informatics

MDPI

Article

CAMBSRec: A Context-Aware Multi-Behavior Sequential Recommendation Model

Bohan Zhuang [1], Yan Lan [1] and Minghui Zhang [2,*]

[1] School of Information and Communication Engineering, Dalian Minzu University, Dalian 116600, China; zhuangbohan0331@163.com (B.Z.); lanyan@dlnu.edu.cn (Y.L.)
[2] School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116034, China
* Correspondence: zhangmh@dlpu.edu.cn

Abstract

Multi-behavior sequential recommendation (MBSRec) is a form of sequential recommendation. It leverages users' historical interaction behavior types to better predict their next actions. This approach fits real-world scenarios better than traditional models do. With the rise of the transformer model, attention mechanisms are widely used in recommendation algorithms. However, they suffer from low-pass filtering, and the simple learnable positional encodings in existing models offer limited performance gains. To address these problems, we introduce the context-aware multi-behavior sequential recommendation model (CAMBSRec). It separately encodes items and behavior types, replaces traditional positional encoding with context-similarity positional encoding, and applies the discrete Fourier transform to separate the high and low frequency components and enhance the high frequency components, countering the low-pass filtering effect. Experiments on three public datasets show that CAMBSRec performs better than five baseline models, demonstrating its advantages in terms of recommendation performance.

Keywords: recommendation; multi-behavior; sequential; positional encoding; inductive bias

Academic Editor: Yang Shi

Received: 29 May 2025
Revised: 20 July 2025
Accepted: 31 July 2025
Published: 4 August 2025

Citation: Zhuang, B.; Lan, Y.; Zhang, M. CAMBSRec: A Context-Aware Multi-Behavior Sequential Recommendation Model. *Informatics* **2025**, *12*, 79. https://doi.org/10.3390/informatics12030079

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

Benefiting from advancements in machine learning, recommendation systems have undergone remarkable development. Currently, they have become a critical application in their field. They are widely applied in various domains, such as e-commerce [1], social media [2], online advertising [3], and healthcare. These systems can be categorized into specific types, including but not limited to sequential recommendation and multi-behavior recommendation. The core objective of recommendation systems is to provide users with personalized content recommendations (e.g., products, movies, music, or articles) through algorithmic models, thereby enhancing the user experience and platform revenue. With the proposal of various superior recommendation algorithms, the user experience has significantly improved.

From early matrix factorization [4] and collaborative filtering [5] to the extensive application of deep learning, the development of recommendation systems has been remarkable. During this period, numerous recommendation systems integrating neural networks have been developed. These systems generate recommendations for users on the basis of their historical interaction data. In most real-world recommendation scenarios, user interaction data exhibit two critical characteristics: sequentiality and multi-behavior. Users interact with items in a sequential manner, and interests vary across users, making it

essential to consider the dynamic sequence of user behaviors. Additionally, users typically interact with items through multiple behaviors. For example, on e-commerce platforms, interactions include clicking, favoriting, adding to carts, and purchasing. The incorporation of multi-behavioral features allows for the understanding of user intentions through different behavior types, the capture of fine-grained user interests, and the delivery of personalized recommendations.

In recent years, several studies on sequential models and multi-behavioral approaches have been proposed. Since the introduction of the transformer [6], its core component, "self-attention," has become one of the prevalent modeling methods. For example, Self-Attentive Sequential Recommendation (SASRec) [7] employs multi-head self-attention to model historical interaction sequences. Owing to the advantages of capturing long-range dependencies and parallel computations, self-attention has significantly enhanced performance. For multi-behavioral recommendation, early methods mostly relied on graph convolutional networks (e.g., MB-GCN [8] and MB-GMN models [9]), whereas recent approaches incorporate self-attention mechanisms in their architectures to capture long-short-term dependencies (e.g., MBHT [10] and MB-STR [11]). Despite their excellent performance in some scenarios, these self-attention-based models still possess several limitations:

(1) Self-attention has an inherent insufficient inductive bias when processing sequential data. Although it can capture long-range dependencies, it may not fully consider certain fine-grained sequential patterns. Additionally, it may overfit the training data, which can lead to a weak generalization ability.
(2) As self-attention processes the entire data range, it may inadvertently ignore important, detailed patterns, causing an oversmoothing problem. This problem hinders the model's ability to capture critical temporal dynamics and provide accurate predictions.
(3) Existing self-attention mechanisms in a sequential recommendation typically achieve position awareness through absolute positional encoding, which assigns a learnable vector to each position. However, user historical behavior sequences may have diverse characteristics. For example, users may purchase items of the same type multiple times within a session, where relative positions or context-dependent positions might be more important than absolute positions. Traditional positional encoding does not consider the contextual information of user behaviors; thus, it fails to capture users' dynamic interests effectively.

In summary, this paper proposes a novel context-aware multi-behavior sequential recommendation model (CAMBSRec) to address these limitations, with the following contributions:

(1) We design a context-aware multi-behavior sequential recommendation approach that models user behaviors and items separately. The method employs an inductive bias self-attention layer to capture long-short-term dependencies in multi-behavioral data, introduces a Fourier transform to balance long-short-term interest preferences, and designs a high-pass filter to alleviate the oversmoothing issue. Additionally, a weighted binary cross-entropy loss function is utilized to balance different behaviors, enabling fine-grained control over the weight ratios of each behavioral type.
(2) For a personalized forward recommendation, we design location encoding on the basis of context similarity. Location information is determined by the dissimilarity between context items and target items, rather than by a fixed order. This allows related items to share similar position codes. Therefore, the semantic relevance of the location representation is enhanced.
(3) We conducted extensive experiments on three multi-behavior datasets, and the results demonstrate that our method achieves better results than five baseline methods on all three datasets.

2. Related Work

Related work can be taxonomically organized into three research streams: sequential recommendation, multi-behavior recommendation, and positional encoding.

2.1. Sequential Recommendation

A sequential recommendation represents a critical subdomain within recommender systems, leveraging users' historical interaction sequences to predict the next item most likely to engage their interest. Numerous sequential recommendation models have been proposed to date. Early approaches, such as FPMC [12], achieved personalized sequential recommendations by integrating matrix factorization with Markov chains [13]. However, when the user behavior data are sparse, it is difficult for the model to learn the parameters effectively, especially the lack of interactions of new users or long-tail items, which leads to a decrease in prediction accuracy. Deep learning advancements have led to architectures such as GRU4Rec [14,15] and Caser [16]. GRU4Rec uses gated recurrent unit (GRU) networks to model user interactive sequences, but GRU4Rec relies heavily on the current moment's input and hidden state to predict the next item, but actual user interests may span multiple time steps (e.g., a user clicks multiple times in between and then purchases the item in question again), which leads to the possibility that it may not be able to effectively correlate these kinds of long-distance dependencies. In addition, Caser treats sequence-embedded matrices as visual inputs. It then extracts user preferences through convolutional neural networks (CNNs [17]). However, it imposes overly strong assumptions on sequence models. Due to the necessity of manually presetting the height of convolutional kernels, it cannot adapt to sequences of varying lengths. Moreover, the single-dimensional interest vector output struggles to encompass the complexity of user behaviors.

Following transformative breakthroughs in natural language processing, particularly the advent of the transformer, attention-based sequential recommendation models have gained prominence. Notable examples are SASRec and TiSASRec [18]. SASRec captures sequential relationships among historical items via a multi-head self-attention mechanism. TiSASRec, an enhanced version, models the temporal dynamics in user behavior sequences by introducing temporal interval information. However, for these models, although parallel computation is achieved through the self-attention mechanism, their time complexity is still proportional to the square of the sequence length, and when the length of the user behavior sequence exceeds 500, the training and inference speed of the model decreases significantly and the performance tends to be saturated; the model relies on the sequence of item IDs only and does not integrate the attributes of the items explicitly, so that it is not able to sense other external factors, resulting in the recommendation that results and real-time demand a disconnect.

2.2. Multi-Behavior Recommendation

Multi-behavior recommendation enhances recommendation efficacy by jointly modeling diverse user-item interactions (e.g., clicks, favorites, cart additions, purchases, ratings, shares) and exploiting behavioral disparities and interdependencies. Recent multi-behavior recommendation methodologies primarily follow two technical trajectories. The first leverages graph convolutional networks (GCNs [19]). For example, MB-GCN explicitly constructs multi-behavior heterogeneous graphs to model user-item interactions. It aggregates cross-behavior neighbor information for embedding learning. In contrast, MB-GMN extends this paradigm by integrating memory networks with external memory units. These models mainly rely on a graph convolution of higher-order neighborhood information of users/goods, but they are limited in capturing the heterogeneous features of different behavior types (e.g., clicking, purchasing, and bookmarking), and they have to construct

independent subgraphs for each behavior and perform GCN operations during training, which leads to a sharp increase in computational complexity and exponential growth in training time.

The second leverage adopts transformer architectures for performance enhancement: MBHT [10] employs low-rank self-attention layers to capture short- and long-term cross-behavior dependencies, complemented by hypergraph convolutional networks for global multi-behavior relationship modeling, but the model also suffers from distortion of sparse behavior representation, as the hypergraph aggregation process tends to amplify the signals of high-frequency behaviors (e.g., "clicks"), resulting in suppression of long-tailed behavioral features, as well as its computational complexity, which makes it extremely inefficient for training and causes inference with large datasets. Concurrently, MB-STR [11] introduces a multi-behavior sequence generator to encode heterogeneous sequential patterns across behaviors, subsequently embedding temporal multi-behavior dependencies into attention bias matrices to effectively capture cross-behavior temporal dynamics; however, sparse behaviors (e.g., user's final purchase) and long behavioral sequences lead to a large variance in its strategy gradient and convergence difficulties.

2.3. Positional Encoding

In early sequential recommendation systems, such as Markov chain-based methods, explicit positional encoding was not employed. Instead, the sequential order was implicitly handled through the model's inherent architecture. With the advent of deep learning, recurrent neural network (RNN [20]) and long-short-term memory (LSTM [21]) models have been introduced. While these architectures inherently process sequential data, they do not require explicit positional encoding because of their temporal recurrence mechanisms. When the transformer model is applied to a recommendation system, since the transformer is proposed for Natural Language Processing (NLP), its core component, self-attention, does not contain positional information. Take machine translation as an example: "I eat the fish" and "the fish eats me" are the same for self-attention, but the actual meanings are completely different. So, assigning a unique positional representation to each element in the sequence—i.e., positional encoding—is essential. For example, SASRec adopts a simplified learnable absolute positional encoding, assigning a trainable embedding vector to each position in the sequence to explicitly represent the positional context. Later, Peter Shaw et al. [22] proposed relative positional encoding to capture positional relationships between items.

Nevertheless, these positional methods [23,24] focus solely on the sequential order of items while neglecting their intrinsic attributes. In contrast, content-aware positional encoding (CoPE [25]) presents a new paradigm. In this paradigm, positional information adapts dynamically to the contextual content. This allows for a more flexible and context-aware processing of sequential data.

3. Problem Formulation

In the sequential multi-behavior recommendation task, we define a set of users $\mathrm{U} = \{1, \cdots, \mathrm{u}\}$. For each user u, there exists a sequence of items $S^u = \left[(x_1, y_1), (x_2, y_2), \ldots, \left(x_{|S^u|}, y_{|S^u|}\right)\right]$, which represents a time-ordered interaction sequence where each interaction item x is associated with an interaction behavior type y, such as browsing and favoriting.

The primary objective of the task is to rank the target item list on the basis of all the historical interactions of the target user u before time step $t + 1$.

In this context, we regard user-item primary behaviors (relations) as our target behaviors, such as purchase behavior. Other behaviors serve as auxiliary behaviors, providing additional information for constructing better user behavior models. The overall system block diagram is shown in Figure 1.

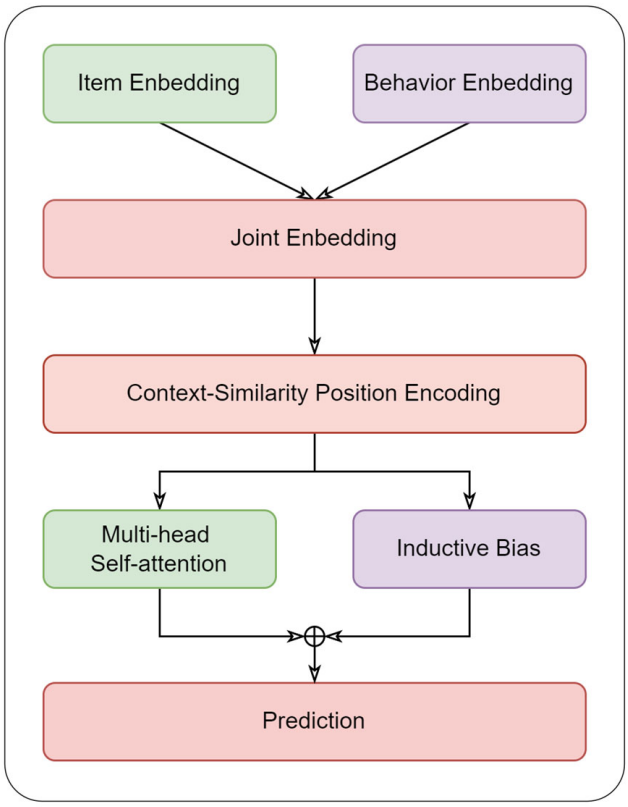

Figure 1. Architecture of our proposed CAMBSRec.

4. Methodology

In this section, we elaborate on the various components of our proposed method. The model consists of four primary components: an item and behavior embedding layer, position encoding, an inductive bias self-attention layer, and model training and prediction. The other key notations used in this paper are defined as follows in Table 1:

Table 1. Description of notations.

Notation	Description
x'_j, y'_j, z'_j	Item Embedding, Behavior Embedding, and Joint Embedding
h_j	Dissimilarity Gate Value
$g_{[q_j]}$	Context-Aware Positional Encoding
LFC, HFC	Low-Frequency Components and High-Frequency Components
$\hat{Y}$	Score Function
$\mathcal{L}$	Loss Function

4.1. Multi-Behavior Encoding

In the sequential recommendation task, each user has a series of interactions of different types with items on the platform over a certain period. Our goal is to predict which product a user is most likely to be interested in next based on the user's history of interaction sequences and recommend it to the user.

In real scenarios, the interaction behaviors between users and platforms are significantly diverse and hierarchical. Taking the e-commerce platform as an example, user behaviors usually include clicking, browsing the detail page, adding to the shopping cart, bookmarking, purchasing, and evaluating. Different behaviors reflect different intensities and stages of user interest: clicking: reflects initial interest, but may be exploratory (e.g., random browsing); favorite/add to cart: indicates that the user has a clear interest in the product, but may not buy for the time being due to price, inventory, and other factors; and purchase: directly reflects the user's final decision and is the core target behavior.

Therefore, in addition to core user behaviors (e.g., historical purchase records), we must also consider other interaction types (e.g., adding to carts, favoriting, etc.), which occur over time and reflect changes in user preferences. For example, at time t, user u adds item x_a to the cart, whereas at time $t + 1$, the user marks item x_b as favorited. These different behaviors provide crucial contextual information for the recommendation, helping us more accurately understand user interests.

To capture such contextual information, we first perform one-hot encoding on the item information of user interactions to obtain the vector x_j, and then input it into a fully connected layer to generate the behavior embedding x'_j:

$$x'_j = x_j W_x + b_x \tag{1}$$

where $W_x \in R^{I \times d}$ is the item weight matrix, I is the number of items, d is the item embedding dimension, and b_x is the item bias. Fully connected layers are chosen here over convolutional or recursive structures because of their flexibility in capturing global features of items and avoiding local bias. Similarly, the user interaction behavior sequence data are processed to obtain the behavior embedding y'_j:

$$y'_j = y_j W_y + b_y \tag{2}$$

where $W_y \in R^{B \times d}$ is the behavior type weight matrix, B is the number of behavior types, d is the behavior type embedding dimension, and b_y is the behavior bias. Here, the embedding of behavioral types y'_j is generated through a separate fully connected layer to ensure that the behavioral semantics are decoupled from the item features. Each item embedding is subsequently combined with its corresponding behavior embedding, and the result is fed into a fully connected layer to obtain the final joint item embedding z_j:

$$z_j = \left[\text{concat}\left(x'_j, y'_j\right) W_z\right] + b_z \tag{3}$$

where $W_z \in R^{2d \times d}$ is the contextual information weight matrix, d is the layer item dimension, and b_z is the bias term. Fusion of item and behavioral features through splicing and linear transformation, preserving the independent information of both and avoiding semantic confusion. The model architecture for this part is shown in Figure 2.

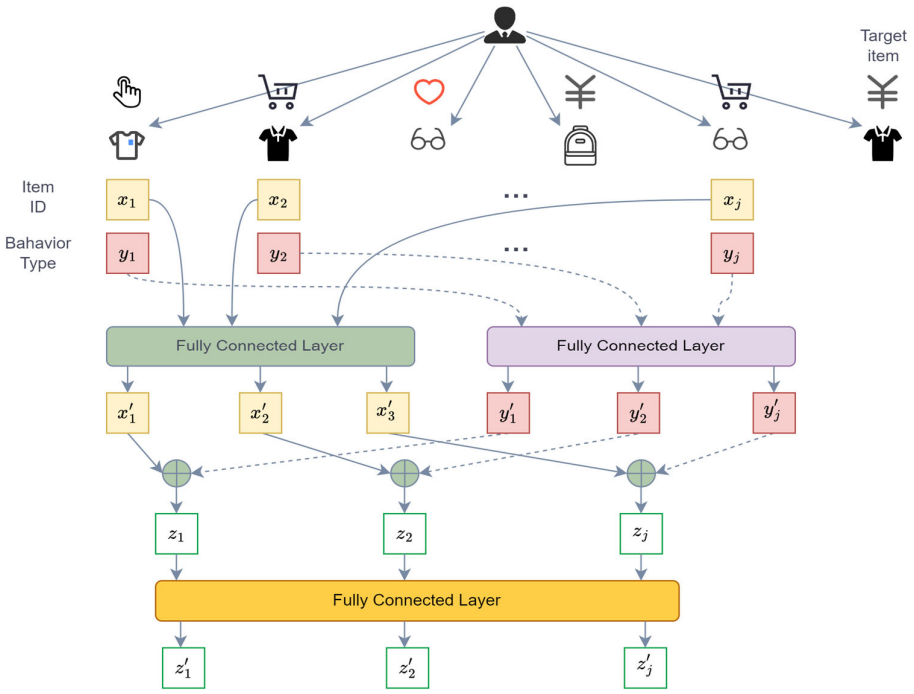

Figure 2. Item ID and behavior type are coded separately and fused to obtain the final joint embedding.

4.2. Position Encoding

We obtain the joint embedding vectors of items and behaviors, and next, we need to incorporate positional information into them. We do not simply add a learnable positional embedding to the vector of each position. This is because such location coding simply preserves the sequence information of the item, whereas what needs to be captured in sequence recommendation is complex location information at the session level and at the time span level. Instead, we need to calculate the dissimilarity between each context item and the target item. Specifically, a gate value that is dissimilar to the target item for each context item embedding x'_k is calculated:

$$h_j = 1 - \sigma\left(sim\left(o, x'_k\right)\right), k \in [1, n] \tag{4}$$

where n denotes the context length, sim is the similarity function, the cosine similarity is used here, which is suitable for measuring semantic relevance as it is sensitive to vector direction. Experiments have shown that dot product similarity is susceptible to dimensionality in long sequences, resulting in unstable values. σ is the sigmoid function, o is the embedding vector of the target item, and x'_k represents the contextual embedding at position k.

Next, we compute the positional value by summing the gating values between the current item and the target item:

$$q_j = \sum_{i=j}^{n} h_i \tag{5}$$

In Formula (5), if the item embeddings h_i at several consecutive positions are all similar to the target embedding, the corresponding h_i will be 0, and the value of q_j remains unchanged, indicating that the same positional embedding is assigned to similar contextual items. Conversely, each irrelevant contextual item causes the value of q_j to change, indicating the assignment of different positional embeddings. Figure 3 illustrates the gate value changes as an example; we use this method to distinguish the positional embeddings of different contextual items.

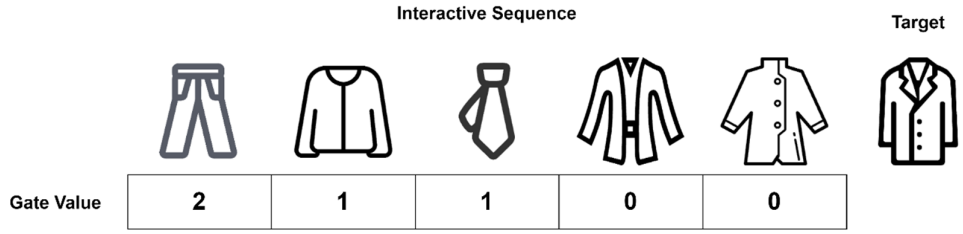

Figure 3. Using the coat example as the target item, the gate value for items with interactive sequences that are closer to it and similar to it does not increase, and the gate value only increases whenever a dissimilar item is encountered.

4.3. Inductive Bias Self-Attention Layer

In recent years, existing research [26] has revealed that the low-pass filtering property of the self-attention mechanism leads to over-smoothing of high-frequency signals (users' short-term interests), which prevents the model from capturing the dynamics of users' behaviors, and also lacks explicit modeling of the fine-grained models in the sequences, which leads to overfitting the training data and weak generalization ability.

To address this issue, we introduce inductive bias. We perform the discrete Fourier transform (DFT) on the joint item embedding z'_j that already contains positional encoding, decomposing it into two parts: low-frequency components (LFCs) and high-frequency components (HFCs). The LFC represents the low-frequency items of long-term interest, whereas the HFC represents the high-frequency items of short-term interest. The model architecture for the inductive bias self-attention layer is shown in Figure 4.

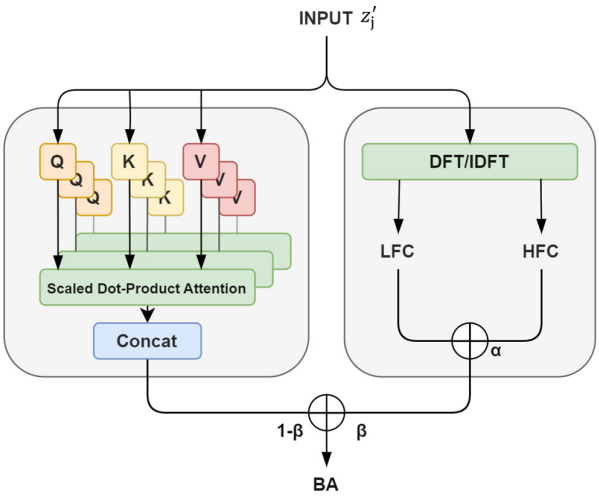

Figure 4. Inductive bias self-attention separates the HFC by the DFT and enhances that part to compensate for the low-pass filtering nature of the self-attention mechanism.

Then, we introduce a learnable parameter α for scaling to adjust the frequency scale of the high-frequency components, and the low-pass filtering characteristic of self-attention is counteracted by amplifying the amplitude of the HFC. Next, we set a hyperparameter β as the coefficient for balancing the inductive bias so that the model is dominated by self-attention and supplemented by a small number of frequency-domain features, avoiding the loss of temporal information caused by over-reliance on Fourier decomposition, and perform a linear weighted fusion of the multi-head self-attention and the Fourier features:

$$BA = \beta\left\{LFC\left[z'_j\right] + \alpha HFC\left[z'_j\right]\right\} + (1 - \beta)Az'_j \tag{6}$$

In Formula (8), α represents the part that enhances the high-frequency components and weakens the low-pass filtering characteristic of the self-attention mechanism. When $\beta = 0$ or 1, it indicates the exclusive use of self-attention or Fourier features. A represents the weight matrix of the multi-head self-attention. Finally, the final output embedding $Z = \left[z_1, z_2, \ldots, z_{|S_t^u|}\right]$ is obtained through the feed-forward neural network.

4.4. Model Training and Prediction

We take the interaction sequences of users and items in the dataset as the input sequences. Each element of the sequence is a tuple of (item, behavior). We shift the entire input sequence to the right and remove the last behavior interaction. Then, we fix the sequence length. For sequences longer or shorter than the specified length, we pad them with empty elements or truncate them to align the lengths. After that, we use the above-mentioned method to model and obtain the final output embedding Z. In addition, we separately model the removed last target interaction to obtain the target item embedding $o_{|S_{t+1}^u|}$. We calculate the score via the last items $z_{|S_t^u|}$ and $o_{|S_{t+1}^u|}$. The scoring function is as follows:

$$\hat{Y} = \sigma(z_{|S_t^u|} \cdot o_{|S_{t+1}^u|}) \tag{7}$$

Since this study models the multi-behavior sequences of users, the input sequences contain different behavior types, and different behavior types have different impacts on the target behavior. Therefore, we assign different weights to different behavior types, which in turn affects the final loss rate. In practical experiments, we use a weighted cross-entropy loss function to measure the prediction error. The specific formula is as follows:

$$\mathcal{L} = -\frac{1}{|S_t^u|} \sum_{(u,t) \in S_t^u} \left[\alpha_{b_{u,t}} log\left(\hat{Y}_{u,t}^{(+)}\right) + \varepsilon log\left(1 - (\hat{Y}_{u,t}^{(-)})\right)\right] \tag{8}$$

where $\hat{Y}_{u,t}^{(+)}$ is the output score of the positive sample, $\hat{Y}_{u,t}^{(-)}$ is the output score of the negative sample, S_t^u is the sequence of all interactions, $\alpha_{b_{u,t}}$ is the behavior weight, and ε is the negative sample sampling weight. Here, the weighted binary cross-loss function we use is mainly aimed at data problems with class imbalance, such as when the ratio of positive and negative samples is severely imbalanced. It can effectively improve the model's performance, but its performance is relatively weak when dealing with sparse data. We will further analyze this issue in Section 5.

5. Experiments

In this section, we aim to answer the following research questions:

Question 1: How does the CAMBSRec model perform compared with state-of-the-art multi-behavior and sequential models?
Question 2: What is the impact of selecting different weights for different behaviors?
Question 3: Is the specially designed context-aware positional encoding useful?
Question 4: How does the performance of the attention mechanisms with inductive bias perform?
Question 5: How does the CAMBSRec model perform in the face of sparse data?

5.1. Datasets and Experimental Setup

We evaluated the proposed CAMBSRec model on the following three publicly available datasets:

Tianchi: These data were originally sourced from the Tmall platform and were constructed for a research challenge focused on predicting users' repeat purchase behavior. It comprehensively records four types of operation logs similar to those in the Taobao user behavior system, namely, product purchase, adding items to the shopping cart, favoriting products, and page browsing behavior.

Beibei: This dataset is from an e-commerce platform in China and covers users' shopping behavior data on both traditional e-commerce platforms and social e-commerce platforms (such as sharing on WeChat). It includes user behaviors such as clicks, favorites, and purchases. It is characterized by high user activity, a short product life cycle, and a high proportion of mobile terminal usage.

MovieLens: This dataset integrates users' explicit rating records of movies. Its preprocessing procedure is similar to that of the Yelp dataset. On the basis of the numerical range of user ratings (from 1 to 5), three fine-grained interaction behaviors are defined: negative feedback (ratings from 1 to 2), neutral attitude (ratings from 3 to 4), and positive preference (ratings from 4 to 5). In such multi-behavior recommendation scenarios, users' explicit emotional expressions towards items (such as liking or disliking) are regarded as the core modeling objectives, whereas other implicit behaviors (such as clicks and searches) serve as auxiliary signals for feature learning.

We preprocessed the dataset based on user IDs, item IDs, and behavior types to ensure that both user IDs and item IDs are counted from 1 and used an inverse frequency weighting strategy for long-tail items to reduce the long-tail effect. A statistical summary of the datasets is shown in Table 2 below.

Table 2. Statistical summary of datasets.

Dataset	Users	Items	Interactions	Behavior Typers
Tianchi	6876	237,700	1,048,575	{Click, Favorite, Cart, Buy}
BeiBei	21,716	7977	3,338,068	{Click, Favorite, Cart, Buy}
MovieLens	67,787	8704	9,922,014	{Dislike, Neutral, Like}

5.2. Model Performance (RQ1)

To ensure the fairness and comparability of the experimental results, this study strictly adheres to the commonly used evaluation criteria in the academic community. The specific experimental design is as follows: The user behavior sequences are sorted according to the timestamps. For each user sequence, the last interaction is retained as the test set, and all the remaining historical interactions are used for training and validation. Moreover, for each positive sample in the test set, 99 randomly selected negative samples are paired with it to form an evaluation pair with a ratio of 1:100. Two evaluation metrics are used: Hit Rate at N (HR@N), which measures whether the recommended results contain the target item, and Normalized Discounted Cumulative Gain at N (NDCG@N), which evaluates the ranking quality of the recommended list. Note: The value ranges of both metrics are [0, 1], and a larger value indicates a better recommendation effect. The specific experimental results are shown in Table 3 below.

We compare the proposed method with various recommendation methods to demonstrate the advantage of our model. To verify the effectiveness of the proposed method, we investigated two types of recommendation baselines. The first group is about single-behavior sequential recommendation models:

SASRec [7]: This model uses multi-head self-attention to capture sequential patterns in users' histories and then applies the dot product to calculate item scores.

TiSASRec [18]: This model is an improvement on SASRec. It explicitly models the time intervals between user behaviors through a time-aware self-attention mechanism.

The second group is about multi-behavior recommendation models:

MB-GCN [8]: This method uses a graph structure to connect various interactions between users and items and makes recommendations by analyzing these interaction patterns.

MB-GMN [9]: This model optimizes the recommendation effect by constructing a user multi-behavior graphlet network and mining the association rules among different behaviors.

MBHT [10]: This is a transformer model improved on the basis of the hypergraph structure. Using the low-rank self-attention mechanism, it can capture the dependency relationships of both users' short-term and long-term behaviors simultaneously, improving the recommendation accuracy across time spans.

Table 3. Comparison of model performance and baseline models.

Method		Tianchi		Beibei		MovieLens	
		HR@10	**NDCG@10**	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
Single-Behavior Sequential	SASRec	0.658	0.484	0.423	0.382	0.815	0.573
	TiSASRec	0.646	0.496	0.493	0.393	0.786	0.543
Multi-Behavior Sequential	MB-GCN	0.836	0.603	0.633	0.384	0.750	0.489
	MB-GMN	0.848	0.633	0.648	0.386	0.744	0.469
	MBHT	0.834	0.654	0.623	0.358	0.829	0.615
	CAMBSRec (our)	0.869	0.690	0.654	0.391	0.841	0.623
Improve (%)		4.37%	5.42%	0.93%	1.24%	1.51%	1.18%

5.3. Impact of Behavior Weights (RQ2)

In the multi-behavior recommendation scenario, there are diverse interaction patterns between users and products, such as browsing, adding to carts, favoriting, and purchasing. Different behaviors have significantly different indicative values for the final recommendation target. If all the behavior data are regarded as signals of equal importance, the model will find it difficult to distinguish the potential intention intensities of the user behaviors. For example, a user's casual click on a product and their active addition of the product to the cart reflect weak and strong interest intensities, respectively. Therefore, during the modeling process, it is necessary to explicitly quantify the contribution weights of different behaviors to the target behavior (such as purchasing), instead of simply treating all interaction records equally. This differential modeling of behavior importance is the key to improving recommendation accuracy.

Here, we take the Tianchi dataset as an example to adjust the weight factors of different behavior types. Since the target behavior of this dataset is purchasing, we adjust the weight factor of the purchasing behavior from 1 to 0.3 for experimental comparison. The experimental results show that the best performance can be obtained when the purchasing weight is set to 0.7. Figure 5 shows that when the purchasing weight is greater than 0.7, the model performance decreases, and when the weight is less than 0.7, the impact is not significant. Therefore, to balance the weights of other behaviors, we set the purchasing weight to 0.7, and the remaining weights are evenly distributed among the other behaviors.

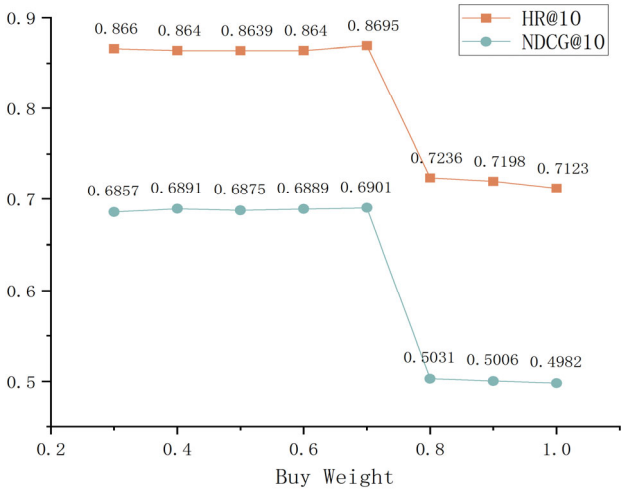

Figure 5. Impact of different purchase behavior weights.

5.4. Impact of Context-Aware Positional Encoding (RQ3)

We also conducted comparative experiments to determine whether our designed context-aware position encoding is effective. We carried out the experiment by replacing the position encoding part with traditional position encoding (using only one learnable embedding as the position encoding). We took the Tianchi dataset as an example. After replacing it with traditional position encoding and comparing it with our designed context-aware position encoding, both performance indicators decreased. The experimental results are presented in Figure 6. HR@10 decreased by 3.8%, from 0.8695 to 0.8363, and NDCG@10 decreased by 5.2%, from 0.6901 to 0.6542. This proves that our designed position encoding can improve the model.

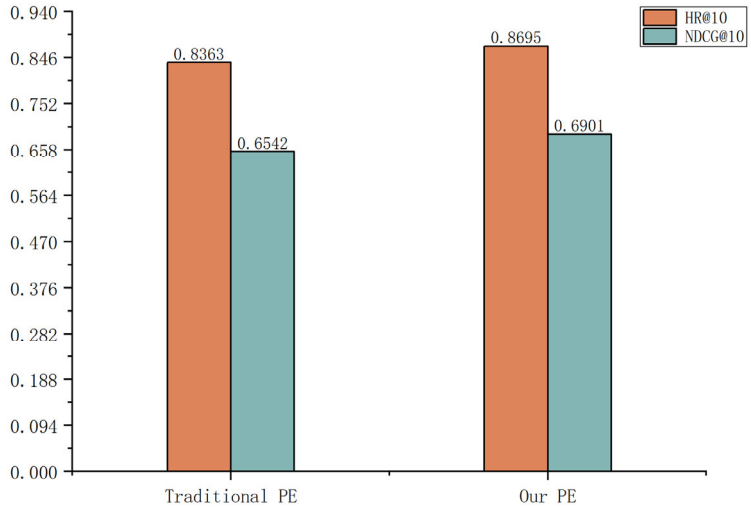

Figure 6. Impact of context-aware positional encoding.

5.5. Impact of Inductive Bias (RQ4)

To verify the impact of inductive bias on the performance of the overall model, we conducted a quantitative analysis on two important parameters, α and β.

α is a learnable parameter used to scale the HFC. Under the premise of keeping other parameters (such as β, embedding dimension, etc.) unchanged, we compare the model performance under different α values by taking different gradient values of α and quantifying their impact. As shown in Figure 7, we conducted experiments with the Tianchi dataset, taking α from 0.5 to 1.5 with an interval of 0.1.

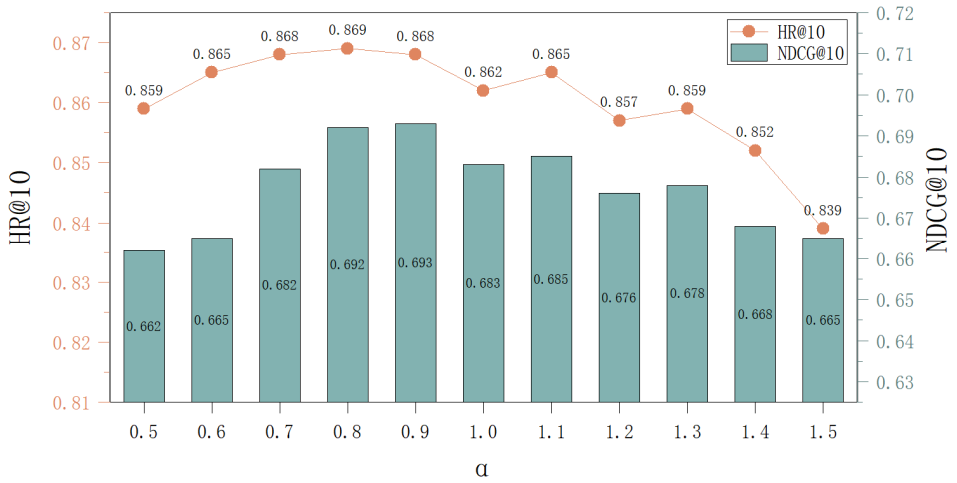

Figure 7. Impact of different values of α.

When α ranges from 0.5 to 0.9, HR@10 and NDCG@10 increase, reaching the highest values of 0.869 and 0.6936, respectively. This indicates that α enhances the high-frequency component (HFC), the model's ability to capture users' short-term interests (such as recent behavioral preferences) improves, and the recommendation accuracy and ranking quality improve simultaneously. When α ranges from 0.9 to 1.1, both indicators fluctuate. At this point, the enhancement of α's short-term interest has reached the "effective threshold"; HFC amplification does not introduce significant noise, and the model performance remains stable at a high level. When α ranges from 1.1 to 1.5, due to the excessive size of α, the HFC is overly enhanced, and the noise in the sequence (such as user misoperation and invalid browsing) is amplified, which interferes with the identification of effective interests and leads to a decline in recommendation performance.

We also set a hyperparameter β to represent the inductive bias weight of the overall Fourier features on the multi-head self-attention. We study the impact of β on the model by setting different values of β. The experimental results are shown in Figure 8. We set β to range from 0 to 1. When $\beta = 0$, the model is completely dependent on the self-attention output (without the participation of Fourier features), and both indicators are at their lowest. This indicates that the inherent "low-pass filtering" characteristic of self-attention (weakening short-term interest) leads to poor recommendation accuracy and ranking quality. When $\beta = 0.1$, for the Tianchi dataset, a small number of Fourier features are introduced to make up for the lack of short-term interest in self-attention, achieving the best balance of long-term and short-term interests and the optimal recommendation effect. When β gradually increases, the proportion of Fourier features becomes too high, squeezing the self-attention's ability to capture "long-term dependence (long-term interest)", resulting in impaired long-term interest modeling and a decline in overall recommendation quality. When $\beta = 1$, it is completely dependent on the Fourier feature output. Since it still contains LFC and does not fully determine short-term interest, the performance is still improved compared to the case where $\beta = 0$.

For this experimental result, we believe that for the whole model, it still works best when the multi-head self-attention is dominated by the inductive bias. It can be seen that when the weight share of β exceeds 0.5, the recommendation effect begins to decline significantly, i.e., the inductive bias accounts for more than the multi-head self-attention. So, injecting a small amount of inductive bias from the side can significantly improve the performance of the multi-head self-attention mechanism.

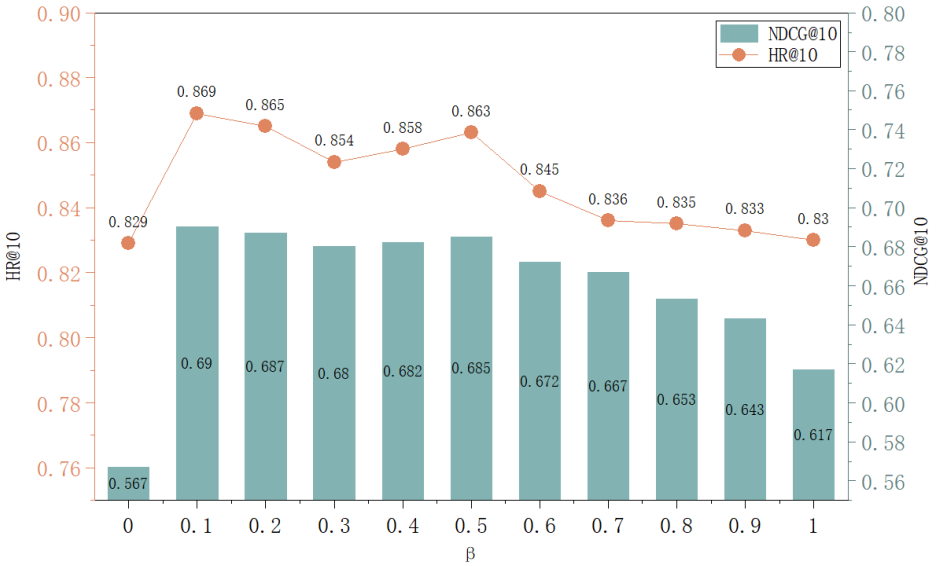

Figure 8. Impact of different values of β.

5.6. Performance in the Face of Sparse Data (RQ5)

As discussed in Section 4.4, handling rare interactions (e.g., interactions with long-tail items or from less active users) is crucial for comprehensive recommendation performance. This section evaluates our proposed model's sparse data performance.

We verify the model's performance by adopting a targeted negative sampling strategy (TNS). First, we focus on the long-tail products in the target behavior (purchase): products with a purchase frequency of less than 20% in the training set. Then, based on the embedding similarity of these goods, a candidate set similar to the positive sample goods is established. Finally, select the top 99 similar negative sample products from them. The specific experimental results are shown in Table 4 below.

Table 4. Performance comparison of long-tail product recommendations.

Model	HR@10	NDCG@10	Change
Baseline	0.869	0.690	—
Baseline + TNS	0.858	0.684	↓1.2%/↓0.9%

Although TNS theoretically improves sparse data performance, experiments reveal a slight decline in long-tail product recommendation effectiveness. This counterintuitive result stems from two factors. First, the weighted BCE already strongly supervises the target behavior (e.g., purchase weight = 0.7), focusing model attention effectively. Introducing TNS's similar product comparisons potentially distracts from this core objective. Second, long-tail items' inherent data sparsity means their similar-product negative samples likely contain noise. Forcing discrimination between these low-frequency combinations risks overfitting, reducing robustness compared to the baseline's simpler "purchase vs. random non-purchase" contrast.

6. Discussion

This section further discusses the experimental results, compares the proposed CAMB-SRec with existing studies, and analyzes its limitations to provide directions for future work.

6.1. Comparison with Existing Studies

The experimental results in Table 3 show that CAMBSRec achieves better performance than five baseline models on three datasets, which can be attributed to its targeted improvements in addressing the limitations of existing methods.

Compared with single-behavior sequential models (SASRec, TiSASRec), SASRec and TiSASRec focus only on a single type of user behavior (e.g., clicks) and ignore the auxiliary role of other behaviors (e.g., adding to carts, favoriting) in reflecting user interests. In contrast, CAMBSRec separately encodes item and behavior type embeddings and fuses them into joint embeddings, enabling it to capture fine-grained user intentions from multiple behaviors. For example, on the Beibei dataset, CAMBSRec's HR@10 (0.654) is higher than TiSASRec's (0.493), indicating that integrating multi-behavior information effectively enhances the model's ability to perceive user preferences.

Compared with multi-behavior models (MB-GCN, MB-GMN, MBHT), MB-GCN and MB-GMN rely on graph convolutional networks to model multi-behavior relationships but struggle to capture long-range sequential dependencies in user behavior sequences. MBHT introduces a hypergraph structure and transformer to address this issue, but it still suffers from the low-pass filtering effect of self-attention, which weakens the model's sensitivity to short-term interest changes. CAMBSRec's inductive bias self-attention layer, which uses the discrete Fourier transform to enhance high-frequency components, alleviates over-smoothing. For instance, on the Tianchi dataset, CAMBSRec's NDCG@10 (0.690) outperforms MBHT's (0.654), demonstrating that separating and enhancing high-frequency components effectively preserves dynamic user interests.

6.2. Complexity and Efficiency of Long Sequences

The traditional self-attention mechanism model captures dependencies by pairwise interaction of all tokens in the sequence, with a time complexity of $O(L^2 d)$ (L represents the sequence length and d represents the feature dimension) and a space complexity of $O(L^2 + Ld)$.

CAMBSRec separates high and low frequency components through the discrete Fourier transform (DFT) and takes advantage of the high efficiency of the Fast Fourier Transform (FFT) to reduce the time complexity to $O(LlogL + Ld)$, and the space complexity only needs to store frequency domain features of $O(L + Ld)$. When integrating Fourier features with the self-attention output, the weights of the Fourier features are controlled by the parameter β to avoid introducing additional high-complexity modules. The overall time complexity is still dominated by the Fourier module, that is $O(LlogL + Ld)$.

Therefore, compared with the traditional self-attention model, the complexity of ($L \geq 1000$) in long sequence scenarios is reduced by 1–2 orders of magnitude, and the efficiency is higher when processing long sequence data.

6.3. Limitations and Future Work

Despite the encouraging results, this study has several limitations:

Computational Complexity in Ultra-Long Sequences: The model integrates the discrete Fourier transform (DFT) and inductive bias self-attention to balance long- and short-term interests, but this introduces non-negligible computational overhead. While this is superior to pure self-attention for long sequences, it still struggles with ultra-long sequences ($L \geq 5000$) in real-world scenarios (e.g., users' cumulative behaviors over years), leading to increased training/inference latency and higher memory consumption.

The context-similarity positional encoding relies on item similarity to determine positional information, which performs well in datasets with rich item attributes (e.g., e-commerce products with clear categories). However, in domains where item semantics

are ambiguous, the similarity calculation may lose discriminative power, reducing the effectiveness of positional information modeling.

The experimental datasets (Tianchi, Beibei, MovieLens) primarily focus on typical interaction behaviors, but real-world recommendation scenarios involve more complex behavioral patterns, such as social interactions (e.g., sharing, commenting, @mentions) or multi-modal behaviors (e.g., text reviews combined with image views). CAMBSRec's current framework does not explicitly model these diverse behavioral types, limiting its adaptability to broader application scenarios.

In summary, CAMBSRec provides a framework for multi-behavior sequential recommendation, but its practicality and adaptability can be further enhanced by addressing computational efficiency, sparse data robustness, and scenario generalization. Future work will focus on these aspects to advance its applicability in real-world recommendation systems.

7. Conclusions

This study proposes a context-aware multi-behavior sequential recommendation model, CAMBSRec. This model captures the features in users' multi-behavior interaction data and separately embeds and encodes items and behavior types. Then, it incorporates context-aware positional encoding to record the sequential position information. After that, it uses the multi-head self-attention mechanism to capture the dependencies between items. Moreover, inductive bias is introduced to alleviate the low-pass filtering characteristic of the attention mechanism itself, preventing the model from oversmoothing. Finally, the model makes predictions through a weighted cross-entropy loss function. Experiments were conducted on three multi-behavior datasets, and the results show that all the indicators of this model are better than those of the other recommendation models.

Author Contributions: B.Z.: Methodology: Designed and implemented a context-aware multi-behavior sequential recommendation model for personalized recommendations; Formal Analysis: Evaluated model performance and analyzed results; Writing—Original Draft: Ensured the accuracy and completeness of the manuscript. Y.L.: Supervision: Oversaw the research project and provided strategic direction; Writing—Review and Editing: Reviewed and edited the manuscript to maintain high-quality research standards. M.Z.: Supervision: Oversaw the research project and provided strategic direction; Writing—Review and Editing: Reviewed and edited the manuscript to maintain high-quality research standards. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Basic Scientific Research Projects of Liaoning Provincial Universities grant number LJ212410152012 and the APC was funded by Dalian Polytechnic University.

Data Availability Statement: All data used in this study were derived from the following resources available in the public domain: Tianchi dataset at https://www.kaggle.com/datasets/yingzhou1313/tianchi; BeiBei dataset at https://github.com/Sunscreen123/Beibei-dataset; MovieLens 1M dataset at https://grouplens.org/datasets/movielens/1m.

Conflicts of Interest: The authors declare that they have no competing interests or financial conflicts to disclose.

References

1. Loukili, M.; Messaoudi, F.; El Ghazi, M. Machine learning based recommender system for e-commerce. *IAES Int. J. Artif. Intell. (IJ-AI)* **2023**, *12*, 1803–1811. [CrossRef]
2. Sharma, K.; Lee, Y.-C.; Nambi, S.; Salian, A.; Shah, S.; Kim, S.-W.; Kumar, S. A survey of graph neural networks for social recommender systems. *ACM Comput. Surv.* **2024**, *56*, 1–34. [CrossRef]
3. Chen, X.; Yao, L.; McAuley, J.; Zhou, G.; Wang, X. Deep reinforcement learning in recommender systems: A survey and new perspectives. *Knowl.-Based Syst.* **2023**, *264*, 110335. [CrossRef]

4. Liu, H.; Zheng, C.; Li, D.; Shen, X.; Lin, K.; Wang, J.; Zhang, Z.; Zhang, Z.; Xiong, N.N. EDMF: Efficient deep matrix factorization with review feature learning for industrial recommender system. *IEEE Trans. Ind. Inform.* **2021**, *18*, 4361–4371. [CrossRef]
5. Papadakis, H.; Papagrigoriou, A.; Panagiotakis, C.; Kosmas, E.; Fragopoulou, P. Collaborative filtering recommender systems taxonomy. *Knowl. Inf. Syst.* **2022**, *64*, 35–74. [CrossRef]
6. Xia, L.; Huang, C.; Zhang, C. Self-supervised hypergraph transformer for recommender systems. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2100–2109.
7. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, Singapore, 17–20 November 2018; pp. 197–206.
8. Jin, B.; Gao, C.; He, X.; Jin, D.; Li, Y. Multi-behavior recommendation with graph convolutional networks. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, 25–30 July 2020; pp. 659–668.
9. Xia, L.; Xu, Y.; Huang, C.; Dai, P.; Bo, L. Graph meta network for multi-behavior recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; pp. 757–766.
10. Yang, Y.; Huang, C.; Xia, L.; Liang, Y.; Yu, Y.; Li, C. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2263–2274.
11. Yuan, E.; Guo, W.; He, Z.; Guo, H.; Liu, C.; Tang, R. Multi-behavior sequential transformer recommender. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, 11–15 July 2022; pp. 1642–1652.
12. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
13. Afsar, M.M.; Crump, T.; Far, B. Reinforcement learning based recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–38. [CrossRef]
14. Xia, H.; Luo, Y.; Liu, Y. Attention neural collaboration filtering based on GRU for recommender systems. *Complex Intell. Syst.* **2021**, *7*, 1367–1379. [CrossRef]
15. Hou, Y.E.; Gu, W.; Yang, K.; Dang, L. Deep Reinforcement Learning Recommendation System based on GRU and Attention Mechanism. *Eng. Lett.* **2023**, *31*, 695.
16. Roy, D.; Dutta, M. A systematic review and research perspective on recommender systems. *J. Big Data* **2022**, *9*, 59. [CrossRef]
17. An, H.-W.; Moon, N. Design of recommendation system for tourist spot using sentiment analysis based on CNN-LSTM. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *13*, 1653–1663. [CrossRef]
18. Li, J.; Wang, Y.; McAuley, J. Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 322–330.
19. Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J.; Jin, D.; He, X.; et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst.* **2023**, *1*, 1–51. [CrossRef]
20. Zhao, N.; Long, Z.; Wang, J.; Zhao, Z.-D. AGRE: A knowledge graph recommendation algorithm based on multiple paths embeddings RNN encoder. *Knowl.-Based Syst.* **2022**, *259*, 110078. [CrossRef]
21. Pradhan, T.; Kumar, P.; Pal, S. CLAVER: An integrated framework of convolutional layer, bidirectional LSTM with attention mechanism based scholarly venue recommendation. *Inf. Sci.* **2021**, *559*, 212–235. [CrossRef]
22. Shaw, P.; Uszkoreit, J.; Vaswani, A. Self-attention with relative position representations. *arXiv* **2018**, arXiv:1803.02155.
23. Qiu, R.; Huang, Z.; Chen, T.; Yin, H. Exploiting positional information for session-based recommendation. *ACM Trans. Inf. Syst.* **2021**, *40*, 1–24. [CrossRef]
24. Mu, Z.; Zhuang, Y.; Tang, S. Position-aware compositional embeddings for compressed recommendation systems. *Neurocomputing* **2024**, *592*, 127677. [CrossRef]
25. Golovneva, O.; Wang, T.; Weston, J.; Sukhbaatar, S. Contextual Position Encoding: Learning to Count What's Important. *arXiv* **2024**, arXiv:2405.18719.
26. Shin, Y.; Choi, J.; Wi, H.; Park, N. An attentive inductive bias for sequential recommendation beyond the self-attention. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 8984–8992. [CrossRef]