Cascaded Cross Attention for Review-based Sequential Recommendation

Bingsen Huang\textsuperscript{1}, Jinwei Luo\textsuperscript{1,3}, Weihao Du\textsuperscript{1}, Weike Pan\textsuperscript{1}\textsuperscript{*} and Zhong Ming\textsuperscript{1,2}\textsuperscript{*}

\{huangbingsen2021, duweihao2022\}@email.szu.edu.cn, jettluo@tencent.com, \{panweike,mingz\}@szu.edu.cn

\textsuperscript{1}College of Computer Science and Software Engineering
Shenzhen University, China

\textsuperscript{2}Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ)
Shenzhen University, China

\textsuperscript{3}Tencent Music Entertainment, China
Motivation

- Most previous works only consider the (user, item, timestamp) interaction sequences, which limits the recommendation performance.
- Existing review-based sequential recommendation (RBSR) methods only use either a user’s review on items or an item’s reviews by users, overlooking their complementary nature.
- Most existing RBSR methods use a simple dot-product operation between the embeddings of a user and the candidate items for scoring, which may not adequately capture the complex relationships among the item sequence, review sequence and candidate items.
Overall of Our Solution

- To release the potential of RBSR, we propose a novel model called cascaded cross attention (CCA), which utilizes aggregated reviews to compensate for the information that is lacking in individual reviews.

- In the embedding layer of our CCA, we propose a BERT-based review embedding aggregator (BREA) to encode and merge the individual and aggregated reviews, ensuring that the review representation contains rich user preference information as well as diverse item attribute information.

- Furthermore, we propose a novel cascaded cross-attention layer to capture both the intra-sequence dependency and target-oriented preferences, which allows for a more comprehensive understanding of user preferences conditioned on item features.
Advantages of Our Solution

- Focusing on the aforementioned two issues, i.e., the insufficient utilization of review information and the inadequate exploration of multiple sequence information in most review-based sequential recommendation models, our CCA model not only effectively integrates individual and aggregated reviews to enrich user and item features, but also leverages cascaded cross-attention layers to fully explore sequence features.

- Empirical studies on three public datasets demonstrate its superiority over the state-of-the-art methods in recommendation performance. Additionally, case studies and visualization experiments are conducted to highlight the interpretability of our CCA.
Related Work (1/3)

**Sequential Recommendation**

- **Markov chains** (MCs) are initially used in early methods to capture the transition relationship between items.
- However, these traditional methods cannot capture complex relationships, and the development of *deep learning* has greatly improved sequential recommendation.
- Hence, the RNN and the CNN are applied to sequential recommendation, such as GRU4Rec [Hidasi et al., 2015] and Caser [Rashed et al., 2022].
- To further explore the features in user interaction sequences, the *attention-based methods* are also employed in sequential recommendation, such as SASRec [Kang and McAuley, 2018] and SSE-PT [Wu et al., 2020].
- In recent years, the *graph neural networks* (GNN) have developed rapidly and have been widely applied in sequential recommendation.
Related Work (2/3)

- **Review-based Recommendation**
  - Early works use *topic models* [Blei et al., 2003] to extract latent semantic factors from reviews, used for recommendation models.
  - To fully express semantic contextual information, many *deep learning techniques* are used to combine user reviews for recommendation.
  - The *CNN-based models* are first used in review-based recommendation, such as *DeepCoNN* [Zheng et al., 2017].
  - Subsequently, to find key elements in review text and improve the interpretability of models, researchers further introduce *attention mechanisms*, such as *D-Attn* [Seo et al., 2017].
  - As *graph convolutional networks* (GCN) can construct users, items and other information in recommendation systems into a graph structure [Kipf and Welling, 2017], researchers suggest incorporating user reviews into graph-based methods to further improve the recommendation accuracy.
Related Work (3/3)

- **Reviewe-based Sequential Recommendation**
  - Past research mainly uses user reviews for rating prediction tasks and does not consider using user reviews to enhance sequential recommendation.
  - RNS [Li et al., 2019] is the first to use user reviews for sequential recommendation, encoding users’ inherent preferences and sequence patterns through the semantic signals of user reviews.
  - PRIASP [Zhang et al., 2021] improves on RNS by integrating review and user-item interaction information, learning short-term and long-term preferences through fusion and convolutional neural networks.
  - DER [Chen et al., 2019] models users’ dynamic preferences by designing a time-aware gated recurrent unit (GRU), and extracts information that has a key role in the user’s current state by analyzing the reviews of products through sentence-level convolutional neural networks.
Problem Definition

- **Input**: In a review-based sequential recommendation system, there are two sets of entities: a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. We denote the records of each user $u \in \mathcal{U}$ as an item sequence (ordered by the interaction time) $S = \{i_1, i_2, \ldots, i_{|S|}\}$, $i_\ell \in \mathcal{I}$. For each user, we also utilize the review sequence $\mathcal{R} = \{r_1, r_2, \ldots, r_{|S|}\}$ and timestamp sequence $\mathcal{T} = \{t_1, t_2, \ldots, t_{|S|}\}$ to unlock the potential of our model. Here, $r_\ell$ represents the textual content of user $u$’s review on item $i_\ell$.

- **Goal**: To exploit the item sequence, review sequence and timestamp sequence and provide a recommendation list for each user $u$, in which we expect the real next interacted item $i_{|S|+1} \in \mathcal{I}\setminus S$ to appear and be ranked as high as possible.
CCA (1/2)

**Proposed Method**

**Figure:** The overview of our proposed CCA.
In general, our CCA takes the item sequence, review sequence, and the corresponding timestamp sequence of user \( u \) as input and obtains the representations of them in the embedding layer.

Specifically, the representations of the review sequence are extracted with our proposed BERT-based Review Embedding Aggregator (BREA), which encodes and merges the individual and aggregated review information.

These representations are then fed into our cascaded cross-attention layer to capture both the intra-sequence dependency and target-oriented preferences, which are used for final prediction.
We utilize an embedding layer to process the input of the model, including an item sequence encoder, a timestamp sequence encoder, and a BERT-based review embedding aggregator (BREA) to process item, timestamp, and review sequences, respectively. Finally, these input embeddings are aggregated into a user feature matrix.
We first select the latest $L$ items interacted by a user, which is abbreviated as $S = \{i_1, i_2, \ldots, i_L\}$. If a user’s interaction records are fewer than $L$, we add padding items in the front of the sequence.

**Item Sequence Encoder.** We construct a learnable item embedding matrix $M \in \mathbb{R}^{|I| \times d_e}$. Then we can represent the item sequence as an embedding matrix $E = [m_1; m_2; \ldots; m_L] \in \mathbb{R}^{L \times d_e}$.

**Timestamp Sequence Encoder.** A context feature extraction function is utilized to process the timestamp sequence $T = \{t_1, t_2, \ldots, t_L\}$. Each timestamp is parsed into temporal elements including year, month, day, week, day of the week, and day of the year. The temporal elements of the timestamp $t_\ell$ are concatenated to form the corresponding context feature embeddings $c_\ell$, resulting in an embedding matrix $C = [c_1; c_2; \ldots; c_L] \in \mathbb{R}^{L \times d_c}$ of the timestamp sequence.
BERT-based Review Embedding Aggregator. We leverage the corresponding user review sequence to increase the diversity of model inputs and enrich information sources to improve the model’s generalization ability and prediction accuracy. To effectively process user reviews, we design a BERT-based review embedding aggregator (BREA) to process review sequences.
**BERT-based Review Embedding Aggregator**

- Firstly, we utilize a pre-trained language model, BERT [Devlin et al., 2019], to encode the user review sequence \( R = \{ r_1, r_2, \ldots, r_L \} \). Specifically, for the review \( r_\ell \), we extract the last hidden state of the model and compute the mean of the token vectors to obtain a fixed-size embedding \( x_\ell \in \mathbb{R}^{1 \times d_r} \). Finally, we can obtain an embedding matrix \( X = [x_1; x_2; \ldots; x_L] \in \mathbb{R}^{L \times d_r} \) of the user review sequence.

- However, if there are numerous irrelevant or unhelpful reviews, it may not only impede users’ decision-making and experience but also adversely impact the model’s performance.
Embedding Layer (5/8)

BERT-based Review Embedding Aggregator

To obtain comprehensive reviews with rich item information, we employ an aggregation approach by combining the embeddings of all user reviews related to item $i$. The resulted aggregated review embedding $x'_i$ for the item $i$ is obtained by taking the average of the embeddings. Finally, based on the item sequence $S$, we obtain the embedding matrix $X' = [x'_1; x'_2; \ldots; x'_L] \in \mathbb{R}^{L \times d_r}$ of the aggregated review sequence.

An individual review sequence provides a wealth of information on user preferences, while an aggregated review sequence offers valuable insights on the item information and overall user evaluations. Integrating these two types of reviews can enrich the text content and better capture the user interests.
Embedding Layer (6/8)

BERT-based Review Embedding Aggregator

- We utilize the aggregated reviews with enrich information to supplement the individual reviews. To balance the importance of the individual and aggregated reviews, we employ a gating mechanism to combine $X$ and $X'$:

$$
g = \sigma \left( x_\ell W_x + x'_\ell W_{x'} + b_g \right) \tag{1}$$
$$a_\ell = g \otimes x_\ell + (1 - g) \otimes x'_\ell \tag{2}$$

where $\ell \in \{1, 2, \ldots, L\}$ is the index, $\otimes$ is the element-wise product, $W_x, W_{x'} \in \mathbb{R}^{d_r \times d_r}$, $b_g \in \mathbb{R}^{d_r}$ are learnable weights and biases, and $\sigma(\cdot)$ is a sigmoid activation function to constrain the value of each element in $g \in \mathbb{R}^{L \times d_r}$ to $(0, 1)$. $a_\ell$ represents the fused review feature that integrates the individual review information and the aggregated review information of the $\ell$th item in the user interaction history.
To integrate these features into the self-attention blocks, we concatenate the item embeddings, review embeddings, and context features as follows:

\[
z_\ell = \text{concat}_{\text{col}}(a_\ell, c_\ell) W_z + b_z \tag{3}
\]

\[
e_\ell = \text{concat}_{\text{col}}(m_\ell, z_\ell) W_e + b_e \tag{4}
\]

where \( W_z \in \mathbb{R}^{(d_r + d_c) \times d_z}, b_z \in \mathbb{R}^{d_z}, W_e \in \mathbb{R}^{(d_e + d_z) \times d_e}, b_e \in \mathbb{R}^{d_e} \) are learnable weights and biases, and \( \text{concat}_{\text{col}} \) represents column-wise concatenation of the vectors. \( z_\ell \) represents the item feature that integrates the review information and context information of the \( \ell \)th item in user interaction, and \( e_\ell \) is the feature embedding that combines the item, review, and context information. The output of this process is the final representations of the user sequences, i.e., \( \hat{E} = [e_1; e_2; \ldots; e_L] \in \mathbb{R}^{L \times d_e} \).
Finally, to achieve improved recommendation performance, we utilize the same embedding pipeline on candidate items within the recommendation list. For a candidate item $i$, we get the feature vector $e_i = f_1 \left( m_i \left\| f_2 \left( x'_i \left\| c_i \right. \right) \right) \right)$, where $f_1(·)$ and $f_2(·)$ are the linear functions in Eq.(3) and Eq.(4), and $\left\| \right.$ is the vector concatenation operator. Note that there are no interaction records between the current user and the candidate items in the dataset, only the aggregated review embedding $x'_i$ is utilized in the process of generating $e_i$. 
After obtaining the user feature matrix and candidate item feature matrix, we propose the cascaded cross-attention layer to continually refine the user embedding and mine the user’s preference information related to each candidate item.
Cascaded Cross-Attention Layer (2/8)

Self-Attention Block

Processed by embedding layers, we obtain the user feature matrix $\hat{E}$ which is comprised of sequential features from items, contexts, and reviews. To capture the internal dependency of the user sequence, we feed the embeddings $\hat{E}$ into a hierarchical self-attention block (SAB), as shown below:

$$SAB(\hat{E}) = FFL(SAL(\hat{E}))$$  \hspace{1cm} (5)$$

$$\hat{E}' = SAL(\hat{E}) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$  \hspace{1cm} (6)$$

$$FFL(\hat{E}') = \text{ReLU}(\hat{E}'W_1 + b_1)W_2 + b_2$$  \hspace{1cm} (7)$$

where $Q = \hat{E}W_Q$ represents the queries, $K = \hat{E}W_K$ the keys and $V = \hat{E}W_V$ the values. $W_Q, W_K, W_V \in \mathbb{R}^{d_e \times d_e}$ are the linear projection matrices. Note that $\sqrt{d}$ in $SAL(\cdot)$ is the scaling factor which is used to prevent overlarge values of the inner product.
Cascaded Cross-Attention Layer (3/8)

Self-Attention Block

To further capture the higher-level sequential features, we stack the self-attention block, and the $b$th ($b > 1$) block, which is defined as follows:

$$
\hat{E}^{(b)} = SAB^{(b)}(\hat{E}^{(b-1)}), \quad b \in \{1, 2, \ldots, B\}
$$

(8)

where the 1-st block is defined as $\hat{E}^{(1)} = \hat{E}$. Finally, we can obtain $\hat{E}^{(B)} = [e_1^{(B)}; e_2^{(B)}; \ldots e_L^{(B)}]$, where $e_\ell^{(B)} \in \mathbb{R}^{1 \times d_e}$ denotes the dynamic preference of the user interaction sequence at step $\ell$.

Note that we omitted the positional encoding in our model since the context features obtained from timestamp already contain explicit information about the sequence of items. We also omitted the causality mask to enable the bidirectional capture of user features.
Low-Order Interaction

Inspired by SASRec [Kang and McAuley, 2018], the next item is recommended by simply multiplying the user features and item features, which can achieve promising results. Therefore, we first use a low-order interaction to predict the probability of a candidate item $i$ being ranked as the $(\ell + 1)$th item in the sequence as follows:

$$\hat{y}_{\ell+1,i}^{(1)} = e_{\ell}^{(B)} (e_i)^T$$  \hspace{1cm} (9)

where $\hat{y}_{\ell+1,i}^{(1)}$ is the first-order predicted rating which can be represented as a factored similarity between the candidate item $i$ and the interaction items of user $u$. 

---

**Proposed Method**

**Cascaded Cross-Attention Layer (4/8)**
Cascaded Cross-Attention Layer (5/8)

Low-Order Interaction

- We adopt the binary cross-entropy loss for the low-order interaction, which is as follows:

\[
\mathcal{L}_1 = - \sum_{u \in U} \sum_{\ell=2}^{L+1} \delta(i_u^\ell)[\log(\sigma(\hat{y}_\ell^{(1)})) + \log(1 - \sigma(\hat{y}_\ell^{(1)}))] \tag{10}
\]

where \( j \in I \setminus S \) is a negative item randomly sampled for each prediction. The indicator function \( \delta(i_u^\ell) = 1 \) if \( i_u^\ell \) is not a padding item, otherwise 0.

- However, relying solely on low-order interactions, such as the dot product between the user and candidate items, may not fully utilize the rich information contained within the individual review information and aggregated review information [Yan et al., 2021]. Therefore, we design a cross-attention block and a high-order interaction to enhance the performance.
Cascaded Cross-Attention Layer (6/8)

Cross-Attention Block

Following the models [Zhang et al., 2021, Rashed et al., 2022] that also have rich review information, we design a dedicated cross-attention block to capture more complex interactions between users and candidate items:

$$e_i^o = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$  \hspace{1cm} (11)

where $Q = e_i W_Q$, $K = \hat{E}^{(B)} W_K$ and $V = \hat{E}^{(B)} W_V$ with $W_Q, W_K, W_V \in \mathbb{R}^{d_e \times d_e}$ as the projected query, key and value matrices. $e_i^o$ is a high-dimensional feature vector associated with the candidate item $i$. The high-dimensional features obtained after cross-attention not only enable the model to focus more on the most relevant aspects of the candidate items and users, but also allow further fine-grained utilization of the rich features present in both the users and items.
Cascaded Cross-Attention Layer (7/8)

High-Order Interaction

By processing the users and candidate items with cross-attention, we can obtain a refined feature vector $e_i^o$. Using these feature vectors after higher-order interactions, we can calculate the second-order predicted rating:

$$\hat{y}^{(2)}_{\ell+1, i} = \sigma(e_i^o W_o + b_o)$$  \hspace{1cm} (12)

where $W_o \in \mathbb{R}^{d_e \times 1}$ and $b_o \in \mathbb{R}$ are the weight matrices and bias vector of the output layer, respectively. $\hat{y}^{(2)}_{\ell+1, i}$ is the likelihood of the candidate item $i$ being the next item.

We also adopt the binary cross-entropy loss for the high-order interaction, which is as follows:

$$\mathcal{L}_2 = - \sum_{u \in U} \sum_{\ell=2}^{L+1} \delta(i_u^{(\ell)})[\log(\sigma(\hat{y}^{(2)}_{\ell, i_u^{(\ell)}})) + \log(1 - \sigma(\hat{y}^{(2)}_{\ell, j}))]$$ \hspace{1cm} (13)
To obtain the final predicted rating, we add the first-order and the second-order predicted rating related to the candidate item $i$:

$$\hat{y}_{\ell+1,i} = \hat{y}_{\ell+1,i}^{(1)} + \hat{y}_{\ell+1,i}^{(2)}$$  \hspace{1cm} (14)

During the training process, we construct a list of candidate items for each user by combining a positive item list and a negative item. The positive item list is composed of the penultimate $L - 1$ items from the user’s interaction sequence $S$ and the final item $i_{L+1}$ the user interacted with. The negative item list is generated by randomly selecting $L$ items with which the user has not interacted, each item carrying a timestamp identical to its corresponding positive item. Finally, the objective function is defined as $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$. The model is trained with the Adam optimizer [Kingma and Ba, 2014].
Research Questions

**RQ1:** How does our CCA perform compared to the baselines?

**RQ2:** What is the impact of different components in our CCA?

**RQ3:** What is the interpretability of the BERA in our CCA?

**RQ4:** What is the rationality of our CCA?
The Amazon dataset$^1$ is a publicly available dataset containing a substantial amount of product reviews. It consists of various fields such as review text and interaction time. Following previous works [Lin et al., 2020, Kang and McAuley, 2018], we conduct experiments on the 5-core version of the dataset, specifically on three categories: Video Games, Toys, and Clothing. This version filters out users and items with less than five reviews and removes invalid data.

$^1$http://jmcauley.ucsd.edu/data/amazon/
In order to prevent the use of future data to predict past data, we propose the following data processing method.

- The **duplicate user-item pairs** are removed, retaining only the latest record, and the resulted interaction records are sorted by chronological order.

- The total number of records is calculated and is then used to find the **timestamp corresponding to 80% of the number of records as the dividing line** [Ying et al., 2018], which is used to divide the dataset into training, validation and test sets.

- If an interaction sequence is **entirely on the right-hand side of the dividing line**, the entire sequence is discarded.
Datasets (3/4)

- If an interaction sequence that is entirely on the left-hand side of the dividing line and has a length of at least 5, it is retained for training only.

- For a user interaction sequence that crosses the dividing line:
  - If the length of the left subsequence of the dividing line is less than 5, the entire sequence is discarded.
  - If the left subsequence of the dividing line has a length of at least 5, the entire sequence is kept. The left subsequence of the dividing line is used as the training set, the first record on the right-hand side of the dividing line is used as the validation set, and the remainder is used as the test set.

- If an item only appears in the validation and test sets, all records related to that item are discarded.
### Datasets (4/4)

<table>
<thead>
<tr>
<th></th>
<th>Video Games</th>
<th>Toys</th>
<th>Clothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>17,796</td>
<td>14,191</td>
<td>27,824</td>
</tr>
<tr>
<td>Number of items</td>
<td>10,086</td>
<td>11,617</td>
<td>22,622</td>
</tr>
<tr>
<td>Number of interactions</td>
<td>180,129</td>
<td>132,003</td>
<td>209,252</td>
</tr>
<tr>
<td>Avg. length</td>
<td>10.12</td>
<td>9.30</td>
<td>7.52</td>
</tr>
<tr>
<td>Density</td>
<td>0.10%</td>
<td>0.08%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

Table: Statistics of the processed datasets, where **Avg.length** denotes the average length of users’ interaction sequences in the datasets and **Density** denotes the sparsity of interaction behavior between users and items.
Evaluation Metrics

We utilize two widely used metrics in the recommender systems community: recall (Rec@10) and normalized discounted cumulative gain (NDCG@10), with Rec@10 being the primary evaluation metric.

To evaluate the model’s performance, we follow [Lin et al., 2020] and randomly select 100 un-interacted items as negative samples, provide them with the same timestamp as their corresponding positive test item, and rank the positive test item among them.

To ensure the statistical significance of the experimental results, each experiment is repeated 5 times, and the average and the standard deviation of the results are reported at last.
Five sequential recommendation methods:

- **GRU4Rec** [Hidasi et al., 2015]. A session-based RNN model which applies GRU to model item sequences, which solves the recommendation problem with short sequence data.

- **Caser** [Tang and Wang, 2018]. A CNN-based sequential recommendation model that uses horizontal and vertical convolutions to capture different sequential patterns in user sequences.

- **SASRec** [Kang and McAuley, 2018]. A self-attention based sequential recommendation model that can capture long-term dependency in a user’s historical behavior sequences.

- **SSE-PT** [Wu et al., 2020]. A personalized Transformer-based sequential recommendation model that effectively captures temporal information in user preferences.

- **FMLP** [Zhou et al., 2022]. A sequential recommendation model using MLP and learnable filters to mitigate overfitting in deep recommendation models with noisy behavior sequences.
Baselines (2/2)

Four review-based sequential recommendation methods:

- **RNS** [Li et al., 2019]. A review-based sequential recommendation model encoding users and items into aspect-aware representations from reviews, accounting for long-term and short-term preferences by utilizing rich text information.

- **PIRSP** [Zhang et al., 2021]. A review-based sequential recommendation model which builds on RNS by encoding items and reviews to capture the item and review sequential patterns.

- **DER** [Chen et al., 2019]. A dynamic interpretable recommendation model that simulates users’ dynamic preferences using a time-aware gated recurrent unit and extracts user preferences from review information through a sentence-level CNN.

- **CARCA** [Rashed et al., 2022]. A cross-attention based recommendation model, which extracts features of users through some multi-head self-attention blocks and calculates the correlation between all users and the candidate items through cross-attention.
We implement SASRec, Caser, FMLP, SSE-PT and CARCA based on the code provided in their respective papers.

For GRU4Rec\(^2\), RNS\(^3\) and DER\(^4\), we use the code available on the open-source platforms. We adapt the RNS code for PIRSP by adding an item sequence encoder block.

Our CCA is based on CARCA, with a BERT-based review embedding aggregator (BREA) added to the embedding layer and adjustments made to the attention block. We use a pre-trained language model [Wolf et al., 2020], bert-base-uncased\(^5\), to process the user reviews and replace the item attributes in CARCA with the resulted embeddings.

---

\(^2\)https://github.com/hidasib/GRU4Rec  
\(^3\)https://github.com/WHUIR/RNS  
\(^4\)https://github.com/gearsuccess/DER  
\(^5\)https://huggingface.co/bert-base-uncased
To ensure a fair comparison, we tune the common parameters for all models: the dimension of project embedding $d$ is tuned in $[10, 500]$ with a step size of 10, the sequence length $L$ is tuned in $[10, 200]$ with a step size of 5. Other key parameters are adjusted based on the recommendations in the corresponding papers.

For our CCA, following [Rashed et al., 2022], we choose the learning rate from \{0.001, 0.0001, 0.00001, 0.000006, 0.000002, 0.000001\}, the number of self-attention blocks from \{1, 2, 3, 4, 5\}, the number of attention heads from \{1, 2, 3, 4, 5\}, and L2 regularization weight from \{0, 0.001, 0.0001, 0.00001, 0.000001\}. The batch size is set at 128, the dropout rate is set at 0.5, and the review embedding size is set at 768.
Performance Comparison (RQ1) (1/4)

Table: Recommendation performance comparison between our CCA and nine baselines on three datasets, where the best results are highlighted in boldface and the second best results are underlined. Improvements over the corresponding best baselines are shown in the last row.

<table>
<thead>
<tr>
<th>Model</th>
<th>Games</th>
<th>Toys</th>
<th>Clothing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rec@10</td>
<td>NDCG@10</td>
<td>Rec@10</td>
</tr>
<tr>
<td>GRU4Rec</td>
<td>0.5873 ± 0.0051</td>
<td>0.3652 ± 0.0012</td>
<td>0.3612 ± 0.0041</td>
</tr>
<tr>
<td>Caser</td>
<td>0.5466 ± 0.0067</td>
<td>0.3434 ± 0.0029</td>
<td>0.3567 ± 0.0042</td>
</tr>
<tr>
<td>SASRec</td>
<td>0.6728 ± 0.0086</td>
<td>0.4411 ± 0.0056</td>
<td>0.4417 ± 0.0048</td>
</tr>
<tr>
<td>SSE-PT</td>
<td>0.6782 ± 0.0040</td>
<td>0.4461 ± 0.0064</td>
<td>0.4638 ± 0.0023</td>
</tr>
<tr>
<td>FMLP</td>
<td>0.6140 ± 0.0016</td>
<td>0.4021 ± 0.0011</td>
<td>0.3779 ± 0.0035</td>
</tr>
<tr>
<td>RNS</td>
<td>0.6123 ± 0.0133</td>
<td>0.2675 ± 0.0099</td>
<td>0.4066 ± 0.0334</td>
</tr>
<tr>
<td>PIRSP</td>
<td>0.6379 ± 0.0206</td>
<td>0.2883 ± 0.0145</td>
<td>0.3941 ± 0.0083</td>
</tr>
<tr>
<td>DER</td>
<td>0.6320 ± 0.0028</td>
<td>0.3939 ± 0.0013</td>
<td>0.4418 ± 0.0102</td>
</tr>
<tr>
<td>CARCA</td>
<td>0.6365 ± 0.0127</td>
<td>0.4001 ± 0.0157</td>
<td>0.4516 ± 0.0047</td>
</tr>
<tr>
<td>CCA</td>
<td>0.7044 ± 0.0035</td>
<td>0.4632 ± 0.0038</td>
<td>0.5069 ± 0.0063</td>
</tr>
<tr>
<td>Improvement</td>
<td>3.86%</td>
<td>3.83%</td>
<td>9.29%</td>
</tr>
</tbody>
</table>

Huang et al., (SZU)
We can have the following observations:

- Firstly, Caser **underperforms other baselines** in terms of Rec@10 on all three datasets and falls significantly behind the best-performing baseline on NDCG@10. The RNN-based model GRU4Rec **comes second** on Rec@10. This result suggests that models based on CNN and RNN are not effective in addressing the feature sparsity issue arising from mining sequence information, compared with the review-based models possessing richer text features.
Secondly, CARCA and SSE-PT outperform other baselines on all metrics except on NDCG@10 on the Clothing dataset. It can be seen from the table that SSE-PT performs better than CARCA in terms of Rec@10 on the relatively dense Video Games dataset by 6.55%, whereas CARCA performs better than SSE-PT by 8.99% on the relatively sparse Clothing dataset. This observation indicates that user reviews contain rich information, and CARCA can effectively capture user features by utilizing cross-attention to overcome the issue of data sparsity.
Thirdly, our proposed model CCA achieves the best performance on all datasets, indicating the superiority of our approach. Our model surpasses the best-performing baseline by an average performance gain of 3.85%, 6.68%, and 10.55% across the three datasets, respectively, demonstrating that its recommendation performance improves as the data sparsity increases. We ascribe the improvement of our model to the following two factors:

- The BERT-based review embedding aggregator (BREA) in the embedding layer can effectively encode and merge the individual and aggregated reviews, ensuring that both user preference and user attribute information are contained in the reviews.
- Our cascaded cross-attention layer fully utilizes the features of both users and candidate items, capturing the intra-sequence dependency and target-oriented preferences of users on different candidate items, leading to an improvement in the model’s recommendation performance.
Ablation Study (RQ2) (1/5)

To investigate the effects of different components in our CCA, we conduct ablation studies.

- we test CCA without individual reviews (denoted as “w/o individual reviews in BREA”), without aggregated reviews (denoted as “w/o aggregated reviews in BREA”), without the gating mechanism (denoted as “w/o gating mechanism in BREA”), without all reviews (denoted as “w/o BREA”), without low-order interaction (denoted as “w/o low-order interaction”), and without the cross-attention block and high-order interaction (denoted as “w/o high-order interaction”).

- The ablation study is about model inputs and framework components, rather than the timestamp embedding, as this part is inherited from CARCA.
### Ablation Study (RQ2) (2/5)

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Games</th>
<th></th>
<th>Toys</th>
<th></th>
<th>Clothing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rec@10</td>
<td>NDCG@10</td>
<td>Rec@10</td>
<td>NDCG@10</td>
<td>Rec@10</td>
<td>NDCG@10</td>
</tr>
<tr>
<td>CCA</td>
<td>0.7044</td>
<td>0.4632</td>
<td>0.5069</td>
<td>0.3071</td>
<td>0.4710</td>
<td>0.2804</td>
</tr>
<tr>
<td>w/o individual reviews in BREA</td>
<td>0.7014</td>
<td>0.4632</td>
<td>0.5041</td>
<td>0.3066</td>
<td>0.4677</td>
<td>0.2792</td>
</tr>
<tr>
<td>w/o aggregated reviews in BREA</td>
<td>0.7005</td>
<td>0.4658</td>
<td>0.4851</td>
<td>0.2929</td>
<td>0.4416</td>
<td>0.2588</td>
</tr>
<tr>
<td>w/o gating mechanism in BREA</td>
<td>0.6941</td>
<td>0.4565</td>
<td>0.4751</td>
<td>0.2803</td>
<td>0.4605</td>
<td>0.2715</td>
</tr>
<tr>
<td>w/o BREA</td>
<td>0.6810</td>
<td>0.4440</td>
<td>0.4619</td>
<td>0.2839</td>
<td>0.3835</td>
<td>0.2274</td>
</tr>
<tr>
<td>w/o low-order interaction</td>
<td>0.6285</td>
<td>0.3947</td>
<td>0.4335</td>
<td>0.2456</td>
<td>0.4077</td>
<td>0.2317</td>
</tr>
<tr>
<td>w/o high-order interaction</td>
<td>0.6622</td>
<td>0.4200</td>
<td>0.4811</td>
<td>0.2821</td>
<td>0.4660</td>
<td>0.2623</td>
</tr>
</tbody>
</table>

**Table:** Recommendation performance in ablation studies with different architectures on three datasets.

---

Huang et al., (SZU)
Ablation Study (RQ2) (3/5)

We have the following observations.

- It is evident that the model’s recommendation performance deteriorates when either individual or aggregated reviews are removed or both are removed. This effect is especially noticeable on the sparse Clothing dataset, highlighting how review text not only enriches user behavior but also adds evaluations and feedback on products, contributing to the mitigation of the data sparsity issue.

- Comparing the impact of removing individual reviews and aggregated reviews, it is determined that the latter has a greater impact on the model’s performance. This is on account of integrating reviews by multiple users on the same item, providing a more comprehensive and cohesive representation of item information and user experience.
When all reviews are deleted, the model’s performance declines more significantly than when either individual or aggregated reviews are removed. This is because the model requires individual reviews in order to deliver personalized recommendations based on user preferences, while aggregated reviews offer vital information to understand the characteristics of both users and items. The gating mechanism utilized in the model aims to balance the relationship between the individual and aggregated reviews.
Ablation Study (RQ2) (5/5)

- The CCA without low-order interaction and high-order interaction also shows a decrease in recommendation performance. The low-order interaction uses the dot product between user embeddings processed by multi-head attention and candidate item embeddings to efficiently obtain the candidate item score.

- Removal of the low-order interaction leads to a considerable decline in model performance (such as 13.44% Rec@10 on Clothing), implying the importance of this structure in capturing user interests. On the other hand, the cross-attention block and the high-order interaction are designed to fully utilize the review text on candidate items by exploring users’ interests and preferences, thereby enhancing the recommendation performance. As a result, the removal of the cross-attention block and high-order interaction similarly leads to a decline in model performance (e.g., by 5.99% on Rec@10 on Video Games).
In our CCA, the **gating mechanism** in the BREA module plays a crucial role in **balancing individual preferences and aggregated perceptions in reviews**.

To provide an intuitive illustration of its function, we present three examples in the following figure, each featuring an individual review and three corresponding other reviews for an item.
Figure: Some examples of reviews that demonstrate the role of the BREA module. Note that the individual reviews are marked in italics, with red text indicating user preferences and blue text reflecting the item attributes.
Specifically, we first take the average of the weight $g \in \mathbb{R}^{L \times d_r}$ to obtain the weight of each individual review in the user’s interaction sequence. The range of the observed weights is $(0.4, 0.6)$.

Therefore, when the weight of an individual review approaches 0.6, the information from the individual review dominates the text features in the interaction record.

For instance, in Example 1, the user ‘A2D04GV8DESKC4’ provides detailed attribute information about the item ‘B003JVF728’ in the review (blue font) and expresses a positive user experience (red font). Despite the other users expressing many dissatisfactions in the reviews and the overall evaluation being low, the gating mechanism assigns a high weight to the individual review to ensure that the user’s personal attitude dominates the text features.
when the weight of an individual review approaches 0.4, it indicates that the individual review contains little information, and the text features are dominated by the aggregated reviews. In Example 3 of Fig. 2, even though the user’s evaluation of the item is much lower than the collective evaluation, the number of words in the individual review is too small to generate effective text features. Hence, the gating mechanism assigns a low weight to the individual review.

These three examples demonstrate that the gating mechanism in our BREA module can effectively balance the individual and aggregated reviews, resulting in text features that contain rich information and improving the accuracy and effectiveness of the recommendation systems. Therefore, using the gating mechanism is an effective method for considering both the individual and aggregated reviews.
To investigate the ability of the cascaded cross-attention layer to capture intra-sequence dependency and target-oriented preferences, we visualize both the attention scores in the self-attention block and the attention scores in the cross-attention block with various candidate items.

We visualize the attention scores of each user as a heatmap matrix shown, where the horizontal axis denotes the items in the user interaction sequence, and the vertical axis represents the candidate items.

Note that the first row in the figure displays the user’s attention scores of the self-attention block, while the remaining rows represent the scores of users with different candidate items through cross-attention.
Figure: Visualization of the attention scores of a user to different candidate items in the cross-attention block of our CCA.
We can see that users’ attention scores change when they interact with different candidate items. This phenomenon demonstrates that the cascaded cross-attention layers can continuously refine user embeddings and fully capture their preferences.

For instance, in the example on the left-hand side, when the candidate item is “Hungry Shark Evolution”, the interaction sequence with the highest score is “Minion Rush: Running game”, which is also classified as a casual game.

Similarly, when the candidate item is “Injustice: Gods Among Us”, the highest score is associated with “Binary Domain”, which falls into the action-adventure game category. In general, these instances further demonstrate the rationality and interpretability of our CCA.
Conclusions

- We present a novel model, named cascaded cross attention (CCA), which is specifically designed to effectively incorporate and leverage user reviews in sequential recommendation.

- We introduce a BERT-based review embedding aggregator (BREA), which serves to encode and merge both the individual and aggregated reviews, thereby ensuring the informativeness from both user and item sides.

- To enhance the utilization of review features for mining user preferences, we design a cascaded cross-attention block that effectively captures both the intra-sequence dependency and target-oriented preferences, which significantly improves the model’s recommendation performance.

- The experimental results show that our method outperforms the state-of-the-art models on three real-world datasets.
Future Work

For future works, we are interested in extending our CCA to capture high-order structured information as that in GNN.
Conclusions

- We thank the support of National Natural Science Foundation of China No. 62172283, No. 62272315 and No. 61836005. We thank Mr. Zinan Lin and Dr. Dugang Liu for their helpful discussions.

- If you have any questions, please feel free to contact us.
Latent Dirichlet allocation.

Dynamic explainable recommendation based on neural attentive models.

BERT: Pre-training of deep bidirectional transformers for language understanding.

Session-based recommendations with recurrent neural networks.
In Proceedings of the 4th International Conference on Learning Representations.

Self-attentive sequential recommendation.

Adam: A method for stochastic optimization.

Semi-supervised classification with graph convolutional networks.
In Proceedings of the 5th International Conference on Learning Representations.

A review-driven neural model for sequential recommendation.
In Proceedings of the 28th International Joint Conference on Artificial Intelligence, pages 2866–2872.
FISSA: Fusing item similarity models with self-attention networks for sequential recommendation.

CARCA: Context and attribute-aware next-item recommendation via cross-attention.

Interpretable convolutional neural networks with dual local and global attention for review rating prediction.

Personalized top-n sequential recommendation via convolutional sequence embedding.

Transformers: State-of-the-art natural language processing.

SSE-PT: Sequential recommendation via personalized Transformer.

Modeling low-and high-order feature interactions with fm and self-attention network.
Applied Intelligence, pages 3189–3201.

Sequential recommender system based on hierarchical attention network.
In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 3926–3932.
Improving current interest with item and review sequential patterns for sequential recommendation.
*Engineering Applications of Artificial Intelligence*, page 104348.

Joint deep modeling of users and items using reviews for recommendation.

Filter-enhanced mlp is all you need for sequential recommendation.