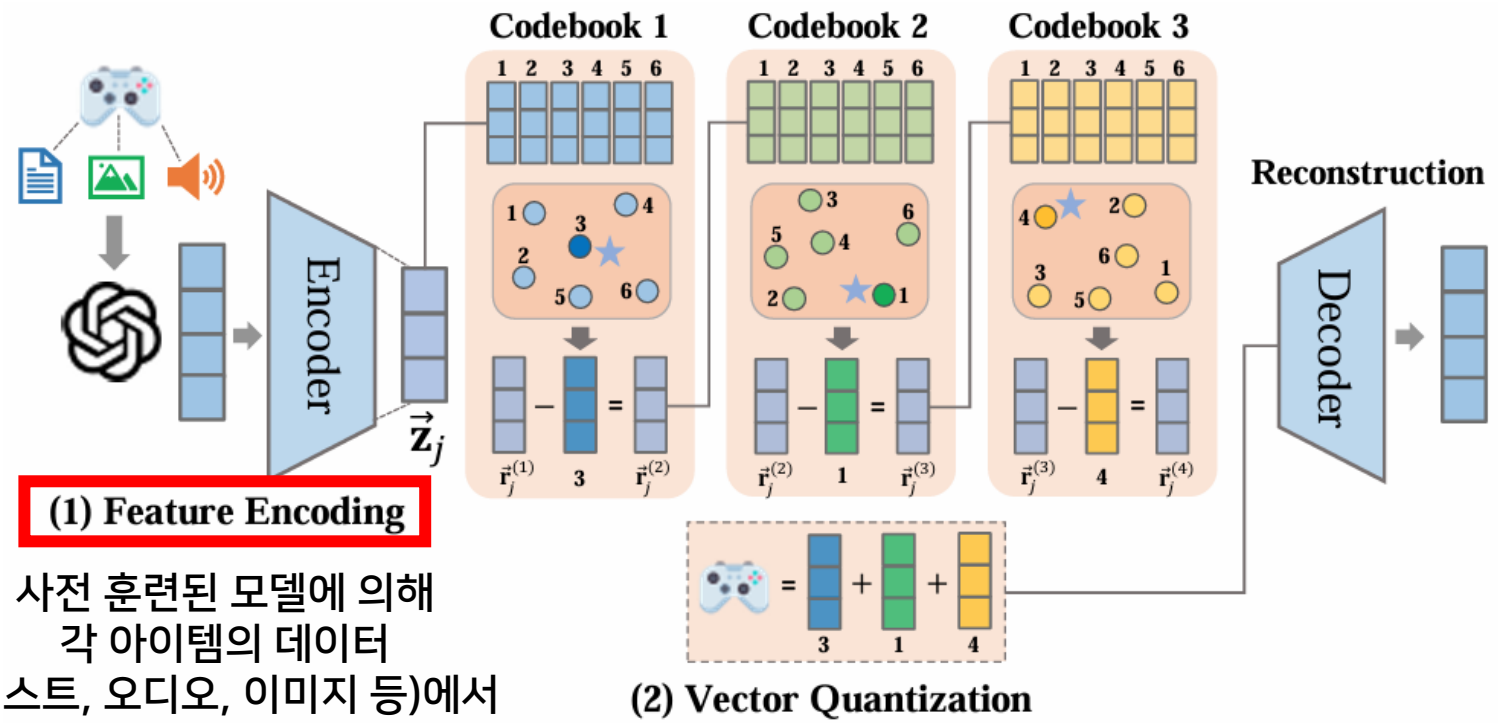
- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드워드를 골라 semantic ID를 만듦

- 대부분 코드북 기반 방법을 통해 Semantic ID를 생성
  - ✓ 클러스터링 기반 방법의 경우 (= Static)
    - 아이템 임베딩을 클러스터링 하여 고정된 상태
    - 한 번 고정되면 다시 조정되기 어려움
  - ✓ 반면, 코드북 기반의 방법 (= Dynamic, 학습 가능)
    - 코드북 벡터가 모델 학습 과정 안에서 함께 최적화 됨
    - 최적화 측면에서 현실적인 이득이 더 크다!

Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드워드를 골라 semantic ID를 만듦

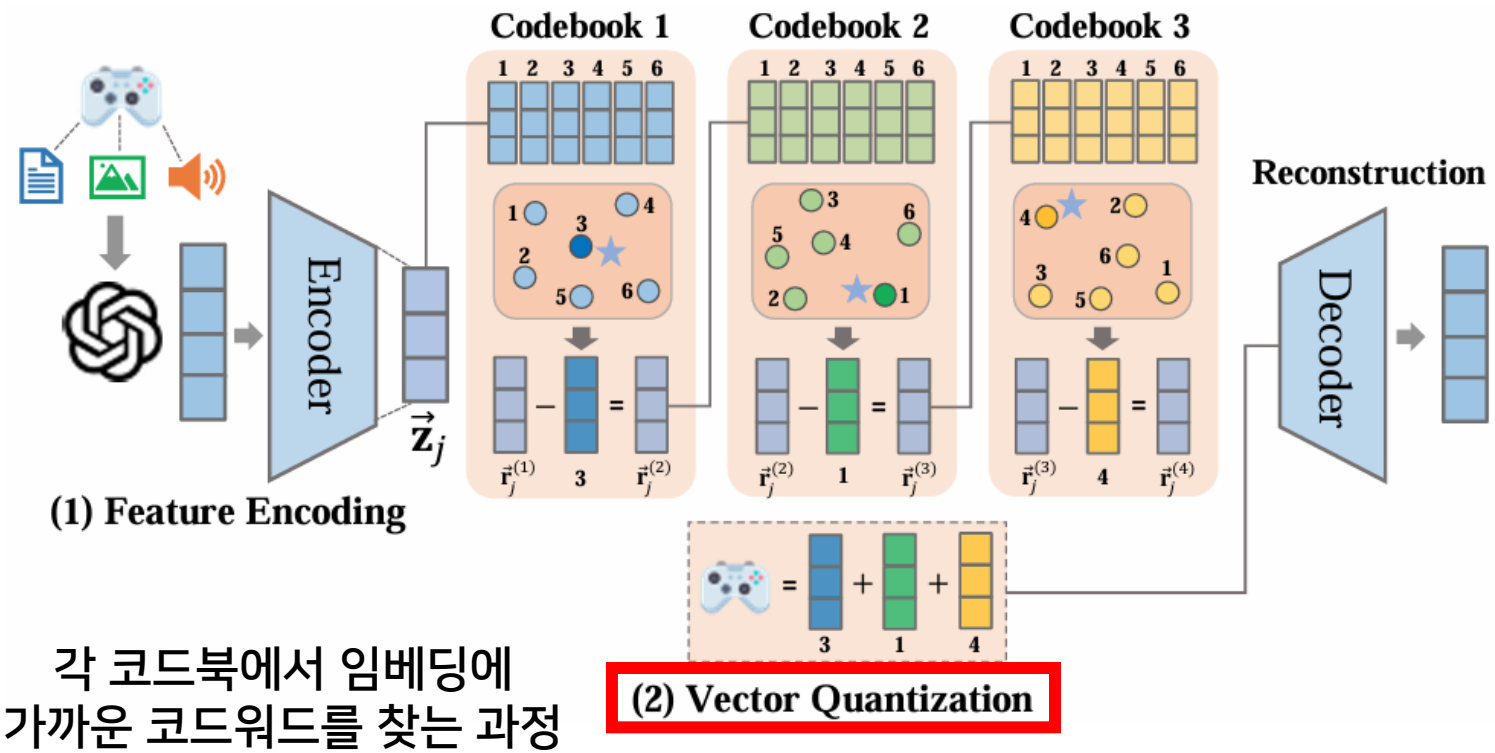


사전 훈련된 모델에 의해
각 아이템의 데이터
(텍스트, 오디오, 이미지 등)에서
추출된 input feature를
통합된 latent space의 임베딩으로 매핑

Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드워드를 골라 semantic ID를 만듦

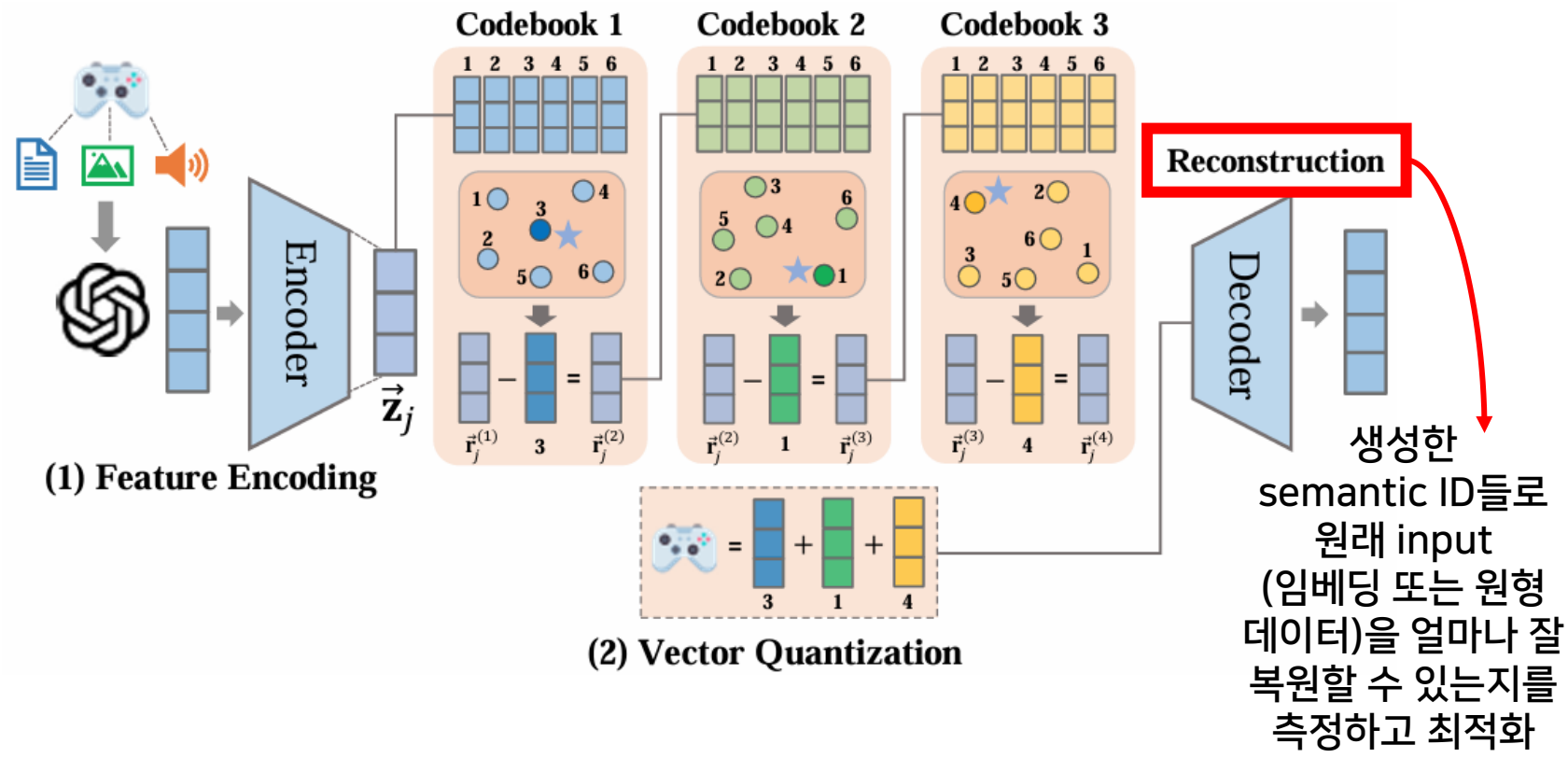


각 코드북에서 임베딩에
가까운 코드워드를 찾는 과정

Background

Semantic ID

- Item을 의미를 담은 숫자로 표현하기 위해서는?
  - ✓ 생성방식 2) Codebook 기반
    - 아이템 임베딩에 대해 코드북에서 가장 가까운 코드워드를 골라 semantic ID를 만듦



Background

코드북 기반의 Tokenizer

- 토크나이저의 종류에 따라 Semantic ID의 생성 과정이 달라짐

- Vector Quantization(벡터 양자화)이란?
  - ✓ 많은 수의 값을 대표되는 몇 개의 값으로 대체하는 기법 (데이터를 압축 및 단순화)
  - ✓ = 연속적인 벡터를 이산적인(discrete) 코드북 벡터로 바꾸는 것
    - 즉, encoder를 통과한 임베딩이 각 코드워드와의 거리를 계산한 뒤
    - 가장 가까운 코드워드로 대체되는 것

- VAE(Variational Autoencoders)이란?
  - ✓ 데이터를 압축했다가 다시 복원하도록 학습하는 확률적 오토인코더
    - *Encoder*: input 데이터 → latent space (연속 공간)
    - *Decoder*: latent space → 원래 데이터 복원

통합해 사용

Background

코드북 기반의 Tokenizer

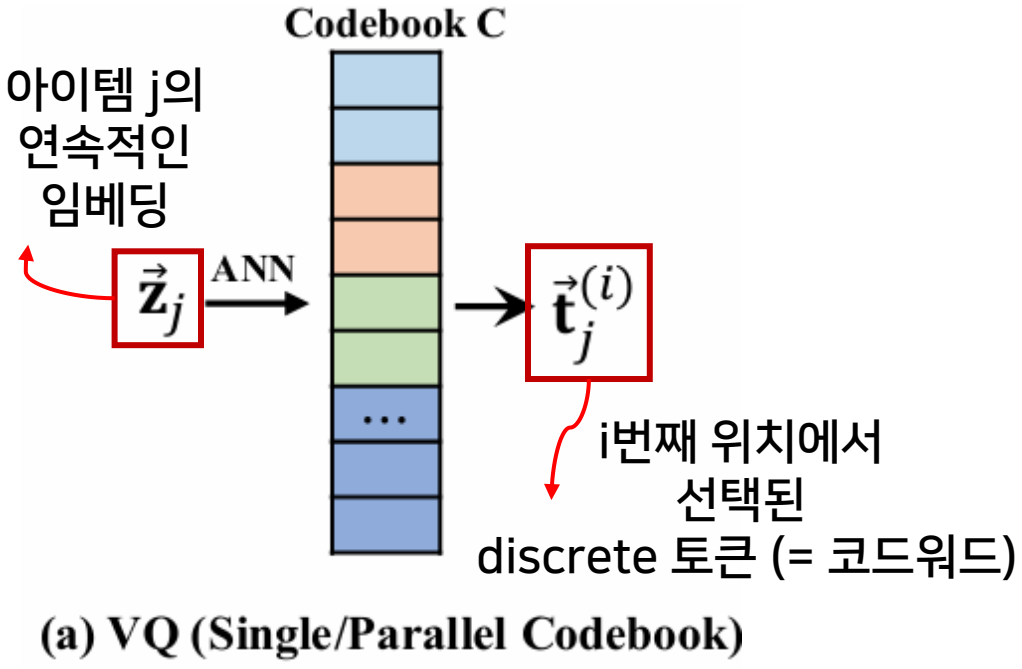
- 종류
  - ✓ (1) Vanilla Quantization(VQ)
  - ✓ (2) Residual Quantization(RQ)
  - ✓ (3) Product Quantization (PQ)
  - ✓ (4) Tree Quantization (TQ)

- Q. 이 외에는 없는지?
  - ✓ 있음.
  - ✓ Vector Quantization 기법은 *주로 신호처리 & 패턴인식 분야*에서 많이 활용
  - ✓ *오디오 도메인*에서 가장 적극적으로 발전되고 응용되고 있음
  - ✓ 이를 *최근(2023년 ~) 추천 시스템*의 semantic ID를 생성하는 데 사용하기 시작한 것
  - ✓ 따라서, 오디오 압축 도메인에서의 새로운 quantization 기법을 적용할 수 있음

Background

코드북 기반의 Tokenizer

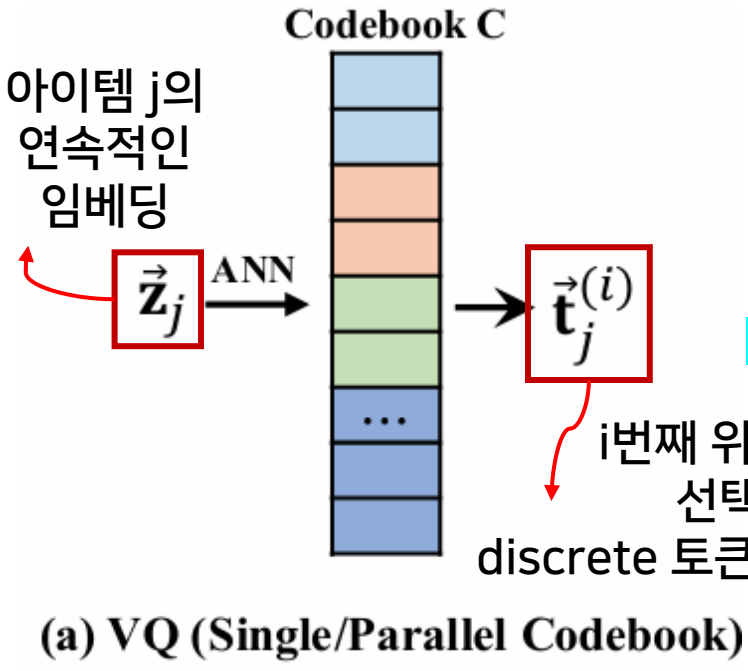
- (1) Vanilla Quantization(VQ) → *가장 기초*
  - ✓ 벡터 양자화를 VAE와 통합하는 것 = VQ-VAE (대표적인 VQ 기법)
  - ✓ Semantic ID 생성 방식
    - 주어진 아이템 임베딩에 대해 L개의 코드북 각각에서 가장 가까운 코드워드를 검색하여 생성



Background

코드북 기반의 Tokenizer

- (1) Vanilla Quantization(VQ) → *가장 기초*
  - ✓ 벡터 양자화를 VAE와 통합하는 것 = VQ-VAE (대표적인 VQ 기법)
  - ✓ Semantic ID 생성 방식
    - 주어진 아이템 임베딩에 대해 L개의 코드북 각각에서 가장 가까운 코드워드를 검색하여 생성



ANN: approximate nearest neighbor
(근사 최근접 탐색)

$$\vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}} \|\vec{\mathbf{z}}_j - \vec{\mathbf{c}}_k^{(\ell)}\|^2.$$

아이템 j의 연속적인 임베딩 - i번째 코드북 안의 k번째 코드워드

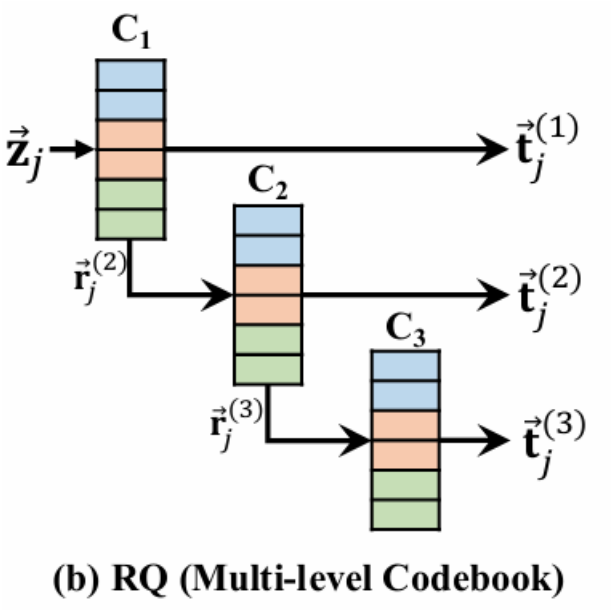
L2(유클리드) 거리의 제곱을
사용해서 구함

Background

코드북 기반의 Tokenizer

- (2) Residual Quantization(RQ) → 현재 가장 많이 쓰이는 방법
  - ✓ RQ-VAE에서 처음 제안된 기법으로,
  - ✓ TIGER(NeurIPS '23)에서 최초로 추천 시스템에 semantic id 개념이 도입됨
  - ✓ Semantic ID 생성 방식
    - 맨 처음 과정은 VQ-VAE와 동일 (= 가장 가까운 코드워드를 선택)
    - 그 다음 과정부터는 residual 벡터와 가장 가까운 코드워드를 선택하여 semantic id를 생성

$$\vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}} \|\vec{\mathbf{z}}_j - \vec{\mathbf{c}}_k^{(\ell)}\|^2.$$



(b) RQ (Multi-level Codebook)

최초 이후의 과정

$$\begin{cases} \vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}} \|\vec{\mathbf{r}}_j^{(\ell)} - \vec{\mathbf{c}}_k^{(\ell)}\|^2, \\ \vec{\mathbf{r}}_j^{(\ell+1)} = \vec{\mathbf{r}}_j^{(\ell)} - \vec{\mathbf{t}}_j^{(\ell)}. \end{cases}$$

L번째 단계에서의 residual(잔차) 벡터

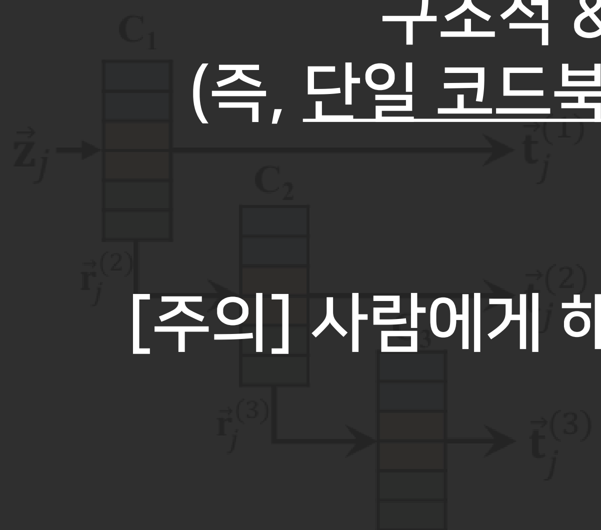
Background

코드북 기반의 Tokenizer

Q. 다층 구조의 코드북이면 생성된 semantic id는 계층적인(hierarchical) id인가?

→ Coarse-to-fine 방식으로 아이템 임베딩을
여러 단계로 나누어 표현하므로,
구조적 & 계층적인 성격을 가짐
(즉, <u>단일 코드북보다 표현력 측면에서 우수함</u>)

[주의] 사람에게 해석 가능한 계층적 의미와는 다름



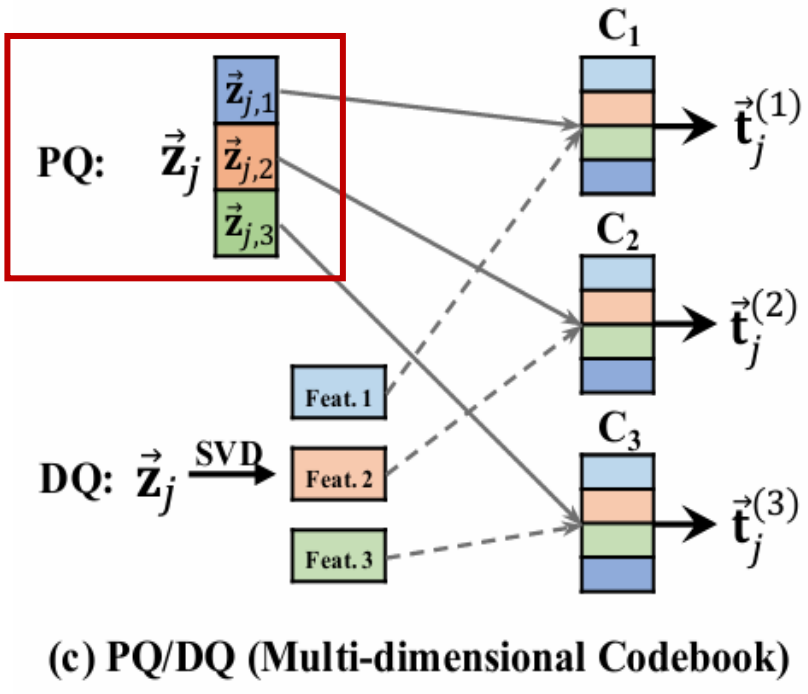
(b) RQ (Multi-level Codebook)

L번째 단계에서의 residual(잔차) 벡터

Background

코드북 기반의 Tokenizer

- (3) Product Quantization (PQ)
  - ✓ 아이템 임베딩을 겹치지 않는 서브벡터로 나눈 뒤,
  - ✓ 각 서브벡터를 독립적인 코드북에 양자화 하여 semantic id를 생성하는 방식



(c) PQ/DQ (Multi-dimensional Codebook)

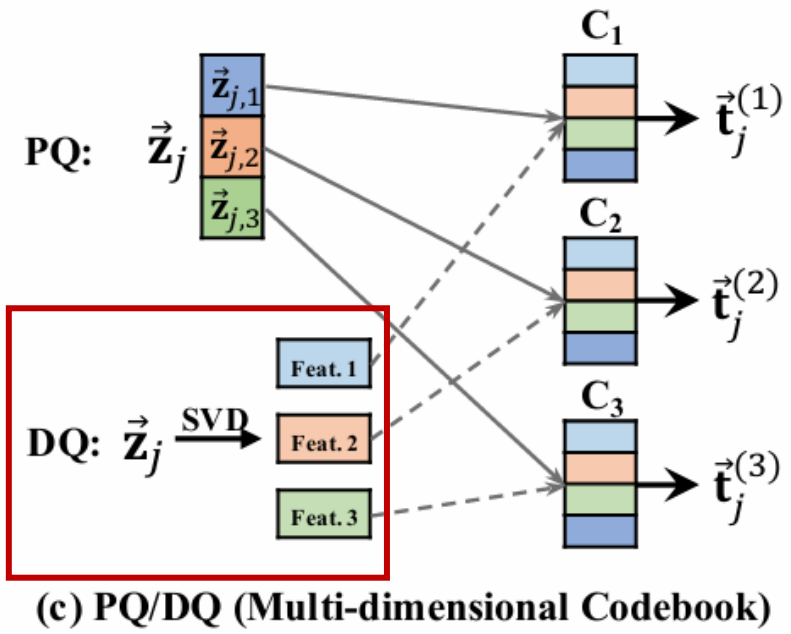
$$\vec{t}_j^{(\ell)} = \arg\min_{\vec{c}_k^{(\ell)} \in C^{(\ell)}} \|\vec{z}_{j,\ell} - \vec{c}_k^{(\ell)}\|^2.$$

즉, Level 1의 아이템 임베딩은
코드북 1의 가장 가까운 코드워드로 양자화
= 첫 번째 semantic id

Background

코드북 기반의 Tokenizer

- (4) Disentangled Quantization (DQ)
  - ✓ SVD(Singular Value Decomposition) 등으로
  - ✓ 아이템 임베딩의 의미 있는 기저(basis)를 찾아 그 기저들을 여러 개의 독립된 서브스페이스로 분할한 뒤, 각 서브스페이스를 별도 코드북으로 양자화하여
  - ✓ "서브스페이스별로 역할이 분리된(disentangled)" semantic id를 만드는 방식



(c) PQ/DQ (Multi-dimensional Codebook)

$$\min_{\{\mathbf{V}_1,\cdots,\mathbf{V}_L\}} \sum_{\ell_1 \neq \ell_2} \left(\sum_{\vec{\mathbf{v}}_k \in \mathbf{V}_{\ell_1}} \mathbf{\Sigma}[k,k]^2 - \sum_{\vec{\mathbf{v}}_k \in \mathbf{V}_{\ell_2}} \mathbf{\Sigma}[k,k]^2 \right),$$

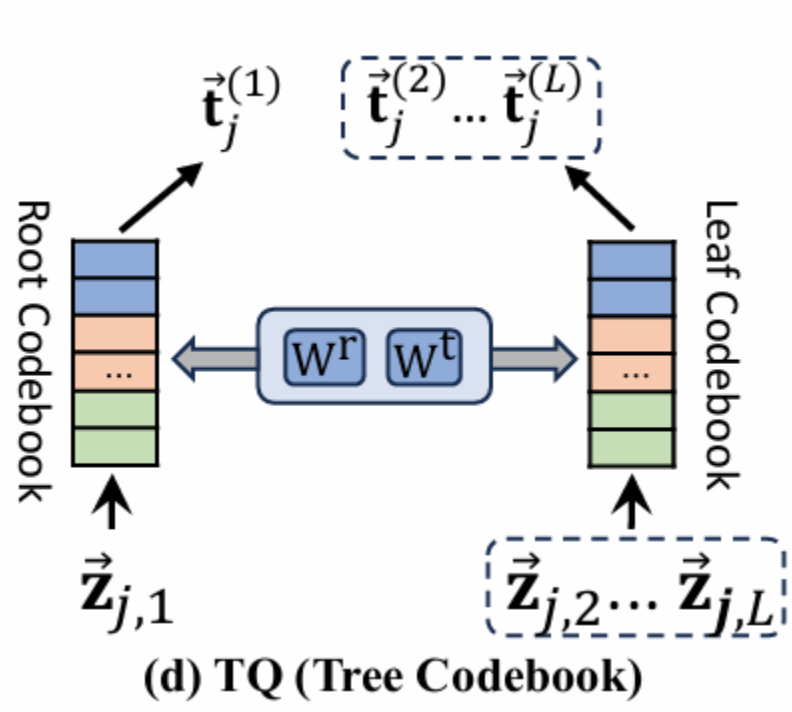
기저(basis): 어떤 벡터 공간의 모든 벡터를 선형결합으로 표현할 수 있는 최소 집합

서브스페이스가 데이터의 주요 변동 방향을 반영하므로 각 semantic id의 의미가 더 분리되어 해석성이 높아짐

Background

코드북 기반의 Tokenizer

- (5) Tree Quantization (TQ)
  - ✓ root와 leaf를 갖는 코드북 트리 구조
  - ✓ 첫 semantic id는 root 코드북에서, 이후 semantic id들은 leaf 코드북에서 선택됨



(d) TQ (Tree Codebook)

$$\hat{\vec{\mathbf{z}}}_j^{\ell+1} = \vec{\mathbf{z}}_j^{\ell+1} - \vec{\mathbf{z}}_j^{\ell} \text{ and } \hat{\vec{\mathbf{z}}}_j^{1} = \vec{\mathbf{z}}_j^{1},$$

첫 레벨에서는 입력 아이템 임베딩 그대로 사용
두 번째 레벨부터는 그 레벨의 임베딩에서
바로 이전 레벨의 임베딩을 뺀 residual을 사용

$$\vec{\mathbf{t}}_j^{(1)} = \arg\min_{\vec{\mathbf{c}}_k^{(1)} \in \mathbf{C}_{\text{root}}} \|\hat{\vec{\mathbf{z}}}_j - \vec{\mathbf{c}}_k^{(1)} \mathbf{W}_{\text{root}}\|^2,$$

$$\vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}_{\text{leaf}}} \|\hat{\vec{\mathbf{z}}}_j - \vec{\mathbf{c}}_k^{(\ell)} \mathbf{W}_{\text{leaf}}\|^2 \ \forall 2 \le \ell \le L.$$

각 레벨에 맞는 거리 척도를 학습해
서로 다른 레벨의 특성을 분리함

2. Related Works

- ✓ TIGER (NIPS ‘23)
- ✓ FACE (NIPS ‘25)

[2023][NIPS][TIGER]

Recommender Systems with Generative Retrieval

Shashank Rajput[*]
University of Wisconsin-Madison

Nikhil Mehta[*]
Google DeepMind

Anima Singh
Google DeepMind

Raghunandan Keshavan
Google

Trung Vu
Google

Lukasz Heldt
Google

Lichan Hong
Google DeepMind

Yi Tay
Google DeepMind

Vinh Q. Tran
Google

Jonah Samost
Google

Maciej Kula
Google DeepMind

Ed H. Chi
Google DeepMind

Maheswaran Sathiamoorthy
Google DeepMind

[2023][NIPS][TIGER]

핵심 아이디어

- 추천 시스템에 semantic id를 도입한 최초의 연구

- 사용한 codebook 기반의 토크나이저
  - ✓ RQ-VAE

- 코드북의 개수 (= semantic id의 길이)
  - ✓ 3개

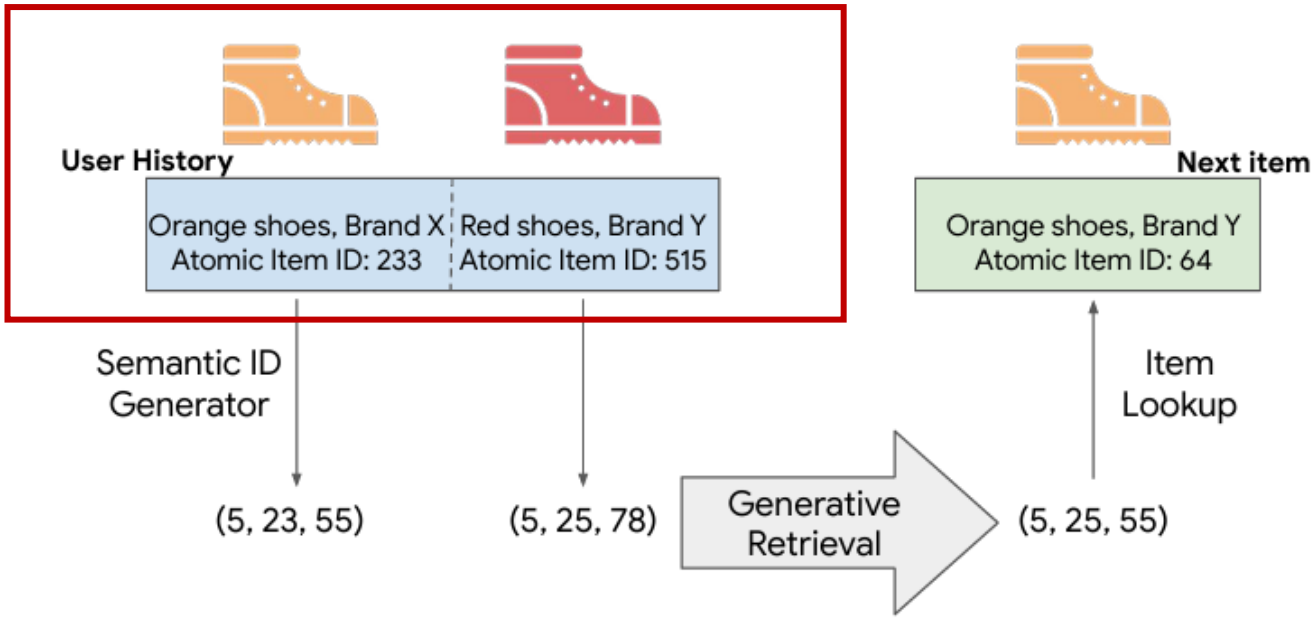
- semantic ID의 충돌(collision) 처리를 위해 네 번째 토큰을 추가
- 따라서 최종 semantic id의 길이는 4
  - ✓ 서로 다른 아이템이 같은 semantic id를 가지는 것을 말함
  - ✓ 이를 해결하기 위해 마지막에 작은 식별 토큰을 붙여 모든 아이템이 고유해지도록 만듦

- 코드워드의 개수
  - ✓ 256개

[2023][NIPS][TIGER]

핵심 아이디어

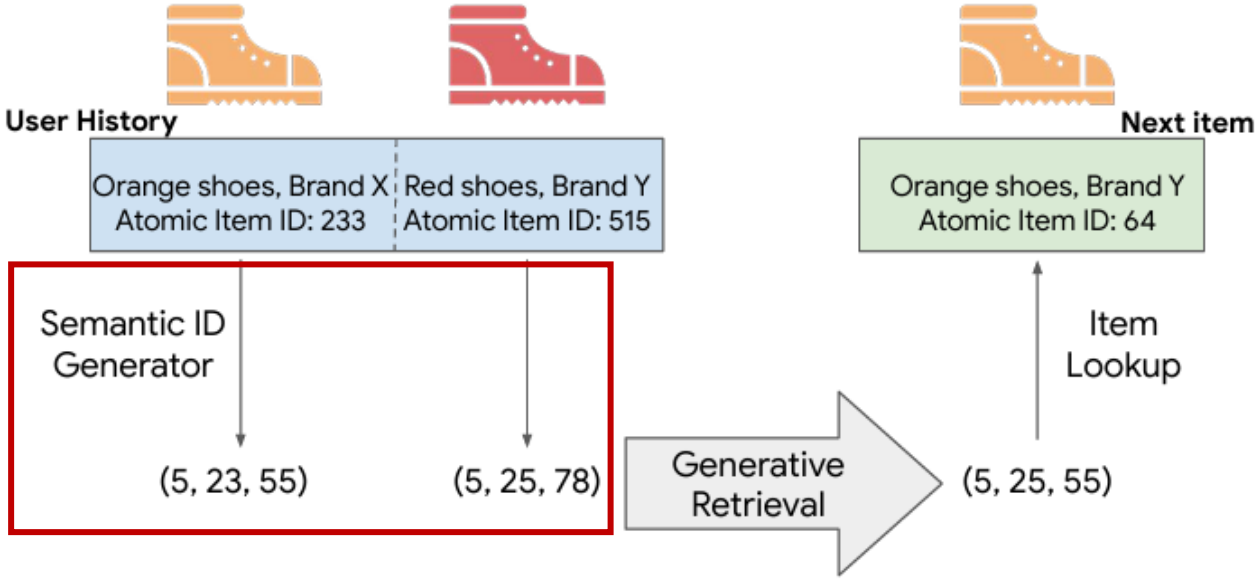
- (1) User History
  - ✓ 사용자가 과거에 본 아이템들을 시간순으로 보여줌 (Orange shoes → Red shoes)
  - ✓ 각 아이템에 기존 방식에서 쓰이던 임의의 정수(Atomic Item ID)가 붙어 있음



[2023][NIPS][TIGER]

핵심 아이디어

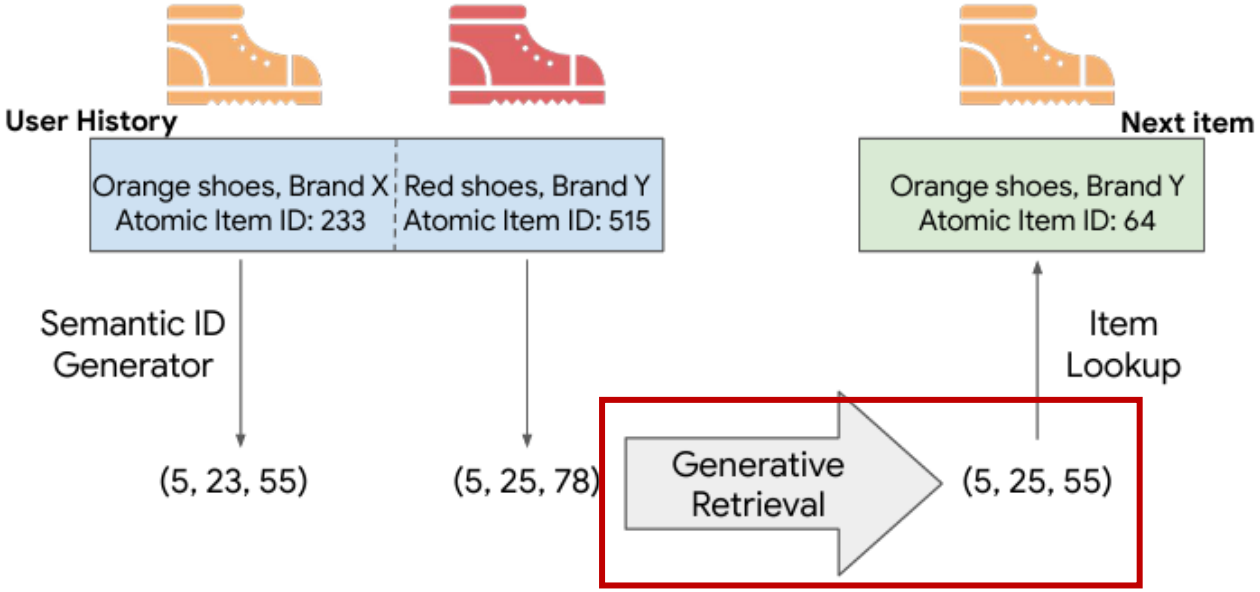
- (2) Semantic ID Generator
  - ✓ RQ-VAE 과정을 통해 semantic id를 생성



[2023][NIPS][TIGER]

핵심 아이디어

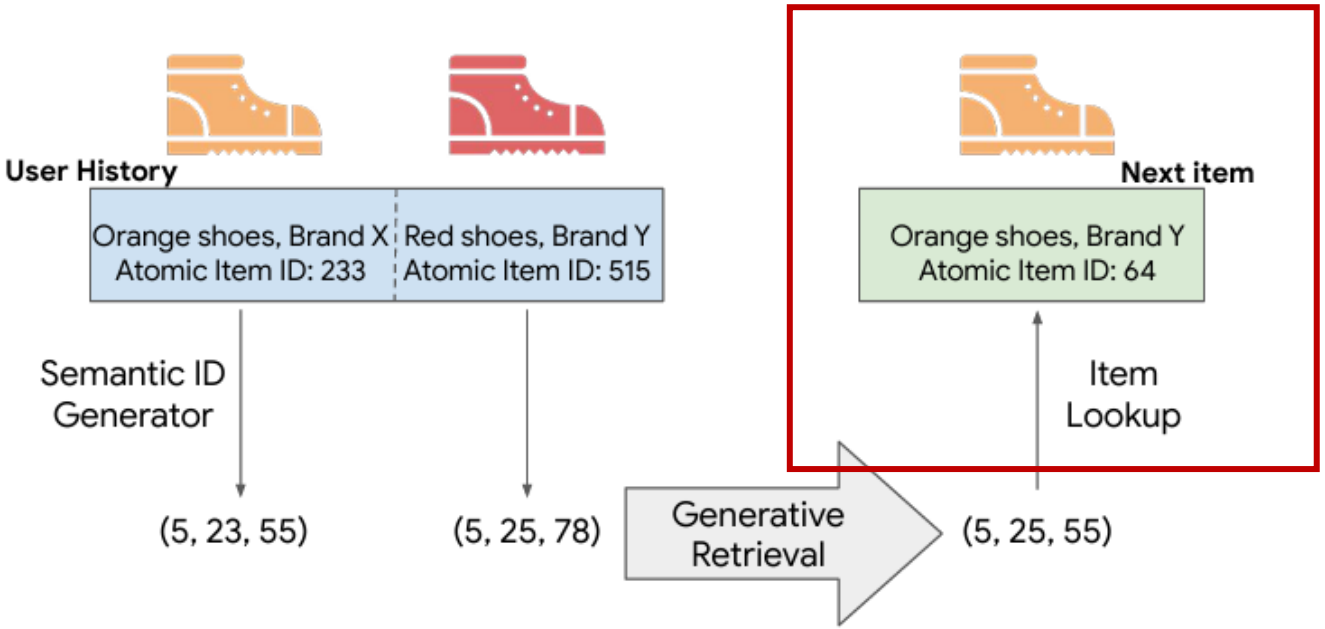
- (3) Generative Retrieval
  - ✓ 생성된 semantic id는 Transformer 기반의 seq-to-seq 모델에 입력됨
  - ✓ 모델은 다음 아이템의 semantic id 토큰을 토큰별로 디코딩해서 생성함



[2023][NIPS][TIGER]

핵심 아이디어

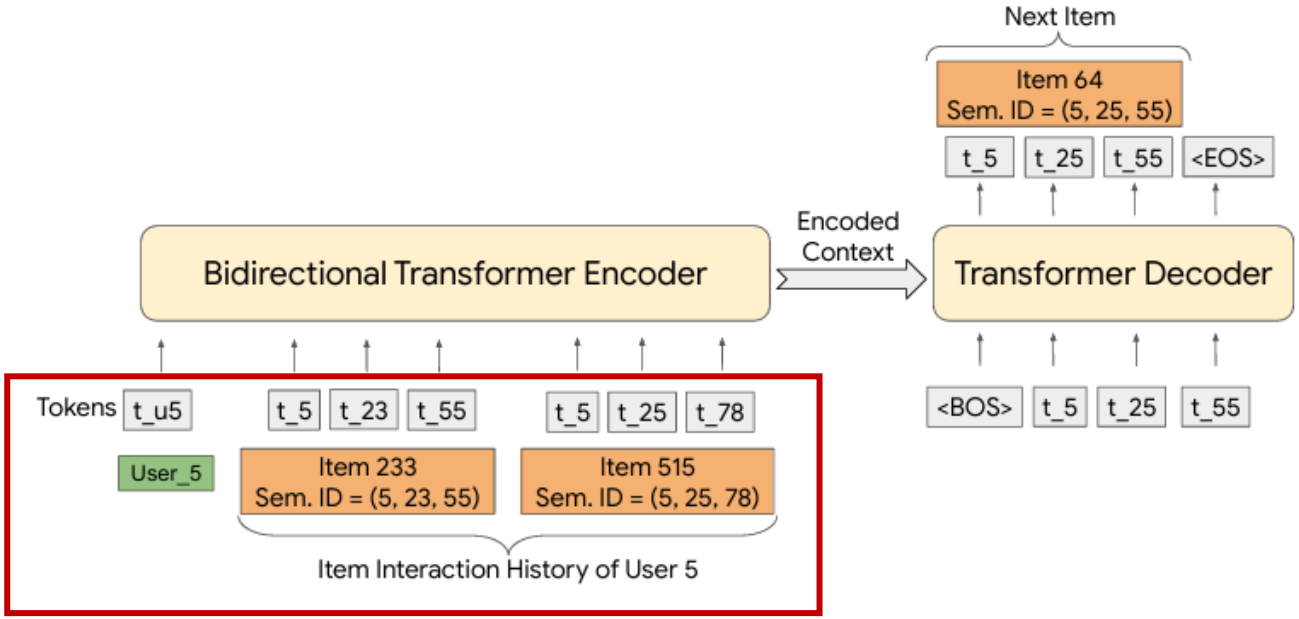
- (4) Item Lookup
  - ✓ 디코더가 생성한 semantic id 튜플을 lookup table로 조회
  - ✓ 실제 아이템(atomic ID)로 변환



[2023][NIPS][TIGER]

Overview

- Input (이미 semantic id는 생성되고 난 후)
  - ✓ 사용자 토큰(user_5) + 사용자의 아이템 상호작용 기록을 각 아이템의 semantic id 토큰을 펼쳐 순차적으로 나열함 (ex: user_5, t_5, t_23, …)

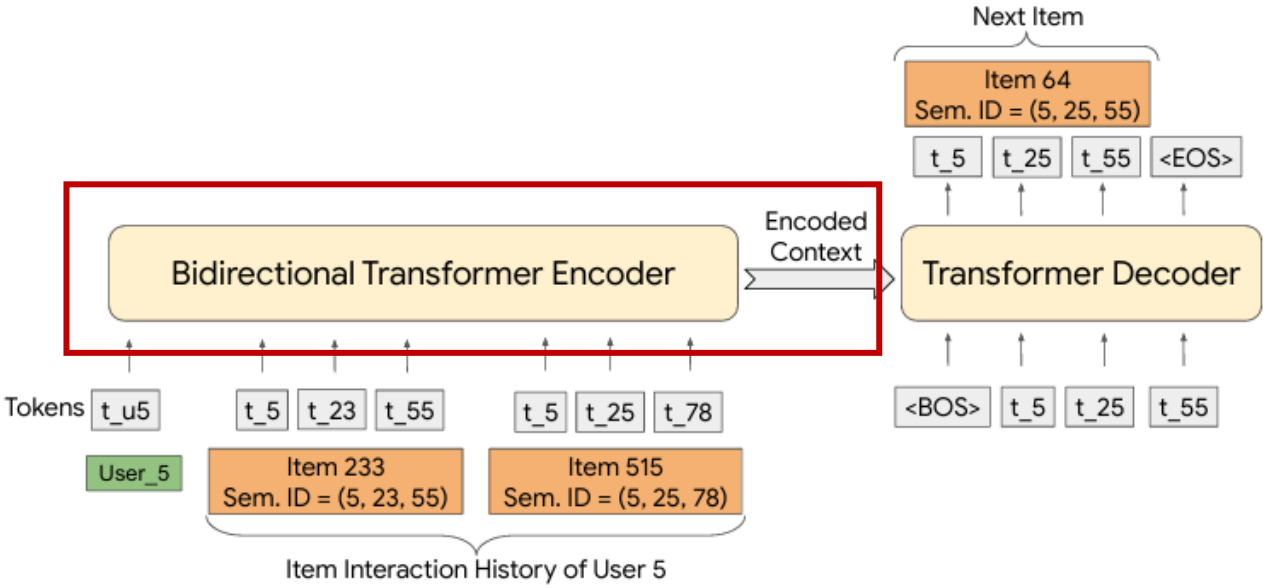


(b) Transformer based encoder-decoder setup for building the sequence-to-sequence model used for generative retrieval.

[2023][NIPS][TIGER]

Overview

- Bidirectional Transformer Encoder
  - ✓ Input을 양방향으로 인코딩하여 encoded context를 생성

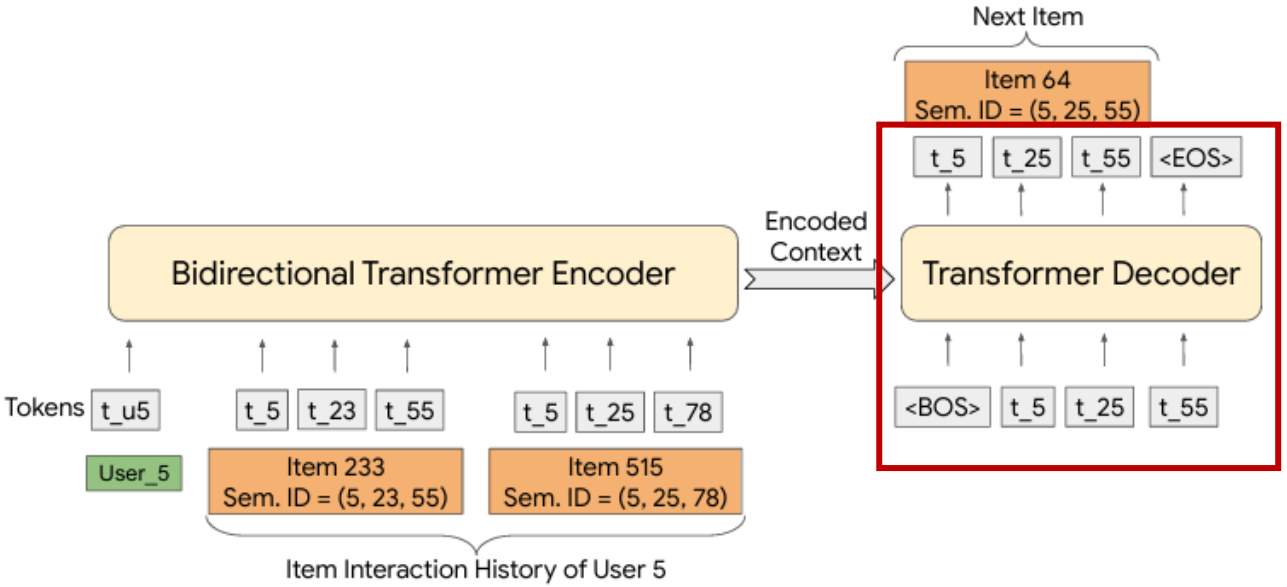


(b) Transformer based encoder-decoder setup for building the sequence-to-sequence model used for generative retrieval.

[2023][NIPS][TIGER]

Overview

- Transformer Decoder
  - ✓ autoregressive 방식으로 다음 아이템의 Semantic ID 토큰을 하나씩 생성
  - ✓ <BOS> → t_5 → t_25 → t_55 → <EOS>

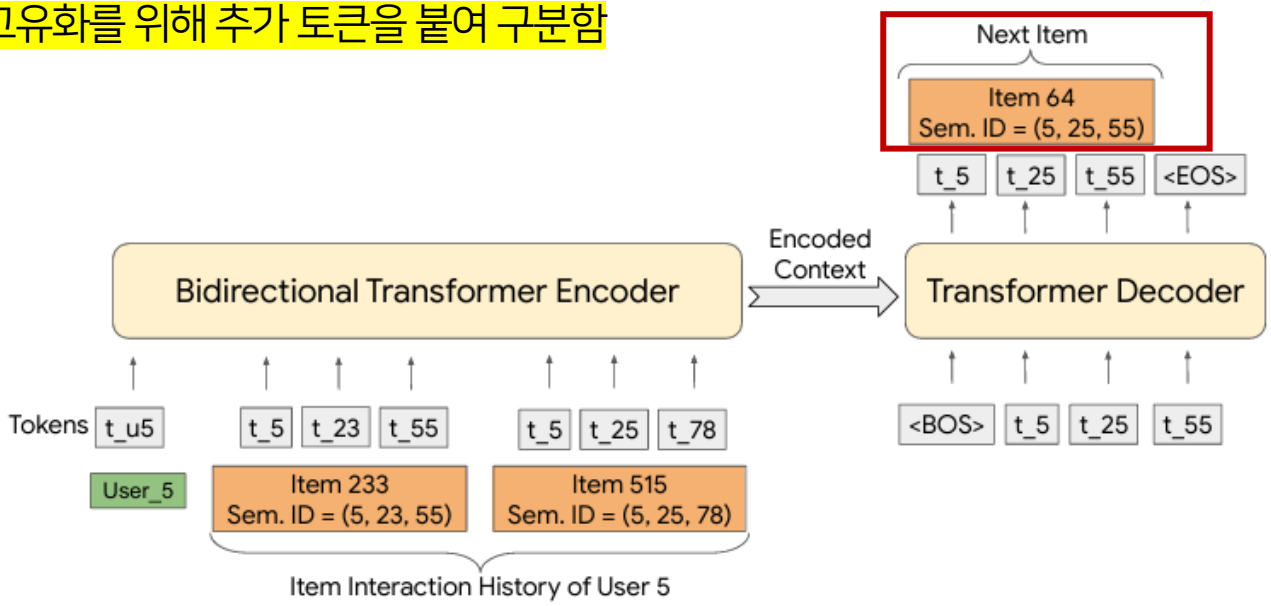


(b) Transformer based encoder-decoder setup for building the sequence-to-sequence model used for generative retrieval.

[2023][NIPS][TIGER]

Overview

- Output
  - ✓ 디코더가 생성한 Semantic ID 튜플을 아이템 조회 테이블(lookup)로 매핑하여 실제 추천 후보 반환
  - ✓ 만약 동일한 semantic id를 가진 여러 아이템이 있을 경우?
    - 고유화를 위해 추가 토큰을 붙여 구분함



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

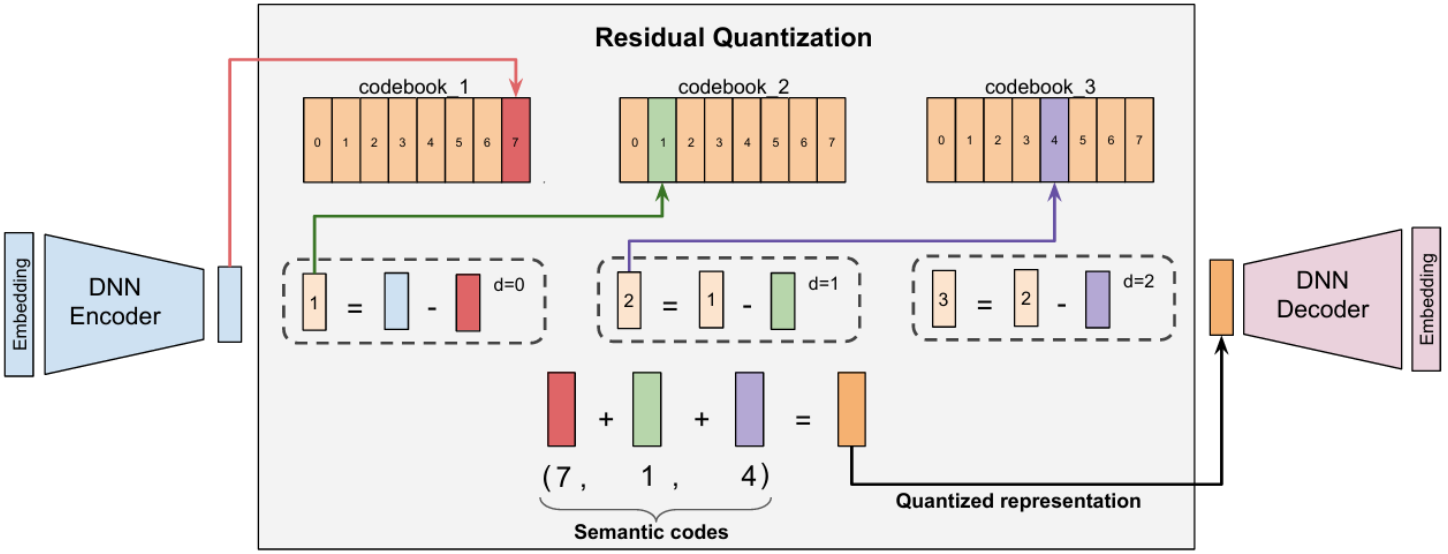
[2023][NIPS][TIGER]

RQ-VAE로 semantic id 생성

- Encoder를 통해 아이템 임베딩이 출력

- 레벨 d=0 (최초)
  - ✓ 아이템 임베딩은 코드북의 가장 가까운 코드워드를 찾아 semantic id가 됨

- 레벨 d=1
  - ✓ d = 0에서의 residual을 활용해 이와 가장 가까운 코드북 2의 코드워드를 찾아 semantic id가 됨 (반복)

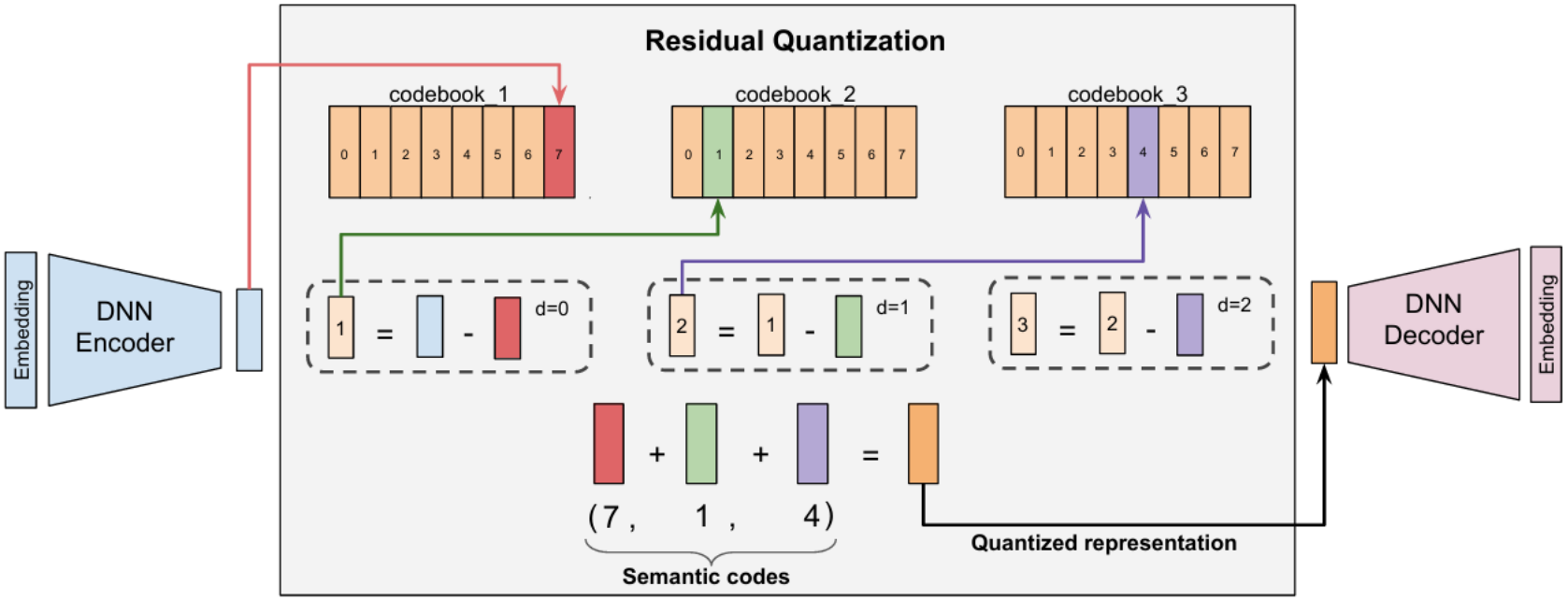


[2023][NIPS][TIGER]

RQ-VAE로 semantic id 생성

- Semantic id 완성
  - ✓ (7, 1, 4)

- Quantization representation (단순 sum)
  - ✓ 각 semantic id를 합해 디코더로 전달되어 input으로 들어왔던 아이템 임베딩을 재구성
  - ✓ *실제 TIGER 프레임워크에서는 quantization representation이 사용되는 것은 아님*

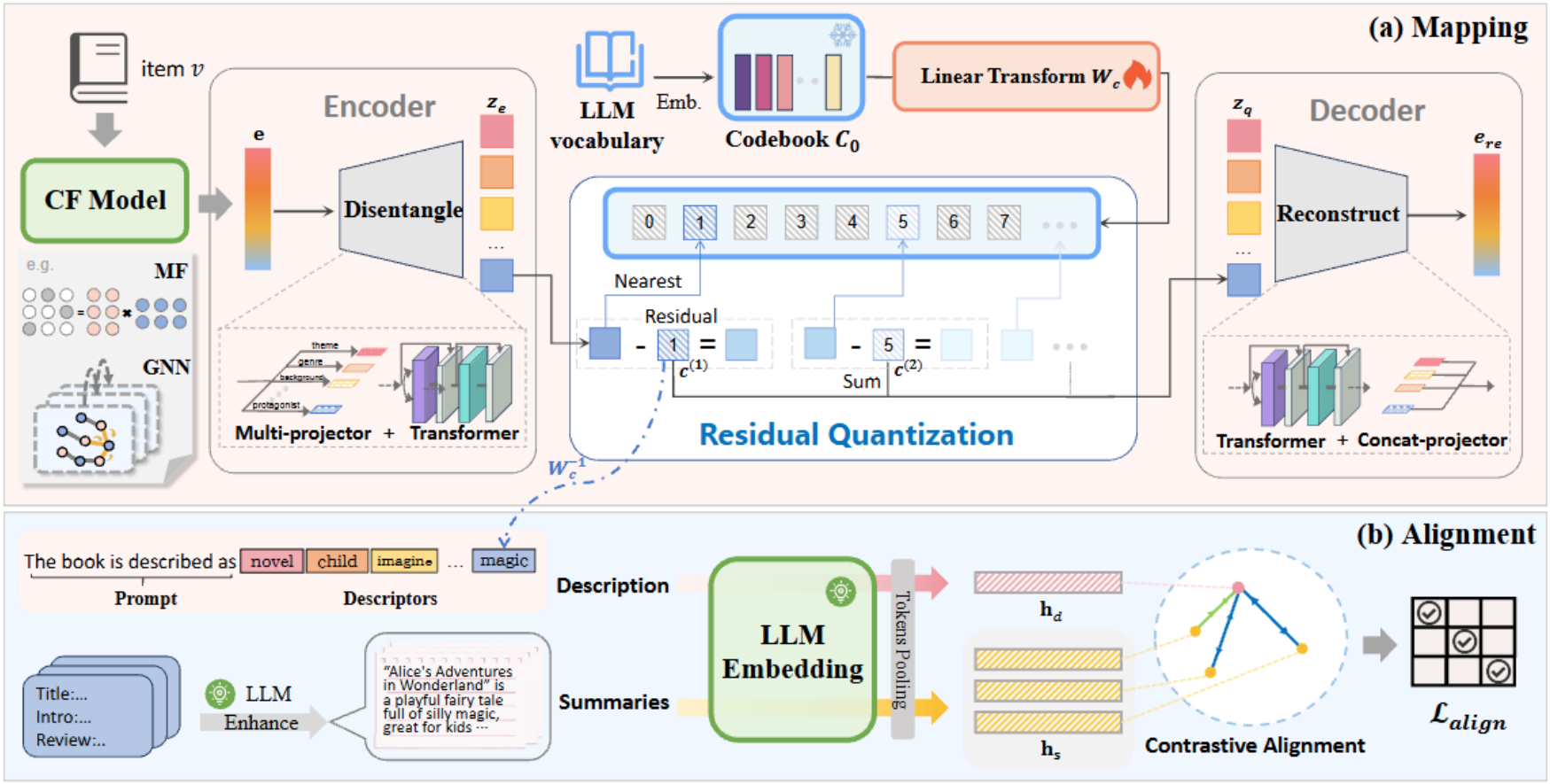


FACE: A General Framework for Mapping Collaborative Filtering Embeddings into LLM Tokens

Chao Wang[1†*] **Yixin Song**[1†] **Jinhui Ye**[2] **Chuan Qin**[3]
Dazhong Shen[4] **Lingfeng Liu**[1] **Xiang Wang**[1] **Yanyong Zhang**[1]
[1]School of Artificial Intelligence and Data Science, University of Science and Technology of China
[2]Hong Kong University of Science and Technology (Guangzhou)
[3]Computer Network Information Center, Chinese Academy of Sciences
[4]Nanjing University of Aeronautics and Astronautics
wangchaoai@ustc.edu.cn, yixinsong@mail.ustc.edu.cn,
jye624@connect.hkust-gz.edu.cn, {chuanqin0426,dazh.shen}@gmail.com,
lfliu@mail.ustc.edu.cn, xiangwang1223@gmail.com, yanyongz@ustc.edu.cn

[2025][NIPS][FACE]

핵심 아이디어



3. Challenges and Future Directions

Challenges and Future Directions

- **생성한 Semantic id의 포괄적인 벤치마킹 부족**
  - ✓ 공정하고 엄격한 평가 및 심층 분석을 위한 포괄적인 벤치마크가 여전히 부족한 상황

- **품질 평가의 부족**
  - ✓ Semantic id의 체계적인 분석, 정량적 평가 및 비교가 부족

- **Hierarchical vs. Flat**
  - ✓ 이 둘을 통합한 하이브리드 조합의 semantic id 연구는 아직 X

- **Order-dependent vs. Order-agnostic**
  - ✓ 대부분의 semantic id는 순서에 의존하고 있음
  - ✓ 아직까지 순서에 의존하지 않는 semantic id 연구는 드문 편

Thank you for listening

Hae-Yoon Koo (hykoo@konkuk.ac.kr)

Graph & Language Intelligence Laboratory
Konkuk University

2026.02.12