UET4REC : A HYBRID U-NET AND TRANSFORMER MODEL FOR SEQUENTIAL RECOMMENDATIONS

A PROJECT REPORT

Submitted by

PAVITRA N

(2023176028)

A report for the phase-I of the project

submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment

for the award of the degree

of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SPECIALIZATION IN AI & DS

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600 025

JAN 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONA FIDE CERTIFICATE

Certified that this project report titled **UET4Rec: A HYBRID U-Net AND TRANSFORMER MODEL FOR SEQUENTIAL RECOMMENDATIONS** is the bona fide work of **PAVITRA N (2023176028)** who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE:	**Dr. K. VIDYA**
DATE:	**ASSISTANT PROFESSOR**
	PROJECT GUIDE
	DEPARTMENT OF IST, CEG
	ANNA UNIVERSITY
	CHENNAI 600025

COUNTERSIGNED

Dr. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

Recommendation system plays a crucial role and has becomes an integral part in modern application for providing personalized content to the users based on their preferences and behavior. There are several models employed to perform these recommendations. Traditional models fails to capture the long-term dependencies and recent user activities. This is overcomed by Sequential model which uses sequence of user-item interactions as input to predict the next item.

The proposed method UET4REC is a Sequential model with novel architecture that uses the strengths of U-Net and Transformer models for performing recommendation system. This architecture places the transformer module between the convolutional Encoder and Decoder. The encoder captures hierarchical and sequential features using the 1D convolution layer. The Transformer models captures the complex temporal dependencies using self-attention mechanism. The decoder layer reconstructs these interactions by using skip connections to preserve the contextual information.

The outcome of this project shows the ability of the model to perform recommendation tasks while maintaining scalability and effectively able to capture temporal information of time-series data. The architecture is effective in addressing challenges common in sequential recommendation system like capturing both short-term and long-term user behavior and achieving computational efficiency.

ABSTRACT (TAMIL)

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude and sincere thanks to our project supervisor, **Dr. K. VIDYA,** Associate Professor, Department of Information Science and Technology, College of Engineering, Guindy, for her invaluable guidance, continuous support, and encouragement throughout the course of this project. Her insightful feedback and constant motivation were instrumental in shaping the outcome of this work.

I am also deeply indebted to **Dr. S. SWAMYNATHAN,** Professor and Head of the Department of Information Science and Technology, Anna University, Chennai, for providing the necessary resources and facilities that enabled me to undertake and complete this project.

I would also wish to express my deepest sense of gratitude to the Members of the Project Review Committee: **Dr. S. SRIDHAR,** Professor, **Dr. G. GEETHA,** Associate Professor, **Dr. D. NARASHIMAN,** Teaching Fellow Department of Information Science and Technology,College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping me improve my project.

I also thank all the faculty and non-teaching staff members of the Information Science and Technology, CEG Campus, Anna University, Chennai, for their valuable support throughout the course of my project work.

PAVITRA N

(2023176028)

TABLE OF CONTENTS

ABSTRACT iii
ABSTRACT (TAMIL) iv
LIST OF TABLES viii
LIST OF FIGURES ix
LIST OF SYMBOLS AND ABBREVIATIONS x

1 INTRODUCTION 1
1.1 INTRODUCTION TO SEQUENTIAL RECOMMENDATION SYSTEMS 1
1.2 TRANSFORMER BASED SEQUENTIAL RECOMMENDATION SYSTEMS 2
1.3 PROPOSED METHODOLOGY 2
1.4 OBJECTIVE OF PROPOSED WORK 3
1.5 PROBLEM STATEMENT 3
1.6 ORGANIZATION OF THE THESIS 4

2 LITERATURE SURVEY 5
2.1 SEQUENTIAL RECOMMENDATION APPROACH 5
2.2 SOCIAL e-COMMERCE RECOMMENDATION SYSTEM 6
2.3 TRANSFORMER BASED RECOMMENDATION SYSTEM 6
2.4 HYBRID MODELS FOR RECOMMENDATIONS 7
2.5 CHALLENGES IN EXISTING SYSTEM 8
2.6 MOTIVATION OF PROPOSED WORK 9

3 SYSTEM ARCHITECTURE 10
3.1 PROPOSED ARCHITECTURE 10
3.2 COMPONENTS OF PROPOSED ARCHITECTURE 11
3.2.1 DATA PROCESSING 11
3.2.1.1 CONVERSION INTO TIME-SERIES SEQUENCE 12
3.2.2 ENCODER 13
3.2.3 TRANSFORMER LAYER 13
3.2.3.1 MULTI-HEAD SELF-ATTENTION (MSA) 14

3.2.3.2 POINT-WISE FEED-FORWARD NETWORK (PFFN) 14
3.2.4 DECODER LAYER 15
3.2.5 SKIP CONNECTIONS 15
3.2.6 MODEL AUGMENTATION 16
3.2.7 PREDICTION LAYER 16
3.3 TRAINING PROCESS 17
3.4 MODEL EVALUATION 17
3.4.1 HIT RATIO 17
3.4.2 NORMALIZED DISCOUNTED CUMULATIVE GAIN (NDCG) 18
3.5 SOFTWARE REQUIREMENTS 19
3.5.1 PROGRAMMING LANGUAGE 19
3.5.2 DEVELOPMENT TOOL 20

4 IMPLEMENTATION 21
4.1 CONVERSION TO TIME-SERIES DATA 21
4.2 SEQUENTIAL RECOMMENDATION WITH UET4Rec 23
4.3 MODEL AUGMENTATION AND PREDICTION LAYER 25

5 RESULTS AND ANALYSIS 27
5.1 DATASET USED FOR ANALYSIS 27
5.2 MODEL EVALUATION RESULTS 27
5.2.1 PERFORMANCE ON MOVIELENS-1M DATASET 28
5.2.2 PERFORMANCE ON RETAILROCKET DATASET 29
5.2.3 PERFORMANCE ON AMAZON-BEAUTY DATASET 29
5.3 COMPARISON WITH SASRec MODEL 31

6 CONCLUSION AND FUTURE WORK 35
6.1 CONCLUSION 35
6.2 FUTURE WORK 35

REFERENCES 36

LIST OF TABLES

5.1 Dataset Statistics 27
5.2 Performance Comparison on MovieLens-1M Dataset with Different Embedding Sizes 29
5.3 Performance Comparison on RetailRocket Dataset with Different Embedding Sizes 30
5.4 Performance Comparison on Amazon-Beauty Dataset with Different Embedding Sizes 31
5.5 Performance Comparison UET4Rec model with SASRec model with Amazon-Beauty Dataset 33

LIST OF FIGURES

3.1 UET4Rec Architecture 10
3.2 General framework of Up/Down layers 15

5.1 Training loss and accuracy curve of SASRec model 32
5.2 Training loss and accuracy curve of UET4Rec model 32
5.3 Training loss curve comparison of UET4Rec and SASRec 34
5.4 Training accuracy curve comparison of UET4Rec and SASRec 34

LIST OF ABBREVIATIONS

UET4Rec	U-Net encapsulated Transformer
BERT	Bidirectional Encoder Representations from Transformers
SASRec	Self-Attentive Sequential Recommendation
UTC	Universal Time Coordinated
PFFN	Point-Wise Feed-Forward Network
MA	Model Augmentation
GELU	Gaussian Error Linear Units
HR	Hit Ratio
NDGC	Normalized Discounted Cumulative Gain
DCG	Discounted Cumulative Gain
IDGC	Ideal Discounted Cumulative Gain

CHAPTER 1

INTRODUCTION

Recommendation system is a type of information filtering system that attempts to provide suggestions based to user based on their preferences. Recommending a sequence of items based on the user's previous history of purchases and clicks in an e- service is challenging, but still an important service to be provided. It plays a crucial role in business, as a good recommendation system can increase customer retention and the sales. A recommendation system uses user data to help predict what the user would look or select among the various number of options available. The traditional recommendation system recommends to the user based on two methods. One is the content- based filtering and the second method is collaborative filtering. The content-based filtering is based on similarity of items and user features, given information about user and items they have interacted with, models the likelihood of new interactions. In collaborative filtering, preferences information is based on preferences information from many uses' similarity of user preferences behavior. The model is built from user's past behavior like items purchased or ratings.However, these traditional recommendation systems have some disadvantages like contextual awareness and finds it difficult in handling long-term dependencies. There are other types of recommendation systems like sequential recommendation system, fairness-aware recommender and reciprocal recommender.

1.1 INTRODUCTION TO SEQUENTIAL RECOMMENDATION SYSTEMS

The sequential recommender uses the user history for recommendations. It works by learning user-interaction patterns over time.

The sequential model creates a relationship between the previous items and the current items by assigning weights to each item, which help the algorithm to learn the sequential behavior of the user. The model is trained to understand the user's history of interactions among multiple users enabling it to formulate the next item prediction. The algorithms used for sequential recommendation systems are Recurrent neural networks, Markov chain and Attention models. As mentioned earlier the sequential recommendation systems uses the order of sequence of user interaction or behaviour it is useful it is useful when users interact with items in a particular sequence like product recommendation on e-commerce sites, songs in a playlist and many more. Sequential models are context aware making an improvement in the quality of recommendation. Since it is trained based on the user data like interactions and preferences the recommendations are personalized to every user. Sequential recommenders explicitly model the order of interactions, which helps in understanding user intent at each step. For example, if a user recently watched several action movies, a sequential model might recommend more action or thriller movies next, rather than relying on general past interests.

1.2 TRANSFORMER BASED SEQUENTIAL RECOMMENDATION SYSTEMS

Transformer based recommendation system uses attention mechanism to capture user decencies. It can handle long-term dependencies efficiently and learn complex user behaviour patterns. By explicitly modelling sequential interactions, transformer-based recommenders can distinguish between similar items that appear at different points in a user's journey. But still there are few drawbacks in a transformer-based model like it required huge computation. It has to use large embedding size and long sequence to perform well. The model must be able to differentiate purchases and clicks. Another problem is transformer models requires large training dataset.

1.3 PROPOSED METHODOLOGY

The proposed work aims to design a sequential recommendation system based on a hybrid architecture called UET4Rec. The proposed model combines the U-Net and Transformer models, though they regularly solve challenges in terms of computational efficiency and performance. The aim is on an efficient system that optimizes user preference predictions smartly while smartly controlling resource consumption.

The UET4Rec model architecture is composed of an encoder, a Transformer block, and a decoder. In the former, the U-Net encoder compresses input data in order to reduce their size without losing any important information. The Transformer block captures the sequence so that it can get better knowledge about user behaviour patterns. The original data dimensions are restored using U-Net decoders by utilizing skip connections to take hold of the sequence throughout the process. The proposed method reduces the computational burden of the transformer without degrading its performance by using the above method.

1.4 OBJECTIVE OF PROPOSED WORK

The Objective of the proposed method is

- To propose a novel sequential model UET4REC which embed transformer model inside U-net.
- To reduce the computational burden of the transformer.
- To recommend items based on the user's interactions.

1.5 PROBLEM STATEMENT

A sequential recommender uses user history as the basis of personalized suggestion. But the common problem that needs to be addressed are computational cost, handling sparse dataset, capturing long and short-term patterns. UET4Rec is designed to overcome limitations in computational efficiency, dependency modeling, optimization for interaction importance, handling of sparse data, and personalization, all while improving recommendation accuracy.

1.6 ORGANIZATION OF THE THESIS

This Thesis is organized into 6 chapters, describing each part of the project with detailed illustrations and system design diagrams.

- Chapter 2 discusses the existing systems and various methods required for the proposed system.
- Chapter 3 discusses the various concepts used in the proposed system along with overall system architecture.
- Chapter 4 discusses the implementation of proposed work.
- Chapter 5 discusses the result and analysis of the project.
- Chapter 6 discusses the conclusion and future work of the findings.

CHAPTER 2

LITERATURE SURVEY

The recommendation systems from which traditional methods for recommendation like collaborative filtering, content based filtering, Matrix factorization, and the challenges faced by these methods and how new models like transformer models and hybrid models addresses these challenges and outperforms are discussed in this chapter.

2.1 SEQUENTIAL RECOMMENDATION APPROACH

The paper survey of sequential recommendation systems gives an overview of the sequential recommendation system[1]. It shows a comparison between the sequential recommendation system and the traditional recommendation system. It discusses the taxonomy of the recommendation systems which includes the models like Markov Chain Models, Matrix Factorization with Sequence Modeling, Deep Learning Approaches, Attention-based Models, Graph-based Models, Contrastive Learning-based Approaches and also about evaluation metrics and the challenges.

The sequential recommendation system uses algorithms like Markov Chain, Attention models and Recurrent neural network for the prediction of the next item. Factorized personalized Markov chain (FPMC) combines Markov chain and Markov factorization [2]. It creates a transition tensor by modeling user–item and item–item relationships to acquire temporal information and long-term user preferences based on past interactions. However, due to the addition of user dimensions, the issue of sparsity can affect the model performance.

2.2 SOCIAL e-COMMERCE RECOMMENDATION SYSTEM

Traditional methods fail to capture the difference between shopping behavior and influence of social media. This is addressed by using CROSS framework that integrates user-item interaction across both platforms and separates user's interest into a static component and the influence by social relation. It is done by combing Neural collab filtering and Matrix factorization [3]. Another approach on social e-commerce platform where users share product link among their group might make purchases based on these recommendations [4]. It is done by using the model called Triad based word-of-mouth recommendation model which also employs matrix factorization to capture influence of the person who shared and interest of the receiver which optimizes recommendations.

2.3 TRANSFORMER BASED RECOMMENDATION SYSTEM

Transformers in general uses attention mechanism which was used in task like natural language processing. Self-attention mechanism allows the transformer to capture the dependencies among a long sequence. In recommendation systems it is used to capture sequence of dependencies of user-item interaction data. The landmark paper highlight of how transformer works on attention-based mechanism [5]. It highlights the advantages of how self-attention reduces computational complexity and parallelizes training making transformer fast and scalable compared to traditional RNN or CNN models. Transformer models like BERT are used for recommendation systems which uses masked language model technique to predict missing items in user interaction history. It can be deployed in areas where the user preferences change dynamically[6].

The traditional methods for recommendation systems like content based filtering and collaborative filtering is combined along with BERT to make even more accurate predictions outperforming BERT4Rec model [7]. SASRec is another recommendation model that uses self-attention for sequential recommendation. The core of this model relies on the transformer architecture. It provides sequential recommendations to users by adaptively assigning weights to previous items in the sequence to predict the next item [8]. Based on these transformer models' attention mechanism advance self-attention mechanism can be used to capture sequential data. The integration of enhanced self-attention mechanism with GRU (Gates Recurrent Unit) and Bi-LSTM to dynamically capture both short- and long-term dependencies [9]. This approach significantly increases model performance from traditional methods which uses Markov Chain to capture long-term dependencies.Transformer models can also be trained to work for specific platforms like MOOC which handles problems like data sparsity. It outperforms the base models like BERT4Rec in next course prediction and achieves better accuracy and ranking performance metrics [10].

2.4 HYBRID MODELS FOR RECOMMENDATIONS

Recent developments in sequential recommendation have used deep learning models. The work Personalized 'Top-N Sequential Recommendation via Convolutional Sequence Embedding' introduced a Convolutional Neural Network (CNN) based model called Caser. It embeds item sequences as images and captures sequence information by extracting local features from the images. It determines Top-N recommendation which is personalized to every user uniquely represents user recent interaction as image and uses convolutional filters to capture sequence patterns and preferences. FNET4Rec uses transformer model's Point wise Feed Forward Network along with Fourier transform to capture relationship between items bi-directionally using BERT4Rec by encoding user interaction sequence and then passing it to the transformer module [11].

Another approach uses graph based embedding technique to capture user-item interactions and their temporal patterns and then uses transformer layers for attention mechanism [12]. This approach enhances global contextual information. A simple convolutional generative network which uses NextItNet, a session-based recommendation model was built for next item recommendation[13]. It uses convolution for capturing long-range dependencies, residual learning for deep-architecture optimization, and masked convolutions to ensure sequential prediction accuracy. But the model complexity is high and also suffers from potential over-fitting when there is sparse dataset. Recent developments in sequential recommendation have used deep learning models. The work Personalized 'Top-N Sequential Recommendation via Convolutional Sequence Embedding' introduced a Convolutional Neural Network (CNN) based model called Caser. It embeds item sequences as images and captures sequence information by extracting local features from the images[14]. A U-Net encapsulated transformer is used for meme classification task by integrating LLM and UET which leverages GPT-4 to generalize contextual information about meme content and address the challenges in meme classification [15]. The paper uses GPT-4 for contextual insights and KeyBERT for key phrase extraction, processing these embeddings through a U-net Encapsulated Transformer (UET) to efficiently classify memes by sentiment, emotion, and intensity.

2.5 CHALLENGES IN EXISTING SYSTEM

Common challenges of the existing system is the inability to effectively capture long-term and short-term user preferences. Traditional methods focus mainly on recent interactions while RNN models cannot capture long-term dependencies. These limitations are partially addressed by transformer models by using self-attention mechanism to model bi-directional dependencies. But, transformer models requires high computational resources and are sensitive to sequence lengths.

Another challenge is handling sparse dataset and cold-start problem. Having sparse dataset with no or less interactions often degrades the performance of advanced models also. Scalability and computational efficiency is another challenge that needs to be addressed when it comes to large datasets. Incorporating neural network with transformer models tries to solve these issues but faces challenges in maintaining prediction accuracy.In sequential recommendation system capturing the temporal and contextual information is a challenging task. Many models fails to address temporal dynamics in user-item interaction.

2.6 MOTIVATION OF PROPOSED WORK

The proposed UET4Rec model encapsulates a Transformer within a U-Net, is a novel method that is designed to address the challenges like

- Capture long-term dependencies effectively
- handle sparse datasets
- Capture meaningful temporal and contexuial information
- Reduce computational complexity

CHAPTER 3

SYSTEM ARCHITECTURE

This chapter focuses on the system architecture of the proposed recommendation system. It represents the overall architecture of the system and how a sequence of time series data is used for the next item prediction is being implemented. The proposed system comprises of U-net architecture encapsulating a transformer module to process the time series data effectively. The UET4Rec encapsulates transformer block in between the encoder-decoder modules.

3.1 PROPOSED ARCHITECTURE

The architecture of the model shown in Figure 3.1 integrates the encoder-decoder framework of U-Net with Transformers. The UET4Rec architecture is particularly designed to process sequence of user-item interaction effectively and to capture both recent interactions and historical trends. The

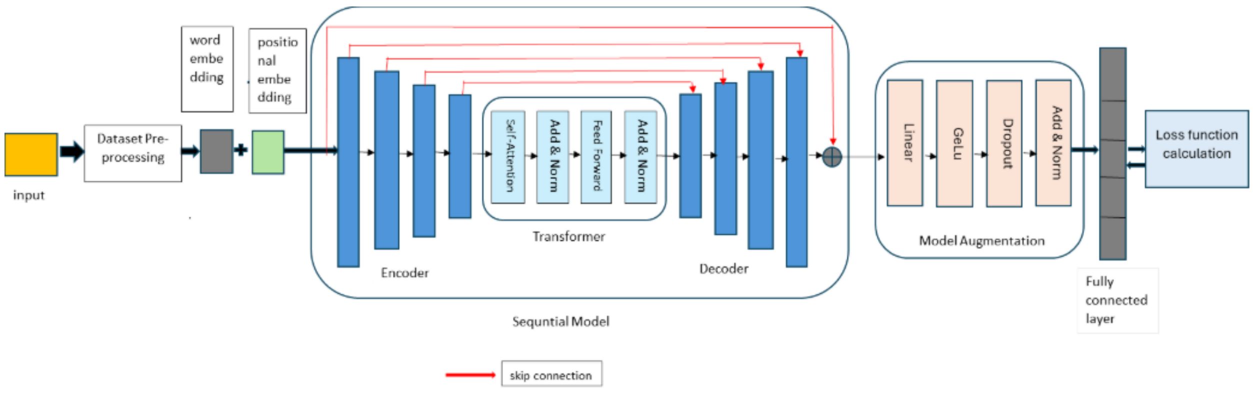

Figure 3.1: UET4Rec Architecture

generated sequence is fed into the model. The Encoder layer of U-Net process this sequence by down-sampling and extracting hierarchical features. It is then passed to a transformer module which captures sequence dependencies using self-attention mechanism. The Decoder layer then reconstructs the and integrates the Encoder features through skip connections. The output from the Decoder is then passed through a Model Augmentation layer and a Fully connected layer for the final prediction.

3.2 COMPONENTS OF PROPOSED ARCHITECTURE

The detailed explanation of each components in the proposed model is discussed here

3.2.1 DATA PROCESSING

The datasets used for this work are public datasets. Three datasets are used to show how the model handles complex and large amount of dataset and to show models performance on sparse dataset. Movielens-1M is a widely used dataset for recommendation systems related works. It was mainly designed for collaborative filtering and content-based filtering recommendations systems. It contains a timestamp data which is used for sorting the user history in a sequence of order in which he has seen a movie Retail Rocket dataset is collected from e-commerce website contains data related to user interactions on an e-commerce platform. Timestamp data present here is used for ordering a user's interaction with a product on time basis Amazon-Beauty dataset is from Amazon- review data which contains UserId, ProductID and Timestamp. This is a sparse dataset and the users with less interactions are more. This is used to show how the model is able to handle a sparse dataset.

3.2.1.1 CONVERSION INTO TIME-SERIES SEQUENCE

Sequential recommendation system uses time series data in order to predict or recommend the next item to the user. So, the dataset should be converted into a representation of user-item interaction sequences in a way that captures sequential dependencies and contextual relationships. After loading the dataset filter and exclude data with very few interactions as it may introduce to noise. The timestamp data provided in each dataset is Unix timestamp value, which represents the number of seconds that have elapsed since January 1, 1970 (00:00:00 UTC). Convert this time stamp into a standard date and time format. The categorical values of UserID and ProductID are then encoded to unique integers as the model requires numerical inputs. The encoded values are then sorted based on the timestamp data to represent the interaction between the user and item in correct sequence of chronological order. The generated sequence represents the real-world behaviour.

The UET4Rec is a sequential model and it requires a fixed-length inputs. The above generated sequence varies from person to person. So a fixed length is given a value N and if the sequence has less interactions than the fixed length N then it is padded to maintain uniform input dimension. If the sequence has more interactions and is a long sequence, then it is divided into smaller chunks. This sequence is then split as inputs and targets where the input is history of interaction and the target is the next item to be predicted. This inputs and targets serves as the model's input which is the encoded value of user's interaction in a sequential manner.

The user-item sequence is passed through a embedding function that converts this sequence into word embeddings which has the dimension of the embedding size of the transformer and the sequence length. The word embedding then adds position embeddings in order to indicate sequence number of each item. The final input for the model will be this word+position embeddings.

3.2.2 ENCODER

U-net Encoder is a series of down layers which is used to reduced the input embedding size and extract features. Encoder contains three down layers. The first layer starts with the dimension BxNxW where B is the batch size, N is the sequence length, and W is the embedding dimension. For the next encoder layer the embedding dimension is reduced to half so the dimension becomes BxNxW/2. This is done again by the following encoder layers. Each encoder layer contains the following:

- Convolution: 1D convolution layer captured features of different sizes. It detects patterns in local features of the sequences
- Batch Normalization: Followed by Convolution the Batch Normalization normalizes the intermediate representation to stabilize the training
- Leaky ReLU: Leaky ReLU activation function is followed after the normalization for introducing non-linearity and to address vanishing gradient problem.
- Dropout: Finally, a Dropout layer is added to prevent over-fitting of the model. This is illustrated in Figure 3.2.

The down sampling done by encoder ensures that the Transformer module operates in reduced dimension and optimize the computational efficiency without needing to sacrifice important information.

3.2.3 TRANSFORMER LAYER

The output from the Encoder is passed as the input to the Transformer. The Transformer captures relationships among different items in the sequence. This ensures the model to identify both short-term and long-term patterns. The transformer layer contains the following components:

3.2.3.1 MULTI-HEAD SELF-ATTENTION (MSA)

The Multi-head Self-Attention contains Query(Q), Key(K) and Value(V) components to compute attention weight and capture the relationships of the items. The attention score is calculated by the following equation

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V \tag{3.1}$$

Where d is the embedding dimension size. The correlation between Q and K obtains weights of historical data of different items by calculating self-attention.

3.2.3.2 POINT-WISE FEED-FORWARD NETWORK (PFFN)

PFFN is followed by the MSA layer which is responsible for applying non-linear transformations to each position of the sequence independently. It processes each token in position embedding to ensure computational efficiency and parallelization. FFN applies two linear transformation function. Each embedding is passed through a fully connected layer where it is expanded into a higher dimension using weight matrix. This output is then passes through a ReLu activation function to add non-linearity which enables the model to capture complex patterns. It is followed by the second transformation function which is a fully connected layer and it reduces the dimension back to its original size. A residual connection is made which adds the input to this output to preserve information and avoid vanishing gradient problem. This is achieved by passing the sum of this input and output to a LayerNorm normalization layer.

3.2.4 DECODER LAYER

The Decoder layer performs up-sampling where the features extracted are constructed into their original dimensions. It mirrors the encoder's structure. Each decoder layer doubles the embedding dimension W. The final output of decoder will be in dimension BxNxW. The up-sampling layer of Decoder contains the same components of the Encoder as 1D Convolution layer, Batch Normalization layer, ReLU activation function and Dropout layer.

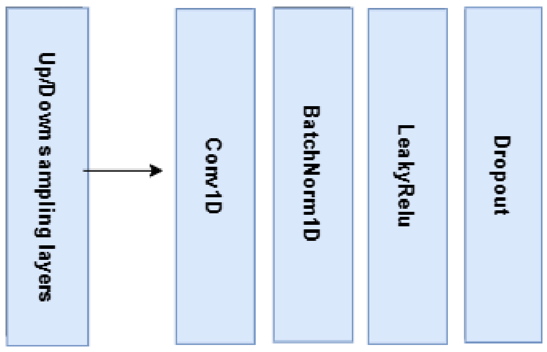

Figure 3.2: General framework of Up/Down layers

3.2.5 SKIP CONNECTIONS

While performing the compression of the original data, there will be a loss of information. Adding skip connection prevents this information loss while up-sampling. It is used to preserve spatial and contextual information which is lost during the down-sampling process which helps in the next-item prediction. Using skip connection, the Decoder gets the low-level information that were preserved in the Encoder part.

3.2.6 MODEL AUGMENTATION

The Model Augmentation is added after the Decoder to enhance the performance of sequence recommendations. The MA layer contains a Feed-Forward Network which consists of Linear, Gaussian Error Linear Units (GELU) and Dropout layer. The purpose of adding these layers is to improve the model to be able to generalize to unseen or noisy data which is crucial when working with sparse dataset.

3.2.7 PREDICTION LAYER

The prediction layer connects work done by all layers to the final task which is recommending the next item. The input of the prediction layer is a dense vector summarizing all information from previous layers. This information is passed to a fully connected layer, which transforms the vectors into probabilities by adding weights and bias.

The highest probability indicates that it is the next item most likely to be in the sequence. The algorithm will select top K items based on these probabilities. The main objective of the prediction layer is to minimize the error between the true item in the sequence and the derived predicted next item. The model is trained using Cross-Entropy Loss which quantifies the error between the predicted probability distribution and actual ground truth. The loss is computed as

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log(y_{i,t}), \tag{3.2}$$

where $y_{i,t}$ is the probability assigned to the true next item i at timestep t. N is the total number of sequences in the batch. Minimizing Cross-Entropy loss at training, the model can make adjustments to its parameters to better align its predictions with true sequence data.

3.3 TRAINING PROCESS

The model UET4Rec is trained using Cross-entropy loss function as mentioned earlier which measures how well the predicted probabilities of the model match to the ground truth. The training process is done as follows:

- **Forward Pass :** The input sequence is passed to the sequential model containing Encoder, Transformer and Decoder layers and generate a probability distribution.
- **Loss Calculation :** The Cross-Entropy loss is calculated using these predicted probabilities and Ground truth items.
- **Update :** The model parameters are updated using Adam optimization algorithm.

This process is iterated until model converges to minimum loss values which indicates that model has learned to make accurate predictions.

3.4 MODEL EVALUATION

After the model is trained, the model outputs top K predictions. The evaluation of these predicted items is done using Hit Ratio(HR) and Normalized Discounted Cumulative Gain (NDGC).

3.4.1 HIT RATIO

Hit Ratio(HR) is evaluation metrics for recommendation system which is widely used to measure effectiveness of predicting relevant items to the users. It evaluates whether the actual item the user interacted with exists in the Top-K recommended items. The formula to calculate Hit Ratio is

$$HR@K = \frac{\text{Number of hits in top-}K}{\text{Total number of users}} \tag{3.3}$$

A ”hit” occurs when true item is present in top-K recommendations. The value of HR lies between 0-1 where 1 indicated perfect predictions for all the users.

3.4.2 NORMALIZED DISCOUNTED CUMULATIVE GAIN (NDCG)

NDGC is another widely used metrics to evaluate recommendation systems by considering the ranking of items within the top-K. It is computed as follows

Discounted Cumulative Gain (DCG):

$$DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)} \tag{3.4}$$

Here, rel_i is the relevance score of the item at position i, which is often binary (1 for the ground-truth item and 0 otherwise).

Ideal Discounted Cumulative Gain (IDCG):

$$IDCG@K = DCG@K \text{ of the ideal ranking} \tag{3.5}$$

This is the maximum possible DCG, obtained when all relevant items are ranked at the top.

NDGC:

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{3.6}$$

NDCG ranges from 0 to 1, where 1 indicates perfect ranking of the recommendations. This ranking order makes it more informative than HR scenarios as it ensures both correctness of recommendations and ranking orders. Both these metrics are used for evaluating the model to show how well the model's prediction. By identifying strengths and weaknesses through evaluation, the process can be iteratively refined to ensure that the results meet the requirements.

3.5 SOFTWARE REQUIREMENTS

This project requires multiple components for recommending user-item interactions. This section elaborates upon various software requirements specific to different phases of the project.

3.5.1 PROGRAMMING LANGUAGE

Python: The primary programming language used for implementing deep learning models Python is chosen for its extensive ecosystem of deep learning frameworks like TensorFlow and PyTorch, as well as its capabilities in handling large-scale data processing and analysis, making it highly suitable for this project.

Required Libraries

Numpy, a fundamental library used for array handling and performing mathematical operations and calculating metrics.

Pandas, is used for loading and pre-processing large datasets and for other pre-processing steps.

Skikit-learn, a machine learning library used for data splitting,evaluation and provides tools for data processing. Here it is used for label encoding the user and product IDs.

PyTorch, a powerful deep-learning framework for defining and training custom deep-learning architectures like UET4Rec. It implements the model augmentation layers, attention mechanism and other neural network modules.

Matplotlib, a visualization library for plotting training metrics like loss and accuracy.

3.5.2 DEVELOPMENT TOOL

The IDE or Editor used for implementing the project is Jupyter Notebook which is an ideal environment for implementing machine learning and deep learning projects.

CHAPTER 4

IMPLEMENTATION

This chapter explains the implementation of the proposed hybrid recommendation system which encapsulated Transformer model between the Encoder and Decoder layer. It explains about the user-item interaction sequence generation and how UET4Rec uses this sequence for performing recommendations.

4.1 CONVERSION TO TIME-SERIES DATA

Three public datasets are used to evaluate the proposed work. The dataset is available is csv format which needs to be converted into a user item interaction data which is numerical value and it must be a temporal data. The dataset contains UserID, ProductID, Timestamp values which needs to be encoded, grouped and sorted according to the time of interaction with the item by the user.The generated user-item sequence data should be temporal which represents the sequence in which the user has interacted with the product or item. Sequences generated are then split into a fixed length value. The last item of the sequence is the target value which the model needs to predict after series of items which is set as targets and the sequence is mentioned as inputs.

Algorithm 4.1 Generate Time Series Data

Require: Raw dataset (`.csv` format)
Ensure: Processed data with inputs and targets

```
 1: procedure GENERATE TIME-SERIES DATA
 2:     Step 1: Load and Filter Dataset
 3:     Load the raw dataset and exclude users or items with fewer interactions.
 4:     Step 2: Timestamp Conversion
 5:     Convert Unix timestamps to a standard date-time format.
 6:     Step 3: Encode User and Item IDs
 7:     Encode categorical values of UserID and ProductID to numerical data.
 8:     Step 4: Sort Interactions
 9:     Sort the dataset by UserID and Timestamp to preserve the chronological
    order of interactions.
10:     Step 5: Group User Sequences
11:     Combine each user with their corresponding item interactions.
12:     Step 6: Generate Fixed-Length Sequences (N)
13:     for each user-item interaction sequence do
14:         if Sequence length < N then
15:             Pad the sequence with zeros to maintain length N.
16:         else if Sequence length > N then
17:             Divide the sequence into sub-sequences of length N.
18:         end if
19:     end for
20:     Step 7: Split as Inputs and Targets
21:     for each sequence do
22:         Input: All items except the last one.
23:         Target: The next item to be predicted.
24:     end for
25:     Step 8: Return Inputs and Targets
26:     Return the processed inputs and targets.
27: end procedure
```

Explanation: Algorithm 4.1 explains how the raw dataset is converted into a sequence of user-item interactions.

The algorithm first takes the raw dataset which is csv format and filters the users and items with less interactions. The dataset contains Timestamp in Unix which is converted into standard date-time format. Label encoder is used to encode the User and Product or Movie or Item ID based on the given dataset which is processed at that time. Now the values are sorted in according to timestamp and grouped with the ItemId which generated a user-item interaction sequence which is the order in which the user has interacted with the items in temporal manner.

The sequence is made to a fixed length sequences based the value of N. Padding is if sequence is short and sub-sequences are generated if the sequence is long. These fixed length sequences are split into inputs are targets where the input is the sequence of interaction by user to the item in temporal order and the target is the item to be predicted in that order. This inputs and targets is given as input to the model.

4.2 SEQUENTIAL RECOMMENDATION WITH UET4Rec

The architecture combines the strength of U-Net and Transformer model. The user-item sequences generates is transformed into embeddings using a combination of word and position embeddings. This is passed to the Encoder layers which reduces the embedding dimension when passed down to each layer. Here the embedding dimension is mentioned as W which is reduced by 2 across each layer. For example the Encoder dimension is given as 128 then it is reduced to 16 when it comes to the Transformer module. Each encoder layer applies a 1D Convolutional layer, Batch Normalization layer, Leaky ReLu and Dropout. The Transfomer models use Mulit-head self attention and Point-Wise Feed Forward Networks. The Self-Attention allows model to focus on imporatant parts of input sequence when processing each item in in the sequence. Multiple heads are used to focus on different aspects of the sequence. PFFN enriches the extracted features from the self-attention by introducing non-linearity. Skip connections between the Encoder and Decoder layers and the Decoder layer performs up-sampling by increasing the embedding size. Decoder layers applies the same layers present in Encoder part.

Algorithm 4.2 Sequential Recommendation with UET4Rec

```
Require: User-item interaction sequence (fixed length $N$), $B$: Batch size, $N$:
    Sequence length, $W$: Embedding dimension
Ensure: Processed embeddings and predictions

1:  Step 1: Input Embeddings
2:  Add word and positional embeddings for input sequence:
        $E_{\text{input}} = E_{\text{word}} + E_{\text{pos}}$
3:  Step 2: U-Net Encoder
4:  Input: $E_{\text{input}}$ of shape $(B, N, W)$.
5:  for each layer in the encoder do
6:      Apply 1D Convolution, Batch Normalization, Leaky ReLU, and Dropout.
7:      Halve the embedding dimension after each layer:
            $(B,N,W) \to (B,N,W/2) \to (B,N,W/4)$
8:      Store outputs for skip connections.
9:  end for
10: Step 3: Transformer Layer
11: Multi-Head Self-Attention:
12: Compute attention scores using:
        $\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$
13: $Q,K,V$: Query, Key, and Value matrices derived from the input.
14: Point-Wise Feed-Forward Network:
15: Apply two linear transformations with a ReLU activation in between.
16: Add a residual connection and normalize the output.
17: Step 4: U-Net Decoder
18: for each layer in the decoder do
19:     Process through three up-sampling layers:
20:     Use skip connections to combine encoder outputs with up-sampled features.
21:     Apply 1D Convolution, Batch Normalization, ReLU activation, and Dropout.
22:     Double the embedding dimension after each layer:
            $(B,N,W/4) \to (B,N,W/2) \to (B,N,W)$
23: end for
```

Algorithm 4.2 describes the step-by-step implementation of UET4Rec model to handle the user-item interaction sequences for sequential recommendations.

4.3 MODEL AUGMENTATION AND PREDICTION LAYER

The purpose of using MA is to enhance the output from the Decoder to improve quality of learned representation before prediction. It computes the augmented features using a Feed-Forward Network with Linear transformation layers, GELU and a Dropout layer which is given as

$$h_{\text{FFN}} = \text{FFN}(h'_t)$$

It is passed through a Normalization layer for stability.

Algorithm 4.3 Model Augmentation (MA)

```
Require: Output from the decoder, shape $(B,N,W)$
Ensure: Augmented representation $h^{t+1}$, shape $(B,N,W)$
1: Initialize Feed-Forward Network (FFN):
2:     - Linear transformation layers
3:     - Gaussian Error Linear Units (GELU)
4:     - Dropout layer
5: Compute augmented features:
6:     $h_{\text{FFN}} = \text{FFN}(h^{t'})$
7: Add residual connection:
8:     $h^{t+1} = \text{LayerNorm}(h^{t'} + h_{\text{FFN}})$
```

Explanation: Algorithm 4.3 explains how the Model Augmentation is implemeted. The FFN layer containing the Linear transformation functions maps the input to higher dimensional space GELU activation function is applied which is followed by a Dropout layer to avoid over fitting. The residual connections ensures that the model retains the original information while adding augmented features. The augmented features are then passed through the prediction layer for final prediction of next item in the sequence.

Algorithm 4.4 Prediction Layer

```
Step 1: Compute next-item prediction
1: Flatten $h^{t+1}$ for the final timestep $t$ into $s_t$
2: Compute linear transformation:
3:     $y_t = s_t W + b$
4: Apply softmax activation to normalize:
5:     $\hat{y}_t = \text{Softmax}(y_t)$
Step 2: Top-K Selection
1: for each batch $b$ do
2:     Select the indices of the top-$K$ highest probabilities in $\hat{y}_t$.
3: end for
4: Return the indices of the top-$K$ items for each batch.
```

Explanation Algorithm 4.4 explains how the prediction layer performs the next-item prediction. The prediction layer translates the learned embedding from the augmentation layer into next-item predictions. From the embedding sequence, the embedding in last time stamp is extracted which represents the user's current state. The extracted embedding is passed to a fully connected layer which applies linear transformation. The softmax activation function is applied to normalize the scores into probabilities that sums to 1.

From these probabilities the system selects the K items with highest probabilities. These K items are considered as next possible interactions. This layered and structured approach ensures the model predictions accurate.

CHAPTER 5

RESULTS AND ANALYSIS

This chapter discusses about evaluation of the proposed model. The model performance is evaluated using three datasets such as MovieLens-1M, RetailRocket and Amazon-Beauty. Metrics like Hit Ratio(HR) and Normalized Discounted Cumulative Gain (NDCG) are used for evaluating the quality of recommendations that the models predicts as the next series of items which the user might like.

5.1 DATASET USED FOR ANALYSIS

Three datasets as mentioned earlier are used for evaluating the model's performance. The below table show the statistics of the three datasets 5.1. It contains information about number of interactions, users and items of each dataset.

Table 5.1: Dataset Statistics

Dataset	**Movielens-1M**	**Beauty**	**RetailRocket**
Interaction	1,000,209	1,348,246	2,756,101
Users	6,040	883,753	1,407,580
Items	3,706	23,838	235,061

5.2 MODEL EVALUATION RESULTS

The results of UET4Rec model performance for each dataset is discussed here. The metrics are measured for different k values as k = [5, 10, 20] where k represents the top-K predictions of the model.

The input sequence length N is set as 10. The model is evaluated with two embedding dimension given as input to the encoder part which is reduced to half after each layer. So the transformer embedding size will be W/8, where W is the input dimension of the Encoder. Here the input dimension is varied as 128 and 64 for Encoder which is reduced to 16 and 8 when it comes to the Transformer. This reduces the computational load of the Transformer and as skip connections are used between Encoder to Decoder so there is no information loss. Adam optimizer is used with learning rate 0.001.

5.2.1 PERFORMANCE ON MOVIELENS-1M DATASET

The performance of the model is done on the MovieLens-1M dataset. The embedding size of the Encoder is varied as 128, 64 which is reduced to 16, 8 when it comes to the Transformer part. The performance comparison of the model on the MovieLens dataset with two different embedding sizes 16 and 8 is given in the table5.2. The batch size for this dataset is 256. The model is trained to 1000 epochs. Here for the embedding size 16 the HitRatio@10 is 0.7540 and NDGC@10 is 0.7064 and for embedding size 8 the HitRatio@10 is 0.6486 and NDGC@10 is 0.5195. This indicates that the better top-10 recommendations is done with large embedding size.

Table 5.2: Performance Comparison on MovieLens-1M Dataset with Different Embedding Sizes

Dataset	Metrics	Embedding Size = 16	Embedding Size = 8
MovieLens-1M	HR@5	0.7277	0.5778
	NDGC@5	0.6979	0.4966
	HR@10	0.7540	0.6486
	NDGC@10	0.7064	0.5195
	HR@20	0.7824	0.7219
	NDGC@20	0.7135	0.5380
	Train loss	1.0262	2.799
	Train Accuracy	0.7104	0.3667

Here we can see a significant increase the the metrics which shows that the model performance for the embedding size 16 is more efficient than 8 for this MovieLens-1M dataset.

5.2.2 PERFORMANCE ON RETAILROCKET DATASET

The RetailRocket dataset is used for evaluvating the performance of the model. The batch size of the dataset is 256 and the model is trained to 1000 epochs. The input embedding dimension is same which is varied as 128 and 64 for the Encoder layers. The performance comparison of the model on the RetailRocket dataset with two different embedding sizes 16 and 8 is given in the table 5.3. From the metrics value mentioned in the table there is no significant difference in the values for both embedding size. For the embedding size 16 the HitRatio@10 is 0.4147 and NDGC@10 is 0.3202 and for embedding size 8 the HitRatio@10 is 0.4206 and NDGC@10 is 0.3246. These values shows that model performs well with the embedding dimension as 8 for the RetailRocket dataset.

Table 5.3: Performance Comparison on RetailRocket Dataset with Different Embedding Sizes

Dataset	Metrics	Embedding Size = 16	Embedding Size = 8
RetailRocket	HR@5	0.3735	0.3746
	NDGC@5	0.3068	0.3097
	HR@10	0.4147	0.4206
	NDGC@10	0.3202	0.3246
	HR@20	0.4515	0.4620
	NDGC@20	0.3295	0.3351
	Train loss	0.4515	0.4620
	Train Accuracy	0.3295	0.3351

5.2.3 PERFORMANCE ON AMAZON-BEAUTY DATASET

Here the Amazon-Beauty dataset is used to show the performance of the model. The batch size is set as 32 because after pre-processing the total number of sequence generated was less when compared to other two dataset. There is no change in the embedding dimension of the input Encoder layer which is varied as 128 and 64 to see how the model performs in each case. he performance comparison of the model on the Amazon-Beauty dataset with two different embedding sizes 16 and 8 is given in the table 5.4 From the metrics value mentioned in the below table it is seen that the model performs well when the embedding dimension is 8. But the training loss and accuracy values are better in the when the embedding dimension is 16 for the Transformer model.

Table 5.4: Performance Comparison on Amazon-Beauty Dataset with Different Embedding Sizes

Dataset	Metrics	Embedding Size = 16	Embedding Size = 8
Amazon-Beauty	HR@5	0.0598	0.0606
	NDGC@5	0.0437	0.0449
	HR@10	0.0781	0.0803
	NDGC@10	0.0496	0.0512
	HR@20	0.1026	0.1066
	NDGC@20	0.0559	0.0579
	Train loss	0.0801	0.3518
	Train Accuracy	0.9746	0.8978

From the metrics shown in the above tables we can see that the change in Transformer embedding size doesn't affect the performance of the model and it can handle any dataset with any embedding dimension without much change in the metric values.

5.3 COMPARISON WITH SASRec MODEL

The comparitive is done to show the UET4Rec model performance is better when compared to another seqentail model. SASRec is another sequential recommendation model that uses self-attention mechanism for predicting the next-item which also uses the user-item interaction sequence as input. The dataset used here for comparison is the Amazon-Beauty dataset and the input embedding dimension is 128 for the transformer. The model is trained for 100 epochs and its performance during the training is shown in the Figure 5.1

The graph shows that the loss function during the model training stops decreasing after a period of time and it is around 8.6 which is very high. The accuracy is also is very less and it ranges between 0.3-0.4 after some epochs.

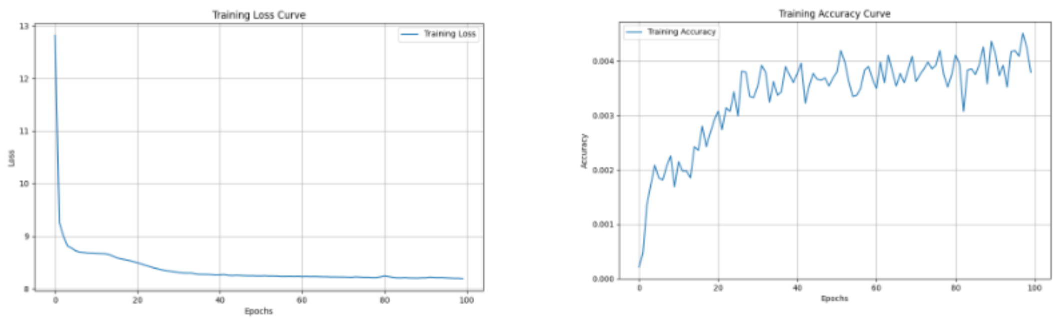

Figure 5.1: Training loss and accuracy curve of SASRec model

The UET4Rec model is also trained for 100 epochs and the embedding size is set as 128 to compare how the model performs with the same dataset. After training to 100 epochs the training loss and accuracy is plotted in the graph which is shown in Figure 5.2. The loss after 100 epochs is around 1.0 which can further be decreased by using more epochs. The accuracy is also increased to 0.96. This comparison shows that the performance of UET4Rec model is better when compared to the SASRec model during the training.

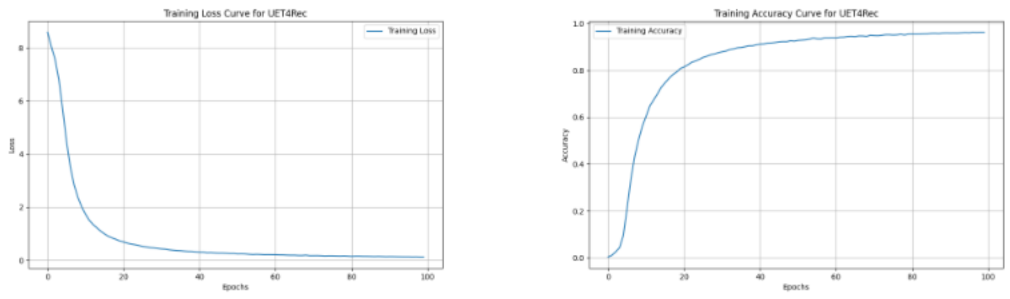

Figure 5.2: Training loss and accuracy curve of UET4Rec model

The test dataset is used to measure the evaluation metrics like Hit Ratio and Normalized Distributed Gain to compare the two models. The table shows to comparison of metrics for different K values as k=[5, 10, 20] where represents the Top-K predictions of the next-item which is recommended to the users.

The performance of UET4Rec model with the SASRec model on the Amazon-Beauty dataset using various evaluation metrics, including Hit Ratio (HR@k), Normalized Discounted Cumulative Gain (NDCG@k) is mentioned in the table 5.5.

Table 5.5: Performance Comparison UET4Rec model with SASRec model with Amazon-Beauty Dataset

Dataset	Metrics	SASRec	UET4Rec
Amazon-Beauty	HR@5	0.0210	0.0541
	NDGC@5	0.0125	0.0404
	HR@10	0.0320	0.0703
	NDGC@10	0.0161	0.0404
	HR@20	0.0513	0.0703
	NDGC@20	0.0209	0.0508
	Train loss	8.6720	0.1263
	Train Accuracy	0.0038	0.9618

From the table values it is shown that the UET4Rec model metrics are high when compared to the SASRec model for each k values. HR@5: UET4Rec achieves 0.0541, compared to SASRec's 0.0210. This represents more than double the performance of SASRec.

Figure 5.3 and 5.4 shows the training loss and training accuracy curves for both the models over the training time. The curves shows the convergence behaviour of each model and its ability to learn during training.

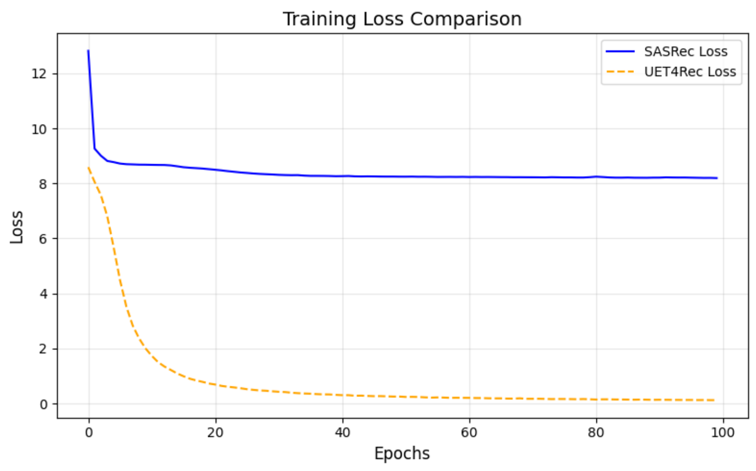

Figure 5.3: Training loss curve comparison of UET4Rec and SASRec

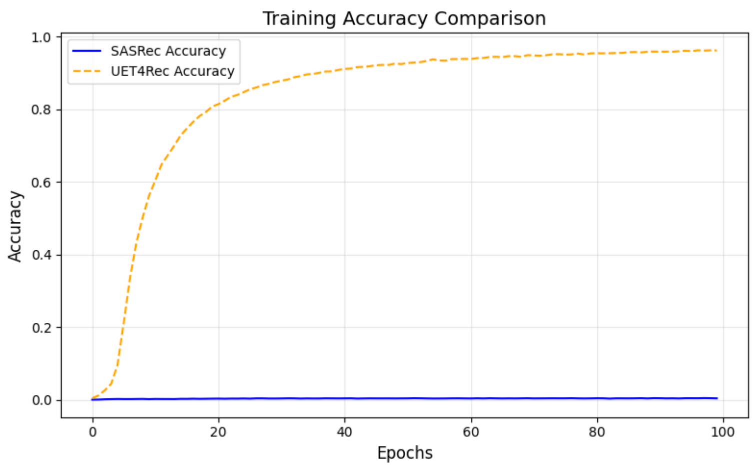

Figure 5.4: Training accuracy curve comparison of UET4Rec and SASRec

The SASRec starts with initial loss as 10 which gradually decreases over epochs and stabilizes at a relatively high loss value. UET4Rec model converges quickly within few epochs to a very low loss value.

In conclusion of the comparison the UET4Rec demonstrates better optimzation efficency than SASRec as it converges faster and the evaluvation in metrics value also shows the propsed UET4Rec model has high metric values.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The proposed hybrid sequential model UET4Rec for recommendation deals with the limitations of Transformer-only models. The compartive study done with with the SASRec models shows that the hybrid model comparatively better even when the transformer embedding size is reduced. Thus computational burden of the transformer is reduced. The model's performance metrics values calculated with Hit Ratio and Normalized Discounted Cumulative Gain for different datasets shows that the model can handle huge and complex datasets and effectively capture long-term dependencies.

6.2 FUTURE WORK

The loss function used for any sequential models are Cross-Entropy loss function or Binary-Cross entropy loss functions. Future works can include Reinforcement learning loss functions based on ratings given by the user for a product or item which makes the recommendation more personalized and also the recommended items with high ratings. This makes the model to perform better. The proposed hybrid architecture performance is done with e-commerce and entertainment related platforms alone. It can be used in other platforms or applications like healthcare, finance and education which has different properties and behaviour.

REFERENCES

[1] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. A survey of sequential recommendation systems: Techniques, evaluation, and future directions. *Information Systems*, 125:102427, 2024.

[2] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. pages 811–820. Association for Computing Machinery, 2010.

[3] Chen Gao, Tzu-Heng Lin, Nian Li, Depeng Jin, and Yong Li. Cross-platform item recommendation for online social e-commerce. *IEEE Transactions on Knowledge and Data Engineering*, 35:1351–1364, 2023.

[4] Chen Gao, Chao Huang, Donghan Yu, Haohao Fu, Tzh-Heng Lin, Depeng Jin, and Yong Li. Item recommendation for word-of-mouth scenario in social e-commerce. *IEEE Transactions on Knowledge and Data Engineering*, 34:2798–2809, 2022.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. pages 6000–6010. Curran Associates Inc., 2017.

[6] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. pages 1441–1450. Association for Computing Machinery, 2019.

[7] Thu Nguyen, Long Nguyen, Khoa Tan-Vo, Thu-Thuy Ta, Mong-Thy Nguyen Thi, Tu-Anh Nguyen-Hoang, Ngoc-Thanh Dinh, and Hong-Tri Nguyen. H-bert4rec: Enhancing sequential recommendation system on moocs based on heterogeneous information networks. *IEEE Access*, 12:155789–155803, 2024.

[8] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. pages 197–206. 2018 IEEE International Conference on Data Mining (ICDM), 2018.

[9] Xiaoyao Zheng, Xingwang Li, Zhenghua Chen, Liping Sun, Qingying Yu, Liangmin Guo, and Yonglong Luo. Enhanced self-attention mechanism for long and short term sequential recommendation models. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8:2457–2466, 2024.

[10] Chanapa Channarong, Chawisa Paosirikul, Saranya Maneeroj, and Atsuhiro Takasu. Hybridbert4rec: A hybrid (content-based filtering and collaborative filtering) recommender system based on bert. *IEEE Access*, 10:56193–56206, 2022.

[11] LiangLiang Chen and Guang Chen. Fnet4rec: A simple and efficient sequential recommendation with fourier transforms. pages 2210–2214. 2022 IEEE 8th International Conference on Computer and Communications (ICCC), 2022.

[12] Yi-Cheng Chen, Yen-Liang Chen, and Chia-Hsiang Hsu. G-transrec: A transformer-based next-item recommendation with time prediction. *IEEE Transactions on Computational Social Systems*, 11:4175–4188, 2024.

[13] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. pages 582–590. Association for Computing Machinery, 2019.

[14] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. pages 565–573. Association for Computing Machinery, 2018.

[15] Marvin John Ignacio, Thanh Tin Nguyen, Hulin Jin, and Yong-Guk Kim. Meme analysis using llm-based contextual information and u-net encapsulated transformer. *IEEE Access*, 12:125993–126005, 2024.