MTSTREC: MULTIMODAL TIME-ALIGNED SHARED TOKEN RECOMMENDER

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential recommendation in e-commerce leverages users' anonymous browsing histories to offer personalized product suggestions without relying on personal information. While item ID-based sequential recommendations are commonly used, they often fail to fully capture the diverse factors influencing user preferences, such as textual descriptions, visual content, and pricing. These factors represent distinct modalities in recommender systems. Existing multimodal sequential recommendation models typically employ either early or late fusion of different modalities, overlooking the alignment of corresponding positions in time of product sequences that represent users' browsing preferences. To address these limitations, this paper proposes a unified framework for multimodal fusion in recommender systems, introducing the Multimodal Time-aligned Shared Token Recommender (MTSTRec). MTSTRec leverages a transformer-based architecture that incorporates a single time-aligned shared token for each product, allowing for efficient cross-modality fusion that also aligns in time. This approach not only preserves the distinct contributions of each modality but also aligns them to better capture user preferences. Additionally, the model extracts rich features from text, images, and other product data, offering a more comprehensive representation of user decision-making in e-commerce. Extensive experiments demonstrate that MTSTRec achieves state-of-the-art performance across multiple sequential recommendation benchmarks, significantly improving upon existing multimodal fusion strategies.

1 INTRODUCTION

In e-commerce and online platforms, Sequential Recommendation Systems (SRS) have become crucial for providing personalized product suggestions based on users' browsing history. SRS has evolved from simple Markov Chains (Shani et al., 2013) to more sophisticated neural networks. Early approaches like GRU4Rec (Hidasi et al., 2016) employed recurrent neural networks (RNNs) to model user behavior sequences. While RNNs can capture short-term dependencies, they struggle with long-term dependencies due to vanishing gradients. Transformer-based models have been proposed to address the above issue, such as SASRec (Kang & McAuley, 2018) and BERT4Rec (Sun et al., 2019). They leverage self-attention mechanisms to capture dependencies across the entire sequence, thereby improving both efficiency and scalability. However, these models still primarily focus on single-modal data and overlook the rich multimodal information available in real-world applications, which could provide deeper insights into user preferences.

Multimodal recommendation systems combine diverse data types, such as images and texts. Image-based methods use pre-trained convolutional neural networks (CNNs) (O'Shea & Nash, 2015) like ResNet (He et al., 2015), to capture the visual features, while text-based methods use models like BERT (Devlin et al., 2019) for product descriptions and reviews. CLIP aligns text and images through a multimodal approach, enabling zero-shot transfer for various vision-language tasks (Radford et al., 2021). Recently, large language models (LLMs) have excelled in extracting hidden textual information, enriching item representations (Geng et al., 2023; Lyu et al., 2024). Rather than full-scale LLM training, we focus on leveraging LLMs to enrich item descriptions by extracting hidden textual features. Additionally, while many recommendation models rely on image classification or recognition to predict purchase intent, we argue that textual data alone is often sufficient

for identifying what a product is. Instead, images should capture product aesthetics and style, which are especially important on platforms that sell the same product in various patterns or designs.

When it comes to multimodal fusion, designing a unified model that effectively integrates different modalities presents significant challenges. Different modalities and their specialized formats learn at different speeds and patterns. This makes it challenging to combine them effectively into a single system. In the context of recommendation systems, the common modalities are typically images and text, which have fundamentally different input representations. Early fusion techniques are often used to unify these features, where modalities are combined before entering the recommendation model (He & McAuley, 2015; Liu et al., 2019). However, this approach fails to account for the significant differences in input representations and neural network architectures across modalities. In contrast, late fusion methods process each modality separately, addressing modality-specific challenges, but overlook the fact that at each time step, different modalities correspond to the same product (Liang et al., 2023).

To address both the modality processing and fusion issues, we propose **MTSTRec (Multimodal Time-aligned Shared Token Recommender)**, a novel framework for multimodal feature integration and cross-modal interaction using time-aligned shared tokens. MTSTRec consists of two main components: the Feature Extractor and the Multimodal Transformer. The feature extractor takes the browsing history sequence, where each item includes a product ID, image, text, and price. We use different extractors for each modality. For text, we enrich the data through LLMs with task-specific prompts to extract implicit consumer preferences. For images, we focus on style rather than classification, using Gram metrics to capture visual patterns that influence purchasing behavior. These inputs are then projected into separate feature embeddings for each modality. In the proposed multimodal transformer, a self-attention encoder is first applied to model the information for each feature. During fusion, we adopt a mid-fusion strategy, where modalities are processed independently and then combined in the intermediate stage. Our proposed Time-aligned Shared Token fusion module (TST) learns cross-modal interactions by aligning features from different modalities at each time step of the product interaction sequence. This ensurs efficient feature sharing across modalities while maintaining the chronological order and consistency of product interactions.

Our contributions can be summarized as follows:

- We propose a unified framework for the multimodal recommendation that seamlessly accommodates various input data types, effectively integrating diverse information such as ID, text, image, and other modalities. This flexible approach ensures that different types of data are processed and combined in a cohesive manner, enhancing the system's adaptability across different tasks.

- We introduce a novel **TST** module that leverages shared tokens to learn cross-modal interactions at each time step of the sequence. This design ensures that information from different modalities is aligned and fused in a time-consistent manner. By maintaining the chronological structure of the sequential data, the TST module effectively captures the evolving relationships between user interactions and product features. This innovative module not only enhances the performance of multimodal recommendation systems but also offers a flexible and scalable that can be applied to various tasks beyond recommendation, making it suitable for a wide range of multimodal applications.

- **MTSTRec** outperforms state-of-the-art methods on three real-world e-commerce datasets, setting a new benchmark in multimodal recommendation systems. Our ablation studies not only highlight the contribution of individual features but also reveal deeper insights into the distinct roles of various modalities across diverse e-commerce environments, offering a valuable understanding for future advancements in the field.

2 Related Work

2.1 Sequential Recommendation Systems

SRS aims to predict the next item a user will interact with based on their browsing history, providing personalized recommendations. Traditional Markov Chain model (Shani et al., 2013) employed simple probabilistic methods but struggled to capture complex user behavior patterns. GRU4Rec

(Hidasi et al., 2016) introduced RNNs to improve sequence modeling by capturing temporal dependencies. Transformer-based models like SASRec (Kang & McAuley, 2018) enhance this approach by using self-attention mechanisms (Vaswani et al., 2017) and causal masking, preserving temporal order and efficiently capturing short and long-term dependencies. BERT4Rec (Sun et al., 2019) extends this further by adopting a bidirectional Transformer model and using a CLOZE task for training. Unlike SASRec's causal masking, BERT4Rec allows the model to attend to both past and future items in the sequence, capturing richer contextual information. However, these models remain focused primarily on single-modal data, overlooking the potential benefits of multimodal information.

2.2 Multimodal Recommendation Systems

Multimodal recommendation systems are essential for integrating information from multiple modalities to create more accurate and comprehensive predictions. These systems generally operate through two main stages: raw feature extraction and feature fusion (Liu et al., 2024).

Different modalities require tailored methods to capture their unique attributes in the raw feature extraction stage. For instance, image-based features are often extracted using CNNs (O'Shea & Nash, 2015) or, more recently, Vision Transformers (ViT) (Dosovitskiy et al., 2021), which excel at processing visual data. Textual features often rely on pre-trained language models (Devlin et al., 2019), capturing semantic meanings from product textual information. In addition, advances in LLMs (Dubey et al., 2024; OpenAI et al., 2024) have significantly enhanced text-based feature extraction. LLM-Rec (Lyu et al., 2024) utilizes LLM and prompts to generate richer contextual representations, enhancing recommendation quality. Other LLM-based approaches have also emerged, further refining text comprehension to optimize the use of textual features in recommendation systems (Zhao et al., 2024; Li et al., 2023; Wu et al., 2024).

After raw feature extraction, the system proceeds to the feature fusion stage, where these multimodal data are combined and processed, it can be categorized into three main approaches (Zhou et al., 2023): (i) Early fusion involves merging the features from different modalities at the initial stages of the model. For example, VBPR (He & McAuley, 2015) integrates visual features into a matrix factorization approach. However, early fusion may miss important temporal relationships and modality-specific behaviors by merging features too early.(ii) Mid-fusion delays modality combination for more refined processing, as seen in MM-Rec (Wu et al., 2022), which uses a cross-modal attention mechanism to combine textual and visual information effectively. However, these methods often fail to capture temporal relationships, which are crucial in sequential recommendations. (iii) Late fusion keeps modalities separate until the final stage, as in MMMLP (Liang et al., 2023), the final outputs from three different modalities are concatenated before making the prediction. While preserving modality-specific features, it fails to capture early interactions between modalities and sequences, leading to suboptimal performance with complex sequential patterns.

To address these limitations, we aim to design a fusion method that can temporally align and integrate different modalities, facilitating cross-modal interactions at each time step. This novel fusion ensures that the chronological order of product interactions, ensuring temporal consistency while allowing for efficient multimodal feature sharing across the entire sequence.

3 Proposed Method: MTSTRec

This section presents the MTSTRec framework and describes how we effectively extract multimodal features from a consumer-centric perspective, including introducing a style and an LLM-based prompt text extractor module to better simulate the consumer's purchasing context. This framework can potentially be extended to integrate additional modalities for even more comprehensive consumer behavior modeling. Furthermore, we present how we utilize TST module to synchronize the features of the same product from multimodal encoders across different layers. An overview of the MTSTRec is illustrated in Figure 1.

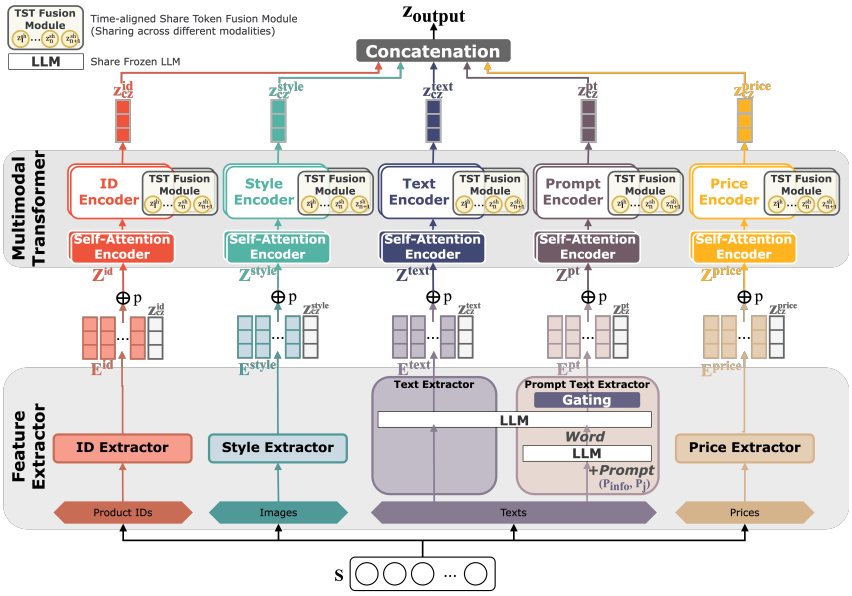

Figure 1: MTSTRec consists of feature extractors and multimodal transformers. The feature extractors take inputs as a browsing history sequence $\boldsymbol{S}$, where each item consists of a product ID, image, text, and price, transforming them into the corresponding feature embeddings, along with a CLOZE embedding $\boldsymbol{z}_{\text{cz}}$ for each modality. The multimodal transformers begin by applying a self-attention encoder to independently process the information for each feature. When fusion begins, we utilize the TST module across different modalities at each time step and across multiple layers, ensuring both modality-specific and cross-modal information are captured effectively.

3.1 Preliminaries

Let the set of items be defined as $\boldsymbol{I} = \{\boldsymbol{i}_1, \boldsymbol{i}_2, \ldots, \boldsymbol{i}_k, \ldots, \boldsymbol{i}_{|I|}\}$, where each item $\boldsymbol{i}_k \in \boldsymbol{I}$ consists of four essential elements: $\boldsymbol{b}_k$, which represents the product ID; $\boldsymbol{v}_k$, which represents the image of the item; $\boldsymbol{t}_k$, which contains the textual information; and $\boldsymbol{c}_k$, which represents the price of the item. Therefore, each item $\boldsymbol{i}_k$ can be expressed as a tuple $\boldsymbol{i}_k = (\boldsymbol{b}_k, \boldsymbol{v}_k, \boldsymbol{t}_k, \boldsymbol{c}_k)$, capturing the multiple modalities that describe it. For each user, their browsing history is represented as $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n]$, where each $\boldsymbol{s}_i \in \boldsymbol{I}$ corresponds to an item the user has interacted with, and n is the length of the session. The goal of SRS is to predict the next item $\boldsymbol{s}_{n+1}$ that a user will interact with, based on the user's interaction history $\boldsymbol{S}$.

3.2 Feature Extractor Module

3.2.1 ID Extractor

The ID extractor converts raw ID data into meaningful embeddings. We construct an item ID embedding matrix $\boldsymbol{M}^{\text{id}} \in \mathbb{R}^{d \times |I|}$, where d represents the dimension of the ID embedding. The input ID embedding matrix for each session $\boldsymbol{E}^{\text{id}} \in \mathbb{R}^{d \times n}$ is then retrieved such that $\boldsymbol{E}^{\text{id}}_i = \boldsymbol{M}^{\text{id}}_{\boldsymbol{s}_i}$, corresponding to the item $\boldsymbol{s}_i$ in the user's interaction sequence, the i th in the row of matrix. A constant zero vector $\mathbf{0}$ is used as the embedding for any padding items within the sequence, typically added when sequences are shorter than the maximum length. This embedding mechanism enables the model to capture item relationships, improving the accuracy of recommendations by understanding item identities within sequences. The processing method for the price extractor is similar; please refer to the Appendix A.1 for details.

3.2.2 STYLE EXTRACTOR

We are inspired by the Neural Style Transfer (NST) algorithm (Gatys et al., 2015; 2016) to extract style features. Instead of transferring style, we focus solely on extracting it. Using the VGG-19 (Simonyan & Zisserman, 2015) model, we pass the image through the network to obtain feature maps, from which we compute Gram matrices $\boldsymbol{G}$. These matrices capture image texture and color patterns and are invariant to spatial transformations, ensuring similar matrices for images with similar styles. We then generate the input style embedding matrix for each session, $\boldsymbol{E}^{\text{style}} \in \mathbb{R}^{d \times n}$. Further implementation details are in the Appendix A.2.

3.2.3 TEXT EXTRACTOR

The text extractor focuses on extracting features from product titles and descriptions. We use Llama 3.1 (Dubey et al., 2024) as our chosen backbone for generating text embeddings that capture the key attributes of the product (see Appendix A.3 for details). Each product is assigned a text embedding, forming the item text embedding matrix $\boldsymbol{M}^{\text{text}} \in \mathbb{R}^{d \times |I|}$, which is generated by feeding the product's textual information into an LLM text encoder to obtain initial text embeddings, followed by a one-layer projection to adjust the dimensionality. The input text embedding matrix for each session $\boldsymbol{E}^{\text{text}} \in \mathbb{R}^{d \times n}$ is then retrieved, such that $\boldsymbol{E}_i^{\text{text}} = \boldsymbol{M}_{\boldsymbol{s}_i}^{\text{text}}$ corresponding to the item $\boldsymbol{s}_i$ in the user's interaction sequence.

3.2.4 PROMPT-TEXT EXTRACTOR

In the second part of text processing, we explore how LLM can be prompted to generate additional textual information to improve recommendation performance. Inspired by Lyu et al. (2024), we employ two prompt strategies: basic prompt and recommendation prompt. Additionally, we utilize five variations tailored to the task, including three for basic tasks, along with two designed for recommendation tasks (see Appendix A.4 for more details). The LLM input is divided into a system and a task-specific prompt. The system prompt provides a brief description of the e-commerce context P_{info} to ensure that the LLM understands the task's background, and a task-specific prompt P^j, selected from five predefined prompt variations described earlier. For each product i, its textual information, such as the title and description, is fed into the user prompt. The LLM then generates a response $\boldsymbol{P}_i^j$ for each product i, using each prompt variant j. These responses are converted into prompt embeddings for further processing (see Appendix A.4 for more details). The process is formalized as follows:

$$\hat{\boldsymbol{P}}_i^j = Encoder_{LLM}(\boldsymbol{P}_i^j). \tag{1}$$

Gating Network: Inspired by (Yu et al., 2024), we introduce a gating network G to manage the varying importance of each prompt variant $\hat{\boldsymbol{P}}^j$. This network controls the information flow to the final prediction layer. The weights of the prompt-text embeddings $\boldsymbol{w}$ are calculated as follows:

$$\boldsymbol{w} = G(\|_j \hat{\boldsymbol{P}}^j) := softmax(W[\|_j \hat{\boldsymbol{P}}^j] + b), \forall j \in \{1, \ldots, 5\}. \tag{2}$$

The item prompt embedding matrix $\boldsymbol{M}^{\text{pt}} \in \mathbb{R}^{d \times |I|}$ is then computed by:

$$\boldsymbol{M}^{\text{pt}} = L_p \left(\sum_{j=1}^{5} \boldsymbol{w}_j \cdot \hat{\boldsymbol{P}}^j \right), \tag{3}$$

where L_p is a linear projection layer applied to the gated prompt embeddings. Further details are provided in the Appendix A.5. The input prompt embedding matrix for each session $\boldsymbol{E}^{\text{pt}} \in \mathbb{R}^{d \times n}$ is then retrieved, such that $\boldsymbol{E}_i^{\text{pt}} = \boldsymbol{M}_{\boldsymbol{s}_i}^{\text{pt}}$ corresponding to the item $\boldsymbol{s}_i$ in the user's interaction sequence.

3.3 MULTIMODAL TRANSFORMER WITH TIME-ALIGNED SHARED TOKEN FUSION

The multimodal transformer is built based on the Transformer encoder architecture proposed by Vaswani et al. (2017), where each feature has its own encoder to begin with. The

user sequence is processed through each feature extractor, producing five distinct embeddings: $\boldsymbol{E}^{\text{id}}, \boldsymbol{E}^{\text{style}}, \boldsymbol{E}^{\text{text}}, \boldsymbol{E}^{\text{pt}}, \boldsymbol{E}^{\text{price}}$, all in $\mathbb{R}^{d \times n}$. Positional embeddings are added to each embedding, and $\boldsymbol{z}_{\text{cz}}$ is appended to the sequence to indicate the item to be predicted. The recommendation is made based on the final representation of $\boldsymbol{z}_{\text{cz}}$. Using the ID embedding matrix $\boldsymbol{E}^{\text{id}}$ as an example, the final input to the ID encoder is:

$$\boldsymbol{Z}^{\text{id}} = [\boldsymbol{E}_1^{\text{id}}, \boldsymbol{E}_2^{\text{id}}, ..., \boldsymbol{E}_N^{\text{id}}, \boldsymbol{z}_{\text{cz}}^{\text{id}}] \oplus \mathbf{p}, \tag{4}$$

where $\mathbf{p} \in \mathbb{R}^{d \times (n+1)}$ is the learnable positional embedding, and $\oplus$ denotes element-wise addition. Thus, we obtain the inputs to the multimodal transformer: $\boldsymbol{Z}^{\text{id}}, \boldsymbol{Z}^{\text{style}}, \boldsymbol{Z}^{\text{text}}, \boldsymbol{Z}^{\text{pt}}, \boldsymbol{Z}^{\text{price}}$, which are passed through separate encoders.

3.3.1 Self-Attention Encoder

In the next step, input embeddings from each modality $\boldsymbol{Z}$ are processed independently through a self-attention encoder Vaswani et al. (2017). This approach enables each modality to learn its unique features, and the resulting representations are then prepared for cross-modal fusion using the TST module. Further details are provided in Appendix A.6.

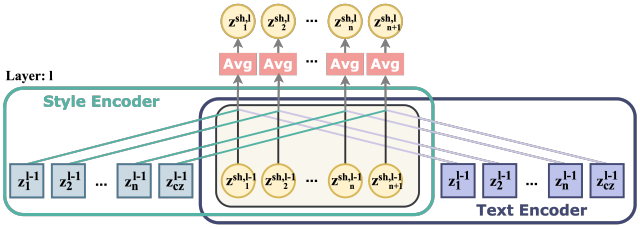

Figure 2: The TST module is shared across different modalities in a manner that aligns each item in the sequence. For example, in the fusion of style and text embeddings, each TST embedding $\boldsymbol{z}^{\text{sh}}$ is updated element-wise by averaging the corresponding tokens from the style and text modalities at the same time step in the sequence, ensuring cross-modal information exchange that aligns in time.

3.3.2 Time-aligned Shared Token Fusion with Sequential Multimodal Integration

We now introduce our TST fusion module, as illustrated in Figure 2, which facilitates multimodal information sharing by aligning item tokens across different modalities. Inspired by the attention bottleneck mechanism (Nagrani et al., 2021), which focuses on efficiently transferring information between modalities, we re-design the mid-fusion module such that the embeddings for each item are aligned in time. This design introduces a strong prior that the embeddings from each modality belong to the same item in the sequence, ensuring that representations of the same item from different modalities are effectively fused.

Each sequence contains $n + 1$ tokens, includding the CLOZE embedding $\boldsymbol{z}_{\text{cz}}$ (Sun et al., 2019). The number of time-aligned shared tokens ($\boldsymbol{z}^{\text{sh}}$) matches the sequence length, with each shared token $\boldsymbol{z}_t^{\text{sh}}$ at time step t corresponding to modality-specific tokens $\boldsymbol{z}_t^{\text{mod}}$, where mod $\in \{\text{id}, \text{style}, \text{text}, \text{pt}, \text{price}\}$. These shared tokens enable cross-modal interaction for the same item at each time step. For each modality, modality-specific tokens $\boldsymbol{Z}^{\text{mod}} = [\boldsymbol{z}_1^{\text{mod}}, \ldots, \boldsymbol{z}_n^{\text{mod}}, \boldsymbol{z}_{cz}^{\text{mod}}]$ and the time-aligned shared tokens $\boldsymbol{Z}^{\text{sh}} = [\boldsymbol{z}_1^{\text{sh}}, \ldots, \boldsymbol{z}_n^{\text{sh}}, \boldsymbol{z}_{n+1}^{\text{sh}}]$ are concatenated and fed into the self-attention encoder Att (Vaswani et al., 2017). It processes input tokens from each modality independently through L transformer layers, where the transformation at layer l is defined as:

$$\boldsymbol{Z}^{\text{mod},l} = Att(\boldsymbol{Z}^{\text{mod},l-1} \| \boldsymbol{Z}^{\text{sh},l-1}). \tag{5}$$

At each layer, modality-specific tokens are updated into $\boldsymbol{Z}^{\text{mod},l}$ for the next layer l. These tokens capture the modality-specific features, which will be further enhanced by interaction with the TST module. Similarly, for time-aligned shared tokens, the tokens from all modalities are averaged at

the end of each layer to form the next layer's shared tokens $\boldsymbol{Z}^{\text{sh},l}$, facilitating further fusion across subsequent layers. This process is defined as:

$$\boldsymbol{Z}^{\text{sh},l} = \frac{1}{\#\text{mod}} \sum_{\text{mod}} Att(\boldsymbol{Z}^{\text{mod},l-1} \| \boldsymbol{Z}^{\text{sh},l-1}), \forall \text{mod} \in \{\text{id}, \text{style}, \text{text}, \text{pt}, \text{price}\}. \tag{6}$$

In the first layer of the fusion encoder, the modality-specific sequences do not take inputs from the time-aligned shared tokens yet, as these tokens are initially unlearned. However, each shared token $\boldsymbol{z}^{\text{sh}}$ can take inputs from its corresponding modality tokens, learning multimodal information. From the second layer onward, modality-specific tokens and their corresponding shared tokens attend to each other, enabling cross-modal learning and information exchange.

After fusion, the learned representations are passed to the prediction layer. The sequence token $\boldsymbol{z}_{\text{cz}}$ plays a critical role in capturing the information needed for predicting the next item in the user's interaction sequence. By integrating information from the modality-specific tokens and time-aligned shared tokens, the model achieves a cohesive multimodal representation, improving the accuracy of the recommendation task. The final modality representations are normalized to ensure stability, then combined and used for prediction and loss computation. The $\boldsymbol{z}_{\text{cz}}$ tokens are concatenated to form the final recommendation representation for each sequence:

$$\boldsymbol{z}_{\text{output}} = \boldsymbol{z}_{\text{cz}}^{\text{id}} \| \boldsymbol{z}_{\text{cz}}^{\text{style}} \| \boldsymbol{z}_{\text{cz}}^{\text{text}} \| \boldsymbol{z}_{\text{cz}}^{\text{pt}} \| \boldsymbol{z}_{\text{cz}}^{\text{price}}. \tag{7}$$

This final representation $\boldsymbol{z}_{\text{output}}$ encapsulates the key information from each modality, allowing the model to make accurate predictions for the next item in the user interaction sequence.

3.4 Loss Function

For each sequence, the model computes the cosine similarity between the final recommendation representation $\boldsymbol{z}_{\text{output}}$ and both the ground truth and negative samples. The embeddings for both the ground truth and negative samples $\boldsymbol{y}$, are generated by concatenating features $\boldsymbol{E}$ extracted for each item through the same feature extractors. Let $\cos(\boldsymbol{z}_{\text{output}}, \boldsymbol{y}^k)$ represent the cosine similarity between $\boldsymbol{z}_{\text{output}}$ and the k-th sample, where $k \in K_{gt}$ for positive samples (ground truth) and $k \in K_n$ for negative samples. These cosine similarities are then used to compute the binary cross-entropy (BCE) loss for the sequence as follows:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{|K_{gt} \cup K_n|} \left(\sum_{k \in K_{gt}} \log\left(\cos(\boldsymbol{z}_{\text{output}}, \boldsymbol{y}^k)\right) + \sum_{k \in K_n} \log\left(1 - \cos(\boldsymbol{z}_{\text{output}}, \boldsymbol{y}^k)\right) \right). \tag{8}$$

This formula drives the model to maximize cosine similarity with ground truth and minimize it with negative samples, improving recommendation accuracy.

4 Experiments

4.1 Experimental Settings

Dataset. Our experiments utilize three datasets: two proprietary datasets from AviviD Innovative Multimedia—*Fresh-Food E-commerce* and *House-Hold E-commerce*, which will be released after the study, and a public dataset from *H&M*. The proprietary datasets capture user interactions, including pageviews and purchases, while the public dataset focuses on purchase records in the trousers category from *H&M*. Detailed information is provided in Appendix A.7.

Dataset Splitting. Each session in the proprietary datasets consists of historical clicks and a final purchase order. The data is split chronologically into 75% for training, 12.5% for validation, and 12.5% for testing based on the purchase orders. For the *H&M (Trousers)* dataset, which contains only purchase actions, items are sorted by purchase time, and those bought on the last day are used as the answer set, ensuring consistency across all datasets (Meng et al., 2020). The model predicts

items in purchase orders as a multi-label recommendation task, where each sequence may have multiple correct answers (purchases).

Evaluation Metrics. We evaluate all models using three popular top-k ranking metrics: Normalized Discounted Cumulative Gain (NDCG@k), Hit Rate (HR@k), and Mean Reciprocal Rank (MRR@k), with k set to 5 and 10. NDCG@k remains unchanged as it inherently supports multiple selections. For HR@k and MRR@k, we adjust the calculations to account for the multi-answer format, ensuring a fair evaluation. For more detail, please refer to Appendix A.8.

Benchmark Models. We compare our MTSTRec model with well-known SRS baselines, which are grouped into two categories. The first category includes general models that rely solely on historical interaction data, using item IDs as the primary feature, such as GRU4Rec (Hidasi et al., 2016; Hidasi & Karatzoglou, 2017), SASRec (Kang & McAuley, 2018), and BERT4Rec (Sun et al., 2019). The second category extends beyond item IDs by incorporating additional modalities, including text, image, and price data. This group includes models like MMMLP (Liang et al., 2023), as well as the enhanced versions of SASRec$^+$ and BERT4Rec$^+$, which concatenate image and text features as in our model. These modifications allow them to act as early fusion models, combining all features upfront to ensure a fair comparison of multimodal inputs. For more details on the benchmark models, refer to Appendix A.9.

Table 1: Performance Comparison of Benchmark Models and MTSTRec on Two Datasets

Dataset	Model	NDCG@5	NDCG@10	HR@5	HR@10	MRR@5	MRR@10
Fresh-Food E-commerce	GRU4Rec	0.7800	0.7782	0.7345	0.7652	0.6912	0.6957
	SASRec	0.8015	0.7999	0.7505	0.7836	0.7099	0.7143
	Bert4Rec	0.8076	0.8094	0.7658	0.8010	0.7146	0.7193
	SASRec$^+$	0.8512	0.8446	0.8013	0.8250	0.7729	0.7760
	Bert4Rec$^+$	0.8441	0.8394	0.7962	0.8218	0.7641	0.7675
	MMMLP	0.8276	0.8239	0.7845	0.8129	0.7450	0.7487
	MTSTRec	**0.8800**	**0.8765**	**0.8407**	**0.8651**	**0.8086**	**0.8118**
House-Hold E-commerce	GRU4Rec	0.7432	0.7652	0.7451	0.7806	0.6857	0.6914
	SASRec	0.7563	0.7736	0.7578	0.7960	0.7024	0.7076
	Bert4Rec	0.7596	0.7761	0.7603	0.7968	0.7068	0.7116
	SASRec$^+$	0.8150	0.8258	0.8144	0.8410	0.7688	0.7723
	Bert4Rec$^+$	0.7969	0.8110	0.7959	0.8287	0.7477	0.7521
	MMMLP	0.8335	0.8425	0.8303	0.8525	0.7930	0.7959
	MTSTRec	**0.8942**	**0.9086**	**0.9067**	**0.9358**	**0.8568**	**0.8607**
H&M (Trousers)	GRU4Rec	0.1302	0.1520	0.1571	0.2132	0.1027	0.1181
	SASRec	0.1520	0.1759	0.1809	0.2489	0.1258	0.1348
	Bert4Rec	0.1468	0.1692	0.1738	0.2378	0.1223	0.1306
	SASRec$^+$	0.1605	0.1811	0.1828	0.2415	0.1371	0.1448
	Bert4Rec$^+$	0.1633	0.1848	0.1878	0.2492	0.1387	0.1466
	MMMLP	0.1754	0.1923	0.1971	0.2451	0.1502	0.1565
	MTSTRec	**0.2307**	**0.2797**	**0.3139**	**0.4481**	**0.1871**	**0.2049**

4.2 Performance Comparison

To assess the generalizability of MTSTRec, we conducted experiments on three datasets and compared the results with baseline models. The results, summarized in Table 1, reveal critical insights into the effectiveness of different models.

MTSTRec surpasses all baselines, leveraging its TST module to outperform both early and late fusion approaches. In the *Fresh-Food E-commerce* dataset, MTSTRec achieves the highest NDCG@5 score of 0.8800, and in the *House-Hold E-commerce* dataset, it achieves 0.8942, outperforming SASRec$^+$ by approximately 7% in both cases. Notably, in the *H&M (Trousers)* dataset, MTSTRec achieves an NDCG@5 of 0.2307, further solidifying its dominance across all metrics. In comparison, self-attention-based models, SASRec and BERT4Rec, have consistently demonstrated superior performance over the RNN-based model, GRU4Rec. Besides, SASRec$^+$ and BERT4Rec$^+$, which integrate text and image features, further boost performance. In the *Fresh-Food E-commerce* dataset, SASRec$^+$ achieves an NDCG@5 of 0.8512, and in the *House-Hold E-commerce* dataset, it reaches 0.8150, which demonstrates the importance of incorporating multimodal data to better capture user

preferences and provide more accurate recommendations. However, these models still fell short of MTSTRec's performance.

In summary, MTSTRec consistently outperforms both early fusion models SASRec$^+$ and BERT4Rec$^+$, and late fusion models MMMLP, across all datasets. The model's effective integration of multimodal information via TST module ensures that feature interactions are fully captured, leading to state-of-the-art performance, particularly in complex datasets like *H&M (Trousers)*, with performance gains of approximately 31.5%–45.5% over competing models in terms of NDCG.

4.3 Ablation Study of Modalities

In this section, we conduct an ablation study on *Fresh-Food E-commerce* and *House-Hold E-commerce* datasets to evaluate the contribution of modalities, including item ID, text, prompt text, images, and price. By removing each modality, we can assess their impact on model performance.

As shown in Table 2, removing the item ID modality has the most significant impact on both datasets, with NDCG@5 dropping sharply from 0.8800 to 0.7582 on the *Fresh-Food E-commerce* dataset. This highlights the critical role of product identity in differentiating items. The absence of item IDs also increased training time as the model struggled to manage without clear product identifiers. Text information proves to be another important modality. In the *House-Hold E-commerce* dataset, removing both text and prompt text leads to a noticeable performance drop, with NDCG@5 decreasing from 0.8942 to 0.8191. This highlights the significance of detailed product descriptions in capturing user preferences. Even when standard text is removed, the prompt text still provides valuable contextual information, enabling the model to maintain reasonable accuracy, as indicated by a smaller drop to 0.8488 NDCG@5 compared to removing both text and prompt text.

In summary, item ID and text modalities are essential for accurate recommendations, with ID playing a particularly crucial role in guiding predictions. Prompt text can compensate for missing standard text. Other modalities are less critical for accurately predicting user preferences in these domains.

Table 2: The Impact of Removing Different Modality Modules across E-commerce Platforms

Ablation Study	*Fresh-Food E-commerce*			*House-Hold E-commerce*		
	NDCG@5	HR@5	MRR@5	NDCG@5	HR@5	MRR@5
MTSTRec	**0.8800**(±0.0023)	**0.8407**(±0.0031)	**0.8086**(±0.0023)	**0.8942**(±0.0035)	**0.9067**(±0.0024)	**0.8568**(±0.0041)
w/o ID	0.7582(±0.0049)	0.7506(±0.0049)	0.6459(±0.0062)	0.7913 (±0.0244)	0.8337(±0.0191)	0.7184(±0.0288)
w/o Text & Prompt	0.8574(±0.0013)	0.8102(±0.0011)	0.7813(±0.0015)	0.8191(±0.0047)	0.8142(±0.0041)	0.7761(±0.0062)
w/o Text	0.8729(±0.0016)	0.8308(±0.0024)	0.7998(±0.0022)	0.8488(±0.0095)	0.8553(±0.0139)	0.8051(±0.0087)
w/o Prompt Text	0.8749(±0.0046)	0.8331(±0.0056)	0.8030(±0.0054)	0.8770(±0.0035)	0.8895(±0.0079)	0.8366(±0.0031)
w/o Style	0.8784(±0.0020)	0.8391(±0.0030)	0.8068(±0.0026)	0.8932(±0.0093)	0.9061(±0.0082)	0.8524(±0.0111)
w/o Price	0.8791(±0.0025)	0.8404(±0.0031)	0.8077(±0.0030)	0.8941(±0.0048)	0.9066(±0.0040)	0.8553(±0.0063)

4.4 Ablation Study of Fusion Module

In this ablation study, we evaluated the impact of different fusion strategies on the *Fresh-Food E-commerce* dataset, comparing TST (used in MTSTRec) with two alternative setups: one resembling late fusion by removing the TST and fusion encoder, and another resembling early fusion by removing the entire multimodal transformer. In the late fusion alternative, features are processed independently and concatenated only at the final step, and in the early fusion alternative, all features are concatenated immediately after feature extraction and passed through a single encoder.

As shown in Table 3, TST outperforms both alternatives. TST achieves an NDCG@5 of 0.8800, compared to 0.8621 for early fusion. In the early fusion, all features are processed together from the start, which can lead to the dilution of unique information from each modality, as the model is not able to treat them distinctly. This reduces the model's ability to fully leverage the strengths of each feature. On the other hand, late fusion limits the interaction between different features until the end of the sequence. This restriction hinders the model's ability to capture cross-modal dependencies throughout the sequence, leading to a decrease in performance. Specifically, the NDCG@5 score dropped from 0.8800 to 0.8211, a decrease of 0.0589.

In conclusion, TST's ability to facilitate cross-modal interaction at each time step leads to better results, whereas early and late fusion approaches fall short due to inefficient handling of feature interactions.

Table 3: The Impact of Removing Different Modules on Fresh-Food E-Commerce Results

Fusion Method	NDCG@5	NDCG@10	HR@5	HR@10	MRR@5	MRR@10
MTSTRec	**0.8800**	**0.8765**	**0.8407**	**0.8651**	**0.8086**	**0.8118**
w/o TST & Fusion Encoder (Late Fusion)	0.8211	0.8301	0.7912	0.8378	0.7271	0.7333
w/o Multimodal Encoder (Early Fusion)	0.8621	0.8590	0.8206	0.8483	0.7862	0.7899

4.5 Comparison of Shared Tokens Configurations

We evaluate the different shared token configurations on the *Fresh-Food E-commerce* dataset, focusing on three key setups: the proposed TST (1:1), TST with multiple shared tokens per time step (TST (1:2) and TST (1:4)), and fusion bottlenecks. Each configuration represents a unique mid-fusion approach to feature sharing during the sequence learning process. Further details are provided in the Appendix A.11.

As shown in Table 4, the proposed TST (1:1) configuration delivers the best performance, with an NDCG@5 of 0.8800. This setup ensures that only one shared token per time step facilitates information transfer, which helps the model focus on product-specific features without introducing unnecessary noise. TST (1:2) and TST (1:4) show a slight drop in performance, with NDCG@5 ranging from 0.8769 to 0.8795. The increase in shared tokens per time step introduces more information exchange but also adds redundancy and potential noise, slightly affecting model efficiency. Bottlenecks configurations perform worse than TST, with bottlenecks (all:5) Nagrani et al. (2021) achieving an NDCG@5 of 0.8737 and bottlenecks (all:21) achieving an NDCG@5 of 0.8754, which is 0.5% lower than TST (1:1). Despite allowing broader information sharing, these setups lack the precise, time-aligned interaction that enhances TST's performance.

The TST (1:1) configuration outperforms other setups, including multi-token and fusion bottleneck approaches, likely due to its ability to maintain precise, time-aligned interactions between product features. This configuration effectively balances information sharing and efficiency, making it the most optimal choice for multimodal sequential recommendation in our scenario.

Table 4: The Impact of Time-aligned Shared Tokens on Fresh-Food E-commerce Dataset Results

Setting	NDCG@5	NDCG@10	HR@5	HR@10	MRR@5	MRR@10
TST (1:1) (Ours)	**0.8800**($\pm$0.0023)	**0.8765**($\pm$0.0021)	**0.8407**($\pm$0.0031)	**0.8651**($\pm$0.0029)	**0.8086**($\pm$0.0023)	**0.8118**($\pm$0.0023)
TST (1:2)	0.8795($\pm$0.0022)	0.8758($\pm$0.0020)	0.8406($\pm$0.0025)	0.8641($\pm$0.0023)	0.8079($\pm$0.0026)	0.8110($\pm$0.0026)
TST (1:4)	0.8769($\pm$0.0041)	0.8737($\pm$0.0038)	0.8384($\pm$0.0039)	0.8631($\pm$0.0036)	0.8044($\pm$0.0051)	0.8077($\pm$0.0050)
Bottlenecks (all:5)	0.8737($\pm$0.0028)	0.8695($\pm$0.0024)	0.8323($\pm$0.0030)	0.8560($\pm$0.0027)	0.8012($\pm$0.0029)	0.8044($\pm$0.0028)
Bottlenecks (all:21)	0.8754($\pm$0.0036)	0.8716($\pm$0.0035)	0.8346($\pm$0.0043)	0.8587($\pm$0.0040)	0.8037($\pm$0.0039)	0.8069($\pm$0.0038)

5 Conclusion

This paper introduces MTSTRec, a novel Multimodal Time-aligned Shared Token Recommender to fuse and transmit essential information across different modalities. Our approach allows for precise integration of multimodal features such as product IDs, images, text, and prices while maintaining the unique contributions of each modality. Through extensive experimentation on real-world e-commerce datasets, we demonstrated that MTSTRec significantly outperforms state-of-the-art baselines across various evaluation metrics. Our ablation studies revealed the critical role of each feature type in improving recommendation accuracy, particularly the importance of item identity and textual descriptions in different e-commerce scenarios. Moreover, we showed that the proposed TST fusion method consistently surpasses both early and late fusion strategies by enabling cross-modal interaction that aligns in time throughout the sequence. In summary, MTSTRec represents a significant advancement in multimodal sequential recommendation, offering a flexible and efficient framework that can be adapted to various e-commerce applications. Future work will focus on extending the model to other domains and exploring additional multimodal features for even more personalized recommendations.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. URL https://arxiv.org/abs/1508.06576.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5), 2023. URL https://arxiv.org/abs/2203.13366.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback, 2015. URL https://arxiv.org/abs/1510.01784.

Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *CoRR*, abs/1706.03847, 2017. URL http://arxiv.org/abs/1706.03847.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016. URL https://arxiv.org/abs/1511.06939.

Chun-Kai Huang, Yi-Hsien Hsieh, Ta-Jung Chien, Li-Cheng Chien, Shao-Hua Sun, Tung-Hung Su, Jia-Horng Kao, and Che Lin. Scalable numerical embeddings for multivariate time series: Enhancing healthcare data representation learning, 2024. URL https://arxiv.org/abs/2405.16557.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 197–206. IEEE, 2018.

Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights, 2023. URL https://arxiv.org/abs/2305.11700.

Jiahao Liang, Xiangyu Zhao, Muyang Li, Zijian Zhang, Wanyu Wang, Haochen Liu, and Zitao Liu. Mmmlp: Multi-modal multilayer perceptron for sequential recommendations. In *WWW*, pp. 1109–1117, 2023. URL https://doi.org/10.1145/3543507.3583378.

Fan Liu, Zhiyong Cheng, Changchang Sun, Yinglong Wang, Liqiang Nie, and Mohan Kankanhalli. User diverse preference modeling by multimodal attentive metric learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19. ACM, October 2019. doi: 10.1145/3343031.3350953. URL http://dx.doi.org/10.1145/3343031.3350953.

Qidong Liu, Jiaxi Hu, Yutian Xiao, Xiangyu Zhao, Jingtong Gao, Wanyu Wang, Qing Li, and Jiliang Tang. Multimodal recommender systems: A survey, 2024. URL https://arxiv.org/abs/2302.03883.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models, 2024. URL https://arxiv.org/abs/2307.15780.

Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pp. 681–686, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. doi: 10.1145/3383313.3418479. URL https://doi.org/10.1145/3383313.3418479.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=KJ5h-yfUHa.

OpenAI. Openai embedding api [text-embedding-3-large]. Technical report, OpenAI, 2024a. URL https://openai.com/index/new-embedding-models-and-api-updates/.

OpenAI. Openai gpt-4o api [gpt-4o-mini]. Technical report, OpenAI, 2024b. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI, Josh Achiam, Steven Adler, and Sandhini Agarwal et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL https://arxiv.org/abs/1511.08458.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Guy Shani, Ronen I. Brafman, and David Heckerman. An mdp-based recommender system. *CoRR*, abs/1301.0600, 2013. URL http://arxiv.org/abs/1301.0600.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR)*, pp. 1–14, 2015.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, 2019. URL https://arxiv.org/abs/1904.06690.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:6000–6010, 2017.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Mm-rec: Multimodal news recommendation, 2022. URL https://arxiv.org/abs/2104.07407.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. A survey on large language models for recommendation, 2024. URL https://arxiv.org/abs/2305.19860.

Chu-Chun Yu, Ming-Yi Hong, Chiok-Yew Ho, and Che Lin. Push4rec: Temporal and contextual trend-aware transformer push notification recommender. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6625–6629, 2024. doi: 10.1109/ICASSP48485.2024.10447336.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. Recommender systems in the era of large language models (llms), 2024. URL https://arxiv.org/abs/2307.02046.

Hongyu Zhou, Xin Zhou, Zhiwei Zeng, Lingzi Zhang, and Zhiqi Shen. A comprehensive survey on multimodal recommender systems: Taxonomy, evaluation, and future directions, 2023. URL https://arxiv.org/abs/2302.04473.

A APPENDIX

A.1 PRICE EXTRACTOR

We represent all product prices in the list using a unified normalization approach for consistency. Following the Scalable Numerical Embedding (SCANE) method (Huang et al., 2024), an item price embedding matrix $\boldsymbol{M}^{\text{price}} \in \mathbb{R}^{d \times |I|}$ is constructed, and an input price embedding matrix $\boldsymbol{E}^{\text{price}} \in \mathbb{R}^{d \times n}$, captures the product prices within a user's interaction history. These embeddings are then scaled by their corresponding price value $\boldsymbol{P}_{\boldsymbol{s}_i}$, such that $\boldsymbol{E}_i^{\text{price}} = \boldsymbol{M}_{\boldsymbol{s}_i}^{\text{price}} \times \boldsymbol{P}_{\boldsymbol{s}_i}$ for each item $\boldsymbol{s}_i$ in the sequence, enhancing the model's ability to understand pricing patterns in recommendations.

Furthermore, We experimented with a variant of the SCANE method, where the entire price embedding matrix M^{price} was replaced such that each element of $M_i^{\text{price}} j$ was set to 1, effectively disabling embedding learning. In this case, the price embeddings were simply the expanded price values P_i themselves. In our scenario, this variant yielded better performance, as the embedding effectively represented the expanded price without the complexity of learning additional embedding weights.

A.2 IMPLEMENTATION DETAILS OF THE STYLE EXTRACTOR

In our experiments, we extract style features from the first two layers of VGG-19 (Simonyan & Zisserman, 2015) to capture the relevant style information. Each of these layers produces 64 feature maps, resulting in Gram matrices of size 64×64 (Gatys et al., 2015; 2016).

The Gram matrix for each layer is calculated as follows:

Let $\boldsymbol{F}^l \in \mathbb{R}^{Q_l \times R_l}$ represent the feature map from layer l, where Q_l is the number of feature maps (or channels) in layer l, and R_l is the product of the height and width of the feature map (i.e., the spatial dimensions of the feature map). The Gram matrix $\boldsymbol{G}^l \in \mathbb{R}^{Q_l \times Q_l}$ for layer l is calculated as the inner product between the vectorized feature maps F_{ik}^l and F_{jk}^l at spatial positions k, across all feature maps i and j. Mathematically, this can be expressed as:

$$\boldsymbol{G}_{ij}^l = \sum_{k=1}^{R_l} \boldsymbol{F}_{ik}^l \boldsymbol{F}_{jk}^l, \tag{9}$$

where G_{ij}^l represents the element of the Gram matrix that captures the correlation between feature map i and feature map j in layer l, and the summation over k accounts for all R_l spatial positions in the feature map. After calculating the Gram matrices, we apply max-pooling to compress these matrices, which reduces the computational complexity while retaining the essential style information. This step compresses the Gram matrices to $2 \times 16 \times 16$ style embeddings.

Once each image undergoes the process of Gram matrix computation and subsequent max-pooling, the compressed and flattened Gram matrices for all images are concatenated across layers to form the item style embedding matrix $\boldsymbol{M}^{\text{style}} \in \mathbb{R}^{d \times |I|}$. This matrix encapsulates the style features of all items in the dataset. The input style embedding matrix for each session $\boldsymbol{E}^{\text{style}} \in \mathbb{R}^{d \times n}$ is then retrieved, such that $\boldsymbol{E}_i^{\text{style}} = \boldsymbol{M}_{\boldsymbol{s}_i}^{\text{style}}$, corresponding to the item $\boldsymbol{s}_i$ in the user's interaction sequence.

A.3 COMPARISON OF LANGUAGE MODELS FOR TEXT EMBEDDING

We evaluated the pre-trained language model, BERT (Devlin et al., 2019), and large language models, text-embedding-3-large (OpenAI, 2024a), Llama 3 (Dubey et al., 2024), and Llama 3.1 (Dubey et al., 2024). These comparisons were conducted with the MTSTRec model, focusing solely on product ID and text. The handling of product IDs is discussed in Section 3.2.1. For the text portion, we used product titles and descriptions, converting them into embeddings via the respective Language Models, followed by a linear reduction layer W_R that reduces the embedding dimension to d, as shown in Equation **??**.

The results are presented in Table 5. Llama 3.1 achieves the best performance across all evaluation metrics (NDCG, HR, and MRR). Thus, we choose Llama 3.1 as the backbone for generating text embeddings in our main experiments.

Table 5: Comparison of Language Models for Text Embedding in MTSTRec

Feature	NDCG@5	HR@5	MRR@5
ID	0.8554	0.8074	0.7792
ID + $\text{text}_{\text{BERT}}$	0.8585	0.8108	0.7829
ID + $\text{text}_{\text{openai}}$	0.8594	0.8119	0.7838
ID + $\text{text}_{\text{Llama3}}$	0.8732	0.8329	0.8000
ID + $\text{text}_{\text{Llama3.1}}$	**0.8754**	**0.8340**	**0.8037**

A.4 Implementation Details and Results of Prompt-Text Feature

As outlined in Section 3.2.3, we employed various prompting strategies to generate rich textual information, with detailed examples of these prompts provided in Table 6. For Basic Prompt, we utilize three variations: P_{para}, P_{tags}, and P_{guess}. P_{para} prompts the LLM to rephrase the original product description while retaining the same information. P_{tags} aims for a concise summary using tags, guiding the LLM to extract key details. Lastly, P_{guess} prompts the LLM to predict what other items the user might purchase based on the product's title and description. Recommendation Prompt extends the Basic Prompt by introducing a recommendation-oriented task. We define two variations: P^{rec}_{para} and P^{rec}_{tags}.

In our experiments, we explored text generation with Llama 3.1 (Dubey et al., 2024) and GPT-4o-mini (OpenAI et al., 2024; OpenAI, 2024b) and compared their effectiveness in creating useful text embeddings. Each prompt was carefully designed with specific settings to match its intended function.

For instance:

- P_{para}: This prompt was used to paraphrase the product title and description, configured with a temperature of 0.7, a maximum token limit of 256, and top_p = 1.
- P_{tags}: For this prompt, which focuses on summarizing product information using tags, we set the temperature to 0.5, a maximum token limit of 128, and top_p = 1.
- P_{guess}: This prompt aims to infer potential additional products the user might purchase based on the current product. It was configured with a temperature of 1, a maximum token limit of 512, and top_p = 1.
- P^{rec}_{para}: This recommendation-focused paraphrase prompt had a temperature of 1, a maximum token limit of 384, and top_p = 1.
- P^{rec}_{tags}: Designed to summarize product information for recommendations, this prompt used a temperature of 1, a maximum token limit of 128, and top_p = 1.

The text generated by Llama 3.1 (Dubey et al., 2024) or GPT-4o-mini (OpenAI et al., 2024; OpenAI, 2024b) was uniformly converted into embeddings using Llama 3.1 (shown in Appendix A.3) to maintain consistency across experiments. This approach allowed for reliable comparisons, focusing on how different prompts and their corresponding embeddings affected recommendation accuracy in MTSTRec. Each experiment incorporated prompt embeddings alongside the product ID to measure performance.

We also assessed the influence of different LLMs—Llama 3.1 and GPT-4o-mini on overall model performance by comparing how each model's generated text, once converted into embeddings, impacted recommendation results. The detailed outcomes are summarized in the following Table 7.

Based on the results, we observe that incorporating prompt embeddings into the model helps improve performance compared to using only the product ID. While the results for prompt embeddings are not as high as directly using text embeddings, this might suggest that the original product titles and descriptions, when processed by the LLM, sufficiently capture the characteristics of the items. However, our findings demonstrate that prompt embeddings positively influence recommendation outcomes. Additionally, the prompt embeddings generated by both Llama 3.1 and GPT-4o-mini show comparable performance across the five strategies. Llama 3.1 has a slight edge on average,

Table 6: All prompts in prompt-text feature extraction module

Prompt Type	System and User Prompt
P_{para}	**System**: *"FFE is an e-commerce website that sells fresh, healthy, high-quality food products without unnecessary additives. You will be provided with the product title and description sold on this e-commerce website. Your task is to paraphrase them."* **User**: *"Title:* `product title`*, Description:* `product description`*."*
P_{tags}	**System**: *"FFE is an e-commerce website that sells fresh, healthy, high-quality food products without unnecessary additives. You will be provided with the product title and description sold on this e-commerce website. Your task is to summarize this product using tags."* **User**: *"Title:* `product title`*, Description:* `product description`*."*
P_{guess}	**System**: *"FFE is an e-commerce website that sells fresh, healthy, high-quality food products without unnecessary additives. You will be provided with the product title and description sold on this e-commerce website. Your task is to infer what other products on the site a consumer might be interested in if they purchase this product."* **User**: *"Title:* `product title`*, Description:* `product description`*."*
P_{para}^{rec}	**System**: *"FFE is an e-commerce website that sells fresh, healthy, high-quality food products without unnecessary additives. Your task is to tell me what else I should say if I want to recommend this product to someone."* **User**: *"Title:* `product title`*, Description:* `product description`*."*
P_{tags}^{rec}	**System**: *"FFE is an e-commerce website that sells fresh, healthy, high-quality food products without unnecessary additives. Your task is to tell me which tags should be used if I want to recommend this product to someone."* **User**: *"Title:* `product title`*, Description:* `product description`*."*

Table 7: Comparison of Prompt Strategies and LLMs (Llama 3.1 vs GPT-4o-mini) in MTSTRec

Feature	Prompt Strategy	Llama 3.1	GPT-4o-mini
ID + prompt	P_{para}	0.8727	0.8747
	P_{tags}	0.8727	0.8723
	P_{guess}	0.8718	0.8702
	P_{para}^{rec}	0.8697	0.8703
	P_{tags}^{rec}	0.8725	0.8716
	Average	0.8719	0.8718

which is why we chose to primarily use Llama 3.1 for generating our prompt-text features in subsequent experiments.

A.5 Gating Weights and Performance of Prompt-Text Feature

As explained in Section 3.2.4, a gating network is applied to the five prompt strategies to learn their relative importance. The gating weights (Table 9) are derived from the validation results of the ID + Text + Prompt features. In addition, we compare the performance of ID + Text versus ID + Text + Prompt, demonstrating improved recommendation accuracy with the inclusion of prompt embeddings, regardless of the LLM used, as shown in Table 8.

Table 8: Performance Comparison of ID + Text and ID + Text + Prompt in MTSTRec

Feature	NDCG@5
ID + Text	0.8754
ID + Text + Prompt (GPT-4o-mini)	0.8790
ID + Text + Prompt (Llama 3.1)	**0.8795**

Table 9: Gating Weights for Different Prompt Strategies

Prompt Strategy	Llama 3.1	GPT-4o-mini
P_{para}	0.1391	**0.2766**
P_{tags}	**0.4485**	0.1846
P_{guess}	0.1367	0.1403
P_{para}^{rec}	0.1328	0.1325
P_{tags}^{rec}	0.1428	0.2659

In conclusion, adding prompt embeddings significantly enhances recommendation performance, as shown by the improved NDCG@5 scores. Furthermore, based on the Llama 3.1 gating weights, P_{tags} emerge as a critical factor in improving recommendation accuracy. Similarly, the gating weights from GPT-4o-mini align with the results from individual prompt embeddings , reinforcing the importance of specific prompts, such as P_{para}, P_{tags}, and P_{tags}^{rec}, in optimizing model performance shown in Table 7.

A.6 Detailed Architecture and Process of the Self-Attention Encoder

Self-Attention Encoder architecture follows transformer encoder architecture proposed by Vaswani et al. (2017). The input embedding from each modality $\boldsymbol{Z}$ are processed independently through independent transformer encoder $Encoder$ with multiple layers. Each layer consists of three key components: Multi-Head Self-Attention (MSA), Layer Normalization (LN), and Multilayer Perceptron (MLP), all connected via residual connections. The transformation at layer l is defined as follows:

$$\boldsymbol{Y}^{l-1} = \text{MSA}(\text{LN}(\boldsymbol{Z}^{l-1})) + \text{LN}(\boldsymbol{Z}^{l-1}), \tag{10}$$

$$\boldsymbol{Z}^{l} = \text{MLP}(\text{LN}(\boldsymbol{Y}^{l-1})). \tag{11}$$

In the MSA block, self-attention computes the attention scores between tokens, allowing each token to dynamically attend to other tokens in the sequence. The attention mechanism is defined as:

$$\text{MSA}(\mathbf{X}) = \text{Attention}(\mathbf{W}^{Q}\mathbf{X}, \mathbf{W}^{K}\mathbf{X}, \mathbf{W}^{V}\mathbf{X}), \tag{12}$$

where $\mathbf{W}^{Q}$, $\mathbf{W}^{K}$, and $\mathbf{W}^{V}$ are the weight matrices used to project the input tensor $\mathbf{X}$ into queries, keys, and values, respectively. The attention scores are computed as the dot product of queries and keys, enabling the model to focus on the most relevant parts of the input.

A.7 DATASET

Our experiments utilize three datasets: *Fresh-Food E-commerce*, *House-Hold E-commerce*, *H&M (Trousers)*. The *Fresh-Food E-commerce* dataset includes a total of 1,507,388 interactions, with an average session length of 4.693 and a total of 216,576 sessions. The *House-Hold E-commerce* dataset consists of 94,984 interactions, an average session length of 5.914, and a total of 12,345 sessions. The *H&M (Trousers)* dataset contains user 3,576,972 purchase records, and a total of 416,794 sessions, with further details provided in Table 10.

We apply preprocessing steps to filter out sessions with fewer than 3 interactions to ensure sufficient data for model training. Session lengths are limited to 20 items. For shorter sessions, we pad them with zeros, while for longer sessions, we retain only the last 20 products, removing any repeat items at the end of the sequence. This adjustment allows the model to better focus on complex patterns, enhancing its applicability to real-world scenarios.

For the first two datasets, each session's purchase order involves multiple items, so we treat the answer set as a multi-choice task. The same preprocessing steps are applied to the third dataset, *H&M (Trousers)*. Although it consists solely of purchase actions, we sort the items by purchase time and treat the products bought on the last day as the answer set. Since users may purchase multiple items on the last day, this is also treated as a multi-choice task. To facilitate model computation, we pad all answers to a fixed length of 50 items.

Table 10: Statistics of Datasets

Dataset	# Sessions	# Products	Avg. Session	Avg. Purchase	# Actions
Fresh-Food	216,576	770	4.693	2.267	1,507,388
House-Hold	12,345	2464	5.914	1.780	94,984
H&M (Trousers)	416,794	11150	7.029	1.553	3,576,972

A.8 EVALUATION METRICS

For each session, we randomly sampled 100 items that the user did not interact with under the target behavior as negative samples. Finally, we report the results on the test set, while selecting the best hyper parameters using the validation set. We detail the three evaluation metrics used in our experiments: NDCG@k, HR@k, and MRR@k, where k is set to 5 and 10. Since our problem involves multiple correct answers (multi-label), we have adjusted the definitions of HR@k and MRR@k accordingly. HR is calculated by treating each correct answer as a separate single-choice task. The model ranks each correct item among negative samples, and we calculate the hit rate for each task. The final HR@k is the average of these hit rates across all correct answers in the sequence. MRR@k follows a similar approach, where we compute the reciprocal rank for each correct answer, and the final MRR@k is the average across all correct answers. Below are the definitions of each metric along with an example calculation.

NDCG@k (Normalized Discounted Cumulative Gain). NDCG@K considers both the relevance and the position of the correct items in the ranked list, with higher-ranked relevant items contributing more to the score.
The Discounted Cumulative Gain (DCG@k) is calculated by summing the relevance scores of the correct items, where the relevance score decreases logarithmically based on the item's rank position. The formula for DCG@k is:

$$\text{DCG@k} = \sum_{r=1}^{k} \frac{rel(r)}{\log_2(r+1)}, \tag{13}$$

where $rel(r)$ represents the relevance score of the item at rank r, where all items in the purchase order are assigned a relevance score of 1. Higher-ranked relevant items contribute more to the final score, as their positions are weighted more heavily in the DCG calculation.
The Ideal DCG (IDCG@k) represents the best possible ranking scenario, where all relevant items are ranked at the top. Since we are normalizing the DCG score, the ideal ranking is computed by

assuming the best-case relevance distribution. The formula for IDCG@k is:

$$\text{IDCG@k} = \sum_{r=1}^{\min(k,n)} \frac{1}{\log_2(r+1)}, \tag{14}$$

where $\min(k, n)$ ensures that we only consider the smaller of k (the cutoff) and n (the number of correct answers).
Finally, the NDCG@k is calculated by normalizing DCG by the ideal DCG. The formula for NDCG@k is:

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}. \tag{15}$$

This normalization makes NDCG range between 0 and 1, allowing consistent comparison across different queries. A higher NDCG score indicates that the correct items are ranked closer to the top.
HR@k (Hit Rate). HR@k is calculated by checking whether any of the ground truth items appear within the top k ranked items. For the multi-label task, we calculate HR for each correct item and then average the results. The formula for HR@k is:

$$\text{HR@k} = \frac{1}{|n|} \sum_{i=1}^{|n|} \mathbf{1}[\text{rank}_i \leq k], \tag{16}$$

where $\mathbf{1}[\text{rank}_i \leq k]$ is 1 if the correct item i is ranked within the top k, and 0 otherwise.
MRR@k (Mean Reciprocal Rank). MRR@k measures the ranking of the first correct item within the top k positions. For the multi-label task, MRR@k is calculated by averaging the reciprocal ranks of the correct items:

$$\text{MRR@k} = \frac{1}{|\text{n}|} \sum_{i=1}^{|\text{n}|} \frac{1}{rank_i}, \tag{17}$$

where $rank_i$ is the rank position of the correct item i.

A.9 Benchmark Models

•**GRU4Rec**(Hidasi et al., 2016; Hidasi & Karatzoglou, 2017): An RNN-based model that captures short-term user interactions, improving accuracy over item-to-item methods.
•**SASRec**(Kang & McAuley, 2018): A self-attention-based sequential model with causal masking that captures long-term user preferences by attending only to previous tokens.
•**BERT4Rec**(Sun et al., 2019): A bidirectional sequential recommendation model that uses self-attention to predict masked items in user behavior sequences, capturing both left and right context.
•**SASRec**$^+$: An enhanced version of SASRec (Kang & McAuley, 2018), which integrates item ID, text, image features. The text and image features are processed using the techniques from our MTSTRec model. To ensure stable training, we reduce the dimensionality to 256, as the model would otherwise fail to converge properly.
•**BERT4Rec**$^+$: An enhanced version of BERT4Rec (Sun et al., 2019), which integrates item ID, text, image features. The text and image features are processed using the techniques from our MTSTRec model. To ensure stable training, we reduce the dimensionality to 256, as the model would otherwise fail to converge properly.
•**MMMLP**(Liang et al., 2023): A multimodal MLP-based model that processes text, image, and price features (with price added for fair comparison) through a Feature Mixer Layer, Fusion Mixer Layer, and Prediction Layer, achieving state-of-the-art performance with linear complexity.

A.10 Implementation details

In our experiments, we tuned the hyperparameters based on validation data to ensure optimal performance. The batch size was uniformly set to 64 for all models, and the input dimension d was fixed at 512. We employed the AdamW optimizer, while the maximum sequence length N was set to 20. The fusion layers were standardized across models, with $L_{fusion} = 3$ and a dropout rate of 0.1.

For the ID feature encoder, we used 2 transformer blocks ($L_{id} = 2$) with 4 attention heads, applying a hidden layer dropout of 0.1 to maintain fairness with other benchmark models.

For other feature encoders, such as text, image, and price, we experimented with different settings. The number of each encoder layer (L_{mod}) was tested across values of {2, 4, 8}, and the number of attention heads across {1, 2, 4, 8, 16}. We also experimented with dropout rates of {0.1, 0.2, 0.3} in the hidden layers. The learning rate was tested across a range of {0.001, 0.0005, 0.0001, 0.00005, 0.00001}, while the L2 regularization penalty was tuned from {0.0001, 0.00005, 0.00001, 0.000005, 0.000001}. A gamma value of {0.9, 0.75, 0.5}was set for learning rate decay.

For the baseline models (e.g., BERT4Rec (Sun et al., 2019), SASRec (Kang & McAuley, 2018)), we ensured that key settings such as batch size, the number of encoder blocks and attention heads were aligned with our model for fair comparison. However, for other settings, we followed the recommended configurations in the original papers.

A.11 CONGIGURATION DETAILS OF THE SHARED TOKEN

Time-aligned Shared Tokens (TST) (1:1), used in our MTSTRec model, pairs each time step in the sequence with a corresponding shared token in a 1:1 relationship. This means that each product in the sequence is aligned with a single shared token, allowing features from different modalities of the same product to interact and share information at that specific time step. This design ensures that information sharing is precise and time-aligned, leading to more accurate feature fusion. **TST (1:2) & TST (1:4)** include configurations where each time step is associated with multiple shared tokens rather than just one. For example, in **TST (1:2)**, each product in the sequence is paired with two shared tokens, and in **TST (1:4)**, each product is paired with four shared tokens. These variants allow for more extensive information sharing at each time step.

Fusion Bottlenecks (all:4+1) is based on a configuration from Google (Nagrani et al., 2021) originally designed for sequence fusion classification tasks involving image and speech data. In this setting, shared tokens are not tied to a specific time step but attend to the entire sequence, enabling broader information exchange across the sequence. For a fair comparison in our multimodal sequential recommendation task, we adapted this method to a **Fusion Bottlenecks (all:20+1)** configuration, matching the number of tokens used in our TST approach.