DISSERTATION

BiCoRec: Bias-Mitigated Context-Aware Sequential Recommendation Model

Author:
Mufhumudzi MUTHIVHI

Supervisor:
Prof. Terence VAN ZYL
Dr. Hairong BAU

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

Submitted in fulfilment of the requirements for the degree of Master of Science in the School of Computer Science and Applied Mathematics, Faculty of Science, University of the Witwatersrand, Johannesburg

September 22, 2024

Declaration of Authorship

I, Mufhumudzi MUTHIVHI, declare that this dissertation titled, "BiCoRec: Bias-Mitigated Context-Aware Sequential Recommendation Model" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: September 22, 2024

UNIVERSITY OF THE WITSWATERSRAND

Abstract

Faculty of Science

University of the Witwatersrand, Johannesburg

MSc in Computer Science

BiCoRec: Bias-Mitigated Context-Aware Sequential Recommendation Model

by Mufhumudzi MUTHIVHI

Sequential recommendation models aim to learn from users' evolving preferences. However, current state-of-the-art models suffer from an inherent popularity bias. This study developed a novel framework, BiCoRec, that adaptively accommodates users' changing preferences for popular and niche items. Our approach leverages a co-attention mechanism to obtain a popularity-weighted user sequence representation, facilitating more accurate predictions. We then present a new training scheme that learns from future preferences using a consistency loss function. The analysis of the experimental results shows that our approach is 7% more capable of uncovering the most relevant items.

Acknowledgements

I would like to express my very great appreciation to my supervisors Prof. Terence van Zyl and Dr. Hairong Bau, whose generosity knows no bounds. Without my supervisors, completing this dissertation would not have been possible.

Furthermore, I owe an immense debt of gratitude to my incredible mom and dad. Their endless love, unwavering encouragement, and belief in my abilities have driven every achievement. Their wisdom, guidance, and unconditional support have been my greatest blessings.

To Prof. Terence van Zyl, Dr. Hairong Bau, my mom, and my dad—thank you for believing in me, inspiring me, and guiding me every step of the way.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Problem Statement . . . 2
- 1.2 Aims . . . 3
- 1.3 Objectives . . . 3
- 1.4 Significance and Motivation . . . 3
  - 1.4.1 Significance . . . 3
    - Findings . . . 3
    - Contributions . . . 4
  - 1.4.2 Motivation . . . 5
- 1.5 Research Questions . . . 5
- 1.6 Limitations and Delineations . . . 8
- 1.7 Notations . . . 8
- 1.8 Outline . . . 8

2 Recommendation Systems 10
- 2.1 Collaborative Filtering . . . 10
  - 2.1.1 Explicit Feedback Models . . . 10
    - Heuristic Algorithms . . . 11
    - Matrix Factorization . . . 11
  - 2.1.2 Implicit Feedback Models . . . 12
    - Pointwise Learning . . . 13
    - Pairwise Learning . . . 13
- 2.2 Context-Aware Recommandtion . . . 16
  - 2.2.1 Bias . . . 16
  - 2.2.2 Factorization Machines . . . 16
  - 2.2.3 Temporal . . . 17

2.3 Sequential Recommendation . . . 19
2.3.1 Markov Chains . . . 19
2.3.2 Deep Learning . . . 23
Convolutional Neural Networks . . . 23
Recurrent Neural Networks . . . 24
Transformers . . . 25
2.4 Context-Aware Sequential Recommendation . . . 27
2.5 Conclusion . . . 28

3 Popularity Bias **29**
3.1 Collaborative Filtering . . . 30
3.2 Sequential Recommendation . . . 30
3.3 Preference Towards Unpopular Items . . . 33
3.4 Contributions . . . 35

4 Bias-Mitigated Context-Aware Sequential Recommendation **37**
4.1 Bias-Mitigation . . . 37
4.1.1 Term Frequency - Inverse Document Frequency . . . 38
4.1.2 Co-Attention Mechanism . . . 39
4.2 Context-Awareness . . . 40
4.2.1 Multi-Modal Auxiliary Information . . . 40
4.2.2 Semantic Relatedness . . . 43
4.3 Sequential Recommendation . . . 44
4.3.1 Next, Mask and Padded Items . . . 45
4.3.2 Cross-Pseudo Supervision . . . 46
4.4 User Representation . . . 48
4.5 Architecture . . . 49
4.5.1 Embedding Layer . . . 50
4.5.2 Attention Layer . . . 51
Self-Attention . . . 51
Co-Attention . . . 51
User Representation . . . 51
4.5.3 Prediction Layer . . . 51

5 Methodology **53**
5.1 Data . . . 53
5.1.1 Dataset Metrics . . . 53
User-Item Ratio . . . 53
Data Sparsity . . . 54

Sequence Sparsity . . . 54
Skewness . . . 55
Gini coefficient . . . 55
Preference Value Variance . . . 55
5.1.2 Datasets . . . 56
MovieLens 1M Dataset (Movies) . . . 56
Amazon Fashion Dataset (Fashion) . . . 57
Steam Dataset (Game) . . . 57
LastFM Dataset (Music) . . . 57
5.1.3 Preprocessing . . . 58
Truncation and Padding . . . 58
5.2 Baselines . . . 59
5.2.1 Naive Models . . . 59
5.2.2 CF-Based Models . . . 59
5.2.3 Context-Aware Models . . . 60
5.2.4 Sequential Recommendation Models . . . 60
5.3 Results Analysis . . . 61
5.3.1 Evaluation Strategy . . . 61
5.3.2 Accuracy Metrics . . . 62
Recall@N . . . 62
Precision@N . . . 62
MRR . . . 62
NDCG@N . . . 63
5.3.3 Fairness Metrics . . . 63
Diversity@N . . . 63
Novelty@N . . . 64
Serendipty@N . . . 64
5.3.4 Popularity Bias Metrics . . . 64
Popular and Niche Items . . . 65
Popular and Niche User Preferences . . . 65
5.3.5 Change in User Preference . . . 66
5.3.6 Models . . . 66
Naive . . . 66
CF-based . . . 66
Context-aware . . . 66
Sequential recommendation . . . 67
BiCoRec . . . 67
Fairness . . . 67

5.4 Implementation . . . 67
5.4.1 Libraries . . . 68
5.4.2 Hardware . . . 68
5.4.3 Protocol . . . 68
5.4.4 Hyperparameters . . . 69
5.5 Limitations and Assumptions . . . 70
5.5.1 Limitations . . . 70
5.5.2 Assumptions . . . 70
5.6 Ethical Considerations . . . 70
5.7 Conclusion . . . 71

6 Results and Discussion **72**
6.1 Results . . . 72
6.1.1 Overall Performance . . . 72
6.1.2 Popularity Bias . . . 77
6.1.3 Change in Preference . . . 80
6.1.4 Ablation Study . . . 81
Cross-Pseudo Supervision . . . 81
Multi-modal Auxiliary Information . . . 82
Co-Attention Layer . . . 82
User Embeddings . . . 82
6.1.5 BiCoRec vs SOTA . . . 83
Statistical Significance . . . 83
Precision-Recall Curves . . . 84
Fairness Metrics . . . 87
6.2 Discussion . . . 88
6.2.1 Main Results . . . 88
6.2.2 Limitations and Delineations . . . 89
Preference towards Popular Items . . . 89
User Embeddings . . . 89
Balanced Performance . . . 90

7 Conclusion **91**
7.1 Summary . . . 91
7.2 Conclusion . . . 92
7.3 Future Work . . . 93

Bibliography **94**

A Formulations **103**

List of Figures

1.1 Illustrates the process of truncating a long sequence and padding a short sequence with null values. The items are arranged chronologically, from the oldest to the newest interactions made by the user. 7

3.1 Depicts the frequency of each item in a dataset ordered by its popularity. Each dataset follows a power law distribution. A smaller fraction of items appear more frequently than the majority of items. 29

3.2 Presents NDCG@100 score achieved for each item ranked by their histories (largest to smallest). Reflects a direct relationship of a higher NDCG score for items with a large number of histories. The red line depicts a threshold of 0.2 which separates frequently occurring items from less frequent items . . . 32

3.3 Illustrates the power law distributions of an actual recommendation dataset. The threshold of 20% divides the short head from the long tail 33

3.4 Plots the ratio of popular to niche items within each position in the sequence. The ratio decreases as the sequence length increases which shows users shift their preference from popular items to unpopular items overtime 34

4.1 Two-component PCA projection of each user's sequence of TF-IDF popularity scores. From these scores, we can easily identify each user's preference for popular or unpopular items . . 38

4.2 The parallel co-attention mechanism attends to the sequence and popularity representation simultaneously and then produces a popularity-weighted sequence representation. The $\odot$ symbol represents the dot product operation and $\oplus$ refers to the summation operation. 39

4.3 Illustrates a 2-component t-SNE conducted over the text, image, and concatenated text-image item representations. Each representation was derived from a pre-trained model. Notably, the representations demonstrate a clear ability to differentiate themselves based on brand categories, with particularly pronounced distinctions observed in the text-image concatenations. 43
4.4 Depicts the proportion of actual and padded items across all the sequences. The padded items are null entries. More than 25% of the items in a user's sequence are padded items. . . . 44
4.5 Depicts a user's sequence of items with padded items represented in black. Two SASRec models initialized with different weights predict the padded items (in green). The unsupervised loss function ensures consistency of predictions between the two models. 46
4.6 An illustration of the architecture of BiCoRec. The summed positional, auxiliary and item embeddings are passed into four layers to obtain the sequence representation. Then cross-pseudo supervision is used to enforce consistency of predictions between the two networks. 49

5.1 Illustrates the power law distributions of each dataset, accompanied by an exemplified threshold value. The green area would be the popular items, and the red area corresponds to less popular items. 64

6.1 Presents the performance of the most consistent models across each dataset. Our proposed approach achieves the highest NDCG@10 and Recall@10 across all datasets, with the exception of the Fashion dataset. In short-sequence datasets like Fashion, biased models such as BPR tend to perform better due to the prevalence of popular items within the sequences. 73
6.2 Presents the performance over two sets of user preferences. BiCoRec achieves the highest NDCG@10 for less popular items. However, our model performs suboptimally over the Movies and Fashion dataset. In contrast, SASRec demonstrates nearly negligible results for less popular items. 78

6.3 Depicts the performance of SASRec and BiCoRec along the users' sequences through a sliding window of size 50. BiCoRec outperforms SASRec as the window slides across the sequence of items from oldest to recent. 79
6.4 Plots the precision-recall curve of the state of the sequential recommendation models. BiCoRec depicts the steepest slope for long sequence datasets. It only falls short of one model in the Fashion dataset that has very short sequences. 85
6.5 Plots the diversity, novelty and serendipity scores achieved by SOTA models and BiCoRec on three datasets. 86

A.1 Movies Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases. 105
A.2 Fashion Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases. 106
A.3 Games Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases. 106
A.4 Music Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases. 107

List of Tables

3.1 Results from fitting a linear regression curve for each position in the sequence against the ratio 34

5.1 The performance of each dataset against the metrics described in Section 5.1.1 56
5.2 Different types of auxiliary information for each dataset. . . . 58
5.3 Hyperparameters selected after applying exhaustive search using hyperopt 68

6.1 Accuracy Metrics of our approach against 12 baselines. The best-performing models are in bold, and the second-best-performing models are underlined. 72
6.2 Accuracy metrics over the users who prefer popular and less popular items, respectively. The best-performing models are in bold, and the second-best-performing models are underlined. 76
6.3 The performance of BiCoRec after removing each proposed component. 81
6.4 Paired t-test of the NDCG@10 scores achieved by the state-of-the-art sequential recommendation models over BiCoRec's achieved results. 84

A.1 List of libraries used during the implementation of each model 105
A.2 List of parameters and their respective search spaces for each model during hyperparameter optimization 108

List of Abbreviations

BiCoRec	**Bi**as-Mitigated **Co**ntext-Aware Sequential **Rec**ommendation Model
CF	**C**ollaborative **F**iltering
MF	**M**atrix **F**actorization
RNN	**R**ecurrent **N**eural **N**etworks
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
NLP	**N**atural **L**anguage **P**rocessing
BPR	**B**ayesian **P**ersonalized **R**anking
NCF	**N**eural **C**ollaborative **F**iltering
MLP	**M**ulti-**L**ayer **P**erceptron
FM	**F**actorization **M**achines
SVD	**S**ingular **V**alue **D**ecomposition
MC	**M**arkov **C**hains
FPMC	**F**actorized **P**ersonalized **M**arkov **C**hain
PRME	**P**ersonalized **R**anking **M**etric **E**mbedding
HRM	**H**ierarchical **R**epresentation **M**odel
TransRec	**Trans**lation-based **Rec**ommender
Caser	**C**onvolution**A**l **S**equence **E**mbedding **R**ecommendation Model
CNN	**C**onvolutional **N**eural **N**etwork
LFM	**L**atent **F**actor **M**odel
GRU4Rec	**G**ated **R**ecurrent **U**nit **for** Sequential **Rec**ommendation
SASRec	**S**elf-**A**ttentive **S**equential **Rec**ommander
BERT4Rec	**B**idirectional **E**ncoder **R**epresentation **T**ransformer for **Rec**ommandation
KeBERT4Rec	**Ke**yword **BERT4Rec**
SA	**S**elf-**A**ttention
FFN	**F**eed-**F**orward **N**etwork
PFFN	**P**oint-wise **F**eed-**F**orward **N**etwork
MH	**M**ulti-**H**ead Attention
ViT	**Vi**sion **T**ransformer
wave2vec	**Wave to Vec**tor
data2vec	**data to vec**tor
t-SNE	**t**-distributed **S**tochastic **N**eighbor **E**mbedding
ReLU	**Re**ctified **L**inear **U**nit
GELU	**G**aussian **E**rror **L**inear **U**nit
LayerNorm	**Layer Norma**lization
Trm	**Tr**anspor**m**er
NDCG	**N**ormalized **D**iscount **C**umulative **G**ain
MRR	**M**ean **R**eciprocal **R**ank
RandomRec	**Random Rec**ommander
PopRec	**Pop**ular **Rec**ommander

List of Symbols

Symbol	Description
u	user
$\mathcal{U}$	set of users
v	item
$\mathcal{V}$	set of items
$\mathcal{S}_u$	user's sequence of items
n	maximum sequence length
t	time of interaction with an item
a	item auxiliary information
$\mathcal{A}$	set of item auxiliary information
$\mathcal{R}$	set of recommended items made by the model
$\mathbf{q}$	user sequence popularity vector
$\mathbf{Q}$	matrix of user sequence popularity vectors
y	integer or binary value reflecting a users preference value
$\hat{y}$	predicted preference value
$\mathbf{Y}$	interaction matrix
$\mathbf{E}_{\mathcal{V}}$	item embedding
$\mathbf{E}_{\mathcal{U}}$	user embedding
$\mathbf{E}$	input embedding
$\mathbf{H}$	matrix of user sequence embedding
$\mathbf{h}$	vector of user sequence embedding
$\mathbf{W}$	weight embedding
$\mathbf{P}$	positional embedding
$\mathbf{b}$	bias term
d	number of features
$\mathcal{T}$	training set
$\mathcal{H}_v$	set of histories that precedes the item
$p()$	probability function
$f()$	recommendation model
$\phi()$	mapping function
$\sigma()$	sigmoid function
$\tanh()$	hyperbolic tangent function
$\text{softmax}()$	softmax function
$\mathcal{P}()$	the power set
$\mathbb{1}$	indicator function
$\log()$	log function
$\exp()$	exponential function
$\ell_{ce}()$	cross-entropy loss function

argmax()	returns the position of the largest value
Θ	model parameters
λ	regularization term
γ	magnitude
α	threshold value
τ	temperature parameter
$\mathbb{R}$	the set of real numbers
$\mathbb{Z}$	the set of integers

Chapter 1

Introduction

Recommendation models help alleviate the information overload problem by suggesting a relevant item to users based on their preferences [1]. User preference is measured either explicitly or implicitly. The user provides explicit feedback, whereas implicit feedback is obtained by observing the user's actions. Hence, implicit feedback is a more reliable measurement of user preference.

Early implicit feedback-based models made use of a Collaborative Filtering (CF) approach. CF predicts the item that a user would prefer by observing the interests of similar users [2]. The most successful CF-based models make use of a representation learning framework. The objective is to find the latent spaces that encode the user-item interaction matrix. Initially, Matrix Factorization was applied to project the user and item features into the same latent factor space. Later, a neural network was used to learn a non-linear function of user-item interactions.

However, CF-based models ignore the contextual information surrounding a user's preference for a particular item. Users may exhibit biases, and items may possess additional side information that enhances their utility [3], [4]. Moreover, user preferences evolve over time [3]. Context-aware models aim to incorporate the contextual information of bias, auxiliary information and temporal dynamics.

The current state-of-the-art recommendation models consider CF a univariate time series prediction problem. They represent the interacted items in sequential order such that the objective is to predict the next item the user would prefer. Early sequential recommendation models made use of first and high-order Markov Chains. However, Markov-Chains infer predictions from a single or few past items, capturing short-range item transitions for

recommendation [5]. Recurrent Neural Networks (RNN) can capture long-range dependencies by utilizing a hidden state to encode the previous items. Subsequently, RNNs require a large amount of data before outperforming baselines [6]. Transformer-based recommendation models alleviate this issue by using an attention mechanism to adaptively assign weights to previous items at each time step [7].

The transformer-based models only embed item identifiers and positional markers. As a result, they learn the chronological order of items to represent a user's sequence of preferences. Context-aware sequential recommendation models aim to incorporate any contextual information associated with an item [8]–[10]. However, current methods are limited to the fusion of item-attribute data.

Recommendation models encounter a challenging trait within datasets. The frequency of items follows a power law distribution [11]. Often, the most popular items sit at the short-head of the distribution [12]. Popularity bias produces a severe class imbalance between two groups: those at the short head of the distribution and those found at the long tail of the distribution [13].

Collaborative Filtering models are particularly susceptible to bias [12]. Furthermore, popularity bias in context-aware sequential recommendation models is an under-explored area in research.

1.1 Problem Statement

The problem addressed in this study is formulated as follows. Given a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$, a set of items by $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$, the auxiliary information data $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{V}|}\}$, the user sequence popularity scores $\mathbf{Q} \in \mathbb{R}^{|\mathcal{U}| \times n}$ and the sequential item data $\mathcal{S}_u = \{v_1^u, v_2^u, \ldots, v_n^u\}$ where $v_t^u \in \mathcal{V}$ represents an interaction of user u with item v at relative time t and n is the maximum sequence length, predict the next item v_{n+1}^u that is most preferred by user u:

$$p(v_{n+1}^u | \mathcal{S}_u, \mathcal{A}, \mathbf{q}_u) \tag{1.1}$$

which is the probability over all possible items for user u at time step $n+1$ and $\mathbf{q}_u \in \mathbf{Q}$ [7], [8], [13], [14].

1.2 Aims

This study aims to develop a model that can mitigate the inherent popularity bias within sequential recommendation models. The proposed model should adaptively accommodate the evolving preferences of users, balancing recommendations between popular and niche items.

1.3 Objectives

The above aims of this research project were achieved through the following objectives:

- design a co-attention mechanism for reasoning between user sequence attention and popularity attention;
- create a cross-pseudo supervision network to adapt to evolving user preferences;
- implement various baseline models for comparative analysis and
- establish an evaluation scheme to assess the prevalence of popularity bias in user sequences.

1.4 Significance and Motivation

1.4.1 Significance

Findings

We found that BiCoRec is deeply personalized and responsive to changing preferences. It is, on average, 7% more capable of uncovering the most relevant items. Our bias mitigation technique improved the performance of users who prefer less popular items by 26%. Finally, BiCoRec achieved an average of 3.14% increase in the NDCG score for long user sequence datasets over current state-of-the-art models.

Based on our ablation study, the cross-pseudo supervision network contributed the most to the performance of our overall strategy. Hence, learning from evolving future preferences is essential. However, BiCoRec struggles to infer accurate predictions for users who strongly favour popular items or have short sequences.

Contributions

To summarize, the main contributions of this work include the following.

1. Incorporated multi-modal auxiliary information to improve the latent description of each item [8]. Our context-aware model is not limited to item-attribute data.
2. Devised a novel technique to measure the popularity bias in each user's sequence of items. We used the Term Frequency - Inverse Document Frequency (TF-IDF) score to describe the item's popularity in relation to other users. Consequently, a vector of TF-IDF scores describes the user's changing preferences towards items of different levels of popularity.
3. Used a co-attention mechanism to capture the dynamic interplay between a user's sequence of preferences and their predisposition towards items of varying popularity.
4. Used a cross-pseudo supervision network to learn from evolving future preferences [13].
5. Incorporated an explicit user embedding to learn better-personalized information about each user.
6. Developed a novel evaluation scheme that measures users' evolving preferences towards popular or niche items.

This dissertation, so far, resulted in the publication of two peer-reviewed conference papers. They are:

1. **Multi-Modal Recommendation System with Auxiliary Information, SACAIR Conference 2022, Springer Publication** [8] - Was our first successful attempt to incorporate multi-modal auxiliary information into a transformer-based sequential recommendation model.
2. **Impacts of Architectural Enhancements on Sequential Recommendation Models, SACAIR Conference 2023, Springer Publication** [13] - Was our first discovery of the inherent popularity bias within sequential recommendation models. We proposed our novel evaluation methodology that accurately reflects users' preferences for items of varying popularity. Finally, we introduced our cross-pseudo supervision network here, which uses pseudo-labelling.

Furthermore, we plan to submit a journal paper on bias mitigation in context-aware sequential recommendation models. Context-aware recommendation

models have recently been adopted and demonstrate state-of-the-art performance. However, popularity bias, despite being a well-known issue in the field of recommendation systems, has not been thoroughly studied.

1.4.2 Motivation

The emergence of multi-modal transformers in both Natural Language Processing (NLP) and computer vision has significantly influenced our work within the domain of recommendation models. These multi-modal transformers assist in broadening the contextual information associated with items. Moreover, the recommendation landscape has transitioned from a traditional user-item interaction matrix to sequence models that capture users' changing preferences. Hence, we developed a context-aware sequential recommendation model.

Furthermore, several investigations have focused on addressing popularity bias in recommendation systems [12], [15]–[18]. Nonetheless, only a few studies have incorporated bias mitigation techniques within the context of sequential recommendations [13], [19]. However, Yang *et al.* [19] does not apply bias mitigation on context-aware sequential recommendation models. Hence, we facilitated the development of a bias-mitigated context-aware sequential recommendation model.

1.5 Research Questions

Below are four research questions we investigated:

1) Why do we need a bias-mitigation strategy for a sequential recommendation model?

Deep learning-based classifiers are known to emphasize their training on the majority set within the training data [12]. The current state-of-the-art sequential recommendation models are transformer-based recommanders [7], [14]. Sequential recommendation models often recommend frequently occurring items [13].

Moreover, our investigation revealed that users increasingly favour less popular items over time. This trend suggests that user preferences evolve toward more niche and personalized items.

This study proposed a few bias mitigation techniques that aid in alleviating the inherent popularity bias in sequential recommendation models. The proposed components of our final model provide three solutions.

1. Recommend items based on their semantic relatedness instead of their popularity.
2. Learn the popularity bias in each user's sequence of preferences.
3. Learn from evolving future preferences.

2) How do we modify recommendation algorithms to include more items of lesser popularity?

Niche items receive less attention during training. Hence, sequential recommendation models struggle to recommend them. Our proposed approach employs two strategies to refocus the recommendations towards less occurring items.

1. We used multi-modal auxiliary information to learn the semantic relatedness between items. We recommend items based on the latent description instead of their popularity.
2. We used a co-attention mechanism to jointly attend to the user sequence and popularity representation. We use the popularity-weighted sequence representation to make predictions.

3) How do we learn from user's evolving future preferences?

The primary learning method in sequential recommendation is to predict the next item or masked item tokens given the previous items [7], [14]. Hence, they ignore the future items. Users tend to prefer less popular items over time. Therefore, future items will contain valuable insights about the users' evolving preferences towards more niche items.

We devised a novel method to learn from future items. Consider the user's sequence of items arranged from oldest to newest. Let n be a fixed sequence length that represents each user's list of interactions. If a user has interacted with more items than n, we truncate their sequence of items. Otherwise, if a user has interacted with fewer items than n, we pad the sequence with zeros. This results in a sequence where the newest items occupy the initial positions of the sequence, and the padded items fill the preceding positions, refer to Figure 1.1.

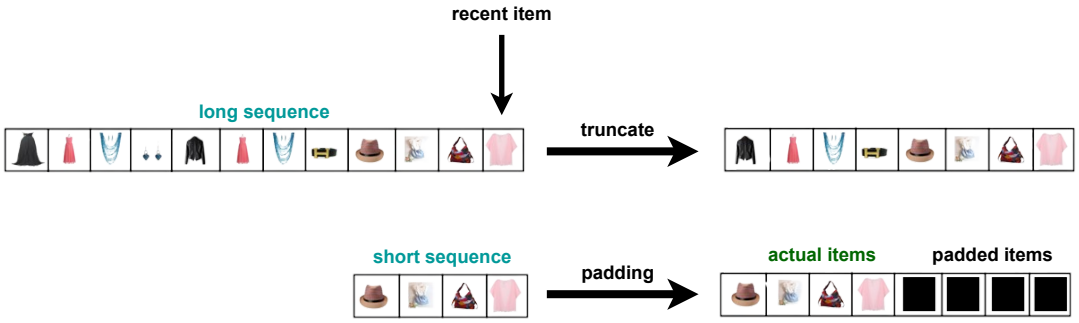


FIGURE 1.1: Illustrates the process of truncating a long sequence and padding a short sequence with null values. The items are arranged chronologically, from the oldest to the newest interactions made by the user.

Since a sequence is arranged from oldest to newest items, the padded items represent the future items. Learning from padded items provides information about the users' evolving future preferences.

We may view the actual items as labelled data and the padded items as unlabeled data. The existence of labelled and unlabeled data transforms the problem into a semi-supervised learning paradigm. We used Cross-pseudo Supervision to ensure the consistency of predictions from both labelled and unlabeled items [13].

4) How do we better evaluate a sequential recommendation model to measure the shift in user preferences?

Applying a conventional accuracy metric to all items within a sequence fails to capture the nuanced shift in preference. In short sequences, popularity bias is more prominent, as these sequences often comprise predominantly popular items. Consequently, a model heavily biased toward popular items may perform well in such scenarios. However, its performance diminishes when deploying the same model in an offline setting with considerably longer user sequences.

We propose a new method to evaluate user sequences. Instead of looking at the entire sequence individually, we break it down into smaller parts using a sliding window approach. For instance, say we set the window size to 50. We then take the first 50 items from the sequence and use a predictive model to guess the next item. After that, we slide the window by one item and repeat the prediction process. We keep doing this until we cover the entire sequence. This way, we can evaluate the predictive model's performance at different user sequence segments.

1.6 Limitations and Delineations

Below are a few challenges that limit the performance of BiCoRec.

1. BiCoRec may not perform optimally in situations where users strongly favour popular items.
2. BiCoRec produces sub-optimal performance for users with short sequences or highly skewed datasets.
3. Our framework may not beat the current state-of-the-art approaches in scenarios involving session-based interactions or situations characterized by short-term dynamics.
4. Our bias-mitigation strategy heavily penalizes popular items in favour of niche items.
5. BiCoRec did not exhibit an impressive performance improvement when incorporating user embeddings.
6. We were not able to fully balance BiCoRec's performance across users who prefer popular items and those who prefer less popular items. Our strategy still performs substantially better for users who prefer popular items [13].

1.7 Notations

To facilitate the reading in this dissertation, we define a few main concepts: matrices are denoted by bold capital letters, e.g. $\mathbf{A}$, vectors appear in bold lowercase letters, e.g. $\mathbf{a}$, scalars are written in italics, e.g. a and sets are represented with italic capital letters, e.g. $\mathcal{A}$.

1.8 Outline

This chapter looked at the background concerning the primary research problem and highlighted its significance in the relevant field. We also outlined our contributions and motivations for this study. The rest of the dissertation is structured as follows: Chapter 2 provides an overview of the background in Recommendation Systems. In Chapter 3, we show that there is an inherent popularity bias within Sequential Recommendation models. In Chapter 4, we present the model framework and some discussions on related work. Chapter 5 discusses the methodology, while Chapter 6 presents our results

compared to state-of-the-art baselines. Finally, Chapter 7 concludes the dissertation by summarising its contributions and presenting recommendations for future work.

Chapter 2

Recommendation Systems

The objective of a recommendation model f is to reduce the sparse nature of recommendation datasets by filling in the interaction matrix $\mathbf{Y}$ with an integer or binary value y_{uv} that we refer to as user u's preference score of item v, such that:

$$\hat{y}_{uv} = f(u, v|\Theta) \tag{2.1}$$

where $\hat{y}_{uv}$ is the predicted value, Θ are the model parameters, $u \in \mathcal{U}$ the set of users and $v \in \mathcal{V}$ the set of items. Finding preference scores for each user-item pair is known as the *matrix completion problem* [20]. This chapter discusses some traditional and current methods used to solve the matrix completion problem.

2.1 Collaborative Filtering

Some of the earliest methods used a Collaborative Filtering (CF) approach to solve the matrix completion problem [2]. CF assumes that users who share similar interests will consume similar items. By clustering users based on their interests, we can recommend items from the same group of users. Early CF methods predict a positive integer value that reflects how much a user prefers an item.

2.1.1 Explicit Feedback Models

Explicit feedback records the user u's preference of item v with a rating $y \in \mathbb{Z}_0^+$ [21]. The user numerically expresses their level of preference, for example, a star rating between 1 ("totally dislike") and 5 ("really like"). Explicit feedback is a well-defined problem with a simple objective.

Heuristic Algorithms

Heuristic algorithms are used to calculate the similarity values between users. Specifically, the nearest-neighbour algorithm is applied to cluster users based on their preferences [22]. The objective is to find the preference score of each item v_i made by a target user u by considering the preference scores of the k most similar items $\{v_1, v_2, ..., v_k\}$. Given a similarity measure (for example, Cosine Similarity), let $s_{v_i v_j}$ be the item-to-item similarity between items v_i and v_j such that:

$$\text{Cosine Similarity} : s_{v_i v_j} = \frac{v_i \cdot v_j}{\|v_i\|_2 \left\|v_j\right\|_2} \tag{2.2}$$

where $i \in \{0, 1, ..., |\mathcal{V}|\}$ and $j \in \{0, 1, ..., k\}$. The predicted preference score of item v_i made by user u is computed as

$$\hat{y}_{uv_i} = \frac{\sum_{j=0}^{k} s_{v_i v_j} y_{uv_j}}{\sum_{j=0}^{k} s_{v_i v_j}} \tag{2.3}$$

where y_{uv_j} is the ground truth preference score made by user u to item v_j.

Matrix Factorization

To solve the matrix completion problem, model-based Collaborative Filtering adopts a representation learning framework. The objective is to find the latent spaces that encode the user-item interaction matrix $\mathbf{Y}$. The first successful realization of model-based CF used Matrix Factorization (MF) [3]. Matrix Factorization projects user and item features into the same latent factor space. MF is tasked with finding the item latent factors $\mathbf{E}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and user latent factors $\mathbf{E}_{\mathcal{U}} \in \mathbb{R}^{|\mathcal{U}| \times d}$, where d denotes the number of features that describe item v and user u. Here, $\mathbf{e}_v \in \mathbf{E}_{\mathcal{V}}$ measures the extent to which the item possesses those factors. The user features $\mathbf{e}_u \in \mathbf{E}_{\mathcal{U}}$ measures the extent of interest the user has in items characterized by those factors. The dot product of $\mathbf{e}_v$ and $\mathbf{e}_u$ captures the user's overall interest in the item's characteristics and provides the estimated preference value

$$\hat{y}_{uv} = \mathbf{e}_v^T \mathbf{e}_u \tag{2.4}$$

We can learn the feature vectors $\mathbf{e}_v$ and $\mathbf{e}_u$ by minimizing the loss between the real and predicted values of y_{uv}. Take a loss function such as the regularized squared error on the set of known values

$$\min_{\mathbf{e}_v^* \mathbf{e}_u^*} \sum_{(u,v)\in\mathcal{T}} (y_{uv} - \mathbf{e}_v^T \mathbf{e}_u)^2 + \lambda(||\mathbf{e}_v||^2 + ||\mathbf{e}_u||^2) \tag{2.5}$$

where $\mathcal{T} = \{(u,v)|y_{uv} \in \mathbb{Z}_0^+\}$ is the set of user-item pairs of the ground truth preference score, that is the training set, and λ is a regularization term. A *stochastic gradient descent approach* popularized by Simon Funk[1] loops over the training set to produce the prediction error $e_{uv} := y_{uv} - \mathbf{e}_v^T \mathbf{e}_u$. The parameters are modified by a magnitude proportional to γ in the opposite direction of the gradient:

$$\mathbf{e}_v \leftarrow \mathbf{e}_v + \gamma(e_{uv} \cdot \mathbf{e}_v - \lambda \cdot \mathbf{e}_v) \tag{2.6a}$$

$$\mathbf{e}_u \leftarrow \mathbf{e}_u + \gamma(e_{uv} \cdot \mathbf{e}_u - \lambda \cdot \mathbf{e}_u) \tag{2.6b}$$

2.1.2 Implicit Feedback Models

The ratings observed from explicit feedback are provided by the user. Hence, they may be biased and inconsistent [23]. Implicit feedback records user preference as a binary value $y_{uv} \in \{0,1\}$, where a value of one indicates that the user has interacted with the item, and zero otherwise. Implicit feedback describes the frequency of actions, which is the *confidence* that we have in a certain observation [21]. Hence, implicit feedback is a more reliable metric. In addition, most feedback in the real world is implicit, for instance, monitoring clicks, view times, purchases, mouse movements, etc. By observing the user's behaviour, we can record their actions. We can simplify the value of an implicit feedback variable to account for actions taken or not taken. However, $y_{uv} = 1$ does not mean user u likes the item v. Moreover, $y_{uv} = 0$ does not mean user u does not like item v. Alternatively, a user may not be aware of the item [21]. The unobserved items can either be negative feedback (the user is not interested in the item) or missing values (the user might be interested in the item). This shows that implicit feedback provides noisy signals about user preference [24].

[1] http://sifter.org/~simon/journal/20061211.html

The missing value problem is handled in two ways. Firstly, all the missing values are ignored, and only the observed feedback is considered. Ignoring the missing values would force the models to learn only from the observed instances. Secondly, all the missing values are treated as negative feedback [21], [25], thus discouraging the model from recommending the unobserved items. Furthermore, a naive optimization of the loss function, such as in Equation 2.5, becomes computationally expensive as the user-item matrix is densely filled with observed and unobserved feedback. Hence, a new learning and optimization method has been adopted for the implicit feedback setting. Two types of objective functions are often used to estimate the parameters Θ: pointwise and pairwise strategies.

Pointwise Learning

Pointwise learning follows a regression framework by minimizing the squared loss between the predicted value $\hat{y}_{uv}$ and the target value y_{uv}. Either the unobserved entries are treated as negative feedback, or the negative instances are sampled from unobserved entries [21], [23], [25]. Then weights w_{uv} are introduced to each user-item pair to differentiate between observed and unobserved feedback. The new objective function for minimizing the loss function is described as

$$\min_{\mathbf{e}_v^* \mathbf{e}_u^*} \sum_{(u,v)\in\mathcal{T}} w_{uv}(y_{uv} - \mathbf{e}_v^T \mathbf{e}_u)^2 + \lambda(||\mathbf{e}_v||^2 + ||\mathbf{e}_u||^2) \tag{2.7}$$

where w_{uv} is the confidence weight of y_{uv} such that its value is significantly lower for missing feedback and higher for observed feedback. These weights help reduce the impact of unobserved examples. The Alternating Least Squares (ALS) method optimizes and decomposes the objective function into user-independent and item-independent parts by differentiation. ALS is particularly effective for sparse datasets, as it handles missing data by focusing on observed user-item interactions. However, it is slower than Stochastic Gradient Descent (SGD).

Pairwise Learning

Pairwise learning takes a ranking-based approach where the observed entries are ranked higher than the unobserved ones. Hence, pairwise learning minimizes the loss between the predicted value $\hat{y}_{uv}$ and the target value y_{uv} while

maximizing the margin between the observed entry $\hat{y}_{uv}$ and unobserved entry $\hat{y}_{uv'}$, where v' is the item that user u has not interacted with. **Bayesian Personalized Ranking** achieves this by correctly ranking item pairs instead of scoring single items [23]. Bayesian Personalized Ranking avoids replacing missing values with negative ones. Finding the correct personalized ranking for all items requires optimizing the maximum posterior probability, which is formulated as

$$p(\Theta|\mathcal{T}) \propto p(\mathcal{T}|\Theta)p(\Theta) \tag{2.8}$$

where $p(\Theta|\mathcal{T})$ is the posterior probability of the model parameters Θ given the training data $\mathcal{T}$. The likelihood of the data $p(\mathcal{T}|\Theta)$ given the parameters Θ represents the probability of observing the user-item interactions under the current model. The prior probability $p(\Theta)$ of the parameters Θ encodes any prior beliefs about the distribution of these parameters. Bayesian Personalized Ranking aims to find the parameter set Θ that maximizes this posterior probability. The optimization process adjusts the model parameters to maximize the likelihood of the observed user-item pairs while regularizing the prior to avoid overfitting, such that

$$\max_{\Theta} \sum_{(u,v,v')\in\mathcal{T}} \ln \sigma(\hat{y}_{uv} - \hat{y}_{uv'}) - \lambda\|\Theta\|^2 \tag{2.9}$$

where $\mathcal{T}$ is now the set of user-item triplets (u, v, v') and user u is observed to interact with item v but not with item v'. The sigmoid function σ models the probability of a user preferring item v over item v'. The term $\hat{y}_{uv} - \hat{y}_{uv'}$ represents the difference in predictions for the user's preference for items v and v', and $\lambda\|\Theta\|^2$ is a regularization term that prevents overfitting by penalizing the complexity of the model parameters Θ. Bayesian Personalized Ranking assumes that the users prefer observed (interacted) items over unobserved ones. However, as previously noted, this may not be true.

The parameters Θ have been learnt through Matrix Factorization (MF). MF estimates a preference value y_{uv} through the inner product of the user and item latent factors; see Equation 2.4. The inner product combines the latent user and item features linearly. The linear computation may not be sufficient to model the complex structure of users' interactions. Neural networks can approximate any continuous function effectively [26]. He *et al.* [27] propose

Neural Collaborative Filtering (NCF) which uses a multi-layer neural network f in collaborative filtering to estimate $\hat{y}_{uv}$ over the function

$$f(u, v|\mathbf{E}_{\mathcal{U}}, \mathbf{E}_{\mathcal{V}}, \Theta_f) = \phi_{out}(\phi_l(...\phi_2(\phi_1(\mathbf{E}_{\mathcal{U}}^T\mathbf{v}_u, \mathbf{E}_{\mathcal{V}}^T\mathbf{v}_v))...)) \tag{2.10}$$

where Θ_f are the model parameters of the interaction matrix, ϕ_{out} and ϕ_l are the mapping functions for the output layer and l-th neural collaborative filtering layer. The feature vectors $\mathbf{v}_u$ and $\mathbf{v}_v$ are derived from the input layer. These input features are the item and user identification numbers transformed into a binarized sparse vector with one-hot encoding. The user and item feature matrices $\mathbf{E}_{\mathcal{U}}$ and $\mathbf{E}_{\mathcal{V}}$ are derived from an embedding layer consisting of a fully connected network. The fully connected layer projects the sparse representation to a dense vector. We obtain the user latent factor $\mathbf{e}_u$ from $\mathbf{E}_{\mathcal{U}}^T\mathbf{v}_u$ and the item latent factors $\mathbf{e}_v$ from $\mathbf{E}_{\mathcal{V}}^T\mathbf{v}_v$. The first layer performs an element-wise product of vectors that mimics MF:

$$\phi^{MF}(\mathbf{e}_u, \mathbf{e}_v) = \mathbf{e}_u \odot \mathbf{e}_v. \tag{2.11}$$

In parallel, a Multi-Layer Perceptron (MLP) model with l-hidden layers learns a non-linear relationship of user-item interactions:

$$\begin{aligned} \mathbf{z}_1 &= \phi_1(\mathbf{e}_u, \mathbf{e}_v) = \begin{bmatrix} \mathbf{e}_u \\ \mathbf{e}_v \end{bmatrix} \\ \phi_2(\mathbf{z}_1) &= \mathbf{ReLU}_2(\mathbf{W}_2^T\mathbf{z}_1 + \mathbf{b}_2), \\ &\dots \\ \phi_L^{MLP}(\mathbf{z}_{L-1}) &= \mathbf{ReLU}_L(\mathbf{W}_L^T\mathbf{z}_{L-1} + \mathbf{b}_L), \end{aligned} \tag{2.12}$$

where $\mathbf{W}_l$ and $\mathbf{b}_l$ denote the weight matrix and bias vector for the l-th layer's perceptron, respectively [28]. The ReLU activation function is well-suited for sparse data because it encourages sparse activations. In addition, ReLU is more biologically plausible and has proven to be non-saturated [29]. The final NCF model employs separate learned embeddings from the MF and MLP layers. Then, their last hidden layers are concatenated together to produce the predicted value

$$\hat{y}_{uv} = \sigma(\mathbf{v}_{out}^T \begin{bmatrix} \phi^{MF} \\ \phi^{MLP} \end{bmatrix}) \tag{2.13}$$

where $\mathbf{v}_{out}^T$ are the weights of the output layer. NCF combines a linear (MF) and non-linear model (MLP) to capture the complex relationships between users and items.

2.2 Context-Aware Recommandtion

CF-based models learn user preference by encoding the collaborative signals within the interaction data. However, they often overlook the contextual information surrounding a user's preference for a particular item. That is, users interact with items under some context [30]. Context is any information that characterizes a situation related to the interaction between humans, applications and the surrounding environment [31]. Therefore, context has an impact on the interaction a user has with the set of items under some environmental conditions. The following subsections present different forms of context and solutions proposed to design a context-aware model.

2.2.1 Bias

The inner product in Equation 2.4 of the CF-based matrix factorization model does not address the fact that much of the variation in preference values is due to the effects of bias inherent within users and items [3]. Since a small number of users and items contribute towards a larger number of interactions, the average preference value of an item is underpinned by a small subset of users. In essence, the average preference value becomes skewed by the context of popularity. Koren, Bell, and Volinsky [3] propose **De-biased Matrix Factorization** that encapsulates the effects of bias, which do not involve user-item interaction, within a baseline predictor

$$b_{uv} = \mu + b_u + b_v \tag{2.14}$$

for an unknown preference value y_{uv}, such that b_u and b_v indicate the observed deviations of user u and item v to the average preference value μ. Hence, the de-biased version of Equation 2.4 is now expressed as:

$$\hat{y}_{uv} = \mu + b_u + b_v + \mathbf{e}_v^T \mathbf{e}_u \tag{2.15}$$

2.2.2 Factorization Machines

Factorization Machines (FMs) can estimate reliable parameters under very high sparsity [4]. Instead of the original user-tem interaction matrix, they use a description matrix. The description matrix is produced by concatenating five-row vectors. The first two rows are one-hot encoded vectors of size $|\mathcal{U}|$ and $|\mathcal{V}|$ that indicate the active user and items, respectively. The third row

indicates the previously seen items with a one-hot encoded vector. Then, the fourth row represents the auxiliary information transformed into a one-hot encoded vector. *Auxiliary information* is additional side information that provides more insights about an item. FM uses item attributes from tabular data, such as name, colour or size. Finally, the fifth row is a one-hot encoded vector consisting of indicators for the last item the user has rated before the active item.

FM then estimates interactions by breaking the independence of the interaction parameters through factorization. The independence assumption means that the prediction of one feature does not depend on the values of other features. However, features interact in complex and nonlinear ways in many real-world scenarios, and their joint effect influences the prediction. By breaking the independence assumption, FMs allow for capturing the interactions by introducing latent factors. The latent factors are expected to capture the interactions between different features. As a result, the model can learn complex patterns and dependencies in the data that the independence assumption would miss.

FM uses the user-item interaction matrix with additional contextual features such as auxiliary information. The description matrix is an enhanced representation of the user-item interaction matrix. However, it is still highly sparse.

MF-based models struggle with high-dimensional sparse data because they cannot effectively model feature interactions. FM model interactions between features by using factorized parameters. Instead of directly learning a weight for each feature, they learn a low-dimensional latent vector.

A latent feature vector is allocated to each column (attribute) to estimate the target variable. The estimated target variable is produced by summing all pairwise interactions within the latent feature space among the attribute columns. The product of their assigned values further influences the computation in the description matrix.

2.2.3 Temporal

Temporal models learn the temporal dynamics of user preference over time. The user and item factors of MF in Equation 2.15 assume that a user's preference remains static. User preference and item popularity change over time [3]. A user's taste may evolve, or a new item may have mainstream appeal.

Hence, the model should incorporate the context of time to account for temporal effects that reflect the dynamic, time-drifting nature of user-item interactions. The updated de-biased MF model accounts for temporal dynamics of the bias terms and user factors as a function of time t to produce the estimate

$$\hat{y}_{uv|t} = \mu + b_{u|t} + b_{v|t} + \mathbf{e}_v^T \mathbf{e}_{u|t} \tag{2.16}$$

where the item factors $\mathbf{e}_v$ remain static since items are static in nature. The user-item interaction matrix $\mathbf{Y}$ is denoted as a quadri-tuple $[u, v, y_{uv}, t]$.

Koren [32] argues that temporal effects may span long or short periods. They suggest that while the likability of movies may not vary daily, it is more likely to change gradually over extended periods. In contrast, they note that user preferences can exhibit daily fluctuations, reflecting the natural inconsistencies in customer behaviour. As a result, we require a finer time resolution when modelling user biases and a lower resolution for capturing item-related time effects. Koren [33] expands the **Singular Value Decomposition** (SVD++) model by incorporating the context of time to the predicted value

$$\hat{y}_{uv|t} = \mu + b_{u|t} + b_{v|t} + \mathbf{e}_v^T \left(\mathbf{e}_{u|t} + |\mathcal{S}^u|^{-\frac{1}{2}} \sum_{v \in \mathcal{S}^u} y_v \right) \tag{2.17}$$

where $\mathcal{S}^u$ is the set of items consumed by user u, and y_v is a second set of item factors that characterize users based on the set of items that they consumed. Equation 2.17 is referred to as **TimeSVD++**, it represents users through the items that they prefer with time drifting parameters $b_{u|t}$, $b_{v|t}$ and $\mathbf{e}_{u|t}$.

Wu *et al.* [34] propose a neural network applied to a temporal collaborative filtering model. The researchers propose using a **Recurrent Neural Network** (RNN) to capture user and item temporal dependencies. The RNN allows us to model past observations while predicting future trajectories in one seamless manner, such that:

$$\hat{y}_{uv|t} = f(\mathbf{e}_{u|t}, \mathbf{e}_{v|t}) \quad \text{and} \quad \begin{aligned} \mathbf{e}_{u|t+1} &= \phi_1(\mathbf{e}_{u|t}, y_{uv|t}), \\ \mathbf{e}_{v|t+1} &= \phi_2(\mathbf{e}_{v|t}, y_{uv|t}) \end{aligned} \tag{2.18}$$

where ϕ_1 and ϕ_2 are functions that are learnt to find the parameters. RNN learns a function that sequentially updates scores and can make forward predictions at each time step t. A Long Short-Term Memory (LSTM) is used as ϕ_1 and ϕ_2 since it can address the vanishing gradients problem. For instance,

take the item's latent state

$$\begin{aligned}
[g_1, g_2, g_3] &= \sigma(\mathbf{W_1}[\mathbf{e}_{v|t-1}, y_{uv|t}] + \mathbf{b_1}), \\
o_{1t} &= \tanh(W_2[\mathbf{e}_{v|t-1}, y_{uv|t}] + \mathbf{b_2}), \\
o_{2t} &= g_1 \cdot o_{2,t-1} + g_2 \cdot o_{1t} \text{ and} \\
\mathbf{e}_{v|t} &= g_3 \cdot \tanh(o_{2t})
\end{aligned} \tag{2.19}$$

where g_1, g_2 and g_3 are forget, input and output gates, respectievly. The same process in Equation 2.19 also applies for $\mathbf{e}_{u|t}$. The objective function now includes a temporal element described as

$$\min_{\Theta} \sum_{(u,v,t)\in\mathcal{T}} \|y_{uv|t} - \hat{y}_{uv|t}\|_2^2 + \lambda\|\Theta\|^2. \tag{2.20}$$

2.3 Sequential Recommendation

A temporal model's parameters Θ reflect the evolution of users, items and preferences. However, TimeSVD++ and the RNN model do not attempt to estimate future preferences; they only interpolate between past interactions. These models focus on extracting insights from historical data, assuming that past user preferences indicate future choices. While temporal dynamics effectively identifies patterns and trends in existing data, it may not always accurately predict how users' preferences evolve given new and unseen items.

Zimdars, Chickering, and Meek [35] define collaborative filtering as a univariate time series prediction problem. They represent the interacted items $\mathcal{S}^u$ in sequential order such that the model becomes a sequential prediction problem. Given a user's past actions, we can estimate their future actions by modelling transitions in the sequences of items. They model the context of users' actions based on their recent activities [35].

2.3.1 Markov Chains

Shani *et al.* [36] further refine the sequential prediction problem as a sequential decision problem where a recommender makes a decision at each time step. The sequential decision recommender uses a first-order Markov Chain to predict the next item that a user would prefer given a set of already consumed items $\mathcal{S}^u_{t-1}$.

Rendle, Freudenthaler, and Schmidt-Thieme [5] extend the idea by using personalized Markov Chains described as

$$p(\mathcal{S}_t^u | \mathcal{S}_{t-1}^u) \tag{2.21}$$

where $\mathcal{S}_t^u \subseteq \mathcal{V}$ so that $\mathcal{S}^u = \{\mathcal{S}_1^u, ..., \mathcal{S}_{t-1}^u\}$ is the set of interacted items made by user u. The transition matrix $\mathbf{B}^u$ is defined on the state space over sets $\mathcal{P}(I)$ to produce

$$b_{ij}^u := p(\mathbb{1}_{\mathcal{S}_t^u(v_i)} | \mathbb{1}_{\mathcal{S}_{t-1}^u(v_j)}) \tag{2.22}$$

where $\mathbb{1}$ is an indicator function representing whether an item has been consumed. Sequential recommendation aims to find the probability of recommending an item to a user given the previous set of consumed items. Specifically, sequential recommendation is the mean over all the transition probabilities from the set of the last consumed items up until the current item, defined as

$$p(\mathbb{1}_{\mathcal{S}_t^u(v_i)} | \mathcal{S}_{t-1}^u) := \frac{1}{|\mathcal{S}_{t-1}^u|} \sum_{\mathbb{1}_{\mathcal{S}_{t-1}^u(v_j)}} p(\mathbb{1}_{\mathcal{S}_t^u(v_i)} | \mathbb{1}_{\mathcal{S}_{t-1}^u(v_j)}) \tag{2.23}$$

By transforming the data from an interaction matrix to a transition matrix $\mathbf{B}^u$ for each user u, we can encode changes in a user's preferences over time. That is, optimal recommendations depend not only on previous items consumed but also on the order in which those items are consumed. Rendle, Freudenthaler, and Schmidt-Thieme [5] propose the use of Matrix Factorization over the transition cube $\mathbf{B}$ such that the estimate

$$\hat{\mathbf{B}} := \mathbf{K} \times \mathbf{E}_{\mathcal{U}} \times \mathbf{E}_{\mathcal{V}}^{\text{out}} \times \mathbf{E}_{\mathcal{V}}^{\text{in}} \tag{2.24}$$

is obtained through Tucker Decomposition such that $\mathbf{K}$ is the core tensor, $\mathbf{E}_{\mathcal{U}} \in \mathbf{R}^{|\mathcal{U}| \times d}$ and $\mathbf{E}_{\mathcal{V}}^{\text{out}} \in \mathbf{R}^{|\mathcal{V}| \times d}$, $\mathbf{E}_{\mathcal{V}}^{\text{in}} \in \mathbf{R}^{|\mathcal{V}| \times d}$ are the latent factor matrices of the users, items of the last transition (outgoing nodes) and the items to predict (ingoing nodes), respectively. Then estimate the transition probabilities $\hat{b}_{ij}^u$ for user u consuming item v_i given item v_j by setting

$$\hat{b}_{ij}^u = \mathbf{e}_u^T \mathbf{e}_{v_i} + \mathbf{e}_{v_i}^T \mathbf{e}_{v_j} + \mathbf{e}_u^T \mathbf{e}_{v_j}. \tag{2.25}$$

The result is a low-rank approximation $\hat{\mathbf{B}}$ such that

$$p(\mathbb{1}_{\mathcal{S}_t^u(v_i)}|\mathcal{S}_{t-1}^u) = \frac{1}{|\mathcal{S}_{t-1}^u|} \sum_{\mathbb{1}_{\mathcal{S}_{t-1}^u(v_j)}} \hat{b}_{ij}^u \tag{2.26}$$

is known as a **Factorized Personalized Markov Chain** (FPMC). Matrix Factorization uses all the data to learn the general taste of the user, whereas Markov Chain captures sequential effects over time using a transition matrix. Applying the (general) transition matrix to the user's last actions produces a personalized set of items. A sequential Bayesian Personalized Ranking is used to obtain the predicted value

$$\hat{y}_{uv} = p(v \in \mathcal{S}_t^u|\mathcal{S}_{t-1}^u) \tag{2.27}$$

$$= \mathbf{e}_u^T\mathbf{e}_v + \frac{1}{|\mathcal{S}_{t-1}^u|} \sum_{v' \in \mathcal{S}_{t-1}^u} \mathbf{e}_v^T\mathbf{e}_{v'} \tag{2.28}$$

where t is the time at which user u prefers item v. Equation 2.28 combines MF with Markov Chains. That is, the model parameters are learnt jointly for both (1) the inner product user and item factors and (2) the inner product of the factors of the previous and next items. The recommended items $\hat{y}_{uv}$ are relevant to past preferences and contextually appropriate based on the user's recent interactions.

Various improvements are made to FPMC to address some of its limitations. The inner product of the item and user factors in Equation 2.25 violates the triangle inequality. The MF can model similarity between user and item representations yet fails to capture the similarity between user-to-user or item-to-item representations. If a user consumes items v and v', then the representations of v and v' should be close within the latent space. Feng *et al.* [37] propose the use of a **Personalized Ranking Metric Embedding** (PRME) model, which enhances the embedding process by incorporating a metric learning approach. PRME models the user-item interaction by learning a personalized distance metric for each user. The key idea is to minimize the distance between the user's embedding and the embedding of the items they prefer while maximizing the distance from the items they do not prefer.

User preference (MF) and sequential dynamics (MC) are inherently correlated. The two components should not be modelled separately through a linear combination. Each component should not influence the user's next

purchase independently. For instance, Wang *et al.* [38] mentions that individuals who purchase pumpkins will also likely buy other vegetables, such as cucumbers or tomatoes. Subsequently, those who purchase candy may opt for additional snacks like chocolate or chips. However, those purchasing pumpkins and candy together are more inclined to buy Halloween costumes next. This example underscores the complexity of recommending items solely based on individual purchases. Independently recommending pumpkins and candy might not yield accurate suggestions. Wang *et al.* [38] propose a **Hierarchical Representation Model** (HRM) that aggregates the representation of the user u and the previous item v_{t-1}, before their compatibility with the next item v_t is measured. HRM employs a two-layer structure to construct a hybrid representation over the user factors and item factors of the previous interaction, computed as

$$z_{uv|t-1}^{Hybrid} = \phi_2(\mathbf{e}_u, \phi_1(\mathbf{e}_{v_{t-1}} | v_{t-1} \in \mathcal{S}_{t-1}^u)) \tag{2.29}$$

where ϕ_1 is the first layer that forms the sequential representation by aggregating item factors $e_{v_{t-1}}$ from the previous interactions, while the second layer ϕ_2 builds the hybrid representation by aggregating the user factors/representation e_u and the sequential representation [38]. The sequential representation models the sequential behaviour, while the user representation captures the general taste in recommendation. They modify the FPMC Equation 2.28 with a softmax function resulting in the predicted preference value

$$\hat{y}_{uv|t} = \frac{\mathbf{e}_{v_t} \cdot z_{uv|t-1}^{Hybrid}}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{e}_{v_t} \cdot z_{uv|t-1}^{Hybrid})}. \tag{2.30}$$

HRM still models user preference and sequential dynamics under two separate latent components. He, Kang, and McAuley [39] propose using a single component that captures the correlation between preferences and sequential continuity. A **Translation-based Recommender** (TransRec) models a user based on the ordered set of items they have consumed. That is, TransRec finds the latent translation vector $\mathbf{h} \in \mathrm{R}^d$ from the user's sequence of items $\mathcal{S}^u \in \mathrm{R}^n$. Where n is the maximum sequence length such that $v_t \in \mathcal{S}^u$ and $t \in \{1, 2, \ldots, n\}$. $\mathcal{S}^u$ captures the user's inherent intent or "long-term preferences" that influenced her to make these decisions. This approach naturally captures personalized sequential behaviour. If user u transitioned from item

v_{t-1} to item v_t, then

$$\mathbf{e}_{v_{t-1}} + \mathbf{h} \approx \mathbf{e}_{v_t} \tag{2.31}$$

such that $\mathbf{e}_{v_t}$ is the nearest neighbour of $\mathbf{e}_{v_{t-1}} + \mathbf{h}$ under some distance metric. Hence, we modify the FPMC Equation 2.28 to

$$\hat{y}_{uv|t} = b_{v_t} - \|(\mathbf{e}_{v_{t-1}} + \mathbf{h}) - \mathbf{e}_{v_t}\|_2^2 \tag{2.32}$$

where $\hat{y}_{uv|t}$ is the predicted transition score for user u from item v_{t-1} to item v_t and b_{v_t} is a single bias term that captures the overall item popularity.

2.3.2 Deep Learning

So far, the models used for sequential recommendation use Matrix Factorization and Markov Chains, which capture linear dependencies. The appeal of MF is that it can incorporate contextual information such as bias, auxiliary information and temporal dynamics. However, MF is still defined by its inner product in Equation 2.4. The inner product linearly combines the multiplication of latent features. A linear model may not capture the complex patterns and relationships within recommendation datasets.

Deep learning is adept at learning non-linear continuous functions. This section investigates a few deep-learning-based sequential recommendation models for learning user preferences and sequential dynamics. Similar to applying neural networks to the temporal recommendation case, we now observe the sequential recommendation problem with neural nets.

Convolutional Neural Networks

Markov chains learn point-level sequential patterns such that each previous item influences the target item. Tang and Wang [40] argues that sequences reflect a union-level influence where several previous items, in that order, jointly influence the target item. For instance, "buying milk and butter together increases the probability of buying flour". In addition, sequences observe skip behaviour. That is, the impact from past behaviours may skip a few steps and impact future items. **Convolutional Sequence Embedding Recommendation Model** (Caser) represents the previous n items as an $n \times d$ matrix $\mathbf{E}$, where d is the number of latent dimensions and the rows preserve the order of the items [40]. Using various convolutional filters, Caser learns an embedding matrix as an image of the n items in the latent space by searching for sequential patterns as local features. The horizontal and vertical

convolutional filters capture sequential patterns at point-level, union-level, and skip behaviours. Caser incorporates the Convolutional Neural Network (CNN) to learn sequential features and the Latent Factor Model (LFM) to learn user-specific features. The embedding retrieves the previous n item features and stacks them together into a matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$ for user u at time t, such that:

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{v_1} \\ \vdots \\ \mathbf{e}_{v_2} \\ \mathbf{e}_{v_n} \end{bmatrix} \tag{2.33}$$

$\mathbf{E}$ can also be considered an "image" of the previous n items in the latent space. We extract features from the embeddings using horizontal and vertical filters. The horizontal filters are of size $h \times d$ where d is the full width of the latent features. Horizontal Convolutional Layer slides over the rows of items and picks up sequential patterns, capturing union-level patterns with multiple union sizes. Then max pooling is applied on the final convolution so that the most significant features are selected from the filter. The Vertical Convolutional Layer slides across $\mathbf{E}$ from left to right along the columns. The vertical filter captures point-level sequential patterns through weighted sums over previous items' latent representations. Max pooling is not applied to these features since we want to keep the aggregation for every latent dimension. Both the Horizontal $\mathbf{c}^{\text{Hor}}$ and Vertical $\mathbf{c}^{\text{Ver}}$ convolution layer outputs are concatenated and fed to a fully connected layer to produce

$$\begin{aligned} \mathbf{z} &= act\left(\mathbf{W_1}\begin{bmatrix} \phi_{\max}(\mathbf{c}^{\text{Hor}}) \\ \phi_{\text{WeightedSum}}(\mathbf{c}^{\text{Ver}}) \end{bmatrix} + \mathbf{b}_1\right) \text{ and} \\ \hat{y}_{uv_t} &= \mathbf{W}_2\begin{bmatrix} \mathbf{z} \\ \mathbf{E}_{\mathcal{U}} \end{bmatrix} + \mathbf{b}_2 \end{aligned} \tag{2.34}$$

where $\mathbf{E}_{\mathcal{U}}$ is the user embedding that captures long-term general user preferences, and $\mathbf{z}$ captures short-term sequential patterns.

Recurrent Neural Networks

Jannach and Ludewig [6] propose the use of **Gated Recurrent Unit for Sequential Recommendation** (GRU4Rec). The network's input is the sequence of items, while the output is the next item. They adopt the same strategy as Equation 2.18. However, instead of using item and user features $\mathbf{e}_v$ and $\mathbf{e}_u$,

a 1-of-N encoding is used. The input vector is a one-hot encoding of all the items with the vector element corresponding to the active item set as one.

A feed-forward layer is added between the last layer and the output. The final output is the predicted preference of the items. Then, they use a learning-to-rank module to rank the relevant item higher in the list. Similar to BPR, a pairwise ranking approach is adopted. Pairwise ranking compares the score or the rank of pairs of a positive and a negative item, and the loss enforces the rank of the positive item to be lower than that of the negative one.

However, RNNs require a large amount of data before outperforming baselines. Conversely, first-order (FPMC, HRM or TransRec) and high-order (Caser) Markov Chains benefit from inferring future preference from single or multiple past actions. The attention mechanism aims to learn from all past actions (RNNs) whilst inferring predictions from a smaller subset of actions (MCs). Attention Mechanisms achieve this by adaptively assigning weights to previous items at each time step [7]. The model places importance on the subset of items relevant to the next observed item.

Transformers

Kang and McAuley [7] proposed using a transformer for sequential recommendation. The transformer is a "self-attention" module that successfully captures complex semantic patterns from sentences in Natural Language Processing (NLP). The **Self-Attentive Sequential Recommander** (SASRec) uses an item embedding $\mathbf{E}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and positional embedding $\mathbf{P} \in \mathbb{R}^{n \times d}$ that encodes the positions of previously observed items since the self-attention model does not include any recurrent or convolutional modules [7]. Finally, n is the maximum sequence length. If the sequence length exceeds n, we truncate the sequence to the most recent n actions. If the sequence length is less than n, we pad the positions at the front with zeros until the length is n. The input embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d}$ is produced by combining the item and position embeddings:

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{v_1} + \mathbf{p}_1 \\ \mathbf{e}_{v_2} + \mathbf{p}_2 \\ \cdots \\ \mathbf{e}_{v_n} + \mathbf{p}_n \end{bmatrix} \tag{2.35}$$

where $\mathbf{p}_t \in \mathbf{P}$ is a vector embedding at position t and item $v_t \in \mathcal{S}^u$. SASRec then uses an attention layer that calculates the weighted sum of all values, computed as [41]

$$\text{Attention}(\mathbf{W}_{\text{query}}, \mathbf{W}_{\text{key}}, \mathbf{W}_{\text{value}}) = \text{softmax}\left(\frac{\mathbf{W}_{\text{query}}\mathbf{W}_{\text{key}}^T}{\sqrt{d}}\right)\mathbf{W}_{\text{value}} \quad (2.36)$$

where $\sqrt{d}$ is the scaling factor to avoid large inner product values for high dimensionalities and $\mathbf{W}_{\text{query}}, \mathbf{W}_{\text{key}}, \mathbf{W}_{\text{value}}$ are projection matrices. SASRec converts $\mathbf{E}$ into three matrices through linear projections and feeds them into an attention layer

$$\text{SA}(\mathbf{E}^l) = \text{Attention}(\mathbf{E}^l\mathbf{W}_{\text{query}}, \mathbf{E}^l\mathbf{W}_{\text{key}}, \mathbf{E}^l\mathbf{W}_{\text{value}}); \text{ with} \quad (2.37)$$

$$\mathbf{E}^{l+1} = \text{SA}(\mathbf{E}^l) \quad (2.38)$$

where $\mathbf{E}^l$ is the embedding matix at layer l. To endow the model with non-linearity and consider interactions between different dimensions, SASRec adopts a Point-wise Feed-Forward Network

$$\text{PFFN}(\mathbf{E}^{l+1}) = [\text{FFN}(\mathbf{e}_1^{l+1})^T, \ldots, \text{FFN}(\mathbf{e}_n^{l+1})^T]^T; \text{ and} \quad (2.39)$$

$$\text{FFN}(\mathbf{e}_t^{l+1}) = \text{ReLU}(\mathbf{e}_t^{l+1}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \quad (2.40)$$

where $\mathbf{e}_t^l \in \mathbf{E}^l$ is the hidden representation of layer l at position t and $\mathbf{W} \in \mathrm{R}^{d\times d}$ and $\mathbf{b} \in \mathrm{R}^d$ are the weights and bias vectors respectively. In some cases, the Self-Attention (SA) layer and feed-forward network (FFN) are stacked against each other to learn more complex item transitions.

Deeper neural networks may result in overfitting and an unstable training process due to vanishing gradients or longer training times due to more parameters. Kang and McAuley [7] propose using residual connections, layer normalization and dropout to alleviate the above issues. Residual connections propagate low-layer features to higher layers. Since embedding the previously visited item is entangled with all previous items after several self-attention blocks, adding residual connections helps propagate the last visited item's embedding to the final layer. Layer normalization helps stabilize the neural network by normalizing the inputs across features towards zero-mean and unit variance [42]. Dropout alleviates overfitting by "turning off" neurons at a probability of p during training [43]. Finally, SASRec adopts an MF layer to predict the next preferred item v_{n+1} from the predicted preference

value

$$\hat{y}_{uv_{n+1}} = \max(\mathbf{h}^l_{n_{n+1}} \mathbf{E}_{\mathcal{V}}{}^T) \tag{2.41}$$

where $y_{v_{n+1}}$ is the preference score of the item v_{n+1}; hence, we select the item that generated the highest preference score.

SASRec encodes the user's historical interactions unidirectionally from left to right. That is, each item can only encode the information from previous items [14]. Sun *et al.* [14] argue that a unidirectional transformer is insufficient in learning the representation of user's behaviours. They argue that it is crucial to incorporate context from left to right and right to left directions. Sun *et al.* [14] adopt a Bidirectional Encoder Representation Transformer (BERT), used in NLP, to iteratively revise the representation of every position by exchanging information across all positions. Unlike the single-head attention mechanism used in SASRec, **Bidirectional Encoder Representation Transformer for Sequential Recommandation** (BERT4Rec) uses a Multi-Head Attention (MH) mechanism. Multi-head attention jointly attends to information from different representation subspaces at different positions. Specifically, multi-head attention first linearly projects $\mathbf{E}^l$ into h subspaces with different learnable linear projections, such that:

$$\text{MH}(\mathbf{H}^l) = [\text{head}_1, \text{head}_2, \ldots, \text{head}_h]\mathbf{W}; \text{ and} \tag{2.42}$$

$$\text{head}_i(\mathbf{H}^l) = \text{Attention}(\mathbf{H}^l\mathbf{W}^{(i)}_{\text{query}}, \mathbf{H}^l\mathbf{W}^{(i)}_{\text{key}}, \mathbf{H}^l\mathbf{W}^{(i)}_{\text{value}}). \tag{2.43}$$

Notably, the number of heads h equals one for the SASRec model.

2.4 Context-Aware Sequential Recommendation

The current state-of-the-art sequential recommendation models are based on the transformer architecture [7], [14], [44]. The transformer-based recommenders embed item identifiers and positional markers. The recommender only learns the chronological order of actions to represent a user's sequence of preferences.

Each item is characterized by fundamental attributes beyond a mere identification number. For instance, a shoe may have a description of its size, brand name, or colour. In most cases, item attributes are keyword descriptions in tabular form. Several other works have reported improved results

when using item attribute data. Fischer *et al.* [9] concatenate the item attributes (such as category, brand and genre keywords) to the item embedding and then trains on a bidirectional transformer. The tabular data is fed into the transformer by one-hot encoding the keyword descriptions. Zhang *et al.* [45] extends the item attributes to text descriptions and employs two separate self-attention blocks for item attribute data and text descriptions. The text description provides further context about the item in an unstructured format. They learn from unstructured data similar to the methods used in NLP. Singer *et al.* [46] track the changes of item attributes over time through a transformer that handles 2D input sequences. Zhou *et al.* [10] design a contrastive learning loss function to maximize the mutual information between items and attributes.

Keyword BERT4Rec (KeBERT4Rec) is an Extension of BERT4Rec [9]. KeBERT4Rec includes item attributes as input to the model. The embedding layer consists of three different learned embeddings: (i) an item identifier embedding $\mathbf{E}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}$, (ii) a positional embedding $\mathbf{P} \in \mathbb{R}^{n \times d}$ and (iii) an auxiliary information embedding $\mathbf{E}_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ [9]. The positional embedding $\mathbf{P}$ encodes the positions of the items within a user's sequential item data $\mathcal{S}_u$ [7]. The auxiliary information embedding $\mathbf{E}_{\mathcal{A}}$ is obtained from encoding categorical data into a multi-hot encoded vector. The vector is scaled to the embedding size d using a linear layer. For each item in the sequence $\mathcal{S}_u$, we have the item embedding $\mathbf{e}_v \in \mathbf{E}_{\mathcal{V}}$, the positional embedding $\mathbf{p}_t \in \mathbf{P}$ and auxiliary information embedding $\mathbf{a}_v \in \mathbf{E}_{\mathcal{A}}$ to produce the input to the transformer

$$h_t^0 = \mathbf{e}_v + \mathbf{p}_t + \mathbf{a}_v \tag{2.44}$$

where t is the position of the item v in the sequence $\mathcal{S}^u$.

2.5 Conclusion

The current state-of-the-art recommendation models are sequential-based. Sequential recommendation encodes each user's sequence of items into a latent representation using deep learning. Moreover, recommendation datasets suffer from popularity bias. Chapter 3 will show that sequential recommendation models emphasize their training on the popular set of items.

Chapter 3

Popularity Bias

Recommendation systems frequently encounter challenging datasets. The frequency of items follows a power law distribution [11]. Figure 3.1 illustrates the frequency distribution of each item across four datasets. A small number of items occupy the short head of the power law probability distribution, representing the most popular items in the dataset [12]. Hence, the distribution is heavily skewed towards a small subset of items, with most in the long tail [13]. This relationship produces an inherent bias within the dataset where a small number of items contribute towards a substantial number of interactions. The inherent bias is formally known as popularity bias [15]. Popularity bias produces a severe class imbalance between two groups: those at the short head of the distribution and those found at the long tail of the distribution.

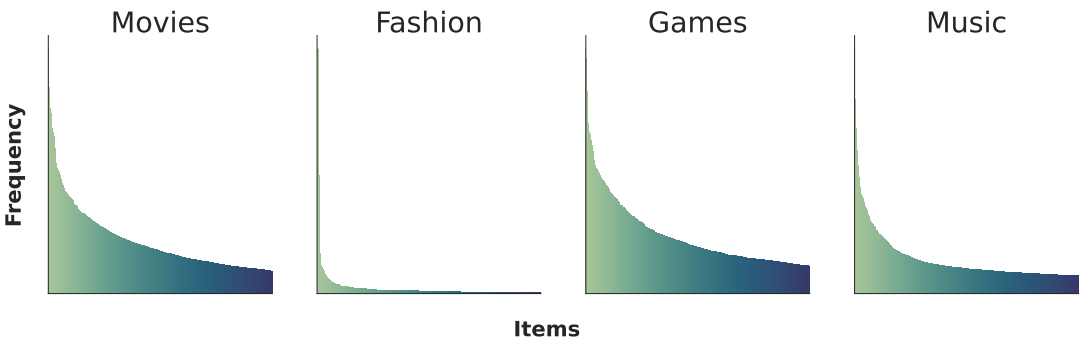


FIGURE 3.1: Depicts the frequency of each item in a dataset ordered by its popularity. Each dataset follows a power law distribution. A smaller fraction of items appear more frequently than the majority of items.

3.1 Collaborative Filtering

Recent studies have found that recommendation models that perform well on accuracy metrics focus their recommendations on a tiny fraction of the item spectrum [47]. Popular items are recommended even more frequently than their popularity would warrant [48].

Collaborative Filtering (CF) based models struggle with popularity bias [12]. Over time, these models may develop an assumption that frequently occurring items inherently carry the most relevance. Essentially, 1) they struggle to recommend niche items [17], 2) they make popular items become even more popular [49], and 3) hence, they focus their recommendations on a small subset of items [18].

While popular items constitute a part of good recommendations, they also reduce the diversity of recommended items [11]. Research in the social sciences argues that besides ensuring the relevance of the recommendation list to the user, it is essential to keep the user engaged as they browse through the list of items [50]. Furthermore, overly recommending popular items will hurt user experience and could lead to unfairness in recommendation systems [48], [51].

Several investigations have focused on addressing popularity bias [15], [16]. Nonetheless, only a few studies have incorporated bias mitigation techniques within the context of sequential recommendations [13], [19]. The following chapter will show that the performance of sequential recommendation models is optimized on the popular set of items [13]. These models are good at modelling users who prefer popular items while performing inadequately for those with preferences characterized by novelty [52].

3.2 Sequential Recommendation

Sequential recommendation models are the current state-of-the-art approach [7], [14], [39], [40]. Sequential models view the items as a sequence of interactions. Specifically, let $\mathcal{S}^u = \{v_1^u, v_2^u, \ldots, v_n^u\}$ be the sequence of items for user u such that n is the maximum sequence length, v_t^u is the item identification number obtained from the set of items $\mathcal{V}$ and t is the position of the item in the user's sequence $\mathcal{S}^u$. Hence, we treat each item v_t^u as an indicator of its occurrence within the user's sequence $\mathcal{S}^u$. Sequential models aim to learn the latent features of the item v_t^u based on the history of items

$v^u_{1:t-1} = \{v^u_1, v^u_2, \ldots, v^u_{t-1}\}$ that precedes it. Let $\mathcal{U}_v$ be the set of users who have interacted with the item v. Then user $u \in \mathcal{U}_v$ selected the item v at time t as well as the items $v_{1:t-1}$ that precedes the item v^u_t. Let $\mathcal{H}_v = \{v^u_{1:t-1} | u \in \mathcal{U}_v\}$ be the set of histories that precedes the item v_t over all the users that have interacted with the item v. The task of sequential recommendation models is to learn the latent features of item v based on its set of histories $\mathcal{H}_v$.

If v_{pop} is a popular item and v_{niche} is a niche item with fewer interactions, then:

$$|\mathcal{H}_{v_{\text{pop}}}| \gg |\mathcal{H}_{v_{\text{niche}}}| \tag{3.1}$$

that is the number of histories $\mathcal{H}_{v_{\text{pop}}}$ that precedes the popular item v_{pop} is always much greater than the number of histories that precedes the niche item. Hence, the training data for a popular item is substantially greater than that of a niche or less popular item.

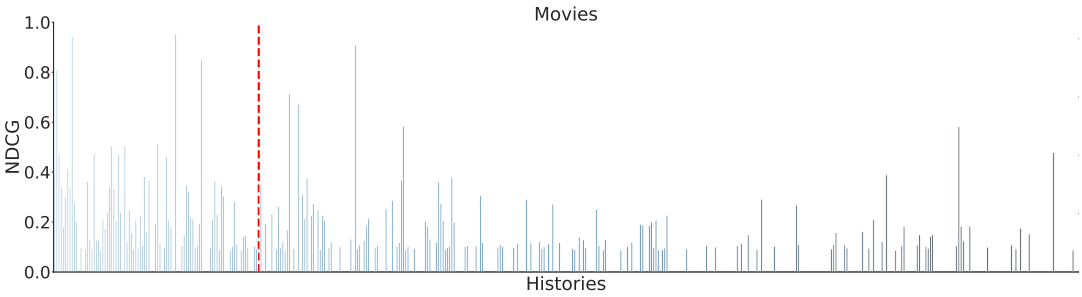
From here onwards, we define an item's popularity by its number of histories:

1. A popular item has **more histories**.
2. A less popular or niche item has **fewer histories**.

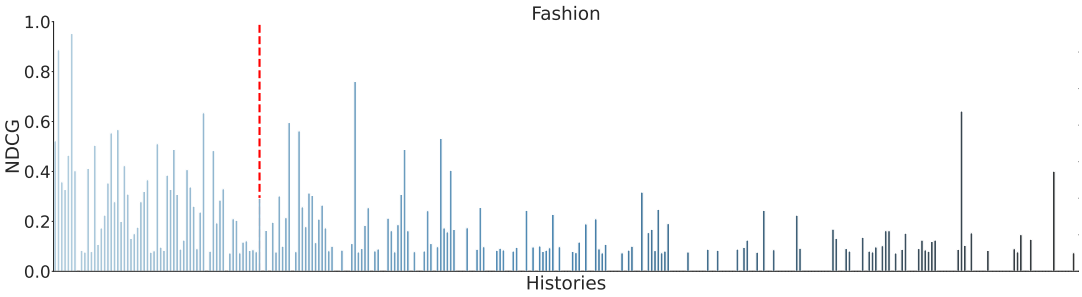
Deep learning-based classifiers are known to emphasize their training on the majority set within the training data [12]. A larger set of histories $\mathcal{H}_v$ contributes to a more comprehensive understanding of item v. Deep learning models can capture more diverse patterns and characteristics of a particular item, ultimately enhancing the quality and richness of its latent features. Conversely, the features of items with fewer histories perform poorly.

Figure 3.2 depicts the performance of the current state-of-the-art sequential recommendation model (SASRec). There exits a direct relationship of a higher NDCG@100 score for items with a higher number of histories. The more training data available for item v, the higher the prediction accuracy. Hence, sequential models develop an assumption that highly interacted with items inherently carry the most relevance. Just like traditional models, sequential models 1) struggle to recommend unpopular items, 2) make popular items become even more popular and 3) focus their recommendations on a small subset of items.

As mentioned, the frequency of items within the recommendation dataset follows a power law distribution; refer to Figure 3.3. Following the approach made by Abdollahpouri, Burke, and Mobasher [11], this work selects the



(A) NDCG@100 for the Movies Dataset



(B) NDCG@100 for the Fashion Dataset

FIGURE 3.2: Presents NDCG@100 score achieved for each item ranked by their histories (largest to smallest). Reflects a direct relationship of a higher NDCG score for items with a large number of histories. The red line depicts a threshold of 0.2 which separates frequently occurring items from less frequent items

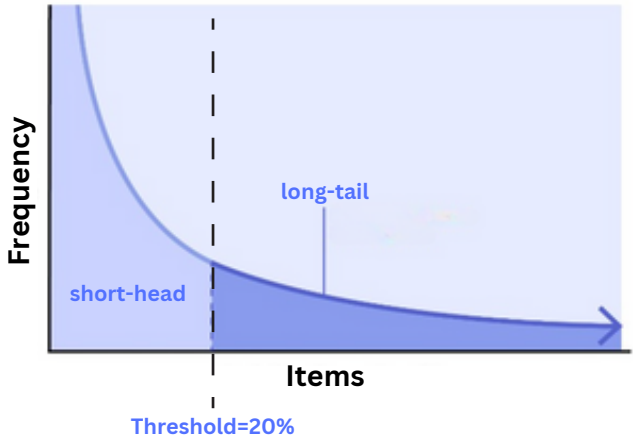


FIGURE 3.3: Illustrates the power law distributions of an actual recommendation dataset. The threshold of 20% divides the short head from the long tail

first 20% of items from the power law distribution, and we refer to them as popular items (items with more histories). The remaining items found at the long tail of the power law distribution constitute less popular or niche items (items with fewer histories). These items are the last 80% of items from the distribution.

3.3 Preference Towards Unpopular Items

One could argue that recommending popular items is not a concern, as these items have earned their popularity by being highly relevant to many users. However, this point is not entirely true. We found that users with longer sequences prefer less popular items over time.

Take the distribution of the frequency of items in Figure 3.1. Say we distinguish the popular set of items $\mathcal{V}_{\text{pop}}$ from the less popular or niche set of items $\mathcal{V}_{\text{niche}}$. Then, for each position t in the sequence $\mathcal{S}^u$, we classified an item as popular $v_t \in \mathcal{V}_{\text{pop}}$ or niche $v_t \in \mathcal{V}_{\text{niche}}$ based on which set it belongs to. The counts of popular and niche items are then tallied for each position across all sequences. Using this simple setting, we obtained the average ratio of popular to niche items for each position over all the sequences as

$$\text{Ratio}(t) = \frac{\sum_{u=1}^{|\mathcal{U}|} \mathbb{1}_{\mathcal{V}_{\text{pop}}}(v_{ut})}{\sum_{u=1}^{|\mathcal{U}|} \mathbb{1}_{\mathcal{V}_{\text{niche}}}(v_{ut})} \tag{3.2}$$

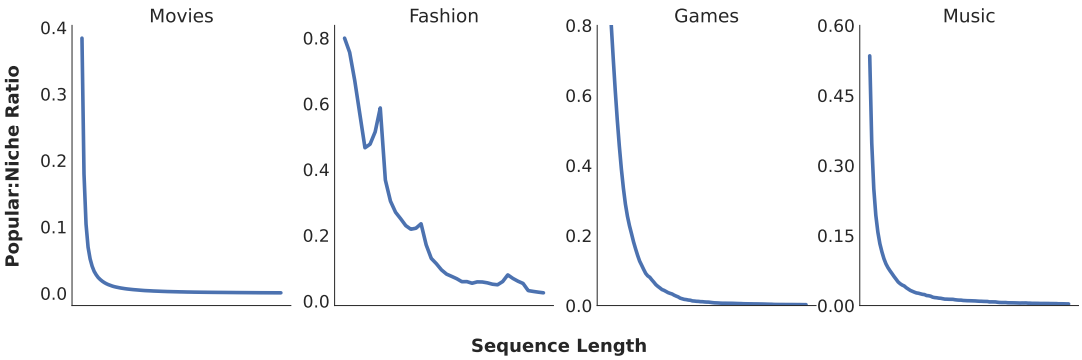


FIGURE 3.4: Plots the ratio of popular to niche items within each position in the sequence. The ratio decreases as the sequence length increases which shows users shift their preference from popular items to unpopular items overtime

where v_{ut} is the item that user u interacted with at position t. A high ratio means that position t predominately consists of popular items, and conversely, a low ratio means position t consists of mostly less popular items.

Figure 3.4 plots this ratio for each position t in the sequence. It is evident that the higher the position of item v, the lower the ratio of popular to niche items. The graph resembles an inversely proportional graph. As the position increases, the number of popular items decreases while the number of niche items increases. As users interact with more items over time, they tend to select more less popular items. This relationship persists throughout all four datasets used. In the appendix section, we plot the same graph but with a higher threshold α value - which inevitably leads to more items in the popular set and fewer items in the less popular set, as depicted in Figure A.1. The inverse relationship still holds even at a threshold of $\alpha = 0.8$. As the sequence length increases, the ratio decreases slowly and then drops for sequences longer than 100. Users prefer less popular items over time, even if the set consists of a small fraction of items.

TABLE 3.1: Results from fitting a linear regression curve for each position in the sequence against the ratio

Dataset	Movies	Fashion	Games	Music
No. Observations	331,140	23,154	115,657	109,681
Ave. Sequence length	67	6	181	116
R-Squared	0.824	0.320	0.717	0.560
t-statistic	1,244.270	104.288	541.506	373.338
p-value	0.000	0.000	0.000	0.000

To verify our assumption, we fit a linear regression curve for each position in

the sequence against the ratio. We found that 82% and 72% of the variation in the ratio can be explained by the position of the item for the Movies and Games datasets, respectively; see Table 3.1. However, the Fashion dataset only produced an R-squared value of 32%. The dataset has an average sequence length of only 6. Hence, most users only interact with popular items. The Music dataset produced an R-squared value of 56%. Users' preference in the Music dataset is heavily skewed towards popular items; see Section 5.1.2. Despite this, more than half of the variation can still be explained. Moreover, the p-values were all less than 0.05, indicating a significant correlation between the ratio and the position of the item.

We, therefore, concluded that *users tend to prefer less popular items over time*. Hence, it is naive to assume that the popular items are indeed the relevant items. Users change their preference towards niche items over time. These items are unique to them and make sense in their context. Hence, better representations of the context of the items could lead to better-personalized recommendations over time.

3.4 Contributions

We have shown empirically that sequential recommendation models cannot learn from items with fewer histories. Hence, they perform poorly on less popular items. We have also shown that users' preferences change towards niche items over time. Consequently, we developed a **Bi**as-Mitigated **Co**ntext-Aware Sequential **Rec**ommendation Model (BiCoRec) to address the following three issues: 1) recommending niche items whilst preventing popular items from becoming even more popular (**Bi**as Mitigation), 2) avoid recommending items based on the number of histories (**Co**ntext-Awareness) and 3) balancing the recommendation of popular and unpopular items based on the user's evolving preferences (Sequential **Rec**ommendation). The following provides a concise summary of the strategies employed to construct the proposed bias-mitigated context-aware sequential recommendation model.

1. We used a co-attention mechanism to capture the dynamic interplay between a user's sequence of preferences and their predisposition towards items of varying popularity - Bias Mitigation.
2. We incorporated multi-modal auxiliary information to improve the latent description of each item. The model effectively recommends items

based on their semantic relatedness and not their frequency count - Context Awareness.

3. The data shows that users tend to prefer less popular items over time. Hence, we develop a novel strategy to adapt to future changes in preferences - Sequential Recommendation.

Chapter 4

Bias-Mitigated Context-Aware Sequential Recommendation

This chapter introduces our proposed model **Bi**as-Mitigated **Co**ntext-Aware Sequential **Rec**ommender (BiCoRec). Section 4.1 outlines our bias-mitigation strategy using a co-attention mechanism. Section 4.2 discusses the model's context-awareness by integrating multi-modal auxiliary information. Section 4.3 addresses the model's adaptability to evolving user preferences through a sequential recommendation approach. Finally, the remaining subsections present the complete architecture.

4.1 Bias-Mitigation

A commonly adopted strategy in NLP and computer vision communities is to improve the transformer models by redesigning the model architecture [53]. Researchers in sequential recommendation have taken a similar approach [54]. By simply redesigning the architecture of SASRec and BERT4Rec, they can continue to report on improved performance when measured with accuracy metrics. Methods such as contrastive learning, sequence-to-sequence training, or distribution-based embeddings are adapted for sequential recommendation tasks in different application domains [44], [55]–[57].

Transformers are deep learning-based models. Deep learning-based classifiers suffer when a class imbalance in the training data exists [58]. Often, smaller populations are overlooked or underrepresented. This issue is known as the class imbalance problem and persists heavily within the recommendation setting [47].

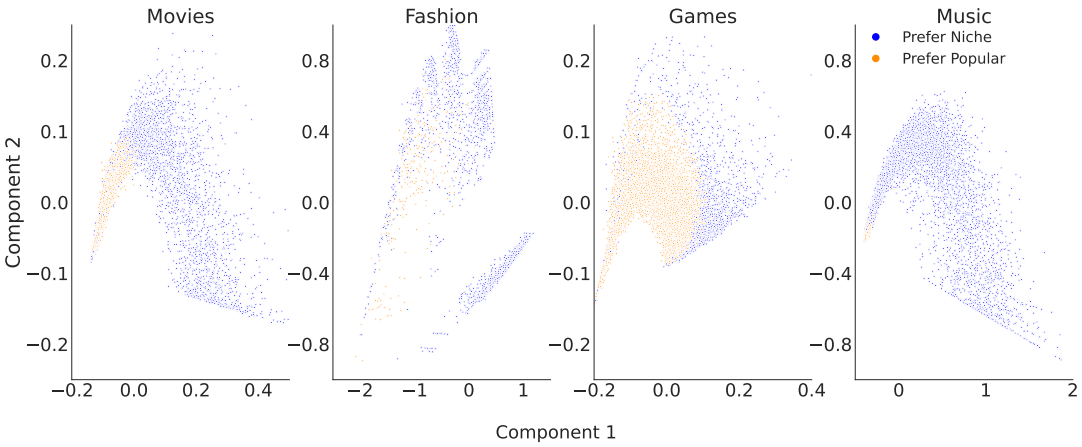


FIGURE 4.1: Two-component PCA projection of each user's sequence of TF-IDF popularity scores. From these scores, we can easily identify each user's preference for popular or unpopular items

4.1.1 Term Frequency - Inverse Document Frequency

The average preference value of an item is underpinned by its popularity [3]. We incorporated a bias mitigation technique similar to the de-biased MF model by Koren, Bell, and Volinsky [3]. We learnt the bias inherent within users and items.

We begin by measuring the popularity bias with the user's sequence. We calculate each item's Term Frequency - Inverse Document Frequency (TF-IDF). In this setting, the term is the item, the document is the sequence of items, and the corpus is all the users' sequences. The Term Frequency (TF) measures the presence of an item within a sequence. Inverse Document Frequency (IDF) of an item reflects the proportion of sequences in the corpus that contain the item. Items unique to a small percentage of sequences (e.g., niche items) receive higher importance than items common across all sequences. TF-IDF balances TF and IDF such that the score reflects the importance of an item for a sequence in the corpus. For each user, we obtain the TF-IDF scores of each item in the user's sequence and refer to the sequence of scores as the popularity scores $\mathbf{Q} \in \mathbb{R}^{|\mathcal{U}| \times n}$. Given an item v_t from the user's sequence of items $\mathcal{S}^u$ we have the TF-IDF score q_{ut} that describes the item's popularity in relation to other users. Hence, the vector $\mathbf{q}_u$ describes the user's preference for popular and unpopular items.

We applied PCA to test the effectiveness of our TF-IDF popularity scores. Figure 4.1 depicts a 2-component visualization of the scores. We grouped the users based on their preference for popular and less popular items. The

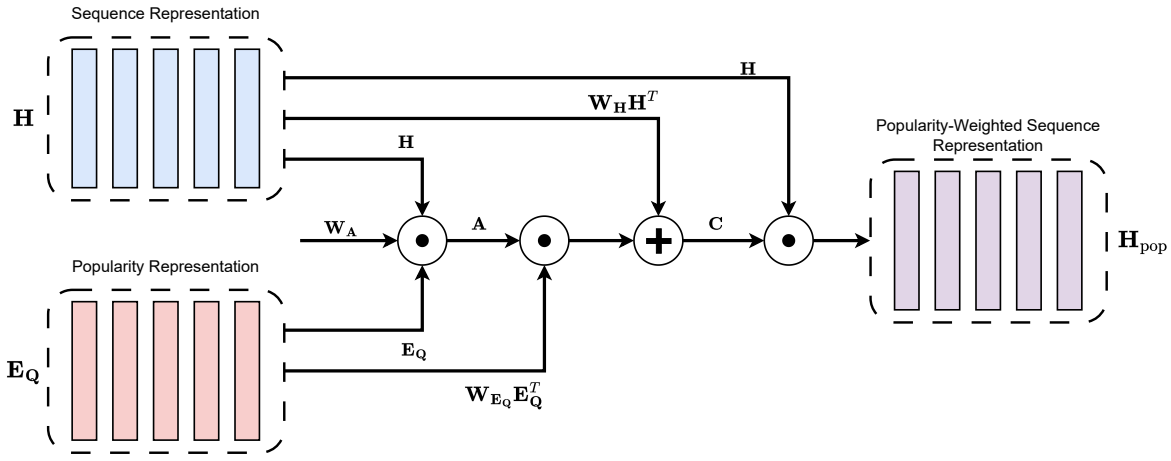


FIGURE 4.2: The parallel co-attention mechanism attends to the sequence and popularity representation simultaneously and then produces a popularity-weighted sequence representation. The $\odot$ symbol represents the dot product operation and $\oplus$ refers to the summation operation.

TF-IDF popularity scores can successfully separate users who prefer popular items from those who prefer less popular items. Interestingly, more users prefer less popular items for each dataset.

We learnt an explicit popularity embedding that encapsulates each user's preferences for popular items, such that:

$$\mathbf{E_Q} = \mathbf{qW} + \mathbf{b} \tag{4.1}$$

where $\mathbf{E_Q} \in \mathbb{R}^{|\mathcal{U}| \times d}$. However, we do not fuse the popularity embedding into the transformer input embedding. Bias is already a characteristic inherent within users and items and should be reflected within their features [3]. The transformer inadvertently absorbs this bias. Therefore, we incorporate the mitigation of bias explicitly into the user's sequential representation. This approach informs the model about the inherent bias present in each user's sequence.

4.1.2 Co-Attention Mechanism

Co-attention uses the sequence representation to guide the popularity attention and the popularity representation to guide sequence attention. The

sequence representation informs the model's understanding of item popularity; concurrently, the item popularity context informs the model's understanding of user sequence. The dual-guidance mechanism captures the dynamic interplay between a user's historical interactions and their predisposition towards various popular items.

Figure 4.2 depicts a diagram of the co-attention mechanism. We used parallel co-attention to attend to the sequence and popularity simultaneously [59]. We calculated the similarity between sequence and popularity features at all pairs of sequence and popularity positions. That is, given the learnt sequence representation $\mathbf{H} \in \mathbb{R}^{n \times d}$ obtained from point-wise FFN and popularity representation $\mathbf{E_Q}$, produce the affinity matrix

$$\mathbf{A} = \tanh(\mathbf{H}\mathbf{W_A}\mathbf{E_Q}^T) \tag{4.2}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{W_A} \in \mathbb{R}^{d \times d}$ are the weights. We consider the affinity matrix to be a feature that we use to learn the sequence and popularity attention maps. Then, we transformed the popularity attention space into the sequence attention space to produce the attention map

$$\mathbf{C}_{\text{map}} = \tanh(\mathbf{W_H}\mathbf{H}^T + (\mathbf{W_{E_Q}}\mathbf{E_Q}^T)\mathbf{A}) \tag{4.3}$$

where $\mathbf{C}_{\text{map}} \in \mathbb{R}^{k \times n}$, $\mathbf{W_H}, \mathbf{W_{E_Q}} \in \mathbb{R}^{k \times d}$ are the weights that transform the sequence $\mathbf{H}$ and popularity $\mathbf{E_P}$ features. Then we get the attention probabilities

$$\mathbf{C}_{\text{prob}} = \text{softmax}(\mathbf{W_H}^T\mathbf{C}_{\text{map}}) \tag{4.4}$$

such that

$$\mathbf{H}_{\text{pop}} = (\tau\mathbf{C}_{\text{prob}})\mathbf{H} \tag{4.5}$$

represents the popularity-weighted sequence representation obtained through parallel co-attention, and τ is the temperature parameter.

4.2 Context-Awareness

4.2.1 Multi-Modal Auxiliary Information

The contextual information of items can exist in tabular, text, audio, and/or visual formats. The additional contextual data is referred to as *auxiliary information*. Each data format offers a distinct context of the item. For instance, the

text description may provide detailed specifications or narrative content, enhancing understanding through qualitative information. On the other hand, images offer a visual representation, capturing aspects like design, colour, or style that text cannot convey [60].

Current context-aware sequential recommendation models are limited to leveraging tabular data or text descriptions. Tabular data (or structured data) consists of dense numerical features, sparse categorical features and weak correlation amongst the features compared to the spatial or semantic relationship in images or speech [61]. Deep learning methods do not outperform gradient-boosted tree ensembles for classification and regression problems with tabular data [62].

We designed a rich contextualized embedding space by exploiting comprehensive knowledge from item auxiliary information. These diverse data types lead to a more holistic and multi-dimensional view of the item's characteristics.

To fuse auxiliary information into the transformer architecture, we took an embedding approach inspired by KeBERT4Rec [9]. KeBERT4Rec only models item attributes in tabular form as auxiliary information. This study used auxiliary information from different modalities. For each item v, we consider any available text, image, audio and/or tabular data as auxiliary information. Then we concatenate each modality into a single vector $\mathbf{a}_v \in \mathcal{A}$.

We obtained each modality's vector representation from a pre-trained language or vision model. [1] We used BERT to extract text vector representations [64]. Given a set of words w_i that describe item v, we concatenate a special symbol [CLS] to produce

$$\mathbf{a}_v^{text} = \text{BERT}([\text{CLS}]; w_1^v, \ldots, w_{N_{text}}^v) \tag{4.6}$$

where N_{text} is the maximum number of words, ";" denotes the concatenation operation, and $\mathbf{a}_v^{text}$ is the final hidden representation corresponding to the first input token, which is the [CLS] token. The [CLS] token represents the contextual features of the entire sentence and is often used for other downstream tasks within NLP.

We used a Vision Transformer (ViT) to extract image vector representations. ViT divides the input image into multiple fixed-size patches and projects

[1] The pre-trained models are made available through HuggingFace [63].

each patch into a fixed-length vector [65]. Similar to BERT, a special symbol $[\text{patch}_{\text{CLS}}]$ is concatenated to the fixed-length vector of patches resulting in

$$\mathbf{a}_v^{image} = \text{ViT}([\text{patch}_{\text{CLS}}]; \text{patch}_1^v, \ldots, \text{patch}_{N_{\text{image}}}^v) \tag{4.7}$$

where N_{image} is the maximum number of image patches and $\mathbf{a}_v^{image}$ is the final hidden representation corresponding to the first input $\text{patch}_{\text{CLS}}$.

We used wave2vec to extract audio vector representations [66]. Given the raw waveform, we extracted audio vector representations using a designated symbol $[\text{audio}_{\text{CLS}}]$, such that

$$\mathbf{a}_v^{audio} = \text{wave2vec}([\text{audio}_{\text{CLS}}]; \text{audio}_1^v, \ldots, \text{audio}_{N_{\text{audio}}}^v) \tag{4.8}$$

where, N_{audio} signifies the maximum number of audio tokens, and $\mathbf{a}_v^{audio}$ represents the final hidden representation corresponding to the first input $[\text{audio}_{\text{CLS}}]$. Since deep learning methods do not outperform gradient-boosted tree ensembles for classification and regression problems with tabular data, we considered the attributes of a table. We flattened the key-value attribute pairs within the table into a sentence. That is, for each item, we have the keyword w^{key} and its associated value w^{value}.

We used data2vec to extract vector representations from the sentence [67]. data2vec adopts a multi-modal self-supervised learning framework. Instead of predicting modality-specific targets, it predicts contextualized latent representations that contain information from the entire input. The final vector representation of the attribute data

$$\mathbf{a}_v^{table} = \text{data2vec}([\text{CLS}]; (w_1^{key}, w_1^{value}), \ldots, (w_{N_{\text{columns}}}^{key}, w_{N_{\text{columns}}}^{value})) \tag{4.9}$$

where N_{columns} is the maximum number of columns and $\mathbf{a}_v^{table}$ is the final hidden representation corresponding to the first input [CLS].

We concatenate the text, image, audio or tabular vector representations into a single vector. Our previous work compared a summation to a concatenation approach [8]. We found that a summation of vector representations often results in information loss. The single-head attention mechanism benefits from longer and more descriptive modality embeddings. Hence, we concatenated

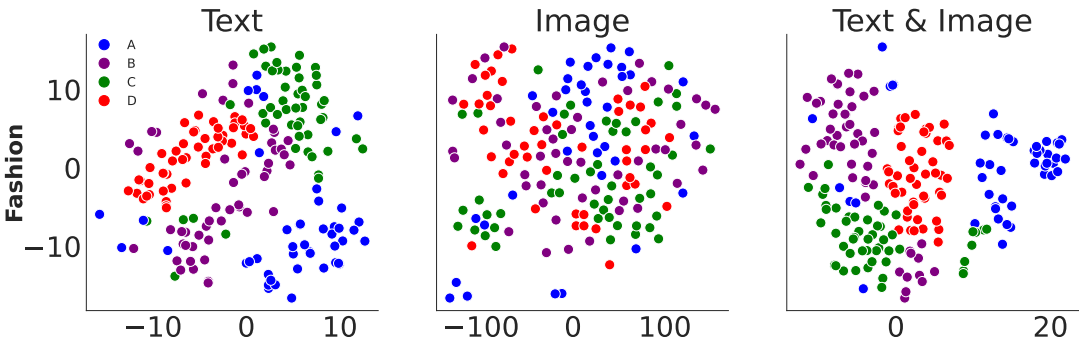


FIGURE 4.3: Illustrates a 2-component t-SNE conducted over the text, image, and concatenated text-image item representations. Each representation was derived from a pre-trained model. Notably, the representations demonstrate a clear ability to differentiate themselves based on brand categories, with particularly pronounced distinctions observed in the text-image concatenations.

the vector representations to obtain the final set of auxiliary information representations:

$$\mathbf{a}_v^{\text{Movies}} = [\mathbf{a}_v^{\text{text}}; \mathbf{a}_v^{\text{table}}; \mathbf{a}_v^{\text{image}}] \tag{4.10}$$

$$\mathbf{a}_v^{\text{Fashion}} = [\mathbf{a}_v^{\text{text}}; \mathbf{a}_v^{\text{table}}; \mathbf{a}_v^{\text{image}}] \tag{4.11}$$

$$\mathbf{a}_v^{\text{Games}} = [\mathbf{a}_v^{\text{text}}; \mathbf{a}_v^{\text{table}}] \tag{4.12}$$

$$\mathbf{a}_v^{\text{Music}} = [\mathbf{a}_v^{\text{text}}; \mathbf{a}_v^{\text{audio}}] \tag{4.13}$$

where $\mathbf{a}_v \in \mathcal{A}$ is the concatenated auxiliary information vector representation for item v over four different datasets; Movies, Fashion, Games and Music.

4.2.2 Semantic Relatedness

The multi-modal auxiliary information representations help improve the latent description of each item. Thus, the model can learn the semantic relatedness between the items. This section demonstrates that the vector representations in $\mathcal{A}$ exhibit some semantic relatedness.

Consider the Fashion dataset as described in Chapter 5.1.2. The dataset consists of fashion products made by different designer brands. Each item consists of an image, text description and item attribute data in tabular form. We aim to determine whether we can differentiate the items based on their brand names using the image, text, or both modalities.

We selected four brands to run our test on. We sampled 50 items from each

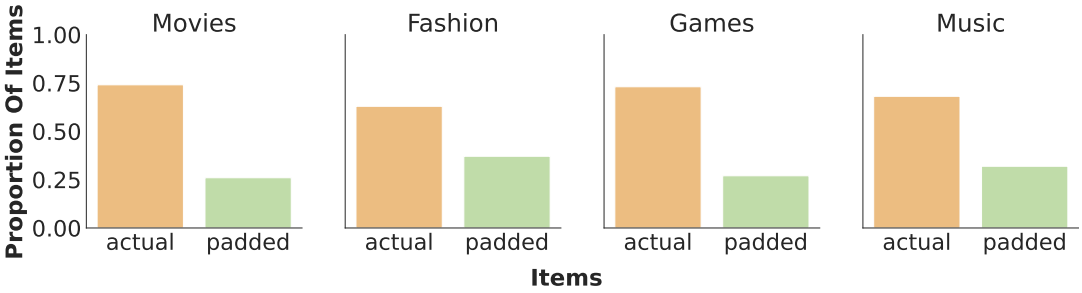


FIGURE 4.4: Depicts the proportion of actual and padded items across all the sequences. The padded items are null entries. More than 25% of the items in a user's sequence are padded items.

brand. We had a total of 200 items over the four brands. For each of the 200 items, we obtained their text, image and joint vector representations. We applied t-distributed Stochastic Neighbor Embedding (t-SNE) and generated a 2-component visualization of the features, as illustrated in Figure 4.3.

The text vector components in Figure 4.3 demonstrate an apparent ability to distinguish each item into their respective brands. Conversely, some items in brand A appear to be mixed with brands B and C. On the other hand, the image representations show a more mixed distribution, making it challenging to identify items by their brand names. However, by concatenating the text and image representations, the items are better separated into their respective brands. The vector components of items belonging to the brands A, B and C are now clearly distinct.

By leveraging all multi-modal auxiliary information, we can enhance the latent description of each item. Auxiliary information facilitates the model's ability to learn the semantic relatedness between items.

4.3 Sequential Recommendation

In sequential recommendation, we often define a fixed sequence length n that represents each user's list of interactions. If a user has interacted with more items than n, we truncate their sequence of items. Otherwise, if a user has interacted with fewer items than n, we pad the sequence with zeros. As a result, many padded items exist within the user's sequence of items. Figure 4.4 depicts the percentage of actual and padded items in each dataset.

4.3.1 Next, Mask and Padded Items

The current state-of-the-art models simply ignore the padded items. They learn from incomplete data. For instance, SASRec predicts the representation of the padded items but does not attempt to train on them since their true items are unknown. We have some expected item v at time t in $\mathcal{S}^u = \{v_1^u, ..., v_n^u\}$, such that

$$v_t = \begin{cases} \text{<pad>} & \text{if } v_t \text{ is a padded item} \\ \text{<next>} & \text{the next item given our knowledge of previous } t-1 \text{ items} \end{cases} \tag{4.14}$$

then SASRec applies binary cross entropy loss as the objective function [7]:

$$-\sum_{u \in \mathcal{U}} \sum_{t \in \{1,...,n\}} \left[\log(\sigma(\hat{y}_{uv|t})) + \sum_{v' \notin \mathcal{S}^u} \log(1 - \sigma(\hat{y}_{v'u})) \right] \tag{4.15}$$

where v' is a randomly sampled negative item such that $v' \notin \mathcal{S}^u$, that is, the user has not interacted with the item. SASRec ignores the padded item $v_t =$ <pad>. SASRec only trains on $v_t =$ <next>, which is when we have the previous items to learn from.

BERT4Rec avoids training on negative samples by using the Cloze task (or Masked Language Model). For each training step, BERT4Rec randomly masks a proportion of all items in the input sequence [68]. Given an item, replace it with a special token "[mask]" and then predict the original IDs of the masked items based solely on its left and right context, such that:

$$v_t = \begin{cases} \text{<pad>} & \text{if } v_t \text{ is a padded item} \\ \text{<next>} & \text{the next item given our knowledge of previous } t-1 \text{ items} \\ \text{[mask]} & \text{randomly masked item} \end{cases} \tag{4.16}$$

The loss for each masked input is the negative log-likelihood of the masked targets:

$$\mathcal{L} = \frac{1}{|\mathcal{S}_m^u|} \sum_{v_m \in \mathcal{S}_m^u} -\log p(v_m = v_m^* | \mathcal{S}_0^u) \tag{4.17}$$

where $\mathcal{S}_0^u$ is the masked version of user's sequence $\mathcal{S}^u$, $\mathcal{S}_m^u$ is the random masked items in it, v_m^* is the true item for the masked item v_m [68]. This training process expands the number of samples for training the model by simply masking more items.

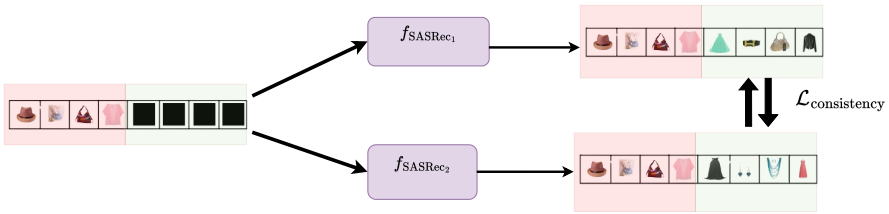


FIGURE 4.5: Depicts a user's sequence of items with padded items represented in black. Two SASRec models initialized with different weights predict the padded items (in green). The unsupervised loss function ensures consistency of predictions between the two models.

However, BERT4Rec still ignores the padded items $v_t =$ <pad>, which make up a large portion of our datasets. Figure 4.4 shows that more than 25% are padded items. Since a sequence is arranged from oldest to newest items, the padded items always precede the actual items. This study found that *users tend to prefer unpopular items over time*. If we simply ignore the padded items, then we fail to train our model on the changing future preferences of that user.

Suppose we train using the <next> token; then, we would only learn how to make short-term predictions. Conversely, training with the [mask] token involves learning from even shorter sequences to make short-term predictions. To address this issue, we propose to train the model using the entire user's sequence of items. We learn from both padded and actual items.

4.3.2 Cross-Pseudo Supervision

We may view the actual items as labelled data and the padded items as unlabeled data. The existence of labelled and unlabeled data transforms the problem into a semi-supervised learning paradigm. We used a novel technique that trains on the padded items. The result is a much larger number of training signals.

We adopted a semi-supervised learning paradigm similar to that used by Chen *et al.* [53] for the image semantic segmentation task. Semantic segmentation training data requires pixel-level manual labelling, which is time-consuming and impractical. Similarly, for recommendation datasets, it is not possible to obtain the preference labels of each user for all items. Chen *et al.* [53]

propose a consistency regularization approach that enforces consistency of the predictions under some perturbation. Two segmentation networks that share the same structure, but are initialized differently, are trained on the labelled data. The predicted outputs from the two networks are supervised separately by the corresponding ground-truth segmentation map. The two networks are also trained on unlabeled data, but the outputs are two pseudo-segmentation maps. The objective is to enforce the consistency between these two maps produced by the two parallel segmentation networks.

We utilized cross-pseudo supervision for the sequential recommendation task. Figure 4.5 depicts a diagram of passing a sequence of items through a cross-pseudo supervision network. The main objective of cross-pseudo supervision is to ensure consistency of predictions from both actual and padded items. We use the dual parallel cross-pseudo supervision network structure as [53]

$$(\mathcal{S}_u, \mathcal{A}) \rightarrow f_{\text{SASRec}}((\mathcal{S}_u, \mathcal{A}) : \theta_1) \rightarrow \hat{\mathbf{S}}_u^{(1)} \rightarrow \mathbf{M}_u^{(1)} \tag{4.18}$$

$$\searrow f_{\text{SASRec}}((\mathcal{S}_u, \mathcal{A}) : \theta_2) \rightarrow \hat{\mathbf{S}}_u^{(2)} \rightarrow \mathbf{M}_u^{(2)} \tag{4.19}$$

where two f_{SASRec} networks maintain the same structure, but their weights, i.e., θ_1 and θ_2, are initialized differently. $\hat{\mathbf{S}}_u = \{\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_n \mid \hat{\mathbf{s}} \in \mathbb{R}^{|\mathcal{V}|}\}$ is the pseudo-sequence confidence map obtained from the network and $\mathbf{M}_u = \{\mathbf{m}_1, \ldots, \mathbf{m}_n \mid \mathbf{m} \in \{0,1\}^{|\mathcal{V}|}\}$ is the predicted one-hot label map. The dual network ensures consistent predictions across differently initialized networks for the same input sequence. Unlike the current state-of-the-art sequential models, we do not limit the training of our model to randomly selecting a negative or masked item. Instead, we predict the preference score $\hat{\mathbf{s}}_t$ for all items $v \in \mathcal{V}$ for the position t in the users sequence $\mathcal{S}^u$ that satisfies

$$v_t = \begin{cases} \text{<pad>} & \text{if } v_t \text{ is a padded item} \\ \text{<next>} & \text{the next item given our knowledge of previous } t-1 \text{ items} \end{cases} \tag{4.20}$$

and then we train for the padded and next items in the sequence.

The objective function optimizes two loss functions; supervision loss $\mathcal{L}_{\text{supervised}}$ and consistency loss $\mathcal{L}_{\text{consistency}}$. Supervision loss uses the standard cross-entropy loss ℓ_{ce} on the labelled data over the two parallel networks

$$\mathcal{L}_{\text{supervised}}^u = \frac{1}{|\mathcal{S}^u|} \sum_{t \in \{1,\ldots,n\}} (\ell_{ce}(\hat{\mathbf{s}}_t^{(1)}, \mathbf{s}_t) + \ell_{ce}(\hat{\mathbf{s}}_t^{(2)}, \mathbf{s}_t)) \tag{4.21}$$

where t is the position of the item that satisfies $v_t =$ <next> and $\mathbf{s}_t \in \{0,1\}^{|\mathcal{V}|}$ is the ground truth one-hot label that indicates the position of the actual item v. Then, we adopt the consistency loss to expand our training data. The cconsistency loss is bidirectional:

$$\mathcal{L}^{u}_{\text{consistency}} = \frac{1}{|\mathcal{S}^u|} \sum_{t \in \{1,\ldots,n\}} (\ell_{ce}(\hat{\mathbf{s}}^{(1)}_t, \mathbf{m}^{(2)}_t) + \ell_{ce}(\hat{\mathbf{y}}^{(2)}_t, \mathbf{m}^{(1)}_t)) \tag{4.22}$$

where t is the position of the item that satisfies $v_t =$ <pad>. The one-hot label output $\mathbf{M}_1$ from one network $f(\theta_1)$ is used to supervise the pseudo-sequence confidence map $\hat{\mathbf{S}}_2$ of the other network and so on. We jointly optimize the supervision and consistency loss

$$\mathcal{L}_{\text{crosspseudo}} = \mathcal{L}_{\text{supervised}} + \lambda \mathcal{L}_{\text{consistency}} \tag{4.23}$$

where λ is a hyper-parameter that controls the contribution from each component.

4.4 User Representation

SASRec and BERT4Rec do not learn an explicit user embedding [7], [14]. They argue that the sequence representation is an implicit user embedding. They also found that the inclusion of an explicit user embedding did not reflect an improvement in their results.

However, traditional approaches, such as MF, FPMC and Caser, often encourage using explicit user embeddings to capture user characteristics [33], [36], [40]. The user characteristics are then fused with the item characteristics through some addition or concatenation approach. The fusion of different knowledge domains encourages the model to learn better personalized information based on the context derived from both the user and item features.

This study argues that transformer-based recommenders could benefit from explicit user embeddings to enhance their ability to capture and utilize user-specific characteristics, which are not directly observable from interaction sequences alone. While implicit embeddings derived from user actions can encapsulate temporal patterns, explicit user embeddings can provide a deeper understanding of user preferences, especially in sparse temporal data. Incorporating explicit user embeddings enriches the model's contextual awareness of user characteristics.

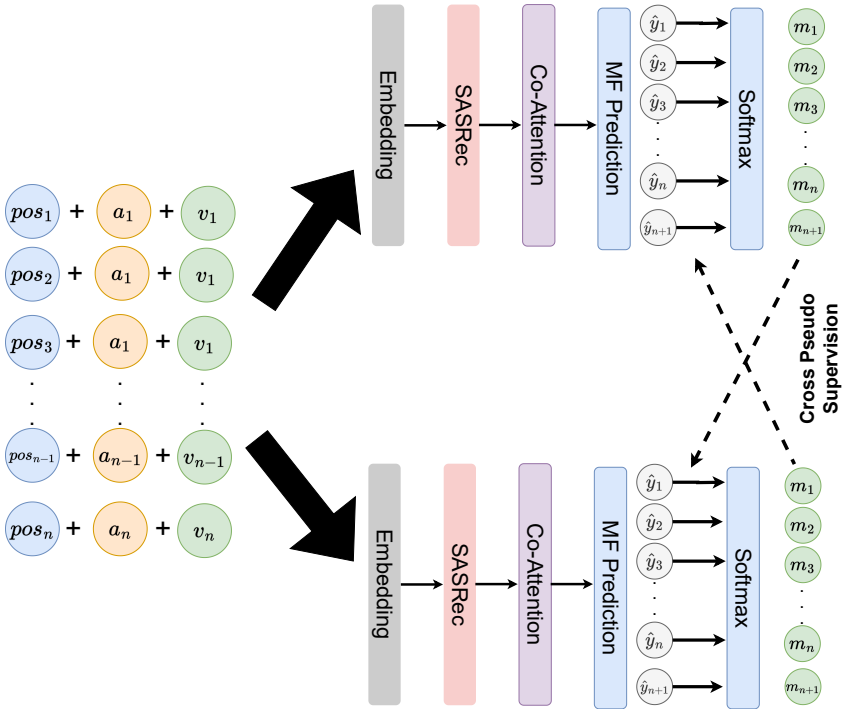


FIGURE 4.6: An illustration of the architecture of BiCoRec. The summed positional, auxiliary and item embeddings are passed into four layers to obtain the sequence representation. Then cross-pseudo supervision is used to enforce consistency of predictions between the two networks.

Different from SASRec and BERT4Rec, we include an explicit user embedding $\mathbf{e}_u \in E_{\mathcal{U}}$ such that the final generated embedding is $\mathbf{h}_t + \mathbf{e}_u \; \forall t \in \{1, 2, \ldots, n\}$. Including item and user characteristics ensures the model exploits knowledge from both user and item domains.

4.5 Architecture

In previous sections, we have discussed our main methodologies on how to mitigate bias, incorporate context awareness and learn better representations of users' evolving preferences in sequential recommendation models. In this section, we put together the proposed solutions into a unified model, i.e., the proposed model termed BiCoRec. Figure 4.6 presents the architecture of BiCoRec. It is divided into four functional components, the input embedding layer, the attention layer and the predictive layer. For completeness, in the following sections, we summarize the details of the implementations of each component.

4.5.1 Embedding Layer

Let $\mathcal{S}^u = \{v_1^u, v_2^u, \ldots, v_n^u\}$ be a sequence where n is the predefined maximum sequence length. $\mathcal{S}^u$ is obtained from truncating or padding sequences which are longer or shorter than n, respectively. The embedding layer embeds the item IDs, positions and auxiliary information:

1. **Item ID** - the item identification numbers are the item identifiers v derived from a user's sequence of items they have interacted with. We produce the learnable embedding $\mathbf{E}_\mathcal{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ which maps the ID of an item to a vector of dimension d.
2. **Positions** - each position t such that $v_t^u \in \mathcal{S}^u$ and $1 \leq t \leq n$ is mapped to a learnable embedding $\mathbf{P} \in \mathbb{R}^{n \times d}$.
3. **Auxiliary information** - Given an item, we consider the text, image, audio and/or tabular data associated with the item. We obtained the pre-trained vector representations of each modality from vision, audio or multi-modal model from HuggingFace [63]. We concatenate each modality into a single vector $\mathbf{a}_v \in \mathcal{A}$ such that $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{|\mathcal{V}|}\}$ denotes the set of concatenated auxiliary information vectors corresponding to each item. Since the pre-trained vector embeddings $\mathbf{a}_v$ are of different dimensions, we use a linear transformation to transform the vector into d dimension:

$$\mathbf{e}_{\mathbf{a}_v} = \mathbf{a}_v \mathbf{W}_1 + \mathbf{b}_1 \text{ such that} \tag{4.24}$$

$$\mathbf{E}_\mathcal{A} = \begin{bmatrix} \mathbf{e}_{\mathbf{a}_1} \\ \mathbf{e}_{\mathbf{a}_2} \\ \cdots \\ \mathbf{e}_{\mathbf{a}_{|\mathcal{V}|}} \end{bmatrix} \tag{4.25}$$

   is the auxiliary information embedding where $\mathbf{E}_\mathcal{A} \in \mathbb{R}^{|\mathcal{V}| \times d}$ over all the items. The weights $\mathbf{W}_1$ and bias $\mathbf{b}_1$ are fixed.

The final input embedding $\mathbf{E} \in \mathbb{R}^{n \times d}$ is produced by summing the item $\mathbf{E}_\mathcal{V}$, position $\mathbf{P}$ and auxiliary embeddings $\mathbf{E}_\mathcal{A}$ to produce

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{v_1^u} + \mathbf{p}_1 + \mathbf{e}_{\mathbf{a}_{v_1^u}} \\ \mathbf{e}_{v_2^u} + \mathbf{p}_2 + \mathbf{e}_{\mathbf{a}_{v_2^u}} \\ \cdots \\ \mathbf{e}_{v_n^u} + \mathbf{p}_n + \mathbf{e}_{\mathbf{a}_{v_n^u}} \end{bmatrix} \tag{4.26}$$

relative to the item v_t^u within the sequence $\mathcal{S}^u$.

4.5.2 Attention Layer

Self-Attention

let $\mathbf{H}^0 = \mathbf{E}$ then we obtain the sequence representation

$$\mathbf{A}^l = \text{LayerNorm}(\mathbf{H}^l + \text{Dropout}(\text{SA}(\mathbf{H}^l))) \tag{4.27}$$

$$\text{Trm}(\mathbf{H}^l) = \text{LayerNorm}\left(\mathbf{A}^l + \text{Dropout}(\text{PFFN}(\mathbf{A}^l))\right) \tag{4.28}$$

$$\mathbf{H}^{l+1} = \text{Trm}(\mathbf{H}^l) \tag{4.29}$$

where l is the number of layers, SA($\bullet$) is the Self-Attention mechanism and PFFN($\bullet$) is the Point-wise FeedForward Network.

Co-Attention

We used parallel co-attention to attend to the sequence and popularity simultaneously. Given the learnt sequential representation $\mathbf{H}^{l+1} \in \mathbb{R}^{n \times d}$ and popularity representation $\mathbf{E_Q} \in \mathbb{R}^{n \times d}$ we produce the popularity-weighted sequence representation $\mathbf{H}_{\text{pop}}^{l+1}$.

User Representation

We include an explicit user learnable embedding $\mathbf{E}_\mathcal{U} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_\mathcal{U} | \mathbf{e}_u \in \mathbb{R}^d\}$ which maps the ID of a user to a vector of dimension d. We combine the sequence representation with the user's embedding such that:

$$\mathbf{H}_{\text{pop}}^{l+1} = \begin{bmatrix} \mathbf{h}_1^{l+1} + \mathbf{e}_u \\ \mathbf{h}_2^{l+1} + \mathbf{e}_u \\ \cdots \\ \mathbf{h}_n^{l+1} + \mathbf{e}_u \end{bmatrix} \tag{4.30}$$

4.5.3 Prediction Layer

We used the final sequence representation $\mathbf{H}_{\text{pop}}^{l+1}$ to predict the next item v_{n+1} given the previous n items. We adopt a matrix factorization prediction layer to predict the relevance of item v_{n+1} [7]:

$$\hat{y}_{uv_{n+1}} = \max(\mathbf{h}_{n+1}^{l+1} \mathbf{E}_\mathcal{V}^T) \tag{4.31}$$

where $\hat{y}_{uv_{n+1}}$ is the preference score of item v_{n+1} for the user u, $\mathbf{h}_{n+1} \in \mathbb{R}^d$ and $\mathbf{E}_{\mathcal{V}}^T \in \mathbb{R}^{|\mathcal{V}| \times d}$. We select the item that generated the highest preference score.

Chapter 5

Methodology

In this chapter, we undertake an extensive confirmatory experimental study to evaluate the proposed BiCoRec. A confirmatory experimental study is a structured and rigorous investigation aimed at testing the proposed model. It involves carefully controlled experiments, statistical analyses, and a systematic approach to gathering evidence that either supports or challenges the model.

We will engage in design science research. Design science focuses on creating and evaluating artifacts to address real-world problems. It involves the iterative process of designing, implementing, and evaluating solutions, often in the form of novel systems, models, or methodologies. Design science aims to contribute practical knowledge and applicable solutions to improve and advance the understanding of complex issues.

5.1 Data

5.1.1 Dataset Metrics

The interactions between users and items distinctly define each dataset's individual characteristics and nuances. We present four distinct characteristics that point to the challenges faced when modelling a recommendation dataset.

User-Item Ratio

It measures the number of users to that of items:

$$\text{user-item ratio} = \log\left(\frac{|\mathcal{U}|}{|\mathcal{V}|}\right) \tag{5.1}$$

Data Sparsity

Users typically interact with a small proportion of the available items [69]. Data Sparsity reflects on the typically large number of all possible user-item interactions compared to the relatively few observations made by each user:

$$\text{DataSparsity} = 1 - \left(\frac{|\mathbf{Y}_{uv=1}|}{|\mathcal{U}| \times |\mathcal{V}|} \right) \tag{5.2}$$

where $\mathbf{Y}_{uv=1}$ are the observed interactions between user u and item v such that $y_{uv} = 1$.

If the interaction matrix is sparse, any two users sampled from the dataset will unlikely have interacted with the same item. Hence, we struggle to find any similarities in preferences between the two users. Adding contextual information to each item helps differentiate items based on their contextual similarities [70]. Items with sparse interactions can still be compared to the other items in the dataset. In contrast, non-context-aware models often struggle with sparse data, as they lack the ability to distinguish between items with few interactions.

Sequence Sparsity

Sequence sparsity reflects on the proportion of actual items to padded items within the sequences and is formulated as

$$\text{SequenceSparsity} = 1 - \left(\frac{|\mathcal{S}^u_{v \in \mathcal{V}}|}{n} \right) \tag{5.3}$$

where $\mathcal{S}^u_{v \in \mathcal{V}}$ is the set of actual items in the sequence of size n.

If the sequence is too sparse, the probability of two users having any items in common is very low. Context-aware sequential recommendation models incorporate contextual data into item representations [10]. Hence, the model can distinguish between users' sequences by identifying similarities in context. Non-context-aware models rely on items appearing frequently across multiple user sequences to build robust item representations.

Skewness

We characterize the structural distribution of the interaction dataset with two metrics [54],

$$\text{skewness} = \frac{\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}(|\mathbf{Y}_{ui=1}| - \mu)^3}{\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}(|\mathbf{Y}_{ui=1}| - \mu)^{\frac{3}{2}}} \tag{5.4}$$

measures the asymmetry of an item frequency distribution where μ is the mean of the distribution. Skewness determines how much of the mass of the distribution is concentrated on the left or right side of the mean.

Gini coefficient

The Gini coefficient measures the concentration of an item frequency distribution [71] and is defined as

$$\text{Gini} = 1 - 2\sum_{i=1}^{|\mathcal{V}|}\frac{|\mathcal{V}| + 1 - i}{|\mathcal{V}| + 1} \times \frac{|\mathbf{Y}_{ui}|}{|\mathcal{U}| \times |\mathcal{V}|} \tag{5.5}$$

where $|\mathbf{Y}_{ui}|$ is the number of interactions belonging to the item i. A Gini coefficient value of zero represents total equality, which occurs when all items are equally popular. A value of one represents maximal inequality and occurs when only a small number of popular items have been interacted with.

Preference Value Variance

The uncertainty of the preference value of an item is measured through its variance

$$\text{Var}(\mathbf{Y}) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}(|\mathbf{Y}_{ui}| - \mu)^2. \tag{5.6}$$

Controversial items tend to have higher variance because they are characterized by a wide range of opinions among different users [72]. High variance often indicates that an item appeals strongly to certain segments of users while being unappealing to others, which leads to it being classified as a niche product [73]. Hence, high variance helps the model learn across a spectrum of different user's tastes. The model can learn to discern nuanced user preferences by analyzing items with high variance.

TABLE 5.1: The performance of each dataset against the metrics described in Section 5.1.1

Dataset	**Movies**	**Fashion**	**Games**	**Music**
Number of users	6041	3719	6708	9873
Number of items	3261	13198	30933	159002
Number of interactions	998539	24097	1217931	1143299
Ave. sequence length	165.32	6.48	181.59	115.81
Sparsity	94.93%	99.95%	99.41%	99.93
Sequence sparsity	25.97%	37.08%	26.98%	31.90%
User-item ratio	0.617	-1.26	-1.53	-2.78
Skewness	.9650	.7100	.6514	1.4294
Gini	.4580	.5136	.5603	.5556
Preference value variance	5.717	.2143	1.713	.7096

5.1.2 Datasets

In this subsection, we introduce four datasets used in this study, and their characterizations using various metrics including those presented in Section 5.1.1. Each datasets characteristics using the described metrics is presented in Table 5.1

MovieLens 1M Dataset (Movies)

MovieLens is a large-scale benchmark dataset with over 1 million user ratings on 3200 movies and TV shows [74]. The interactions in Movies datasets are usually more evenly distributed across users than items. Below are several key characteristics of the Movies dataset:

- It has a large average sequence length of 165.
- About 26% of the items in the sequence are padded items.
- A skewness of 0.9650 means that the distribution of items is moderately skewed.
- The Gini coefficient is relatively low at 0.4580 compared to the other datasets.
- The dataset produces the highest preference value variance.

The Movie dataset is fairly distributed around the preferences of the users. The users have seen a large proportion of the items. Hence, the models can infer preference by observing the tastes of other users.

Amazon Fashion Dataset (Fashion)

The Amazon Fashion dataset is a widely used benchmark dataset in the field of recommendation systems [75]. The dataset comprises more than 3700 customer reviews and ratings for over 13000 fashion products on Amazon. Its key characteristics present a few challenges:

- The dataset has a sequence length of 6 and a sequence sparsity of 37%. Hence, finding users with common items within their sequences becomes challenging.
- The interaction matrix is highly sparse because there are substantially more items than users.
- The distribution is moderately skewed.
- The Gini coefficient reflects an evenly concentrated distribution.
- The preference variance is the lowest at 0.2143. The low value suggests that the users typically engage with popular items and hence have a standard agreement in preference.

Steam Dataset (Game)

Steam Dataset was crawled from a large online video game distribution platform [7]. The dataset contains 6708 users and 30933 games. The dataset exhibits the following key properties:

- It has the highest average sequence length but low preference value variance. That is, users engage with a small fraction of the item set.
- The skewness is lower than the Movies and Fashion dataset but still moderately skewed.
- However, the dataset produces the highest Gini coefficient, suggesting users' unequal engagement of each item.
- The variance of the preference value is less than half of the Movies dataset but remains higher than the other two datasets. Hence, users still have diverse tastes in items.

LastFM Dataset (Music)

Music4All-Onion is a large-scale, multi-modal music dataset extracted from the online music platform LastFM [76]. The dataset is characterized by the following:

- The dataset has more items than users, with the lowest user-item ratio of -2.78.

- The distribution is highly skewed meaning that users engage unequally with each item.
- The Gini coefficient reflects a slightly concentrated distribution compared to the Games dataset.
- The variance of the preference values is the lowest compared to all other datasets. Hence, users have a strong preference for popular items. Learning from more niche or less popular items is difficult since their interactions are much less.

TABLE 5.2: Different types of auxiliary information for each dataset.

Dataset	Tabular	Text	Image	Audio
Fashion	✓	✓	✓	✗
Movies	✓	✓	✓	✗
Games	✓	✓	✗	✗
Music	✗	✓	✗	✓

Each dataset consists of multi-modal auxiliary information. Table 5.2 depicts the type of data each dataset provides.

5.1.3 Preprocessing

Truncation and Padding

Each user has an arbitrary number of preferred items. The user sequences are truncated or padded to a fixed length to standardize the input for each model. The fixed sequence length n is chosen based on the average sequence length across the dataset.

A sequence is truncated if its length exceeds n, or padded if it falls short of n. Excessive padding reduces the model's focus on actual user interactions resulting in increased noise. Similarly, truncating sequences can lead to a loss of valuable information, especially for users with longer sequences. In this study, we assume that the average sequence length n is adequate for capturing the preferences of most users. However, outlier users, those with either very few interactions or exceptionally long interaction histories, may experience reduced performance as a result.

5.2 Baselines

We use the baselines mentioned in Chapter 2. Each baseline is adopted from four categories: naive, Collaborative Filtering (CF), context-aware and sequential recommendation models.

5.2.1 Naive Models

Naive models often rely on basic heuristics. They are not tailored to the preferences of an individual user. The following models are included in the baslines.

1. **RandomRec** - Randomly selects an item from the entire set of items. This strategy helps serve more niche items to users but fails to provide accurate recommendations.
2. **PopRec** - Ranks items according to their popularity. It assumes that most users prefer popular items, perpetuating the popularity bias inherent in recommendation datasets.
3. **Epsilon greedy** - A simple exploration strategy widely used in reinforcement learning [77]. Epsilon Greedy hopes to balance the explorative capability of RandomRec whilst improving the model's overall accuracy with PopRec's exploitative strategy.

5.2.2 CF-Based Models

CF models use well-established statistical or machine-learning techniques. They focus on clustering users based on their interests and then recommending items from the same group of users. The following models are frequently used as benchmark models in the literature.

1. **Bayesian Personalized Ranking (BPR)** [23] - Aims to find the correct personalized ranking for all items by optimizing the maximum posterior probability. The observed entries should be ranked higher than the unobserved ones. The pairwise learning minimizes the loss between the predicted and target values while maximizing the margin between the observed and unobserved entries.
2. **Neural Collaborative Filtering (NCF)** [5] - Utilizes a multi-layer neural network architecture for collaborative filtering. NCF uses a non-linear function to learn from complex user-item interactions.

5.2.3 Context-Aware Models

Context-aware models incorporate contextual information into the model. These models can recommend items with limited interaction history.

1. **De-biased Matrix Factorization (MF)** [3] - Projects both user and item features into the same latent factor space. It is effective in dealing with large and sparse datasets. However, it can struggle with new users or items as it requires existing data to infer the latent factors. We use the De-bias MF described in Chapter 2.
2. **Factorization Machines (FM)** [4] - Uses the user-item interaction matrix with additional contextual features such as auxiliary information and previous actions. It can recommend new and less popular items. FM can estimate reliable parameters under very high sparsity.

5.2.4 Sequential Recommendation Models

Sequential recommendation considers collaborative filtering as a univariate time series prediction problem. The premise is that the sequence of items a user interacts with carries important information about their preferences and future actions. We select the following baseline models as they often represent the SOTA in this category.

1. **Translation-based Recommendation (TransRec)** [39] - TransRec models user preference and sequential dynamics under one latent component. It models a user based on the ordered set of items they have consumed, efficiently capturing how a user's preference evolves with each new item interaction.
2. **Convolutional Sequence Embedding Recommender (Caser)** [40] - Is a neural network-based recommender that employs a CNN to learn high-order MCs. The model applies convolutional operations to the embedded sequence of interacted items, enabling it to learn complex patterns and dependencies among items over successive time steps.
3. **Graph Recurrent Unit for Recommendation (GRU4Rec)** [6] - Uses RNNs to model user action sequences for session-based recommendation. GRUs are a type of RNN that effectively manages both short-term and long-term dependencies in sequential data. We treat each user's feedback sequence as a session [7].

4. **SASRec** [7] - Employs a self-attention mechanism to weigh the importance of different items in a user's interaction history. Unlike RNN-based models, it can directly model the relationship between any two items in a sequence, regardless of their distance, allowing it to capture long-range dependencies effectively. However, it may not inherently capture the sequential order of interactions, as it gives equal opportunity for all items to attend to each other. SASRec might sometimes lead to less emphasis on the more recent interactions.
5. **BERT4Rec** [14] - Employs a bidirectional Transformer to model user-item interaction sequences. Unlike traditional sequential models that process sequences in a single direction, BERT4Rec simultaneously considers both past and future interactions within a sequence, providing a more comprehensive understanding of user preferences. The bidirectional approach allows it to consider the complete context of a sequence, leading to a richer understanding of user behaviour.

5.3 Results Analysis

5.3.1 Evaluation Strategy

A typical recommendation system aims to predict the next item a user will engage with based on their historical interactions. The leave-one-out evaluation specifically simulates this by removing the last item the user interacted with and testing whether the system can successfully recommend that missing item from the remaining interactions.

Other popular alternative strategies used in machine learning such as k-fold cross-validation are not suitable for the recommendation setting. K-fold cross-validation shuffles the data, potentially using future interactions to predict past ones. The leave-one-out strategy ensures that the test interaction occurs after the training interactions, maintaining temporal consistency.

Most researchers utilize the leave-one-out strategy by selecting a relevant item from the user's preference profile and ranking it against a randomly chosen subset of items. This evaluation approach, known as sampling metrics, accelerates the computation of metrics [78]. However, the sampling metrics approach has attracted significant scrutiny in recent years [79]. Krichene and Rendle [79] argue that the routine use of sampling metrics produces high-bias and low-variance estimators of the exact metrics. They propose that

researchers evaluate their models by ranking the relevant item against the entire collection.

5.3.2 Accuracy Metrics

We assess our models using three accuracy-based metrics. We ranked the relevant item against the entire item collection for each user and computed the aggregate score attained for each metric.

Recall@N

The Recall at N (Recall@N) evaluates the proportion of users for whom a query of size N includes the relevant item [80]:

$$\text{Recall@N} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbb{1}_{\mathcal{S}^u}(\hat{v}) \tag{5.7}$$

where $\hat{v}$ is the predicted item. A high recall value means most users have seen their preferred item amongst the N recommended items.

Precision@N

Precision measures the ratio of recommended items that are relevant to the total number of recommended items [81]:

$$\text{Precision@N} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\mathbb{1}_{\mathcal{S}^u}(\hat{v})}{N} \tag{5.8}$$

A high precision score indicates that a larger number of predictions are relevant.

MRR

Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of the relevant item across all queries. The reciprocal rank is obtained by taking the position at which the first relevant item is found, such that:

$$\text{MRR} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{\text{rank}(\hat{v})} \tag{5.9}$$

where rank is the relevant item's position in the predicted recommendation list. A high MRR means the relevant item is ranked higher in the recommendation list across all users.

NDCG@N

The Normalized Discounted Cumulative Gain at N (NDCG@N) assigns higher importance to more relevant items at the top of the query [82]. NDCG emphasizes placing relevant items at the top of the list and considers the diminishing returns of relevance as we move down the list.

$$\text{DCG@N} = \frac{1}{Z} \sum_{i=1}^{N} \frac{\mathbb{1}_{\mathcal{S}^u}(\hat{v})}{\log_2(i+1)} \tag{5.10}$$

$$\text{IDCG@N} = \frac{1}{Z} \sum_{i=1}^{N} \frac{\mathbb{1}_{\mathcal{S}^u}(\hat{v})}{\log_2(i+1)} \tag{5.11}$$

$$NDCG@N = \frac{\text{DCG@N}}{\text{IDCG@N}} \tag{5.12}$$

A high NDCG suggests that the recommendation system not only ranks the first relevant item well (like MRR) but also considers the overall quality of the entire recommendation list.

5.3.3 Fairness Metrics

This section presents the evaluation metrics used to assess the quality of the recommendations. We present three distinct fairness metrics that measure variety, unfamiliarity and surprise.

Diversity@N

Diversity measures how varied the recommended items are [52]. Diversity encourages the model to recommend dissimilar items. The study adopts the Average Intra-List Distance diversity metric described as [83]:

$$\text{Diversity@N} = \frac{1}{N(N-1)} \sum_{\hat{v}_1 \in \mathcal{R}_u} \sum_{\hat{v}_2 \in \mathcal{R}_u} 1 - \text{cosine}(\mathbf{a}_{\hat{v}_1}, \mathbf{a}_{\hat{v}_2}) \tag{5.13}$$

where, $\mathcal{R}_u$ is a list of recommended items for user u and $\mathbf{a}_{\hat{v}}$ is the auxiliary information embedding for the predicted item $\hat{v}$. It outputs a value between 0, identical recommendations, and 2, very diverse recommendations.

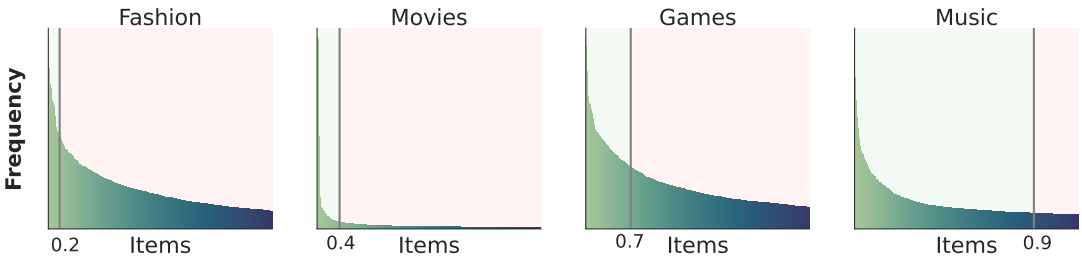


FIGURE 5.1: Illustrates the power law distributions of each dataset, accompanied by an exemplified threshold value. The green area would be the popular items, and the red area corresponds to less popular items.

Novelty@N

Novelty measures how new or unfamiliar a recommended item is to the user [52]. It helps maintain the user's interest and keeps them engaged. Novelty is formulated based on its popularity:

$$\text{Novelty@N} = \frac{1}{|\mathcal{U}| \times N} \sum_{\hat{v} \in \mathcal{R}_u} \frac{|\mathbf{Y}_{u\hat{v}=1}|}{|\mathcal{U}|} \tag{5.14}$$

which is the fraction of users that have interacted with the item in the training set. A novelty score of zero means that the item is popular. A high novelty score means that the item rarely been interacted with.

Serendipty@N

Serendipity refers to recommendations that are unexpected, but pleasantly surprising [52]. These items are both novel and relevant to the user. Unexpectedness is measured using the cosine distance [84]:

$$\text{Serendipty@N} = 1 - \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{U}||\mathcal{S}^u|} \sum_{v \in \mathcal{S}^u} \sum_{\hat{v} \in \mathcal{R}_u} \frac{\text{cosine}(v, \hat{v})}{N} \tag{5.15}$$

A value of zero means only popular and expected relevant recommendations were made by the model. Two means that the predicted recommendations are completely unexpected but relevant.

5.3.4 Popularity Bias Metrics

Recommendation datasets suffer from inherent popularity bias. Hence, evaluating the entire dataset across all users might not provide comprehensive

insights. As a solution, we categorize users into two groups based on their preference for popular or less popular items.

Popular and Niche Items

First, we divide all the items into two groups: the popular set $\mathcal{V}_{\text{pop}}$ and the niche set $\mathcal{V}_{\text{niche}}$. Given the power-law frequency distribution of items, see Figure 3.1; we distinguish the short head of the distribution from the long tail of items. Let $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ be a threshold value. We select the top $|\mathcal{V}| \times \alpha$ items from the power-law distribution as our popular set $\mathcal{V}_{\text{pop}}$. The remaining items belong to the niche set of items $\mathcal{V}_{\text{niche}}$. We select the items in $\mathcal{V}_{\text{pop}}$ from left to right, starting from the short head. Hence, we select the items ranked by their popularity. For instance, if $\alpha = 0.2$, we take the top 20% of the items ranked according to their popularity and classify them as popular. The remaining 80% of items would belong to the niche set of items.

Figure 5.1 illustrates the power law distributions of each dataset, accompanied by an exemplified threshold value. The green area would be the popular items, and the red area corresponds to the less popular items.

Popular and Niche User Preferences

Next, we categorize users into two groups based on their preferences. The objective of sequential recommendation is to predict the next item that a user would prefer. The next item may belong to the popular or niche set of items. Thus, we determine a user's preference by observing the last item $v_n \in \mathcal{S}^u$ in their sequence of preferences. If $v_n \in \mathcal{V}_{\text{pop}}$ than the user u prefers a popular item. Conversely, if $v_n \in \mathcal{V}_{\text{niche}}$, then the user u prefers a less popular item.

Since we now have two sets of users, we calculate the accuracy metrics on the two separate sets. Specifically, we calculate the Recall and NDCG scores separately for users with popular and niche preferences. A practical recommendation model should cater to both preferences, ensuring high performance for both categories [85], [86].

It is essential to highlight that these metrics vary with changing α threshold values. If a recommendation model predominantly excels at lower threshold values, it indicates an inherent bias towards popular items.

5.3.5 Change in User Preference

Section 3.2 found that *users tend to prefer more niche items over time*. Applying a conventional accuracy metric to all items within a sequence fails to capture this nuanced shift in preference. In short sequences, popularity bias is more prominent, as these sequences often comprise predominantly popular items. Consequently, a model heavily biased toward popular items may perform well in such scenarios. However, its performance diminishes when deploying the same model in an offline setting with considerably longer user sequences.

We took a sliding window approach to evaluate each sequence $\mathcal{S}^u$. Let the length of the sliding window be N=10, then we took a subset of items $\mathcal{S}^u_{1:N} = \{v_1, \ldots, v_N\}$. We passed $\mathcal{S}^u_{1:N}$ into the three sequential models to predict the item v_{N+1}. Then we slide the window to get $\mathcal{S}^u_{N:N+10}$ and predict v_{N+10+1}. We iteratively perform this process to obtain the NDCG@10 for each window over all the users.

5.3.6 Models

BiCoRec addresses popularity bias by focusing on users who prefer less popular items. Often these users are overlooked by traditional models.

Naive

Naive models like RandomRec and 0.2-greedy may avoid popularity bias, but their random selection approach often overlooks relevant items entirely. PopRec, by design, represents the most extreme form of popularity bias.

CF-based

CF-based models face challenges with data sparsity. Items with few interactions often develop poor representations. As a result, only popular items are accurately clustered in the embedding spaces.

Context-aware

Context-aware models aim to mitigate sparsity by incorporating contextual information. Context allows them to recommend niche items despite their limited interactions. However, these models often rely on similarity measures to make recommendations. Recommendations done through item similarity

struggle with finding relevant items since users interact with items due to preference and not similarity.

Sequential recommendation

Sequential recommendation models learn a user's preference profile to identify relevant items. However, sequences can be sparse, making it difficult to find common items between any two user sequences. While context-aware sequential models can help mitigate some of this sparsity, they may not fully address the challenge. Context-aware models are dependent on the availability and quality of the contextual data. If the context is insufficient or not perfectly aligned with user preferences, the model might struggle to recommend relevant items.

BiCoRec

BiCoRec explicitly incorporates contextual information about item popularity. Our model employs co-attention mechanisms to simultaneously consider both item popularity and user preferences. Additionally, cross-pseudo supervision is utilized to align past and future user preference profiles. This alignment is done due to the assumption that user preferences shift from popular to niche items over time.

Fairness

BiCoRec is designed to promote more equitable recommendations across different user groups. Our model mitigates the dominance of popular items by improving the recommendation accuracy for users with a preference for less popular items. BiCoRec helps ensure that both popular and niche item preferences are represented fairly, offering a more balanced experience for diverse users.

5.4 Implementation

The complete code for BiCoRec and the baselines used is available on GitHub. [1] Each model consists of a configuration file which has the settings used in this experiment.

[1] https://github.com/pxpana/BiCoRec

TABLE 5.3: Hyperparameters selected after applying exhaustive search using hyperopt

Dataset	Movies	Fashion	Games	Music
Parameters				
number of layers	2	2	2	2
number of heads	1	1	1	1
unsupervised loss weight	0.3	0.1	0.4	0.3
co-attention temperature	1	0.8	1	1
layer norm epsilon	1e-12	1e-6	1e-12	1e-12
mask ratio	0.6	0.4	0.3	0.7
gradient clipping	5	5	5	10
weight decay	0.001	0.0001	0.0001	0.001
hidden size	64	64	64	128
hidden dropout	0.5	0.2	0.6	0.3
attention dropout	0.2	0.4	0.2	0.4

5.4.1 Libraries

We used the popular open-source recommendation library RecBole [87] to implement each baseline. RecBole provides a unified framework for developing and reproducing recommendation algorithms for research purposes. We used HuggingFace to extract vector representations from multimodal data [63]. HuggingFace provides pre-trained transformer-based models through its public API. Hyperopt is used for hyperparameter optimization [88]. Table A.1 provides a list of additional libraries used as well as their versions.

5.4.2 Hardware

The models are implemented using the Pytorch library. Pytorch is focused on array expressions and is usually faster than other frameworks [89]. All the models are trained from scratch without any pre-training [14]. We used a single NVIDIA RTX A4000 16GB GPU with an Intel(R) Xeon(R) Gold 5315Y CPU @ 3.20GHz for our computations. The CPU has eight cores and 16 threads.

5.4.3 Protocol

Following previous research, we applied the *leave-one-out* strategy to recommend the next item [7], [14]. For each user interaction sequence, we used the last item as the test data and the item before it as the validation data. The remaining items are used for training. We ranked a relevant item against the entire item collection for each user and computed the aggregate score

attained for each metric [79]. We conducted ten independent runs of the experiments and conducted paired t-tests to determine the significance of our approach over the baselines.

Each of the models presented consists of a configuration file with a random seed set to 2020. The random seed is used to initialize the random number generators for NumPy and PyTorch random modules. The seed ensures that whenever the experiment is run, the same random numbers are produced for data shuffling and weight initialization.

5.4.4 Hyperparameters

We report the results of each baseline under its optimal hyperparameter settings. We sample 10% of the training data for each dataset to use for hypreparameter optimization. Then we split the training data again into train and validation sets. The hyperparameters are searched using exhaustive grid search from HyperOpt. Tuning is performed using the validation set and performance monitoring ceases after 100 epochs with early stopping at 20 steps. We select the parameters that produced the highest NDCG@10 score.

Each model consists of their unique parameters. Table A.2 presents a list of parameters and their search space for each model. For common hyperparameters we consider the hidden dimension size chosen from {16, 32, 64, 128}, a dropout rate selected from {0, 0.1, 0.2, ..., 0.9}, ℓ_2 and regularization parameters from {0.0001, 0.001, 0.01, 0.1, 1}. All other hyperparameters and initialization strategies are followed by the authors' suggestions from the respective papers or tuned on the validation sets.

For a fair comparison of our model, we fix the number of attention layers to two with a single head, similar to SASRec. We optimize all the methods with Adam optimizer, and the learning rate is set to 0.001 [90]. The selected hyperparameters for our model are presented in Table 5.3.

We consider varying batch sizes depending on the size of the dataset to speed up computation. For each dataset we use a batch size of 256 except for the Fashion dataset. The Fashion dataset is a much smaller dataset and may need more steps to converge.

5.5 Limitations and Assumptions

5.5.1 Limitations

Below, we name a few challenges we faced during our research. We hope to address these limitations in future research.

- Due to limited compute resources, we performed hyperparameter tuning by sampling 10% of the users from the large datasets, including Movies, Games and Music datasets.
- The vector representations for tabular data were obtained by flattening the keys and values into a sentence. Then, a text, image and audio pre-trained multi-modal transformer was used to extract the vector representations.
- We only consider the auxiliary information of items, not the users. We refrained from using user's biographical data due to the sensitive data privacy constraints. We only considered the users' IDs.
- Our research was focused on sequential recommendation models. Other forms of recommendation, such as graph-based or session-based, require using datasets with graph-like structures or where users appear anonymously.

5.5.2 Assumptions

Below are a few assumptions we made in this research.

- We assume that users tend to prefer more novel and niche items over time. Hence, we evaluate our models using this principle.
- We fix some of the hyperparameters for fair comparison. These parameters are adopted by the original authors and are a significant part of the model design. Hence, we assumed these parameters are optimized as stated in their respective papers.

5.6 Ethical Considerations

A random identification number uniquely identifies all users. We only obtained the items each user interacted with so that we could learn from their preferences. The datasets do not include sensitive information beyond users' identification numbers. All auxiliary information is associated only with the items and not the users. This auxiliary information has already been made

public and is used to provide descriptive information about the item. All datasets used have already been made public for academic work. The results and implementation of this study are available on GitHub and are free to access by any member of the public for cross-referencing purposes.

5.7 Conclusion

This chapter outlined the methodology of our work. We elaborated on the libraries used for implementing the algorithms and the structure of our datasets. We also presented a few baseline models adopted from the literature, the type of results we expected to see, and several data analysis objectives. Finally, we reflected on some limitations and assumptions of our research project while stating some ethical considerations.

Chapter 6

Results and Discussion

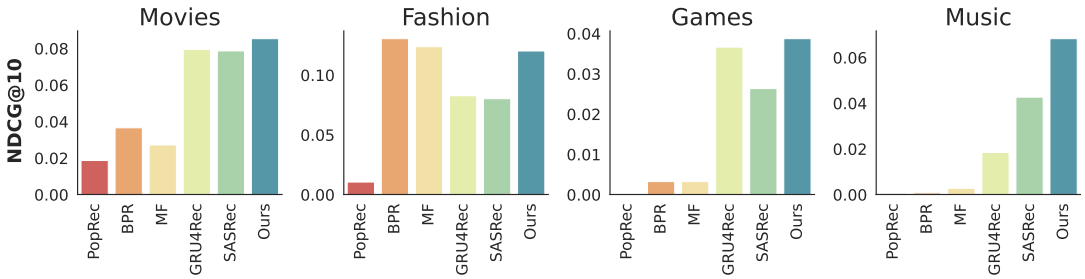
6.1 Results

TABLE 6.1: Accuracy Metrics of our approach against 12 baselines. The best-performing models are in bold, and the second-best-performing models are underlined.

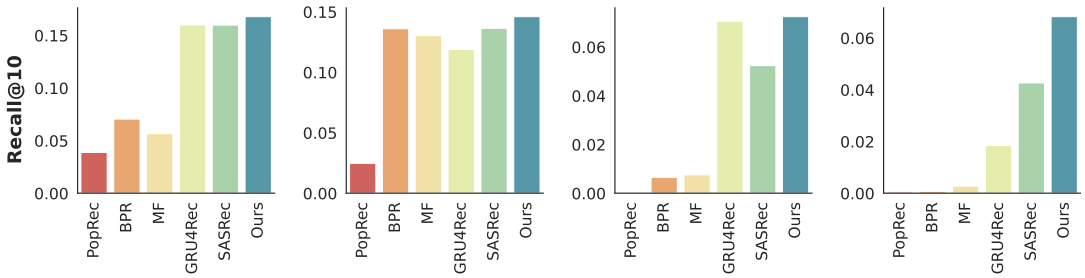
Dataset	**Movies**			**Fashion**			**Games**			**Music**		
Metrics	**Recall**	**NDCG**	**MRR**	**Recall**	**NDCG**	**MRR**	**Recall**	**NDCG**	**MRR**	**Recall**	**NDCG**	**MRR**
Naive												
RandomRec	.0037	.0017	.0011	.0008	.0003	.0002	.0003	.0002	.0001	.0000	.0000	.0000
PopRec	.0387	.0186	.0126	.0246	.0104	.0064	.0000	.0000	.0000	.0005	.0002	.0001
0.2-greedy	.0096	.0048	.0034	.0222	.0092	.0056	.0043	.0065	.0189	.0000	.0000	.0000
Collaborative Filtering												
BPR	.0704	.0365	.0263	.1359	**.1303**	**.1286**	.0065	.0032	.0022	.0007	.0003	.0002
NCF	.0310	.0147	.0099	.1130	.0693	.0557	.0303	.0157	.0113	.0007	.0006	.0005
Context-aware												
De-biased MF	.0566	.0271	.0183	.1303	.1236	.1216	.0075	.0032	.0019	.0027	.0011	.0006
FM	.0467	.0221	.0149	.1088	.1087	.1001	.0089	.0041	.0011	.0042	.0020	.0013
Sequential												
TransRec	.0211	.0098	.0065	**.1464**	.1127	.1013	.0101	.0038	.0020	.0005	.0001	.0000
Caser	.1601	.0810	.0577	.0429	.0172	.0096	.0000	.0000	.0000	.0000	.0000	.0000
GRU4Rec	.1599	.0794	.0531	.1188	.0826	.0713	.0705	.0366	.0263	.0184	.0083	.0052
BERT4Rec	.0303	.0132	.0082	.1187	.0485	.0283	.0006	.0003	.0002	.0000	.0000	.0000
SASRec	.1597	.0786	.0542	.1363	.0801	.0620	.0523	.0263	.0186	.0426	.0251	.0199
Sequential & Context												
Ours	**.1677**	**.0853**	**.0604**	.1459	.1200	.1112	**.0724**	**.0387**	**.0284**	**.0682**	**.0352**	**.0253**
Improvement	4.74%	5.30%	4.68%	-0.34%	-7.90%	-13.53%	2.70%	5.74%	7.98%	60.09%	40.23%	27.15%

6.1.1 Overall Performance

We begin by comparing the performance of BiCoRec over three accuracy-based metrics: Recall@10, NDCG@10, and MRR. Recall reflects the model's ability to identify relevant items correctly. MRR assesses the rank of the relevant item. NDCG evaluates the model's overall ranking quality. That is, a high Recall@10 value means that the model can capture a large proportion of relevant items amongst the top ten recommendations. A high MRR indicates that relevant items are placed at higher positions in the recommendation list.



(A) NDCG@10



(B) Recall@10

FIGURE 6.1: Presents the performance of the most consistent models across each dataset. Our proposed approach achieves the highest NDCG@10 and Recall@10 across all datasets, with the exception of the Fashion dataset. In short-sequence datasets like Fashion, biased models such as BPR tend to perform better due to the prevalence of popular items within the sequences.

A high NDCG@10 means the relevant item is ranked higher among the top ten recommendations.

Figure 6.1 presents the performance of the most consistent models. The full results are presented in Table 6.1. At first glance, BiCoRec scored the highest against each baseline for three datasets. However, it struggled to find the relevant item for very short sequences in the Fashion dataset. CF-based models excel in this setting, with BPR achieving the highest NDCG and MRR. This performance is expected as short sequences usually consist of primarily popular items. Hence, users' preferences are concentrated amongst a small subset of items.

Interestingly, SASRec achieved the third highest Recall@10 for the Fashion dataset. Then, SASRec achieved the second-best performance for the Music dataset, which has a much longer sequence length. However, the variance of the preference values in the Music dataset is the lowest compared to all other datasets. That is, users have a strong preference for popular items. Moreover, the distribution of the Music dataset is highly skewed. Similar to the Fashion dataset, users' preferences are concentrated amongst a small subset of items, despite the long sequence length. Consequently, SASRec proves to be particularly effective in scenarios with highly skewed datasets, resembling the performance characteristics of CF-based models.

BiCoRec produced 5.30%, 5.74% and 40.23% NDCG@10 improvements over the best-performing baseline on the Movies, Games and Music datasets, respectively. All three datasets have an average sequence length higher than 100. This performance shows that our model can learn each user's changing preferences. GRU4Rec displayed competitive performance on the Movies and Games datasets. The learning-to-rank component in GRU4Rec demonstrates robust retrieval capabilities for longer sequences. However, GRU4Rec reveals its limitations on datasets with an exceptionally large number of items, as observed in the Music dataset with a user-item ratio of -2.78. A similar pattern is observed with another ranking-based model, BPR, which is based on pairwise ranking and achieves an NDCG@10 of 0.0003. The suboptimal performance of ranking-based models becomes evident in scenarios with a large item set.

In contrast, BiCoRec incorporates the context of previous actions and distinguishes items based on their semantic relatedness. By observing each item's

latent description, our approach excels on datasets characterized by long sequences and a substantial number of items. The multi-modal auxiliary information can represent items beyond their occurrences. Hence, this strategy is capable of distinguishing items within a large corpus despite the highly skewed nature of the Music dataset.

BERT4Rec performs the worst among all the sequential recommendation models, Table 6.1. Presumably, the bidirectional structure of BERT4Rec is trained on past and future items within the user's sequence. A unidirectional model like SASRec is only trained on past values. The poor performance of the bidirectional transformer proves that a user does not interact with an item based on future consumption patterns. Instead, we should learn from past actions to predict future actions. The unidirectional transformer architecture is far superior in modelling the chronological order of items. Hence, it justifies our adoption of SASRec within BiCoRec.

On average, sequential models performed better than models that use the user-item interaction matrix. Consequently, temporal data is more effective than interaction matrices. The context-aware models, such as the debiased MF, perform better than CF recommenders on the Music and Games dataset. Context-aware models learn the semantic relatedness between the items. Hence, they can distinguish between items through their latent descriptions. Our approach uses both sequences and context data to achieve superior performance. Specifically, we achieved an average of 6.57% improvement on MRR. BiCoRec ranks the relevant item about 7% higher than current state-of-the-art models.

In summary, BiCoRec demonstrates strong performance across key accuracy metrics—Recall@10, NDCG@10, and MRR. However, BiCoRec faces challenges with very short sequences. SASRec also shows notable performance, especially with long sequences and skewed item distributions. Learning to rank models struggle with large item sets, while BERT4Rec's bidirectional approach proves less effective than unidirectional models like SASRec. Sequential models generally outperform interaction matrix-based models. Overall, BiCoRec's integration of past actions and semantic information enables it to excel in handling long sequences and large item sets.

TABLE 6.2: Accuracy metrics over the users who prefer popular and less popular items, respectively. The best-performing models are in bold, and the second-best-performing models are underlined.

Dataset / **Metrics**	RandomRec	PopRec	ϵ-greedy	BPR	NCF	De-biased MF	FM	TransRec	GRU4Rec	SASRec	**BiCoRec**	Improvement
Movies												
Popular												
Recall@10	.0050	.0135	.0081	.0122	.0086	.0008	.0013	.0008	**.0106**	.0078	.0073	-16.03 %
NDCG@10	.0023	**.0075**	.0052	.0059	.0037	.0004	.0006	.0004	.0050	.0036	.0067	-10.67 %
less popular												
Recall@10	.0020	.0028	.0028	.0052	.0035	.0065	.0088	.0005	.0044	.0017	**.0098**	11.36 %
NDCG@10	.0007	.0011	.0010	.0029	.0013	.0077	.0094	.0002	.0075	.0053	**.0102**	8.51 %
Fashion												
Popular												
Recall@10	.0006	.0394	.0376	.0480	.0417	.0434	.0122	.0457	.0406	**.0485**	.0412	-15.05 %
NDCG@10	.0003	.0129	.0123	**.0439**	.0247	.0353	.0062	.0281	.0231	.0402	.0270	-38.50 %
less popular												
Recall@10	.0010	.0000	.0005	.0005	.0043	.0075	.0046	.0000	.0000	.0010	**.0098**	30.67 %
NDCG@10	.0007	.0000	.0002	.0002	.0010	.0039	.0019	.0000	.0000	.0003	**.0047**	20.51 %
Games												
Popular												
Recall@10	.0002	.0000	.0000	.0004	.0085	.0008	.0015	.0043	.0052	.0056	**.0106**	11.47 %
NDCG@10	.0000	.0000	.0000	.0003	.0034	.0004	.0007	.0021	.0030	.0024	**.0057**	67.65 %
less popular												
Recall@10	.0000	.0000	.0000	.0000	.0000	.0009	.0012	.0055	.0005	.0000	**.0089**	61.81 %
NDCG@10	.0000	.0000	.0000	.0000	.0000	.0002	.0009	.0017	.0003	.0000	**.0021**	23.53 %
Music												
Popular												
Recall@10	.0000	.0000	.0000	.0000	.0000	.0005	.0011	.0000	.0056	.0083	**.0092**	10.84 %
NDCG@10	.0000	.0000	.0000	.0000	.0000	.0002	.0005	.0000	.0026	.0039	**.0048**	23.07 %
less popular												
Recall@10	.0000	.0000	.0000	.0000	.0000	.0008	.0016	.0000	.0004	.0000	**.0024**	50.00 %
NDCG@10	.0000	.0000	.0000	.0000	.0000	.0002	.0004	.0000	.0003	.0000	**.0005**	25.00 %

6.1.2 Popularity Bias

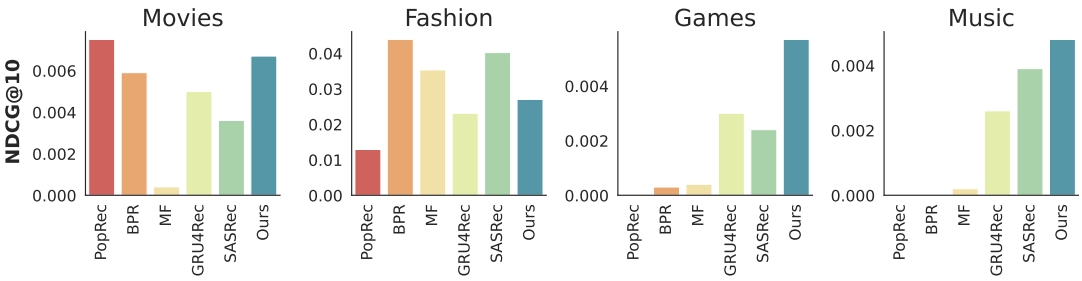
Recommendation datasets suffer from inherent popularity bias. Hence, evaluating the entire dataset across all users might not provide comprehensive insights. As a solution, we categorize users into two groups based on their preference for popular or less popular items. We analyze the last item in the user's sequence to determine a user's preference. Table 6.2 provides a comprehensive overview of the results achieved over the two sets of user preferences.

SASRec scored a Recall@10 of 0.0485 for users who prefer popular items and performed less effectively for users inclined towards less popular items with a score of 0.003. A similar trend holds for CF-based models. BPR achieved a Recall@10 of 0.0480 for users favouring popular items and 0.0005 for users favouring less popular items. This trend is observed across all models. Users' performance in the popular set significantly outweighs those in the less popular set of items. Hence, the primary contribution to the overall Recall and NDCG score is derived from recommending popular items. Consequently, these models optimize themselves over one group of users and perpetuate the popularity bias.

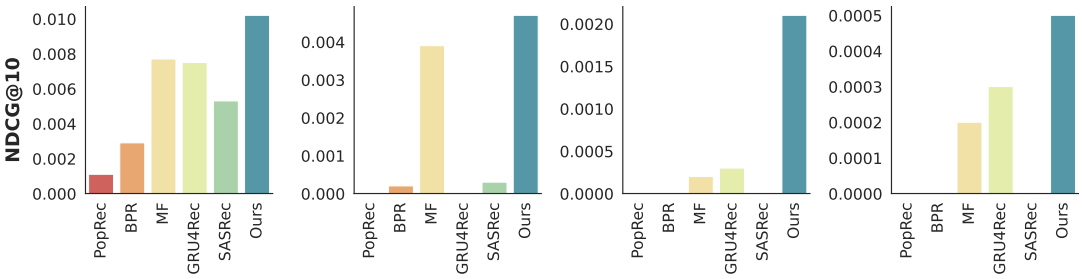
BiCoRec aimed to mitigate this effect by improving the recommendation performance of users who preferred less popular items. For these users, the model achieved 8.51%, 20.51%, 23.53% and 25% improvement on NDCG@10 over the highest-performing baseline for the Movies, Fashion, Games, and Music datasets, respectively. The parallel co-attention mechanism learns the bias inherent within the user's preferences. Hence, the model can identify users who prefer more niche items with fewer histories.

Considering the Movies dataset in Table 6.1, PopRec achieved an overall Recall@10 of 0.0387. That is, around 4% of users have interacted with the top 10 most popular items. These users prefer popular items. Hence, a naive popularity-based recommendation model can cater for their preferences. Achieving a higher Recall@10 than PopRec would require a model to recommend relevant items to users who do not have a fixed preference towards popular items. ϵ-greedy algorithm with $\epsilon = 0.2$ lower Recall@10 of 0.0097. The greedy algorithm tries to recommend popular items 80% of the time while randomly recommending less popular items 20% of the time.

Table 6.2 shows that ϵ-greedy attempts to balance the exploration and exploitation strategy. ϵ represents the likelihood of selecting an item from the



(A) NDCG@10 for users who prefer popular items



(B) NDCG@10 for users who prefer less popular items

FIGURE 6.2: Presents the performance over two sets of user preferences. BiCoRec achieves the highest NDCG@10 for less popular items. However, our model performs suboptimally over the Movies and Fashion dataset. In contrast, SASRec demonstrates nearly negligible results for less popular items.

less popular set of items. Since $\epsilon = 0.2$, we always select more popular items first. As a result, ϵ-greedy performs better for the popular set than the less popular set. Nevertheless, all baseline models display a similar pattern, performing significantly better for the popular set of users than the less popular one. Hence, they attempt to balance their exploitative and exploration strategy while optimizing for accuracy.

For instance, in Table 6.2 of the Fashion dataset, SASRec achieved an NDCG@10 of 0.0402 for users who prefer popular items and 0.0003 for less popular items. Which is a 99% reduction in performance. BiCoRec achieved an NDCG@10 of 0.0270 for the popular set and 0.0047 for the less popular set, which is an 83% reduction in performance. Our explorative strategy is superior. Note that our model performed worse than SASRec on the Fashion dataset. Therefore, BiCoRec was inferior in this regard, but it still had better exploitation and explorative strategy on a suboptimal dataset.

The Games and Music datasets are characterized by many items. The large item set leads to many fractional performance values in Table 6.2. Consequently, only a select few models exhibit reasonable performance across both

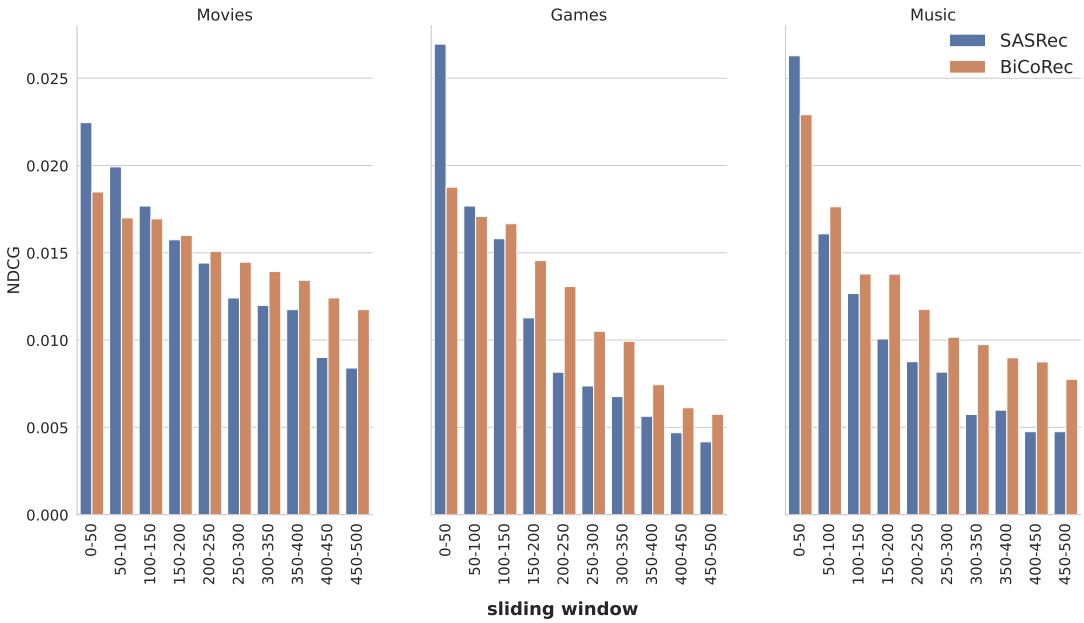


FIGURE 6.3: Depicts the performance of SASRec and BiCoRec along the users' sequences through a sliding window of size 50. BiCoRec outperforms SASRec as the window slides across the sequence of items from oldest to recent.

sets of preferences. Context-aware and sequential models demonstrate performance with precision up to four decimal places, whereas the remaining models fall short.

BiCoRec was particularly superior for users who preferred less popular items. The results show a 26% average improvement in NDCG@10 over state-of-the-art baselines across all datasets. Our bias mitigation technique improved the ranking of items with fewer histories.

To summarize, higher ranking scores are obtained for users favoring popular items, but much lower scores for those inclined toward less popular items. This disparity indicates that much of the overall performance is driven by recommending popular items. In contrast, BiCoRec aimed to mitigate this effect and showed marked improvements for users preferring less popular items. Its parallel co-attention mechanism allowed BiCoRec to learn the biases in user preferences and effectively recommend niche items. BiCoRec demonstrated superior exploration and exploitation strategies overall. Particularly in datasets with large item sets like Games and Music dataset.

6.1.3 Change in Preference

Section 3.2 found that *users tend to prefer niche items over time*. Users exhibit a higher preference for popular items earlier in their consumption history; this inclination diminishes rapidly as their preference shifts towards less popular items. Such a dynamic further explains the bias in sequential recommendation models. These models tend to be trained predominantly on popular items initially and consequently struggle to recommend niche items due to their relatively limited exposure.

Figure 6.3 depicts the performance of SASRec and BiCoRec. Specifically, we took a sliding window approach for each sequence $\mathcal{S}^u$ to evaluate the sequence. Let the length of the sliding window be N=50, then we took a subset of items $\mathcal{S}^u_{1:N} = \{v_1, \ldots, v_N\}$. We predict the next item v_{N+1} given $\mathcal{S}^u_{1:N}$. Then we slide the window to get $\mathcal{S}^u_{N:N+50}$ and predict v_{N+50+1}, and so on. We iteratively perform this process to obtain the NDCG@10 for each window over all the users. A non-overlapping sliding window ensures that each item in the sequence is included in only one window. This process prevents duplication and ensures that each item contributes uniquely to evaluating the model's performance.

For SASRec, Figure 6.3 depicts a diminishing NDCG@10 as we slide the window across the sequence. SASRec struggles to adapt to changing preferences over time. The highest NDCG@10 is achieved within the first 50 items in the sequence. This is when users are starting to engage with the system. Hence, they primarily interact with more popular items. However, as we slide the window across to the next 50 items, the NDCG score decreases gradually. This is when the users select more niche items that cater to their specific preferences. These items are less popular, so SASRec struggles to recommend them.

BiCoRec also depicts a diminishing NDCG@10 as we slide the window across the sequence. However, we initially obtained a lower NDCG@10 than SASRec around the first 50 to 150 items. As the sequence becomes longer, the overall NDCG@10 obtained is higher than that of SASRec. Our approach can recommend niche items at a better rate than SASRec. Hence, BiCoRec can adapt to changing preferences.

This study utilized a sliding window of size 50. The window size is arbitrary and other sizes would still yield similar effects. A window of 50 was chosen to better visualize the sharp decline in performance. A smaller sliding

window merely postpones this gradual decrease.

Briefly, the sliding window approach was used to evaluate users with changing preferences. For both SASRec and BiCoRec, the NDCG@10 scores were highest within the first 50 items of a sequence. However, as the window moved further into the sequence, the NDCG scores gradually decreased. BiCoRec outperforms SASRec as the window gradually slides across the sequence of items. Hence, BiCoRec can adapt to users' changing preferences and recommend niche items overtime.

6.1.4 Ablation Study

TABLE 6.3: The performance of BiCoRec after removing each proposed component.

Dataset	**Movies**		**Fashion**	
Metrics	**Recall**	**NDCG**	**Recall**	**NDCG**
BiCoRec	**.1677**	**.0853**	**.1459**	**.1200**
w/o cross-pseudo	.1350	.0535	.0982	.0813
w/o auxiliary information	.1529	.0611	.1170	.0838
w/o co-attention layer	.1498	.0599	.1251	.1112
w/o user embedding	.1658	.0844	.1438	.1195

We performed an ablation study of BiCoRec over two datasets. The critical difference between the Fashion and Movies dataset is that they consists of short and long sequences, respectively. We made four architectural enhancements to the SASRec base model. Table 6.3 presents the results of the ablation study. We tested how removing each proposed component affects the model's overall performance.

Cross-Pseudo Supervision

Cross-pseudo supervision reduces the sparsity in user sequences by learning from padded items. Specifically, we pad each sequence and learn from the padded tokens by assigning pseudo labels. Removing cross-pseudo supervision in Table 6.3 led to a 37% and 32% reduction in the NDCG@10 score for the Movies and Fashion datasets, respectively. This is the highest drop in performance and highlights the significance of learning from padded items.

Cross-pseudo supervision forces the model to generate consistent representations for both past and future items. The regularization effect helps the

model learn the shift in preferences from popular to niche items. When cross-pseudo supervision is removed, the representations are learned by predicting the next item given the previous items. Hence, the future items are ignored resulting in weaker generalization to evolving preferences.

Multi-modal Auxiliary Information

Multi-modal auxiliary information helps to improve the latent description of each item. Thus, the model can learn the semantic relatedness between the items. Removing auxiliary information in Table 6.3 led to 28% and 30% reduction in the NDCG@10 score for the Movies and Fashion datasets, respectively. The Fashion dataset consists of four times the number of items than the Movies dataset. Therefore, incorporating auxiliary information generates more distinct item embeddings for datasets with many items.

Auxiliary information helps the model differentiate items by providing additional features that describe their properties. Without this information, the model relies solely on user interaction data. Since the Movies and Fashion interaction data are highly sparse some items have less distinct embeddings.

Co-Attention Layer

The co-attention layer injects the context of popularity within the user's sequence. The context informs the model about the inherent bias present in each user's sequence. Removing the co-attention layer in Table 6.3 led to a 30% and 7% drop in the NDCG@10 score for the Movies and Fashion dataset, respectively. The Movies dataset has a much longer average sequence length than the Fashion dataset. Hence, the preferences of the users are changing towards more niche items. The co-attention layer can capture this shift in preferences for longer sequences.

The co-attention layer may be less effective in cases where user sequences are very short, such as in session-based datasets. This is because users often interact primarily with popular items and tend to leave before engaging with more niche items.

User Embeddings

Finally, removing the user embeddings in Table 6.3 does not lead to a substantial, less than 1%, drop in performance for both datasets. The user embeddings are ID-based. They use a unique identification number to map each

user to an embedding vector. The process is similar to the ID-based item embeddings. Hence, not enough contextual information can be derived from the user embeddings.

Enhancing user representations with auxiliary information from user profiles could potentially improve performance. However, this approach is often avoided due to concerns about data sensitivity and the unreliability of such data.

To conclude, cross-pseudo supervision is the most significant component, as its removal led to the largest drop in performance. Cross-pseudo supervision highlights the importance of learning from both past and future items. The multi-modal auxiliary information also played a critical role, with its absence causing a 28% and 30% drop in NDCG@10 for Movies and Fashion, respectively. Auxiliary information helps enrich the representations of the items within the embedding space. The co-attention layer was particularly effective for longer sequences though it had less impact on short user sequences. Finally, removing user embeddings resulted in less than a 1% decrease in performance, suggesting that user embeddings contribute minimally to the model's overall effectiveness.

6.1.5 BiCoRec vs SOTA

We conduct a comprehensive analysis of BiCoRec in comparison to state-of-the-art sequential recommendation models. First, we present the results of a statistical significance test evaluating BiCoRec against each baseline. Next, we plot precision-recall curves to assess the retrieval capabilities of each model. Finally, we examine three fairness metrics to evaluate the quality and equity of the recommendations.

Statistical Significance

Table 6.4 reports the statistical significance of BiCoRec using the NDCG@10 score. We reject the null hypothesis at a p-value of 0.05, that is, the observed performance of BiCoRec over the baseline model could not have occurred by chance. Table 6.4 depicts the mean and standard deviation of NDCG@10 for each model. Below each state-of-the-art model, the table also displays the statistical significance of the comparison with BiCoRec.

TABLE 6.4: Paired t-test of the NDCG@10 scores achieved by the state-of-the-art sequential recommendation models over BiCoRec's achieved results.

Dataset		TransRec	Caser	GRU4Rec	BERT4Rec	SASRec	**BiCoRec**
Fashion	mean	.1127	.0172	.0826	.0485	.0801	.1200
	Standard deviation	.0098	.0131	.0090	.0051	.0083	.0002
	degrees of freedom						16
	t statistic	2.600	11.64	10.66	42.90	11.80	
	p-value	.0193	1.6e-09	1.1e-08	6.0e-18	2.6e-09	
Movies	mean	.0098	.0810	.0794	.0132	.0786	.0853
	Standard deviation	.0004	.0259	.0042	.0029	.0067	.0113
	degrees of freedom						16
	t statistic	19.06	.5032	1.552	17.57	2.065	
	p-value	2.0e-12	.6217	.1400	7.0e-12	.0555	
Games	mean	.0038	.0000	.0366	.0003	.0263	.0387
	Standard deviation	.0010	1.5e-05	.0023	.0001	.0088	.0051
	degrees of freedom						16
	t statistic	18.99	21.44	1.107	21.40	11.76	
	p-value	2.1e-12	3.3e-13	.2846	3.4e-13	2.7e-09	
Music	mean	.0001	.0000	.0083	.0000	.0251	.0352
	Standard deviation	2.1e-12	8.9e-06	.0005	2.7e-05	.0077	.0042
	degrees of freedom						16
	t statistic	23.39	23.44	17.86	23.43	3.266	
	p-value	8.5e-14	8.1e-14	5.5e-12	8.2e-14	.0048	

BiCoRec achieves statistically significant results across all datasets except in four cases. The first three are; Caser, GRU4Rec and SASRec with p-values of 0.6217, 0.1400 and 0.0555, respectively, for the Movies dataset. Lastly, GRU4Rec has a p-value of 0.2846 for the Games dataset. These high p-values stem from BiCoRec's marginal NDCG@10 improvement over these models.

Both the Movies and Games datasets have the longest average sequence length. Long sequences make it challenging to accurately rank a relevant item that matches a long list of user preferences. Furthermore, standard deviations of .0259, .0042, and .0067 for Caser, GRU4Rec, and SASRec indicate that these models sometimes match BiCoRec's .0853 mean NDCG@10 performance. On average, BiCoRec surpasses baselines, suggesting a real effect, though larger samples may yield more conclusive results. Nonetheless, BiCoRec consistently delivers statistically significant results for the Fashion and Music datasets, excelling in shorter and moderately long sequences.

Precision-Recall Curves

Figure 6.4 plots the precision-recall curve of each state-of-the-art baseline model and BiCoRec. Recall records how well the model can find the relevant item. Precision reflects on how many of the predicted items are relevant

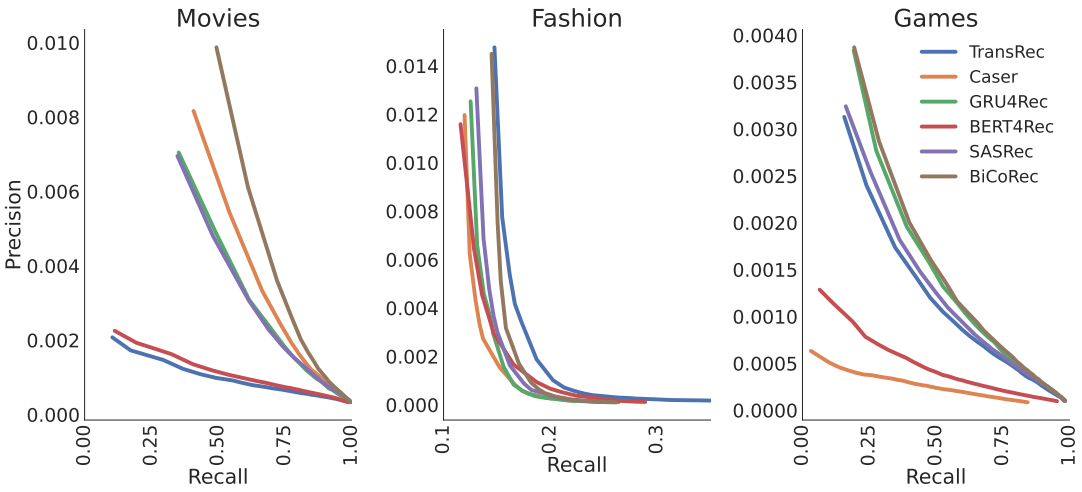


FIGURE 6.4: Plots the precision-recall curve of the state of the sequential recommendation models. BiCoRec depicts the steepest slope for long sequence datasets. It only falls short of one model in the Fashion dataset that has very short sequences.

items. The precision-recall curve focuses on the trade-off between precision and recall. This curve is useful in the recommendation setting since the relevant item is much rarer when ranked against the entire collection. In particular, the precision-recall curve evaluates the models' rare event detection capability.

At first glance, the precision-recall curves portray an inverse relationship. As recall increases the precision scores decrease. This trend reveals a moderately good classifier. A more effective setting would be a curve that maintains a high precision score even as recall decreases.

TransRec shows an almost flat curve for the Movies and Games datasets but has the steepest slope for the Fashion dataset compared to all other models. The Fashion dataset is more skewed toward popular items. TransRec demonstrates strong retrieval capabilities on highly biased datasets but struggles with datasets featuring longer sequences and evolving user preferences.

BiCoRec depicts the steepest slope for the Movies and Games dataset and second to TransRec for the Fashion dataset. Hence, BiCoRec can retrieve relevant items under biased and evolving preferences better than the current state-of-the-art sequential recommendation models.

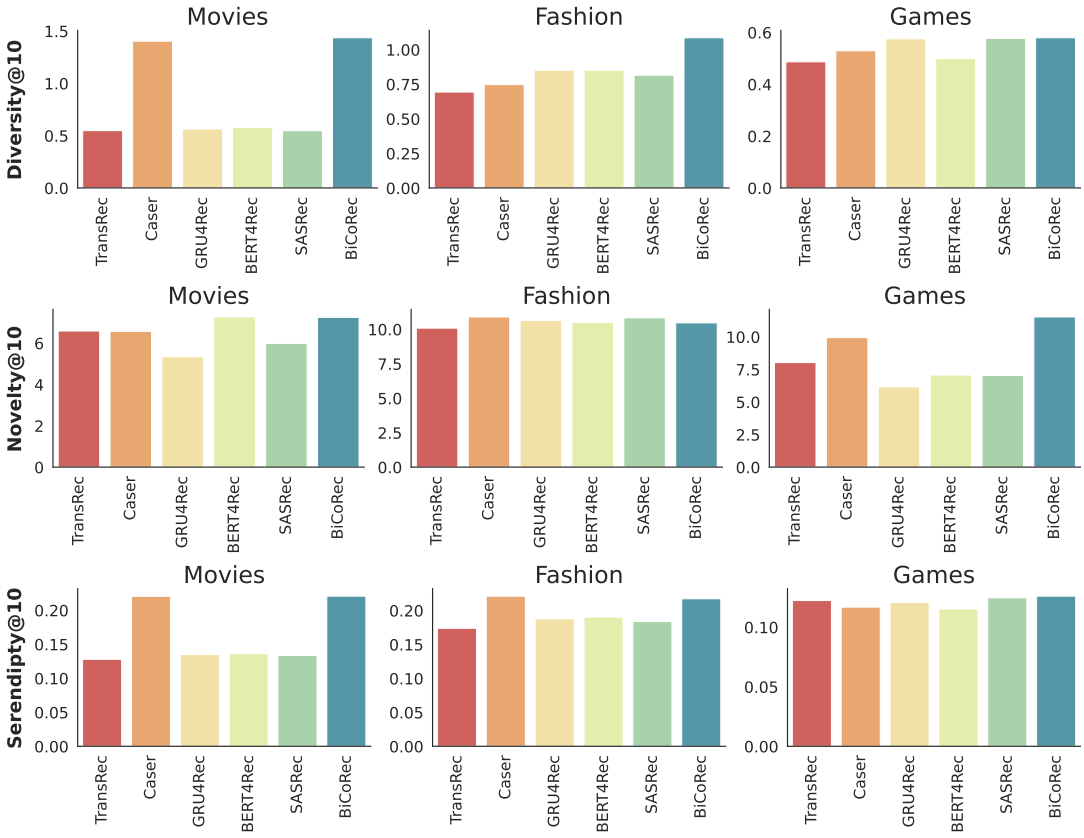


FIGURE 6.5: Plots the diversity, novelty and serendipity scores achieved by SOTA models and BiCoRec on three datasets.

Fairness Metrics

This section describes the performance of each model on non-accuracy-based metrics. Figure 6.5 presents the diversity, novelty and serendipity scores obtained by each state-of-the-art model.

Diversity measures how varied the recommendations are for a user. Novelty refers to how unfamiliar the recommended items are to the user. Serendipity refers to recommending items that are not only novel but also surprisingly relevant.

BiCoRec achieved the highest diversity@10 score for the Fashion dataset, despite ranking second to TransRec in accuracy metrics. TransRec's low diversity and novelty scores indicate that it primarily recommends popular items to boost its accuracy on biased datasets. Although BiCoRec's novelty@10 score is slightly lower than SASRec's, its high serendipity score suggests that our approach prioritizes recommending relevant and novel items.

BiCoRec consistently produces a high diversity, novelty and serendipity score only competitive to Caser. Caser performs worse then BERT4Rec on the novelty score. Suggesting that Caser recommends popular relevant items while BiCoRec recommends more novel and relevant items as the user's preferences change towards niche items. BERT4Rec's high novelty score combined with low performance on accuracy metrics depicts an almost random approach to recommendations.

The Games dataset has a large number of items. GRU4Rec ranked as the second-best model in terms of accuracy, benefiting from its learning-to-rank mechanism. While it achieved the highest diversity@10 score it still had the lowest novelty score. The uneven performance indicates that it struggles to rank new or unseen items. Instead, GRU4Rec prefers selecting from a diverse subset of previously seen items. Similar to its performance on the Movies dataset, BiCoRec consistently outperformed other models across diversity, novelty, and serendipity metrics.

BiCoRec's strong performance on fairness metrics for the Fashion dataset highlights its superior bias mitigation strategy, particularly in short, heavily biased sequences. However, this comes at the cost of some accuracy loss. On the Movies dataset, BiCoRec consistently excels in fairness, demonstrating its advantage in handling long sequences with evolving user preferences. It delivers diverse, novel recommendations that remain relevant to the

user's interests. Additionally, BiCoRec outperforms learning-to-rank models on datasets with a large item corpus by effectively identifying relevant novel items while avoiding over-reliance on popular ones. Although the current state-of-the-art model, SASRec, also performs well in recommending novel items, BiCoRec distinguishes itself by prioritizing the relevance of these novel items to the user.

6.2 Discussion

The main objective of sequential recommendation models is to learn from the user's evolving preferences. Current state-of-the-art recommendation models fail to cater to users who change their preferences towards less popular items. Our solution addresses this challenge by implementing a bias-mitigation strategy that realigns its recommendation towards niche items.

6.2.1 Main Results

Since the frequency of items exhibits a power-law distribution, it is necessary not to resemble the inherent popularity bias within user preferences. Koren, Bell, and Volinsky [3] explicitly model the effects of popularity bias within their model. BiCoRec adopts this strategy by explicitly learning a popularity embedding for each user sequence. This approach results in a 26% improvement in performance over the best baseline model for users who prefer less popular items.

User sequences are initially designed by padding null entries with zeros. The primary learning method in sequential recommendation is to predict the next item given the previous items [7], [14]. Hence, the future items are ignored. Since users tend to prefer less popular items over time, future items contain valuable insights about the users' evolving preferences.

Conversely, these future items are often padded. Our model learns from padded items to effectively learn from evolving future preferences. This learning method contributed the most to the performance of our overall strategy.

For highly sparse datasets, two users are unlikely to have any items in common. Hence, less occurring items do not get recommended. Rendle [4] use auxiliary information to estimate the parameters of their model even under

high data sparsity. Our approach extends this concept by utilizing multimodal auxiliary information to recommend items based on their semantic relatedness. This strategy demonstrates superiority in handling sparse datasets, such as the Games and Music datasets.

Interestingly, current state-of-the-art sequential models perform substantially better for users who prefer popular items; see Section 6.1.2. The primary contribution to overall Recall and NDCG scores is derived from recommending popular items. Consequently, these models optimize themselves over one group of users and perpetuate the popularity bias.

6.2.2 Limitations and Delineations

Preference towards Popular Items

BiCoRec may not perform optimally in situations where users strongly favour popular items. Our results reflect sub-optimal performance for users with short sequences who only engage with popular items or highly skewed datasets. Our bias-mitigation strategy heavily penalizes popular items in favour of niche items. The TF-IDF popularity sequences ensure that less occurring items weigh substantially higher than frequent items. Hence, our proposed framework is limited to scenarios involving session-based interactions or situations characterized by short-term dynamics.

Future research could explore alternative metrics for measuring popularity bias beyond TF-IDF scores. Ideally, such a metric would balance the weighting of popular and niche items, ensuring that the weight discrepancies between them are not too pronounced. An ideal metric would dynamically quantify the user's preferences for popular or niche items.

User Embeddings

BiCoRec did not exhibit an impressive performance improvement when incorporating the user embedding. This work only embeds the user's identification number. No other auxiliary information about the user was fused into the model. Consequently, our strategy struggles to embed information with limited context.

Future work could investigate integrating richer user profiles that do not infringe on their privacy. Information such as social connections or online activity data can be used to find similarities between users.

Balanced Performance

BiCoRec performs best among users who prefer niche items against all baselines. However, our model still performs substantially better for users who prefer popular items. We were not able to fully balance BiCoRec's performance across the two sets of user preferences.

BiCoRec builds on SASRec as its base model. The results indicate that SASRec performs significantly better for users with a preference for popular items. Despite the bias-regularizing components introduced in BiCoRec, fully mitigating this bias remains challenging. The transformer architecture used by SASRec may not be well-suited for managing highly skewed datasets. Future work could explore alternative deep learning architectures better equipped to address this bias effectively.

Nonetheless, this study marks the initial steps towards a bias-mitigated sequential recommendation model.

Chapter 7

Conclusion

7.1 Summary

Recommendation models suffer from an inherent popularity bias. This bias results from the frequency of items following a power law distribution. Popularity bias produces a severe class imbalance between popular and less popular items. It can be argued that recommending popular items is not a concern, as these items have earned their popularity by being highly relevant to many users. However, we found that users prefer niche items over time. This research devised a **Bi**as-Mitigated **Co**ntext-Aware Sequential **Rec**ommendation Model (BiCoRec) that adapts to evolving user preferences.

Chapter 2 delves into the background of recommendation models. We begin by highlighting the limitations of collaborative filtering models and advocate for context-aware models instead. We explore how context-aware models leverage various contextual factors like bias, auxiliary information, and temporal data. Additionally, we provide insights into the current state-of-the-art and some progress made in context-aware sequential recommendation models.

Chapter 3 shows the existence of popularity bias with sequential recommendation models. We reflect on the limitations of bias and justify the need for a model that adapts to users evolving preferences.

Chapter 4 reviews some related work on incorporating auxiliary information, mitigating popularity bias, and the current learning task used for sequential recommendation models. Moreover, we detail the framework of our enhanced approach and present the final architecture of BiCoRec.

Chapter 5 outlines the methodology of our research, including the datasets used, evaluation metrics, and implementation details. We also discuss the limitations of our study and some ethical considerations.

Finally, Chapter 6 presents the outcomes of our research. We compare the performance of our proposed approach against established baselines. We analyze the efficacy of BiCoRec in accommodating user preferences for both popular and less popular items. The model's adaptability to changing preferences is assessed through a novel evaluation scheme. Lastly, we conduct an ablation study to examine the impact of individual proposed components.

7.2 Conclusion

Our model is on average 7% more capable of uncovering the most relevant items. Our bias mitigation technique improved the performance of users who prefer unpopular items by 26%. We achieved an average of 3.14% increase in the NDCG score for long user sequence datasets over current state-of-the-art models. Finally, we observed a more consistent performance as we slid the window across the user's sequence. That is, our model can recommend relevant items to users changing preferences.

The cross-pseudo supervision component contributes to the largest performance increase. This demonstrates the importance of learning from padded items. Padded items contain information about the user's future preferences. Hence, our model can adapt to changing preferences. However, our model remains sub-optimal in cases where users strongly prefer popular items and have very short sequences.

This study contributes a bias-mitigated multi-modal auxiliary information sequential recommendation model. We provide a novel technique to measure popularity bias within user sequences. Then, we incorporate a bias-mitigation strategy using a co-attention mechanism that attends to the user sequence and popularity representation simultaneously. We expand the training data by learning from padded items using cross-pseudo supervision. Lastly, we present a novel evaluation scheme for user sequences.

7.3 Future Work

The broader implications of this research are threefold. First, bias mitigation results in a model that adapts to evolving preferences. This process enables users to engage with a wider array of items. Second, our framework allows for the inclusion of items under different modalities. We can now recommend items beyond their item ID or attribute data. Lastly, learning from padded items transforms the sequential recommendation task from next-item recommendation to multi-item recommendation. Our objective is to predict a set of items, which ensures the consistency of predictions within the sequence.

We believe our research will motivate the development of a bias-mitigated context-aware user sequential model. This study specifically focused on popularity due to its ease of measurement. However, other qualitative contextual factors may change over time, such as the user's intentions. Measuring intent is considerably more challenging and requires more nuanced observation. Nonetheless, exploring changes in user intent presents an intriguing avenue for future research.

Moreover, incorporating contextual information in a dynamic environment is a complex interplay of various factors that extends beyond recommendation systems. As technology and methodologies advance, future studies may delve deeper into these complexities to enhance our understanding of contextual information in dynamic and evolving environments.

Bibliography

[1] D. Billsus, M. J. Pazzani, *et al.*, "Learning collaborative information filters.," in *Icml*, vol. 98, 1998, pp. 46–54.

[2] D. Goldberg *et al.*, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992, ISSN: 0001-0782.

[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. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 995–1000.

[5] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10, New York, NY, USA: Association for Computing Machinery, 2010, pp. 811–820, ISBN: 9781605587998. DOI: 10.1145/1772690.1772773. [Online]. Available: https://doi.org/10.1145/1772690.1772773.

[6] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17, New York, NY, USA: Association for Computing Machinery, 2017, pp. 306–310.

[7] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 197–206.

[8] M. Muthivhi, T. van Zyl, and H. Wang, "Multi-modal recommendation system with auxiliary information," in *Southern African Conference for Artificial Intelligence Research*, Springer, 2022, pp. 108–122.

[9] E. Fischer *et al.*, "Integrating keywords into bert4rec for sequential recommendation," in *KI 2020: Advances in Artificial Intelligence*, U. Schmid, F. Klügl, and D. Wolter, Eds., Cham: Springer International Publishing, 2020, pp. 275–282.

[10] K. Zhou *et al.*, "S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1893–1902.

[11] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," *CoRR*, vol. abs/1901.07555, 2019.

[12] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, ser. RecSys '08, New York, NY, USA: Association for Computing Machinery, 2008, pp. 11–18, ISBN: 9781605580937.

[13] M. Muthivhi, T. L. van Zyl, and H. Wang, "Impacts of architectural enhancements on sequential recommendation models," in *Southern African Conference for Artificial Intelligence Research*, Springer, 2023, pp. 315–330.

[14] F. Sun *et al.*, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 1441–1450.

[15] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17, New York, NY, USA: Association for Computing Machinery, 2017, pp. 42–46, ISBN: 9781450346528.

[16] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 896–911, 2012.

[17] J. Wei *et al.*, "Collaborative filtering and deep learning-based recommendation system for cold start items," *Expert Systems with Applications*, vol. 69, pp. 29–39, 2017, ISSN: 0957-4174.

[18] B. Sarwar *et al.*, "Application of dimensionality reduction in recommender system - a case study," *University of Minnesota Digital Conservancy*, 2000.

[19] Y. Yang *et al.*, "Debiased contrastive learning for sequential recommendation," in *Proceedings of the ACM Web Conference 2023*, ser. WWW

'23, New York, NY, USA: Association for Computing Machinery, 2023, pp. 1063–1073, ISBN: 9781450394161.

[20] Y. Hu *et al.*, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2117–2130, 2013.

[21] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 263–272.

[22] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004, ISSN: 1046-8188.

[23] S. Rendle *et al.*, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[24] S. Rendle, *Context-aware ranking with factorization models*. Springer, 2011.

[25] X. He *et al.*, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '16, New York, NY, USA: Association for Computing Machinery, 2016, pp. 549–558.

[26] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080.

[27] X. He *et al.*, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[28] F. Liu *et al.*, "User diverse preference modeling by multimodal attentive metric learning," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 1526–1534.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323.

[30] L. A. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, 1987.

[31] M. Bazire and P. Brézillon, "Understanding context before using it," in *Modeling and Using Context: 5th International and Interdisciplinary Conference CONTEXT 2005, Paris, France, July 5-8, 2005. Proceedings 5*, Springer, 2005, pp. 29–40.

[32] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, New York, NY, USA: Association for Computing Machinery, 2009, pp. 447–456, ISBN: 9781605584959.

[33] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, New York, NY, USA: Association for Computing Machinery, 2008, pp. 426–434.

[34] C.-Y. Wu *et al.*, "Recurrent recommender networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17, New York, NY, USA: Association for Computing Machinery, 2017, pp. 495–503.

[35] A. Zimdars, D. M. Chickering, and C. Meek, "Using temporal data for making recommendations," *CoRR*, 2013. arXiv: 1301.2320.

[36] G. Shani *et al.*, "An mdp-based recommender system.," *Journal of Machine Learning Research*, vol. 6, no. 9, 2005.

[37] S. Feng *et al.*, "Personalized ranking metric embedding for next new poi recommendation," 2015.

[38] P. Wang *et al.*, "Learning hierarchical representation model for next-basket recommendation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15, New York, NY, USA: Association for Computing Machinery, 2015, pp. 403–412, ISBN: 9781450336215.

[39] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17, New York, NY, USA: Association for Computing Machinery, 2017, pp. 161–169, ISBN: 9781450346528.

[40] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ser. WSDM '18, New York, NY, USA: Association for Computing Machinery, 2018, pp. 565–573.

[41] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[42] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[43] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[44] X. Xie *et al.*, "Contrastive learning for sequential recommendation," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 1259–1273.

[45] T. Zhang *et al.*, "Feature-level deeper self-attention network for sequential recommendation.," in *IJCAI*, 2019, pp. 4320–4326.

[46] U. Singer *et al.*, "Sequential modeling with multiple attributes for watchlist recommendation in e-commerce," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '22, New York, NY, USA: Association for Computing Machinery, 2022, pp. 937–946.

[47] D. Jannach *et al.*, "What recommenders recommend: An analysis of recommendation biases and possible countermeasures," *User Modeling and User-Adapted Interaction*, vol. 25, pp. 427–491, 2015.

[48] H. Abdollahpouri and M. Mansoury, "Multi-sided exposure bias in recommendation," *CoRR*, vol. abs/2006.15772, 2020. arXiv: 2006.15772.

[49] W. Sun *et al.*, "Debiasing the human-recommender system feedback loop in collaborative filtering," in *Companion Proceedings of The 2019 World Wide Web Conference*, ser. WWW '19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 645–651.

[50] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: How accuracy metrics have hurt recommender systems," in *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '06, New York, NY, USA: Association for Computing Machinery, 2006, pp. 1097–1101, ISBN: 1595932984.

[51] Ò. Celma and P. Cano, "From hits to niches? or how popular artists can bias music recommendation and discovery," in *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, ser. NETFLIX '08, New York, NY, USA: Association for Computing Machinery, 2008, ISBN: 9781605582658.

[52] P. Castells, N. Hurley, and S. Vargas, "Novelty and diversity in recommender systems," in *Recommender Systems Handbook*, Springer, 2021, pp. 603–646.

[53] X. Chen *et al.*, "Semi-supervised semantic segmentation with cross pseudo supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2613–2622.

[54] G. Adomavicius and J. Zhang, "Impact of data characteristics on recommender systems performance," *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 1, 2012, ISSN: 2158-656X.

[55] C. Pei *et al.*, "Personalized re-ranking for recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 3–11.

[56] J. Ma *et al.*, "Disentangled self-supervision in sequential recommenders," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 483–491.

[57] Z. Li, A. Sun, and C. Li, *Diffurec: A diffusion model for sequential recommendation*, 2023. arXiv: 2304.00686 [cs.IR].

[58] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[59] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *Computer Vision – ECCV 2016*, B. Leibe *et al.*, Eds., Cham: Springer International Publishing, 2016, pp. 451–466.

[60] S. Yang *et al.*, "Unimf: A unified framework to incorporate multimodal knowledge bases intoend-to-end task-oriented dialogue systems.," in *IJCAI*, 2021, pp. 3978–3984.

[61] V. Borisov *et al.*, "Deep neural networks and tabular data: A survey," *arXiv preprint arXiv:2110.01889*, 2021.

[62] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[63] HuggingFace, *Huggingface*, https://huggingface.co/docs/transformers/v4.21.2/en/model_doc/data2vec.

[64] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[65] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[66] A. Baevski *et al.*, "Wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, H. Larochelle *et al.*, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 12 449–12 460.

[67] A. Baevski *et al.*, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International Conference on Machine Learning*, PMLR, 2022, pp. 1298–1312.

[68] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism Quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[69] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[70] S. Kulkarni and S. F. Rodd, "Context aware recommendation systems: A review of the state of the art techniques," *Computer Science Review*, vol. 37, p. 100 255, 2020.

[71] C. Gini, "Measurement of Inequality of Incomes," *The Economic Journal*, vol. 31, no. 121, pp. 124–125, 1921, ISSN: 0013-0133.

[72] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '00, New York, NY, USA: Association for Computing Machinery, 2000, pp. 241–250, ISBN: 1581132220.

[73] M. Sun, "How does the variance of product ratings matter?" *Management Science*, vol. 58, no. 4, pp. 696–707, 2012.

[74] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[75] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 188–197.

[76] M. Moscati *et al.*, "Music4all-onion - A large-scale multi-faceted content-centric music recommendation dataset," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, M. A. Hasan and L. Xiong, Eds., ACM, 2022, pp. 4339–4343.

[77] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[78] C. M. K. John S. Breese David Heckerman, "Empirical analysis of predictive algorithms for collaborative filtering," *CoRR*, vol. abs/1301.7363, pp. 43–52, 2013. arXiv: 1301.7363.

[79] W. Krichene and S. Rendle, "On sampled metrics for item recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1748–1757.

[80] G. Karypis, "Evaluation of item-based top-n recommendation algorithms," in *Proceedings of the Tenth International Conference on Information and Knowledge Management*, ser. CIKM '01, New York, NY, USA: Association for Computing Machinery, 2001, pp. 247–254.

[81] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, vol. 23, 2011, p. 53.

[82] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[83] B. Smyth and P. McClave, "Similarity vs. diversity," in *International conference on case-based reasoning*, Springer, 2001, pp. 347–361.

[84] Y. C. Zhang *et al.*, "Auralist: Introducing serendipity into music recommendation," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 13–22.

[85] A. Bellogín, P. Castells, and I. Cantador, "Statistical biases in information retrieval metrics for recommender systems," *Information Retrieval Journal*, vol. 20, pp. 606–634, 2017.

[86] H. Abdollahpouri *et al.*, *The unfairness of popularity bias in recommendation*, 2019. arXiv: 1907.13286 [cs.IR].

[87] W. X. Zhao *et al.*, "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21, New York, NY, USA: Association for Computing Machinery, 2021, pp. 4653–4664, ISBN: 9781450384469.

[88] J. Bergstra, D. Yamins, D. D. Cox, *et al.*, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in Science Conference*, Citeseer, vol. 13, 2013, p. 20.

[89] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach *et al.*, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[90] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, Ieee, 2018, pp. 1–2.

[91] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Appendix A

Formulations

TF-IDF

The Term Frequency - Inverse Document Frequency (TF-IDF) are calculated of each item. In this setting, the term is the item, the document is the sequence of items and the corpus is all the users' sequences. The Term Frequency (TF) measures the presence of an item within a sequence.

$$\text{TF}_v = \frac{\text{number of times the item } v \text{ appears in the sequence}}{\text{total number of items in the sequence}} \tag{A.1}$$

which produces $\frac{1}{n_u}$ if the item is in the sequence since an item can only appear once within a sequence. Inverse Document Frequency (IDF) of an item reflects the proportion of sequences in the corpus that contain the item. Items unique to a small percentage of sequences (e.g., niche items) receive higher importance values than items common across all sequences.

$$\text{IDF}_v = log\left(\frac{\text{total number of sequences in the corpus}}{\text{number of sequences that contain the item } v}\right) \tag{A.2}$$

TF-IDF balances the item commonality within a sequence measured by TF with the rarity between sequences measured by IDF, such that:

$$\text{TF-IDF}_v = \text{TF}_v \times \text{IDF}_v \tag{A.3}$$

the score reflects the importance of an item for a sequence in the corpus. For each user, we obtain the TF-IDF scores of each item in the sequence and refer to the new sequences as the popularity scores $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}|\times n}$. That is, for an item v_t from the user's sequence of items $\mathcal{S}^u$ we have the TF-IDF score $\mathbf{p}_{ut}$ that describes the item's popularity in relation to other users. Hence, the vector $\mathbf{p}_u$ describes the user's preference to popular and niche items.

Dropout and Layer Normalization

$$\phi_1(\mathbf{x}) = \mathbf{x} + \text{Dropout}(\phi_2(\mathbf{x})) \tag{A.4}$$

where $\mathbf{x}$ is a vector containing all features of a sample and ϕ_2 is the self-attention SA or feed-forward network FFN function [7]. Residual connections propagate low-layer features to higher layers by residual connection [91]. Since embedding the last visited item is entangled with all previous items after several self-attention blocks, adding residual connections helps propagate the last visited item's embedding to the final layer. Layer normalization helps stabilize the neural network by normalizing the inputs across features towards zero-mean and unit variance [42]:

$$\text{LayerNorm}(\mathbf{x}) = \alpha \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{A.5}$$

where μ and σ are the mean and variance $\mathbf{x}$ while α and β are the learned scaling factors and bias terms, respectively.

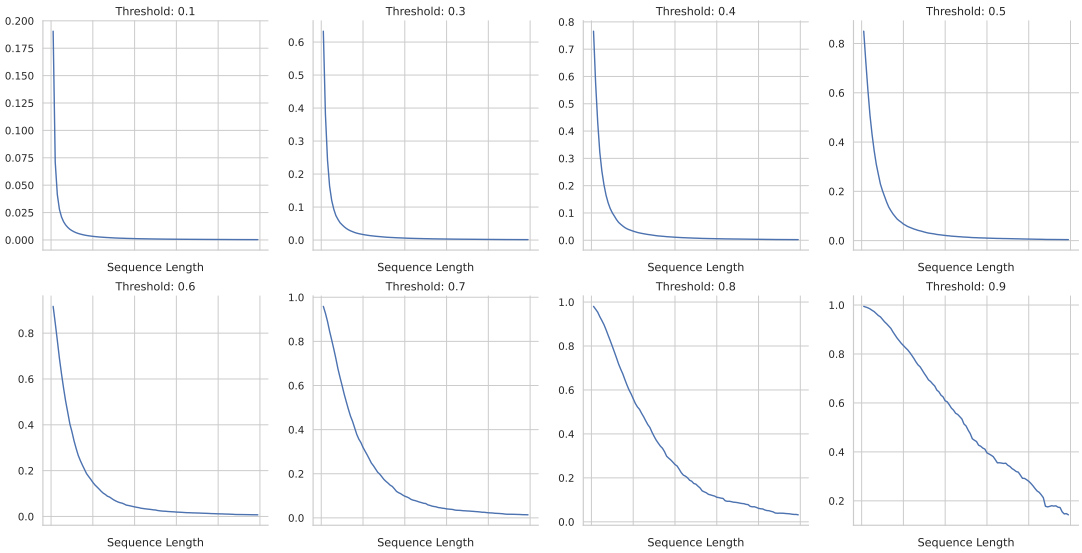


FIGURE A.1: Movies Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases.

TABLE A.1: List of libraries used during the implementation of each model

Library	Description	Version
Recbole	For developing recommendation algorithms	1.0.1
HuggingFace	Pretrained Machine Learning Models	4.44.2
Pytorch	Ecosystem of machine learning tools	1.10.0
HyperOpt	Hyperparameter optimization functionalities	0.2.7
Numpy	For multi-dimensional arrays and matrices	1.26.4
Pandas	For data manipulation and analysis	1.3.0
Scipy	For optimization and image processing	1.6.0

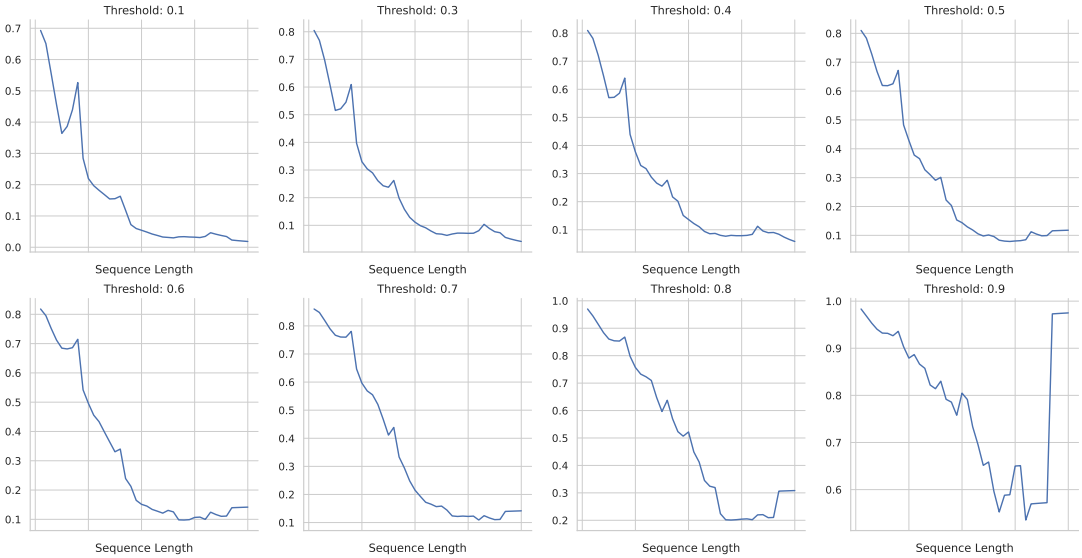


FIGURE A.2: Fashion Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases.

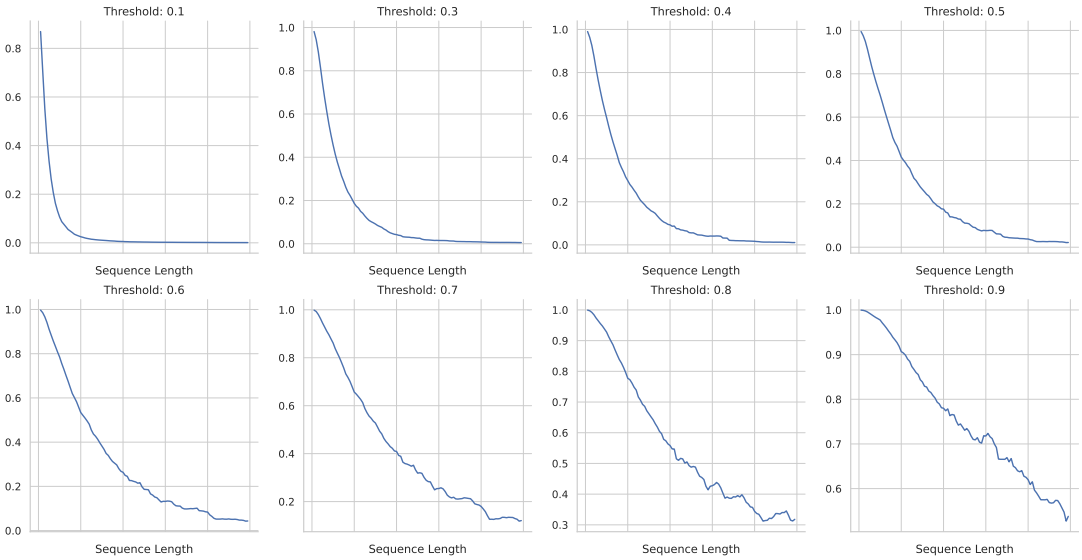


FIGURE A.3: Games Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases.

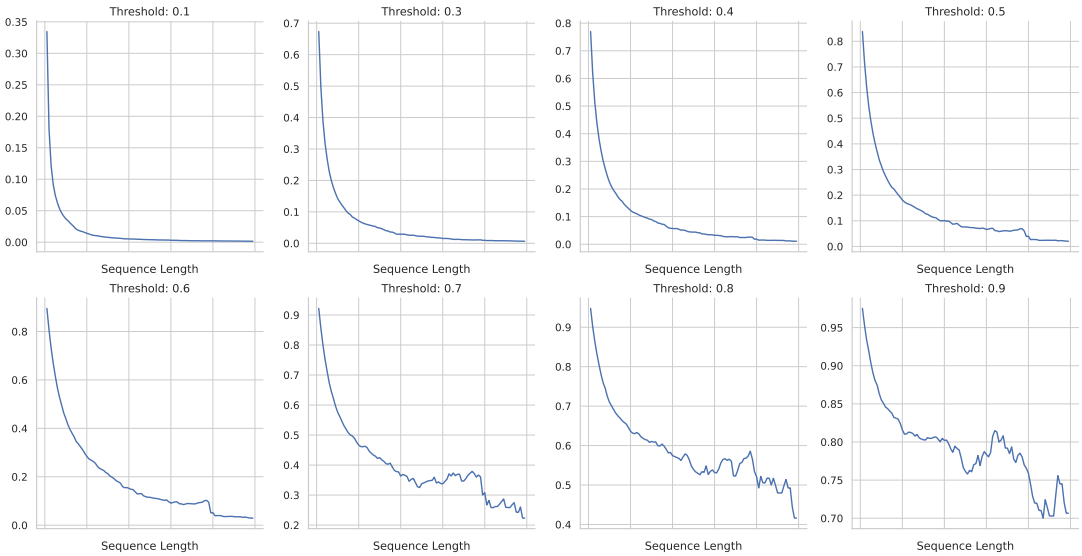


FIGURE A.4: Music Dataset: plots the ratio of popular to unpopular items, within each position in the sequence. The ratio decreases as the sequence length increases.

TABLE A.2: List of parameters and their respective search spaces for each model during hyperparameter optimization

Model	Parameter	Search Space
ϵ-greedy	Epsilon	0.1-0.8
	Threshold	0.1-0.4
BPR	Embedding size	16, 32, 64, 128
NCF	Embedding size	32, 64, 128
	Neighbourhood Embedding size	16, 32, 64
	Number of Neighbours	5, 10, 15, 20
	Number of Convolutional Kernels	32, 64, 128
	Convolutional Kernel Size	3, 4, 5, 8
	Pool Kernel Size	3, 4, 5, 8
De-biased MF	Embedding size	10, 20, 30, 50
FM	Learning Rate	0.01,0.005,0.001,0.0005,0.0001
	Embedding size	10, 20, 30, 40, 50
TransRec	Embedding size	16, 32, 64, 128
GRU4Rec	Embedding size	16, 32, 64, 128
SASRec	Layer Normalization Epsilon	1e-4, 1e-5, 1e-6, 1e-8, 1e-12
	Hidden dropout probability	0.1-0.9
	Attention dropout probability	0.1-0.9
BERT4Rec	Layer Normalization Epsilon	1e-4, 1e-5, 1e-6, 1e-8, 1e-12
	Hidden dropout probability	0.1-0.9
	Attention dropout probability	0.1-0.9
	Mask ratio	0.1-0.9
BiCoRec	Layer Normalization Epsilon	1e-4, 1e-5, 1e-6, 1e-8, 1e-12
	Hidden dropout probability	0.1-0.9
	Attention dropout probability	0.1-0.9
	Unsupervised loss weight	0.0-1.0
	Temperature	0.0-1.0
	Weight input dimension	10, 20, 30, 50, 100