國 立 中 央 大 學

資 訊 管 理 學 系

碩 士 論 文

基於個性化項目頻率和時間注意力的下一個購物籃推薦混合模型

PIFTA4Rec: A Hybrid Model of Personalized Item Frequency and Temporal Attention for Next-basket Recommendation

研 究 生：林莉庭

指導教授：陳彥良 博士

中 華 民 國　113　年　6　月

基於個性化項目頻率和時間注意力的下一個購物籃推薦混合模型

摘要

在資訊量爆炸的數位時代中，推薦系統扮演著至關重要的角色。它們不僅緩解資訊過載的問題，還能顯著提升用戶體驗並增加電子商務的收入潛力。因此，推薦系統在多種場景中得到廣泛應用，尤其是在下一個購物籃推薦任務上。然而，傳統推薦方法傾向於僅關注短期的局部關係，而忽略了長期的全局關係。雖然基於神經網絡的推薦方法已改善了推薦性能，但仍存在一些侷限與進步空間，特別是在利用個性化項目頻率（PIF）進行推薦。PIF 是一種基於用戶對商品重複購買行為的衡量指標，在多個推薦任務中已證明其有效性和價值。本研究認為，將這種個人化重複購買行為的向量模型結合進現有的神經網絡模型中，將有助於進一步提升推薦效率與準確性。有鑑於此，本研究提出了一個結合個性化項目頻率資訊與時間注意力機制的混合模型，稱為 PIFTA4Rec。該模型整合傳統資料探勘和當代深度學習技術的優勢，充分利用 KNN 的局部敏感性對帶有 PIF 資訊的向量數據點進行精確預測。同時，結合了考慮每次購買時間的時間注意力機制和 Transformer 的多頭注意力機制，以深入學習用戶與項目之間的複雜關聯資訊。最終，本模型整合兩個推薦預測過程的結果，以生成最終的推薦結果。在本研究中，我們將這樣的技術融合模型 PIFTA4Rec 在兩個真實世界資料集中進行了實驗。結果顯示，本研究提出之方法在推薦性能上優於現有的其他下一個購物籃推薦方法，顯著證實了本模型的有效性與穩定性。

關鍵詞：下一個購物籃推薦、個性化項目頻率、深度學習、K Nearest Neighbor、Item2Vec

PIFTA4Rec: A Hybrid Model of Personalized Item Frequency and Temporal Attention for Next-basket Recommendation

ABSTRACT

In the digital age of information explosion, recommendation systems play a crucial role. They not only alleviate the problem of information overload but also significantly enhance user experience and increase the revenue potential of e-commerce. Consequently, recommendation systems are widely applied in various scenarios, especially in the task of predicting the next basket. However, traditional recommendation methods tend to focus only on short-term local relationships, overlooking long-term global interactions. Although recommendation methods based on neural networks have improved performance, there are still limitations and room for improvement, particularly in the use of Personalized Item Frequency (PIF). PIF, a metric based on the user's repeated purchase behavior, has proven its effectiveness and value in numerous recommendation tasks. This study suggests that integrating this vector model of personalized repeated purchase behavior into existing neural network models will further enhance the efficiency and accuracy of recommendations. In light of this, the study proposes a hybrid model that combines Personalized Item Frequency information with a Temporal Attention mechanism, called PIFTA4Rec. This model integrates the advantages of traditional data mining and contemporary deep learning techniques, leveraging the local sensitivity of K Nearest Neighbor (KNN) to precisely predict vector data points containing PIF information. Simultaneously, it incorporates a temporal attention mechanism that considers the timing of each purchase, along with the multi-head attention mechanism of Transformers, to deeply learn the complex associative information between users and items. Ultimately, the model integrates the results of these two recommendation prediction processes to generate the final recommendation outcomes. In this study, the hybrid model PIFTA4Rec was experimentally tested on two real-world datasets. The results demonstrated that the proposed method outperformed other existing

next-basket recommendation methods in terms of recommendation performance, significantly confirming the model's effectiveness and stability.

Keywords: Next Basket Recommendation、Personalized Item Frequency、Deep Learning、K Nearest Neighbor、Item2Vec

誌謝

兩年的碩士生涯轉瞬即逝，在這段旅程中，我得到了許多人的幫助和鼓勵，使我得以順利完成我的學業。在此，我要向所有支持我的人致以最誠摯的感謝。

首先，我衷心感謝我的指導教授，陳彥良教授，他在整個研究和論文撰寫過程中給予我無數的指導、鼓勵和支持。另外，我也要特別感謝張嘉玲教授，她的寶貴建議和幫助讓我在研究上有了更深入的思考和突破。

其次，我要感謝我的實驗室同學郭羿德、姜道宣和李泳輝，他們時常在我實驗遇到困難時伸出援手。在撰寫論文的過程中，我們一同在實驗室努力，相互鼓勵和支持，使我在努力的路上並不感到孤單。

此外，我要感謝所有在碩士班中結識的朋友，包括張浩玫、葉宜潔、林鈺倫、王聖智、黃永璿、翁賢灝、蔣銘皓、李育慈、黃郁涵、鄭心愉、柳成彥、張華哲、盧俊吉和李孟儒。在課堂上與課後，我們一同學習與探索了許多知識，同時也在休閒時刻創造無限歡樂。這些友誼在學業上為我提供了支持，在生活中帶來了歡笑，成為我珍貴的回憶。對於那些未能一一提及的同學和朋友，我同樣衷心感謝你們的陪伴和支持。每一段共同度過的時光都留下了難忘的記憶，對我的學業和生活都有著不可或缺的影響。

最後，我要特別感謝我的家人，他們一直都是我最堅強的後盾。在我求學期間，家人的關心電話常常給予我鼓勵和安慰。每次回家，都能深切感受到他們的溫暖和愛，這使我返回學業之路時都滿載力量。如今，隨著碩士生涯的結束，我將帶著這兩年的經驗與回憶，在未來努力探索自我，並為社會貢獻我的力量。

林莉庭 謹誌於

國立中央大學資訊管理學系研究所

中華民國一一三年六月

Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 Research Background

In today's era of technological convenience, e-commerce platforms have become increasingly prevalent, with many industries gradually developing Online-to-Offline (O2O) retailers. Under this trend, sellers can post any product information online without being constrained by time and space, thereby expanding their audience reach. However, with the explosive growth of information, the issue of information overload has become more severe. In this context, the emergence of recommendation systems has become crucial. A recommendation system is an information retrieval technology designed to analyze users' historical behaviors and preferences to recommend content or products that they may find interesting. For sellers, recommendation systems help in understanding users' needs and preferences more accurately, thereby improving marketing strategies to effectively reach target audiences and potential customers, increasing sales opportunities, and saving resources. For buyers, they offer a personalized shopping experience, helping them quickly find products of interest and discover new, previously unconsidered options, thus enriching their shopping choices. Therefore, recommendation systems have been widely applied in various scenarios, such as movie recommendations[1], news recommendations[2], music recommendations[3], and social recommendations[4], and more.

The research directions of recommendation systems can mainly be divided into two categories: rating prediction[5] and Top-N prediction[6]. Rating prediction methods focus on predicting a specific user's potential rating for an item based on their historical rating records. In contrast, Top-N prediction concentrates on recommending a ranked list of the top N items that a user is most likely to be interested in, based on their personal preferences and behaviors. It is worth noting that compared to rating prediction, Top-N prediction occupies a more

prominent position in current practical applications and research[7-9]. Therefore, this study also chooses to utilize the Top-N prediction method. The top N items are considered as the most likely candidates for the user's next basket combination, and recommendations are made accordingly.

Existing Top-N prediction tasks can be categorized based on their objectives as follows. Top-N Items Recommendation involves predicting and providing a ranked list of the top N items that the user is most interested in based on their purchase history[10]. Next-Item Recommendation predicts the next item with which the user will interact, considering the sequential buying behavior during the shopping process, based on the user's historical behavior sequence[11]. Within-Basket Recommendation provides recommendations for additional items based on the user's current basket, aiming to ensure the basket's content is more complete and coherent[12]. Next-Basket Recommendation, as shown in Figure 1, predicts the combination of items in the user's next basket based on their past basket history sequence[13]. Notably, in recent years, the research direction of Next-Basket Recommendation (NBR) has received widespread attention and continuous evolution. Numerous researchers have been dedicated to this field, exploring ways to enhance its predictive capability and application scope[14, 15]. This study also focuses on this direction, aiming to provide deeper insights and solutions for Next-Basket Recommendation tasks.

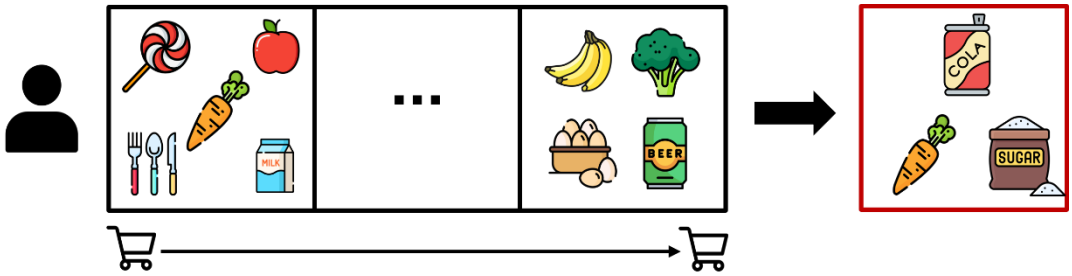

Figure 1 Next-Basket Recommendation

Traditional recommendation technologies, such as collaborative filtering[16], content filtering[17], matrix factorization[18], etc., have achieved a lot of research results, especially collaborative filtering is widely used[19-21]. However, despite these methods' ability to

effectively capture users' static interests, they gradually come into question due to their lack of consideration for temporal and sequential factors, which prevents them from fully understanding the complex relationships and dynamic information between users and items, such as users' continuous preference information. Consequently, modern session-based and sequence-based approaches have improved upon these limitations. Session-based recommendation treats all user behaviors within a single session as a sequence of item sets, focusing on the order of actions (e.g., browsing, clicking, adding to the cart) and the time intervals between them within a session. This approach aims to determine the context and user intent within the session to predict the next action[22]. On the other hand, sequence-based recommendations treat each user's behavior as a sequence over a longer period, such as a month or a year's purchase history. This approach focuses on analyzing long-term user preferences and trends to predict future actions[23]. Since sequence-based recommendation methods can effectively analyze users' behavior sequences over a longer period, they better understand users' enduring interests rather than just their short-term interests. This approach offers significant value for personalized recommendations[24-26]. Consequently, this study adopts sequence-based recommendation methods to gain a deeper understanding of users' long-term preferences for more personalized recommendations.

1.2 Research Motivation

In contemporary personalized recommendation research, whether based on session or sequence methods, many model architectures are designed using RNNs[27] or Transformers[28]. The rationale is that RNNs, with their recurrent structure, effectively capture the order and time-related dependencies within time series and handle long-term dependencies, thereby better understanding the impact of past behaviors on future actions. RNNs can store past information and use it in subsequent predictions, thus fully considering the context when processing sequence data. On the other hand, Transformers handle time series with their

attention mechanism, considering dependencies between positions in parallel and automatically adjusting weights based on the importance of different positions. This allows them to consider both the context within and outside the sequence, better capturing the associations between positions.

On the other hand, theoretical and empirical research in economics and psychology has shown that despite the highly complex psychological processes involved in human purchasing behavior, the decision to re-consume an item is largely influenced by its recent consumption history[29, 30]. Inspired by this perspective, numerous studies have begun to incorporate information on users' repeat purchase behaviors to optimize sequence-based and session-based recommendation tasks[31-33]. Among these, Personalized Item Frequency (PIF) based on repeat purchase behaviors provides two primary purchasing modes for next-basket recommendation: the Repeat Purchase Mode and the Collaborative Purchase Mode[34]. As shown in Figure 2, the Repeat Purchase Mode assesses whether an item from the target user's past baskets will reappear in their next shopping. At the same time, the Collaborative Purchase Mode evaluates whether an item from similar users' past baskets will appear in the target user's next shopping.

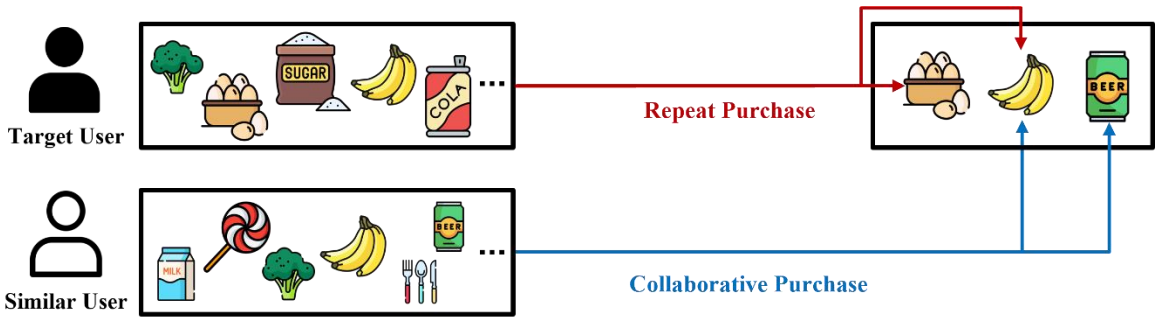

Figure 2 Repeat Purchase Pattern and the Collaborative Purchase Pattern

Incorporating features like PIF into basket recommendation tasks has been shown to enhance recommendation effectiveness in several studies[14, 15, 35, 36] Among these, the TIFU-KNN method based on the traditional Vector Space Model (VSM) utilizes PIF

information and the weighted sum of vectors with time decay, combined with the K-Nearest Neighbors (KNN) approach, to predict the next basket. In such a vector space model, we can objectively assess the extent of consumers' repeat purchase behaviors for each item. However, current neural network-based recommendation models not only fail to integrate with such vector models but also perform worse than TIFU-KNN in recommendation performance. Thus, TIFU-KNN demonstrates strong capability in using PIF within the vector space model for recommendation tasks. It is noteworthy, however, that despite its excellent recommendation results, there is room for improvement in performance due to its non-learning parameters. On the other hand, a series of experiments have indicated that although complex models based on RNNs possess powerful representation capabilities, their performance in learning PIF information and making recommendations falls short of expectations[34]. As for the effectiveness of learning through attention mechanisms in Transformers, there is currently no research demonstrating its impact.

1.3 Research Objectives

Therefore, given the outstanding performance of PIF information in the traditional vector space model TIFU-KNN, and the previous ineffective results of learning PIF information using RNN technology, this study proposes a hybrid model for next-basket recommendation tasks that integrates PIF information with Temporal Attention, termed PIFTA4Rec (Personalized Item Frequency and Temporal Attention for Next-basket Recommendation). Considering that recent purchasing behaviors are more likely to trigger repeat purchases compared to behaviors from long ago, PIFTA4Rec integrates traditional data mining with modern deep learning techniques. It leverages Temporal Attention with time-based mechanisms and Self-Attention from Transformers to learn complex relational information between users and items from their original purchase sequences. On the other hand, it employs KNN methods and attention mechanisms to utilize PIF information effectively.

Overall, PIFTA4Rec maximizes the strengths of both deep learning's attention mechanisms and the traditional KNN method. This combination not only ensures that the model captures deep features and relationships within the data but also leverages the local sensitivity of KNN for more precise predictions of approximate and similar data points. Furthermore, the attention mechanisms of deep learning provide superior representational learning capabilities on large-scale data, while KNN ensures accuracy in detail and individual cases. This study aims to improve upon the shortcomings of past recommendation methods while continuing to leverage the advantages brought by PIF information to NBR tasks. By integrating the currently popular attention mechanisms with traditional vector space models, we explore whether such an architecture can more effectively utilize PIF information, focusing on key details to deliver more outstanding recommendation performance.

The contributions of this study are as follows:

- Introducing a novel hybrid model that combines traditional vector space models with contemporary neural network models.
- Integrating traditional data retrieval techniques with current deep learning technologies to capture complex feature information between users and items.
- Considering personalized item frequency information and temporal preferences in next-basket sequence recommendations.
- Improving the method of manually setting parameters in current best-performing models by adopting a learning approach to obtain optimal parameter values.
- Enhancing the recommendation performance of the next-basket recommendation model to achieve further improvements.

The remaining structure of this study is as follows:

- Section 2 will conduct a literature review.

- Section 3 will detail the proposed model architecture.
- Section 4 will design experiments for the proposed model.
- Section 5 will analyze the experimental results.
- Section 6 will summarize the findings and discuss directions for future improvements.

2. RELATED WORK

With the rapid development of e-commerce, the Next-Basket Recommendation has garnered increasing attention from scholars in recent years. Many studies have focused on addressing the challenges of this recommendation task, with repeated purchase behavior and personalized item frequency becoming increasingly significant. In this section, we will briefly review the methods previously used in the next-basket recommendation task and the challenges encountered in learning item frequencies. We will divide this discussion into several subsections: traditional methods, neural network approaches, and personalized item frequency, as well as the difficulties encountered in learning item frequency.

2.1 Traditional Methods

In this section, we will first review the relevant literature on traditional methods used in past next-basket recommendation tasks.

2.1.1 Sequential Methods

In reality, users' basket histories are considered sequential data, with each shopping event representing a time step, and users' preferences and purchasing behaviors possibly changing over time. Consequently, sequence-based methods are employed to explore the temporal order of baskets to predict future purchases. In traditional sequence-based approaches, Markov chains are commonly used. These chains learn a probability transition matrix based on the current state's previous item or basket state and then use this matrix to predict the next item or basket. For instance, Wanrong Gu et al. [37] proposed a model that combines personalized Markov chains with purchase time intervals. It utilizes a Hidden Markov Model (HMM) and makes predictions through sequence connections between adjacent clicks. Guy Shani et al. [38] introduced a recommendation system based on Markov Decision Processes (MDPs). This

system defines a user's state through a series of ordered items (such as products selected by the user sequentially) and creates transition functions based on these states to predict the user's next choice, considering the long-term effects and expected value of each recommendation.

However, Using Markov chains for a recommendation has several drawbacks. Firstly, it overlooks the complex relationships between baskets or items, making it difficult to capture products that users frequently purchase together. Secondly, due to the Markovian nature which only considers the current state, it fails to fully reflect users' long-term preferences and past purchasing behaviors, potentially leading to poor performance in long-term sequence prediction.

2.1.2 Collaborative Filtering Methods

User-Based Collaborative Filtering is a method that recommends the next item or basket by calculating similarities between users. The core idea is that if two users have shown similar preferences in their past behaviors, they are likely to have similar preferences in the future, so items favored by one user can be recommended to the other. This method constructs a user-item rating matrix; however, since each user usually rates only a few items, this matrix often has a sparsity issue. Matrix Factorization (MF) [39] has been proposed to address this problem. Factorization Machines (FM) [40] extends the concept of matrix factorization to all features, using latent vectors to represent each feature and capable of modeling interactions between any two features. Compared to matrix factorization, FM offers greater flexibility and can handle more application scenarios and feature combinations.

Related research, such as the method proposed by Shi et al. [41], utilizes factorization machines to integrate user interaction patterns from different domains as auxiliary information. By treating interactions of certain item types as specific domains, it uses auxiliary domain information to optimize recommendations in the target domain. Yuya Miyamoto et al. [42] employ factorization machines to develop a method based on sales data for classifying customers in each store by the value they seek, using this information to predict user purchasing

behavior, thereby enhancing the efficiency of direct mail (DM) delivery.

Item-based Collaborative Filtering calculates the similarities between items to recommend the next item or basket. For example, Suyun Wei et al. [43] proposed a collaborative filtering algorithm called CICF, which combines item categories and interestingness to calculate item similarities. This approach uses a specific similarity matrix and interestingness assessment to ensure that items only become neighbors if they have been commonly rated by enough users and the similarity in ratings is high, thus addressing the shortcomings of traditional methods for calculating user similarities. In contrast, Content-based Recommendation relies on the content features of items themselves (such as text in articles, genres, directors of movies, etc.) for recommendations. For instance, Barman et al. [44] se dynamic content-based filtering methods to generate item recommendations and continuously monitor changes in user shopping behavior. Although item-based collaborative filtering and content-based recommendations are effective in certain application scenarios, they share a common limitation. Their recommendation strategies are implemented in a flat, static manner, considering only a cross-sectional view of time, and not making recommendations along a timeline, failing to account for the dynamic and long-term sequences. As a result, they are unable to capture the evolution of users' long-term behavioral preferences and interests.

Popularity-based Recommendation is based on a simple assumption: the most popular or frequently interacted items are likely to be favored by new users or the majority of users. Popularity is usually determined by counting the number of interactions with an item, such as clicks, purchases, ratings, reviews, etc. The advantage of this method is that it does not require complex calculations or model training to make recommendations for new users who lack a history of interactions, thus mitigating the cold start problem. In research by Zhu et al. [45] exploring the influence of social factors in online recommendation systems, it was noted that users might change their minds and reverse their initial choices when they receive information about the popularity of items. Rajeev Kumar et al. [46] suggested incorporating social

popularity factors (such as user reviews and ratings) as implicit feedback into the SVD++ decomposition method to improve the accuracy and scalability of recommendations, analyzing the relationships between users and items and predicting items that users might like. Amin Javari et al. [47] proposed four collaborative filtering and Markov recommendation algorithms based on popularity and novelty scores for the next-basket recommendation, noting that the accuracy of popularity-based methods decreases as the size of the basket increases. While popularity-based recommendation methods are effective in certain scenarios (such as for new users or large-scale events), they have significant limitations. They tend to over-recommend items that are already very popular, resulting in similar recommendations for all users rather than differentiating based on unique user preferences. Additionally, they often overlook newly launched or less popular items, thus reducing the diversity of recommendations.

2.1.3 Hybrid Methods

Reviewing methods based on Markov chains and collaborative filtering, it's evident that each has limitations that make it difficult to fully capture users' preferences and purchasing behaviors. To address these issues, many researchers have begun to explore combining these techniques, aiming to provide more accurate and comprehensive recommendations by integrating users' long-term preferences with behavioral sequences. For example, Ruining He et al. [48] proposed a method called Fossil, which combines item-based collaborative filtering with Markov chains to consider both long-term and short-term dynamics for more accurate sequence recommendations. Steffen Rendle et al. [49] introduced a method called Factorizing Personalized Markov Chains (FPMC), which models the transitions between a user's general interests and their baskets. This method learns a unique transition matrix for each user and employs a special factorization technique to ensure that each transition effectively learns the user's item preferences and sequential behaviors, thus recommending the next item to purchase. Pengfei Wang et al. [50] proposed a novel recommendation method called the Hierarchical

Representation Model (HRM), which uses aggregation operations to combine items in a basket into different levels to establish a hierarchical representation. This layered representation helps the model understand the relationships between items at different levels. However, these methods primarily focus on local, short-term relationships, such as FPMC emphasizing personalization and HRM emphasizing hierarchical representation, and lack modeling of longer-term, global relationships. Therefore, many neural network-based recommendation models have emerged to address the limitations of traditional methods.

2.2 Neural Network Methods

Neural network-based recommendation models have undergone a series of evolutions, from early Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to later developments such as Attention mechanisms, Graph Neural Networks (GNNs), and the recent Transformer architectures. Each evolutionary stage has brought unique features and enhancements to better address the challenges in basket recommendation tasks. Initially, RNN-based methods, like the Dynamic Recurrent Basket Model (DREAM) proposed by Feng Yu et al. [51], used max/average pooling to encode baskets. This not only learned dynamic user representations but also captured global sequential features between baskets. Wang et al. [52] introduced a model called IntNet, which treats user behavior as heterogeneous sequence data driven by multiple intentions, focusing on understanding and predicting the complex behaviors behind users' basket choices. However, while RNNs are adept at capturing the temporal dependencies of sequence data, they often struggle with long-distance dependencies and may not easily identify local features. By incorporating CNNs, it's possible to effectively recognize local patterns and hierarchical structures within data and enhance computational efficiency through their parallelization capabilities. For example, Chengfeng Xu et al. [53] proposed a Recurrent Convolutional Neural Network (RCNN) model that uses RNNs to capture complex long-term dependencies while utilizing CNN convolutional operations to extract short-term

sequential patterns between recurrent hidden states. Huang et al. [54] introduced a multiscale model combining RNNs and CNNs, focusing on capturing multi-resolution online shopping behaviors. Their model uses a pyramidal modulation architecture to identify shopping behaviors across different time scales and a gating mechanism to adjust the importance of resolution views, while Graph Neural Networks reveal dependencies among purchases.

The methods based on RNNs and CNNs primarily focus on the sequential relationships of items, often overlooking specific similarities or interconnections between items, which makes it challenging to capture users' short-term and long-term preferences. Consequently, subsequent research efforts have been dedicated to addressing this issue through the Attention mechanism. Specifically, the Attention mechanism can easily remember long-range dependencies within RNNs and help CNNs focus on the important parts of the input. It dynamically determines which parts of the sequence the model should focus on, thus more effectively identifying and learning the relationships between items. For example, Shuai Zhang et al. [55] proposed a recommendation model called AttRec, which combines a self-attention module to model users' short-term intentions and a collaborative metric learning component to capture users' long-term preferences. This model not only focuses on consecutive items but also covers all user interactions within the current window, thereby considering both local and global user interactions. Ting Bai et al. [56] introduced an attribute-aware neural attention model, ANAM, which explicitly considers the temporal changes in users' product preferences and takes into account product attributes through a hierarchical structure to integrate user preferences and product attribute information for recommendations. Yongjun Chen et al. [57] proposed a model named AnDa, which includes a periodic-aware encoder designed to capture various temporal patterns of dynamic attributes and introduces an intra-basket attention module to learn the relationships between items.

However, despite improvements in capturing both local and global interactions between items using Attention-based methods, the complex structural relationships between users and

items in actual recommendation scenarios are still not fully explored. In fact, these interactions often form a vast interaction network containing multi-level hidden structural information. Consequently, Graph Neural Networks (GNNs), as a technology capable of deeply parsing such structural information, have gradually attracted scholars' attention. For example, Mengting Wan et al. [58] proposed a new framework based on Graph Convolutional Neural Networks (GCNN) called BasConv, which emphasizes high-order collectivity and heterogeneity. It gathers semantic information from nodes in high-order paths and propagates information on the UBI graph by differentiating between node types of users, baskets, and items, thereby more comprehensively capturing the interactions between users and items. Yitong Pang et al. [59] introduced a Heterogeneous Global Graph Neural Network (HG-GNN) that builds a heterogeneous global graph covering all sessions, coupled with two graph-enhanced preference encoders, to capture user preferences comprehensively across historical and current sessions. Additionally, the traditional assumption of a single intent in basket recommendation is challenged. For instance, when a basket contains coffee and milk, the system might only consider that the user intends to make coffee. However, Zhiwei Liu et al. [60] proposed the MITGNN method, which breaks this assumption by aiming to uncover multiple intents within a basket. MITGNN is a new framework that combines a translation-based model with a graph neural network, learning multiple distinct translations from a basket to construct intents, where each translation models the interrelationships among intents.

Following the surge in GNN research, the rise of the Transformer architecture has marked another revolutionary advancement in the domain of basket recommendation. With its unique self-attention mechanism, encoder-decoder architecture, and robust parallel processing capabilities, Transformers have significantly improved recommendation performance and training speed, gradually replacing earlier neural network methods based on RNNs and CNNs across various tasks. For example, Sun et al. [61] proposed a next-basket recommendation method based on Graph Attention Networks (GAT) and the Transformer model. This model

constructs an item-basket relationship graph, uses GAT to learn the interaction characteristics of baskets, and ultimately obtains a probability distribution of items. Its results surpass many existing neural network-based methods, such as the previously mentioned ANAM model [56]. Yang et al. [62] introduced a BERT-based IERT model that considers transaction contexts for context-aware item representation. IERT pre-trains item representations based on transaction backgrounds during the offline phase and fine-tunes these representations with a new output layer during online recommendations, thereby capturing user purchasing behavior based on contextualized embeddings. Bai et al. [63] proposed a dual-sequence network for time-set prediction, DSNTSP, which independently learns item-level and set-level representations based on the Transformer framework, and then captures multiple temporal dependencies between items and sets through a co-transformer module. Finally, these dependencies are integrated using a gated neural module to predict the next basket.

Overall, in past research, we have observed the evolution of neural network models in the recommendation domain, from RNNs and CNNs to attention mechanisms, GNNs, and Transformer architectures. However, despite these advanced neural network approaches achieving commendable performance in some recommendation tasks, they have not been integrated with traditional vector space models. In such models, each user's purchasing behavior is represented by a multidimensional vector, which objectively assesses the degree of repeated purchasing behavior of each consumer towards each product. Therefore, if previous neural network models incorporated this type of personalized repeated purchase behavior vector model, it might lead to more efficient recommendation results.

2.3 Personalized Item Frequency & the Difficulties Encountered in Learning Item Frequency

Repeated purchase behavior, which refers to a user buying or interacting with the same item or service multiple times within a certain timeframe, is particularly common in various

sectors such as e-commerce, music streaming, and video streaming. This behavior not only reflects user preferences and loyalty but also becomes an indispensable factor in recommendation systems. Indeed, repeated purchase behavior has gradually been incorporated into many sequence-based and session-based recommendation methods[31-33], as well as next-basket recommendation tasks[14, 15, 35, 36], and has been proven to significantly enhance recommendation performance. For example, Ren et al. [31] introduced the RepeatNet model, which incorporates a repeat-explore mechanism into the encoder-decoder structure, using two decoders to learn item recommendation probabilities under two different modes, and automatically learn the switching probabilities between modes to recommend the right items to the user at the right time. Wang et al. [33] proposed the SLRC model, which integrates the Hawkes Process with collaborative filtering elements to explicitly handle the temporal dynamics of short-term and lifetime effects related to items, achieving a balance between recommending new and previously consumed items. Hu et al. [35] presented the Sets2Sets model within an encoder-decoder framework. This model not only focuses on the information within the same basket and between different baskets but also considers the user's personal repeat consumption patterns and time-decaying weights, giving more weight to the most recent baskets. Ariannezhad et al. [14] introduced the ReCANet model, which considers the frequency and recency of user purchases, combines the similarities of items and users, and the user’s personal repeat consumption patterns, utilizing multiple LSTM layers to simulate the user's consumption patterns toward items.

In recent years, Personalized Item Frequency (PIF), based on repeat purchase behavior, has gradually gained attention in the academic community for its application in recommendation tasks. PIF represents the ratio of the number of times a product appears in a user's basket to the total number of baskets. For example, if George made five purchases at a supermarket and bought milk three times, the PIF value for milk for George would be 0.6. PIF is primarily used to assess the frequency of interaction between users and products; the higher

the PIF value, the greater the likelihood that the user will repurchase the item. This not only reflects the user's preference for specific products but also reveals their purchasing habits and behavior patterns. Due to the deep insights into user behavior that PIF provides, many scholars have begun to explore its application in recommendation systems to offer more personalized recommendations. Several studies have indicated that a high PIF value for an item suggests a high preference for that item by the user[34, 35, 64]. Among them, Hu and others proposed a KNN-based method called TIFU-KNN[34], which primarily features PIF. This method has outperformed many deep learning-based recommendation models such as FPMC[49], SHAN[65], and DREAM[51] in NBR tasks. Additionally, it was a pioneering study that explored the learning of PIF information with RNNs. Specifically, the study experimented with a general RNN model, excluding some variant models. Training results showed that even after trying different optimizers (such as SGD and RMSprop), increasing the size of the training data, or removing the embedding layer and directly using One-Hot vectors as input to the RNN units, the training error remained similar and did not improve. This study revealed an important finding: while deep learning-based RNN models theoretically have the capability to learn PIF, they tend to fall into local optima during the actual training process, making it difficult to learn an accurate model. This discovery provides profound insights into the application and limitations of RNNs in the recommendation system domain concerning PIF and has been referenced in subsequent research[66].

2.4 Summary

Summarizing the content from the previous sections, in the research on basket recommendation, traditional methods like those based on Markov chains and collaborative filtering have demonstrated certain effectiveness but have also revealed numerous drawbacks, such as ignoring long-term user preferences, overlooking the complex relationships between baskets or items, and encountering cold start problems with new users or new items. While

popularity-based recommendation methods can address some cold start problems, they often overemphasize extremely popular items, leading to a decrease in recommendation diversity. To counter these challenges, scholars are increasingly exploring hybrid approaches that merge users' long-term preferences with their behavioral sequences. Meanwhile, neural network-based models, including RNNs, CNNs, attention mechanisms, GNNs, and Transformer architectures, have become the focal point of research. These models are adept at capturing both the short-term and long-term preferences of users, the interconnections among items, and the intricate structural relationships between users and items. However, despite their advancements, these models still encounter specific challenges in practical applications, particularly in effectively modeling more extensive, global relationships. Additionally, they have not attempted to integrate with traditional vector space models, which are advantageous in reflecting the repeated purchase behaviors of different consumers.

On the other hand, basket recommendation tasks are increasingly influenced by repeated purchase behavior. Under the non-deep learning method TIFU-KNN, PIF information has been proven to have significant potential and practical value in NBR tasks. Simultaneously, the use of RNNs to utilize PIF information in deep learning has been shown to be ineffective for learning. Based on these findings, this study aims to combine traditional vector models with existing neural networks to explore whether the integration of these two models can continue the excellent performance of traditional KNN combined with PIF information. At the same time, it aims to delve into how PIF's recommendation efficacy and application potential can be enhanced under the deep learning-based Attention mechanism. Through this approach, we hope to more accurately capture users' repeated purchase behaviors and attempt to provide a more comprehensive and precise basket recommendation system by combining different models and strategies.

3. PROPOSED RESEARCH

This section introduces the Personalized Item Frequency and Temporal Attention for Next-basket Recommendation (PIFTA4Rec) model architecture proposed in this study. Initially, we will explain the problem definition and the meaning of the symbols used, followed by an overview of the overall architecture. Then, we will describe in detail each component of the PIFTA4Rec model, as well as the loss function used during the final training.

3.1 Problem Definition and Symbol Description

In practical applications, assume there is a transaction dataset that includes a set of user collections $U = \{u_1, u_2, \dots, u_{|U|}\}$, a set of item collections $I = \{i_1, i_2, \dots, i_{|I|}\}$, and a set of basket collections $B = \{B^1, B^2, \dots, B^u, \dots, B^{|U|}\}$ generated by all user-item interactions. Here, $u \in U$ represents a user, $i \in I$ represents an item, B^u represents all basket sequences of user u, and $|U|$ and $|I|$ respectively represent the number of users and items. Each user u's purchase history is represented by a time-ordered sequence of baskets $B^u = \{b_1^u, b_2^u, \dots, b_t^u\}$, where $b_t^u \subseteq I$ is the t-th basket purchased by user u. Each basket in B^u, $b_t^u = \{i_1^u, i_2^u, \dots, i_n^u\}$, represents a collection of items in one transaction, where n represents the number of items in the basket. Thus, the research task is defined as follows: given a series of historical basket sequences $B^{u\prime} = \{b_1^u, b_2^u, \dots, b_{t-1}^u\}$ of user u, predict the collection of items $Y = \{y_1, y_2, \dots, y_k\}$ that the user is most likely to purchase in the next basket b_t^u, where $Y \subseteq I$、$y_k \in I$、$k \in N$. Table 1 provides an explanation of the symbols used in this study

Table 1 Notation

Notation	Description
$u \in U, i \in I$	User u in the user set U and item i in the item set I.
$\|U\|$, $\|I\|$	The total number of users and items.
B	All historical basket collections.
B^u	The historical basket sequences of user u.
B_i^u	The embedding of the i-th basket of user u.
$b_t^u \subseteq I$	The t-th basket purchased by user u, which is a subset of the item set I.
$Y \subseteq I$	The predicted set of items.
$y_k \in I$	An item in the predicted set.
U_u	The embedding vector of user u.
U_{target}^{knn}	The target user vector in KIM.
U_{nbr}	The neighborhood set of U_{target}^{knn} in KIM.
$U_{n_k}^{knn}$	The embedding vector of the k-th neighbor in the KIM neighborhood set U_{nbr} .
$U_{nbr_{arg}}^{knn}$	The average vector of the K nearest neighbors of U_{target}^{knn} in KIM.
Ans_1 、Ans_2	The output prediction scores for KIM and DLIM, respectively.
U_{Target}	The target user vector in DLIM after Temporal Attention.
U_{n_k}	The vector of the k-th neighbor in DLIM after Temporal Attention.
$U_{Neighbor}$	The vector of the K neighbors of U_{target}^{knn} in DLIM after Self-Attention.
I_i	The embedding of item i.
i_n^u	The n-th item in the basket.
Δt_i	The time interval between the i-th basket and the last basket.

In this section, we will outline the PIFTA4Rec method framework proposed in this study. As shown in Figure 3, PIFTA4Rec primarily consists of two sub-models: the KNN-based Information Module (KIM) and the Deep Learning-based Information Module (DLIM), as well as a Final Prediction Module (FPM) for predicting the final results.

Firstly, the KNN-based Information Module (KIM) on the left is a network model based on user PIF information. It is divided into two parts: the data preprocessing stage and the prediction stage. Data preprocessing stage: As depicted in Figure 4, the purpose of this stage is to generate basket embedding vectors B_i^u by considering users' purchasing frequencies and adjusting the importance of each basket to the user using grouped time-decay weights. We refer to the encoding method proposed by previous research [34] to generate each user's embedding vectors $U_1, U_2, U_u, \dots, U_{|U|}$, and use the KNN method to find K nearest neighbors for each target user U_{target}^{knn}, denoted as $U_{nbr} = \{U_{n_1}^{knn}, U_{n_2}^{knn}, \dots, U_{n_k}^{knn}\}$, and finally obtain the average vector of these K neighbors $U_{nbr_{arg}}^{knn}$. Prediction stage: In the original method from prior research [34], the target user vector U_{target}^{knn} and the average vector of K neighbors $U_{nbr_{arg}}^{knn}$ were multiplied by a hyperparameter to produce the prediction result. In our research method, these vectors serve as inputs, and a deep learning architecture is used to generate the first prediction score Ans_1.

The Deep Learning-based Information Module (DLIM) on the right is a network model based on users' original purchase sequences. It is divided into three parts: data preprocessing, preference learning, and prediction stages. Data preprocessing stage: This stage uses Item2Vec to embed the items in all users' historical basket sequences, producing embedding vectors for all items $I_1, I_2, I_i, \dots, I_{|I|}$. Then, for the target user and the K neighbors found via KNN in the KIM, the average of the item embedding vectors is taken to obtain the basket embedding vectors for $k+1$ users, which are used as input for the preference learning stage. Preference learning stage: The objective is to learn the sequence relationships and purchasing preferences of $k+1$

users from their historical basket sequences. We use a temporal attention mechanism to generate the embedding vector U_{Target} for the target user, as well as the embedding vectors for each neighbor $U_{NBR} = \{U_{n_1}, U_{n_2}, \ldots, U_{n_k}\}$. Then, a neighbor vector $U_{Neighbor}$ is generated for all neighbors using Self-Attention in the Transformer. Prediction stage: The U_{Target} and $U_{Neighbor}$ are element-wise summed and then passed through an MLP layer with a ReLU activation to generate the second prediction score Ans_2.

Finally, the Final Prediction Module (FPM) considers the prediction scores Ans_1 and Ans_2 produced by the two sub-models, KIM and DLIM, respectively. It uses the Sigmoid function to obtain the predicted probability of each item appearing in the next basket. The items are then ranked, and the top K items are selected as the final recommended item set $Y = \{y_1, y_2, \ldots, y_K\}$.

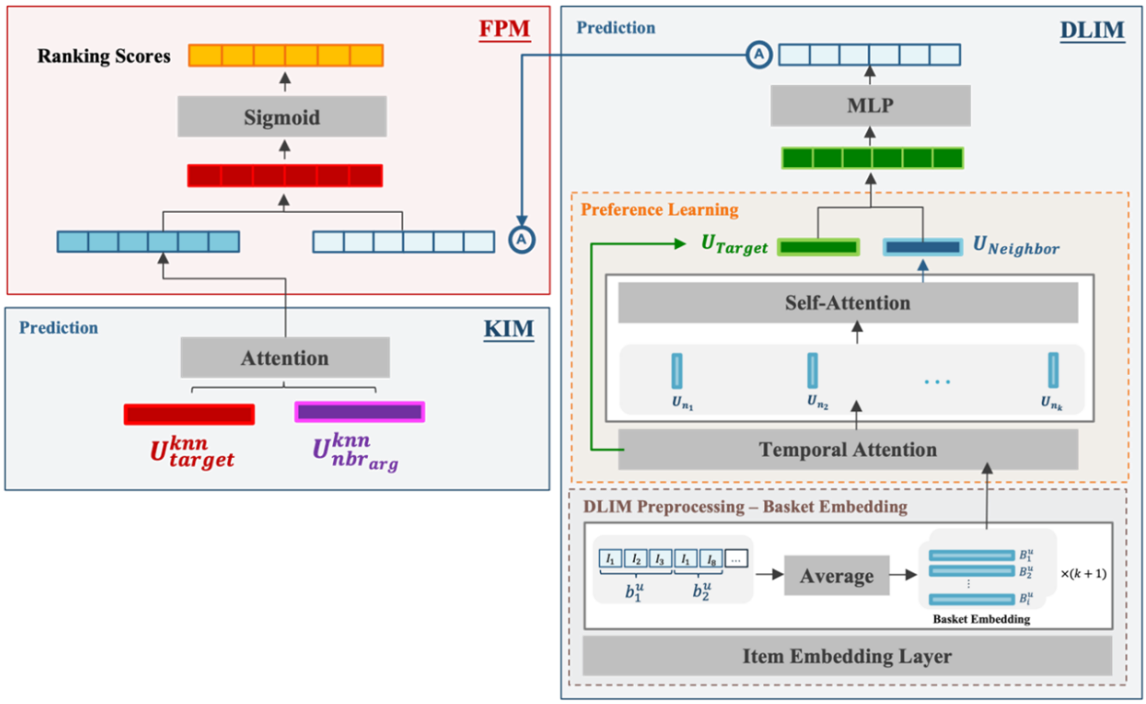

Figure 3 Architecture Diagram of the PIFTA4Rec Model

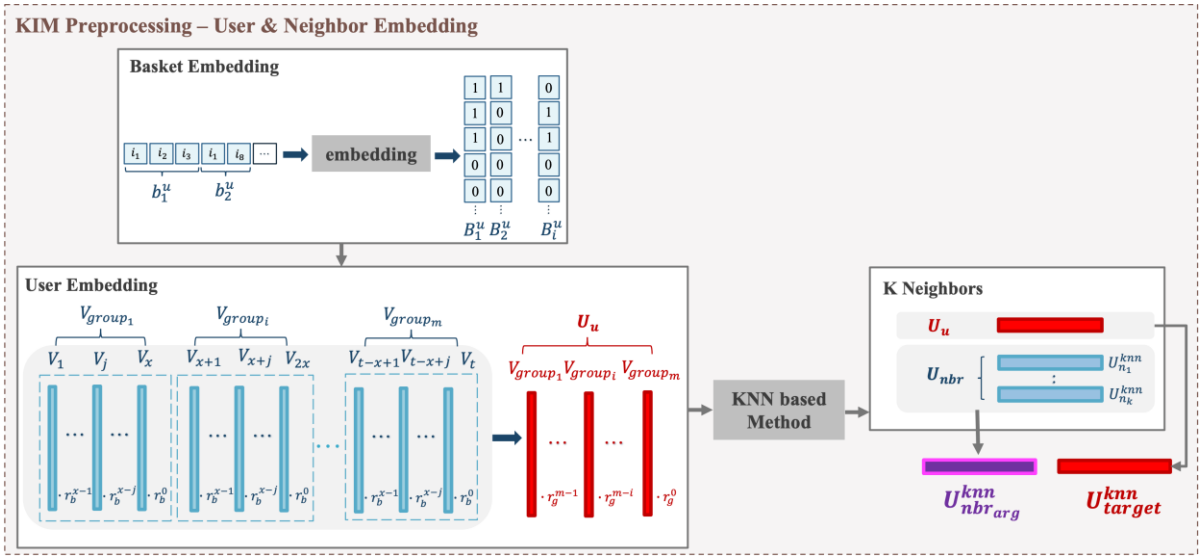

Figure 4 KIM Data Pre-processing Stage

3.2 Preprocessing

3.2.1 KIM – User Embedding Vector & Neighbor Embedding Vector

The preprocessing flow of the KNN-based Information Module (KIM) is illustrated in Figure 4. This stage utilizes the encoding method proposed in previous research [34] to obtain the embedding vector of the target user U_{target}^{knn} and the average embedding vector of the K nearest neighbors $U_{nbr_{arg}}^{knn}$ of each target user. The method is as follows:

Firstly, in the Basket Embedding process, given the historical basket sequences of all users, we intuitively use the concept of "which items are present in the basket" to generate an embedding vector for each basket B_i^u. Specifically, we use the size $|I|$ as the dimension of the basket embedding vectors. For each item $i \in I$, its position in the embedding vector is determined by its ID. If an item is present in the basket, the corresponding value at its position will be 1; if not, it will be 0. This representation is similar to One-Hot encoding, where each item only has a value at its corresponding ID position. For example, if a basket contains only items with IDs 1 and 3, the embedding vector for that basket would be [1, 0, 1, ..., 0].

After obtaining each basket embedding vector B_i^u, in the User Embedding section, for

user u, we use all their basket embedding vectors B_1^u , B_2^u,... B_i^u, along with a clustering method to calculate time weights, to measure the importance of each purchase and generate the embedding vector U_u for user u. Here, the process is divided into the following two steps:

1. We divide the user's t basket embedding vectors into m groups, denoting the group size as $x = \frac{t}{m}$. Each vector in a group (ordered by time) is multiplied by a time decay weight r_b^{x-j}, where r_b is the time decay rate within the group. We then calculate the average vector of the weighted vectors within each group, representing it as the group vector v_{group_i}. If the vectors cannot be evenly divided, the first group's size is $t - x \cdot (m-1)$, while the size for the remaining groups, x, is calculated as $\left\lceil \frac{t}{m} \right\rceil$.
2. After obtaining several group vectors v_{group_i}, we multiply the i-th group vector v_{group_i} by a time decay weight r_g^{m-i}, where r_g is the time decay rate for the groups. Finally, we compute the average of these weighted group vectors to produce the final embedding vector U_u for user u.

After obtaining the embedding vector U_u for each user, we continue to follow the methodology of previous research [34], employing the traditional KNN method to search for the closest other users to each target user U_{target}^{knn}. To determine the similarity between users, we use the Euclidean distance as the distance metric. The Euclidean distance is the straight-line distance between two points in space, where a smaller (larger) distance implies a higher (lower) similarity. This method is intuitive, simple, and very effective in multidimensional spaces. Suppose there are two points P and Q, with coordinates in n-dimensional space as $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$, respectively, then the Euclidean distance between these two points is as shown in formula (1):

$$\sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \cdots + (y_n - x_n)^2} \quad (1)$$

Therefore, for each target user U_{target}^{knn}, we calculate the Euclidean distance between their embedding vector and the embedding vectors of all other users. We then select the K nearest users as the "neighbors" of that target user to form the neighbor set $U_{nbr} = \{ U_{n_1}^{knn}, U_{n_2}^{knn}, \dots, U_{n_k}^{knn} \}$. Finally, as shown in formula (2), this study uses the "average method" to take the average of the sum of the embedding vectors of these K neighbors, which serves as the average vector of the K closest neighbors $U_{nbr_{arg}}^{knn}$ for the target user U_{target}^{knn}. Here, $U_{n_i}^{knn}$ represents the embedding vector of the i-th neighbor.

$$U_{nbr_{arg}}^{knn} = \frac{1}{K} \sum_{i=1}^{K} U_{n_i}^{knn} \tag{2}$$

3.2.2 DLIM –Basket Embedding Vector

The preprocessing flow of the Deep Learning-based Information Module (DLIM) is shown in Figure 5. In the domain of recommendation systems, many collaborative filtering methods measure similarity by analyzing relationships between items, whereas the Word2vec [67] method is widely adopted. However, Item2vec [68], proposed by Oren Barkan et al., is a Word2Vec-based algorithm, and this study utilizes Item2vec to encode each item $i \in I$, generating embedding vectors I_i for the items. Item2vec is trained using the Skip-gram with Negative Sampling (SGNS) algorithm, thereby generating embeddings for items in the latent space. Skip-gram in word embedding works by using a central word to predict surrounding context words, resulting in a word vector matrix after training. When applied to item embedding, it predicts other items in the same basket from a target item, capturing the associations between the target item and other items, and obtaining the learned item embedding matrix. Negative Sampling is a technique that optimizes the Skip-gram training process. Traditionally, for each training instance, the Skip-gram model would update the weights of all words in the vocabulary, making the training process very time-consuming. To improve this, Negative Sampling selects

a few items not in the same basket as negative samples for each positive sample and updates the weights of these negative and positive items only, thereby speeding up the model's training process.

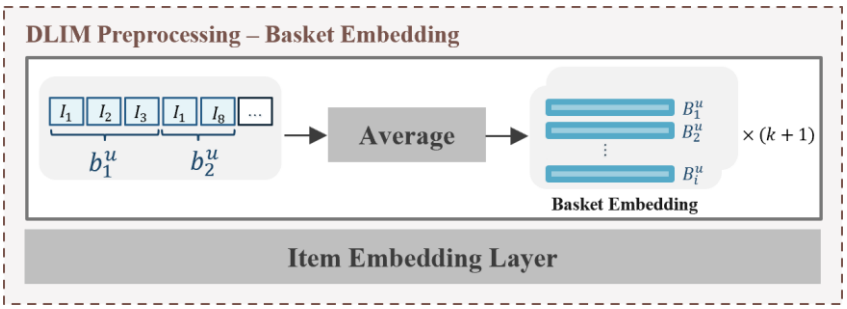

Figure 5 DLIM Data Pre-processing Stage

Specifically, as shown in Figure 6, during Skip-gram, each item in the basket is first converted into One-Hot Encodings and then multiplied by a randomly initialized weight matrix $W_{|\mathrm{I}|\times N}$ to obtain the N-dimensional vector of the target item. Then, another weight matrix $W'_{N\times|I|}$ is initialized randomly. For positive samples, the dot product is performed between the N-dimensional vector of the target item and the vectors of all other items in the same basket in $W'_{N\times|I|}$. Similarly, for negative samples, the dot product is performed between the target item's N-dimensional vector and the vectors of a few other items not in the same basket in $W'_{N\times|I|}$. These are then passed through a Sigmoid function to output a prediction result between [0, 1]. Afterward, the objective function is adjusted through Stochastic Gradient Descent (SGD) to modify the weight matrices $W_{|\mathrm{I}|\times N}$ and $W'_{N\times|I|}$, ultimately allowing the trained matrices $W_{|\mathrm{I}|\times N}$ or $W'_{N\times|I|}$ to serve as item embeddings for subsequent applications.

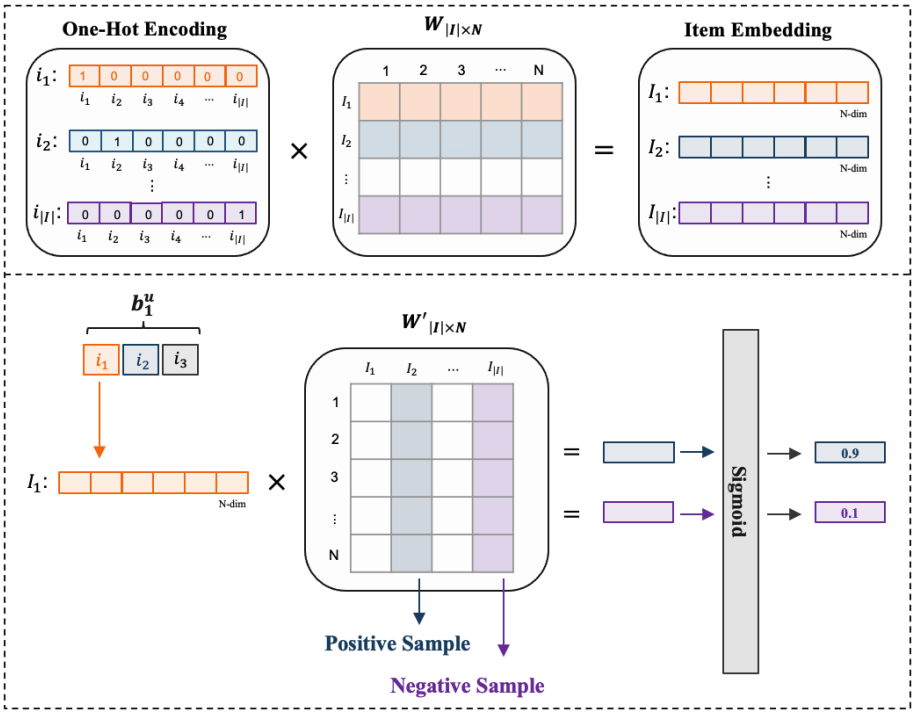

Figure 6 Skip-gram in Item2Vec

Therefore, in a set of K items $i_1, i_2, \ldots, i_K$, our objective function as shown in formula (3) aims to maximize the probability of co-occurrence of other items in the same basket with the target item, while minimizing the probability of co-occurrence of items not in the same basket with the target item.

$$\frac{1}{K}\sum_{m=1}^{K}\sum_{n\neq m}^{K} log\, p(i_n|i_m) \tag{3}$$

Here, i_m is the m-th target item, and i_n is the n-th other item in the same basket. In $p(i_n|i_m)$, negative sampling is performed, where some items not in the same basket are randomly selected as negative samples for training, and only a portion of the weights are updated to reduce computation, as calculated in formula (4).

$$p(i_n|i_m) = \sigma(u_m^{\top} v_n)\prod_{k=1}^{N}\sigma(-u_m^{\top} v_k) \tag{4}$$

In this process, each positive sample undergoes N negative samplings. Here, $\sigma(u_m^{\top} v_n)$

represents the prediction result for the positive sample, where u_m is the final representation of the m-th target item, and v_n is the latent vector of the n-th other item in the same basket. $\sigma(-u_m^\top v_k)$ represents the prediction result for the negative sample, where v_k is the latent vector of the k-th other item in a different basket. $\sigma(\cdot)$ is the Sigmoid nonlinear activation function, which maps the prediction results between [0, 1]. If the value is closer to 1, it indicates that the item is in the same basket as the target item; conversely, if it is closer to 0, it indicates they are in different baskets.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{5}$$

Additionally, as each item appears with different frequencies, to address the imbalance between rare and frequent items during negative sampling, a probability of discarding samples can be used as per formula (6), where $f(i)$ is the frequency of item i, and ρ is the prescribed threshold. The higher the frequency of an item, the greater the probability of it being discarded, thus increasing the chance of sampling less frequently appearing items:

$$p(\, discard \mid i\,) = 1 - \sqrt{\frac{\rho}{f(i)}} \tag{6}$$

After obtaining the item embedding vectors $I_1, I_2, I_i, \ldots, I_{|I|}$, for each target user, we continue to use the K nearest neighbors found through the KNN method in the first sub-model, KIM. We encode all historical basket sequences of these $K+1$ users to obtain all basket embedding vectors for user u, $B_1^u, B_2^u, \ldots B_i^u$. Specifically, for user u's t-th basket $b_t^u = \{i_1^u, i_2^u, \ldots, i_n^u\}$, its basket embedding vector B_i^u is as shown in formula (7). We represent it by taking the average of the sum of the embedding vectors of all items in the basket. Here, n represents the number of items in the basket, and $I_{i_j^u}$ is the embedding vector of the j-th item in the basket.

$$B_i^u = \frac{1}{n}\sum_{j=1}^{n} I_{i_j^u} \tag{7}$$

3.3 Model

3.3.1 KNN-based Information Module (KIM)

In the data preprocessing phase, we utilized the encoding method from previous research [34] to obtain the target user vector U_{target}^{knn} and the average vector of the K neighbors $U_{nbr_{arg}}^{knn}$. Originally, the researchers in that study simply adjusted the importance ratio between the target user and similar neighbors using a hyperparameter, generating prediction results through simple multiplication operations. This method did not set parameter values through learning but adjusted them based on the results of multiple experiments to find an "optimal parameter estimate". Therefore, in the prediction stage of our study's KIM, we replace this predictive method with the attention mechanism from deep learning. The aim is to utilize its learning capability to find a more precise parameter configuration, thereby making the recommendations more accurate. Specifically, for the target user vector U_{target}^{knn} and the average vector of the K neighbors $U_{nbr_{arg}}^{knn}$, we set a learnable parameter α to control the weight between the target user vector and the neighbor average vector and obtain the prediction score Ans_1 using formula (8).

$$Ans_1 = \alpha \times U_{target}^{knn} + (1 - \alpha) \times U_{nbr_{arg}}^{knn} \tag{8}$$

3.3.2 Deep Learning-based Information Module (DLIM)

After obtaining the embedding vectors B_i^u for all the baskets of the target user and each neighbor, we proceed to a Temporal Attention layer. Here, through a learnable weighted averaging method, we can discern and weigh the relative importance of each data point when aggregating multiple data points. This allows us to aggregate basket embedding vectors obtained at different time points into a temporally informed feature representation. This step is particularly crucial when dealing with time-series data, as data from different time points contribute differently to the final prediction. We will use an attention mechanism to learn and

assign an optimal weight to each data point, addressing this issue by reflecting the relative importance of each data point. Specifically, for a series of historical basket sequences of user u, $B^{u\prime} = \{b_1^u, b_2^u, \dots, b_{t-1}^u\}$, we first calculate the time intervals between each basket and the last basket b_{t-1}^u to generate a sequence of time intervals $T_{u_k} = \{\Delta t_1, \Delta t_2, \dots \Delta t_{t-1}\}$, as shown in formula (9).

$$\Delta t_i = t_{t-1} - t_i \tag{9}$$

Next, an exponential decay formula is applied to compute the unnormalized weights of each basket, assuming that recent events have a greater impact on the current prediction than past events, as shown in formula (10). Here, λ is the decay rate, a learnable parameter that determines the extent to which time intervals affect the weights. As λ increases, the model's sensitivity to time also increases, quickly diminishing the influence of past baskets on the current prediction. However, to normalize these weights, we divide the weight of each basket by the sum of all basket weights to ensure that the sum of all basket weights equals one. This step is crucial as it weights the contribution of each basket according to its relative temporal importance. Finally, as shown in formula (11), we multiply each basket's embedding vector B_i^u by its corresponding normalized weight W_i, and sum these products to obtain the embedding vector U_{Target} representing the target user, and the embedding vectors for each neighbor, U_{n_1}, $U_{n_2}, \dots, U_{n_k}$.

$$W_i = \frac{e^{-\lambda \Delta t_i}}{\sum_{j=1}^{t-1} e^{-\lambda \Delta t_j}} \tag{10}$$

$$U = \sum_{i=1}^{t-1} B_i^u \cdot W_i \tag{11}$$

After obtaining the set of embedding vectors for the target user's K neighbors, $U_{NBR} = \{U_{n_1}, U_{n_2}, \dots, U_{n_k}\}$, we input these vectors into the Multi-Head Attention component of the Transformer to learn the complex relationships between the neighbors and obtain a composite

embedding vector $U_{Neighbor}$ representing all the target user's neighbors. Specifically, in a single self-attention layer, we first calculate self-attention for each neighbor. To do this, we multiply all neighbors' embedding vectors $U_{NBR} = \{U_{n_1}, U_{n_2}, \dots, U_{n_k}\}$ with three learnable weight matrices W^Q、W^{K}, and W^V, thus obtaining three matrices Q (Query), K (Key), and V (Value). Finally, we perform the Scaled Dot-Product Attention calculation, as shown in formula (12). Here, d_k is the dimension of Q and K, where Q represents the neighbor currently used to find relationships with other neighbors, K represents all neighbors that need to be searched (including itself), and V represents the neighbor information weighted. We perform a dot product between Q and K to compute the attention scores of the current neighbor for all neighbors, then divide by $\sqrt{d_k}$ to prevent excessively large scores. Afterward, we use the Softmax function to obtain attention weights and multiply these weights with V to obtain the weighted output vector.

$$self - attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

However, the Multi-Head Attention in the Transformer uses h parallel self-attention layers, performing h instances of scaled dot-product attention simultaneously. This mechanism allows the model to have more parameters to learn complex relationships. Initially, the input matrices Q, K, and V are each split into h matrices W_i^Q、W_i^K、W_i^V, serving as the input matrices for the h heads, where $i = 1, \dots, h$. Then, each of these h heads undergoes its self-attention to output $head_1, \dots, head_h$. After performing scaled dot-product attention, these h output vectors are concatenated and then linearly transformed through a weight matrix W^o to obtain the output Z. The formula for Multi-Head Attention is as follows, where W is the weight matrix, d_{model} is the dimension of each element or word embedding vector in the model input sequence, and d_V is the dimension of the value vectors, d_k is the dimension of the key and query vectors, with both d_V and d_k set to d_{model}/h, where h is the number of heads.

$$\text{head}_\text{i} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right);\ i = 1, \dots, h \tag{13}$$

$$\text{MultiHead}(Q, K, V) = \ \text{Concat}(\text{head}_1, \dots, \text{head}_\text{h})W^O \tag{14}$$

$$Z\ = \text{MultiHead}(Q, K, V) \tag{15}$$

$W_i^Q \in \mathbb{R}^{d_{model} \times d_k},\ W_i^K \in \mathbb{R}^{d_{model} \times d_k},\ W_i^V \in \mathbb{R}^{d_{model} \times d_v},\ W^O \in \mathbb{R}^{hd_v \times d_{model}}$

After passing through Multi-Head Attention, the output vector Z is combined with the original input neighbor embedding vectors U_{NBR} in the Add & Norm layer, calculated using formula (16). Initially, the original input is added to the vector from the multi-head attention output to perform residual connection, aimed at mitigating the problem of vanishing gradients. Then, layer normalization is applied to enhance performance.

$$\text{Z}' = \text{LayerNorm}(\ \text{U}_\text{NBR} + \text{Z}\ ) \tag{16}$$

The resulting Z' continues to the Feed Forward Network Layer, where it is calculated using the formula (17). In this process, the vector is projected into a higher-dimensional space and passed through Dropout and ReLU to extract important information and project back to the original space. Then, it enters the Add & Norm layer again, producing a normalized vector matrix Z_l. W_1, b_1, W_2, and b_2 are learnable parameters, and LayerNorm is a standard normalization layer. Finally, the last neighbor embedding vector from the vector matrix Z_l, which now contains information from other neighbors, is taken as the composite embedding vector $U_{Neighbor}$ for all the target user's neighbors.

$$\begin{aligned} Z_l\ &= \text{LayerNorm}(\ Z' + FFN(Z')) \\ &= \text{LayerNorm}(\ Z' + \text{Dropout}(\text{ReLU}(\ Z'W_1 + b_1\ )W_2 + b_2\ )) \end{aligned} \tag{17}$$

Once the target user vector U_{Target} and the composite vector $U_{Neighbor}$ of all its neighbors are obtained, as shown in formula (18), we add corresponding elements of both vectors to obtain the composite user preference U_{All} for subsequent recommendation

predictions. Then, as shown in formula (19), U_{All} is fed into a MLP layer and passed through the ReLU function, outputting the second prediction score vector Ans_2. This vector then serves as another input for the FPM, combined with the recommendation score vector Ans_1 obtained from KIM, to produce the final recommendation.

$$U_{All} = U_{Target} + U_{Neighbor} \tag{18}$$

$$Ans_2 = softmax\big(MLP(U_{All})\big) \tag{19}$$

3.3.3 Final Prediction Module (FPM)

After obtaining the first prediction score vector Ans_1 from KLM and the second prediction score vector Ans_2 from DLIM, we normalize them and sum them to get Ans_{all}, which is then passed through a Sigmoid function to obtain the final ranking scores F_{all}. For the loss function, we use the BCEWithLogitsLoss, which measures the discrepancy between the model's predicted probability distribution and the actual label probability distribution to determine the optimal parameters of the model. BCEWithLogitsLoss combines the computation of the Sigmoid function and cross-entropy loss, which helps to improve numerical stability and makes the design more stable and efficient in practical applications.

The formula is as follows: For each user's basket sequence $B^{u\prime} = \{b_1^u, b_2^u, \dots, b_j^u, \dots, b_{t-1}^u\}$ after removing the last basket, we predict the probability $\hat{y}$ that each item i will appear in the user u's next basket b_t^u. The goal is to minimize this function, i.e., to make the model's predicted item probability $\hat{y}$ as close as possible to the actual label y_i. Where N is the total number of items, y_i is the true label indicating whether the i-th item is in the next basket, with values of 0 or 1. If y_i is 0, it indicates that the i-th item is not in the next basket; if y_i is 1, it indicates that it is in the next basket. $\hat{y}_i$ is the model's logits output for the i-th item. σ represents the Sigmoid activation function.

$$\mathcal{L}(B^u) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot log(\sigma(\hat{y}_i)) + (1 - y_i) \cdot log(1 - \sigma(\hat{y}_i))] \tag{20}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{21}$$

$$\Theta * = \arg\min_{\Theta} \sum_{B^u \in B_{train}} \mathcal{L}(B^u) \tag{22}$$

4. EXPERIMENTAL DESIGN

4.1 Datasets

To validate the effectiveness of our model in real-world scenarios, this study utilizes two real-world datasets for experimentation:

- **TaFeng**: A public dataset available on Kaggle, which contains transaction data from a grocery store in China from November 2000 to February 2001, featuring 32,266 users, 23,812 products, and 817,741 transactions.
- **Dunnhumby**: Released by the commercial data analytics company Dunnhumby, this dataset includes all shopping done by 2,500 families over two years at a retailer, comprising 92,399 different products. For this study, we use the transactions from the first three months for experimentation.

For these two datasets, we consider all items purchased by a user on the same day as a single basket and perform the following preprocessing (in sequence):

- **Process 1**: Remove the least frequent items, retaining items that account for over 95% of all transactions.
- **Process 2 (only for Tafeng)**: Remove baskets that contain fewer than 3 items.
- **Process 3**: Remove data from users who have fewer than n baskets. For the TaFeng dataset, n is set to 3, and for the Dunnhumby dataset, it is set to 7.

Table 2 shows the data statistics after preprocessing. We use each user's last basket as the prediction target, with the remaining data serving as the user's historical sequence. The datasets are divided into training, validation, and test sets in a 7:1:2 ratio.

Table 2 Statistical Data after Data Set Preprocessing

Data	#items	#users	#transaction date	#basket quantity	average #baskets /user	average basket size
TaFeng	12,085	13,972	120	93,901	6.72	6.26
Dunnhumby	3,003	12,826	84	250,580	19.54	7.13

4.2 Baselines

To evaluate the effectiveness of our proposed model, we draw upon comprehensive insights developed by previous research[69] on the next-basket recommendation task and compare it against the following seven baseline methods. These include frequency-based PersonTopFreq, neighbor-based TIFU-KNN, and neural network-based DREAM, CLEA, Beacon, SHAN, and Sets2Sets methods. For these baseline methods, we use the hyperparameter settings provided in their respective papers or related GitHub repositories.

1. **PersonTopFreq**[69]: Utilizes Personalized Item Frequency (PIF) information directly, considering the most frequent k items from a user's history as the next basket, only focusing on repetitive items in predictions.
2. **TIFU-KNN**[34]: Models the temporal dynamics of item frequency information from past baskets using a neighbor-based approach to leverage PIF information for recommendations, outperforming many state-of-the-art NBR methods (including deep learning methods using RNN).
3. **DREAM**[51]: An embedding and RNN-based model that accounts for personal dynamic interests over different times and the global interactions of all baskets as time progresses.
4. **CLEA**[13]: An RNN-based contrastive learning model that automatically extracts items related to target items and generates representations using a GRU-based encoder.
5. **Beacon**[70]: An RNN-based method that considers merged information on pairwise correlations between items to encode baskets.

6. **SHAN**[65]: A hierarchical attention network model that divides historical baskets into long and short-term sections, carefully learning long and short-term preferences based on respective items.
7. **Sets2Sets**[35]: An Encoder-Decoder framework utilize pooling operations to obtain basket embeddings, adopt a set-based attention mechanism to learn representations from past user interactions, and consider repeat purchase patterns to enhance performance.

4.3 Evaluation Metrics

To compare the performance gaps with the baselines, this study employs the F1-score and NDCG as evaluation metrics. F1-score: The harmonic mean of precision and recall, which considers both the accuracy and comprehensiveness of the recommendation system, with a value range from 0 to 1. Precision measures the proportion of positive instances among the predicted positives, while recall measures the proportion of positives correctly predicted as positive among all actual positives. Calculations for the Top-K results are as follows:

$$F1 - score@K = 2\,\frac{precision@K \times recall@K}{precision@K + recall@K} \tag{23}$$

$$precision = \frac{TP}{TP + FP} \tag{24}$$

$$recall = \frac{TP}{TP + FN} \tag{25}$$

NDCG: Normalized Discounted Cumulative Gain. It accounts for the position and relevance of items, where each item's score is discounted based on its position in the ranking, and items ranked higher contribute more to the DCG than those ranked lower. DCG@K denotes the cumulative gain, discounted and summed for the top K items. Here, $r(i)$ represents the score of the i-th recommended item, valued at 1 if it is a hit, otherwise 0. However, comparing DCG across different users is not meaningful (different users have varying interests, and the same item may have different rankings), so to standardize DCG across users for comparison, it is

normalized by dividing by IDCG (the ideal DCG score, which is the DCG calculated from items ranked according to their relevance).

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{26}$$

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

4.4 Experimental Setting and Platform

For the hyperparameter settings of this model, the KIM part will reference the best combination from previous research[34] as shown in Table 3. Here, k denotes the number of neighbors, m denotes how many groups a user's baskets are divided into, r_b is the time decay rate within each group, and r_g is the decay rate across groups. For the neural network parts, this study utilizes the Adam optimizer[71] to optimize the model and conducts sensitivity analysis on different hyperparameters to study their impact on the model and select the best values for the experiments. During evaluation, the Top-K values of K are set to [5, 10, 30, 50, 65].

Table 3 KIM Optimal Hyperparameter Settings

#k	#m	#r_b	#r_g
300	7	0.9	0.7

After numerous experiments, the optimal hyperparameter settings for the two datasets are shown in Table 4. We will use this hyperparameter combination for subsequent baseline comparisons and ablation experiments. In the TaFeng dataset, we observed that setting Epochs to 64 and Learning Rate to 0.0001 yielded the best performance. For the Dunnhumby dataset, the optimal performance was obtained with Epochs set at 32 and Learning Rate at 0.00001. Additionally, the rest of the hyperparameters have the same settings across both datasets.

Initially, in the preprocessing of Item Embedding training, we found the optimal value within the range [16, 32, 64, 128], ultimately setting the Item Embedding Size and the dimensions d_{model} of the parameter matrices W^Q、W^K、W^V in the Transformer to 64. The dimension of the MLP hidden layers was set to 256 for best performance within the range [64, 128, 256, 512]. The time decay rate λ in Temporal Attention and the Dropout rate in the Transformer were tested within the range [0.1, 0.3, 0.5], both ultimately set to 0.3. Finally, the number of Heads and the number of layers in the Transformer were optimized within the range [1, 2, 4, 8], finally setting them to 4 and 1 respectively.

Table 4 Optimal Hyperparameter Settings for PIFTA4Rec Model

	Batch Size	Learning Rate	Embedding size	Hidden Dimension	Decay Rate	Dropout Rate	Multi-Head	Transformer Layer
TaFeng	64	0.0001	64	256	0.3	0.3	4	1
Dunnhumby	32	0.00001	64	256	0.3	0.3	4	1

Lastly, to ensure a fair comparison, all experiments in this study were implemented in Python using the Pytorch deep learning framework. The computer specifications used are as follows:

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

5. EXPERIMENTAL RESULTS

5.1 Experiment 1: Comparison with Baselines

To validate the predictive performance of the proposed method, we conducted comparisons with seven common baseline methods across two real-world datasets. Tables 5 and 6 respectively show the evaluation results of our model, PIFTA4Rec, and other baselines on the TaFeng and Dunnhumby datasets for the next-basket recommendation task. Each method utilized the F1-score and NDCG metrics, with K values set at 5, 10, 30, 50, and 65. These comparisons yielded the following observations:

- **Observation 1**: Among the seven baselines, the frequency-based P-TopFreq and KNN-based TIFUKNN are two traditional non-deep learning methods. Nevertheless, they demonstrate high competitiveness in both datasets. They outperformed most of the deep learning methods, with TIFUKNN being the best-performing baseline in the TaFeng dataset and P-TopFreq in the Dunnhumby dataset. This illustrates that traditional methods can achieve excellent performance on suitable datasets even without learning-based adjustments.
- **Observation 2**: Among the five deep learning methods, we observe varying performances across different datasets, with no single method demonstrating clear superiority on both datasets. In the TaFeng dataset, the RNN-based Beacon model performs best in terms of the F1-score, while the Encoder-Decoder-based Sets2Sets model excels in the NDCG metric. On the other hand, in the Dunnhumby dataset, CLEA achieves the best performance across all K values, followed by Sets2Sets. Overall, considering the larger basket sizes in Dunnhumby, this may indicate that CLEA is more suited for datasets with more baskets, and Sets2Sets shows robust performance in both datasets, while DREAM and SHAN underperform in both datasets, lacking generalizability.

- **Observation 3**: The model proposed in this study, PIFTA4Rec, showed significant superiority in both datasets compared to all baseline models, achieving the best performance. In the TaFeng dataset, PIFTA4Rec outperformed all other models across all evaluation metrics, with the similarly PIF-based TIFUKNN achieving second-best performance. This not only confirmed the significant impact of PIF information on predicting the next basket but also highlighted that combining attention mechanisms to learn from this information can effectively utilize it, thereby achieving superior recommendation performance. In the Dunnhumby dataset, both the frequency-based P-TopFreq and PIFTA4Rec showed competitiveness. Specifically, PIFTA4Rec performed better than P-TopFreq at most K values, but when the number of recommendations increased (K at 30, 50, 65), P-TopFreq showed better performance on the NDCG metric. This may be due to the very small number of items in the Dunnhumby dataset, which could prevent PIFTA4Rec from fully utilizing its capacity to learn complex relationships between users and items, possibly even negatively impacting its performance due to overfitting on a small dataset. By comparison, the P-TopFreq method, which considers the most common k items from a user's historical records as the next basket, may find it relatively easier to predict the order of recommended items in such an environment.

Table 5 Performance Comparison of PIFTA4Rec with Other Baselines on F1 Score

	TaFeng					Dunnhumby				
	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65
TIFUKNN	0.0688	0.0650	0.0497	0.0390	0.0335	0.1759	0.1829	0.1503	0.1241	0.1087
P-TopFreq	0.0595	0.0576	0.0429	0.0323	0.0270	0.1924	0.1989	0.1550	0.1230	0.1060
DREAM	0.0506	0.0410	0.0287	0.0252	0.0233	0.0842	0.0789	0.0581	0.0474	0.0420
CLEA	0.0501	0.0422	0.0289	0.0235	0.0210	0.1313	0.1168	0.0849	0.0701	0.0619
Beacon	0.0553	0.0480	0.0316	0.0259	0.0233	0.0819	0.0770	0.0588	0.0490	0.0443
SHAN	0.0509	0.0387	0.0273	0.0230	0.0216	0.0822	0.0784	0.0581	0.0492	0.0442
Sets2Sets	0.0536	0.0508	0.0391	0.0310	0.0274	0.0806	0.0878	0.0752	0.0613	0.0554
PIFTA4Rec	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	**0.2191**	**0.2246**	**0.1703**	**0.1343**	**0.1156**

Table 6 Performance Comparison of PIFTA4Rec with Other Baselines on NDCG

	TaFeng					Dunnhumby				
	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
TIFUKNN	0.0905	0.1098	0.1434	0.1553	0.1606	0.2184	0.2646	0.3341	0.3634	0.3758
P-TopFreq	0.0797	0.0947	0.1160	0.1215	0.1235	0.2385	0.2885	**0.3521**	**0.3766**	**0.3863**
DREAM	0.0715	0.0742	0.0779	0.0800	0.0810	0.0841	0.0893	0.0949	0.0971	0.0982
CLEA	0.0813	0.0922	0.1089	0.1175	0.1223	0.1766	0.1957	0.2276	0.2452	0.2537
Beacon	0.1065	0.1117	0.1248	0.1335	0.1383	0.1516	0.1349	0.1427	0.1548	0.1621
SHAN	0.0839	0.0892	0.1028	0.1112	0.1170	0.1035	0.1198	0.1430	0.1569	0.1641
Sets2Sets	0.0740	0.0885	0.1140	0.1232	0.1279	0.0958	0.1231	0.1648	0.1805	0.1892
PIFTA4Rec	**0.1243**	**0.1362**	**0.1613**	**0.1677**	**0.1784**	**0.3216**	**0.3105**	0.3419	0.3682	0.3800

5.2 Experiment 2: Sensitivity Analysis

This study conducted a sensitivity analysis to observe the impact of various parameters on the model and to ensure that the set parameters could achieve optimal performance. Firstly, the initial parameter is Batch Size. The results in Table 7 show that the best performance occurs when the Batch Size is set to 64 and 32 for the Tafeng and Dunnhumby datasets, respectively. When the setting is higher than the optimal value, performance decreases, which may be due to larger batches causing the model to overfit and reduce its generalization ability. Conversely, when the setting is below the optimal value, it leads to unstable parameter updates and prevents convergence to the optimal solution.

Table 7 Results of PIFTA4Rec under Different Batch Sizes

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	16	0.0620	0.0516	0.0549	0.0254	0.0360	0.0949	0.0990	0.1577	0.1217	0.1751
	32	0.0745	0.0740	0.0554	0.0421	**0.0364**	0.1074	0.1220	0.1587	**0.1687**	0.1783
	64	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	**0.1243**	**0.1362**	**0.1613**	0.1677	**0.1784**
	128	0.0926	0.0822	0.0555	0.0413	0.0359	0.1210	0.1256	0.1529	0.1608	0.1686
Dunnhumby	16	0.2166	0.2235	0.1693	0.1337	0.1146	0.3204	0.3078	0.3418	**0.3686**	0.3788
	32	**0.2191**	**0.2246**	**0.1703**	**0.1343**	0.1156	**0.3216**	**0.3105**	**0.3419**	0.3682	**0.3800**
	64	0.2168	0.2241	0.1699	0.1336	**0.1158**	0.3183	0.3079	0.3395	0.3650	0.3786
	128	0.2152	0.2199	0.1684	0.1313	0.1134	0.3148	0.3020	0.3345	0.3596	0.3716

The second parameter tested was the Learning Rate. Table 8 shows that the best performance was achieved when the Learning Rate was set to 0.0001 and 0.00001 for the Tafeng and Dunnhumby datasets, respectively. When the Learning Rate is set too low, the model's parameter update step is too small, leading to slow convergence in training and thus failing to update weights sufficiently for better solutions. Conversely, when the Learning Rate is set too high, it can cause the model to oscillate around the minimum, making it difficult to converge correctly to the optimal solution.

Table 8 Results of PIFTA4Rec under Different Learning Rates

	Rate	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	0.001	0.0563	0.0421	0.0251	0.0207	0.0179	0.0933	0.0932	0.1032	0.1115	0.1147
	0.0001	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	**0.1243**	**0.1362**	**0.1613**	**0.1677**	**0.1784**
	0.00001	0.0876	0.0806	0.0549	0.0419	0.0362	0.1156	0.1248	0.1509	0.1626	0.1697
	0.000001	0.0863	0.0809	0.0554	0.0416	0.0353	0.1142	0.1248	0.1518	0.1616	0.1664
Dunnhumby	0.001	0.0713	0.0596	0.0357	0.0274	0.0238	0.1149	0.0993	0.0979	0.1031	0.1062
	0.0001	0.2163	0.2212	0.1695	0.1333	0.1148	0.3170	0.3032	0.3391	0.3656	0.3773
	0.00001	**0.2191**	0.2246	**0.1703**	**0.1343**	**0.1156**	**0.3216**	0.3105	**0.3419**	**0.3682**	**0.3800**
	0.000001	0.2179	**0.2247**	0.1694	0.1320	0.1139	0.3204	0.**3109**	0.3409	0.3645	0.3767

The third parameter tested was the size of the Embedding Size in Item Embedding. Table 9 shows that setting the vector dimension to 64 in both datasets achieved the best performance. The size of the embedding dimension determines the level of detail the model can capture and represent. Too small a dimension may prevent the model from fully capturing the feature information between items, affecting recommendation performance; whereas too large a dimension can not only make it difficult for the model to effectively express item features but also tend to overfit by learning too much detail, including unnecessary noise.

Table 9 Results of PIFTA4Rec with Different Embedding Sizes in Item Embedding

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	16	0.0958	**0.0862**	0.0559	0.0422	0.0357	**0.1243**	0.1360	0.1600	0.1649	0.1714
	32	0.0952	0.0849	**0.0564**	0.0422	0.0352	0.1239	0.1344	0.1555	0.1656	0.1665
	64	**0.0959**	**0.0862**	0.0562	**0.0424**	**0.0364**	**0.1243**	**0.1362**	**0.1613**	**0.1677**	**0.1784**
	128	0.0902	0.0849	0.0558	0.0421	0.0360	0.1176	0.1345	0.1602	0.1652	0.1746
Dunnhumby	16	0.2180	**0.2246**	0.1691	0.1341	0.0941	0.3208	0.3101	0.3400	0.3676	0.3357
	32	0.2184	0.2242	0.1656	0.1339	0.1109	0.3207	0.3102	0.3355	0.3678	0.3711
	64	**0.2191**	**0.2246**	**0.1703**	0.1343	**0.1156**	**0.3216**	**0.3105**	**0.3419**	0.3682	**0.3800**
	128	0.2185	0.2245	0.1700	**0.1344**	0.1154	0.3211	0.3104	0.3413	**0.3683**	0.3797

The fourth parameter tested was the Hidden Size in the MLP. When using an MLP model, choosing the appropriate hidden layer dimension (i.e., the number of neurons) is crucial for achieving good performance. Table 10 shows that setting its size to 256 yielded the best performance in both datasets. When the hidden layer dimension is too small, the neurons do not have enough capacity to capture the feature information of the data, thereby limiting performance; conversely, when the hidden layer dimension is too large, it may lead to an overly complex model, and too many parameters in the model may cause overfitting to the training data.

Table 10 Results of PIFTA4Rec with Different Embedding Sizes in the MLP Hidden Layer

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	64	0.0954	0.0843	0.0559	0.0422	0.0344	0.1242	0.1335	0.1604	0.1632	0.1699
	128	0.0955	0.0846	0.0560	0.0420	0.0353	0.1241	0.1341	0.1607	0.1658	0.1707
	256	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	0.1243	**0.1362**	**0.1613**	**0.1677**	**0.1784**
	512	**0.0959**	0.0852	0.0560	0.0420	0.0361	**0.1244**	0.1345	0.1584	0.1670	0.1740
Dunnhumby	64	0.2179	0.2235	0.1621	0.1339	0.1062	0.3211	0.3093	0.3309	0.3677	0.3607
	128	0.2170	**0.2247**	0.1668	0.1288	0.1086	0.3196	0.3103	0.3370	0.3587	0.3673
	256	**0.2191**	0.2246	**0.1703**	**0.1343**	**0.1156**	**0.3216**	**0.3105**	**0.3419**	0.3682	**0.3800**
	512	0.2185	0.2236	0.1697	**0.1343**	0.1038	0.3212	0.3093	0.3411	**0.3684**	0.3559

The fifth parameter tested was the Decay Rate in Temporal Attention, which determines the impact of the timing of each basket on future baskets. Results in Table 11 show that setting this value to 0.3 yielded the best results. When the decay rate is set too low, it causes the weights to decay too slowly, overly emphasizing historical data while neglecting recent changes and trends; but when the decay rate is set too high, it causes the weights to decay too quickly, considering only the most recent information as important and disregarding valuable historical data, thus leading to the model's insufficient capture of long-term trends.

Table 11 Results of PIFTA4Rec under Different Decay Rates

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	0.1	0.0955	0.0846	0.0559	0.0421	0.0361	0.1241	0.1341	0.1602	0.1638	0.1769
	0.3	**0.0959**	0.0862	**0.0562**	**0.0424**	**0.0364**	**0.1243**	0.1362	**0.1613**	**0.1677**	**0.1784**
	0.5	0.0958	**0.0863**	0.0559	0.0420	0.0362	0.1242	**0.1384**	0.1601	0.1630	0.1771
Dunnhumby	0.1	0.2179	0.2243	0.1702	0.1336	0.1154	0.3202	0.3100	0.3416	0.3670	0.3794
	0.3	**0.2191**	**0.2246**	**0.1703**	**0.1343**	**0.1156**	**0.3216**	**0.3105**	**0.3419**	**0.3682**	**0.3800**
	0.5	0.2185	0.2240	0.1701	0.1324	0.1155	0.3210	0.3091	0.3418	0.3644	0.3795

The sixth parameter tested was the Dropout Rate in the Transformer, which directly influences the probability of neurons being randomly dropped during model training, thus significantly affecting model performance. If the Dropout value is set too low, the number of neurons dropped is insufficient to provide adequate regularization, making the model prone to overfitting; however, if the Dropout value is too high, too many neurons are dropped, causing the model to lose too much important information and thereby hinder effective learning, impacting performance. Table 12 shows that setting the Dropout Rate to 0.3 achieves the best performance on both datasets.

Table 12 Results of PIFTA4Rec under Different Dropout Rates

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	0.1	0.0955	0.0848	**0.0562**	0.0421	0.0362	0.1241	0.1343	0.1607	0.1652	0.1710
	0.3	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	0.1243	**0.1362**	**0.1613**	**0.1677**	**0.1784**
	0.5	**0.0959**	0.0860	0.0558	0.0420	0.0358	**0.1244**	0.1358	0.1542	0.1648	0.1761
Dunnhumby	0.1	0.2184	0.2241	0.1693	0.1340	0.1155	0.3209	0.3100	0.3403	0.3671	0.3792
	0.3	**0.2191**	**0.2246**	**0.1703**	**0.1343**	0.1156	**0.3216**	**0.3105**	**0.3419**	**0.3682**	**0.3800**
	0.5	0.2185	0.2241	0.1699	0.1339	**0.1157**	0.3213	0.3093	0.3412	0.3673	**0.3800**

The seventh parameter tested was the number of Heads in the Transformer's Multi-Head Attention, which enhances the model's ability to capture feature information by learning the input data in parallel. Although differences between various numbers of Heads were minor, Table 13 shows that using four Heads yields the best results in both datasets. If the number of Heads is set too low, it restricts the model's capacity to express and capture important features. Conversely, if the number of Heads is set too high, it not only increases the computational cost of training the model but may also lead to overfitting, affecting performance.

Table 13 Results of PIFTA4Rec in the Transformer with Different Numbers of Heads

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	1	0.0955	0.0818	0.0559	0.0421	0.0356	0.1241	0.1266	0.1557	0.1652	0.1685
	2	0.0958	0.0850	0.0560	0.0422	0.0361	**0.1243**	0.1344	0.1604	0.1654	0.1695
	4	**0.0959**	**0.0862**	**0.0562**	**0.0424**	**0.0364**	**0.1243**	**0.1362**	**0.1613**	**0.1677**	**0.1784**
	8	0.0954	0.0860	0.0560	0.0421	0.0362	0.1241	0.1360	0.1539	**0.1677**	0.1726
Dunnhumby	1	0.2182	0.2244	0.1696	0.1340	**0.1156**	0.3205	0.3100	0.3411	0.3675	0.3796
	2	0.2179	0.2245	0.1698	0.1332	0.1155	0.3201	0.3101	0.3412	0.3666	0.3799
	4	**0.2191**	**0.2246**	**0.1703**	**0.1343**	**0.1156**	**0.3216**	**0.3105**	**0.3419**	**0.3682**	**0.3800**
	8	0.2167	0.2243	0.1699	0.1338	0.1153	0.3197	0.3099	0.3414	0.3674	0.3789

The final test was the number of layers in the Transformer Encoder. Results from Table 14 show that the differences between using various numbers of layers are minimal and that using just one layer of the Transformer Encoder achieves optimal performance. This finding indicates that one layer is sufficient to effectively capture key information in this recommendation task without the need for an overly complex multi-layer model. Setting too many layers can lead to overfitting to noise and details in the training data, as well as potential issues with gradient vanishing or exploding, making the model difficult to train.

Table 14 Results of PIFTA4Rec in the Transformer with Different Numbers of Layers

	Size	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	**1**	**0.0959**	0.0862	**0.0562**	**0.0424**	**0.0364**	**0.1243**	0.1362	**0.1613**	**0.1677**	**0.1784**
	2	0.0953	0.0851	0.0559	0.0422	0.0362	0.1240	0.1344	0.1599	0.1668	0.1772
	4	0.0954	**0.0868**	0.0546	0.0423	0.0361	0.1240	**0.1387**	0.1509	0.1655	0.1708
	8	0.0954	0.0848	0.0556	0.0422	0.0362	0.1240	0.1342	0.1525	0.1654	0.1697
Dunnhumby	**1**	**0.2191**	**0.2246**	**0.1703**	**0.1343**	0.1156	**0.3216**	**0.3105**	**0.3419**	**0.3682**	**0.3800**
	2	0.2189	0.2241	0.1700	0.1341	0.1155	0.3214	0.3101	0.3414	0.3677	0.3797
	4	0.2184	0.2237	0.1702	0.1337	**0.1159**	0.3208	0.3098	0.3417	0.3676	**0.3800**
	8	0.2189	0.2243	0.1698	0.1340	0.1154	0.3212	0.3103	0.3411	0.3676	0.3791

5.3 Experiment 3: Ablation Experiment

To verify the effectiveness of the two sub-modules, KIM and DLIM, in the model, we conducted ablation experiments on these sub-modules. By individually removing specific components from the complete model and analyzing their contributions, we determined their impact on the overall performance of our study model, PIFTA4Rec. The model names and descriptions for the ablation experiments are as follows:

Table 15 Ablation Experiment Model Names and Descriptions

Model	Descriptions
KIM	PIFTA4Rec with the DLIM sub-module removed, leaving only the KIM part which uses a learnable parameter α to control the weights for the target user vector and the average neighbor vector for making recommendations.
DLIM	PIFTA4Rec with the KIM sub-module removed, leaving only the DLIM part which uses Temporal Attention and Multi-Head Attention to train on the user's original historical basket sequences to learn sequence relationships and user purchasing preferences for recommendations.
PIFTA4Rec	The complete model incorporates both KIM and DLIM.

Table 16 shows the results of the ablation experiments in the two datasets, from which we can make the following observations:

- **Observation 1**: In both datasets, using KIM alone achieves performance close to the full PIFTA4Rec model while using DLIM alone shows a noticeable performance gap. This indicates that in the PIFTA4Rec model, KIM contributes more significantly to the overall performance. Additionally, this finding reveals that traditional methods, when paired with simple attention mechanisms, may suffice to match or even surpass more complex deep learning models, offering potential value in enhancing the efficiency and accuracy of basket recommendation systems.
- **Observation 2**: There is a noticeable difference in performance between the complete model and the KIM sub-model across two datasets with slightly different characteristics.

Specifically, in the Tafeng dataset, which has fewer baskets but a larger number of items, KIM performs better at lower recommendation sizes (K values of 5 and 10). However, as the recommendation size increases (K values of 30, 50, 65), the complete model PIFTA4Rec significantly enhances its performance, particularly in terms of the NDCG metric. Conversely, in the Dunnhumby dataset, which has more baskets but fewer items, although the complete model PIFTA4Rec generally outperforms KIM, the gap between them is much smaller compared to the Tafeng dataset. This suggests that while the simple KIM model can effectively handle scenarios with fewer items, a more comprehensive model like PIFTA4Rec is needed as item quantity increases. This is because PIFTA4Rec not only considers the repeated purchase behavior information of each user but also learns the complex short-term and long-term associations between users and items, thus providing more accurate recommendations when faced with a rich array of item choices.

- **Observation 3**: Although DLIM performs slightly less well than PIFTA4Rec and KIM, it still surpasses many current neural network-based recommendation models. As shown in Table 17, in the Tafeng dataset, DLIM outperforms four of the five baseline models; in the Dunnhumby dataset, it surpasses three. This demonstrates that while DLIM may fall short in direct comparisons with PIFTA4Rec and KIM, its overall recommendation effectiveness remains competitive. Furthermore, following on from Observation Two, the complete model PIFTA4Rec (including DLIM) typically shows more stable or even superior recommendation performance across different datasets. This indicates that DLIM has a certain level of influence and importance in handling large-scale or item-rich datasets in the complete model PIFTA4Rec.

Table 16 Comparison of Ablation Experiment Results

	Model	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	PIFTA4Rec	0.0959	0.0862	**0.0562**	**0.0424**	**0.0364**	0.1243	0.1362	**0.1613**	**0.1677**	**0.1784**
	KIM	**0.0981**	**0.0870**	0.0552	0.0421	0.0362	**0.1283**	**0.1379**	0.1518	0.1633	0.1695
	DLIM	0.0653	0.0528	0.0359	0.0286	0.0255	0.1014	0.1051	0.1231	0.1326	0.1373
Dunnhumby	PIFTA4Rec	**0.2191**	**0.2246**	**0.1703**	0.1343	0.1156	**0.3216**	**0.3105**	**0.3419**	**0.3682**	0.3800
	KIM	0.2186	0.2245	0.1701	**0.1344**	**0.1161**	0.3212	0.3101	0.3415	**0.3682**	**0.3804**
	DLIM	0.0955	0.0893	0.0683	0.0562	0.0503	0.1484	0.1311	0.1419	0.1557	0.1630

Table 17 DLIM Compared to Other Neural Network Baselines

	Baseline	F1-score @5	F1-score @10	F1-score @30	F1-score @50	F1-score @65	NDCG @5	NDCG @10	NDCG @30	NDCG @50	NDCG @65
Tafeng	DREAM	0.0506	0.0410	0.0287	0.0252	0.0233	0.0715	0.0742	0.0779	0.0800	0.0810
	CLEA	0.0501	0.0422	0.0289	0.0235	0.0210	0.0813	0.0922	0.1089	0.1175	0.1223
	Beacon	0.0553	0.0480	0.0316	0.0259	0.0233	**0.1065**	**0.1117**	**0.1248**	**0.1335**	**0.1383**
	SHAN	0.0509	0.0387	0.0273	0.0230	0.0216	0.0839	0.0892	0.1028	0.1112	0.1170
	Sets2Sets	0.0536	0.0508	**0.0391**	**0.0310**	**0.0274**	0.0740	0.0885	0.1140	0.1232	0.1279
	DLIM	**0.0653**	**0.0528**	0.0359	0.0286	0.0255	0.1014	0.1051	0.1231	0.1326	0.1373
Dunnhumby	DREAM	0.0842	0.0789	0.0581	0.0474	0.0420	0.0841	0.0893	0.0949	0.0971	0.0982
	CLEA	**0.1313**	**0.1168**	**0.0849**	**0.0701**	**0.0619**	**0.1766**	**0.1957**	**0.2276**	**0.2452**	**0.2537**
	Beacon	0.0819	0.0770	0.0588	0.0490	0.0443	0.1516	0.1349	0.1427	0.1548	0.1621
	SHAN	0.0822	0.0784	0.0581	0.0492	0.0442	0.1035	0.1198	0.1430	0.1569	0.1641
	Sets2Sets	0.0806	0.0878	0.0752	0.0613	0.0554	0.0958	0.1231	0.1648	0.1805	0.1892
	DLIM	0.0955	0.0893	0.0683	0.0562	0.0503	0.1484	0.1311	0.1419	0.1557	0.1630

6. SUMMARY

6.1 Conclusion

This study introduces a hybrid recommendation model, PIFTA4Rec, which combines traditional data mining techniques with contemporary deep learning methods for the next-basket recommendation task. The model is implemented in two sub-models: KIM and DLIM. Firstly, KIM builds upon the previous research[34] by directly utilizing user-specific item frequency information through simple learning and combines this with the traditional KNN method to obtain initial predictions from similar users. Additionally, KIM also identifies specific neighbors for each target user for the next sub-model, DLIM. DLIM then utilizes the users' original purchase sequences and employs a complex architecture with stacked temporal attention and multi-head attention mechanisms to learn the intricate associations between users and items, thereby generating the second stage of prediction results. Ultimately, the model combines the predictions from these two sub-models to provide the final recommendations.

Focusing on integrating personalized item frequency with attention mechanisms, the PIFTA4Rec model achieved the best recommendation performance in two real-world datasets compared to other baselines. These experimental results not only demonstrate the effectiveness of such a hybrid model in handling time-sensitive basket data, especially in capturing long-term user preferences and the dynamic changes in their purchasing behaviors but also confirm that combining attention mechanisms can utilize PIF information more effectively, thus achieving superior recommendation results.

Through the sensitivity analysis conducted in Experiment 2, we gained further insights into the impact of each hyperparameter on the model's performance, identifying the optimal parameter configuration for achieving the best performance. The analysis revealed that PIFTA4Rec is sensitive to the Batch Size and Learning Rate. When the Batch Size is too small,

parameter updates are unstable, and if the Learning Rate is too high, effective learning cannot occur. Moreover, PIFTA4Rec generally exhibits stable performance across most parameter settings, demonstrating its robustness and adaptability to different training conditions. Experiment 3's ablation study revealed how each of the two sub-models affects the final recommendations of PIFTA4Rec. The results indicated that in datasets with fewer items, the first sub-model, KIM, achieved good recommendation performance through simple learning methods. However, as the number of items increased, incorporating the second, more complex sub-model, DLIM, resulted in more robust and even superior recommendation outcomes. This not only shows that PIFTA4Rec can flexibly adapt to the characteristics of different datasets but also further proves the necessity and importance of each sub-model to the overall architecture.

6.2 Future Work

The PIFTA4Rec method proposed in this study significantly improves basket recommendation performance, yet it still presents limitations and avenues for further research. Firstly, in the KIM stage, the PIFTA4Rec model primarily uses PIF information to construct vectors for users and items, which may limit the richness of vector representations. In the future, we can enrich vector representations by introducing more features and data sources. For example, integrating multi-source data such as users' browsing history, click behavior, and rating data, and even considering adding contextual information like holidays, location, weather, and users' social relationships. These diverse data sources will help build more comprehensive vector representations for each user and item, better capturing complex behavioral patterns and potential preferences.

Secondly, in the DLIM stage, we can improve the Temporal Attention method to enhance the model's prediction accuracy and adaptability. The current approach uses a learnable decay rate with an exponential decay formula to determine the weight of each basket, but this method

may not sufficiently capture temporal feature information. Therefore, we can consider introducing more complex, multi-layer time decay, or combining time embeddings and multi-head time attention mechanisms. Additionally, we can explore integrating some models that are very effective in handling time-series data, such as LSTM[72], GRU[73], and TCN[74], leveraging their advantages in dealing with long-term dependencies and sequential data to further enhance the model's capability in temporal dynamic modeling.

Lastly, the study observed distinctly different performances between the complete model and its sub-models when using two datasets with slightly different characteristics. In response to this phenomenon, we plan to use a more diverse set of datasets for extensive experiments in the future. This will help us better understand the adaptability of our model comprehensively and continue to fine-tune and optimize the model to enhance its performance and reliability across different application scenarios.

REFERENCES

[1] Darban, Z.Z. and M.H. Valipour, *GHRS: Graph-based hybrid recommendation system with application to movie recommendation.* Expert Systems with Applications, 2022. **200**: p. 116850.

[2] Zhang, Z. and B. Wang, *Prompt learning for news recommendation.* arXiv preprint arXiv:2304.05263, 2023.

[3] Singh, J., *An efficient deep neural network model for music classification.* International Journal of Web Science, 2022. **3**(3): p. 236-248.

[4] Liao, J., W. Zhou, F. Luo, J. Wen, M. Gao, X. Li, and J. Zeng, *SocialLGN: Light graph convolution network for social recommendation.* Information Sciences, 2022. **589**: p. 595-607.

[5] Steck, H. *Evaluation of recommendations: rating-prediction and ranking*. in *Proceedings of the 7th ACM conference on Recommender systems*. 2013.

[6] Cremonesi, P., Y. Koren, and R. Turrin. *Performance of recommender algorithms on top-n recommendation tasks*. in *Proceedings of the fourth ACM conference on Recommender systems*. 2010.

[7] Anelli, V.W., A. Bellogín, T. Di Noia, D. Jannach, and C. Pomo. *Top-n recommendation algorithms: A quest for the state-of-the-art*. in *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*. 2022.

[8] Deshpande, M. and G. Karypis, *Item-based top-N recommendation algorithms.* ACM Trans. Inf. Syst., 2004. **22**(1): p. 143–177.

[9] Karypis, G. *Evaluation of item-based top-n recommendation algorithms*. in *Proceedings of the tenth international conference on Information and knowledge management*. 2001.

[10] Palumbo, E., G. Rizzo, and R. Troncy. *Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation*. in *Proceedings of the eleventh ACM conference on recommender systems*. 2017.

[11] Wang, J., K. Ding, L. Hong, H. Liu, and J. Caverlee. *Next-item recommendation with sequential hypergraphs*. in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 2020.

[12] Ariannezhad, M., M. Li, S. Schelter, and M. de Rijke. *A personalized neighborhood-based model for within-basket recommendation in grocery shopping*. in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 2023.

[13] Qin, Y., 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*. 2021.

[14] Ariannezhad, M., S. Jullien, M. Li, M. Fang, S. Schelter, and M. de Rijke. *ReCANet: A*

repeat consumption-aware neural network for next basket recommendation in grocery shopping. in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022.

[15] Shen, Y., B. Ou, and R. Li, *MBN: Towards multi-behavior sequence modeling for next basket recommendation.* ACM Transactions on Knowledge Discovery from Data (TKDD), 2022. **16**(5): p. 1-23.

[16] He, X., L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, *Neural Collaborative Filtering*, in *Proceedings of the 26th International Conference on World Wide Web*. 2017, International World Wide Web Conferences Steering Committee: Perth, Australia. p. 173–182.

[17] Lops, P., M. De Gemmis, and G. Semeraro, *Content-based recommender systems: State of the art and trends.* Recommender systems handbook, 2011: p. 73-105.

[18] Koren, Y., R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems.* Computer, 2009. **42**(8): p. 30-37.

[19] Xia, L., C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang. *Hypergraph contrastive collaborative filtering*. in *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*. 2022.

[20] Srifi, M., A. Oussous, A. Ait Lahcen, and S. Mouline, *Recommender systems based on collaborative filtering using review texts—a survey.* Information, 2020. **11**(6): p. 317.

[21] Wang, X., X. He, M. Wang, F. Feng, and T.-S. Chua. *Neural graph collaborative filtering*. in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 2019.

[22] Xia, X., H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang. *Self-supervised hypergraph convolutional networks for session-based recommendation*. in *Proceedings of the AAAI conference on artificial intelligence*. 2021.

[23] Tang, J. and K. Wang. *Personalized top-n sequential recommendation via convolutional sequence embedding*. in *Proceedings of the eleventh ACM international conference on web search and data mining*. 2018.

[24] Kumar, G., H. Jerbi, and M.P. O'Mahony, *A sequence-based and context modelling framework for recommendation.* Expert Systems with Applications, 2021. **175**: p. 114665.

[25] Damak, K., O. Nasraoui, and W.S. Sanders, *Sequence-based explainable hybrid song recommendation.* Frontiers in big Data, 2021. **4**: p. 693494.

[26] Wang, D., D. Xu, D. Yu, and G. Xu, *Time-aware sequence model for next-item recommendation.* Applied Intelligence, 2021. **51**: p. 906-920.

[27] Salehinejad, H., S. Sankar, J. Barfett, E. Colak, and S. Valaee, *Recent advances in recurrent neural networks.* arXiv preprint arXiv:1801.01078, 2017.

[28] Lin, T., Y. Wang, X. Liu, and X. Qiu, *A survey of transformers.* AI Open, 2022.

[29] Anderson, A., R. Kumar, A. Tomkins, and S. Vassilvitskii. *The dynamics of repeat*

consumption. in *Proceedings of the 23rd international conference on World wide web*. 2014.

[30] Jacoby, J. and D.B. Kyner, *Brand loyalty vs. repeat purchasing behavior.* Journal of Marketing research, 1973. **10**(1): p. 1-9.

[31] Ren, P., Z. Chen, J. Li, Z. Ren, J. Ma, and M. De Rijke. *Repeatnet: A repeat aware neural recommendation machine for session-based recommendation*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

[32] Rappaz, J., J. McAuley, and K. Aberer. *Recommendation on live-streaming platforms: Dynamic availability and repeat consumption*. in *Proceedings of the 15th ACM Conference on Recommender Systems*. 2021.

[33] Wang, C., M. Zhang, W. Ma, Y. Liu, and S. Ma. *Modeling item-specific temporal dynamics of repeat consumption for recommender systems*. in *The world wide web conference*. 2019.

[34] Hu, H., 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*. 2020.

[35] Hu, H. 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*. 2019.

[36] Wan, M., D. Wang, J. Liu, P. Bennett, and J. McAuley. *Representing and recommending shopping baskets with complementarity, compatibility and loyalty*. in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018.

[37] Gu, W., S. Dong, and Z. Zeng, *Increasing recommended effectiveness with markov chains and purchase intervals.* Neural Computing and Applications, 2014. **25**: p. 1153-1162.

[38] Shani, G., D. Heckerman, R.I. Brafman, and C. Boutilier, *An MDP-based recommender system.* Journal of Machine Learning Research, 2005. **6**(9).

[39] Bokde, D., S. Girase, and D. Mukhopadhyay, *Matrix factorization model in collaborative filtering algorithms: A survey.* Procedia Computer Science, 2015. **49**: p. 136-146.

[40] Rendle, S. *Factorization machines*. in *2010 IEEE International conference on data mining*. 2010. IEEE.

[41] Loni, B., Y. Shi, M. Larson, and A. Hanjalic. *Cross-domain collaborative filtering with factorization machines*. in *Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36*. 2014. Springer.

[42] Chen, C., C. Hou, J. Xiao, and X. Yuan, *Purchase behavior prediction in e-commerce with factorization machines.* IEICE TRANSACTIONS on Information and Systems, 2016. **99**(1): p. 270-274.

[43] Wei, S., N. Ye, S. Zhang, X. Huang, and J. Zhu. *Item-based collaborative filtering*

recommendation algorithm combining item category with interestingness measure. in *2012 international conference on computer science and service system*. 2012. IEEE.

[44] Tewari, A.S. and A.G. Barman, *Collaborative recommendation system using dynamic content based filtering, association rule mining and opinion mining.* International Journal of Intelligent Engineering & Systems, 2017. **10**(5).

[45] Zhu, H., B. Huberman, and Y. Luon. *To switch or not to switch: understanding social influence in online choices*. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012.

[46] Kumar, R., B. Verma, and S.S. Rastogi, *Social popularity based SVD++ recommender system.* International Journal of Computer Applications, 2014. **87**(14).

[47] Javari, A. and M. Jalili, *Accurate and novel recommendations: an algorithm based on popularity forecasting.* ACM Transactions on Intelligent Systems and Technology (TIST), 2014. **5**(4): p. 1-20.

[48] He, R. and J. McAuley. *Fusing similarity models with markov chains for sparse sequential recommendation*. in *2016 IEEE 16th international conference on data mining (ICDM)*. 2016. IEEE.

[49] Rendle, S., 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*. 2010.

[50] Wang, P., 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*. 2015.

[51] Yu, F., 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*. 2016.

[52] Wang, S., L. Hu, Y. Wang, Q.Z. Sheng, M. Orgun, and L. Cao. *Intention nets: psychology-inspired user choice behavior modeling for next-basket prediction*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[53] Xu, C., P. Zhao, Y. Liu, J. Xu, V.S.S. S. Sheng, Z. Cui, X. Zhou, and H. Xiong. *Recurrent convolutional neural network for sequential recommendation*. in *The world wide web conference*. 2019.

[54] Huang, C., X. Wu, X. Zhang, C. Zhang, J. Zhao, D. Yin, and N.V. Chawla. *Online purchase prediction via multi-scale modeling of behavior dynamics*. in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019.

[55] Zhang, S., Y. Tay, L. Yao, and A. Sun, *Next item recommendation with self-attention.* arXiv preprint arXiv:1808.06414, 2018.

[56] Bai, T., 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*. 2018.

[57] Chen, Y., J. Li, C. Liu, C. Li, M. Anderle, J. McAuley, and C. Xiong, *Modeling dynamic attributes for next basket recommendation.* arXiv preprint arXiv:2109.11654, 2021.

[58] Liu, Z., M. Wan, S. Guo, K. Achan, and P.S. Yu. *Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network*. in *Proceedings of the 2020 SIAM International Conference on Data Mining*. 2020. SIAM.

[59] Pang, Y., L. Wu, Q. Shen, Y. Zhang, Z. Wei, F. Xu, E. Chang, B. Long, and J. Pei. *Heterogeneous global graph neural networks for personalized session-based recommendation*. in *Proceedings of the fifteenth ACM international conference on web search and data mining*. 2022.

[60] Liu, Z., X. Li, Z. Fan, S. Guo, K. Achan, and S.Y. Philip. *Basket recommendation with multi-intent translation graph neural network*. in *2020 IEEE International Conference on Big Data (Big Data)*. 2020. IEEE.

[61] Yang, J., J. Xu, J. Tong, S. Gao, J. Guo, and J. Wen, *Pre-training of context-aware item representation for next basket recommendation.* arXiv preprint arXiv:1904.12604, 2019.

[62] Liu, T. and B. Liu. *Next basket recommendation based on graph attention network and transformer*. in *Journal of Physics: Conference Series*. 2022. IOP Publishing.

[63] Sun, L., Y. Bai, B. Du, C. Liu, H. Xiong, and W. Lv. *Dual sequential network for temporal sets prediction*. in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020.

[64] Faggioli, G., 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*. 2020.

[65] Ying, H., F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu. *Sequential recommender system based on hierarchical attention network*. in *IJCAI International Joint Conference on Artificial Intelligence*. 2018.

[66] Luo, Y., Q. Liu, and Z. Liu. *Stan: Spatio-temporal attention network for next location recommendation*. in *Proceedings of the web conference 2021*. 2021.

[67] Rong, X., *word2vec parameter learning explained.* arXiv preprint arXiv:1411.2738, 2014.

[68] Barkan, O. and N. Koenigstein. *Item2vec: neural item embedding for collaborative filtering*. in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2016. IEEE.

[69] Li, M., S. Jullien, M. Ariannezhad, and M. de Rijke, *A next basket recommendation reality check.* ACM Transactions on Information Systems, 2023. **41**(4): p. 1-29.

[70] Le, D.-T., H.W. Lauw, and Y. Fang, *Correlation-sensitive next-basket recommendation.* 2019.

[71] Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint

arXiv:1412.6980, 2014.

[72] Yu, Y., X. Si, C. Hu, and J. Zhang, *A review of recurrent neural networks: LSTM cells and network architectures.* Neural computation, 2019. **31**(7): p. 1235-1270.

[73] Dey, R. and F.M. Salem. *Gate-variants of gated recurrent unit (GRU) neural networks*. in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. 2017. IEEE.

[74] Fan, J., K. Zhang, Y. Huang, Y. Zhu, and B. Chen, *Parallel spatio-temporal attention-based TCN for multivariate time series prediction.* Neural Computing and Applications, 2023. **35**(18): p. 13109-13118.