VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Programming Integration Project

Final Report

Session-based Recommendation System

Advisor(s): Assoc.Prof. Thoại Nam
Student(s): Nguyễn Ngọc Song Thương ID 2252803

HO CHI MINH CITY, JUNE 2026

Contents

1 Introduction

The Recommendation System (RS) has become a cornerstone of modern E-commerce services and applications, shaping how users discover and interact with products. From personalized product suggestions to tailored content delivery, RS is critical in enhancing user satisfaction and driving business growth. Traditional recommendation methods typically rely on analyzing the interaction between users and items with all users' historical data. While effective in many domains, these methods face significant challenges in industries with rapidly changing trends, such as fashion or entertainment. In these dynamic industries, user preferences can shift within short timeframes, rendering traditional RS approaches less effective or obsolete. To address these challenges, **Session-Based Recommendation Systems (SR)** have emerged as a promising solution.

The primary objective of this project is to delve into the core concepts and methodologies underpinning SR. Specifically, the project aims to explore state-of-the-art models and techniques that address the sequential nature of session-based data. Recent research highlights the potential of Transformers, a deep learning architecture originally developed for natural language processing, in effectively modeling the sequential dependencies and patterns inherent in session-based recommendations. By harnessing the power of Transformers, SR systems can capture intricate session dynamics and deliver more relevant recommendations.

This report encompasses three key areas:

- Literature Review: We investigate current advancements in SR, focusing on models and techniques using Transformer-based approaches.
- Implementation: The project includes an attempt to reproduce the results of selected papers, with a focus on understanding the implementation details and challenges of state-of-the-art models.

Through this project, we aim to build foundational knowledge in recommendation systems and gain hands-on experience in implementing and evaluating cutting-edge SR models. The insights and outcomes of this work are intended to contribute to a deeper understanding of the field and inspire further exploration into the potential of session-based recommendation systems in dynamic application domains.

2 Literature review

2.1 Recommendation system

2.1.1 Traditional recommendation system

Recommendation systems (RS)[2] are systems that predict items and content that a user might be interested in. The prediction is made based on the interaction between users and items or behavior of similar users. Foundational models in recommendation systems are the early techniques that established the core principles and methodologies of the field:

- **Content-Based Filtering (CBF)**: CBF recommends items similar to those a user has liked in the past. It analyzes the features of items and user profiles to identify similarities. CBF models have evolved from traditional methods like vector-space models and decision trees to more advanced techniques using deep learning.
- **Collaborative Filtering (CF)**: CF recommends items based on the preferences of similar users. It assumes that users who have agreed in the past will agree again in the future. CF can be divided into memory-based and model-based methods. Memory-based CF uses similarities between users or items directly from user ratings. Model-based CF, like matrix factorization, uncovers latent factors representing user preferences and item characteristics.
- **Hybrid Approaches**: Hybrid models combine multiple recommendation techniques, such as CBF and CF, to improve accuracy and address the limitations of individual approaches. These models can leverage the strengths of both content-based and collaborative filtering methods.

Although foundational models are widely used and have achieved significant success, they have several problems[2]:

- Cold-Start Problem: This issue arises when new users or items have limited interaction data, making it difficult for the system to make accurate recommendations.
- Data Sparsity: Recommender systems often deal with sparse data, meaning that there are many missing user-item interactions. This sparsity can affect the accuracy of both the CBF and CF models.

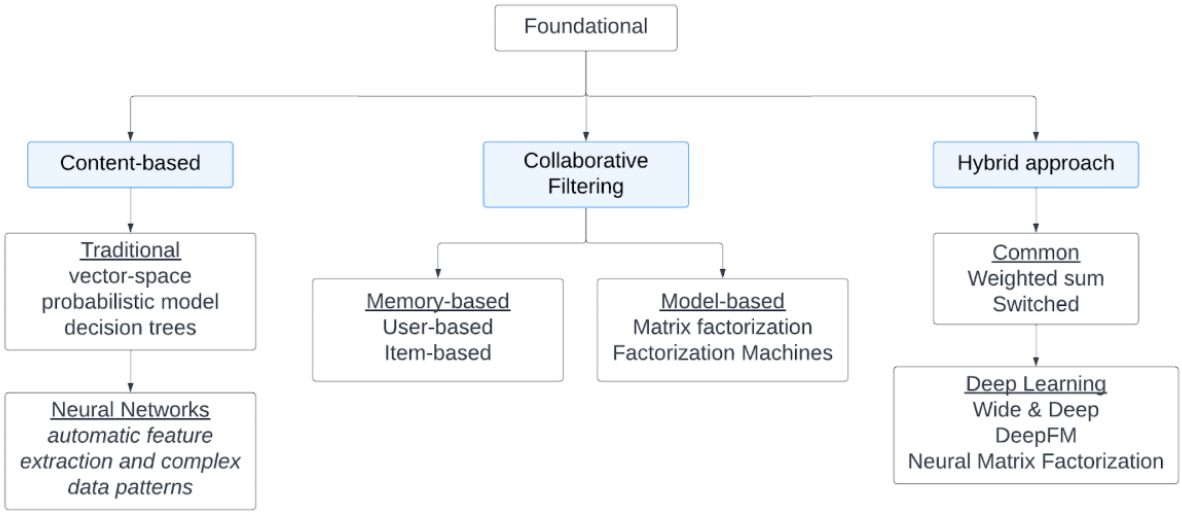

Figure 2.1: Foundational models of RS, compiled from ref[2]

- Scalability: As the number of users and items grows, the computational complexity of some recommendation algorithms, especially memory-based CF, can become a bottleneck.
- Over-Specialization: CBF models tend to recommend items that are very similar to those that a user has already interacted with, limiting the diversity of recommendations.

2.1.2 Sequential and session-based recommendation system

Sequential and session-based recommender systems can address data sparsity and cold start issues by considering both short-term session-based and long-term sequential preferences. They adapt to rapidly changing user interests and provide real-time recommendations, making them effective in industries like entertainment and news. [2]

Sequential and session-based recommender systems both leverage the temporal order of user interactions to make predictions, but they differ in the scope of the user behavior they consider and the time frame in which they operate. [2]

Sequential recommender systems take into account a user's entire interaction history to capture long-term preferences and trends, making them well-suited for predicting future behavior over extended periods. They focus on modeling the evolution of user interests over time. [2]

In contrast, session-based recommender systems focus on a user's activity within a single session, aiming to understand their immediate needs and intent based on recent

interactions. These models excel in providing real-time recommendations, especially for users who are new to the platform, as they do not require extensive historical data to generate relevant suggestions.

The table below compares these two types of models in key aspects.

Table 2.1: Comparison between Sequential and Session-Based Recommendation Models

Aspect	**Sequential Models**	**Session-Based Models**
Scope of User History	Analyze the full interaction history of the user.	Focus on the current session's interactions.
Time Horizon	Predict long-term user behavior.	Provide real-time recommendations during the session.
Focus on User Intent	Capture long-term preferences and trends.	Understand short-term user needs and immediate intent.
Handling New Users	Less effective for new users due to reliance on historical data.	Better suited for new users, as recommendations can be made based on session activity alone.

In essence, sequential models excel in capturing evolving user preferences over time, while session-based models provide highly relevant recommendations for users based on their current behavior, making them ideal for real-time contexts and new user scenarios.

2.2 Session-based recommendation system

2.2.1 Definitions

Session-based recommendation (SR) refers to generating recommendations based on a user's interactions within a session, where sessions are sequences of interactive items $s = [i_1, i_2, \ldots, i_m]$ from a set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$. Given a session s, the task is to predict probabilities (or scores) $\hat{y}$ for all items and recommend the top-K items.[1]

Personalized Session-based Recommendation (PSR) PSR extends SR by incorporating user-specific historical sessions. Let $\mathcal{U}$ denote the set of users, and $\mathcal{S}^u = \{S_j^u\}_{j=1}^{n_u}$ be the set of all historical sessions of user u, where n_u is the total number of sessions for u. Each session $S_j^u = [i_{j,1}, i_{j,2}, \ldots, i_{j,m_j}]$ consists of m_j interactive items.

- Define S_c^u as the current session of user u.

- Define S_h^u as the historical sessions of user u.

The task of PSR is to predict the next interactive item in the current session S_c^u by leveraging historical sessions S_h^u. This is also referred to as streaming session-based recommendation [1].

Session-based Social Recommendation (SSR) SSR incorporates social relationships into PSR. Let $\mathcal{U}$ denote the set of users, and $\mathcal{N}(u) = \{u_k\}_{k=1}^{N(u)}$ represent the neighbors or friends of user u. The sessions of user u_k are denoted as S^{u_k}. The task of SSR is to predict the next interactive item of the current session S^u using both the current session S_c^u and the sessions of neighbors $\{S^{u_k}\}_{k=1}^{N(u)}$ [1].

2.2.2 The overall framework

Several models were developed for session-based recommendation systems, and they have a common framework. The overall framework of session-based recommendation (SR) using sequential neural networks, such as Transformers, involves processing a user's interaction sequence within a session to predict the next item they are likely to interact with. The pipeline typically consists of three main layers [1]:

1. Sequence Modeling Layer: This layer takes the sequence of items within a session as input and applies a sequential neural network to model the order and dependencies between items. Models such as BERT4Rec employ the Transformer architecture, which uses self-attention mechanisms to capture long-range dependencies between items in a sequence, effectively understanding the context of user actions within a session. Earlier models often relied on recurrent neural networks (RNNs), such as GRUs and LSTMs, to process the sequence sequentially and capture temporal dynamics.

2. Session Representation Layer: This layer transforms the output of the sequence modeling layer into a fixed-length vector that represents the user's preferences and intent within the session. Techniques like average or max pooling can be used to aggregate the hidden states from the sequence modeling layer. Attention mechanisms, especially in combination with RNNs or Transformers, allow the model to focus on specific items or parts of the sequence that are most relevant for predicting the next item.

3. Prediction Layer: This layer takes the session representation vector and uses it to predict the probability of the user interacting with each item in the item set. A common approach is to calculate the inner product between the session representation and the embedding vectors of all candidate items. The items with the highest scores are

then recommended to the user. A softmax function is often applied to the output of the prediction layer to obtain a probability distribution over all candidate items.

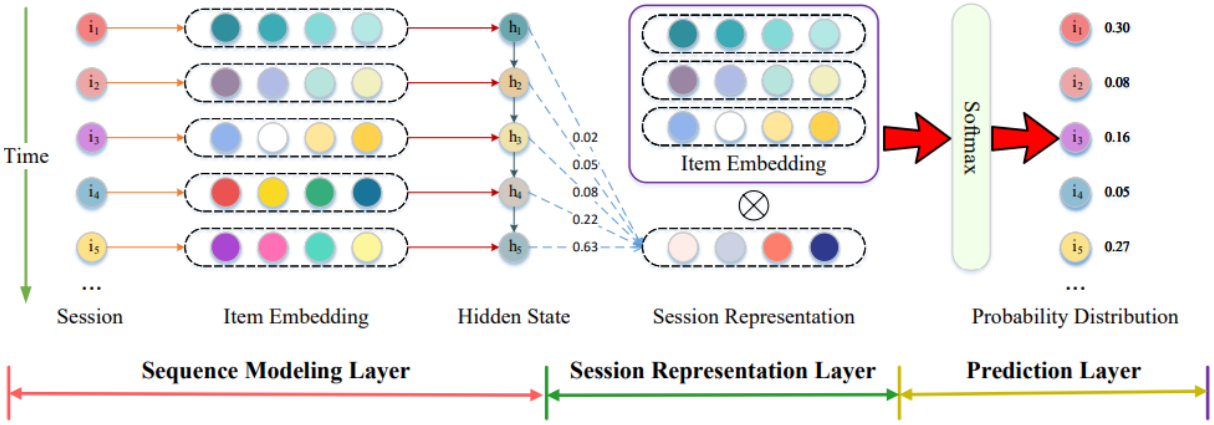

Figure 2.2: The overall framework of sequential neural network for SR [1]

2.2.3 Graph or Sequential Neural Network for SR

While sequential methods have shown some success in SR, their limitations in capturing complex item dependencies and the non-sequential nature of session data have led to the adoption of graph-based methods as a more flexible and effective solution for transition pattern modeling. [1]

- Sequential methods rely heavily on the sequential order of items in a session. This rigid assumption hinders their performance in SR, where item relationships often extend beyond their order of appearance.
- GNNs, on the other hand, excel at capturing these complex, non-sequential relationships by modeling the interactions between items as a graph. This enables them to understand the overall context of the session and provide more accurate recommendations.
- The preference for GNNs over SNNs in SR stems from the observation that session data is often short and lacks a clear sequential dependency between items.

In conclusion, the shift towards graph-based methods in SR signifies the need for models that can effectively capture the intricate and often non-sequential relationships between items to provide more accurate and context-aware recommendations.

2.3 BERT4Rec and Session-aware BERT4Rec

2.3.1 BERT4Rec

BERT4Rec is a deep learning model for sequential recommendation that leverages the bidirectional encoder representations from transformers (BERT), originally developed for natural language processing tasks. It was specifically designed to address the limitations of traditional sequential models like RNNs and unidirectional attention-based models like SASRec[4].

Key features of BERT4Rec:

- Unlike unidirectional models that process sequences from left to right, BERT4Rec uses a **bidirectional architecture** to encode the entire sequence context, enabling it to capture relationships between items regardless of their position in the sequence.
- To avoid information leakage and facilitate bidirectional training, BERT4Rec employs a **Cloze task** where random items in the sequence are masked, and the model is trained to predict these masked items based on their surrounding context. This approach allows BERT4Rec to learn bidirectional representations without directly predicting the next item in the sequence, preventing information from future items from influencing the representation of past items.

Limitations of BERT4Rec:

- BERT4Rec, like other transformer-based models, can be computationally expensive, especially for long sequences.
- The basic BERT4Rec architecture primarily focuses on modeling item IDs, limiting the ability to directly incorporate additional item features or user information.

2.3.2 Session-aware BERT4Rec

While BERT4Rec was initially designed for sequential recommendation, its architecture and training methodology can be adapted to effectively handle session-based recommendation. This idea was tested in BERT-based session-aware recommender systems.

Session-aware recommender systems represent a hybrid approach, combining the strengths of both sequential and session-based models. These systems aim to capture both long-term preferences across all sessions and short-term preferences within a particular session[3]. This approach provides a more comprehensive understanding of user behavior and enables more personalized recommendations.

Its key features include:

- **Input Representation with Session Segmentation**
  - **Session Tokens:** Introduce special tokens to mark the beginning and end of each session within a user's interaction sequence. This helps BERT4Rec distinguish between different sessions and learn session-specific representations.
  - **Session Segment Embeddings:** Similar to segment embeddings used in BERT for differentiating sentences, incorporate session segment embeddings to indicate the position of each session in a user's interaction history. This enables the model to capture relationships between items both within and across sessions.
- **Masked Item Prediction with Session Awareness**: during training, instead of randomly masking items across a user's entire interaction sequence, mask items within each session independently. This encourages the model to learn representations sensitive to the context of each session.
- **Time-Aware Self-Attention** Integrate temporal: information, such as timestamps of interactions, into the self-attention mechanism of BERT4Rec. This allows the model to consider the time elapsed between interactions and capture the dynamics of user interests within a session.

This approach leverages BERT4Rec's strengths in sequence modeling while addressing its limitations in handling session-based data. The use of session tokens, segment embeddings, and time-aware self-attention provides contextual information, while session-based data augmentation enhances the model's ability to generalize to different session scenarios.

3 Implementation

3.1 Objective

The goal of this project was to reproduce the results of BERT4Rec[4] and session-aware BERT4Rec[3] on the MovieLens 1M (ml-1m) dataset, then extend the evaluation to the session-based RetailRocket dataset. However, due to compatibility issues with the original BERT4Rec source code, which relies on TensorFlow 1.12 (GPU version), an alternative implementation provided by the session-aware BERT4Rec repository was used. This implementation ensures consistency with the original BERT4Rec model while allowing integration with newer dependencies.

3.2 Environment Setup

All experiments were conducted using the Kaggle environment with two NVIDIA T4 GPUs. Kaggle's preconfigured environment provided the necessary computational resources and compatibility for running the models effectively.

3.3 Data Preparation

- **MovieLens 1M (ml-1m):** Preprocessed interaction records were provided in the implementation. Users with fewer than five interactions were filtered, and interaction sequences were sorted by timestamp, as recommended in both BERT4Rec [4] and session-aware BERT4Rec [3] papers.
- **RetailRocket:** To enable session-based sequential recommendation on RetailRocket, a custom preprocessing function was implemented to take only neccessary columns.

3.4 Training and Hyperparameter Tuning

- Both BERT4Rec and session-aware BERT4Rec were trained using AdamW optimization with a learning rate of 0.001.
- Specific hyperparameter settings for ml-1m followed published guidelines [4].
- For RetailRocket, adjustments were made to accommodate its higher sparsity and session-based structure. In particular, the configuration for RetailRocket follows the configuration for Steam2, an Amazon dataset with similar sparsity and average session length.

3.5 Evaluation

- **Metrics:** Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Recall were used to evaluate performance, consistent with prior studies.
- **Experimental Setup:** Leave-one-out evaluation was used, with the last item in each sequence held as the test instance.

4 Result and Discussion

In the original BERT4Rec, the authors use a common evaluation strategy: pairing each ground truth item in the test set with 100 randomly sampled negative items that the user has not interacted with, these 100 negative items are sampled according to their popularity[4].

For session-aware BERT4Rec, 3 negative sampling strategies were used to assess the models' accuracy[3]:

- Random Negatives: 100 items not present in the user's interaction history are randomly sampled. While commonly used, this approach can introduce bias.
- Popular Negatives: Items are sorted by popularity and used as negative samples, acting as "hard negatives". This helps assess the model's ability to distinguish highly popular items from truly relevant recommendations.
- All Negatives: The entire item set, including the user's positive items, is used as candidates for ranking.

Model	**#params**	**R@10 Ran.**	**N@10 Ran.**	**R@10 Pop.**	**N@10 Pop.**	**R@10 All**	**N@10 All**
BERT4Rec	4,918,016	0.7500	0.5140	0.4933	0.3224	0.1254	0.0569
BERT4Rec*	4,918,016	0.7341	0.5100	0.5025	0.3211	0.1129	0.0508
SA-BERT	4,920,160	0.7592	0.5280	0.4992	0.3154	0.1388	0.0640
SA-BERT*	4,920,160	0.7458	0.5298	0.5217	0.3283	0.1505	0.0701

Table 4.1: Comparison of Metrics for BERT4Rec and SA-BERT4Rec with (*)Reported results[3] on ml-1m dataset

The original BERT4Rec reported an NDCG@10 score of 0.4818 for the ml-1m dataset. The discrepancy in results may stem from the use of a different framework and variations in hyperparameter choices between the two implementations. Additionally, the reproduction work was conducted in a different computational environment compared to the original study, making subtle differences unavoidable.

Model	#params	R@10 Ran.	N@10 Ran.	R@10 Pop.	N@10 Pop.	R@10 All	N@10 All
BERT4Rec	10331541	0.6346	0.6008	0.6126	0.5707	0.5091	0.3851
SA-BERT	10333685	0.6354	0.5986	0.6119	0.5675	0.4954	0.3632

Table 4.2: Comparison of Metrics for BERT4Rec and SA-BERT4Rec[3] on RetailRocket dataset

There is no noticeable improvement between the session-aware BERT4Rec and the original BERT4Rec, as noted in[3]. This lack of enhancement may be attributed to the short session lengths in the RetailRocket dataset (averaging around 3 items per session), which limits the model's ability to leverage BERT4Rec's strength in capturing long-range dependencies.

References

[1] Zihao Li, Chao Yang, Yakun Chen, Xianzhi Wang, Hongxu Chen, Guandong Xu, Lina Yao, and Michael Sheng. Graph and sequential neural networks in session-based recommendation: A survey. *ACM Computing Surveys (CSUR)*, 57(2):Article 40, 37 pages, February 2025.

[2] S. Raza, M. Rahman, S. Kamawal, A. Toroghi, A. Raval, F. Navah, and A. Kazemeini. A comprehensive review of recommender systems: Transitioning from theory to practice. *arXiv preprint*, arXiv:2407.13699, 2024.

[3] Jinseok Jamie Seol, Youngrok Ko, and Sang goo Lee. Exploiting session information in bert-based session-aware sequential recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*, pages 2639–2644, New York, NY, USA, 2022. Association for Computing Machinery.

[4] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, pages 1441–1450, New York, NY, USA, 2019. Association for Computing Machinery.