Recommender Systems

Lecture 4: Generative approaches to recommender systems

Yubao Tang, Kidist Amde Mekonnen
University of Amsterdam

June 5, 2025

y.tang3@uva.nl, k.a.mekonnen@uva.nl

Where are we?

- **Lecture 1**
  - Introduction to RecSys
- **Lecture 2**
  - Evaluation in RecSys
- **Lecture 3**
  - a. Sequential RecSys
  - b. Large Language Model-based RecSys
- **Lecture 4**
  - a. Generative models in RecSys
  - b. Case studies in GenRec

Acknowledgements

This lecture is based on a number of published papers.

Part 1
Generative models in recommender system

Generative models in RecSys

- Generative models are used in multiple ways:
  - VAE for embedding generation
  - Language models for review/explanation generation
  - RAG for text-to-item generation

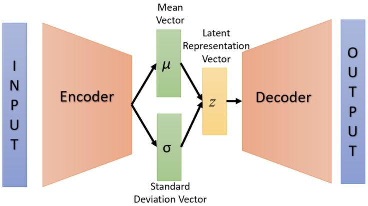


Figure 1: VAE-based recommender systems [Fraihat et al., 2024]

Generative models in RecSys

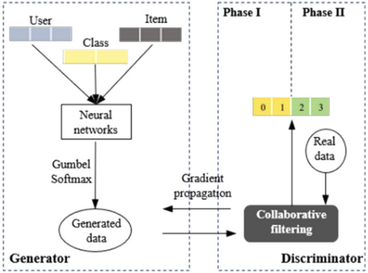


Figure 2: The generative model constructs data [Wang et al., 2019]

- The core mechanism of most modern recommendation models—including matrix factorization and deep learning approaches—relies on **matching user and item embeddings** in a latent space

Limitations of embedding-based recommenders

- Tightly coupled with the indexing structure
- Matching is limited to item-level semantic similarity (e.g., a simple matching function over fixed-length embedding vectors)

Is this simple fixed-length embedding interaction truly enough for recommendation?

Terminology clarification

- **Embedding-based methods using generative structure models (GenStruct):** Use generative models to learn user/item representations or assist in reranking, but still rely on embedding matching
- **Generative recommendation (GenRec):** User history $\rightarrow$ next item identifier
- **Generative information retrieval (GenIR):** Query $\rightarrow$ relevant document identifier
- *Note:* Some papers use "generative recommendation" broadly to include both GenStruct and GenRec above

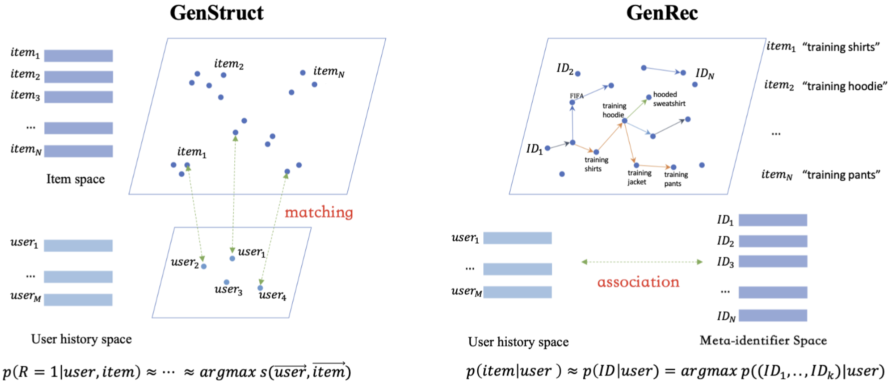


Figure 3: Mechanism: matching vs. association / generation

Definition: Generative recommendation (GenRec)

- **Definition:** A recommendation paradigm where the model directly **generates item identifiers** (ID) given a user's interaction history [Rajput et al., 2023]
- The task is cast as a sequence-to-sequence generation problem:
  - Input: user interaction sequence
  - Output: next item ID
- No explicit index is required

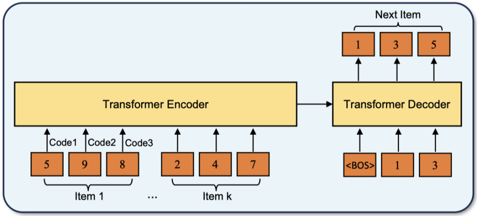


Figure 4: The GenRec model generates the next item ID [Zhu et al., 2024]

From generative retrieval to generative recommendation

- **Generative information retrieval (GenIR):** A GenIR model directly generates relevant document identifiers in a sequence-to-sequence fashion, for a query

From generative retrieval to generative recommendation

- **Generative information retrieval (GenIR):** A GenIR model directly generates relevant document identifiers in a sequence-to-sequence fashion, for a query
- **Why it matters:**
  - Moves beyond index-based retrieval
  - Enables end-to-end learning with strong generalization

From generative retrieval to generative recommendation

- **Generative information retrieval (GenIR):** A GenIR model directly generates relevant document identifiers in a sequence-to-sequence fashion, for a query
- **Why it matters:**
  - Moves beyond index-based retrieval
  - Enables end-to-end learning with strong generalization
- **Inspiration for GenRec:**
  - Items $\equiv$ documents
  - User history $\equiv$ query
  - Next-item prediction $\equiv$ identifier generation

Advantages of GenRec [Rajput et al., 2023]

- **Unified paradigm:** Item corpus is indexed implicitly via a generative model
- **Flexible conditioning:** Easily incorporate user history, context, and auxiliary information as prompt input
- ...

Q&A

Questions, . . .

Basic workflow of GenRec models

Basic pipeline

- Input: tokenized user interaction history
- Output: item ID via autoregressive decoding
- Model architecture: encoder-decoder

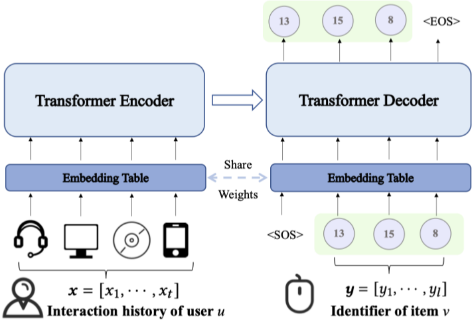


Figure 5: GenRec pipeline [Si et al., 2024]

Identifiers in GenRec

- **What is an identifier?** A unique, textual representation of an item (e.g., "item_12345"), used as the decoding target

Identifiers in GenRec

- **What is an identifier?** A unique, textual representation of an item (e.g., "item_12345"), used as the decoding target
- **Why identifiers?**
  - Serve as a symbolic reference to items in the catalog
  - Make it possible to reframe recommendation as a sequence generation problem

Identifiers in GenRec

- **What is an identifier?** A unique, textual representation of an item (e.g., "item_12345"), used as the decoding target
- **Why identifiers?**
  - Serve as a symbolic reference to items in the catalog
  - Make it possible to reframe recommendation as a sequence generation problem
- **Output format:** Generated as token sequences (e.g., "item", "_", "12", "345")

Examples

- Random identifiers [Geng et al., 2022]
  - Semantically meaningless: the token structure contains no information about item content (e.g., "item_12345")
  - The model should learn to map user history to arbitrary string tokens
  - Harder to generalize, especially to unseen or cold-start items [Rajput et al., 2023]
- Semantic identifiers [Rajput et al., 2023]
  - Residual quantization codes
  - Improving learnability and generalization

Training

- Given the user history, the model learns to maximize the likelihood of the next item identifier

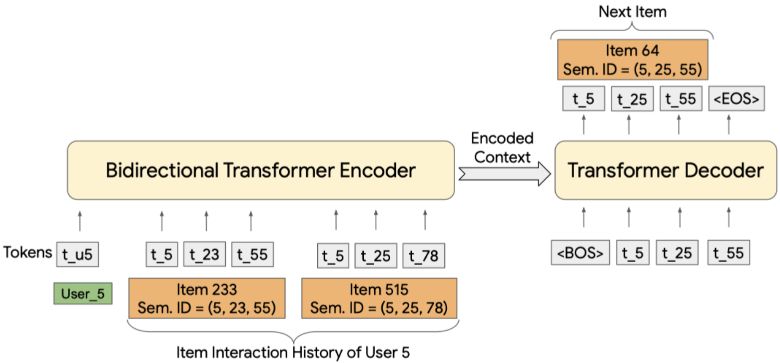


Figure 6: Training illustration [Rajput et al., 2023]

Inference

- Beam / greedy search

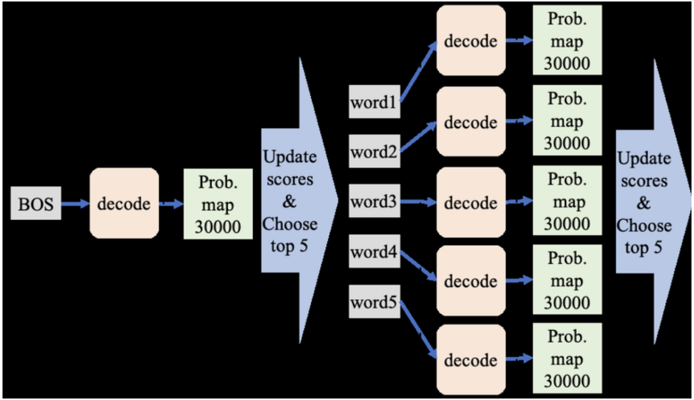


Figure 7: Beam search illustration [Li et al., 2020]

Inference

- Constrained beam search: ensure only valid IDs
- Prefix tree structure:
  - Nodes are annotated with tokens from the vocabulary
  - For each node, its children indicate all the allowed continuations from the prefix defined traversing the trie from the root to it

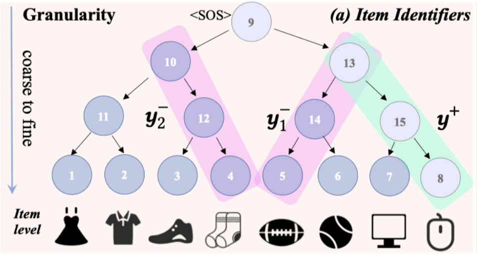


Figure 8: Constrained decoding with a prefix tree [Si et al., 2024]

Recent advances in GenRec

Roadmap

P5

[Geng et al., 2022]

(Randomly IDs)

Roadmap

P5 [Geng et al., 2022] → **TIGER** [Rajput et al., 2023]

(Randomly IDs) (RQ codes as IDs)

Roadmap

P5 [Geng et al., 2022] → **TIGER** [Rajput et al., 2023] → **SEATER** [Si et al., 2024]

(Randomly IDs)

(RQ codes as IDs)

(Tree-structured IDs
and contrastive ranking)

Roadmap

P5

[Geng et al., 2022]

(Randomly IDs)

TIGER

[Rajput et al., 2023]

(RQ codes as IDs)

SEATER

[Si et al., 2024]

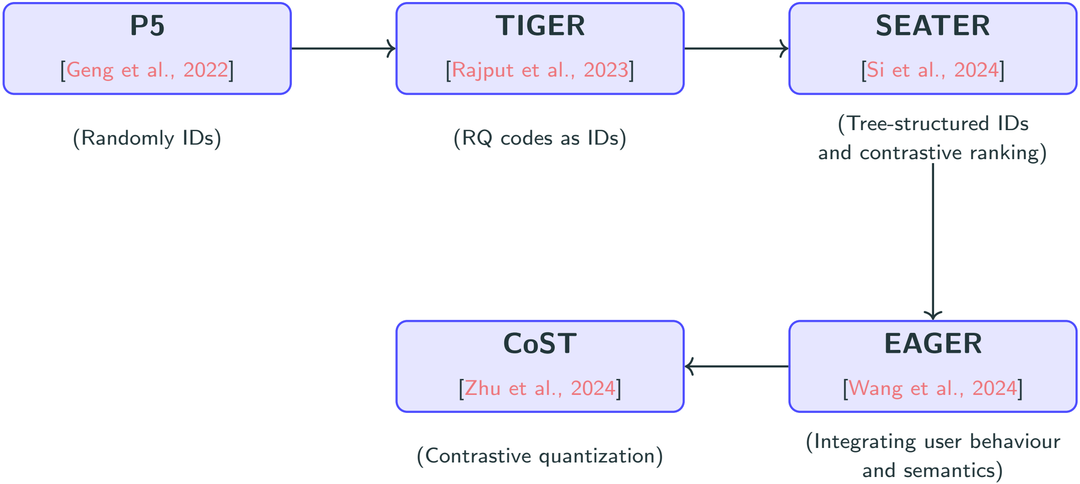
(Tree-structured IDs
and contrastive ranking)

EAGER

[Wang et al., 2024]

(Integrating user behaviour
and semantics)

Roadmap


P5
[Geng et al., 2022]
(Randomly IDs)
TIGER
[Rajput et al., 2023]
(RQ codes as IDs)
SEATER
[Si et al., 2024]
(Tree-structured IDs
and contrastive ranking)
EAGER
[Wang et al., 2024]
(Integrating user behaviour
and semantics)
CoST
[Zhu et al., 2024]
(Contrastive quantization)

Roadmap

P5
[Geng et al., 2022]
(Randomly IDs)

→

TIGER
[Rajput et al., 2023]
(RQ codes as IDs)

→

SEATER
[Si et al., 2024]
(Tree-structured IDs
and contrastive ranking)

↓

EAGER
[Wang et al., 2024]
(Integrating user behaviour
and semantics)

←

CoST
[Zhu et al., 2024]
(Contrastive quantization)

←

MQL4GRec
[Si et al., 2024]
(Generative multi-modal
recommendation)

Part 2
Case studies in GenRec: TIGER and SEATER

Recommender systems with generative retrieval

TIGER [Rajput et al., 2023]

- **T**oken-based **I**tem **G**eneration for **E**nd-to-end **R**ecommendation (TIGER)
- **Key idea:** Reformulate recommendation as sequence-to-sequence generation
- Uses a shared encoder-decoder architecture, trained to decode item IDs

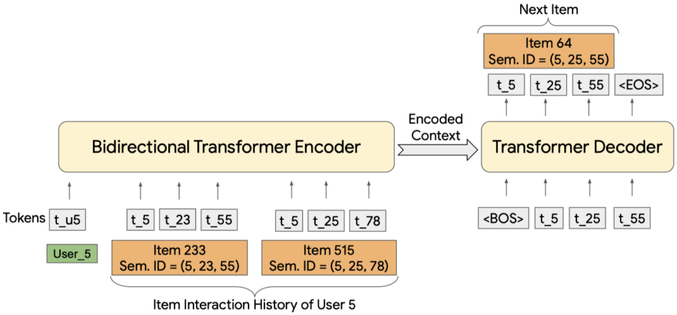


Figure 9: TIGER overview [Rajput et al., 2023]

Identifier

- **Goal:** Represent each item with a structured, semantically meaningful identifier
- **Steps:**
  - Encode (Sentence-T5) item metadata into dense vectors
  - Apply **residual quantization** (RQ) $\rightarrow$ a sequence of discrete tokens

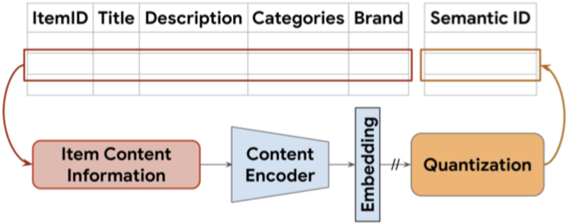


Figure 10: ID generation [Rajput et al., 2023]

Why semantic item IDs matter

- Arbitrary IDs (e.g., item123) are difficult to be learned, since there is a big semantic gap between the model vocabulary and IDs
- Semantic IDs provide structured, informative targets for generative modeling
- Advantages:
  - Generalization: Easier for models to decode and recover unseen or rare items
  - Compositionality: Token overlap reflects semantic similarity across items

Training pipeline

- **Stage 1: Train the VAE-based ID generator**
  - Item metadata $\rightarrow$ encode $\rightarrow$ RQ
- **Stage 2: Generate item IDs**
  - Each item is assigned a multi-token ID from learned codebooks
  - IDs are fixed and used as targets in next stage
- **Stage 3: Train the GenRec model**
  - Autoregressive model trained to predict next item's ID based on user history

Training the VAE-based ID generator

- **Step 1: Encode item metadata**
  - Metadata (title, category, etc.) encoded to a dense vector z via an encoder.
- **Step 2: Initialize residual**
  - Initialize residual $r_1 = z$

Training the VAE-based ID generator

- **Step 3: Iterative quantization**
  - For each level $i = 1$ to m:
    - Select the closest codeword c_i from codebook $\mathcal{V}_i$:

$$c_i = \arg\min_{v \in \mathcal{V}_i} \|r_i - v\|$$

    - Update residual: $r_{i+1} = r_i - c_i$

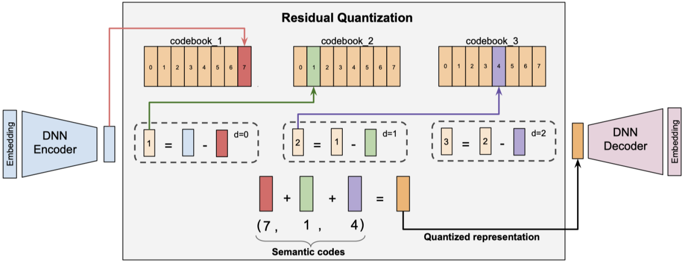


Figure 11: RQ pipeline

Training the VAE-based ID generator

- **Step 4: Form IDs**
  - The item ID is the token sequence: $[c_1, c_2, ..., c_m]$
  - Each c_i is interpretable and belongs to a specific semantic level

Training the VAE-based ID generator

- **Step 4: Form IDs**
  - The item ID is the token sequence: $[c_1, c_2, ..., c_m]$
  - Each c_i is interpretable and belongs to a specific semantic level
- **Loss:**

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}} + \beta \cdot \text{KL}(q(z|x)||p(z))$$

Training the VAE-based ID generator

- **Step 4: Form IDs**
  - The item ID is the token sequence: $[c_1, c_2, ..., c_m]$
  - Each c_i is interpretable and belongs to a specific semantic level
- **Loss:**

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}} + \beta \cdot \text{KL}(q(z|x)||p(z))$$

- Once trained, the encoder + RQ is used to generate IDs for downstream training

Training the GenRec model

- **Input:** A sequence of previously interacted item IDs
- **Target:** The ID tokens of the next item
- **Training strategy:**
  - Teacher forcing
  - Use cross-entropy loss on token prediction (MLE)

Inference

- Beam search or greedy decoding
- Post-processing: Map generated ID tokens back to item via lookup table or similarity match
- Flexible decoding: Support diverse decoding (e.g., sampling, diverse beam)

Results

- Evaluated on Amazon Product Recommendation datasets (Books, Beauty, etc.)
- Compared against:
  - Traditional RecSys (SASRec, GRU4Rec)
  - Retrieval+Generation (Two-stage)
- TIGER outperforms the GenRec baseline P5

Limitations of TIGER

- Token-based decoding still struggles with:
  - **Out-of-catalog items**
  - **No structural encoding** of topic hierarchy or semantic relations
- No explicit modeling of:
  - **Fairness, diversity**, or long-tail bias
  - Personalized decoding strategies
  - **Discriminative training signals**

Q&A

Questions, . . .

Generative retrieval with semantic tree-structured identifiers and contrastive learning

SEATER [Si et al., 2024]

- **S**emantic tr**E**e-based gener**A**tive re**T**ri**E**val with cont**R**astive learning (SEATER)
- Compared to TIGER:
  - TIGER uses flat semantic IDs; SEATER introduces tree structure
  - SEATER unifies generative & contrastive signals

Tree-structured item IDs

- **Idea:** Use tree-structured identifiers to reflect topic granularity
- Balanced K-ary tree structure
- **Benefits:**
  - Identifiers encode semantic hierarchy
  - Better generalization and interpretability

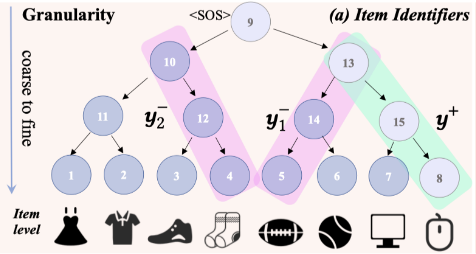


Figure 12: Tree-structured IDs [Si et al., 2024]

Constructing tree-structured IDs

- **Input:** Item embedding retrieved from pretrained SASRec model
- Apply m-level RQ to encode the item embedding into a discrete token sequence
- **Balanced k-ary tree structure:**
  - Each level corresponds to a specific semantic granularity
  - The token path forms a leaf-to-root path in a semantic tree
  - Token space partitioned into subtrees (e.g., genre $\rightarrow$ subgenre $\rightarrow$ item)

Training: Generation loss

- **Backbone**: T5
- **Input:** User history sequence
- **Target:** Structured ID tokens
- **Objective**: MLE loss/ cross-entropy loss

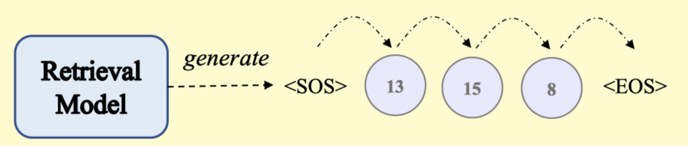


Figure 13: MLE loss [Si et al., 2024]

Training: Alignment loss

- The parent token should align closely with the centroid of its child tokens
- This loss pulls the representations of tokens with parent-child relationships closer and pushes the representations of unrelated tokens apart

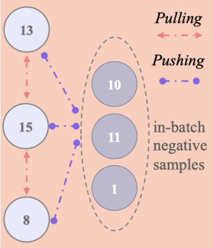


Figure 14: Alignment loss [Si et al., 2024]

Training: Ranking contrastive loss

- **Goal:** Train the model to distinguish similar IDs by learning hierarchical structure
- **The intuition**: longer shared prefixes → more similar items in the hierarchy
- **Approach:**
  - Select sampled item ID with varying prefix lengths shared with the true ID
  - Use triplet contrastive loss to let the decoder learn ranking preferences based on these prefix overlaps

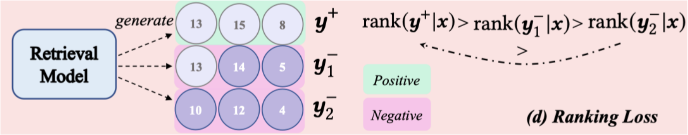


Figure 15: Ranking loss [Si et al., 2024]

Training: Total loss

- Combine three objectives with tuned weights:

$$\mathcal{L} = \mathcal{L}_{\text{gen}} + \lambda_{\text{a}} \cdot \mathcal{L}_{\text{align}} + \lambda_{\text{r}} \cdot \mathcal{L}_{\text{rank}}$$

Results

- Datasets: Yelp, Books, News, Micro-video
- **Findings:**
  - SEATER outperforms TIGER
  - Tree structure + ranking contrastive loss improves both generalization and robustness.

Limitations

- **Limitations:**
  - Tree-based IDs still require careful design — poor trees lead to bad generalization
  - Does not yet support real-time dynamic index updates
  - High training complexity due to hybrid loss terms
- **Future work:**
  - Explore neural tree construction (learnable hierarchies)
  - Integrate reinforcement signals for better decoding feedback
  - Apply to recommendation and multi-modal retrieval

Q&A

Questions,. . .

References i

S. Fraihat, Q. Shambour, M. A. Al-Betar, and S. N. Makhadmeh. Variational autoencoders-based algorithm for multi-criteria recommendation systems. *Algorithms*, 17(12):561, 2024.

S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pages 299–315, 2022.

Q. Li, X. Zhang, J. Xiong, W.-M. Hwu, and D. Chen. Efficient methods for mapping neural machine translator on fpgas. *IEEE Transactions on Parallel and Distributed Systems*, 32(7):1866–1877, 2020.

A. Moreno, H. Castro, and M. Riveill. Client-side hybrid rating prediction for recommendation. In *User Modeling, Adaptation, and Personalization: 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings 22*, pages 369–380. Springer, 2014.

S. Rajput, N. Mehta, A. Singh, R. Hulikal Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Tran, J. Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

Z. Si, Z. Sun, J. Chen, G. Chen, X. Zang, K. Zheng, Y. Song, X. Zhang, J. Xu, and K. Gai. Generative retrieval with semantic tree-structured identifiers and contrastive learning. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 154–163, 2024.

Q. Wang, H. Yin, H. Wang, Q. V. H. Nguyen, Z. Huang, and L. Cui. Enhancing collaborative filtering with generative augmentation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 548–556, 2019.

Y. Wang, J. Xun, M. Hong, J. Zhu, T. Jin, W. Lin, H. Li, L. Li, Y. Xia, Z. Zhao, et al. Eager: Two-stream generative recommender with behavior-semantic collaboration. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3245–3254, 2024.

J. Zhu, M. Jin, Q. Liu, Z. Qiu, Z. Dong, and X. Li. Cost: Contrastive quantization based semantic tokenization for generative recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 969–974, 2024.