JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Machine Learning Algorithms Used for Recommender Systems in Streaming Services

[1]Anant Kumar Dixit, [2]Mr. Kamalendra Verma

[1]Research Scholar, [2]Assistant Professor

[1]Department of Computer Science Engineering,

[1]Patel College of Science & Technology, Indore, India

Abstract : Streaming platforms (video/audio) depend heavily on recommender systems to personalize content, increase engagement, and reduce churn. This paper surveys the machine-learning algorithms commonly used in streaming-service recommender systems, compares classical and modern approaches (collaborative filtering, matrix factorization, neural models, session-based RNNs, Transformer-based sequential models, and graph neural networks), describes evaluation metrics and engineering considerations unique to streaming, and outlines an experimental evaluation plan and recommendation for production deployment. Key strengths, weaknesses, and practical trade-offs for each class of algorithm are discussed.

IndexTerms - recommender systems, streaming services, collaborative filtering, matrix factorization, RNN, Transformer, graph neural networks, personalization,deep neural network.

1 INTRODUCTION

Streaming services (e.g., video and music platforms) rely on recommendation engines to surface relevant content to users and thereby increase watch/listen time and retention. Recommender Systems are the main pillar of the use of large-scale machine learning and data mining in mainstream industry. Diverse applications in areas like e-commerce, search, Internet music and video, gaming, and even online dating apply similar techniques that leverage large volumes of data to better fulfill a user's needs in a personalized fashion.All of these techniques have wide applicability because they have been demonstrated to be effective in increasing the core business metrics such as customer satisfaction and revenue. Industry practitioners combine diversified approaches (candidate generation, ranking, re-ranking, and personalization pipelines) to satisfy latency, freshness, and diversity requirements; the Netflix recommender architecture is a canonical example of such multi-component systems.

This paper focuses on machine learning algorithms used for content recommendation in streaming contexts, emphasizing algorithms that address sequential behavior, session dynamics, and large-scale personalization.

2 BACKGROUND & PROBLEM DEFINITION:

2.1 Recommender system tasks

The main tasks of a recommender system for a streaming service are to analyze user data, build user profiles, and generate personalized content recommendations by using algorithms like collaborative filtering, content-based filtering, and hybrid approaches. These systems aim to enhance user experience by helping them discover content and increase engagement.

- **Rating prediction:** In streaming services, rating prediction is a key component for the broader task of generating personalized item recommendations to enhance user engagement and retention. Rating prediction involves predicting the specific numerical rating or preference a user would give to an item (e.g., a movie, song, or show) they haven't yet experienced. This task is foundational to several recommendation approaches:

a) **Collaborative Filtering**: It is a core mechanism in collaborative filtering systems, which use the predicted ratings to identify users with similar tastes (user-based) or items with similar properties (item-based).

b) **Model-Based Systems**: Matrix factorization and other model-based methods use a user-item matrix to predict missing ratings, which are then used to generate a list of potential recommendations. The Netflix Prize famously used the Root Mean Squared Error (RMSE) of predicted ratings as its primary evaluation metric.

c) **Evaluation**: The accuracy of rating prediction models is often evaluated using metrics like Mean Absolute Error (MAE) and RMSE during offline testing to compare different algorithms.

- **Top-N recommendation Task:**

  The goal of a Top-N recommendation system is to present a small, manageable, and highly relevant list of items (e.g., movies, songs, shows) that a user is most likely to enjoy or watch next. This task is focused on discovery and engagement, aiming to surface relevant content from a vast library and reduce user churn.

  Objective: To identify the N best items for a specific user within a given context.

  Evaluation Metrics: The effectiveness is typically measured using metrics that assess the presence of relevant items within the top N results, such as Precision@K, Recall@K, and F-score.

- **Ranking Task**

  Ranking is a core component within the multi-stage pipeline of a modern recommender system, following an initial candidate generation phase. Its primary function is to optimize the order in which the recommended items are presented to maximize user satisfaction and content consumption.

  Objective: To determine the optimal ordering of a given set of candidate items. It emphasizes the position of an item in the list rather than just its individual score.

  Evaluation Metrics: Performance is evaluated using metrics that account for the position of relevant items, such as NDCG (Normalized Discounted Cumulative Gain), MRR (Mean Reciprocal Rank), and AUC (Area Under the Curve).

  In essence, "Top-N recommendation" describes the user-facing outcome and overall task of content suggestion, while "ranking" refers to the specific machine learning technique used internally to sort the generated suggestions in the most effective order. Streaming services like Netflix use hybrid systems that employ sophisticated ranking algorithms to personalize the user experience.

- **Session-based/sequential recommendation**

  Session-based/sequential recommendation in recommender systems for streaming services focuses on a user's immediate interactions within a single viewing session to predict the next item they'll engage with, rather than relying on their entire long-term history. This approach is crucial for handling "cold-start" problems for new users or content, as it prioritizes the sequence of actions like recent views or plays to suggest relevant content. Models like Recurrent Neural Networks (RNNs) and Transformers are commonly used to capture these temporal patterns.

Focus on the session: The system analyzes a user's current and very recent interactions within a single visit to predict what they might want to watch next. For example, watching several action movies in a row might trigger a recommendation for another action movie.

Sequential dependency: It models the order in which users interact with items to understand the sequence of their interest.

Short-term and immediate context: Unlike traditional methods that use a user's entire viewing history, this approach is more immediate, focusing on the "here and now" of the user's activity.

Handles cold-start: It is particularly useful for new users who haven't built a long-term profile, or for new content that hasn't been widely watched yet.

Why it's important for streaming services

Personalized experience: It provides a more dynamic and responsive user experience by constantly updating recommendations based on current browsing and viewing habits.

Improves discovery: It helps users discover new content they might like, even if they are just browsing or have very limited history.

Handles anonymous users: It works effectively even for users who are not logged in, as it doesn't require a long-term user profile.

Techniques used:

Recurrent Neural Networks (RNNs): These models are designed to handle sequential data and are well-suited for this task.

Transformers: A more advanced deep learning architecture that excels at capturing long-range dependencies in sequential data, often outperforming RNNs.

Markov Chains: A simpler statistical method for modeling sequences, though deep learning methods are more common now.

2.2 Key challenges in streaming services

Scale & latency: In recommender systems for streaming services, scale presents challenges in processing massive datasets for millions of users and items, while latency imposes strict constraints on response times, typically requiring recommendations within milliseconds to maintain user engagement.

- **Scale Challenges and Solutions:**The enormous scale of streaming platforms introduces several issues:

a) **Data Volume:** Handling terabytes of user interaction data and large item catalogs is computationally intensive.

   Solution: Use distributed computing frameworks (like Apache Spark) and scalable data infrastructure. Employ horizontal scaling (adding more machines) and load balancing to manage traffic spikes.

b) **Model Complexity:** Traditional, complex models like matrix factorization can become slow at scale.

Solution: Simplify algorithms or use approximate methods, such as approximate nearest neighbor (ANN) search with libraries like FAISS or Spotify's Annoy, which find similar items in milliseconds.

c) **Feature Engineering:** Extracting meaningful features from vast amounts of user and item data in a timely manner is difficult.

Solution: Use efficient data pipelines and feature stores (e.g., Feast) to manage and cache precomputed features.

- **Latency Challenges and Solutions:**Users expect instant, personalized recommendations, meaning high latency can directly impact engagement and retention.

a) **Real-time Responsiveness:** The system must react instantly to user actions (e.g., starting a show, skipping a song) to provide fresh recommendations.

Solution: Employ caching of frequently accessed data (e.g., user profiles, popular content lists) in in-memory databases like Redis to reduce database load.

b) **Model Inference Speed:** Large models can have slow prediction times.

Solution: Use model optimization techniques like quantization (reducing model size without significant accuracy loss) and leverage hardware acceleration (GPUs or TPUs) for faster parallel processing.

c) **Data Freshness vs. Speed:** Balancing the use of up-to-date user data with the need for immediate responses is a key trade-off.

Solution: Utilize asynchronous processing for non-critical tasks and online learning methods to incrementally update models with new data in real-time, avoiding full retraining.

Cold start & long-tail content

In a streaming service recommender system, the cold start problem is when there isn't enough data to recommend content to new users or new content itself. The long-tail problem is when the system focuses on recommending only the most popular content, neglecting the vast number of less popular (long-tail) items. Both issues can be addressed by using user and item metadata, hybrid approaches, active learning, and by strategically exposing users to less-popular content.

- **Cold start:**This occurs when the system lacks sufficient data for either new users or new items.

a) **User cold start**: A new user joins, and the system doesn't know their preferences.

Solutions: Ask for initial preferences during onboarding, use social network data, or present a diverse set of popular or niche content to gauge initial reactions. Active learning can also be used to ask for feedback on specific items.

b) **Item cold start**: A new movie or show is added to the platform and has no viewing history.

Solutions: Use content-based filtering by recommending the new item to users who have liked similar items based on genre, actors, or other metadata.

- **Long-tail content:**This problem refers to the imbalance where a few popular items dominate recommendations, while many other items with potential audience are ignored. The problem is that Recommender systems often favor popularity, leading to a "long tail" of niche or less popular content that gets overlooked.

- **Solutions**:

1. Strategic exposure: Actively promote long-tail content to users whose preferences might align with it, even if they aren't the most obvious matches.

2. Hybrid models: Combine collaborative filtering (user-based) with content-based filtering (item metadata) to give new or less-popular items a better chance of being recommended.

3. Explore/Exploit strategies: Balance recommending content the user is likely to enjoy (exploit) with suggesting new and different content to broaden their horizons (explore), including long-tail items.

Session dynamics & temporal patterns:

In recommender systems for streaming services, session dynamics and temporal patterns are crucial for capturing the evolution of user preferences and contextual changes in item popularity. They influence system tasks by enabling the distinction between a user's short-term interests and their long-term preferences, ultimately leading to more accurate and timely recommendations. Issues of user preferences change quickly and short sessions matter (e.g., binge session vs. casual drop-in).

Diversity, fairness, and freshness:

Diversity

Diversity refers to the variety of recommended items. In a streaming context, this means ensuring a user's recommendation list is not limited to a single genre or type of content (e.g., only crime dramas).

Fairness

Fairness ensures equitable treatment across different users and content providers. Without it, systems can amplify existing biases, such as consistently recommending popular content from large studios while ignoring quality content from smaller creators.

Freshness

Freshness ensures that newly added or currently trending content is included in recommendations, preventing the system from becoming static and outdated.

In recommender systems for streaming services, diversity, fairness, and freshness are crucial for long-term user satisfaction and a healthy content ecosystem. The primary challenge is balancing these concerns with the traditional goal of accuracy, as they often present trade-offs.

3. Classical Approaches

3.1 Collaborative Filtering (CF) & Nearest-Neighbors

User-based and item-based nearest-neighbor CF compute similarity across users or items and recommend items liked by similar entities. Simple, interpretable, and effective as a baseline but struggle with sparsity and scale.

3.2 Matrix Factorization (MF)

Matrix factorization represents users and items as low-dimensional vectors (latent factors) learned to approximate the user–item interaction matrix (e.g., SVD, probabilistic MF). MF became a standard after success in large recommender competitions and real systems due to compact representations and good scalability in batch training.

Strengths: compact, good for collaborative signals.

Weaknesses: limited modeling of sequential/session dynamics, cold-start without side information.

Content-Based Filtering (CBF)

- **How it works:** Recommends new items based on the attributes of items a user has liked in the past.
- **For streaming:** If a user frequently watches action movies, the system will recommend more action movies, using features like genre, actors, and director to find similar content.
- **Key principle:** Similarity between items.

Collaborative Filtering (CF)

- **How it works:** Makes recommendations by finding users with similar tastes or identifying items that are frequently liked together.
- **Types:**
- **User-based:** Finds users with similar viewing histories and recommends what those "neighbor" users have enjoyed.
- **Item-based:** Recommends items that are similar to those a user has previously watched. For example, if a user watches many items from a specific series, the system will suggest other items from that series or related content.
- **Key principle:** Similarity between users or items.

Hybrid Filtering (HBF)

- **How it works:** Combines collaborative and content-based methods to improve recommendation accuracy and diversity.
- **For streaming:** A common strategy is to start with content-based filtering for new users who have little interaction history, and then gradually integrate collaborative filtering as more data becomes available.
- **Key principle:** Combining multiple approaches to overcome their individual limitations.

4. Deep Learning & Neural Models

Deep learning and neural models are used in streaming service recommender systems to predict user preferences by capturing complex patterns in user behavior and item features through models like Deep Neural Networks (DNNs), Autoencoders, and Transformers. DNNs analyze user demographics and historical interactions, Autoencoders compress user preferences into a compact latent space, and Transformers model the sequential nature of user viewing habits to predict what they'll watch next. These models go beyond traditional methods by handling non-linear data and incorporating a wider range of data sources for more personalized and accurate recommendations.

Deep Neural Networks (DNNs): These models use multiple layers of interconnected neurons to transform raw data into higher-level representations. They are used to model the intricate relationships between users and items by considering a variety of features like demographics, item attributes, and historical interactions to predict the likelihood of a user interacting with an item.

Autoencoders: This type of neural network learns to compress input data (user-item interactions) into a smaller, compressed "latent space" and then reconstructs the original data from this compressed representation. This helps in dimensionality reduction and can capture the core user preferences.

Transformers: Specifically designed for sequential data, transformers are effective in streaming services because they can identify relationships across a user's viewing history (e.g., the sequence of movies watched in a session). This allows the model to predict the next item a user is likely to watch based on their recent activity.

Neural Collaborative Filtering (NCF): This technique extends traditional collaborative filtering using neural networks. It uses embedding layers to create dense vector representations for users and items, which are then fed into a multi-layer perceptron (MLP) to model complex user-item interactions and predict a rating.

5. Sequential & Session-based Models

Sequential and session-based models are crucial for streaming services because they analyze the order of a user's recent interactions to predict immediate interests, unlike traditional methods that focus on long-term preferences. These models, using techniques like RNNs, Transformers, and attention mechanisms, capture the dynamic nature of viewing habits, making them effective for recommending the next video based on what the user is watching right now. This is particularly useful for anonymous users or in real-time scenarios where a user's tastes can shift rapidly.

5.1 RNN / GRU models (GRU4Rec)

Session-based RNNs (notably GRU4Rec) model an entire session as a sequence and predict next-item recommendations from sequence state; they showed large improvements over static baselines in session scenarios and inspired many follow-ups and open-source implementations. However, simpler heuristics or nearest-neighbor session methods sometimes outperform neural models depending on dataset and evaluation setup.

5.2 Transformer / Self-Attention models (SASRec, BERT4Rec)

Transformers adapt self-attention mechanisms to sequential recommendation (SASRec, BERT4Rec, and variants). They handle long sequences efficiently and learn which past interactions are most relevant for predicting the next item; Transformers rapidly became state-of-the-art for many sequential tasks in recommendation. Extensions include personalized transformers and training tasks specialized for ranking.

Strengths: flexible handling of long sequences and complex dependencies; good empirical performance. **Weaknesses:** higher compute & memory cost; require careful negative sampling and training strategies for ranking.

6. Graph Neural Networks (GNNs) for Recommendation

GNNs model user-item interactions as graphs capturing higher-order connectivity (user → item → user). They effectively propagate collaborative signals across the interaction graph and can incorporate social/follow graphs or item similarity graphs. Recent surveys show rapid adoption of GNN-based recommenders in research and highlight challenges like scalability and over-smoothing.

7. Hybrid & Production Architectures

Real streaming services rarely rely on a single algorithm. Typical production pipeline stages:

1. **Candidate generation:** lightweight, high-recall models (e.g., MF, item-based CF, ANN over embeddings, popularity or trending lists)
2. **Ranking:** heavier models (gradient boosted trees, deep neural networks, Transformer encoders) score and rank candidates using features (user/item/context).

Re-ranking / business rules: enforce diversity, freshness, and editorial constraints.

Netflix, for example, uses ensembles of algorithms and business logic to balance accuracy and engagement metrics.

8. Evaluation Metrics & Experimental Design

8.1 Offline metrics

Offline evaluation of recommender systems for streaming services uses metrics like ranking metrics (NDCG, Precision@K, Recall@K) to measure performance on historical data, while experimental design focuses on setting up and validating these offline experiments to predict online performance. Key aspects include choosing appropriate metrics based on the business goal, using hold-out datasets for validation, and incorporating other quality dimensions like diversity, novelty, and bias.

8.2 Online metrics (A/B testing)

In streaming services, A/B testing uses online metrics to directly compare the performance of different recommender system versions in a live environment, using real-time user interactions. These metrics are crucial for establishing a causal link between a change in the system and user behavior, bridging the gap between offline model evaluation and real-world business impact.

Key Online Metrics for A/B Testing

Metrics for streaming services generally fall into engagement, retention, and business value categories:

- **Click-Through Rate (CTR)**: The ratio of clicks on recommended items to the total number of impressions. It is a fundamental measure of the immediate appeal of recommendations.
- **Watch Time / Session Duration**: The total time users spend watching content or interacting with the platform. For services like Spotify ("Time spent listening") or Netflix, this is often a key "North Star Metric" (NSM) that indicates core user satisfaction and engagement.
- **Content Completion Rate**: The percentage of recommended movies, shows, or songs that users finish watching or listening to. This helps to identify "clickbait" recommendations (high CTR, low completion rate).
- **Interaction Rate (Likes, Saves, Additions to Queue)**: Broader metrics that capture various positive user interactions beyond just starting playback.
- **Daily/Weekly Active Users (DAU/WAU)**: Measures the impact of the recommendation system on the regularity of user visits, indicating user retention and habit formation.
- **Churn/Retention Rate**: The rate at which users cancel their subscriptions or stop using the service. An effective recommender system should improve long-term retention by providing a consistently valuable experience.
- **Revenue Per User/Session**: Directly measures the financial impact of the recommendations, such as in-app purchases or subscription upgrades influenced by recommended content.
- **Conversion Rate**: The percentage of users who perform a desired action, which could be signing up for a free trial, subscribing, or upgrading their plan.
- **Novelty and Diversity**: Metrics to ensure the system recommends a wide variety of content and avoids a "popularity bias," which can lead to long-term user fatigue or dissatisfaction, even if short-term engagement metrics are high.

8.3 Experimental considerations

Temporal split for sequential tasks (train on older interactions, test on newer).

Session appearance: maintain realistic session boundaries.

Negative sampling and evaluation protocol critically affect outcomes—careful alignment with production objectives is necessary. Surveys stress reproducible experimental setups for sequential recommenders.

Evaluating recommender systems for streaming services requires a blend of offline metrics (for speed and reproducibility) and online experimentation (A/B testing) (for real-world user behavior), while explicitly considering streaming-specific challenges such as novelty and data sparsity.

9. Representative Comparative Summary (table)

Algorithm family	Typical use in streaming	Strengths	Weaknesses
Item/user CF (kNN)	Candidate generation, baselines	Simple, interpretable	Sparse, scale issues
Matrix factorization	Candidate generation, embeddings	Compact, fast scoring	Static preferences, cold start
Neural CF / DNN	Ranking & re-ranking	Learn complex interactions	Needs lots of data
RNN (GRU4Rec)	Session recommendation	Captures short-term session patterns	Training & hyperparam sensitivity
Transformer (SASRec/BERT4Rec)	Sequential recommendation & ranking	Long-range dependencies, SOTA	Compute/memory heavy
GNN	Social/item graph modeling, cold-start	High-order connectivity capture	Scalability & complexity

10. Proposed Experimental Plan (for an empirical comparison)

10.1 Datasets

MovieLens (various sizes) for benchmarking collaborative tasks

Public streaming-style datasets (e.g., Last.fm, Amazon/YouTube-like logs) and synthetic session data for controlled experiments.
(When deploying in a company, use internal streaming logs for deployment validation.)

10.2 Algorithms to compare

Popularity-Based Recommender System:Popularity-based recommendation is the simplest baseline. It recommends items that are most frequently consumed or highly rated by the overall user base. Items are ranked based on metrics such as total views, average ratings, or number of likes. The top-ranked items are then recommended to all users.

Item-kNN (k-Nearest Neighbors): Item-kNN is a type of collaborative filtering that recommends items similar to those a user has already enjoyed.It identifies the k most similar items to a target item based on user interactions (e.g., co-occurrence in user watch history). If a user has interacted with item A, and item B is highly similar to item A, then item B is recommended.

Matrix Factorization (SVD - Singular Value Decomposition): Matrix factorization, particularly SVD, is a powerful collaborative filtering technique that decomposes the user-item interaction matrix into lower-dimensional matrices. The user-item interaction matrix (e.g., users as rows, items as columns, and ratings/interactions as values) is decomposed into two smaller matrices: a user-feature matrix and an item-feature matrix. These "features" are latent factors that represent underlying characteristics of users and items. The product of these matrices approximates the original interaction matrix, allowing for the prediction of missing values (unseen items).

10.3 Evaluation

Offline Evaluation Metrics:

Offline evaluation utilizes historical data to assess the algorithm's ability to predict user preferences. Key metrics include:

- **Accuracy Metrics:**
  Precision@K and Recall@K: It measure the proportion of relevant items among the top K recommendations and the proportion of relevant items successfully recommended, respectively.

  F1-score@K: The harmonic mean of precision and recall which offering a balanced view of accuracy.

- **Ranking Metrics:**
  Mean Average Precision (MAP): Evaluates the average precision across different recall levels, considering the order of recommendations.

  Normalized Discounted Cumulative Gain (NDCG): Measures the relevance of recommended items, giving higher weight to relevant items appearing higher in the ranked list.

- **Regression Metrics (for rating prediction):**
  Mean Absolute Error (MAE): The average absolute difference between predicted and actual ratings.

  Root Mean Squared Error (RMSE): Measures the average magnitude of the errors, penalizing larger errors more heavily.

- **Beyond Accuracy Metrics:**
  Diversity: Measures the variety of recommended items to prevent over-personalization and exposure to novel content.

  Novelty: Assesses the extent to which recommended items are new or unexpected to the user.

  Coverage: Indicates the proportion of items in the catalog that are ever recommended.

Online Evaluation Metrics:

Online evaluation involves deploying the recommender system in a live environment and measuring its impact on user behavior. This is typically done through A/B testing or other controlled experiments. Key metrics include:

1. **Click-Through Rate (CTR):** The percentage of recommendations that users click on.
2. **Conversion Rate:** It is the percentage of recommendations that lead to a desired action (e.g., watching a movie, adding to a playlist).
3. **Engagement Metrics:**
   a) **Watch time/listening time:** The duration users spend interacting with recommended content.
   b) **Session duration:** The total time a user spends on the platform after receiving recommendations.
   c) **Repeat visits:** The frequency with which users return to the platform.
4. **User Satisfaction:** Measured through surveys or implicit feedback mechanisms (e.g., likes, dislikes).

11. Practical Considerations for Production Deployment

Latencies & model size: Candidate generators should be low-latency and memory-efficient; ranking models may be heavier but must still meet SLAs.

Model refresh & streaming updates: Near-real-time personalization requires incremental updates or fast retraining.

Explainability & editorial control: Business rules and explainable signals are often required for moderation and trust.

Fairness & consumption diversity: Avoid over concentration on already-popular content and ensure long-tail exposure.

Monitoring & feedback loops: Continually monitor online metrics and address feedback loops (e.g., popularity bias). Industry papers discuss the business integration of recommender models and operational trade offs.

12. Discussion & Future Directions

Pretrained sequence models & transfer learning: Adapting large sequence models across domains (e.g., Transformer pretraining) is an active area.

Contrastive learning & self-supervised signals: Using SSL tasks to improve cold-start and representation learning.

Causal & counterfactual evaluation: Move beyond observational metrics to causal inference for better long-term optimization.

Scalable GNNs & graph sampling: Making GNNs practical at streaming scale remains a research and engineering frontier.

13. Conclusion

Streaming services use a mix of classical and modern machine-learning algorithms. Matrix factorization and item-based collaborative filtering remain important for candidate generation; RNNs and Transformers excel at session and sequential modeling; GNNs capture rich relational structure; deep ranking models combine many signals for final ranking. The best production systems combine multiple components—balancing accuracy, latency, freshness, and business constraints. Future improvements will likely come from better sequence pretraining, scalable graph methods, and causal approaches to evaluation.

REFERENCES

1. X. Amatriain and J. Basilico, "Recommender Systems in Industry: Netflix Case Study," *ACM Recommender Systems Conf.*, 2015.
2. P. Resnick and H. Varian, "Recommender Systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.
3. Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
4. S. Zhang et al., "Deep Learning Based Recommender System: A Survey," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 4, pp. 450–467, 2020.
5. H. Wang et al., "Neural Collaborative Filtering," *Proc. WWW Conf.*, pp. 173–182, 2017.
6. W. Kang and J. McAuley, "Self-Attentive Sequential Recommendation," *IEEE ICDM*, pp. 197–206, 2018.
7. R. Ying et al., "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," *ACM KDD*, pp. 974–983, 2018.
8. H. Chen et al., "Top-K Off-Policy Correction for a Reinforcement Learning Recommender System," *ACM WSDM*, 2020.
9. Gomez-Uribe, C. A. & Hunt, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 2015.
10. Koren, Y., Bell, R., & Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Data Science & Recommender Systems* (Netflix tech report). 2009.
11. Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv* / ICLR workshop (2016).
12. Kang, W.-C., & McAuley, J. Self-Attentive Sequential Recommendation (SASRec). *ICDM 2018 / arXiv* (2018).
13. Survey: A Survey on Recommender Systems Using Graph Neural Networks. ACM (2024).
14. Zhang, S., et al. Deep Learning based Recommender System: A Survey. *arXiv* (2017).
15. Ludewig, M., & Jannach, D. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* (2021).