國　立　中　央　大　學

資　訊　管　理　學　系

碩　士　論　文

Next Basket Recommendation with Basket Size Prediction

研 究 生：郭羿德

指導教授：陳彥良 博士

中 華 民 國　　113　　年　　6　　月

搭配籃子大小的下一個購物籃推薦

摘要

隨著電子商務的迅速發展，消費者雖然能獲得日益增多的商品選擇，不過這也同時導致他們面臨訊息過載的挑戰，而推薦系統的出現能有效緩解這個問題。推薦系統可以進一步細分為不同的推薦任務，其中，下一個購物籃推薦任務主要是從用戶的購物籃歷史序列資料中，捕捉用戶的購買偏好，進而為用戶推薦下一個購物籃的商品。

然而，過去的購物籃推薦方法大多僅在學習用戶的購買歷史後，直接向用戶推薦相同大小的下一個購物籃，卻忽略了不同用戶可能因消費習慣或經濟能力，而導致籃子大小有所不同。因此，本研究提出了一種搭配籃子大小的下一個購物籃推薦模型 NBR-WBS。此模型先是透過注意力機制與長短期記憶分別為購物籃與購物籃大小進行編碼，接著將兩者的嵌入一起輸入到 Transformer 中進行多任務學習，以獲得購物籃的最終狀態向量。隨後，我們將該向量輸入到各別組件中進行特定任務的預測，並根據兩項任務的預測結果，為用戶進行動態大小的購物籃推薦。在本研究中，我們對三個真實世界的資料集進行了實驗，結果顯示我們所提出的方法在推薦性能上優於其他在下一個購物籃推薦任務的現有方法。

關鍵詞：下一個購物籃推薦、多任務學習、Transformer、動態購物籃大小

Next Basket Recommendation with Basket Size Prediction

ABSTRACT

With the rapid development of e-commerce, consumers benefit from an increasing variety of products but also face the challenge of information overload. Recommendation systems can effectively address this issue by helping users filter through the abundance of choices. These systems include various tasks, such as session-based recommendation, sequential recommendation, and the next basket recommendation task. Among them, the next basket recommendation task aims to capture users' purchasing preferences from their historical basket sequences and recommend items for their next basket.

However, most of the previous basket recommendation methods merely recommend the same size basket based on the user's purchase history, ignoring that different users may have varying basket sizes due to consumption habits or economic capability. Therefore, in this study, we propose a next basket recommendation model that incorporates basket size prediction, named **NBR-WBS** (**N**ext **B**asket **R**ecommendation **w**ith **B**asket **S**ize). This model first encodes the baskets and basket sizes separately using an attention mechanism and Long Short-Term Memory (LSTM), respectively. The embeddings of both are then input into a Transformer for multi-task learning (MTL) to obtain the final state vector of the basket. Subsequently, we input this vector into specific components for task prediction. Based on the results of these two tasks, we recommend dynamically sized baskets to users. Our experiments on three real-world datasets demonstrate that our proposed method outperforms existing methods regarding recommendation performance for the next basket recommendation task.

Keywords: Next Basket Recommendation, Multi-task Learning, Transformer, Dynamic Basket Size

Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 Research Background

The proliferation of e-commerce has offered consumers a diverse range of product choices, but it has also brought the challenge of information overload. Faced with numerous products, consumers may feel overwhelmed and struggle to compare options, hindering purchase decisions and ultimately impacting customer satisfaction.

Recommendation systems use machine learning techniques to suggest personalized content, products, and services to users based on their preferences and behaviors. These systems effectively alleviate the problem of information overload and enhance the usability and attractiveness of platforms. Currently, recommendation systems are widely applied in various scenarios, such as movie recommendations [1], music recommendations [2], news recommendations [3], fashion recommendations [4], and travel recommendations [5].

The goal of recommendation systems is to provide personalized and valuable suggestions to help users more easily find content, products, or services of interest to them. To achieve this goal, researchers have developed various recommendation tasks, each designed for specific user behaviors and data characteristics. These tasks include session-based recommendation [6], [7], [8], sequential recommendation [9], [10], and next-basket recommendation [11], [12], [13]. The choice of these methods depends on different application scenarios and user needs. The following will explore the specific details and characteristics of these recommendation tasks.

Session-based recommendation (Figure 1) models user behavior based on clicks or purchases within a specific time frame. During this interval, the items that the user continuously interacts with the system form short sequences. This task assumes that items recently browsed or purchased by the user are more relevant to current interests. Therefore, in session-based recommendations, the user's past click sequences are used as input, focusing on recent sequences to predict the next item that interests the user. This approach does not overly rely on the user's historical preferences, thus allowing effective recommendations even without a user login.

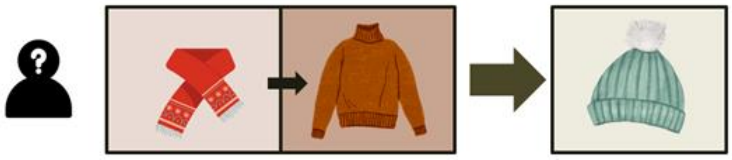


Figure 1: Session-based recommendation.

Sequential recommendation (Figure 2) models users based on their historical sequences or auxiliary information such as occupation and location to represent the user and captures user preferences from the interaction order of items in the user's past sequences. Utilize the user's past sequence data as input, and the model processes it to predict the next item the user may interact with.

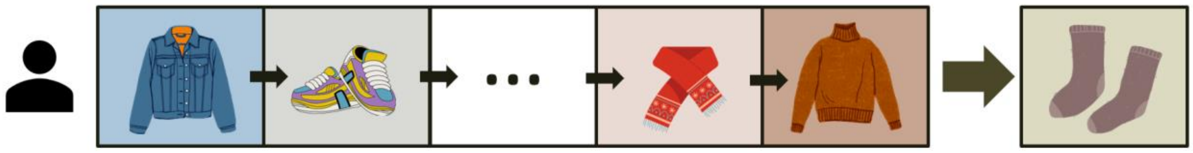


Figure 2: Sequential recommendation.

Next-basket recommendation (Figure 3) takes the user's historical purchase sequences as input, aiming to recommend the complete basket of items the user is most likely to purchase next. This method considers the relationships between items within the basket, including complementary or mutually exclusive items, to predict the user's next purchase.

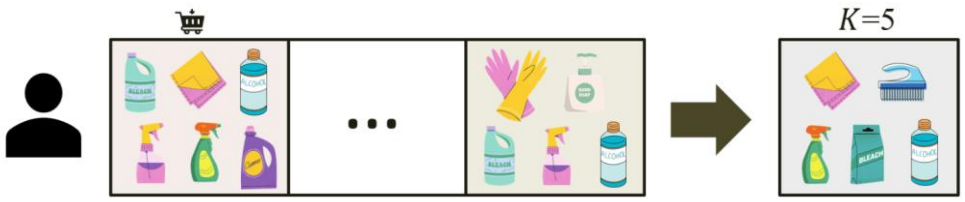


Figure 3: Next-basket recommendation.

Unlike session-based and sequential recommendations, which primarily analyze temporal order and historical behavior, next-basket recommendation focuses on the contents of a user's basket. The main difference is that next-basket recommendation deals with multiple items within a set, as items in a basket have no temporal order and do not have a ranking. Therefore, session-based and sequential recommendations cannot be directly applied to the task of next-basket recommendation.

1.2 Research Motivation

In this study, we aim to address the next-basket recommendation problem by more accurately predicting users' next baskets. Traditional methods for next-basket recommendation, such as collaborative filtering [14], personalized Markov chains [11], and hierarchical representation models [15], primarily focus on modeling behavior between adjacent baskets. However, these methods often fail to capture long-term user interests and may overlook that adjacent baskets can sometimes be unrelated.

With the development of deep learning, which addresses numerous challenges, it can efficiently extract features from high-dimensional spaces to obtain latent representations and effectively capture long-term dependencies between items from various sources such as context, auxiliary information, and different data types. Unlike traditional next-basket recommendation methods, which solely rely on recent information for recommendations and fail to address long-term dependencies between baskets, recent research has increasingly focused on the application of deep learning methods to address the next-basket recommendation task. Studies [13], [16], [17] have shown that using deep learning methods for next-basket recommendation can achieve excellent performance.

However, whether using traditional methods or employing deep learning techniques, in past studies on next-basket recommendation, the next-basket size K is mainly considered as fixed. In other words, they regard the next-basket size as a hyperparameter, employing search methods such as grid search or random search to find the optimal K value in the validation set. Therefore, these methods recommend the same K items as the next-basket size for all users. However, this standardized approach may fail to effectively tailor the recommended quantity of items for different users. Two primary reasons contribute to this challenge: Firstly, individual purchasing behaviors change over time, leading to variations in basket sizes. Secondly, consumers have diverse lifestyle habits, financial capacities, and commercial attributes, all of which influence basket sizes differently. For instance, as illustrated in Figure 4, professional cleaning companies tend to have larger basket sizes, purchasing more items compared to typical households.

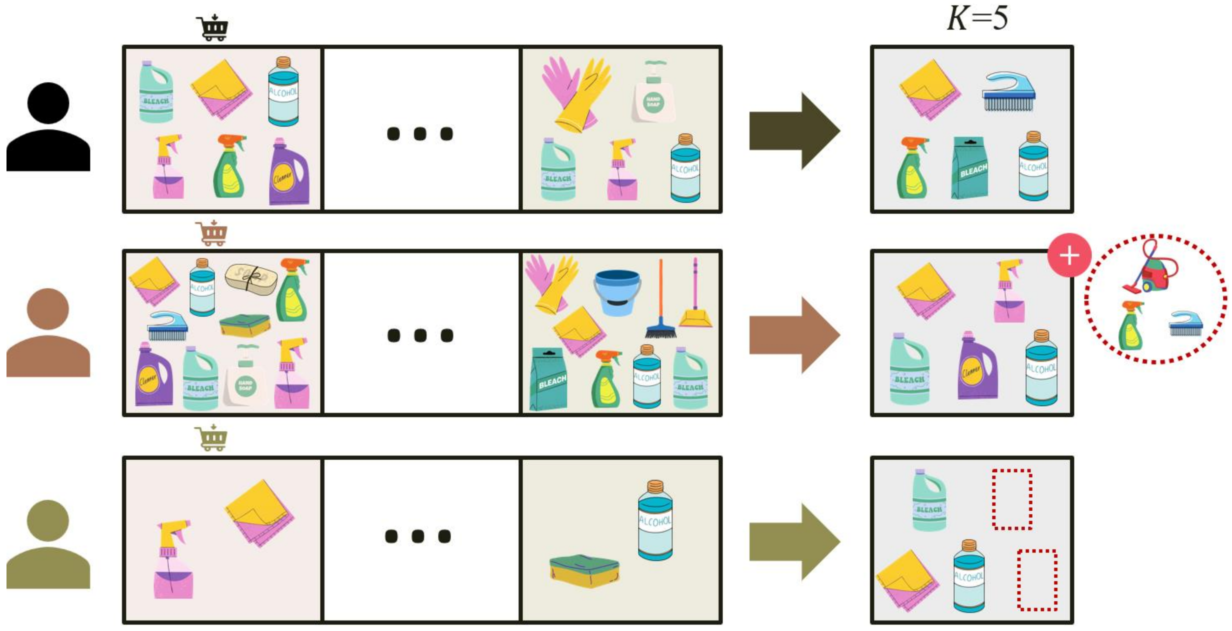


Figure 4: Different users have varying basket sizes due to factors such as consumption habits and financial capacities.

1.3 Research Purpose

To address the limitations of existing methods that treat basket size as fixed, this study proposes a novel approach that not only aims to predict the probability of items being included in a basket but also simultaneously predicts the size K of the next basket. This approach enables dynamic basket sizes, accommodating user variability and potentially even adapting to the same user's changing needs over time. This method offers two main benefits: firstly, it enhances prediction accuracy. In traditional methods, setting basket size K too high increases false positives (FP), which reduces precision. On the other hand, setting K too low increases false negatives (FN), which lowers recall. Thus, dynamically adjusting the basket size effectively resolves this issue. Secondly, the dynamic prediction method enables more precise real-time prediction and adjustment based on users' individual characteristics. This helps businesses better manage inventory, improve efficiency, and provide a better user experience, making it highly valuable in the fields of e-commerce and retail.

In this study, we propose a model named **NBR-WBS** (**N**ext **B**asket **R**ecommendation **w**ith **B**asket **S**ize) that dynamically adjusts the size of baskets. The model consists of three main modules:

- The embedding module includes a basket encoder using attention mechanisms for item importance and a basket-size encoder using Long Short-Term Memory (LSTM) for basket size embeddings.
- The basket preference module combines basket and size embeddings, processed by a shared Transformer for multi-task learning to generate the final state vector.
- The prediction module utilizes the final state vector of baskets for two tasks: predicting basket items and sizes using separate predictors. The results from these predictors are then combined to provide users with dynamically sized basket recommendations.

The contributions of this study are as follows:

- To the best of our knowledge, this study is the first to take into account basket size dynamically in the next basket recommendation.
- The proposed NBR-WBS model serves as a framework for multi-task learning. It not only forecasts the probability of items within the baskets but also predicts the size of users'

baskets at different time points.

- Extensive experiments are conducted on three real datasets, demonstrating that our proposed NBR-WBS outperforms other existing methods.

The remaining structure of this study is outlined as follows: Section 2 will review the literature. Section 3 will provide a detailed description of the proposed model architecture. Section 4 will present the experimental design and analyze the experimental results. Finally, Section 5 will conclude the research.

2. RELATED WORK

In recent years, next-basket recommendation has gained significant focus in the academic community, leading to the proposal of numerous methods to address basket recommendation challenges. This section aims to review relevant literature from the past, which can be categorized into the following aspects: traditional methods, Neural Network methods, and methods for finding optimal hyperparameters.

2.1 Traditional Methods

In this section, we first review a comprehensive overview of the relevant literature concerning traditional methods in next-basket recommendation.

2.1.1 Markov Chains

User's basket history is used as sequential input. Markov Chains (MC) learn probability matrices based on the state of the previous item or basket, and predict the next item or basket based on the transition matrix. For instance, Shani et al. [18] proposed a recommendation system based on Markov Decision Processes (MDPs), which generates the next item for recommendation based on the user's current basket state using transition functions. Gu et al. [19] integrated Markov Chains with time intervals to enhance temporal diversity for recommendation. He et al. [20] proposed combining Markov Chains with similarity models for recommendation in sparse data scenarios. However, Markov Chains assume that the future state depends only on the current state, thus not considering the long-term preferences of users' past purchasing behavior, and not designing different transition matrices for different users.

2.1.2 Collaborative Filtering Methods

Collaborative filtering methods can be further categorized into user-based collaborative filtering [21] and item-based collaborative filtering [22]. User-based collaborative filtering identifies similar users and their common preferences from the interaction history between users and items to predict the next item or basket. On the other hand, item-based collaborative filtering transforms items into item embeddings and calculates the similarity between embeddings to recommend similar items to users.

The most intuitive method for item embedding is to use one-hot encoding. However, this approach becomes extremely sparse and computationally intensive when dealing with large-scale product data. Item2Vec [23] effectively addresses this issue by mapping the original high-dimensional, sparse one-hot representation into a lower-dimensional vector space and measuring the similarity between two items by calculating the similarity between their low-dimensional vectors. In the past, SALAMPASIS et al. [24] conducted a study comparing different embedding methods on various recommendation tasks, including Item2Vec and Doc2Vec [25]. The study concluded that using Item2Vec for item embedding in next-basket recommendation yielded better results, whereas Doc2Vec performed better for next-item recommendation.

Additionally, a previous study [26] explored the use of matrix factorization (MF) techniques in recommendation systems, considering both user and item matrices. This study used MF to address the problem of data sparsity by decomposing the user-item interaction matrix into two lower-dimensional matrices representing users and items, thus inferring users' preferences for unrated items to achieve personalized recommendations.

In recent years, some studies [27], [28] have focused on extending matrix factorization methods with zero-shot learning algorithms to tackle the cold start problem in recommendation systems. However, despite the fact that collaborative filtering methods learn personalized user interests, many of them often ignore the sequential information of user behavior.

2.1.3 Hybrid Methods

Hybrid methods combine multiple recommendation algorithms to leverage their strengths and address individual limitations. For instance, Kumar et al. [29] combined item-based and user-based collaborative filtering to effectively address challenges faced by traditional collaborative filtering, including data sparsity and scalability. However, these methods often neglect the sequential nature of user behavior.

As mentioned earlier, Markov Chains (MC) capture sequence information through non-personalized transition matrices, while Matrix Factorization (MF) learns personalized interests for each user based on interaction data. To leverage the strengths of both methods, Rendle et al. [11] proposed Factorized Personalized Markov Chains (FPMC), which models a probability transition cube where each slice represents a personalized transition matrix for each user. This allows FPMC to recommend personalized items based on users' past basket sequences. However, the main drawback of FPMC lies in its reliance on linear combinations, making it challenging to capture complex relationships between items, resulting in suboptimal recommendation performance.

To address this issue, Wang et al. [15] introduced the Hierarchical Representation Model (HRM), which represents each user and item as vectors in continuous space and utilizes a two-layer structure to construct a hybrid representation of users and the last basket to predict the items in the next basket. However, by solely relying on the last basket in the sequence to represent users' short-term preferences, this method overlooks the global information of the entire sequence. As a result, it is less effective for capturing long-term user behavior, which limits its suitability for next-basket recommendation.

2.2 Neural Network Methods

Due to the challenges encountered by traditional methods in addressing this issue, researchers have begun to explore the use of deep learning techniques to enhance the performance of recommendation systems. In this section, we will focus on reviewing previous literature related to deep learning methods.

2.2.1 RNN-based Methods

With the advancement of deep learning, RNN methods model the entire sequence of baskets through their internal memory to learn users' long-term preferences. For example, Yu et al. [16] proposed the Dynamic Recurrent Basket Model (DREAM), which not only learns dynamic representations of users but also captures global sequence features among baskets. However, the items recommended by the above methods do not consider the correlation between items. Therefore, Le et al. [17] proposed a hierarchical network architecture called Beacon, which considers the correlation between paired items in the basket and uses RNN to capture the sequential association in the basket sequence. Bai et al. [30] introduced the Attribute-aware Neural Attention Model (ANAM), which utilizes RNN to model users' sequential behavior and shares attention weights between baskets at different levels to propagate user preferences to the next basket. Hu et al. [31] proposed a Sets2Sets model that utilizes an RNN encoder-decoder with attention mechanism to learn user representations from past interaction data, achieving state-of-the-art performance in the task of basket recommendation based on RNN models. However, traditional RNN models need to compute the features of the current data before processing the next data, making it unable to handle multiple data simultaneously and requiring significant computational resources for large-scale computations. Moreover, traditional RNN models have limited capability to capture dependency relationships in longer sequences. Therefore, many studies have opted to use the Transformer [32] for modeling.

2.2.2 Transformer

The Transformer architecture [32] has emerged as a powerful alternative to RNNs in recommendation systems. Unlike RNNs, which process data sequentially, Transformers leverage an encoder-decoder structure and self-attention mechanisms, enabling parallel processing and improved efficiency for capturing long-term dependencies in sequences. This advancement has inspired numerous Transformer-based methods for next-basket recommendation. Inspired by BERT, Yang et al. [33] propose the IERT model, which first pre-trains a Transformer module to generate contextual item representations, and then fine-tunes the parameters for specific recommendation tasks. Li et al. [34] introduced the BTBR model, a bidirectional Transformer approach that focuses on learning item correlations within the basket to enhance recommendation accuracy. Additionally, Sun et al. [35] proposed the GenRec model, which generates a paradigm for the next basket to capture the relevance of items within the basket and utilizes an autoregressive decoder to generate recommended items individually for the next basket.

2.2.3 GNN

In recent years, GNN methods have also been widely applied in recommendation systems. These methods enrich the representations of items and their neighbors by propagating messages between adjacent items, aiming to better learn the data structure of the graph. For instance, Liu et al. [36] proposed the Multi-Intent Translation Graph Neural Network (MITGNN) framework, which employs GNNs to learn the relationships between baskets and intents. Additionally, Liu et al. [37] introduced the GAT-TransNBR model, which utilizes Graph Attention Networks (GAT) to learn basket interaction features and employs Transformer to obtain user interest representations. These results outperform many existing neural network-based methods. However, they often require significant computational time and resources.

2.3 Finding Optimal Hyperparameters

While some of the above methods have shown promising performance in next-basket recommendation, they mainly treat the basket size K as a hyperparameter and utilize various search methods to find optimal hyperparameter settings.

For instance, Grid Search is the most intuitive method, which involves defining a range for hyperparameters and exhaustively trying each combination on a validation set to find the best hyperparameter configuration. However, this method requires a significant amount of training and evaluation, thus consuming considerable computational time and resources. On the other hand, Random Search [38] does not need to traverse all possible combinations of hyperparameters. Instead, it randomly samples a set of hyperparameters within the specified range for evaluation, making it typically more efficient than Grid Search. Bayesian Optimization [39], as a probabilistic model-based global optimization method, constructs a probability model of the objective function to search for the optimal hyperparameters. This approach requires fewer evaluations to find the best hyperparameters, making it generally more efficient than Random Search and Grid Search.

Although these hyperparameter tuning methods have shown promising results for models, they treat the basket size K as a fixed hyperparameter, overlooking the fact that the basket size may vary for different users or at different times for the same user. Factors influencing consumer purchasing behavior include economic, social, and psychological aspects, as mentioned by [40]. Additionally, experiments conducted by Badgaiyan et al. [41] have shown a significant correlation between individual impulse-buying tendencies and their sense of responsibility and extraversion. Moreover, it is intuitive that users tend to purchase more items during specific holidays such as Christmas or shopping festivals like Double Eleven. Therefore, dynamically adjusting the basket size is evidently more reasonable and can help provide recommendation results that better align with user needs.

2.4 Summary

Through the relevant literature discussed above, it can be observed that traditional methods overlook the global information of sequences, making them unsuitable for long-term prediction in recommendation systems. While methods based on RNNs can capture users' short-term and long-term preferences, they often neglect the correlations between items. Although Transformers leverage attention mechanisms to consider the correlations between data, they overlook the variations in basket size due to consumption habits or item attributes.

To overcome the limitations of existing methods, in this study, we propose an innovative approach that integrates basket size information into the basket recommendation process. This framework can be conceptualized as a multi-task learning architecture, where a Transformer model, which excels in capturing sequential contexts, serves as the core of the proposed architecture. The method first generates embeddings for both the basket representation and its size using respective encoders. These embeddings are then concatenated and used as input to the Transformer. After obtaining the Transformer's output through weighted calculation, the Basket Size Predictor is employed to predict the size of the basket. Finally, based on the predicted basket size, the Basket Predictor generates recommendations for items in the next basket.

3. METHODOLOGY

In this section, we will introduce the model framework for NBR-WBS. We begin by formally defining the problem and outlining the notation used throughout the paper. Next, we provide a high-level overview of the NBR-WBS architecture. Following this, we delve into a detailed description of each component within the model, explaining their functionalities. Finally, we discuss the loss function employed during the training process of the NBR-WBS model.

3.1 Problem Definition and Notation Explanation

Consider an e-commerce platform that includes a set of users $U = \{u_1, u_2, \ldots, u_{|U|}\}$, a set of items $I = \{i_1, i_2, \ldots, i_{|I|}\}$, and a set of baskets $B = \{B^1, B^2, \ldots, B^u, \ldots, B^{|U|}\}$, generated by user-item interactions. Here $u \in U$ represents a user from the user set, $i \in I$ represents an item from the item set, and B^u is the sequence of all baskets for user u. $|U|$ and $|I|$ represent the number of users and items, respectively. Each user's purchase history $B^u = \{b_1^u, b_2^u, \ldots, b_t^u\}$ is an ordered sequence of baskets, where $b_t^u \subseteq I$ is the basket of user u at time t. Therefore, each basket $b_t^u = \{i_1^u, i_2^u, \ldots, i_k^u\}$ is a set of items from a single transaction, without any order among the items, and k represents the number of items in the basket. In this paper, our goal is to predict the K items that a user is most likely to purchase in the next basket. Thus, the task is defined as follows: given a user's sequence of historical baskets $B^{u'} = \{b_1^u, b_2^u, \ldots, b_{t-1}^u\}$, predict the set of items $Y = \{y_1, y_2, \ldots, y_K\}$ that the user is likely to purchase in the next basket b_t^u, where $Y \subseteq I$, $y_K \in I$, and K represents the predicted basket size. $K \in \mathbb{N}$ is a non-negative integer. Table 1 provides the notation used in this paper.

Table 1: Description of Notation.

Notation	Description
$u \in U$	User u from the set of users U.
$i \in I$	Item i from the set of items I.
b^u	Historical basket sequence of user u.
k^u	Historical basket size sequence of user u.
$b_t^u \subseteq I$	The t-th basket purchased by user u, which is a subset of item set I.
$k_t^u \in \mathbb{N}$	Size of the t-th basket purchased by user u.
I_i	Embedding of item i.
B_t^u	Embedding of the t-th basket of user u.
K^u	Embedding of the basket size of user u.
E^u	Embedding vector combining the embeddings of basket size and basket
B_f	The final state vector of the basket
$Y \subseteq I$	The predicted set of items
$y_K \in I$	Item(s) in the predicted set of items
K	The predicted basket size

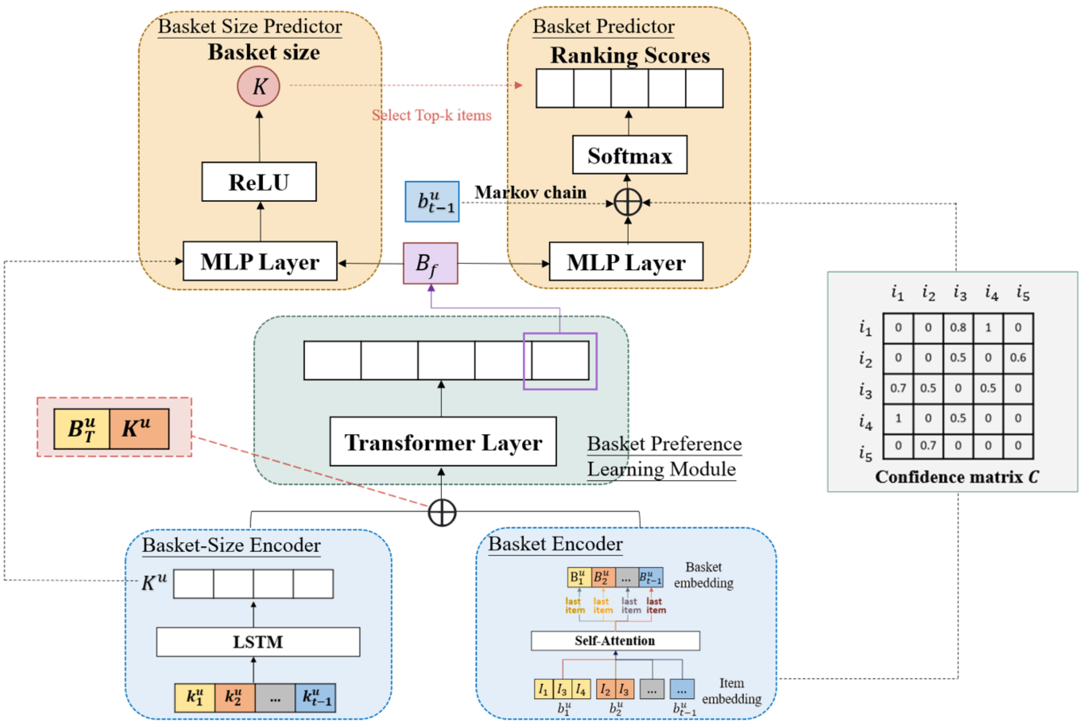
3.2 Model Overview

In this section, we outline the framework of the proposed NBR-WBS model. As illustrated in Figure 5, NBR-WBS consists of three main parts: the embedding module, the basket preference learning module, and the prediction module.

- **Embedding Module**: The purpose of the embedding module is to convert basket information into embedding vectors, which is further divided into the Basket Encoder and Basket-Size Encoder. The former takes the user's historical basket sequence b^u as input and generates the corresponding embeddings B_T^u for each basket using attention mechanisms, where $T = 1, 2, \ldots, t-1$. The latter uses an LSTM model to obtain embeddings K^u for the basket sizes k_T^u of each basket.
- **Basket Preference Learning Module**: This module primarily learns sequence relationships and user purchasing preferences from the user's historical basket sequence. By concatenating the basket embeddings B_T^u and basket size embeddings K^u, the embedding

representation E^u of the basket sequence is obtained. Transformer is then used to learn the relationships between basket sequences, outputting another sequence vector, with the last item of the sequence serving as the final basket state vector B_f. This process can be seen as a multi-task learning, with both tasks sharing the parameters of the Transformer.

- **Prediction Module:** The prediction module is divided into two components: Basket Predictor and Basket-size Predictor. Both utilize the final state vector B_f from the previous layer as input and are trained using MLPs. In Basket Predictor, inspired by ASBRec[42], we establish a trust matrix C using association rules to capture complementary and substitute relationships between items. Additionally, we introduce a Markov chain to comprehensively model user behavior. Finally, the Softmax activation function is applied to obtain the predicted probabilities $\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{|I|}\}$ for each item appearing in the next basket. In Basket-size Predictor, we add the basket size embedding K^u to the final state vector B_f. inputting it into an MLP for training. The ReLU activation function is used to obtain the predicted quantity K of items in the next basket. Lastly, we sort the predicted probabilities $\{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_{|I|}\}$ obtained from Basket Predictor and select the top K items as the final recommended item set $Y = \{y_1, y_2, \dots, y_K\}$.



	i_1	i_2	i_3	i_4	i_5
i_1	0	0	0.8	1	0
i_2	0	0	0.5	0	0.6
i_3	0.7	0.5	0	0.5	0
i_4	1	0	0.5	0	0
i_5	0	0.7	0	0	0

Figure 5: NBR-WBS model structure.

3.3 Embedding Module

3.3.1 Basket Encoder

In the Basket Encoder, we first adopt the Item2Vec method [23] for item embedding. The idea of Item2Vec is mainly inspired by the Word2Vec [43] model, which considers the user's behavior sequence as sentences composed of multiple items and uses the Skip-gram with negative sample (SGNS) method to generate item embedding vectors. Skip-gram originally appeared in Word2Vec, using a center word to predict surrounding words, and obtaining a word vector matrix after training, where similar words are closer in the vector space. Item2Vec employs a similar concept, predicting other items in the same basket through the target item, thereby capturing the relationship between the target item and other items, and obtaining the learned item embedding matrix. In recommendation systems, due to the large number of items, the Negative sample is mainly used to address the problem of excessive computational complexity caused by updating all item embedding vectors for each training sample. This method selects several items from different baskets to serve as negative examples and only updates the weights related to negative sampling to enhance training efficiency.

In particular, we first regard the basket as a sentence and the items in the basket as words. When performing Skip-gram, each item in the basket is converted into One-Hot Encodings and multiplied by a randomly initialized weight matrix $W_{|I| \times N}$ to obtain its N-dimensional latent vector. Subsequently, another weight matrix $W'_{N \times |I|}$ is randomly initialized. For positive samples, The N-dimensional vector of the target item is subjected to dot-product operations with the vectors of all other items in the same basket from $W'_{N \times |I|}$, followed by a sigmoid function to map the predicted results between [0,1]. The same procedure is applied to negative samples, but only selecting several vectors of items from different baskets in $W'_{N \times |I|}$ to dot-product with the target item. During training, the target function is corrected through Stochastic Gradient Descent (SGD) to update the weight matrices $W_{|I| \times N}$ and $W'_{N \times |I|}$. Finally, the trained matrix $W_{|I| \times N}$ or $W'_{N \times |I|}$ can be used as item embeddings for subsequent applications, The Skip-gram process as shown in Figure 6.

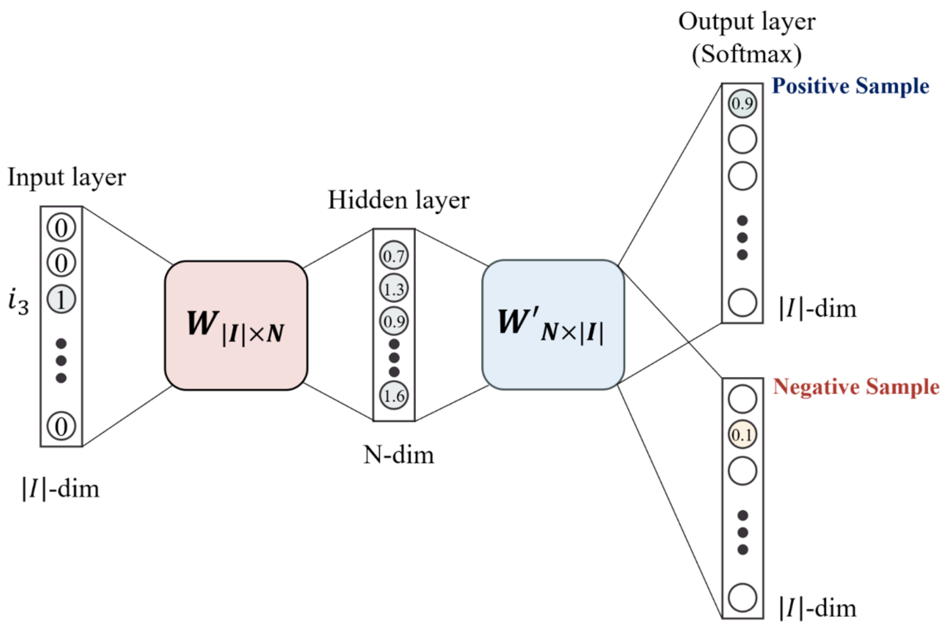


Figure 6: Skip-gram in Item2Vec

Therefore, given a basket containing a set of items of length k, denoted as$i_1,\ i_2,\ \ldots, i_k$, we utilize Skip-gram to maximize the probability of the target item appearing together with other items in the same basket, while minimizing the probability of other items from different baskets appearing together with the target item. Namely, maximizing the following objective function.

$$\frac{1}{k}\sum_{m=1}^{k}\sum_{n\neq m}^{k} log\, p(i_n|i_m) \tag{1}$$

Where i_m represents the m-th target item, and i_n represents the n-th other item in the same basket, negative sampling is applied in $p(i_n \mid i_m)$. This involves randomly selecting some items not in the same basket as negative examples for training, where only partial weights are updated to reduce computational complexity. The computation formula is as follows:

$$p(i_n|i_m) = \sigma(u_m^{\top} v_n)\prod_{k=1}^{N}\sigma(-u_m^{\top} v_k) \tag{2}$$

Where $\sigma(u_m^{\top} v_n)$ represents the prediction result for positive samples, with u_m and v_n being the latent vectors for the m-th target item and the n-th other item in the same basket, respectively. $\sigma(-u_m^{\top} v_n)$ represents the prediction result for negative samples, where N is the number of negative samples, and v_k is the latent vector for the k-th item in a different basket. Here, $\sigma(\cdot)$ denotes the sigmoid non-linear activation function. If the result is close to 1, it indicates that the item is in the same basket as the target item, while a result close to 0 indicates that they are in different baskets.

Furthermore, since items in the basket may appear with varying frequencies, those with high frequencies contain less information. Therefore, to balance the probabilities of rare and frequent item occurrences during negative sampling, the following formula is utilized to determine the probability of discarding each item.

$$p(\,discard \mid i\,) = 1 - \sqrt{\frac{\rho}{f(i)}} \tag{3}$$

Where $f(i)$ represents the frequency of item i appearing, and ρ is the threshold. The larger the frequency of an item, the higher the probability it is discarded, thereby increasing the likelihood of rare items being sampled.

After obtaining the item embeddings, the next step is to encode the baskets. Here, we utilize Self-attention [32] to compute attention weights for each item in the user's basket, capturing the relationships between items within the basket, and outputting this as the basket's embedding vector. Therefore, given the items in the user's basket, $b_t^u = \{i_1^u, i_2^u, \ldots, i_k^u\}$, we independently multiply them by three learnable parameter matrices W^Q, W^K, and W^V to obtain the Q(Query), K(Key), and V(Value) transformation matrices.

$$\begin{aligned} Q &= b_t^u \times W^Q \\ K &= b_t^u \times W^K \\ V &= b_t^u \times W^V \end{aligned} \tag{4}$$

Next, we input these three matrices into the Self-attention module to perform Scaled Dot-Product Attention calculation, a process that provides each item with a vector weighted by attention. Here, since these vectors have already learned from each other during Self-attention, we directly extract the vector of the last item, which already contains information from other items, and use it as the representative embedding vector B_t^u for the entire basket.

3.3.2 Basket-Size Encoder

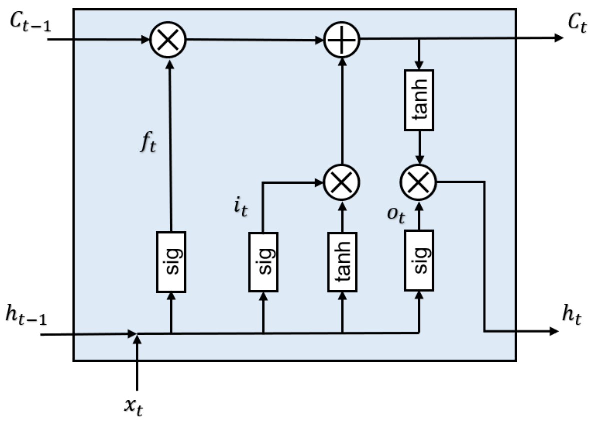


Figure 7: Modern representation of LSTM.

Due to the traditional Recurrent Neural Network (RNN) potentially encountering gradient vanishing or exploding issues when handling long sequences, we use Long Short-Term Memory

(LSTM) [44] to encode the user's historical basket size sequence. LSTM is a special form of RNN, as shown in Figure 7. It controls the flow of information through gate units, including the forget gate, input gate, and output gate. This allows for better regulation and control of information, effectively capturing long-term dependencies. The LSTM calculation formulas are as follows:

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t * c_{t-1} + i_t * \sigma_c(W_c x_t + b_c) \\ h_t &= o_t * \ \sigma_h(c_t) \end{aligned} \tag{5}$$

In the Basket-Size Encoder, the input is the user's historical basket size sequence. Using the LSTM model as the embedding layer for basket sizes, each size in the basket size set is projected into the latent space to obtain the representation K^u for basket size. Let $k_T^u = \{k_1^u, k_2^u, \ \ldots k_{t-1}^u\}$ denote the user's historical basket size sequence. The calculation formulas are as follows:

$$K^u = LSTM(k_T^u) \tag{6}$$

After obtaining the basket size embedding K^u, we concatenate it with the basket embeddings $\{B_1^u, B_2^u, \ \ldots B_{t-1}^u\}$ obtained from the previous Basket Encoder section to produce the combined embedding basket vector E^u. This vector contains information about both the basket and its size. The formula is as follows:

$$E^u = \{B_T^u, \ K^u\}; \quad T = 1, \, 2, \, \ldots, \, t-1 \tag{7}$$

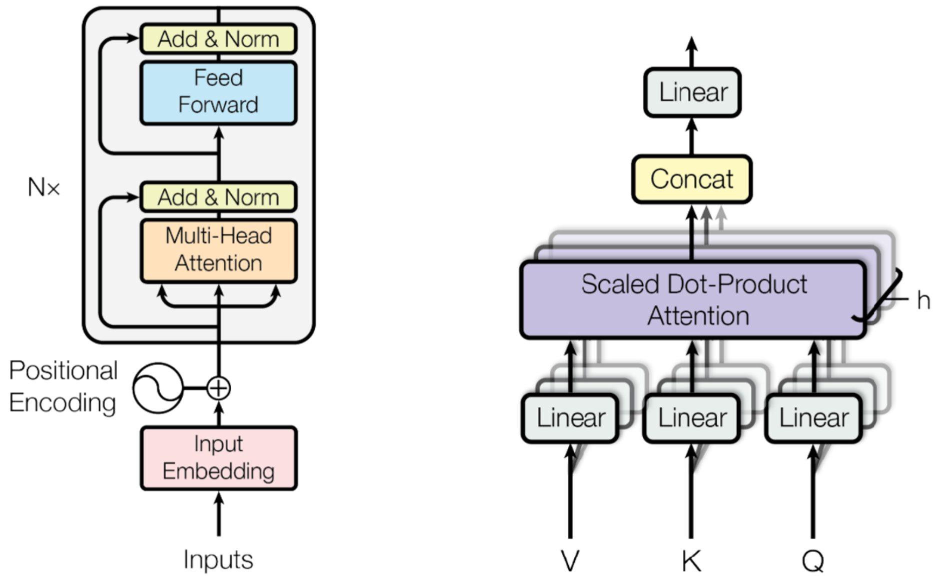
3.4 Basket Preference Learning Module

Next, we use a Transformer to encode the basket sequence, as shown in Figure 8(a), to learn the sequential relationships between baskets. As previously mentioned, the Transformer processes each input in parallel, enabling it to focus on the correlations between data without being constrained by their spatial distance, thereby improving training efficiency.

However, this method loses the order relationships between inputs, resulting in the model's inability to capture the temporal information in the sequence. Therefore, to consider the sequential information of the baskets, positional information needs to be added to each input. Here, we combine the embedding vectors produced in the previous step, $E^u = \{E_1^u, E_2^u, \dots, E_{t-1}^u\}$, with Position Encoding to obtain basket representations that include positional information. The Position Encoding is calculated as follows:

$$\begin{aligned} PE_{(pos,\,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,\,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{8}$$

Where pos is the position of the input vector, $2i$ denotes even indices in the encoding, and $2i+1$ denotes odd indices in the encoding. d_{model} refers to the dimension of the embedding. The combined embedding vector E^u are added to the positional encoding PE to serve as the input to the self-attention layer in the subsequent encoder stage: $E'^u = \{E'^u_1, E'^u_2, \dots, E'^u_{t-1}\}$.



(a) (b)

Figure 8: (a)Transformer Encoder in Transformer model. (b)Multi-Head Attention. [32]

In the self-attention layer, the relevance weights between different positions in the sequence are computed to understand their importance. Therefore, the combined embedding basket vector with positional encoding, $E'^{u} = \{E'^{u}_{1}, E'^{u}_{2}, \dots, E'^{u}_{t-1}\}$, is input into the self-attention mechanism to perform self-attention calculations for each basket, considering the relationships between baskets. Similar to the process of learning basket embeddings, the basket embedding vector E'^{u}, multiplied by parameter matrices W^{Q}, W^{K}, and W^{V}, yields three transformed matrices Q, K, and V. These matrices are then used in the scaled dot-product attention calculation, as described by the formula below.

$$Attention(Q, K, V) = softmax\left(\frac{QK^{T}}{\sqrt{d_k}}\right)V \tag{9}$$

In this process, we use Q as the query vector for the current basket, K as the key vector for the searched baskets, and V as the weighted basket matrix. Here, Q and K are of equal dimensions, denoted as d_k. We compute the dot product of the query vector Q and the key vector K, divided by $\sqrt{d_k}$, to ensure that the dot product results do not suffer from the vanishing or exploding gradient problem in larger dimensions. Then, we apply Softmax to obtain the scaled attention weights. Finally, we multiply the calculated attention weights by V to obtain the weighted vectors.

In Transformer, Multi-Head Attention is additionally employed, as illustrated in Figure 8(b), allowing the model to capture richer features in parallel. Given the transformed matrices Q, K, and V, we multiply them by multiple sets of matrices W_i^{Q}, W_i^{K}, and W_i^{V}, where $i = 1, \dots, h$, to obtain h sets of transformed matrices. Subsequently, the scaled dot-product attention calculation is performed separately for each of the h sets of matrices to obtain the attention values for each head $head_i$. Finally, these $head_i$ are concatenated together and linearly transformed using the projection matrix W^O to obtain the output Z. The formula for Multi-Head Attention is shown in Equations (10) and (11).

$$head_i = Attention(QW_i^Q,\ KW_i^K,\ VW_i^V) \tag{10}$$
$$Z = MultiHead(Q, K, V) = Concat(head_1,\ head_2, \dots, head_h)W^O \tag{11}$$

While it's common practice to stack multiple Transformer encoder modules in the model to capture hierarchical features of sequential representations, such multi-layer deep networks are susceptible to issues like gradient vanishing or network degradation. Therefore, additional steps are taken to perform residual connections and layer normalization to enhance model stability and expedite the training process. As described in Equation (12), we input the vectors Z outputted by the Multi-Head Attention and the basket embedding vector E'^u into the Add & Norm layer for residual connection, mitigating the problem of gradient vanishing, followed by layer normalization to improve performance.

$$Z' = LayerNorm\left(E'^u + Z\right) \tag{12}$$

The normalization formula is as follows:

$$LayerNorm(x) = \alpha \odot \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta \tag{13}$$

Where $\odot$ denotes the element-wise product, μ and σ represent the mean and standard deviation of the input x, respectively, while α, β, and ε are learnable scaling and bias parameters.

Next, the Transformer introduces a Feed-Forward Network (FFN) to endow the representations with non-linear capabilities and integrate the interaction effects among different dimensional features. This involves two operations with projection matrices: first, mapping the input vectors into a high-dimensional space to expand the feature dimensions, then, employing the ReLU non-linear activation function to better extract important features, and finally, projecting the results back to the original space. The formula is as follows:

$$FFN(Z') = ReLU(Z'W_1 + b_1)W_2 + b_2 \tag{14}$$

Where W_1, W_2, b_1, and b_2 are learnable parameters. Additionally, we apply an Add & Norm layer to enhance the convergence speed during model training, resulting in the normalized vector matrix Z^*, as depicted in Equation (15). Subsequently, we extract the last basket embedding vector from the Transformer output matrix Z^*, which serves as the final basket state vector B_f for the user.

$$Z^* = LayerNorm(Z' + FFN(Z')) \tag{15}$$

The data mentioned above is used to train the Transformer for two concurrent tasks: predicting basket size and predicting items in the basket. This approach is known as Multi-Task Learning. Compared to using two separate networks for prediction, Multi-Task Learning, through a shared Transformer, not only reduces the number of trainable parameters and the complexity of the model but also enhances its generalization ability. After obtaining the basket features B_f from the shared Transformer, we input them into task-specific components for prediction to better address the specific requirements of each task.

3.5 Prediction Module

3.5.1 Basket Predictor

In the final stage, we input the ultimate basket state vector B_f into individual components for specific prediction tasks. Firstly, in the Basket Predictor, we feed B_f into an MLP (Multilayer Perceptron) layer for training. The MLP is a deep neural network structure typically consisting of multiple hidden layers, each containing multiple neurons. Through the nonlinear transformations of these hidden layers, the MLP can learn complex feature representations. The computation formula is as follows:

$$p = MLP(B_f) \tag{16}$$

Additionally, to capture the similarities and complementarities between items, we employed the methodology proposed by ASBRec [42] to create a confidence matrix C for item associations. Confidence refers to the probability of finding item i_j in a transaction that contains item i_x, denoted as $P(i_j|i_x)$. We compute the confidence using the conditional probability formula: $P(i_j|i_x) = \frac{P(i_j \cap i_x)}{P(i_x)}$. This formula allows us to determine the probability of other items appearing together when one item is present. Subsequently, we establish a confidence matrix C for all items and integrate it with the outcomes of the aforementioned MLP. The combination formula is as follows:

$$p' = p \circ \omega + p\,C + p \tag{17}$$

Here, $\odot$ denotes the element-wise product and ω represents the importance of items, where we use the proportion of item occurrences as their importance. Additionally, to capture specific user behaviors over short periods, we introduce a Markov chain method in our model to comprehensively model user preferences. The Markov chain method is implemented by constructing a transition probability matrix, which describes the probability of transitioning from one item to another. This method focuses on analyzing recent interactions with users' baskets and predicts the probability M of items appearing in the next basket based on the items in the current basket. Finally, we aggregate the results from these three methods to generate predictive scores for all items appearing in the next basket. After applying the Softmax activation function, we output the probability of each item appearing in the next basket. The calculation formula is as follows:

$$\hat{y} = Softmax\big(\beta\big(Z_{score}(M) + Z_{score}(p')\big) + (1-\beta)p'\big) \tag{18}$$

Where M represents the probability of items appearing in the next basket calculated using the Markov chain method, while β is employed to coordinate the weighting coefficients between the Markov chain and the method combining the MLP with Confidence.

3.5.2 Basket-Size Predictor

Similarly, in the Basket-Size Predictor, we add the final basket state vector B_f to the basket size embedding K^u via residual connection and input it into an MLP layer for training, to further extract features representing the basket size. Since the basket size is a non-negative integer, we employ the ReLU activation function to ensure that the predicted value of the basket size K remains within the range of non-negative integers. The computation formula is as follows:

$$K = ReLU\left(MLP\left(B_f + K^u\right)\right) \tag{19}$$

Finally, we rank the probabilities y of items appearing in the next basket and select the top K items with the highest scores as our final prediction results, where K denotes the predicted basket size. The formula is presented below:

$$Y = \text{Top-}K\ (\hat{y}) \tag{20}$$

3.6 Loss Function

Since our model needs to simultaneously predict the items in the user's basket and the basket size, we employ two different loss functions during the training process to optimize these two prediction tasks. Additionally, we introduce a weighting coefficient α to balance the importance of the two tasks. By adjusting α, we can effectively minimize the total loss. The calculation formula for the total loss is as follows:

$$\mathcal{L} = \alpha\mathcal{L}_1 + (1-\alpha)\mathcal{L}_2 \tag{21}$$

Here, L_1 represents the loss function for basket size prediction. We use Mean Squared Error (MSE) as the loss function to compute the error between the predicted and actual values, as given in Equation (22). On the other hand, L_2 represents the loss function used for basket item prediction. Here, we aim to predict the probability y of each item appearing in the user's next basket. The objective is for these predicted probabilities y to closely match the actual outcomes of the next basket. Therefore, we minimize the Binary Cross-Entropy loss function to determine the set of parameters in the model, as given in Equation (23).

$$\mathcal{L}_1 = \frac{1}{t}\sum_{i=1}^{t}(K_i - \widehat{K}_i)^2 \tag{22}$$

$$\mathcal{L}_2 = -\frac{1}{n}\sum_{i=1}^{n}\hat{y}_i \cdot \log(y_i) + (1-\hat{y}_i) \cdot \log(1-y_i) \tag{23}$$

In this context, t represents the total number of data points for the basket size prediction task, and $\widehat{K}_i$ denotes the actual size of the basket. n is the total number of items, and $\hat{y}_i$ indicates whether the i-th item is present in the next basket, where 0 represents absence and 1 represents presence. If $\hat{y}_i$ is 0, it means the i-th item does not appear in the next basket; if $\hat{y}_i$ is 1, it means the next basket contains this item. Additionally, y_i represents the probability predicted by the model for the i-th item to appear in the next basket.

4. EXPERIMENTS AND RESULTS

4.1 Datasets

To validate the effectiveness of the proposed model in this study, we selected three real-world datasets for our experiments.

- **TaFeng**: The dataset comprises a total of 817,741 records of transactional data collected from a grocery store in China between November 2000 and February 2001, where all products purchased by the same user on the same day are treated as one basket.
- **Dunnhumby**: Released by a commercial data analytics firm, Dunnhumby, this dataset encompasses all purchases made by 2,500 households over two years at a retail store. It includes 92,399 distinct products. However, our study utilizes only two months' worth of data from the dataset
- **Instacart**: This dataset originates from the American grocery delivery service company, Instacart, and comprises over 3 million orders from 206,209 users in 2017, involving 49,688 distinct products. To manage the dataset effectively, our study selects only 10% of users for analysis.

For data preprocessing, we consider all items purchased by a user on the same day as one basket. Items appearing fewer than 5 times are removed, along with baskets containing fewer than 3 items and users with fewer than 3 baskets. Table 2 provides statistical data after preprocessing.

Table 2: The dataset details after preprocessing.

	# Items	**# Users**	**Avg. basket size**
Ta Feng	15,764	10,654	8.55
Dunnhumby	3,977	19,132	10.09
Instacart	27,095	19,485	11.02

Finally, we take the last basket of each user as the target for prediction, while the remaining data serve as the user's historical sequence. The dataset is split into training, validation, and testing sets in an 8:1:1 ratio.

4.2 Baselines

To evaluate the effectiveness of the proposed model, we compare it with the following baseline methods: FPMC, HRM, TIFUKNN, DREAM, SHAN, CLEA, Beacon, and Sets2Sets. For all baseline methods, we fine-tune the hyperparameters based on the settings in their respective papers. Depending on the dataset, we set the number of neighbors for TIFUKNN between 30 and 50, commonly used in collaborative filtering.

1. FPMC[11]: A framework based on Markov chains, this method predicts the next basket based on the transition probability of items in the previous basket, considering user-item interactions.
2. HRM[15]: A model based on hierarchical representation, this method recommends the next basket by considering user preferences and behavioral sequences.
3. TIFUKNN[12]: Using a time-dynamic modeling approach to capture the frequency information of past user baskets, this method then employs a KNN-based approach on personalized item frequency (PIF) to recommend the next basket.
4. DREAM[16]: The first method to apply deep learning-based approaches in recommending the next basket, this model first generates basket representations through pooling layers, then inputs them into an RNN to learn user representations and predict the next set of items.
5. SHAN[45]: A dual-layer attention network, the first layer learns long-term user preferences based on historical purchased items representations, while the second layer outputs the final user representation by combining long-term and short-term preferences.
6. CLEA[13]: A contrastive learning model that designs a denoising generator to automatically identify and extract items relevant to the target item and generates basket

representation using a context encoder based on GRU.

7. Beacon[17]: An RNN-based method, considering merging pairwise correlation information between items to encode the basket.
8. Sets2Sets[31]: An RNN-based encoder-decoder approach for basket prediction. In addition, an attention mechanism focusing on item frequency is proposed to improve the performance.

4.3 Evaluation Metrics

In recommendation systems, common evaluation metrics include F1-Score, Precision, Recall, Hit Ratio (HR), and NDCG. However, when using Hit Ratio (HR) as an evaluation metric, if the basket size K is sufficiently large, it means that the recommendation model includes most of the correct items in the recommendation list, thereby increasing the likelihood of hits, but also causing HR to lose discrimination ability for model performance. F1-Score is the harmonic mean of Precision and Recall, therefore, we only adopt F1-Score and NDCG to evaluate the performance of the proposed model and other baselines. These metrics can provide more representative evaluations, especially when the basket size K is dynamically adjusted, enabling a better understanding of model performance. For F1-Score, we calculate the following for the Top-K results:

$$F1@K = \frac{2 \times (Precision@K \times Recall@K)}{Precision@K + Recall@K} \tag{24}$$

As previously mentioned, when K is set too large, the increase in false positives (FP) will lower the model's Precision. Conversely, if K is set too small, the increase in false negatives (FN) will reduce the model's Recall. Therefore, to achieve an optimal F1-Score, both Precision and Recall need to perform well. The formulas for Precision@K and Recall@K are as follows:

$$Precision@K = \frac{TP@K}{TP@K + FP@K} \tag{25}$$

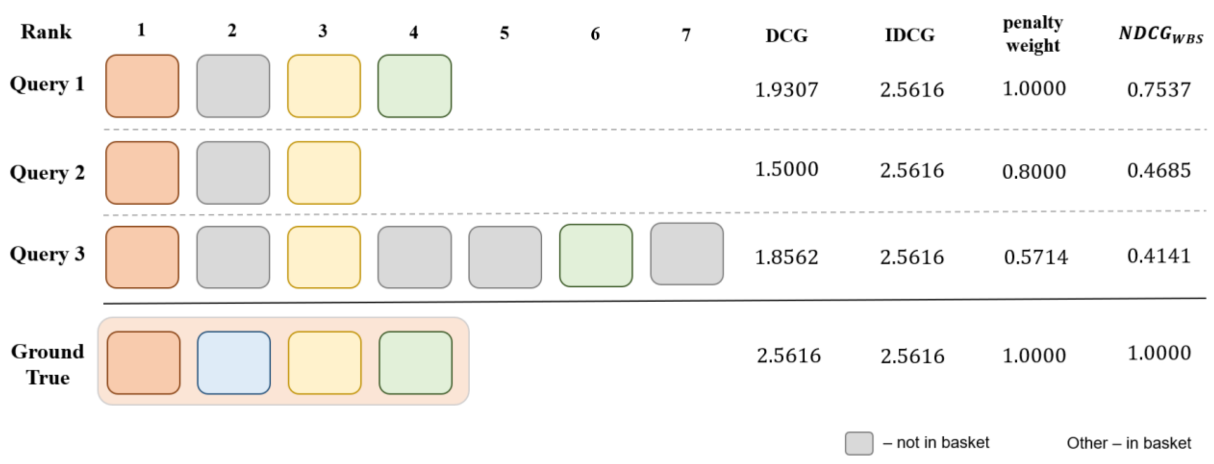
$$Recall@K = \frac{TP@K}{TP@K + FN@K} \tag{26}$$

NDCG@k (Normalized Discounted Cumulative Gain) is a position-aware evaluation metric used to reflect the relevance of an item's position. It assigns higher gains to items predicted closer to their correct positions and discounts the value of items ranked lower. Since our model considers the dynamic characteristics of basket sizes, we have modified the NDCG metric for fair comparison with other baseline models. We introduce a penalty term for basket size. When the model correctly predicts the basket size, the penalty value is 1. However, if the predicted value is greater or less than the actual K value, the penalty value will be less than 1, thereby affecting the NDCG value. The modified NDCG formula, also known as $NDCG_{WBS}$, is as follows:

$$NDCG_{WBS}@K = \frac{DCG@K}{IDCG@K} \times \frac{\widehat{K}}{\widehat{K} + \left|\widehat{K} - K\right|} \tag{26}$$

$$DCG@K = \sum_{i=1}^{K} \frac{2^{r(i)} - 1}{log_2(1 + i)} \tag{27}$$

In Equation (27), $DCG@K$, $r(i)$ denotes the score of the i-th recommended item, which is assigned a value of 1 if the recommendation is a hit, and 0 otherwise, with the scores of the top K items being discounted or accumulated. To ensure a consistent comparison of DCG across different size of basket, it is normalized by dividing by the IDCG (Ideal Discounted Cumulative Gain). Specifically, if K is set too small, it may overlook the scores of other items in the basket, resulting in a lower DCG value than the actual scores. Conversely, if K is set too large, the accumulated DCG value may be influenced by irrelevant items, leading to a decrease in NDCG value.



Rank	1	2	3	4	5	6	7	DCG	IDCG	penalty weight	$NDCG_{WBS}$
Query 1	🟧	⬜	🟨	🟩				1.9307	2.5616	1.0000	0.7537
Query 2	🟧	⬜	🟨					1.5000	2.5616	0.8000	0.4685
Query 3	🟧	⬜	🟨	⬜	⬜	🟩	⬜	1.8562	2.5616	0.5714	0.4141
Ground True	🟧	🟦	🟨	🟩				2.5616	2.5616	1.0000	1.0000

⬜ – not in basket   Other – in basket

Figure 9: $NDCG_{WBS}$ values with different predicted basket sizes.

As illustrated in Figure 9, firstly, if the model accurately predicts the correct baskets, the DCG value is 2.5616 ($\frac{1}{\log_2 2}+\frac{1}{\log_2 3}+\frac{1}{\log_2 4}+\frac{1}{\log_2 5}$), where the IDCG value is also 2.5616 ($\frac{1}{\log_2 2}+\frac{1}{\log_2 3}+\frac{1}{\log_2 4}+\frac{1}{\log_2 5}$). The penalty term for basket size is 1 ($\frac{4}{4+|4-4|}$), thus resulting in $NDCG_{WBS}$ of 1.0000 ($\frac{2.5616}{2.5616}\times 1$).

In Query 1, as the model fails to predict all items in the baskets accurately, the DCG decreases to 1.9307($\frac{1}{\log_2 2}+\frac{0}{\log_2 3}+\frac{1}{\log_2 4}+\frac{1}{\log_2 5}$), while the IDCG remains 2.5616. Since the prediction of the basket size is correct, the penalty term for basket size remains 1, resulting in a decrease of $NDCG_{WBS}$ to 0.7537 ($\frac{1.9307}{2.5616}\times 1$).

In Query 2, apart from incorrectly predicting the items in the baskets, the model also fails to accurately predict the basket size, resulting in a decrease in DCG to 1.5000 ($\frac{1}{\log_2 2}+\frac{0}{\log_2 3}+\frac{1}{\log_2 4}$). Consequently, the penalty term for the basket size becomes 0.8 ($\frac{4}{4+|4-3|}$), leading to a final decrease in the $NDCG_{WBS}$ value to 0.4685 ($\frac{1.5000}{2.5616}\times 0.8$).

In Query 3, although the model predicts more items correctly compared to Query 2, the DCG is 1.8562 ($\frac{1}{\log_2 2}+\frac{0}{\log_2 3}+\frac{1}{\log_2 4}+\frac{0}{\log_2 5}+\frac{0}{\log_2 6}+\frac{1}{\log_2 7}+\frac{0}{\log_2 8}$). However, due to a significant difference between the predicted and actual basket sizes, the penalty term for basket size becomes 0.5714 ($\frac{4}{4+|4-7|}$). Consequently, the $NDCG_{WBS}$ value further decreases to 0.4141 ($\frac{1.8562}{2.5616} \times 0.5714$).

4.4 Experimental Setup

In Section 4.6, we conducted a sensitivity analysis on various parameters in the experiment to study the effects of different hyperparameter settings on the model. Simultaneously, we selected the optimal hyperparameter configurations for application in subsequent experiments. It is worth noting that as we are the first to propose a model that dynamically adjusts basket size, we need to compare our model with other baselines that use fixed basket sizes in the experiment. Specifically, to ensure fair comparison and prevent other baselines from being influenced by the K value, thereby affecting their performance, our model is compared with baselines where results for each method across different K values are averaged (with K values of [5, 10, 20, 40]). Detailed results for each method without averaging across K values are provided in the Appendix.

To ensure that our proposed model has the best model structure, we conducted multiple experiments. In both Self-attention and Transformer, the dimensionality of the parameter matrices W^Q, W^K and W^V was set to 32 dimensions. We explored the optimal number of layers and heads in Transformer within the range of [1, 2, 4, 8], and ultimately set them to 4 layers and 4 self-attention heads in TaFeng and Dunnhumby to achieve the best results.

During the training process, we observed that in the TaFeng dataset, the model performed best when Epoch was set to 25 and Learning Rate was set to 0.0001. In the Dunnhumby and Instacart datasets, we found that setting Epoch to 15 and Learning Rate to 0.00001 and 0.001, respectively, yielded the best results. Additionally, we chose to use the Adam optimizer to update the model parameters. To balance the impact of different loss functions, we adjusted the weight coefficient α in the range of 0.1 to 0.9. Finally, we observed that in the TaFeng dataset, setting the weight coefficient α of the loss function to 0.1 yielded the best performance, while in the Dunnhumby and Instacart datasets, setting it to 0.01 resulted in the best performance. Table 3 summarizes the optimal hyperparameter settings of our model.

Table 3: Presents the optimal hyperparameter settings of the NBR-WBS model across different datasets.

DataSet	Epoch	Learning Rate	Embedding Size	Batch Size	Multi Head	Transformer Layer	α	β
Ta Feng	25	0.0001	32	8	4	4	0.1	0.95
Dunnhumby	15	0.00001	32	8	4	4	0.01	0.001
Instacart	15	0.001	32	8	2	4	0.01	0.001

Finally, to ensure fair comparison, all our experiments were conducted using the PyTorch deep learning framework. Additionally, the specifications of the computer equipment used during the experiments are as follows:

- Processor: Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz
- Memory: 128GB
- Graphics Card: NVIDIA GeForce RTX 3090 Ti

4.5 Performance Comparison

Table 4 and 5 presents the comparison results between our model and different approaches across the three datasets, using F1-Score and $NDCG_{WBS}$ as evaluation metrics.

Table 4: The performance of our model compared to other baselines in terms of F1-Score.

Model	TaFeng	Dunnhumby	Instacart
FPMC	0.02934	0.08206	0.06423
HRM	0.03156	0.08586	0.08051
SHAN	0.03193	0.07715	0.06202
DREAM	0.03657	0.07606	0.06543
Beacon	0.03944	0.07779	0.06326
CLEA	0.04184	0.10884	0.10980
Sets2Sets	<u>0.04944</u>	0.13868	0.17141
TIFUKNN	0.04895	<u>0.14721</u>	<u>0.17871</u>
NBR-WBS	**0.05164**	**0.15182**	**0.19143**

Table 5: The performance of our model compared to other baselines in terms of $NDCG_{WBS}$.

Model	TaFeng	Dunnhumby	Instacart
FPMC	0.02537	0.06860	0.05013
HRM	0.02857	0.07146	0.06412
SHAN	0.02757	0.06460	0.04885
DREAM	0.02934	0.06341	0.05105
Beacon	0.03162	0.06426	0.04914
CLEA	0.03400	0.09227	0.09153
Sets2Sets	<u>0.03683</u>	0.10934	0.12025
TIFUKNN	0.03563	<u>0.11654</u>	<u>0.13747</u>
NBR-WBS	**0.04743**	**0.13139**	**0.17622**

Finding 1: Among the baselines, traditional methods, although relatively simple, can achieve excellent performance. In the Dunnhumby and Instacart datasets, TIFUKNN is the best-performing method among all baselines. This method combines item attributes with a user-based nearest neighbor approach. This indicates that traditional methods, despite their simplicity, can be highly effective on suitable datasets.

Finding 2: Traditional methods can outperform deep neural network-based methods on most datasets. We observe that the performance of different methods varies across different datasets, and no single method consistently outperforms all others on every dataset. Although Sets2Sets shows better performance than TIFUKNN on the TaFeng dataset, it is outperformed by TIFUKNN on the Dunnhumby and Instacart datasets.

Finding 3: Sets2Sets, a deep neural network-based method, and TIFUKNN, a traditional KNN-based method, achieve the best performance among baselines on different datasets. Considering that Table 2 shows the average basket length in the TaFeng dataset is shorter than in Dunnhumby and Instacart, this might suggest that Sets2Sets is more suitable for datasets with shorter baskets, while TIFUKNN is more effective for datasets with longer baskets.

Finding 4: Our proposed model, NBR-WBS, achieves the best results compared to all baselines, demonstrating that dynamically predicting basket size has a significant impact on the performance of next-basket recommendations. Further analysis shows that our model can more accurately capture the trend of changes in users' basket sizes, dynamically adjusting the basket size based on different user characteristics, thereby enhancing the overall performance of the recommendation system.

4.6 Sensitivity analysis

Next, we will conduct sensitivity analysis to assess the impact of different parameters on the model and validate whether the parameter settings achieve optimal performance.

We start by examining the learning rate settings. If the learning rate is set too small, it might slow down the model's convergence speed, thereby impeding effective weight updates to achieve better solutions. Conversely, if set too large, it might hinder the model from learning meaningful features, leading to convergence challenges and suboptimal solutions. Figure 10 illustrates that optimal performance is achieved with learning rates of 0.0001, 0.00001, and 0.001 for TaFeng, Dunnhumby, and Instacart, respectively.

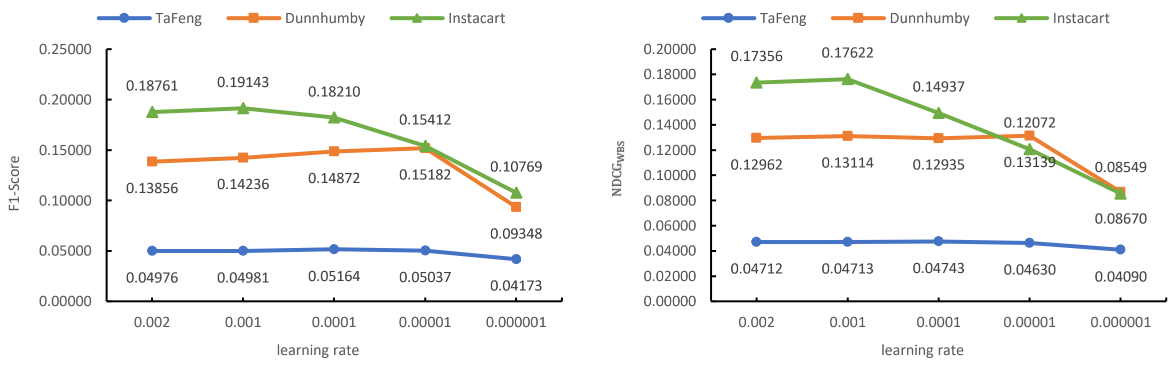


Figure 10: The results of different learning rate settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The second parameter to test is the alpha value in the loss function. Setting alpha too high can cause the model to focus excessively on predicting basket size, neglecting the prediction of items within the basket. Conversely, setting alpha too low can lead to the opposite issue. Figure 11 shows that the best results are achieved with an alpha of 0.1 in the TaFeng dataset, while an alpha of 0.01 yields the best performance in the Dunnhumby and Instacart datasets. Overall, the model's performance remains relatively stable across different alpha values.

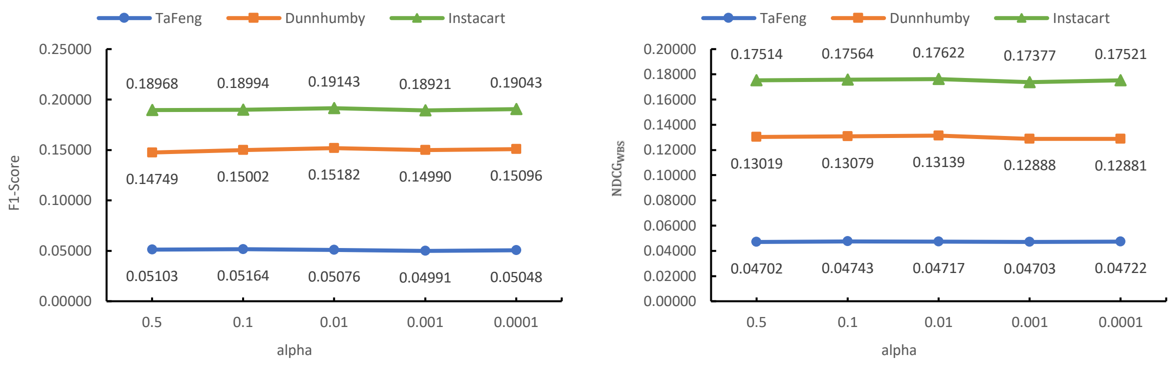


Figure 11: The results of different alpha settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The third parameter to test is the batch size. As shown in Figure 12, the best performance across all three datasets is achieved with a batch size of 8. When the batch size is increased to 16, the performance declines, which is likely due to a reduced generalization capability. On the other hand, with a batch size of 4, the parameter updates become unstable and prevent convergence to the optimal solution.

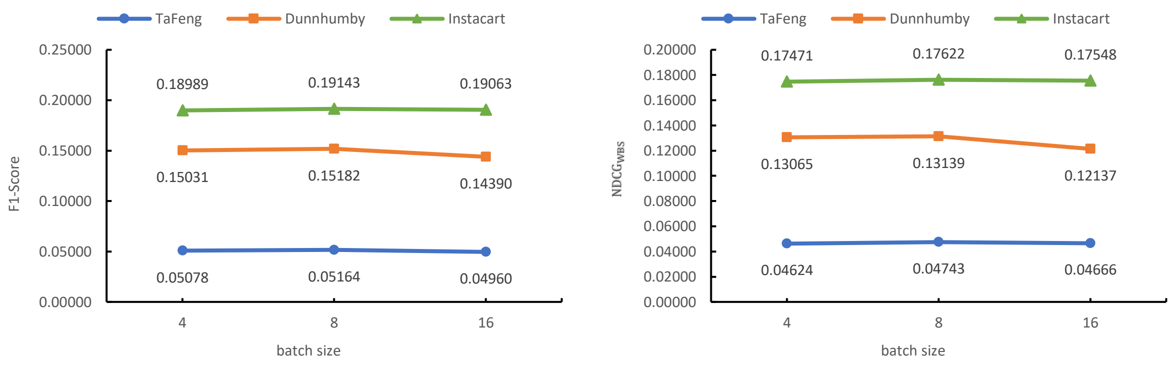


Figure 12: The results of different batch size settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The fourth parameter to test is the embedding dimension. An excessively large embedding dimension may make it difficult for the model to effectively represent item features, while an overly small dimension may fail to capture important feature information, thus impacting recommendation performance. The experimental results in Figure 13 show that the model's performance remains relatively stable when adjusting the embedding size, while the optimal performance achieved at an embedding dimension of 32.

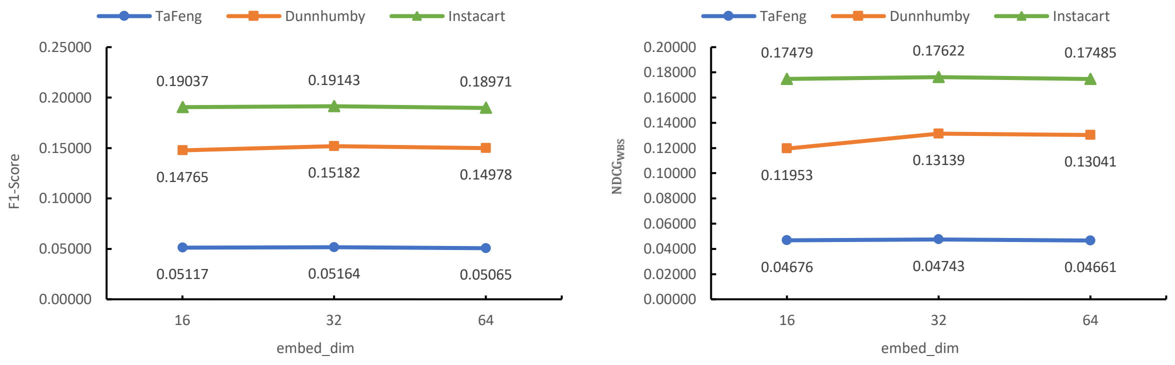


Figure 13: The results of different embedding dimension settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The fifth parameter to be tested is the dimension of the hidden layers in the MLP. When the hidden layer dimension is too small, the neurons may not capture sufficient feature information. Conversely, if the dimension is too large, the model may become overly complex, and the excessive parameters might lead to overfitting the training data. As shown in Figure 14, setting the MLP hidden dimension to 128 yields the best results.

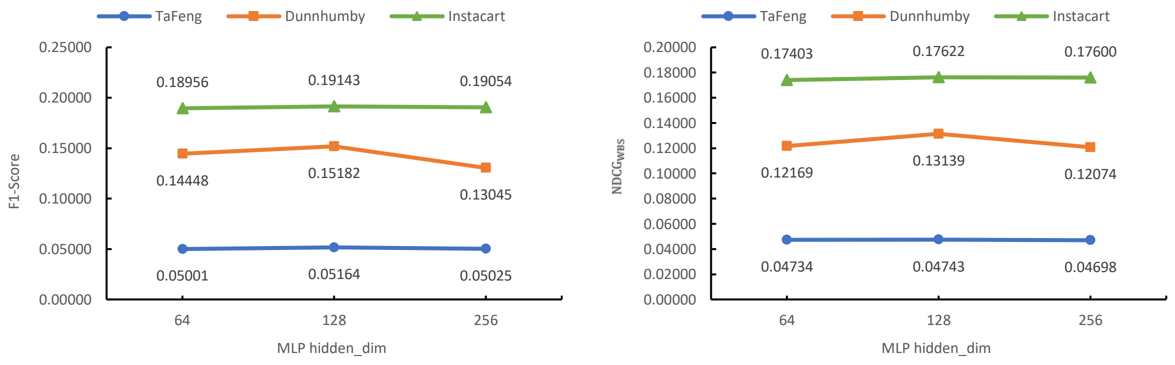


Figure 14: The results of different MLP hidden dimension settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The sixth parameter to be tested is the hidden state dimension of the LSTM. Setting this too small might prevent the model from adequately capturing the variations in basket size. On the other hand, setting it too large could cause the model to overfit the training data, thereby reducing its ability to generalize to new data. The experimental results in Figure 15 indicate that setting the LSTM hidden state dimension to 16 provides the optimal performance. Nonetheless, the influence of the LSTM hidden state dimension on the model's performance is relatively stable overall.

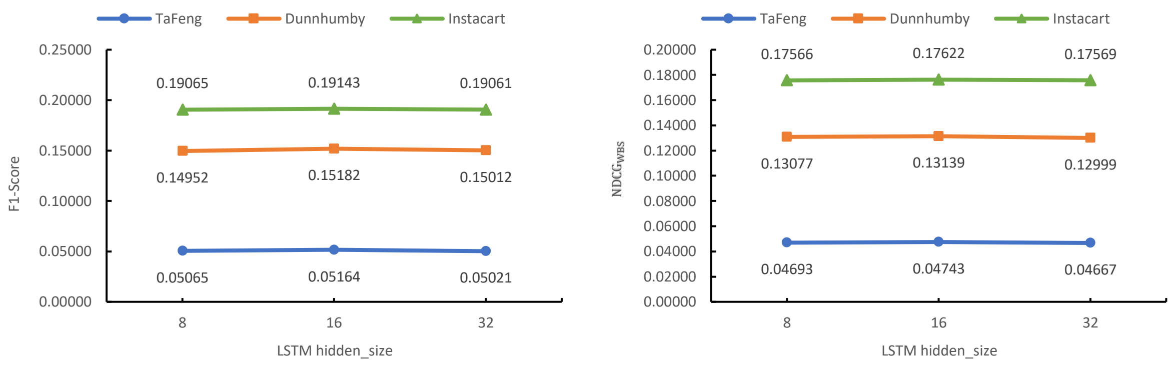


Figure 15: The results of different LSTM hidden size settings on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

The seventh parameter to be tested is the number of multi-heads in the Transformer. According to the results in Figure 16, using 4 heads achieves the best results in the TaFeng and Dunnhumby datasets, while 2 heads perform best in the Instacart dataset. Notably, using only 1 head results in the worst performance, likely because a single head is insufficient to capture the relationships between different feature subspaces. However, too many heads may lead to overfitting the training data, thus reducing predictive performance.

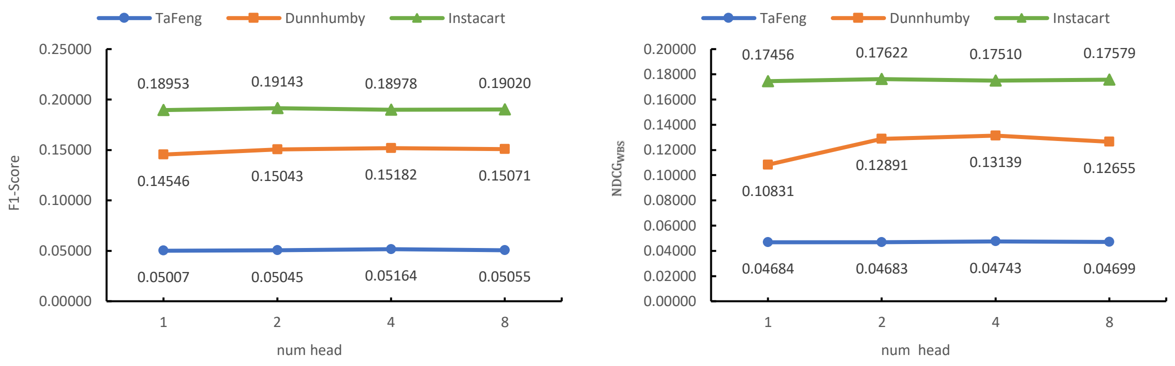


Figure 16: The results of varying the number of Transformer heads on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

Finally, we test the number of layers in the Transformer's encoder. Figure 17 shows that using 4 layers in the Transformer encoder achieves the best performance. Although the differences in performance across different layer counts are not substantial, the model performs the worst with 1 layer. This suggests that when the number of layers is too small, the model can only learn lower-level features, resulting in diminished predictive performance. However, too many layers may cause the model to overfit due to the sparsity of the training data, thereby reducing predictive performance on the test set.

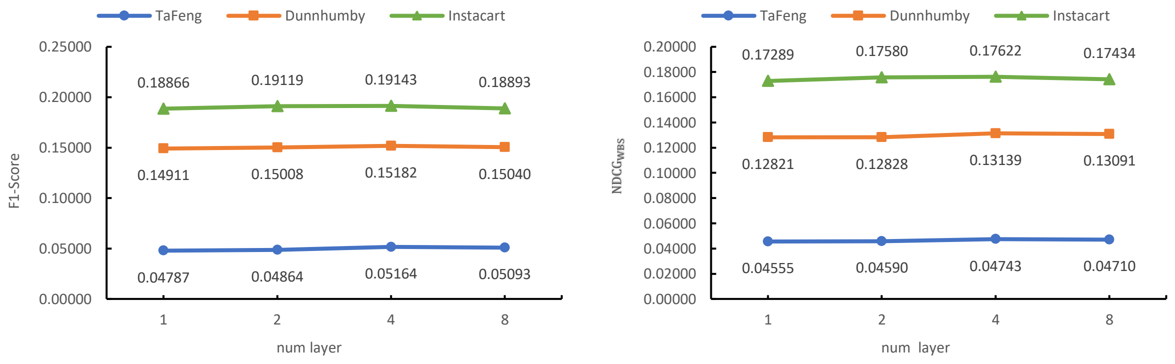


Figure 17: The results of varying the number of Transformer layers on the model's performance in terms of F1-Score and $NDCG_{WBS}$.

4.7 Ablation Study

Ablation studies offer a clearer understanding of the model's workings and the contributions of its individual parts to overall performance. Here, we examine the impact of removing Markov chains, the Confidence matrix, and components related to basket size on model performance. The models and their descriptions are as follows:

- **NBR-WBS w/o size**: The NBR-WBS model without components related to basket size, preventing the model from dynamically adjusting basket size based on user preferences.
- **NBR-WBS w/o mc**: The NBR-WBS model without Markov chains, which hinders the model from capturing specific user behaviors over a short period, consequently affecting comprehensive modeling of user basket sequences.
- **NBR-WBS w/o conf**: The NBR-WBS model without the Confidence matrix, which prevents the model from considering confidence scores between items, thereby making it challenging to select more relevant items.

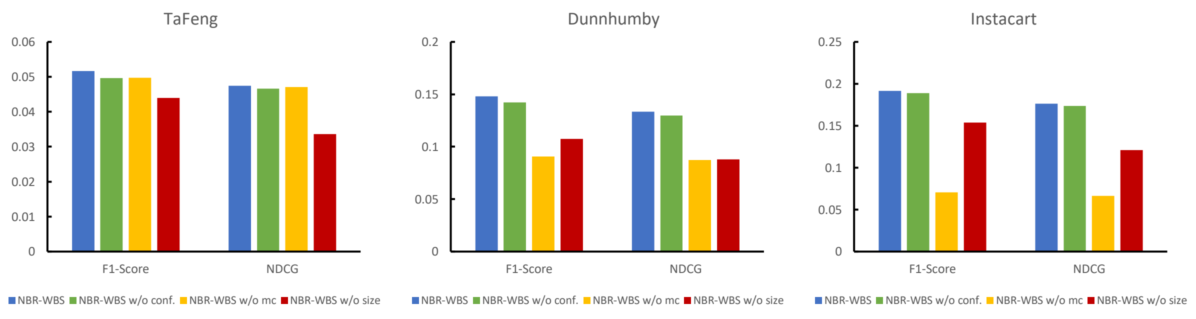


Figure 18: Ablation study of NBR-WBS on different datasets.

Based on the results in Figure 18 (detailed results shown in Appendix Tables 9 and 10), several important observations are noted. Firstly, removing the confidence matrix prevents the model from effectively capturing the complementary and mutually exclusive relationships between items, thereby impacting predictive performance and resulting in a decrease in model performance. Secondly, the impact of removing Markov chains is less noticeable for the TaFeng dataset, but significantly affects performance for the Dunnhumby and Instacart datasets. We speculate that this could be due to users' tendency to purchase recent items more frequently in the Dunnhumby and Instacart datasets. However, regardless of the extent of the performance drop, the results indicate that Markov chains can capture specific short-term behavioral patterns of users and provide a more comprehensive modeling of user preferences. Ultimately, the absence of components related to basket size prevents the model from dynamically adjusting basket sizes based on user characteristics, leading to a significant reduction in model performance.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this study, we propose a novel next-basket recommendation model, NBR-WBS, which incorporates basket size prediction. This model not only predicts the probability of items appearing in the next basket but also forecasts the size of the user's next basket. Our implementation integrates embedding modules, basket preference learning modules, and prediction modules. Specifically, the embedding module employs attention mechanisms to encode the basket content and LSTM networks to encode the basket size, providing a comprehensive understanding of the basket sequences. Subsequently, these embeddings are input into the basket preference learning module for multi-task learning, enhancing the model's generalization capability and producing the final state vector of the basket. This vector is then used in individual components for specific task predictions, allowing the model to dynamically recommend basket sizes based on user characteristics.

Experimental results on three real-world datasets demonstrate that our proposed NBR-WBS outperforms other baselines. This indicates that our model can more accurately capture changes in user basket sizes and long-term preferences, dynamically adjust basket sizes based on different user characteristics, and recommend corresponding items, thereby improving the overall performance of the recommendation system.

Through sensitivity analysis, we further determine the impact of model hyperparameters on performance. The analysis shows that our model exhibits robust performance under different parameter settings and can adapt to different user characteristics and basket scenarios, thereby enhancing the robustness and generality of the recommendation system.

In ablation studies, we further confirm the importance of each component of NBR-WBS. The experimental results indicate that compared to the complete model, the absence of Markov chains fails to effectively model users' short-term specific behaviors. Additionally, the absence of the confidence matrix results in the model's inability to capture the complementarity and exclusivity relationships between items. Moreover, the absence of modules related to basket size prevents the model from timely adjusting basket sizes based on user preferences. These findings emphasize the effectiveness and contributions of these components to the model, further highlighting their indispensability in the entire model framework.

5.2 Future Work

While our proposed NBR-WBS method has shown significant improvements in basket recommendation performance compared to previous methods, there are still limitations and avenues for further exploration.

Firstly, the model proposed in this study primarily focuses on analyzing user purchase behavior sequences while overlooking other auxiliary user behaviors. Future research could explore integrating multimodal data, considering not only purchase history records but also incorporating other modes of data such as user click behaviors, reviews, social media activities, among others, to enhance the performance of the next basket recommendation task.

Secondly, investigating how to integrate more external data sources such as geographical location, time, social relationships, etc., to enrich user features, and further exploring the application of more advanced neural network models such as Graph Neural Networks (GNNs) to handle these complex data. Through GNNs, it's possible to effectively model social relationships between users, geographical similarities, and dynamic changes over time to more comprehensively capture user behaviors and preferences, thus improving the accuracy of the next basket recommendation.

Furthermore, given that this study involves dynamic adjustment of basket size, to ensure fair comparison, we introduced a penalty term for basket size in the NDCG evaluation metric. However, future research could explore more sophisticated evaluation methods to more accurately reflect the impact of differences between real and predicted basket sizes on recommendation results.

Lastly, future research directions could consider introducing time correlation into basket size prediction. For example, incorporating holidays, seasons, or other time factors into the model consideration to dynamically adjust basket size based on specific times. Such time correlations can more accurately reflect changes in user purchasing behaviors and preferences at different times, thus enhancing the personalization and accuracy of the recommendation system.

REFERENCES

[1] Z. Zamanzadeh Darban and M. H. Valipour, “GHRS: Graph-based hybrid recommendation system with application to movie recommendation,” *Expert Systems with Applications*, vol. 200, p. 116850, Aug. 2022, doi: 10.1016/j.eswa.2022.116850.

[2] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, “Contextual and Sequential User Embeddings for Large-Scale Music Recommendation,” in *Proceedings of the 14th ACM Conference on Recommender Systems*, in RecSys ’20. New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 53–62. doi: 10.1145/3383313.3412248.

[3] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, “DRN: A Deep Reinforcement Learning Framework for News Recommendation,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ’18*, Lyon, France: ACM Press, 2018, pp. 167–176. doi: 10.1145/3178876.3185994.

[4] X. Li, X. Wang, X. He, L. Chen, J. Xiao, and T.-S. Chua, “Hierarchical Fashion Graph Network for Personalized Outfit Recommendation,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Virtual Event China: ACM, Jul. 2020, pp. 159–168. doi: 10.1145/3397271.3401080.

[5] P. Nitu, J. Coelho, and P. Madiraju, “Improvising personalized travel recommendation system with recency effects,” *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 139–154, Sep. 2021, doi: 10.26599/BDMA.2020.9020026.

[6] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural Attentive Session-based Recommendation,” in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, in CIKM ’17. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1419–1428. doi: 10.1145/3132847.3132926.

[7] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-Based Recommendation with Graph Neural Networks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, Art. no. 01, Jul. 2019, doi: 10.1609/aaai.v33i01.3301346.

[8] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, and T. Tan, “TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, in SIGIR ’20. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 1921–1924. doi: 10.1145/3397271.3401319.

[9] W.-C. Kang and J. McAuley, “Self-Attentive Sequential Recommendation,” in *2018 IEEE International Conference on Data Mining (ICDM)*, Jan. 2018, pp. 197–206. doi: 10.1109/ICDM.2018.00035.

[10] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, “BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer,” in

Proceedings of the 28th ACM International Conference on Information and Knowledge Management, in CIKM '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1441–1450. doi: 10.1145/3357384.3357895.

[11] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*, in WWW '10. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 811–820. doi: 10.1145/1772690.1772773.

[12] H. Hu, X. He, J. Gao, and Z.-L. Zhang, "Modeling Personalized Item Frequency Information for Next-basket Recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, in SIGIR '20. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 1071–1080. doi: 10.1145/3397271.3401066.

[13] Y. Qin, P. Wang, and C. Li, "The World is Binary: Contrastive Learning for Denoising Next Basket Recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, in SIGIR '21. New York, NY, USA: Association for Computing Machinery, Jul. 2021, pp. 859–868. doi: 10.1145/3404835.3462836.

[14] G. Faggioli, M. Polato, and F. Aiolli, "Recency Aware Collaborative Filtering for Next Basket Recommendation," in *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, in UMAP '20. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 80–87. doi: 10.1145/3340631.3394850.

[15] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning Hierarchical Representation Model for NextBasket Recommendation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, in SIGIR '15. New York, NY, USA: Association for Computing Machinery, Aug. 2015, pp. 403–412. doi: 10.1145/2766462.2767694.

[16] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A Dynamic Recurrent Model for Next Basket Recommendation," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, in SIGIR '16. New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 729–732. doi: 10.1145/2911451.2914683.

[17] D.-T. Le, H. W. Lauw, and Y. Fang, "Correlation-Sensitive Next-Basket Recommendation," presented at the the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macau, China, Aug. 2019. Accessed: Sep. 22, 2023. [Online]. Available: https://www.ijcai.org/proceedings/2019/389

[18] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *Journal of Machine Learning Research*, vol. 6, 2005.

[19] W. Gu, S. Dong, and Z. Zeng, "Increasing recommended effectiveness with markov

chains and purchase intervals," *Neural Comput & Applic*, vol. 25, no. 5, pp. 1153–1162, Oct. 2014, doi: 10.1007/s00521-014-1599-8.

[20] R. He and J. McAuley, "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation," Sep. 2016, doi: 10.48550/arXiv.1609.09152.

[21] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," Jan. 2013, doi: 10.48550/arXiv.1301.7363.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, Hong Kong Hong Kong: ACM, Apr. 2001, pp. 285–295. doi: 10.1145/371920.372071.

[23] O. Barkan and N. Koenigstein, "ITEM2VEC: Neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Vietri sul Mare, Salerno, Italy: IEEE, Sep. 2016, pp. 1–6. doi: 10.1109/MLSP.2016.7738886.

[24] M. SALAMPASIS, T. SIOMOS, A. KATSALIS, K. DIAMANTARAS, K. CHRISTANTONIS, M. DELIANIDI, and I. KARAVELI, "Comparison of RNN and Embeddings Methods for Next-item and Last-basket Session-based Recommendations," in *Proceedings of the 2021 13th International Conference on Machine Learning and Computing*, in ICMLC '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 477–484. doi: 10.1145/3457682.3457755.

[25] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, in ICML'14. Beijing, China: JMLR.org, Jun. 2014, p. II-1188-II–1196.

[26] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi: 10.1109/MC.2009.263.

[27] H. Wang, "ZeroMat: Solving Cold-start Problem of Recommender System with No Input Data," in *2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, Sep. 2021, pp. 102–105. doi: 10.1109/ICISCAE52414.2021.9590668.

[28] H. Wang, "PoissonMat: Remodeling Matrix Factorization using Poisson Distribution and Solving the Cold Start Problem without Input Data," in *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, Aug. 2022, pp. 245–249. doi: 10.1109/MLISE57402.2022.00055.

[29] N. P. Kumar and Z. Fan, "Hybrid User-Item Based Collaborative Filtering," *Procedia Computer Science*, vol. 60, pp. 1453–1461, Jan. 2015, doi: 10.1016/j.procs.2015.08.222.

[30] T. Bai, J.-Y. Nie, W. X. Zhao, Y. Zhu, P. Du, and J.-R. Wen, "An Attribute-aware Neural Attentive Model for Next Basket Recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, Ann Arbor MI

USA: ACM, Jun. 2018, pp. 1201–1204. doi: 10.1145/3209978.3210129.

[31] H. Hu and X. He, “Sets2Sets: Learning from Sequential Sets with Neural Networks,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 2019, pp. 1491–1499. doi: 10.1145/3292500.3330979.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS’17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.

[33] J. Yang, J. Xu, J. Tong, S. Gao, J. Guo, and J. Wen, “Pre-training of Context-aware Item Representation for Next Basket Recommendation,” Apr. 2019, doi: 10.48550/arXiv.1904.12604.

[34] M. Li, M. Ariannezhad, A. Yates, and M. de Rijke, “Masked and Swapped Sequence Modeling for Next Novel Basket Recommendation in Grocery Shopping,” in *Proceedings of the 17th ACM Conference on Recommender Systems*, in RecSys ’23. New York, NY, USA: Association for Computing Machinery, Sep. 2023, pp. 35–46. doi: 10.1145/3604915.3608803.

[35] W. Sun, R. Xie, J. Zhang, W. X. Zhao, L. Lin, and J.-R. Wen, “Generative Next-Basket Recommendation,” in *Proceedings of the 17th ACM Conference on Recommender Systems*, in RecSys ’23. New York, NY, USA: Association for Computing Machinery, Sep. 2023, pp. 737–743. doi: 10.1145/3604915.3608823.

[36] Z. Liu, X. Li, Z. Fan, S. Guo, K. Achan, and P. S. Yu, “Basket Recommendation with Multi-Intent Translation Graph Neural Network,” in *2020 IEEE International Conference on Big Data (Big Data)*, Atlanta, GA, USA: IEEE, Dec. 2020, pp. 728–737. doi: 10.1109/BigData50022.2020.9377917.

[37] T. Liu and B. Liu, “Next basket recommendation based on graph attention network and transformer,” *J. Phys.: Conf. Ser.*, vol. 2303, no. 1, p. 012023, Jul. 2022, doi: 10.1088/1742-6596/2303/1/012023.

[38] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 281–305, Feb. 2012.

[39] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, in NIPS’12. Red Hook, NY, USA: Curran Associates Inc., Dec. 2012, pp. 2951–2959.

[40] R. N. and D. Ali, “Factors affecting consumer buying behavior,” Sep. 2016.

[41] A. J. Badgaiyan, A. Verma, and S. Dixit, “Impulsive buying tendency: Measuring important relationships with a new perspective and an indigenous scale,” *IIMB Management Review*, vol. 28, no. 4, pp. 186–199, Dec. 2016, doi: 10.1016/j.iimb.2016.08.009.

[42] W.-Y. Lee, “Association-Based Sequential Basket Recommendation.” Accessed: Mar. 13, 2024. [Online]. Available: https://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/login?o=dnclcdr&s=id=%22111NCU05396072%22.&searchmode=basic

[43] Y. Goldberg and O. Levy, “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method,” Feb. 2014, doi: 10.48550/arXiv.1402.3722.

[44] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Jan. 1997, doi: 10.1162/neco.1997.9.8.1735.

[45] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, “Sequential recommender system based on hierarchical attention network,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, in IJCAI’18. Stockholm, Sweden: AAAI Press, Jul. 2018, pp. 3926–3932.

APPENDIX

Table 6: Detailed comparison results for the TaFeng dataset.

TaFeng								
	F1-socre				$NDCG_{WBS}$			
Model	F1-score@5	F1-score@10	F1-score@20	F1-score@40	$NDCG_{WBS}$@5	$NDCG_{WBS}$@10	$NDCG_{WBS}$@20	$NDCG_{WBS}$@40
FPMC	0.03787	0.03430	0.02625	0.01894	0.03785	0.03190	0.02017	0.01157
FPMC(Avg.)	**0.02934**				**0.02537**			
SHAN	0.03755	0.03324	0.02905	0.02789	0.03806	0.03306	0.02360	0.01558
SHAN(Avg.)	**0.03193**				**0.02757**			
HRM	0.03967	0.03589	0.02953	0.02118	0.03882	0.03461	0.02487	0.01599
HRM(Avg.)	**0.03156**				**0.02857**			
DREAM	0.04096	0.03928	0.03514	0.03089	0.03988	0.03537	0.02559	0.01653
DREAM(Avg.)	**0.03657**				**0.02934**			
Beacon	0.04522	0.04274	0.03880	0.03099	0.04313	0.03877	0.02779	0.01681
Beacon(Avg.)	**0.03944**				**0.03162**			
CLEA	0.04413	0.04387	0.04123	0.03813	0.04366	0.04089	0.03092	0.02052
CLEA(Avg.)	**0.04184**				**0.03400**			
Sets2Sets	0.05208	0.05416	0.05095	0.04055	0.04561	0.04506	0.03506	0.02160
Sets2Sets(Avg.)	**0.04944**				**0.03683**			
TIFUKNN	0.05154	0.05505	0.05116	0.03806	0.04382	0.04405	0.03419	0.02046
TIFUKNN(Avg.)	**0.04895**				**0.03563**			
NBR-WBS	**0.05164**				**0.04743**			

Table 7: Detailed comparison results for the Dunnhumby dataset.

Dunnhumby								
	F1-socre				$NDCG_{WBS}$			
Model	F1-score@5	F1-score@10	F1-score@20	F1-score@40	$NDCG_{WBS}$@5	$NDCG_{WBS}$@10	$NDCG_{WBS}$@20	$NDCG_{WBS}$@40
FPMC	0.08648	0.08693	0.08248	0.07238	0.07264	0.08047	0.07327	0.04802
FPMC(Avg.)	**0.08206**				**0.06860**			
SHAN	0.08340	0.08022	0.07843	0.06656	0.06996	0.07536	0.06846	0.04460
SHAN(Avg.)	**0.07715**				**0.06460**			
HRM	0.09251	0.09319	0.08612	0.07164	0.07752	0.08498	0.07498	0.04836
HRM(Avg.)	**0.08586**				**0.07146**			
DREAM	0.08197	0.08145	0.07636	0.06448	0.06871	0.07466	0.06697	0.04331
DREAM(Avg.)	**0.07606**				**0.06341**			
Beacon	0.08186	0.08303	0.07893	0.06734	0.06869	0.07619	0.06794	0.04421
Beacon(Avg.)	**0.07779**				**0.06426**			
CLEA	0.12379	0.12019	0.10628	0.08509	0.10495	0.11126	0.09388	0.05899
CLEA(Avg.)	**0.10884**				**0.09227**			
Sets2Sets	0.14139	0.15434	0.14535	0.11364	0.11535	0.13040	0.11763	0.07399
Sets2Sets(Avg.)	**0.13868**				**0.10934**			
TIFUKNN	0.15890	0.16853	0.14977	0.11162	0.12520	0.14117	0.12391	0.07589
TIFUKNN(Avg.)	**0.14721**				**0.11654**			
NBR-WBS	**0.15182**				**0.13139**			

Table 8: Detailed comparison results for the Instacart dataset.

Instacart								
	F1-socre				$NDCG_{WBS}$			
Model	F1-score@5	F1-score@10	F1-score@20	F1-score@40	$NDCG_{WBS}$@5	$NDCG_{WBS}$@10	$NDCG_{WBS}$@20	$NDCG_{WBS}$@40
FPMC	0.06200	0.06854	0.06620	0.06019	0.04717	0.05949	0.05574	0.03813
FPMC(Avg.)	**0.06423**				**0.05013**			
SHAN	0.06030	0.06663	0.06502	0.05613	0.04619	0.05826	0.05467	0.03626
SHAN(Avg.)	**0.06202**				**0.04885**			
HRM	0.07691	0.08760	0.08486	0.07266	0.06461	0.07716	0.06918	0.04554
HRM(Avg.)	**0.08051**				**0.06412**			
DREAM	0.06346	0.06917	0.06830	0.06081	0.04923	0.06038	0.05654	0.03807
DREAM(Avg.)	**0.06543**				**0.05105**			
Beacon	0.06048	0.06792	0.06562	0.05900	0.04636	0.05829	0.05442	0.03749
Beacon(Avg.)	**0.06326**				**0.04914**			
CLEA	0.11945	0.12123	0.10936	0.08916	0.10104	0.11057	0.09447	0.06004
CLEA(Avg.)	**0.10980**				**0.09153**			
Sets2Sets	0.17594	0.19471	0.17751	0.13749	0.13167	0.15005	0.12315	0.07612
Sets2Sets(Avg.)	**0.17141**				**0.12025**			
TIFUKNN	0.21365	0.20814	0.17429	0.11877	0.16400	0.17400	0.13479	0.07711
TIFUKNN(Avg.)	**0.17871**				**0.13747**			
NBR-WBS	**0.19143**				**0.17622**			

Table 9: Comparison of Ablation Experiment Results Based on F1-Score Evaluation

	TaFeng	Dunnhumby	Instacart
NBR-WBS w/o conf.	0.04960	0.14239	0.18873
NBR-WBS w/o mc	0.04974	0.09064	0.07078
NBR-WBS w/o size	0.04398	0.10760	0.15364
NBR-WBS	**0.05164**	**0.15182**	**0.19143**

Table 10: Comparison of Ablation Experiment Results Based on $NDCG_{WBS}$ Evaluation

	TaFeng	Dunnhumby	Instacart
NBR-WBS w/o conf.	0.04661	0.12971	0.17363
NBR-WBS w/o mc	0.04711	0.08739	0.06658
NBR-WBS w/o size	0.03361	0.08790	0.12108
NBR-WBS	**0.04743**	**0.13139**	**0.17622**