PersonalTM: Transformer Memory for Personalized Retrieval

Ruixue Lian
Department of Electrical and Computer Engineering,
University of Wisconsin-Madison
Madison, WI, USA
ruixue.lian@wisc.edu

Sixing Lu, Clint Solomon, Gustavo Aguilar,
Pragaash Ponnusamy, Jialong Han, Chengyuan Ma, Chenlei Guo
Amazon Alexa AI
Seattle, WA, USA
cynthialu,matclint,gustalas,ponnup
jialongh,mchengyu,guochenl@amazon.com

ABSTRACT
The Transformer Memory as a Differentiable Search Index (DSI) [32] has been proposed as a new information retrieval paradigm, which aims to address the limitations of dual-encoder retrieval framework based on the similarity score. The DSI framework outperforms strong baselines by directly generating relevant document identifiers from queries without relying on an explicit index. The memorization power of DSI framework makes it suitable for personalized retrieval tasks. Therefore, we propose a Personal Transformer Memory (PersonalTM) architecture for personalized text retrieval. PersonalTM incorporates user-specific profiles and contextual user click behaviors, and introduces hierarchical loss in the decoding process to align with the hierarchical assignment of document identifier. Additionally, PersonalTM also employs an adapter architecture to improve the scalability for index updates and reduce computation costs, compared to the vanilla DSI. Experiments show that PersonalTM outperforms the DSI baseline, BM25, fine-tuned dual-encoder, and other personalized models in terms of precision at top 1st and 10th positions and Mean Reciprocal Rank (MRR). Specifically, PersonalTM improves p@1 by 58%, 49%, and 12% compared to BM25, Dual-encoder, and DSI, respectively.

CCS CONCEPTS
• Information systems → Personalization.

KEYWORDS
Search, Personalization, Transformer Memory

ACM Reference Format:

*Work was done as an intern at Amazon Alexa AI.

1 INTRODUCTION
In recent years, IR has undergone substantial progress owning to the resurgence of deep neural networks, with transformer-based Language Models (LMs) playing a particularly significant role. These models possess the capability of learning natural language representations from vast amounts of data, contributing to the advancement of IR [3, 11, 14, 20, 29, 30, 33]. Considering the Dual Encoder (DE) approach [9, 39] is unable to learn the deep interaction between queries and documents, several sequence-to-sequence frameworks have been proposed for IR, enabling generate documents relevant to the queries directly [2, 8, 18, 19, 21, 31, 32, 35, 42]. For instance, WebGPT proposes a method to answer long-form questions in a text-based web-browsing environment by fine-tuning GPT-3 [3, 28], DSI utilizes a transformer-based encoder-decoder network to generate a ranked list of relevant document id [32]. The goal of DSI is to implicitly learn the relationship between query document id, and between document and document id, by encoding all information into the model parameters. This framework simplifies the system architecture by utilizing a single LM, and leverages the memorization ability of the transformer.

Meanwhile, personalized retrieval is becoming increasingly important, aiming to refine queries and tailor retrieval results to specific preferences of individual users [10, 25, 26, 38, 41, 43]. Numerous studies have been made to personalized retrieval, demonstrating precision improvement by incorporating user profiles and contextual history [22, 24]. For example, P-Click [1] reranks the documents based on a user’s clicks for a given query, and SLTB [12] outputs personalized ranking list by utilizing diverse clicked-based or topic-based features. [13, 40] uses RNN-based networks to extract short- and long-term user profiles from personalized historical information and apply it to DE.

In this work, we enhance performance for personalized retrieval by incorporating user identification, contextual history, and user click behaviors into the transformer memory framework. Our method involves a novel decoding architecture and applying hierarchical loss aligned with the document id framework. Furthermore, we propose a flexible adapter strategy to facilitate retraining in the event of an index update. The main contributions of our work are:

• Compared with other personalized retrieval works, we utilize an end-to-end LM to generate relevant document id given the query directly.
• We leverage user identifiers and user-specific contextual data as personalized features into the DSI architecture. Different features are fed to different decoder cross-attention
layers to effectively integrate the contextual features without increasing trainable parameters. 
- We apply a hierarchical loss function at the decoding stage to further boost the performance aligned with the semantic hierarchical document id assignment.
- We adopt the prefix adapter to learn the interaction between query and contextual data. Model parameters and training time are reduced by 10x and 2x compared to fine-tuning, making it suitable for practical deployment.

2 PROPOSED METHODS

In this work, we propose a novel approach for personalized retrieval using a personal end-to-end transformer model (PersonalTM). In this section, we describe a model architecture that integrates the two personalized features into the PersonalTM, with a hierarchical loss at the decoding stage to enhance performance. Additionally, we implement relevant information selection to optimize the utilization of contextual features and incorporate a prefix-adaptor for more agile training.

2.1 PersonalTM model architecture integrating personalized features

Given a query, we utilize a transformer-based encoder-decoder network (such as T5) as the base structure to generate relevant document ids with two types of additional personalized information: 1) user profiles such as user identifiers; 2) personal context such as user browsing histories in cookies. The model is trained to learn the relationships between the query and document id, document and document id, as well as the query and the personalized information.

As shown in Figure 1, let \( q \) be the query, \( p \) be static user identifier, \( d \) be target document, and \( h \) be personal context feature. We concatenate \( p \) to \( q \) into one sequence \([p; q]\) so that \( p \) is attended to \( q \) by self-attention layers in the encoder, so the model can learn the connection between this specific user and his preferred \( d \). \( p \) is a synthetic user identifier constructed for each user by randomly selecting and concatenating \( k \) tokens from BERT tokenizer dictionary [27], we use \( k = 4 \). \( h \) is user-clicked documents.

Instead of concatenating \( h \) to \( q \), we design a decoding structure so that they interact in the embedding space as shown in Figure 2 for the following reasons: 1) it is not effective to concatenate \( h \) with \( q \) due to input length limitation; 2) the long \( h \) cannot fully interact with \( q \) with simple concatenation; 3) decoding stage serves more on learning the connection between \( q \) and \( d \), and that is where \( h \) contributes to. To better use \( h \) since relevant contextual information would be more useful, we propose two similarity measurement modules, which are introduced in section 2.3.

Let \( f \) be the encoder, we extract \( f([p; q]) \) and \( f(h) \) as shown in Figure 2, and input them to the higher and lower decoder layers via cross-attention, separately. Each feature representation is a vector with dimension \( n \times l \times d \), where \( n \) is the batch size, \( l \) is the max length of input sequence, \( d \) is the feature dimension. Specifically, \( f(h) \) input to the first decoder layer and \( f([p; q]) \) input to the rest decoding layers. This allows \( h \) to be integrated without adding additional layers or increasing the trainable parameters of the model.

Because information about the past vanishes in left-to-right language models [34], the information injected in the upper layers would have higher weights than that in the lower levels, reflecting the different importance of \([p; q] \text{ and } h \).

Multiple works have shown the effectiveness of the fusion of different types of features at decoder layers via cross-attention [4–7]. We also experiment with another decoding structure that involves a fusion layer to merge \( f([p; q]) \) and \( f(h) \), and feed this fused information to each decoder cross-attention layer. Specifically, we extract \( f([p; q]) \) and \( f(h) \), and concatenate these embeddings by \( [f([p; q]); f(h)] \). An MLP layer consisting of two linear layers is used to project \( [f([p; q]); f(h)] \) to the original dimension aligning the decoder input. This way, query and personalized information are fused, and are input into every decoder cross-attention layer.

![Figure 1: The training workflow of PersonalTM model architecture, integrating personalized features.](image)

![Figure 2: An overview of our proposed encoder-decoder architecture. A similarity measurement is used to select relevant contextual personalized information. \( f([p; q]) \) and \( f(h) \) are fed into higher and lower decoder layers. Prefix-adaptor is applied and their parameters are marked in yellow and pink.](image)
training. The keys and values of the attention head in each self-attention and cross-attention layer are prepended with randomly initialized parameters. As shown in Figure 2, the newly injected prefix parameters are marked yellow and pink.

2.2 Hierarchical loss

Through semantic clustering, we assign each document a unique document id, similar to [32]. For instance, any country music lyrics and country lyrics tabs chords for country music fun should have closer or even the same high-level clusters because they have similar meanings. However, decoding the document id by digit has the problem of accumulated errors. To leverage the semantic hierarchy of the document id and penalize more severe semantic retrieval errors, we apply an additional hierarchical loss based on the document id hierarchy at the decoding steps. Therefore, prediction error on the first several digits in the document id (top-level clusters) will have a higher impact than lower digits (lower-level clusters) because a such error will end up with documents having different topics with the query.

The overall loss function we use is shown in E.q. (2), where the first term \( l_0 \) refers to cross-entropy loss as shown in E.q. (1)

\[
l_0 = \text{cross-entropy}(\text{logits, labels}) \quad (1)
\]

\[
l = l_0 + \sum_i w_i \cdot h_i; \quad \sum_i w_i = 1, w_1 > w_2 > \ldots \geq w_{n-1} \geq w_n \quad (2)
\]

The second term in E.q. (2) refers to the hierarchical loss. Specifically, we multiply each digit of the original loss with a scalar \( w_i \), by which different penalties are applied to different positions in the document id, modulating their importance accordingly. We apply higher weights to higher-level positions as shown in E.q. (2).

2.3 Relevance denoise for personal context

Intuitively, only relevant personal context is useful because it implicitly increases the weights on tokens that imply user’s preference. In contrast, irrelevant context introduces noise and thus misleads the model to the wrong predictions. For instance, given an incoming query python running environment. Among this user’s browsed history, some of them (e.g. python programming) are relevant, while some (e.g. what do pythons eat or country music) are irrelevant to this query. Our experiment has demonstrated purely relying on a model to do denoise is very inefficient. Therefore, we adopt the following two mechanisms to retain as much useful personalized information for the predictions.

The first mechanism is feature selection for \( h \) on the sequence level among this user’s browsing cookies, as shown in Figure 1. We apply a lightweight algorithm to select relevant personal context among user-browsed history. The algorithm selects the relevant documents according to the ratio of how many tokens in the document overlapped in the incoming query. The user-clicked documents in the session with this ratio greater than a certain value are considered personal context kept. We use the clicked documents in the latest session as the default context.

The second mechanism is similarity measurement between \( f([p; q]) \) and \( f(h) \) in the latent space, as shown in Figure 2. A cosine similarity score \( s \) is calculated between \( f([p; q]) \) and \( f(h) \). If \( s \) is greater than a certain threshold, \( h \) is kept and integrated into the decoding stage. Otherwise, \( h \) is omitted.

Algorithm 1 Relevant personal context selection

1: for row in testset do
2: \( u \_ \text{history} = \text{history}[\text{row}['\text{user}\_\text{id}']]; \) query = row["query"]
3: for \( h \_ \text{history} \) in \( u \_ \text{history} \) do
4: \( \text{ratio} = \text{Similarity}(\text{history, query}) \)
5: \( \text{relevant} \_ \text{history} = \text{list}() \)
6: if \( \text{ratio} \geq \text{threshold} \) then
7: \( \text{relevant} \_ \text{history}.\text{append}((\text{ratio, history})) \)
8: end if
9: end for
10: return relevant \_ history[\text{argmax}(\text{ratio})] if relevant \_ history is not None else return u \_ history[1-1]
11: end for

3 EXPERIMENTS

3.1 Datasets

We use AOL4PS [15] dataset in our experiments. It contains the query, the corresponding clicked documents, and timestamps indicating the timeline of each user’s search history. This dataset has a range of 12 weeks. We select relevant personal context from the first 9 weeks which is considered as historical data. The samples in the last 3 weeks are divided into training and test sets. The size of the training set and test set are 218,559 and 53,357, respectively. The number of distinct queries is 382,222 in the total dataset. Among them, 19,957 queries in the test set do not appear in the training set, and they are considered as zero-shot case samples.

3.2 Experimental Setup

The pretrained T5 (15-base) is our backbone model [36]. We use AdamW as an optimizer with a learning rate of \( 2e-5, 5e-5 \) for the training, and adopt batch size = 128 in all settings. In our experiments with prefix adaptor, we use a prefix length of 5. The hidden states dimension of the MLP layer is 512, and the dropout is 0.1 in all settings. To construct the hierarchical document ids, k-means (\( k = 10 \)) clustering provided by fast-kmeans \(^1\) is applied recursively over all document embeddings generated from the pre-trained BERT (bert-base-uncased) model. All documents are clustered into 10 clusters. The algorithm is applied recursively if the cluster size exceeds 100. Clusters with sizes smaller than 100 documents are assigned an arbitrary unique number from 0-99 as the cluster id. The next level’s cluster id is appended to the current level. The average document id length is 6. The threshold in Algorithm 1 and \( s \) are 0.6, and 0.8, respectively. For the weights in Function (2), \( w_1 \) is 3/6, \( w_2 \) is 2/6, \( w_3 \) is 1/6.

3.3 Experimental Results and Analysis

Experiment results between PersonalTM and several baseline methods are presented in Table 1. We use p@k and Mean Reciprocal Rank (MRR) as our evaluation metrics, where p@k measures the accuracy of the top k documents for the incoming query, MRR

\(^1\)https://pypi.org/project/fast-pytorch-kmeans/
calculates the mean of the inverse of the ranks at which the first relevant document is retrieved for the incoming query. Both of them are in %. Consistent with the observations in [32], DSI significantly outperforms BM25 and finetuned DE (encoder in Finetuned DE is the same as the encoder of finetuned DSI). With our proposed hierarchical loss function, p@1 increases by 0.66% compared to the one with cross-entropy loss only (3 – 4 in Table 1). It is because hierarchical loss grants higher penalties for the error of higher-level digits, it improves the precision. To further investigate how the hierarchical loss contributes to the performance, we find that hierarchical loss improves the precision at every document id digit, especially at the first three digits, with an average p@1 increment of 4.83%.

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>p@1</th>
<th>p@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BM25</td>
<td>21.61</td>
<td>32.87</td>
<td>25.46</td>
</tr>
<tr>
<td>2</td>
<td>Finetuned DE</td>
<td>30.58</td>
<td>42.13</td>
<td>34.25</td>
</tr>
<tr>
<td>3</td>
<td>DSI</td>
<td>67.42</td>
<td>75.84</td>
<td>70.58</td>
</tr>
<tr>
<td>4</td>
<td>DSI + HieLoss</td>
<td>68.08</td>
<td>77.33</td>
<td>71.52</td>
</tr>
<tr>
<td>5</td>
<td>DSI + UserIdentifier</td>
<td>69.16</td>
<td>77.60</td>
<td>72.19</td>
</tr>
<tr>
<td>6</td>
<td>DSI + HieLoss + UserIdentifier</td>
<td>70.06</td>
<td>78.47</td>
<td>73.04</td>
</tr>
<tr>
<td>7</td>
<td>PersonalTM (h in latest session)</td>
<td>71.20</td>
<td>80.10</td>
<td>74.25</td>
</tr>
<tr>
<td>8</td>
<td>PersonalTM (h relevant to q)</td>
<td>79.60</td>
<td>87.47</td>
<td>82.51</td>
</tr>
</tbody>
</table>

Table 1: Results of baseline methods and PersonalTM. Note all baselines and treatments are finetuned with the same training dataset.

By comparing lines 3 (no user identifier p) and 5 (with user identifier p) in Table 1, p@1 is improved by 1.74%. This indicates p can effectively help the model remember user’s preference, particularly for predictions of unseen queries from the existing users.

Comparison of lines 6 and 7 in Table 1 demonstrates the strength of the proposed model structure involving personal context h. By only using h from the latest session, p@1 can be improved by 1.14%. By comparing line 8 with lines 6 and 7, we can conclude that after applying the similarity selection to denoise personalized context, p@1 is significantly improved by 9.54% and 8.40% respectively.

We also compare the performance with other personalization methods proposed in previous work in Table 2. PersonalTM has a comparable or better performance compared to other personalization methods.

The results of the prefix adapter are shown in Table 3. By comparing lines 1 – 2, the model parameters and training time of the adapter are reduced by 10x and 2x compared to fine-tuning, respectively. It significantly reduces cost and improves agility for index updates while achieving comparable performance.

The performance of relevant personal context similarity measurements is shown in lines 3.1 and 4.1 in Table 3. Comparing 3.1 and 3.2 as well as 4.1 and 4.2, we find that our proposed PersonalTM beats the naive method that simply uses the latest clicked document by 1.09% and 1.62% for p@1, respectively. It is because our model structure explicitly learns the mapping between the query and the personal context. Besides, the proposed PersonalTM forces the decoder to digest this information without diluting it by the long forwarding network. The result of fusion performance is shown in line 4.3 of Table 3. It improves p@1 by 2.16% compared to PersonalTM without fusion. Note fusion increases the computation time by an additional 30%, especially during the decoding process. Therefore, there is trade-offs between the computation cost, model size or storage space, and performance.

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>p@1</th>
<th>p@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>w/o h</td>
<td>finetune</td>
<td>70.06</td>
<td>78.47</td>
</tr>
<tr>
<td>2</td>
<td>w/o h</td>
<td>prefix</td>
<td>69.90</td>
<td>78.25</td>
</tr>
</tbody>
</table>

Using latest clicked document

| 3.1 | \([p; q; h]\) | prefix | 68.92| 77.83| 72.10|
| 3.2 | PersonalTM | prefix | 70.01| 78.53| 73.01|

Table 3: Results with prefix adaptor.

Furthermore, we evaluate the model performance on the zero-shot set, whose queries are never seen in the past. The results are shown in Table 4. We found that DSI’s zero-shot performance is not ideal, which aligns with the observation in [32]. But by comparing line f with the rest, we achieve significant improvement by using PersonalTM and the relevant personal context.

<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>p@1</th>
<th>p@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>BM25 (with relevant history)</td>
<td>20.97</td>
<td>31.03</td>
<td>24.53</td>
</tr>
<tr>
<td>b</td>
<td>DualEncoder (T5)</td>
<td>6.66</td>
<td>17.81</td>
<td>9.90</td>
</tr>
<tr>
<td>c</td>
<td>DSI</td>
<td>8.14</td>
<td>16.76</td>
<td>10.71</td>
</tr>
<tr>
<td>d</td>
<td>DSI + HieLoss</td>
<td>11.45</td>
<td>23.39</td>
<td>15.14</td>
</tr>
<tr>
<td>e</td>
<td>DSI + HieLoss + UserIdentifier</td>
<td>15.17</td>
<td>28.20</td>
<td>19.21</td>
</tr>
<tr>
<td>f</td>
<td>PersonalTM (prefix adaptor)</td>
<td>37.19</td>
<td>56.94</td>
<td>43.76</td>
</tr>
</tbody>
</table>

Table 4: Evaluation on Zero-shot cases.

4 CONCLUSION

In this work, we propose a transformer memory framework for personalized retrieval, integrating two essential types of personalized information into the model. We propose a novel decoder architecture that effectively leverages personalized context without increasing trainable parameters, and apply a hierarchical loss function to optimize the performance. Our framework also incorporates the prefix adaptor mechanism to facilitate the learning of the interaction between the query and contextual personalized features. In the future, we will further enhance the design by incorporating incremental learning and enabling dynamic updates to overcome the limitations of frequent index updates and zero-shot retrieval.