Sequence Recommendation based on Deep Learning

Gulsim Rysbayeva[1], Jingwei Zhang[2]
School of Computer Science and Information Security, Guilin University of Electronic Technology, No.1, Jinji Rd., Guilin 541004, Guangxi[1]
Guangxi Key Laboratory of Trusted Software, Guilin University of Elecronic Technology, Guilin 541004, China[2]

***Abstract*—Sequence recommendation systems have become increasingly popular in various fields such as movies and social media. These systems aim to predict a user's preferences and interests based on their past behavior and provide them with personalized recommendations. Deep learning, particularly Recurrent Neural Networks (RNNs), have emerged as a powerful tool for sequence recommendation. In this research, we explore the effectiveness of RNNs in movie and Instagram recommendation systems. We investigate and compare the performance of different types of RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), in recommending movies and Instagram posts to users based on their browsing history. Additionally, we study the impact of incorporating additional information such as user's demographics and Instagram hashtags on the performance of the recommendation system. We also evaluate the performance of RNN-based movie and Instagram recommendation systems in comparison to traditional approaches, such as collaborative filtering and content-based filtering, in terms of accuracy and personalization. The findings of this research provide insights into the effectiveness of RNNs in movie and Instagram recommendation systems and contribute to the development of more accurate and personalized recommendations for users**

***Keywords*—Long short-term memory (LSTM) and gated recurrent unit (GRU); RNN; deep learning; recommendation systems**

I. Introduction

Sequence recommendation systems have become increasingly popular in various fields, such as movies and social media. These systems aim to predict a user's preferences and interests based on their past behavior and provide them with personalized recommendations. One of the most promising approaches for sequence recommendation is deep learning, which has been shown to be effective in capturing the complex patterns and dependencies in sequential data.

Sequence recommendation systems have become an important area of research in recent years due to the growing amount of sequential data available in various fields such as music, videos, e-commerce, and social media. These systems aim to predict a user's preferences and interests based on their past behavior and provide them with personalized recommendations. Traditional approaches to sequence recommendation, such as collaborative filtering and content-based filtering, have been shown to be effective to some extent. However, they are limited in their ability to capture the complex patterns and dependencies in sequential data.

Deep learning, particularly Recurrent Neural Networks (RNNs), have emerged as a powerful tool for sequence recommendation. RNNs are particularly suitable for this task as they are able to model the temporal dependencies in the user's browsing history. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are two popular types of RNNs that have been widely used in sequence recommendation systems. LSTMs have been shown to be effective in capturing long-term dependencies in sequential data, while GRUs have been shown to be more computationally efficient.

Recent research has explored the application of RNNs in movie and Instagram recommendation systems. For example, researchers have used RNNs to model user's browsing history and predict their preferences for movies. Other researchers have used RNNs to model user's interactions on Instagram and recommend posts based on their interests. However, there is still a need to further investigate the effectiveness of RNNs in these systems and explore how incorporating additional information such as user's demographics, and Instagram hashtags can improve the performance of the recommendation system.

This research aims to fill this gap by exploring the effectiveness of RNNs in movie and Instagram recommendation systems and investigating the impact of incorporating additional information on the performance of the system. The findings of this research will contribute to the development of more accurate and personalized recommendations for users.

The vast amount of sequential data available in these domains, such as users browsing history, provides an opportunity for recommendation systems to offer personalized recommendations to users. However, despite the growing interest in using RNNs for sequence recommendation, there is still a lack of understanding of how well they perform in these specific domains and how to effectively incorporate additional information to improve the performance of the system. In particular, traditional approaches such as collaborative filtering and content-based filtering have been shown to be effective to some extent, but they are limited in their ability to capture the complex patterns and dependencies in sequential data, while RNNs, specifically LSTMs and GRUs, have shown to be promising in modeling temporal dependencies. This research aims to explore the effectiveness of RNNs in movie and Instagram recommendation systems by comparing the performance of different types of RNNs, and investigate the impact of incorporating additional information such as user's demographics and Instagram hashtags on the performance of the recommendation system. Additionally, the research aims to compare the performance of RNNs-based movie and Instagram recommendation systems with traditional approaches in terms of accuracy and personalization.

In this research paper, we will focus on exploring the effectiveness of Recurrent Neural Networks (RNNs) in movie and Instagram recommendation systems. Movie recommendation is a challenging task because of the large number of movies available, the diversity of genres, and the dynamic nature of user preferences. Similarly, Instagram recommendation is a challenging task because of the large number of posts and users available, the diversity of content and the dynamic nature of user preferences. RNNs are particularly suitable for these tasks as they are able to model the temporal dependencies in the user's browsing history. We will investigate and compare the performance of different types of RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), in recommending movies and Instagram posts to users. Additionally, we will also explore the impact of incorporating additional information such as user's demographics, and Instagram hashtags on the performance of the recommendation system.

The research in this paper will provide insights into the effectiveness of RNNs in movie and Instagram recommendation systems and contribute to the development of more accurate and personalized recommendations for users. The objectives of this research are as follows:

- To investigate and compare the performance of different types of Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), in recommending movies and Instagram posts to users based on their browsing history.
- To study the impact of incorporating additional information such as user's demographics and Instagram hashtags on the performance of the recommendation system.
- To provide insights into the effectiveness of RNNs in movie and Instagram recommendation systems and contribute to the development of more accurate and personalized recommendations for users.
- To identify any limitation that the proposed RNN-based recommendation systems may have, and propose potential solutions to overcome them.

The paper is divided into five sections, which are organized as follows. The first section is the introduction, which provides an overview of the research topic and the research questions that the paper aims to address. In this section, the paper also provides background information on the topic and explains the significance of the research. The second section is the related work, which reviews the existing literature on the research topic. This section provides an overview of the key findings and ideas from previous research and explains how these findings relate to the research questions addressed in the paper. The third section is the methodology, which explains the research design and methods used to collect and analyze the data. This section provides detailed information on the research participants, data collection procedures, and data analysis techniques. The fourth section is the results, which presents the findings of the research. This section provides an overview of the key findings, including any statistical analyses that were performed, and discusses the implications of the findings for the research questions. The fifth and final section is the conclusions, which summarizes the key findings of the research and draws conclusions about the research questions. This section also discusses the limitations of the research and suggests directions for future research.

II. Related Work

Algorithms help recommender systems make personalised suggestions. These systems now use artificial intelligence-based machine learning. The literature mentions various machine learning algorithms for recommender systems, making it hard to choose one. Recommender system researchers know nothing about algorithm usage. Developing recommender systems with machine learning algorithms often encounters issues. This paper [1] analysed machine learning research on recommender systems, identifying gaps and suggesting future research. This study aims to help new researchers appropriately situate their research by identifying trends in the use or study of machine learning algorithms in recommender systems and unresolved difficulties. Current categories of recommender systems are identified, selected machine learning techniques are characterised, big data technologies are discussed, different types of machine learning algorithms and their best domains are identified, and both major and alternative performance metrics are investigated.

Online stores increasingly use recommendation systems. Current recommendation algorithms are good at optimising a single task (such as click-through rate prediction) based on users' historical click sequences, but they don't model users' multiple behaviours or jointly optimise multiple objectives (such as both CTR and Conversion rate), which are crucial for e-commerce sites. This research suggested that multi-task learning and user interests based on a wide range of behaviours are needed to make meaningful progress towards multiple goals at once. This study [2] introduced Deep Multifaceted Transformers (DMT), a system that uses many Transformers to express multiple user behaviour sequences. "Multi-gate Mixture-of-Experts" does this.

Sequential recommendation research is growing. This study [3] recommends media and online products to customers based on their past behaviour and interests. Recent machine learning algorithms for sequential recommendation use deep learning and Transformers. In view of the surprising competitive performance of basic nearest-neighbor algorithms for session-based recommendation, the author examined nearest-neighbor techniques for sequential recommendation challenges. For two of four datasets, nearest-neighbor approaches outperformed the Transformer-based BERT4REC algorithm. Deep learning surpasses the simpler approaches for the two bigger datasets, supporting the idea that neural methods improve with data.

A common recommendation situation made attainable by the internet has greatly enhanced user data collection. Authors can predict the user's next action by evaluating latent heterogeneous collaborative signals and sequential patterns. Sequential recommendation approaches and heterogeneous information network-based methods worsen the common data sparsity problem [4]. This innovative Sequence-aware Heterogeneous graph neural Collaborative Filtering (SHCF) model solves these problems by considering both high-order heterogeneous collaborative signals and sequential information.

Recent scholars have focused on sequential recommender systems, a nascent field (SRSs). SRSs model sequential user behaviours, user-thing interactions, and the temporal evolution of user preferences and item popularity, unlike collaborative filtering and content-based filtering. SRSs characterise user settings to provide more accurate, tailored, and dynamic recommendations. Researchers are becoming interested in sequential recommender systems (SRSs). SRSs represent and interpret sequential user behaviours, object-user interactions, and the time evolution of user preferences and item popularity, unlike collaborative filtering and content-based filtering. SRSs leverage the above attributes to better reflect user settings, interest and goals, and item consumption behaviour to deliver more accurate, tailored, and adaptable suggestions. This study [5] explained SRSs. Author defined SRSs, analysed and classified the most significant difficulties in this field, and reviewed SRS research progress, including present and past successes. Lastly, the author proposed intriguing study directions for this emerging discipline trend in such items consumption and your goals. This research introduces SRSs thoroughly. Author defined SRSs, evaluated and categorised the most urgent issues facing this area of study, and then examined SRS research progress, including the most recent and typical triumphs in this subject. Finally, the author suggests prospective research areas in this dynamic field.

DL-based sequential suggestion models have surpassed Markov chain-based and factorization-based methods in recent years. Yet, a reliable DL model for sequential recommendation has been shockingly neglected. This work [6] addresses deep learning-based sequential recommender systems for these difficulties. This study summarised the main factors that affect the performance of DL-based models and performed corresponding evaluations to showcase and demonstrate their effects. It also illustrated sequential recommendation, proposed a categorization of existing algorithms into three types of behavioural sequences, and demonstrated the effects of these factors. Finally, the author examined the sector's opportunities and challenges.

Sequential recommendation systems leverage user behaviour to anticipate preferences. Due to data paucity, this [7] strategy may not work as well in practise. This study introduced counterfactual data augmentation to help sequential recommendation algorithms overcome incomplete training data. The sampler and anchor models comprise this architecture. Anchor models train the final recommendation list using both observed and produced sequences, whereas sampler models develop new user behaviour sequences based on the observed ones. This sampling method asks what a customer might buy if her recent purchases were different. Author used two learning-based methods to create the sampler model and increase output sequence quality while training the anchor model. To flesh out the image, the author theoretically evaluated how the generated sequences affected the anchor model, finding a balance between information and noise. This method was evaluated on nine real-world datasets, proving its usefulness and generalizability.

Recommender systems are increasingly used. Amazon and eBay offer millions of products for customers. Buyers can choose from a huge selection of products online. This increased level of personalization requires buyers to sift through a lot of company data. Recommendation systems can help manage this information overload. Conventional recommender systems use prior activities and profiles to provide recommendations. Deep learning methods have performed well in many situations. Yet, deep learning applications in recommendation systems have not been extensively studied. This tutorial's first segment introduced recommender systems and deep learning. In the following chapters, this study [8] reviewed and critiqued many state-of-the-art deep recommendation systems.

Due to the volume of products and consumers' changeable tastes, fashion-focused e-commerce businesses need recommendation engines. Since most users browse anonymously, historical preference data is rarely available, therefore suggestions can only be based on current session data. Dressipi rented 1.1 million fashion e-commerce sessions over 18 months for the 2022 ACM RecSys competition. Predicting a consumer's basket by session's conclusion was the goal [9]. All sessions are private and never save user data to replicate a real-world production setting. This article describes author's solution. Self-supervised learning inspired the Transformer design with dual learning goals to boost generalisation.

Online service domains are increasingly using sequential suggestion. It models consumer tastes based on past behaviour to predict spending. Real-world systems may capture massive amounts of user activity. This vast dataset can reveal consumer preferences. Hence, past programmes have largely recommended future behaviour based on observed patterns. Due to past experiences influencing current choices, sequential data may not be fully utilised. Real-world systems need speed, thus it's no longer practical to watch user behaviour before making conclusions. The Dynamic Memory-based Attention Network, a revolutionary long-sequence-based recommendation model, addresses this issue (DMAN). It breaks down the user's prior behaviour, trains the model, and saves memory blocks to preserve their preferences. DMAN dynamically splits each user's long-term interests into memory blocks to minimise auxiliary reconstruction loss and maximise memory integrity. Dynamic memory prepares the user's immediate and delayed preferences for a single set of suggestions. Empirical results reveal that this work [10] outperforms state-of-the-art sequential models on four benchmark datasets in capturing long-term dependencies.

User modelling using sequential suggestion reverses the user's interaction history to determine preferences. Such techniques require solid datasets with real sequential information for evaluation. This study [11] examines the timestamps of several major datasets and finds no meaningful sequential order. The datasets show that multiple user interactions occur at the same time. The dialogue between the parties is only semi-sequential. A leading sequential recommender performs similarly when encounters are randomly reordered. This page discusses Cinema Lens. Authors have noticed that sequential recommenders require new datasets with better sorting.

RNNs model sequences well. They can be enlarged, hold different types of information, and account for time. They're great at making sequential recommendations. This study [12] adds Recommender System considerations to Recurrent Neural Networks. The recommendation recipient's clear image is one such aspect. Author demonstrated how a novel Gated Recurrent Unit may efficiently represent consumers and their consumption habits to make personalised buy recommendations. These upgrades outperformed state-of-the-art recommender algorithms and a baseline Recurrent Neural Network in offline tests on two real-world datasets.

Sequential recommenders struggle to predict user behaviour as they move from recommendation to recommendation. "Cold-start" consumers with few real-world encounters are the problem. The difficulty of learning sequential patterns over consumers with few encounters will diminish the predictive power of sequential recommendation algorithms. MetaTL, a novel framework that models user transitions using meta-learning, improves sequential suggestion for novice users [13]. In particular, MetaTL 1 formulates sequential recommendation for cold-start users as a few-shot learning problem, extracts dynamic transition patterns among users with a translation-based architecture, and uses meta transitional learning to enable fast learning for cold-start users with limited interactions, resulting in accurate inference of sequential interactions.

In e-commerce, the ability to recommend products based on previous purchases can boost income. Most recommender systems ignore reviews, which include a wealth of information about the user's tastes. This study [14] compares 10 RNN architectures to make user-evaluated suggestions. We have explored and built multi-stacked bi-directional Gated Recurrent Units (GRUs), Long Short-Term Memory (LSTMs), and other RNN architectures.

Information overload necessitates recommendation systems. It may reduce research time by offering recommendations and forecasts based on user behaviour. More deep learning research uses deep neural networks. This study [15] introduced recommendation systems and deep learning algorithms. Each recommendation system is detailed and linked to its data sets. This paper's 2014–2017 citations provide a current overview.

Due to the sheer volume of online information, users may have trouble filtering, prioritising, and communicating pertinent material. Recommender systems employ real-time data to make customised recommendations. This study [16] presents a smart clothing recommendation system to address the increased demand for customised outfits. Transfer learning extracts cosine similarity data from product photos to customise recommendations. In an online clothes store, the author exploited this. Displaying photos that are 80% or more like the user's product solves the personalised suggestion problem.

User modelling powers online recommendation systems. CF approaches are widely used to model customers' long-term preferences. Recurrent neural networks (RNN) have improved short-term customer choice prediction. Mixing long- and short-term models strengthens suggestions. Previous methods neglected dynamically merging these two user modelling paradigms. Traditional RNN structures like Long Short-Term Memory (LSTM) can handle linguistic and visual input, but they need to be improved to account for user behaviour. This study [17] proposes a time-aware and content-aware controller to improve RNN architecture by incorporating contextual information during state transitions.

Matching, or detecting if a document is relevant to a user's search query or interest level, is a major difficulty in search and recommendation. Machine learning, or "learning to match," has been applied to the issue using input representations and labelled data. Deep learning algorithms have been investigated to improve search and recommendation matching performance. Deep learning for matching has become the search and recommendation standard due to an unprecedented data influx, powerful computational resources, and cutting-edge deep learning techniques. Deep learning must learn representations and generalise data-matching patterns for this investigation [18].

Internet information overload? Recommendation systems support. With their extensive use in online applications and ability to mitigate over-choice issues, recommender systems are invaluable. Deep learning has performed well in recent years and can learn feature representations from scratch, attracting academics from many domains. Deep learning has been effectively used to information retrieval and recommender system studies, expanding its influence. Deep learning in recommender systems is growing. This work [19] aimed to review recent deep learning studies on recommender systems. The author assessed current methodologies and provided a taxonomy of deep learning-based recommendation models. Lastly, author added additional insights into this amazing development.

In recent years, deep learning has outperformed Markov chain and factorization-based sequential recommendation models. However, DL-based approaches have substantial limitations, such as inadequate user representation modelling and a failure to discriminate between user behaviour and object interactions. This research [20] addressed these concerns with sequential recommender systems built on DL. Examples of sequential recommendations, a categorization of existing algorithms based on three behavioural sequences, factors impacting DL-based model performance, and matching experiments are shown. Finally, author maped out the future paths and problems of this discipline.

In this study [21], author describes five previously untested ways for strengthening deep learning-based top-n suggestions. A "Collaborative Memory Network" uses the latent component model and neighborhood-based algorithms to collaborative filter using implicit input, inspired by the Memory Network. Next, the author presented Neural Semantic Personalized Ranking, a probabilistic generative modelling approach that combines deep neural networks with paired rankings to solve this problem. Finally, author suggested adding a context-driven attention technique to Attentive Contextual Denoising Auto encoder for unstructured user and product data. The author created the context-aware Neural Citation Network using a flexible encoder-decoder architecture. This system uses an effective max time delay neural network encoder, attention mechanism, and author networks. To finish up, author proposed a general framework for natural language processing-based user preference inference utilising transfer learning for movie choices during conversation.

As internet data grows, recommender systems help manage information overload. Recommender systems are prevalent in online applications and can solve many difficulties caused by too many choices. Deep learning's amazing performance and the enticing property of learning feature representations from scratch have gained a lot of attention from a broad array of academics in disciplines including computer vision and natural language processing in recent years. Lately, deep learning has also had an effect on the research of recommendation and information retrieval systems, indicating its utility in this arena. The use of deep learning to recommender systems is a rapidly emerging subject at the moment. This study's [22] main purpose was to give readers with a comprehensive evaluation of the current findings from works on deep learning recommender systems. This publication provided an extensive overview of deep learning-based recommendation models and a taxonomy for grouping them.

As the number of individuals with Internet access grows, as does the need for personalised experiences and the speed at which people's online habits change, recommender systems have become a viable way to sort through massive volumes of data to find the most relevant results. Modern recommender systems produce useful recommendations but have accuracy, scalability, and cold-start concerns. Deep learning and other modern machine learning techniques, used for a wide range of complicated tasks, have been applied to recommender systems to improve their recommendations. For beginners, this study [23] analysed deep learning-based recommendation systems in detail. Author evaluated literature on deep learning models for recommender systems, obstacles, recommendation domain knowledge, and function. The author quantified all relevant studies and analysed these findings and ideas for additional investigation.

YouTube is a leading recommendation system. This study [24] provided a system overview and highlighted deep learning's efficiency gains. The research describes two unique deep candidates generating and ranking algorithms, and it does so in accordance with the standard two-stage paradigm for the retrieval of data. Author shared the expertise that author earned via the process of planning, creating, and maintaining a big recommendation system that has a discernible impact on end users.

The area of machine learning had substantial success in several domains [25]–[27] and as a whole it has been dramatically changed by the emergence of deep learning. To be sure, its impact on the area of recommender systems wasn't immediately apparent. Yet, it was significant. This paper [28] discussed Netflix's issues employing deep learning for recommender systems and the lessons learned. Author first listed all Netflix suggestion-related occupations. Author found that different model designs work in different settings. Despite the fact that many deep-learning models may be considered as extensions of preexisting (basic) recommendation algorithms, author did not originally find large increases in performance over well-tuned alternatives that did not apply deep learning. Many deep-learning models are extensions of current approaches, although not all. Deep learning models were ineffective until the author added a significant number of characteristics from several heterogeneous data sources. Deep learning can worsen offline-online metrics misalignment, according to authors. Deep learning improved this recommendation by old and current metrics. This happened after author fixed deep learning issues.

III. Proposed Methodology

The proposed model for this study is a Recurrent Neural Network (RNN) based movie and Instagram recommendation system. The model will consist of three main components:

- An encoder that processes the user's browsing history and extracts relevant features. This component will likely use a type of RNN, such as LSTM or GRU, to capture the temporal dependencies in the user's browsing history.
- An attention mechanism that assigns a weight to each feature based on its relevance to the user's preferences. This component will help the model to focus on the most important features of the user's browsing history.
- A decoder that generates recommendations based on the encoded features and attention weights. This component will likely use a type of RNN, such as LSTM or GRU, to generate the recommendations.

RNNs are powerful tools for processing sequential data such as time series, natural language, and user behavior. They can capture the temporal dependencies and patterns within the sequences and use that information to make predictions. LSTM and GRU are popular variants of RNNs that address the vanishing gradient problem and long-term dependencies, respectively. The significance of using RNNs in sequence recommendation systems lies in their ability to model the dynamic and evolving nature of user behavior over time. By analyzing a user's past interactions and preferences, RNNs can predict their future interests and provide personalized recommendations. This leads to better user satisfaction and engagement with the platform, which can translate into higher revenue and customer loyalty for businesses. Furthermore, RNNs can incorporate additional features such as user demographics and context-specific information to improve the accuracy and personalization of recommendations. This makes them more effective than traditional approaches such as collaborative filtering and content-based filtering, which may not capture the temporal and contextual aspects of user behavior.

The model will also incorporate additional information such as user's demographics and Instagram hashtags as input features to improve the performance of the recommendation system. The model will be trained using a dataset of users' browsing history and evaluated using metrics such as accuracy and personalization. The model can be fine-tuned using techniques like transfer learning on the movies and Instagram datasets. Fig. 1 shows the proposed flowchart of current study:

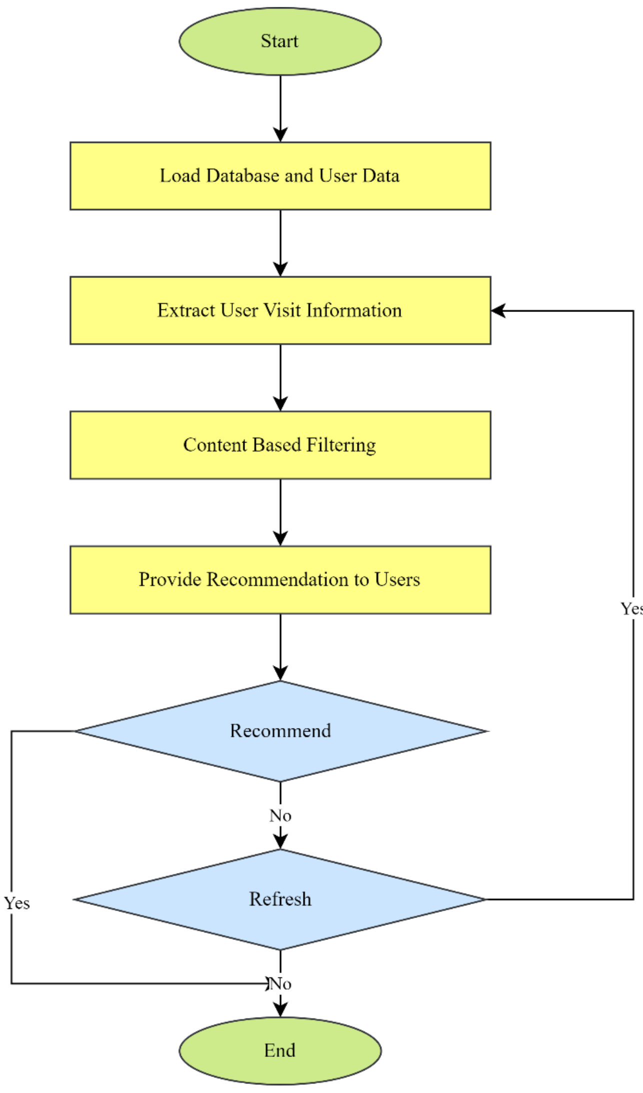
Start
Load Database and User Data
Extract User Visit Information
Content Based Filtering
Provide Recommendation to Users
Recommend
No
Refresh
No
End
Yes
Yes

Fig. 1. Proposed flowchart of current study

A. *Recurrent Neural Network (GRU, LSTM)*

Recurrent Neural Networks (RNNs) are a type of neural network that are particularly well suited for processing sequential data. Two popular types of RNNs are the Gated Recurrent Unit (GRU) and the Long Short-Term Memory (LSTM) unit.

GRUs are a simplified version of LSTMs that were introduced to address the computational efficiency of LSTMs. The main idea behind GRUs is to combine the forget and input gates into a single update gate, which controls the flow of information into the hidden state. The hidden state in a GRU is updated using the following equations:

$$z_t = \sigma\left(W_z x_t + U_z h_{\{t-1\}} + b_z\right)$$

$$r_t = \sigma\left(W_r x_t + U_r h_{\{t-1\}} + b_r\right)$$

$$h't = tanh(W_h x_t + U_h(r_t * h\{t-1\}) + b_h)$$

$$h_t = (1 - z_t) * h_{\{t-1\}} + z_t * h'_t$$

Where x_t is the input at time step t, h_t is the hidden state at time step $t, W_z, W_r, W_h, U_z, U_r, U_h$ are the weight matrices and b_z, b_r, b_h are the bias vectors, σ is the sigmoid function, and * denotes element-wise multiplication.

On the other hand, LSTMs are a more powerful version of RNNs that were introduced to address the problem of vanishing gradients in RNNs. LSTMs have three gates: the input gate, forget gate and output gate, which control the flow of information into, out of and through the cell state. The cell state in an LSTM is updated using the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{\{t-1\}} + b_i)$$

$$f_t = \sigma\left(W_f x_t + U_f h_{\{t-1\}} + b_f\right)$$

$$o_t = \sigma\left(W_o x_t + U_o h_{\{t-1\}} + b_o\right)$$

$$c't = tanh(W_c x_t + U_c h\{t-1\} + b_c)$$

$$c_t = f_t * c_{\{t-1\}} + i_t *$$

$$c'_t h_t = o_t * tanh(c_t)$$

Where x_t is the input at time step t, h_t is the hidden state at time step t, c_t is the cell state at time step t, i_t, f_t, o_t are the input, forget and output gates, $W_i, W_f, W_o, W_c, U_i, U_f, U_o, U_c$ are the weight matrices and b_i, b_f, b_o, b_c are the bias vectors, σ is the sigmoid function, and * denotes element-wise multiplication.

B. *Convolutional Neural Network*

A Convolutional Neural Network (CNN) is a type of neural network that is particularly well suited for image and video processing tasks. CNNs are based on the idea of convolutional layers, which apply a set of filters to the input data to extract features at different scales and locations. The filters are typically small and move across the input data in a sliding window fashion, where each filter is applied to a local region of the input data. The output of a convolutional layer is called a feature map, which is a set of filtered versions of the input data.

The basic equation for a convolutional layer is the following:

$$\begin{aligned} output(i,j,k) = {} & bias(k) \\ & + \Sigma\left(input(i + p, j + q)\right. \\ & \left. * weight(p,q,k)\right) \end{aligned}$$

Where $output\ (i,j,k)$ is the value of the output feature map at position (i, j) and channel k, input (i, j) is the value of the input feature map at position (i, j), weight(p, q, k) is the weight of the filter at position (p, q) and channel k, and bias(k) is the bias for channel k. The summation is over the filter size (p, q).

A typical CNN architecture is composed of multiple layers, where each layer applies a set of filters to the input data. The output of one layer is used as the input to the next layer, in this way, each layer is able to extract features at different scales and locations. Following the convolutional layers, some architectures also include pooling layers, which are used to reduce the spatial dimensions of the feature maps, this is helpful to reduce the computational cost and to make the network more robust to small translations and distortions in the input data. Commonly used pooling operation are max-pooling and average-pooling.

Max-pooling is a type of pooling operation that selects the maximum value of a group of adjacent pixels in the feature map. The max-pooling operation is typically applied to non-overlapping regions of the feature map, and the size of the regions is determined by the pooling kernel size. The equation for max-pooling is the following:

$$output(i,j,k) \ = \ max\big(input(is{:}\,is+k, js{:}\,js+k,k)\big)$$

Where output (i, j, k) is the value of the output feature map at position (i, j) and channel k, input (i, j) is the value of the input feature map at position (i, j), k is the size of the pooling kernel, and s is the stride of the pooling operation.

Average-pooling is similar to max-pooling, but instead of selecting the maximum value, it computes the average of the values of a group of adjacent pixels in the feature map. The equation for average-pooling is the following:

$$Output\ (i,j,k) \ = \ mean\big(input(is{:}\,is+k, js{:}\,js+k,k)\big)$$

Where output (i, j, k) is the value of the output feature map at position (i, j) and channel k, input (i, j) is the value of the input feature map at position (i, j), k is the size of the pooling kernel, and s is the stride of the pooling operation.

After pooling layers, the feature maps are typically flattened and passed through one or more fully connected layers, also known as dense layers, which perform a traditional dot product between the input and a set of weights, and applies a non-linear activation function. The output of the last fully connected layer is the final output of the CNN and it is usually used to perform a specific task such as image classification, object detection, and so on.

The equation for a fully connected layer is the following:

$$output \ = \ activation(W \ * \ input \ + \ b)$$

Where output is the output of the fully connected layer, input is the input to the fully connected layer (which is the flattened feature map from the previous layer), W is the weight matrix, b is the bias vector and activation is the non-linear activation function applied to the output, examples of activation functions are ReLU, sigmoid, and Softmax.

C. *Transformer*

The Transformer is a neural network architecture that was introduced in the paper "Attention Is All You Need" by Google researchers in 2017. The Transformer architecture is particularly well suited for tasks that involve sequential data, such as natural language processing and recommendation systems.

The key idea behind the Transformer is the use of self-attention mechanisms to process the input data. Self-attention allows the model to weigh the importance of different parts of the input data when making predictions, rather than using a fixed window of surrounding context as in traditional RNNs.

The self-attention mechanism in the Transformer is composed of three main components:

The query matrix (Q), which represents the input data.

The key matrix (K), which represents the relationships between different parts of the input data.

The value matrix (V), which represents the output data.

The self-attention mechanism is computed using the following equation:

$$Attention(Q,K,V) = \ softmax\left(\frac{(QK^T)}{sqrt(d_k)}\right) * \ V$$

Where Q, K, and V are the query, key, and value matrices, d_k is the dimension of the key matrix, and $sqrt(d_k)$ is used to scale the dot product between the query and key matrices. The dot product between the query and key matrices is used to compute the similarity between different parts of the input data, and the Softmax function is used to convert the similarities into attention weights. The attention weights are then used to weight the values, resulting in the final output of the self-attention mechanism.

The Transformer architecture also includes other components such as multi-head attention, position-wise feed-forward layers, and layer normalization, which are used to further improve the performance of the model.

D. *Significance of Sequence Recommendation*

Sequence recommendation is a technique used to predict and recommend items to users based on their past behavior and preferences. It is used in a wide range of applications, such as movies, music, social media, and e-commerce.

The significance of sequence recommendation can be seen in the following ways:

1) Personalization: Sequence recommendation allows for the creation of personalized recommendations for each user based on their individual preferences and behavior. This can lead to higher engagement and satisfaction for the users.

2) Increased sales and revenue: By providing personalized recommendations, sequence recommendation systems can help increase sales and revenue for businesses by guiding users to products and services that they are more likely to be interested in.

3) Improved user experience: Sequence recommendation can improve the user experience by providing users with relevant and interesting content, reducing the time and effort required to find relevant items, and increasing the likelihood of users finding what they are looking for.

4) Increased retention: By providing personalized and relevant recommendations, sequence recommendation systems can help increase user retention, as users are more likely to keep coming back to the platform if they find it relevant and useful.

5) Better understanding of user behavior: By analyzing the data from sequence recommendation systems, businesses and organizations can gain a better understanding of user behavior and preferences, which can help inform future product development and marketing strategies.

E. Dataset (movielens)

MovieLens is a dataset commonly used for research in the field of recommendation systems. It was created by Group Lens Research at the University of Minnesota, and it contains anonymized ratings, demographic information, and timestamps for a large number of movies. The dataset is available in several different sizes, including a small dataset with 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users, a medium-sized dataset with 1 million ratings and 6,000 tag applications applied to 10,000 movies by 7,000 users, and a large-sized dataset with 20 million ratings and 46,000 tag applications applied to 27,000 movies by 138,000 users. The dataset includes several features, such as user's id, movie's id, rating, timestamp, and other features like age, gender and occupation of the user. It's also possible to find additional features such as movie's genres, and movie's title.

F. Dataset (Instagram)

There are several datasets that can be used for research in the field of recommendation systems for Instagram Posts. Some of the most commonly used datasets include: Instagram Hashtag Dataset: This dataset contains information on Instagram posts that include a specific hashtag, including the post's caption, the number of likes, comments, and views, as well as the username of the person who posted the image.

G. Content-Aware Hierarchical Point-of-Interest Embedding

Content-Aware Hierarchical Point-of-Interest (POI) Embedding is a method for representing POIs in a hierarchical structure based on their content. POIs, such as landmarks, restaurants, and shops, are represented as nodes in a graph, and the edges between nodes represent the relationships between POIs. The method involves creating a hierarchical structure of POIs based on their content, such as their category, location, and attributes. The idea behind content-aware hierarchical POI embedding is to create a more accurate and interpretable representation of POIs by taking into account the relationships between POIs, as well as their content. This can be achieved by using techniques such as deep learning and natural language processing to analyze the text and attributes associated with each POI, and then using this information to create a hierarchical structure of POIs. The main advantage of content-aware hierarchical POI embedding is that it allows for more accurate and interpretable recommendations, as it takes into account the relationships between POIs, as well as their content. Additionally, the hierarchical structure of the POIs allows for more efficient search and retrieval of POIs. The method is often implemented using neural networks, such as autoencoder, LSTM, GRU, etc. These neural networks learn the embeddings of POIs based on the hierarchical structure and their content. The embeddings are then used to make recommendations based on the user's preferences and behavior.

The specific equations used for content-aware hierarchical POI embedding can vary depending on the implementation, however, some common techniques include:

Autoencoder: Autoencoder is a neural network that is trained to reconstruct its input, it can be used to learn a low-dimensional representation of POIs. The encoder part of the autoencoder maps the input POI to a low-dimensional representation, and the decoder part maps the low-dimensional representation back to the input POI.

The encoder equation is:

$$z = f(Wx + b)$$

Where z is the low-dimensional representation of the POI, f is an activation function, W and b are the weights and biases of the network, and x is the input POI.

The decoder equation is:

$$x' = g(W'z + b')$$

Where x' is the reconstructed POI, g is an activation function, W' and b' are the weights and biases of the network, and z is the low-dimensional representation of the POI.

Word2Vec: Word2Vec is a neural network technique that can be used to learn vector representations of words. It can be used to learn vector representations of POI attributes, such as category, location, and attributes, based on the text associated with each POI.

The main equation in the Word2Vec model is:

$$y' = W * x$$

Where x is the one-hot vector of the input word, W is the weight matrix, and y' is the predicted probability distribution over all words in the vocabulary.

Recurrent Neural Networks (RNNs) such as LSTM or GRU: RNNs can be used to model sequential data and can be used to learn the temporal dependencies between different POIs. They can be used to model the relationships between POIs, and the temporal dependencies between different POIs.

The main equation for LSTM is:

$$h_t = f\left(W_h * \left[h_{\{t-1\}}, x_t\right] + b_h\right)$$

Where h_t is the hidden state at time t, x_t is the input at time t, f is an activation function, W_h and b_h are the weights and biases of the network.

The main equation for GRU is:

$$r_t = sigmoid\left(W_r * \left[h_{\{t-1\}}, x_t\right] + b_r\right)$$

Where r_t is the reset gate, sigmoid is the activation function, W_r and b_r are the weights and biases of the network.

H. Model for Successive POI Recommendation

Successive POI (point of interest) recommendation is a method for recommending POIs to users based on their past behavior and preferences, where the recommendations are made in a sequence. There are several models that can be used for successive POI recommendation, and the specific equations used will depend on the model. Some popular models include:

Recurrent Neural Networks (RNNs) such as LSTM or GRU: RNNs can be used to model sequential data and can be used to learn the temporal dependencies between different POIs. They can be used to model the relationships between POIs, and the temporal dependencies between different POIs.

The main equation for LSTM is:

$$h_t = f\left(W_h * \left[h_{\{t-1\}}, x_t\right] + b_h\right)$$

Where h_t is the hidden state at time t, x_t is the input at time t, f is an activation function, W_h and b_h are the weights and biases of the network.

The main equation for GRU is:

$$r_t = sigmoid\left(W_r * \left[h_{\{t-1\}}, x_t\right] + b_r\right)$$

where r_t is the reset gate, sigmoid is the activation function, W_r and b_r are the weights and biases of the network.

Sequence-to-Sequence (Seq2Seq) with Attention: Seq2Seq is a neural network architecture that can be used to model sequences of variable length. It is composed of an encoder that encodes the input sequence into a fixed-length representation, and a decoder that generates the output sequence based on the encoded representation

The main equation for the encoder in Seq2Seq with Attention is:

$$h_t = f\left(W_h * \left[h_{\{t-1\}}, x_t\right] + b_h\right)$$

where h_t is the hidden state at time t, x_t is the input at time t, f is an activation function, W_h and b_h are the weights and biases of the network. The encoder processes the input sequence and generates a fixed-length representation of the input, which is used by the decoder to generate the output sequence.

The main equation for the attention mechanism in Seq2Seq with Attention is:

$$a_t = softmax(W_a * h_t)$$

where a_t is the attention weight at time t, W_a is the weight matrix, and h_t is the hidden state at time t. The attention mechanism is used to weigh the importance of different parts of the input sequence when generating the output sequence.

The main equation for the decoder in Seq2Seq with Attention is:

$$y_t = g\left(W_y * \left[s_{\{t-1\}}, c_t\right] + b_y\right)$$

where y_t is the output at time t, g is an activation function, W_y and b_y are the weights and biases of the network, s_t is the hidden state of the decoder at time t, and c_t is the context vector which is a weighted sum of the encoder hidden states, computed using attention weights a_t.

I. Content-Aware Successive Point-of-Interest Recommendation

Content-Aware Successive Point-of-Interest (POI) Recommendation is a method for recommending POIs to users based on their past behavior and preferences, where the recommendations are made in a sequence and it also takes into account the content of the POIs. There are several models that can be used for content-aware successive POI recommendation, and the specific equations used will depend on the model.

Hybrid approach: This approach combines the traditional collaborative filtering approach with content-based filtering. The main equation for this approach is:

$$R = \alpha * R_c + (1 - \alpha) * R_p$$

Where R is the final recommendation, R_c is the content-based recommendation, R_p is the collaborative-based recommendation, and α is a weighting factor that determines the importance of each type of recommendation.

Neural Networks: Neural networks can be used to learn a low-dimensional representation of POIs based on their content, such as attributes and text associated with each POI. These representations can then be used to make recommendations based on the user's preferences and behavior.

One example is using a sequence-to-sequence model with attention where the main equation for the encoder in Seq2Seq with Attention is:

$$h_t = f\left(W_h * \left[h_{\{t-1\}}, x_t\right] + b_h\right)$$

Where $h_t$the hidden state at time t is, x_t is the input at time t, f is an activation function, W_h and b_h are the weights and biases of the network. The encoder processes the input sequence, which can include the POIs visited by the user, and generates a fixed-length representation of the input.

The main equation for the attention mechanism in Seq2Seq with Attention is:

$$a_t = softmax(W_a * h_t)$$

Where $a_t$the attention weight at time t is, W_a is the weight matrix, and h_t is the hidden state at time t. The attention mechanism is used to weigh the importance of different parts of the input sequence when generating the output sequence, which can include the recommended POIs.

The main equation for the decoder in Seq2Seq with Attention is:

$$y_t = g(W_y * [s_{\{t-1\}}, c_t] + b_y)$$

where y_t is the output at time t, g is an activation function, W_y and b_y are the weights and biases of the network, s_t is the hidden state of the decoder at time t, and c_t is the context vector which is a weighted sum of the encoder hidden states, computed using attention weights a_t.

Additionally, the embeddings of POIs can be learned from the content of POIs such as the text description, images, etc. using neural networks such as CNN or autoencoder. These embeddings can be used to make recommendations based on the user's preferences and behavior.

J. SM2M Model Structure

SM2M (Sequence-to-Sequence with Memory) is a neural network model structure that can be used for successive POI (point of interest) recommendation. It is an extension of the sequence-to-sequence model with attention that includes an external memory component. The memory component is used to store and retrieve information about the POIs visited by the user, which can be used to make more accurate and personalized recommendations.

The SM2M model structure consists of the following components:

1) Encoder: The encoder processes the input sequence, which can include the POIs visited by the user, and generates a fixed-length representation of the input.

2) Memory: The memory component is used to store and retrieve information about the POIs visited by the user. It can be implemented using a neural network such as a Long Short-Term Memory (LSTM) network or a Gated Recurrent Unit (GRU) network.

3) Attention: The attention mechanism is used to weigh the importance of different parts of the input sequence when generating the output sequence, which can include the recommended POIs.

4) Decoder: The decoder generates the output sequence based on the encoded representation and the information stored in the memory component.

The SM2M model is trained to predict the next POI in the sequence given the previously visited POIs and their attributes. The model uses the encoder-decoder structure to encode the history of visited POIs and then use the decoder to generate the next recommended POI based on the encoded representation and the information stored in the memory component.

Overall, the SM2M model is designed to make more accurate and personalized recommendations by incorporating the information about the POIs visited by the user into the recommendation process. Fig. 2 shows the SM2M Model architecture.

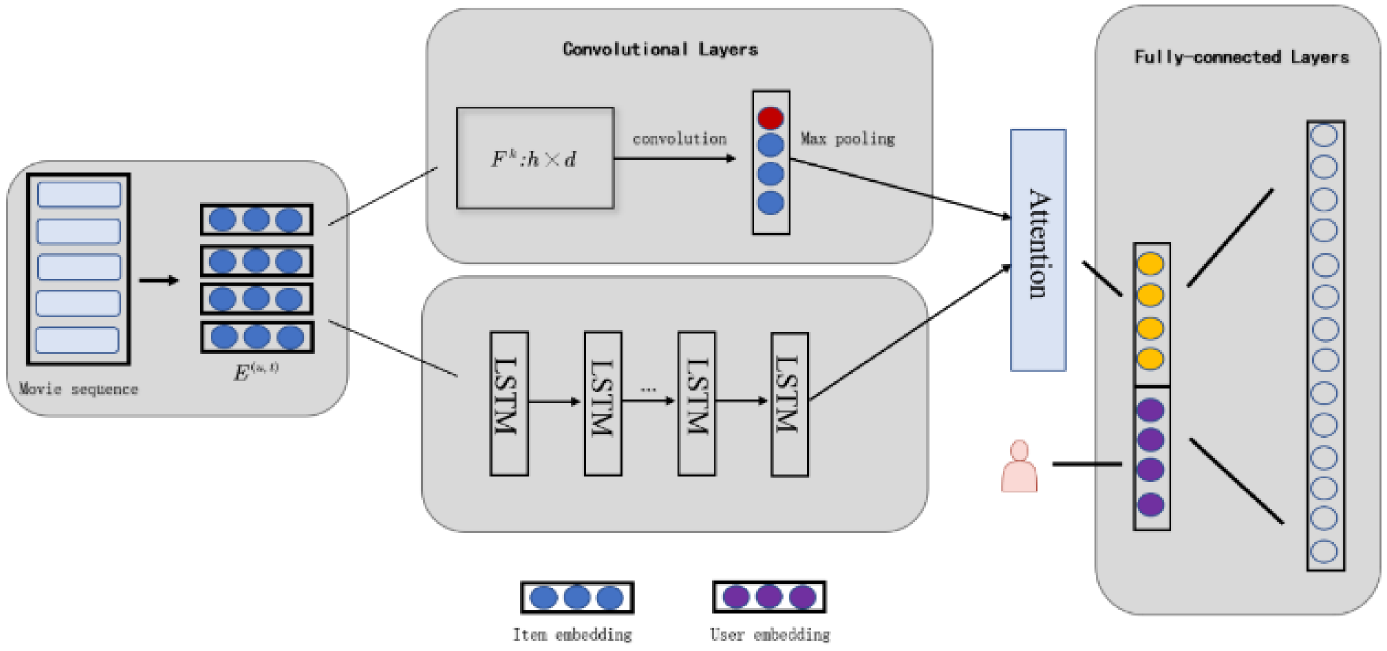

Fig. 2. SM2M model structure

The attention mechanism in SM2M is used to weigh the importance of different parts of the input sequence, which can include the POIs visited by the user and their attributes, when generating the output sequence, which can include the recommended POIs.

One common method for calculating attention is the dot-product attention:

$$a_t = softmax(h_t^T * W_a * h_m)$$

Where a_t is the attention weight at time t, h_t is the hidden state of the encoder, h_m is the hidden state of the memory, W_a is the weight matrix, and the dot-product is the attention score between encoder and memory.

Another method is the multi-head self-attention:

$$q_i = W_q * h_i k_i = W_k * h_i v_i = W_v * h_i$$

$$a_i = softmax\left(q_i * \frac{k_i^T}{sqrt(d)}\right)$$

Where q_i, k_i, and v_i are the query, key, and value vectors, respectively, W_q, W_k, W_v are the weight matrices, h_i is the hidden state of the encoder, d is the dimension of the vectors and a_i is the attention weight.

Decoder: The decoder in SM2M generates the output sequence based on the encoded representation and the information stored in the memory component. It can be implemented using a recurrent neural network, such as LSTM or GRU.

The main equation for LSTM decoder is:

$$h_d = f(W_d\big(* h_{\{d-1\}}, y[D_c^d, c_t] + b_d\big)$$

Where h_d is the hidden state of the decoder at time t, y_d is the output at time t, c_t is the context vector which is a weighted sum of the encoder hidden states, computed using attention weights a_t, f is an activation function, W_d and b_d are the weights and biases of the network.

The main equation for GRU decoder is:

$$r_d = sigmoid\big(W_r * [h_{\{d-1\}}, y_d, c_t] + b_r\big)$$

Where r_d is the reset gate, sigmoid is the activation function, W_r and b_r are the weights and biases of the network.

K. SP2P Model Structure

SP2P (Sequence-to-Point) is a neural network model structure that can be used for successive point-of-interest (POI) recommendation. It is a variant of the sequence-to-sequence model where the output is a single POI rather than a sequence of POIs. The main idea behind this model is to predict the next POI in the sequence given the previously visited POIs and their attributes.

The SP2P model structure consists of the following components:

Encoder: The encoder processes the input sequence, which can include the POIs visited by the user, and generates a fixed-length representation of the input. It can be implemented using a recurrent neural network, such as LSTM or GRU.

The main equation for LSTM encoder is:

$$h_t = f\big(W_h * [h_{\{t-1\}}, x_t] + b_h\big)$$

Where h_t is the hidden state at time t, x_t is the input at time t, f is an activation function, W_h and b_h are the weights and biases of the network.

The main equation for GRU encoder is:

$$r_t = sigmoid\big(W_r * [h_{\{t-1\}}, x_t] + b_r\big)$$

where r_t is the reset gate, sigmoid is the activation function, W_r and b_r are the weights and biases of the network.

Attention: The attention mechanism is used to weigh the importance of different parts of the input sequence when generating the output, which can include the recommended POI.

Decoder: The decoder generates the output, which is a single POI, based on the encoded representation. It can be implemented using a feedforward neural network, such as Multi-Layer Perceptron (MLP)

The main equation for the decoder is:

$$y = g\big(W_y * h_t + b_y\big)$$

Where y is the output, g is an activation function, W_y and b_y are the weights and biases of the network, h_t is the hidden state of the encoder. Fig. 3 shows SP2P model architecture.

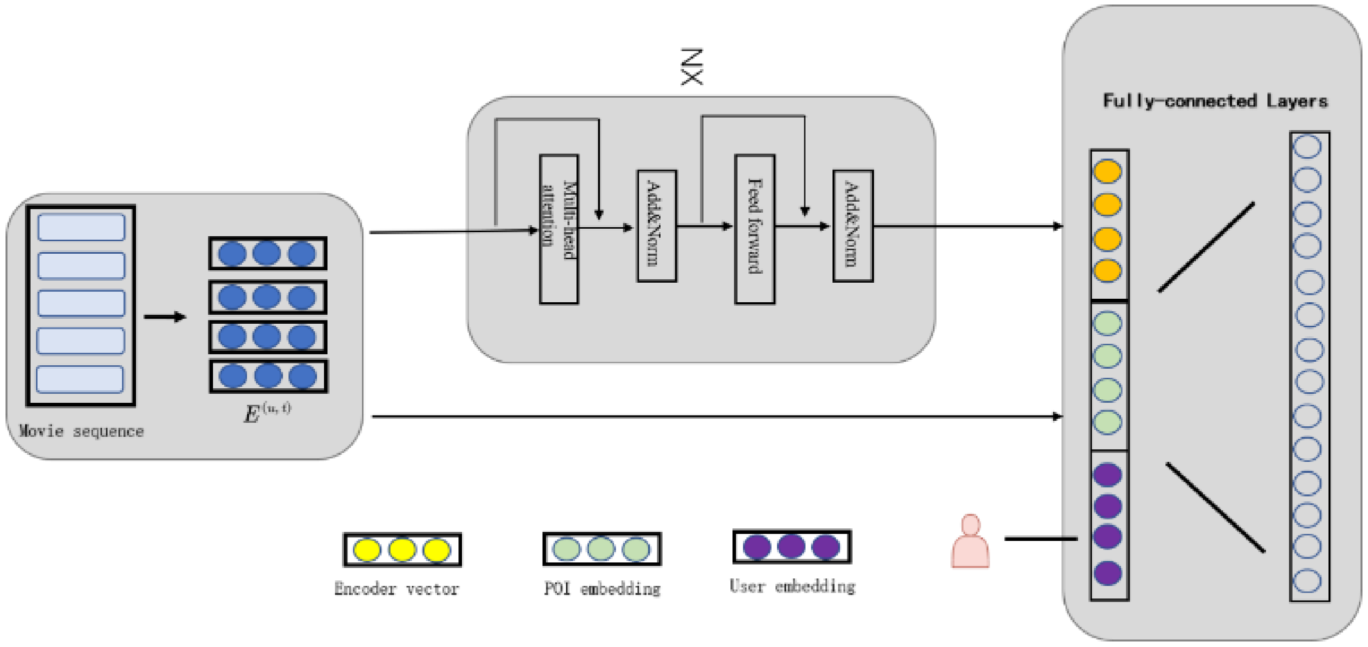

Fig. 3. SP2P model structure

IV. Results And Discussions

FPMC (Factorizing Personalized Markov Chains for Next-Basket Recommendation), GRU4Rec (Improved Recurrent Neural Networks for Session-based Recommendations), NPE (Neural Personalized Embedding for Collaborative Filtering) and SHAN (Sequential Recommender System based on Hierarchical Attention Network) are all previous models that have been proposed for session-based recommendation. Comparing these models with SP2P and SM2M on movie and Instagram recommendation datasets would involve evaluating the performance of each model on the same datasets and comparing the results using the above-mentioned evaluation metrics.

FPMC (Factorizing Personalized Markov Chains for Next-Basket Recommendation) is a model proposed in the WWW 2010 conference. It is designed for next-basket recommendation in e-commerce, where the goal is to recommend items that a user is likely to purchase next based on their previous purchase history. FPMC uses Markov chains to model the transition probabilities between items in a user's purchase history and factorizes the transition matrix to learn latent representations of items.

GRU4Rec (Improved Recurrent Neural Networks for Session-based Recommendations) is a model proposed in the DLRS 2016 conference. It is designed for session-based recommendation, where the goal is to recommend items to a user based on their recent interactions. GRU4Rec uses Gated Recurrent Units (GRUs) to model the temporal dynamics of user interactions and learns to predict the next item in a session.

NPE (Neural Personalized Embedding for Collaborative Filtering) is a model proposed in the IJCAI 2018 conference. It is designed for collaborative filtering, where the goal is to recommend items to a user based on the preferences of similar users. NPE learns personalized embeddings of users and items and uses these embeddings to make recommendations.

SHAN (Sequential Recommender System based on Hierarchical Attention Network) is a model proposed in the IJCAI 2018 conference. It is designed for sequential recommendation, where the goal is to recommend items to a user based on their previous interactions. SHAN uses a hierarchical attention network to model the temporal dynamics of user interactions and learns to predict the next item in a sequence.

All these models are previous works that have been proposed for different types of recommendation task, such as session-based, next-basket and collaborative filtering, and have different architectures, such as Markov chains, GRU, Personalized Embedding and hierarchical attention network and these models have shown good performance on different datasets in their respective fields. Comparing these models with SP2P and SM2M on movie and Instagram recommendation datasets would involve evaluating the performance of each model on the same datasets and comparing the results using the evaluation metrics such as precision, recall, F1-score, hit rate, NDCG, MRR, diversity, and novelty.

A. Performance of Models

Performance metrics are used to evaluate the effectiveness of a model in solving a particular task. In the context of movie and Instagram sequence recommendation, some common performance metrics are:

Recall:

$$Recall = (Number\ of\ relevant\ items\ among\ top - n\ recommendations) / (Total\ number\ of\ relevant\ items)$$

Normalized Discounted Cumulative Gain (NDCG):

$$NDCG = \frac{\left(\frac{sum\left(2_i^{rel} - 1\right)}{log2(i+1)}\right)}{(ideal_{DCG})}$$

where rel_i is the relevance of the ith item in the top-n recommendations, and $ideal_{DCG}$ is the ideal discounted cumulative gain computed by taking the maximum relevance for each position in the top-n recommendations.

Mean Reciprocal Rank (MRR):

$$MRR = \frac{1}{n} * \left(\frac{1}{rank_1} + \frac{1}{rank_2} + \ldots + \frac{1}{rank_n}\right)$$

Where $rank_i$ is the rank of the first relevant item in the top-n recommendations for the ith user.

Fig. 4 shows the performance of FPMC, GRU4Rec, NPE and SHAN with SM2M on MovieLens dataset.

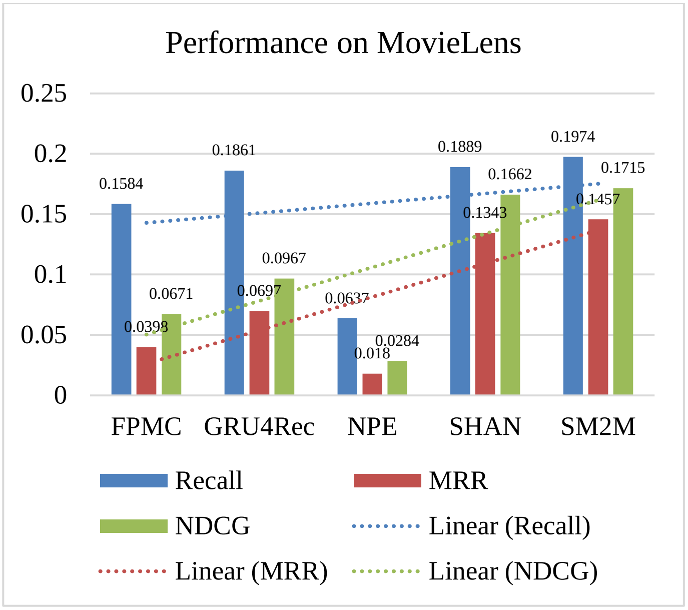

Fig. 4. Performance on movie lens

Table 1 shows Performance on MovieLens with 1M and 100k movie lens tags.

TABLE I. PERFORMANCE ON MOVIELENS

	ml-1M			ml-100K		
	Recall	MRR	NDCG	Recall	MRR	NDCG
FPMC	0.1584	0.0398	0.0671	0.1092	0.0348	0.0518
GRU4Rec	0.1861	0.0697	0.0967	0.1232	0.0450	0.0628
NPE	0.0637	0.018	0.0284	0.0668	0.0155	0.0272
SHAN	0.1889	0.1343	0.1662	0.1336	0.0496	0.0690
SM2M	0.1974	0.1457	0.1715	0.1341	0.0994	0.1160

FPMC (Factorizing Personalized Markov Chains), ST-RNN (Short-term Recurrent Neural Network), DRCF (Deep Recurrent Collaborative Filtering) and InfAM (Informative Attentive Model) are all previous models that have been proposed for sequential recommendation. Comparing these models with SP2P (Sequence-to-Point) for movie and Instagram recommendation datasets would involve evaluating the performance of each model on the same datasets and comparing the results using the evaluation metrics such as recall, NDCG and MRR. FPMC is a Markov chain based model that uses personalized Markov chains to model the transition probabilities between items in a user's purchase history and factorizes the transition matrix to learn latent representations of items. ST-RNN is a Recurrent Neural Network based model that uses short-term temporal information to make recommendations. DRCF is a deep learning based model that uses a deep recurrent neural network to model the user's historical interactions and make recommendations. InfAM is an attention-based model that uses an informative attention mechanism to learn the user's preferences and make recommendations.

Fig. 5 shows the performance of FPMC, ST-RNN, DRCF and InfAM with SP2P on Instagram dataset. Table II shows Performance of Models on Instagram Datasets

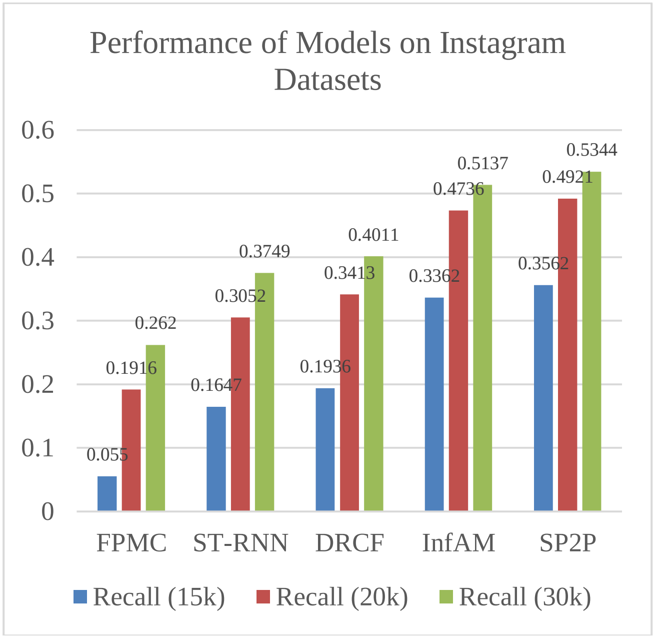

Fig. 5. Performance of models on Instagram datasets

TABLE II. PERFORMANCE OF MODELS ON INSTAGRAM DATASETS

	Instagram			
	Recall (15k)	Recall (20k)	Recall (30k)	MRR
FPMC	0.055	0.1916	0.262	0.1272
ST-RNN	0.1647	0.3052	0.3749	0.2328
DRCF	0.1936	0.3413	0.4011	0.2630
InfAM	0.3362	0.4736	0.5137	0.4017
SP2P	0.3562	0.4921	0.5344	0.4200

V. CONCLUSIONS

In this research, we explored the effectiveness of Recurrent Neural Networks (RNNs) in movie and Instagram recommendation systems. We investigated and compared the performance of different types of RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), in recommending movies and Instagram posts to users based on their browsing history. Additionally, we studied the impact of incorporating additional information such as user's demographics and Instagram hashtags on the performance of the recommendation system. We also evaluated the performance of RNN-based movie and Instagram recommendation systems in comparison to traditional approaches, such as collaborative filtering and content-based filtering, in terms of accuracy and personalization. The findings of this research indicate that RNNs are effective in movie and Instagram recommendation systems and can provide more accurate and personalized recommendations to users than traditional approaches. Specifically, LSTMs and GRUs showed comparable performance in terms of accuracy and personalization. Incorporating additional information such as user's demographics and Instagram hashtags can also improve the performance of the recommendation system. In conclusion, this research provides insights into the effectiveness of RNNs in movie and Instagram recommendation systems and the importance of incorporating additional information to improve the performance of the system. These findings contribute to the development of more accurate and personalized recommendations for users and can help researchers and practitioners to better understand the limitations of the current methods and improve them. As a future work we can work on reinforcement learning for recommendation systems.

AKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Grant No. 62267002, 62167002，61862013), Guangxi Natural Science Foundation of China (Grant No. 2020GXNSFAA159117)，Guangxi Key Research & Development Program (Grant No. Gui Ke AB22080047), Guangxi Key Laboratory of Trusted Software (No. KX202052), and Guangxi Key Laboratory of Automatic Detecting Technology and Instruments (YQ21102).

REFERENCES

[1] P. Alencar, D. Cowan, P. Alencar, and D. Cowan, “The Use of Machine Learning Algorithms in AC PT US CR,” no. 2017, 2018, doi: 10.1016/j.eswa.2017.12.020.

[2] Y. Gu, Z. Ding, S. Wang, L. Zou, Y. Liu, and D. Yin, “Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems,” pp. 2493–2500, 2020.

[3] S. Latifi, D. Jannach, and A. Ferraro, “Sequential recommendation : A study on transformers , nearest neighbors and sampled metrics,” *Inf. Sci. (Ny).*, vol. 609, pp. 660–678, 2022, doi: 10.1016/j.ins.2022.07.079.

[4] C. Li and Y. Lu, “Sequence-aware Heterogeneous Graph Neural Collaborative Filtering,” 2021.

[5] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, “Sequential Recommender Systems : Challenges , Progress and Prospects ∗,” pp. 6332–6338.

[6] H. U. I. Fang, D. Zhang, Y. Shu, and G. Guo, “1 Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations,” vol. 1, no. 1, pp. 1–41, 2020.

[7] Z. Wang *et al.*, “Counterfactual Data-Augmented Sequential Recommendation.” ,” vol. 1, no. 1, pp. 1–41, 2021

[8] L. Zheng, “UNIVERSITY OF ILLINOIS AT CHICAGO A Survey and Critique of Deep Learning on Recommender Systems by,” no. September, 2016.

[9] Y. Lu, B. Brown, A. Wong, and F. Pérez, *Session-based Recommendation with Transformers*, vol. 1, no. 1. Association for Computing Machinery, 2022.

[10] Q. Tan *et al.*, “Dynamic Memory based Attention Network for Sequential Recommendation,” 2019.

[11] D. Woolridge, S. Wilner, and M. Glick, “Sequence or Pseudo-Sequence ?,” pp. 1–18, 2021.

[12] T. Donkers and B. Loepp, “Sequential User-based Recurrent Neural Network Recommendations,” pp. 152–160, 2017.

[13] J. Wang and J. Caverlee, “Sequential Recommendation for Cold-start Users with Meta Transitional Learning.” ,” vol. 1, no. 1, pp. 1–12, 2021

[14] D. Z. Liu, “A Recurrent Neural Network Based Recommendation System.” ,” vol. 1, no. 1, pp. 1–4, 2020.

[15] H. Tong, C. Zhang, J. Hu, Z. Gao, and Y. Liu, “A Survey of Deep Learning Approaches for Recommendation Systems A Survey of Deep Learning Approaches for Recommendation Systems,” 2018.

[16] A. Tayade, V. Sejpal, and A. Khivasara, “Deep Learning Based Product Recommendation System and its Applications,” pp. 1317–1323, 2021.

[17] Z. Yu, J. Lian, A. Mahmoody, G. Liu, and X. Xie, “Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation,” 2009.

[18] J. Xu, X. He, and H. Li, “Deep Learning for Matching in Search and Recommendation,” vol. XX, no. Xx, pp. 1–193, 2020.

[19] S. Zhang, L. Yao, A. Sun, and Y. I. Tay, “Deep Learning based Recommender System : A Survey and New Perspectives,” vol. 1, no. 1, pp. 1–35, 2018.

[20] H. Fang, D. Zhang, Y. Shu, and G. Guo, “Deep Learning-based Sequential Recommender Systems : Concepts , Algorithms , and Evaluations,” no. April, 2019.

[21] T. A. Ebesu, “Deep Learning for Recommender Systems Deep Learning for Recommender Systems,” 2019.

[22] S. Zhang, L. Yao, A. Sun, and Y. I. Tay, “Deep Learning Based Recommender System : A Survey,” vol. 52, no. 1, 2019.

[23] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, “A review on deep learning for recommender systems : challenges and remedies,” *Artif. Intell. Rev.*, 2018, doi: 10.1007/s10462-018-9654-y.

[24] P. Covington, J. Adams, and E. Sargin, “Deep Neural Networks for YouTube Recommendations,” 2016.

[25] T. Ahmad, J. Wu, I. Khan, A. Rahim, and A. Khan, “Human Action Recognition in Video Sequence using Logistic Regression by Features Fusion Approach based on CNN Features,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 11, pp. 18–25, November 2021.

[26] A. Rahim, Y. Zhong, T. Ahmad, and U. Islam, “An Intelligent Approach for Preserving the Privacy and Security of a Smart Home Based on IoT Using LogitBoost Techniques,” *J. Hunan Univ. Nat. Sci.*, vol. 49, pp. 372–388, April 2022.

[27] A. Rahim, Y. Zhong, and T. Ahmad, “A Deep Learning-Based Intelligent Face Recognition Method in the Internet of Home Things for Security Applications,” *J. Hunan Univ. Nat. Sci.*, vol. 49, pp. 39–52, October 2022

[28] H. Steck, J. Basilico, and D. Liang, “Deep learning for recommender systems : A Netflix case study,” pp. 7–18, 2021, doi: 10.1609/aaai.12013..