GLAD: Graph-based Long-term Attentive Dynamic Memory for Sequential Recommendation

Deepanshu Pandey, Arindam Sarkar, and Prakash Mandayam Comar

Amazon Development Center, Bengaluru, India
{deepnsp,arindsar,prakasc}@amazon.com

Abstract. Recommender systems play a crucial role in the e-commerce stores, enabling customers to explore products and facilitating the discovery of relevant items. Typical recommender systems are built using n most recent user interactions, where value of n is chosen based on trade-off between incremental gains in performance and compute/memory costs associated with processing long sequences. State-of-the-art recommendation models like Transformers, based on attention mechanism, have quadratic computation complexity with respect to sequence length, thus limiting the length of past customer interactions to be considered for recommendations. Even with the availability of compute resources, it is crucial to design an algorithm that strikes delicate balance between long term and short term information in identifying relevant products for personalised recommendation. Towards this, we propose a novel extension of Memory Networks, a neural network architecture that harnesses external memory to encapsulate information present in lengthy sequential data. The use of memory networks in recommendation use-cases remains limited in practice owing to their high memory cost, large compute requirements and relatively large inference latency, which makes them prohibitively expensive for online stores with millions of users and products. To address these limitations, we propose a *novel* transformer-based sequential recommendation model GLAD, with external graph-based memory that *dynamically* scales user memory by adjusting the memory size according to the user's history, while facilitating the flow of information between users with similar interactions. We establish the efficacy of the proposed model by benchmarking on multiple public datasets as well as an industry dataset against state-of-the-art sequential recommendation baselines.

1 Introduction

On e-commerce stores with large product catalogs, personalized product recommendations are essential to improve the customers' journey by helping them easily find relevant products from millions of available items. Customer preferences are influenced by various factors such as seasonality, trends, personal experiences, social media etc. Thus, it is essential to effectively process the dynamic nature of product preferences, by clearly differentiating their immediate needs from their long term interests. Balancing the long-term and short-term considerations can be challenging because short-term interactions can be sporadic that change rapidly

and may not always be indicative of the user's true long term preferences. On the other hand, understanding long-term preferences require significant storage and processing resources to identify trends and patterns in the user's historical behavior. This is further complicated in situations where users have changing preferences/interests over time. Developing a recommendation system that can effectively balance both short-term and long-term considerations is an ongoing challenge in e-commerce recommendation systems [19,25,34].

State-of-the-art transformer based sequential recommendation models like Bert4Rec [32], CORE [12] and SASRec [13] are expensive to scale to long sequence lengths at inference time due to the quadratic nature of self-attention. Since the interaction history can be extremely long for some users, a common practice is to make the predictions based on the last n items in the user history, where n is a hyperparameter which depends on the resource availability and inference time constraints. However, this results in discarding of rich interaction data, especially for the highly active users with long histories [19,33,21]. To efficiently represent changing user preferences over time, and effectively combine long and short term preferences, we propose a *novel* sequential recommendation model, GLAD (Graph-based Long-Term Attentive Dynamic memory), based on Memory Networks [37,31], by introducing the concept of discretionary memory cells based on length of users' interaction history. Not only does the proposed model enhance quality of recommendations, but being modular, it provides a way to add dynamic memory as a component to any of the existing sequential models. where efficiently processing and remembering long term dependencies in sequential data is essential. The major contributions of our work are as follows:

- We propose a *novel* deep neural network model for sequential recommendation which leverages a *dynamic* graph-based external memory. The proposed approach has significantly less memory complexity compared conventional memory networks while maintaining low inference latency.
- We propose an *attentive* reading mechanism over graph memory conditioned on the target/query item to be scored by the model, which further enhances the quality of the recommendations.
- For real world applications, which are often memory constrained, we propose a *clustered* graph memory variant of our model which reduces memory consumption and also enables information flow between similar users.

2 Methodology

In this work, we focus on the task of next item prediction, where the goal is to predict the next item the user is likely to interact with based on the past history. Specifically, given a list of N_i items which a user U_i has interacted with, we want to rank a candidate set of target items $\{T_1, T_2, .., T_K\}$ on the basis of relevance to the user. The sequence of items interacted with by a given user is sorted based on the interaction time, and is then split into $d_i = \lceil \frac{N_i}{c} \rceil$ sub-sequences of maximum size c. We denote the m^{th} subsequence corresponding to the i^{th} user U_i as S_{im}. Intuitively, the sequence of items S_{im} captures short-term behaviour in a local

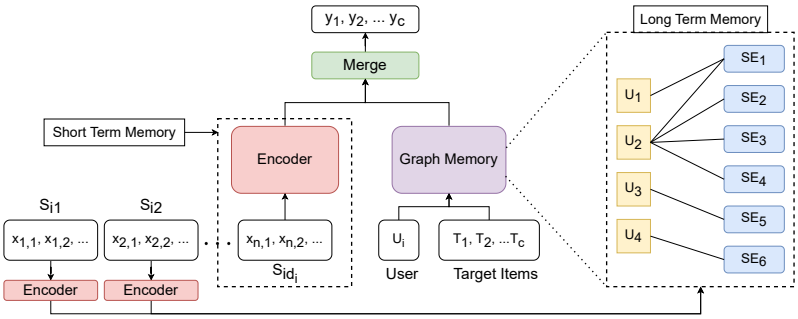


Fig. 1: The proposed GLAD model architecture.

temporal window, forming an abstract short term memory of product preferences. We then create a bi-partite graph structure between users and their subsequences. Corresponding to each subsequence S_{im}, we create a subsequence node in the graph. Since it is possible for multiple users to share exactly the same subsequence (especially when c is small), corresponding to each unique subsequence, we create a node in the graph SE_j independent of user index i. Note that the users with rich past interaction history will be connected to more subsequence nodes compared to users with sparse interactions. We use a self-attention based sequence encoder to generate embedding h_{SE_j} for each subsequence. These are used to initialize the embeddings of the corresponding sub-sequence nodes in the graph, which forms the *dynamic* long-term memory. Finally, the short-term (h_{SE_j}) and the long-term representations captured by a Graph Neural Network (GNN) over the graph-structured memory are fused via a gating mechanism to generate the final representation for the user. Proposed model architecture is presented in Fig. 1.

2.1 Interaction Sequence Encoder

We use Bert4Rec [32] as the backbone transformer model to process user interaction sequences, and refer to it as the Interaction Sequence Encoder. Since the interaction sequences for users can be quite large, we divide each sequence into smaller chunks (S_{im}) of equal lengths c (last subsequence is padded if required), and pass them independently through the encoder, which is pretrained on the Masked Item Prediction (*Cloze*)[32] task. The representation of the final chunk S_{id_i} for the user U_i from the trained model is denoted by z_{U_i} and can be thought of as a representation of short-term interaction preferences for the user, as it considers the last L items in the user's interaction sequence. Finally, the encoder is trained end-to-end with the graph-based memory to make the final recommendations (section 2.5). Note that, it is possible to replace Bert4Rec with similar models that can produce a sequence representation from interaction sequences.

2.2 Dynamic Graph Based Memory

To capture and retain long-term user preferences, we create an external memory component in the form of a user interaction graph. The memory is in the form of

a bipartite graph with two type of nodes: 1) user nodes U_i, and 2) sequence nodes SE_j. Each of the unique sub-sequence chunks S_{im} is given a distinct sequence node in the graph and connected to the corresponding user node, as shown in Figure 1. Each of the nodes is represented by a high-dimensional node embedding h_{SE_j}, and are initialised using the Interaction Sequence Encoder. Once the graph is initialised, a Graph Neural Network (GNN) is used to aggregate information into the user node embeddings from the neighbouring sequence nodes. This results in aggregation of all the historical interaction information stored for the user in their user node U_i, and the resulting user node embedding is then used as the user's long-term memory representation. For our experiments, GraphSAGE [6] architecture was used as GNN for the graph memory, where the representation of a graph layer is as follows:

$$h^{l+1}_{\mathcal{N}(U_i)} = agg(\{h^l_{SE_j}, \forall SE_j \in \mathcal{N}(U_i)\});\ h^{l+1}_{U_i} = \sigma(W^l_U\,(h^l_{U_i}||h^{l+1}_{\mathcal{N}(U_i)})) \quad (1)$$

Here, $h^{l+1}_{U_i}$ is the node embedding for user node U_i at $l+1^{th}$ layer, $\mathcal{N}(U_i)$ represents the set of neighbouring nodes of U_i and W^l_U is projection matrix for user nodes at layer l. $||$ represents the concatenation operator. The aggregate (agg) function used here is sum. Similarly for the sequence nodes:

$$h^{l+1}_{\mathcal{N}(SE_j)} = agg(\{h^l_{U_i}, \forall U_i \in \mathcal{N}(SE_j)\});\ h^{l+1}_{SE_j} = \sigma(W^l_{SE}\,(h^l_{SE_j}||h^{l+1}_{\mathcal{N}(SE_j)})) \quad (2)$$

Note that unlike conventional memory networks that allocate a fixed-size memory for each user[1,33], the graph-based memory model allocates memory based on the size of user interaction histories, which results in efficient use of physical memory. The choice of GraphSAGE for neighborhood aggregation is motivated by the fact that it takes into account both the neighborhood embeddings and the node embedding of the current node to generate node embeddings for the next layer. We believe that this is helpful in propagating the information from initial node embeddings to subsequent layers, since the initial representations for the nodes are rich and contain compressed information about user sequences. It is possible to replace GraphSage with other GNN architectures in the architecture, and we leave this as an area for future research and exploration.

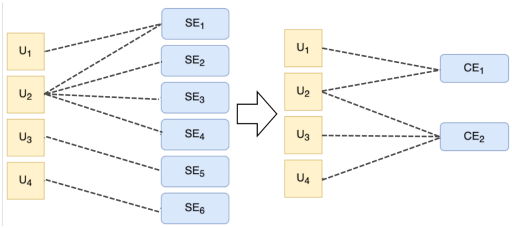


Fig. 2: Visualization of the compression of user-item interaction graph. Here CE_j are Cluster Embeddings. Semantically similar nodes (1,2,3), (4,5,6) are clustered.

Clustered Sequence Representations: Although the graph-based memory model is able to aggregate historical information for the user, there is little to no

information exchange between users since they are connected only to their own sub-sequence nodes. Further, the GNN captures less information for users with lesser interactions, since these users are connected to only a few sequence nodes. To solve these issues, we cluster similar sequence nodes together into cluster nodes. In the clustered memory model, the outgoing connections from user nodes to sequence nodes are now grouped to form connections to cluster nodes CE_j. Cluster node embeddings are computed by mean-pooling over sequence node embeddings. This connects the users who have interacted with similar sequences as *2*-hop neighbors, enabling the model to capture information from similar users via multi-hop message-passing. Further, it reduces the memory footprint of the model as now only the cluster embeddings need to be stored, rather than storing all the sequence embeddings for user interaction sequences. To accommodate the new user interactions and tackle the gradual drift in user preference and purchase patterns, several sequential clustering mechanisms [4,22] can be leveraged to update the clustered graph memory with new user interaction sequences.

2.3 Output Gating

To appropriately weigh the recent and historical interactions, a gating function is learned from data, and is used to combine the the short-term sequence embedding from the encoder and the long-term memory representation from the graph memory to generate the final representation for user U_i:

$$g = \sigma(W_T(z_{U_i}) + W_G(h^L_{U_i})) \tag{3}$$

where σ is the sigmoid function, $z_{U_i} \in R^e$ is the output of the encoder, and $(h^L_{U_i}) \in R^e$ is the final (L^{th}) GNN layer node representation for user node U_i. $W_T \in R^{e \times e}$ and $W_G \in R^{e \times e}$ are projections for the transformer and graph outputs respectively and e is the hidden dimension of the model. The final output of the model $h_{U_i} \in R^e$ is a weighted combination of the two outputs, aggregated via elementwise product ($\odot$) with the gating weights $g \in R^e$. h_{U_i} gathers all the relevant information for user U_i from both the transformer encoder and the graph memory. For each target item T_i in the candidate set, the relevance to the user U_i is computed by taking dot (.) product of the final representation h_{U_i} and the target item representation h_{T_k}, resulting in similarity score for item, s_k:

$$s_k = h_{U_i} \cdot h_{T_k}; \text{ where } h_{U_i} = g \odot z_{U_i} + (1 - g) \odot h^L_{U_i} \tag{4}$$

During inference, the target items are re-ranked based on these similarity scores and top items are chosen as the final recommendations for user U_i. Note that the candidate set of target items $\{T_1, T_2, ...\}$ can vary for different users, depending on the candidate generation [5,43] algorithm.

2.4 Attentive Reading of Graph Memory

The graph memory model aggregates the long term user preferences. However, for a given target item, some of the memory nodes in the user memory graph

might be more relevant than others. To selectively capture useful information, the graph memory is further enhanced by replacing the last layer of the GNN with an attention based aggregation of neighbourhood information. The aggregation of neighbourhood information for user node U_i in the final L^{th} layer is determined by the equation (for clustered graph, SE will be replaced CE):

$$\alpha_{SE_j} = (h_{T_k} \cdot h^{L-1}_{SE_j});\ h^{L}_{U_i T_k} = \sum_{SE_j \in N(U_i)} \alpha_{SE_j} h^{L-1}_{SE_j} \tag{5}$$

which replaces eq. 1 for final layer computation for user node U_i. Here, $N(U_i)$ represents the set of neighbouring nodes of node U_i and T_k is the k^{th} target item. The target conditioned attentive read operation is similar to the target attention proposed by [44], and enables our model to selectively read over the most relevant parts of the user history conditioned on the target item. The final output $h_{U_i T_k}$ gathers all the relevant information from the graph memory for user U_i with respect to target item T_k. Once the user node representation $h^{L}_{U_i T_k}$ is computed, it replaces $h^{L}_{U_i}$ in eq. 3, while eq. 4 is modified to:

$$s_k = h_{U_i T_k} \cdot h_{T_k}, \text{ where}, h_{U_i T_k} = g \odot z_{U_i} + (1 - g) \odot h^{L}_{U_i T_k} \tag{6}$$

2.5 End-to-End Training

We train our model in two phases. In *Phase 1*, we pretrain the interaction sequence encoder on Masked Item Prediction task, with the sequence chunks S_{im}'s. We then create the graph-based external memory with user and sequence nodes. We pass each of the training chunks, with a mask token concatenated at the end through the trained interaction sequence encoder and use the final layer transformer output for the mask token as the sequence representation. We initialise the sequence node embeddings in the graph with their corresponding sequence representations while the user node embeddings are initialised with random vectors. For clustered graph, the sequence nodes are first clustered together using k-means clustering, and the cluster nodes are stored in the graph instead of sequence nodes.

In *Phase 2*, we combine representations from the transformer encoder and the graph memory using the gating layer and train the entire model end-to-end. For each sequence chunk for a user U_i, we capture the transformer sequence representation and consider this to be the short term user preference. We then sample a sub-graph containing the user node for U_i from the graph memory, along with all its 2-hop neighbors and compute the user node representation for U_i using the GNN, which is used as the long-term memory component for the user. The model is trained using the cross-entropy loss given by:

$$\mathcal{L}_{CE} = -\sum_{k=0}^{K} y_k \log p_k, \text{ where, } p_k = \frac{e^{s_k}}{\sum_{m=0}^{K} e^{s_m}} \tag{7}$$

where T is the set of candidate items to be ranked, and |T| = K. p_k is the probability score generated from the similarity scores s_k given by the model for the k^{th} target item, and $y_k \in \{0, 1\}$ denotes the actual label for the item.

3 Related Work

The early years of recommendation systems saw extensive use of Collaborative Filtering (CF) [30,16] methods. Matrix Factorization [17], used inner product between user and item representations to model the user-item interactions. This was followed by neural approaches like Neural Collaborative Filtering (NCF) [7]. However, these models fail to capture the sequential nature of user interactions.

Sequential Recommendation: Early Markov Chain based models like FPMC [26] showed that a user interaction is closely related to the last interactions made by the user, but lacked the capability to utilise long user histories. Following the success of RNNs [28,11,3] in NLP tasks, models like GRU4Rec [10,9] leveraged RNNs for next-item prediction, but suffered from non-parallelizable operations and catastrophic forgetting over long sequences. The use of attention mechanism in later models like NARM [18], STAMP [19] etc., managed to successfully alleviate some of these issues. Models like SRGNN[38] and GCSAN [39] additionally leveraged information from graph based structures constructed from user sessions, but are difficult to scale to large datasets due to item graph construction for each user. Further improvements followed with the introduction of Transformer [35] based models. Engrained with the powerful self-attention mechanism to process sequential data, models like SASRec [13], Bert4Rec [32], CORE [12] have become the go-to approach for building sequential recommendation systems.

Memory Networks: While sequential models effectively capture the short-term user preferences, they often lack in capturing the long-term user interest due to their inability to process very long user histories. To tackle this problem, the idea of having a dedicated memory has been proposed. Memory Networks [37,31] with dedicated external storage are capable of explicitly storing historical information, and have read, write and update operations defined for the memory store. User-Memory Networks (RUM) [1] proposed a dedicated external memory per user for more personalised recommendations. MA-GNN [21] uses a graph-based structure as short-term memory, and uses a shared memory network for long term memory. HPMN [25] proposes hierarchical memory units per user to capture multiscale user behavior for lifelong learning. In DMAN [33], the authors build on the idea of memory networks and couple a sequential recommendation model with conventional user memory, thus leveraging both long and short-term user preference information. While the user-specific memory cells intuitively capture more personalized patterns per user, a major drawback of memory networks is that the memory requirement can be prohibitively large, and poor scalability with increasing number of users, as they allocate fixed size memory matrices per user. Naturally, this either results in excessive compression of memory for users with long histories, and/or wastage of memory space for users having short histories. In contrast to the existing approaches, the proposed model GLAD leverages state-of-art transformer encoder network to capture short term user preferences, and efficiently maintains long term user interest representation by allocating differential memory space to the users based on their interaction history size via a *novel* graph-based *dynamic memory*.

4 Experiments

We establish the efficacy of the proposed model by performing extensive quantitative and qualitative analyses. We benchmark our model against representative baseline models on both public and industry datasets to answer multiple research questions (RQ) on predictive performance and incremental benefit of each component, and address them by performing extensive experimentation.

Datasets: We use the publicly available Amazon Reviews datasets, which have been a popular benchmark for recommendation system models [1,14,44,32]. Specifically, we use the Electronics and Home & Kitchen categories [23] from the Amazon Reviews dataset for our experiments. We also experiment with a private dataset (hereby referred to as E-COM dataset) created by sampling anonymized user interaction logs over a fixed period of time for an e-commerce site. Summary statistics for the datasets are: (1) **E-COM**: 1.2M users, 2.2M items, 39.6M interactions, (2) **Electronics**: 204K users, 247k items, 20M interactions, and (3) **Home & Kitchen**: 207k users, 310k items, 22M interactions. The maximum number of interactions for E-COM, Electronics and Home & Kitchen are 8919, 592 and 446, while the median values are 20, 13 and 13 respectively.

Baselines: We compare our model against 8 competitive baselines (Table 1). We have included representative models with memory component: **FRUM** and **STAMP**, RNN/Transformer based models: **GRU4Rec**, **Bert4Rec**, **CORE**, **CL4SRec** (the latter two additionally utilise contrastive learning), and graph based models: **SRGNN** and **GCSAN**. Among the baselines, only FRUM explicitly captures the long-term user context by using external memory, but it adds to the memory footprint due to a large fixed size memory per user, unlike our model which is endowed with a dynamic graph-based memory. Also, due to the sequential nature of memory updates in FRUM, the training and inference latency is high (Table 5). In contrast, for GLAD, memory updates are simply edge updates in the memory graph. Models like Bert4Rec, CORE and CL4SRec, lack an efficient mechanism to represent the long term user preferences. Among graph models, both SRGNN and GCSAN create a separate graph per user, based on the session data available for them, which is expensive over long sequences. We propose an efficient long term preference representation via a subsequence graph instead. We have omitted early works like Collaborative Filtering [27,8] and models like SASRec [14] and Caser [34] since the newer models like Bert4Rec, GCSAN etc. outperform them, and some of the larger memory networks like DMAN due to their extremely large memory and compute requirements.

Experimental Setup: We compare the performance of the models on the popular task of next-item prediction [8,14,34]. The per-user interactions are sorted by the timestamp and converted into sequences. The last item from each user sequence is retained for the test data, second last item for validation and the rest of the sequence is used for training the models. Since the training sequences can be very large for some of the users, we further divide these into smaller, equal-sized chunks. For training, the last item in the sequence chunks is considered to be ground truth and is paired with a set of 100 negatively sampled items to be ranked and the top-k items with highest scores are taken as recommendations

from a model. For the recommendation task, we only use item ID as feature. We report Hit-Rate (hit@k) which measures the proportion of users for whom the item they interacted with was in the top-k predictions by the model, and Mean Reciprocal Rank (mrr@k), which additionally quantifies if the model is able to rank the interacted item towards the top of the list. For all results on the ECOM dataset, metrics are reported as relative gains with respect to the FRUM baseline to maintain dataset anonymity.

Sequence Masking: We pre-train the Bert4Rec backbone on *Cloze* task. For GLAD, the last item in each of the chunks is masked and the chunks are passed sequentially through the model, where the model tries to predict the masked item correctly. During validation and testing, for both the models, only the last chunk for each of the users is passed after adding a mask token at the end of the sequence. The sequence masking process is as follows:

Training sequence: [$x_1, x_2, ..., x_k, x_{k+1}, ..., x_{2k}, x_{2k+1}, x_{2k+2}$]

Pretraining (Bert4Rec):
Sub-sequence 1 = [$x_1, \{mask\}, x3..., \{mask\}, x_k$]
Target 1 = [x_2, x_{k-1}]
Sub-sequence 2 = [$x_{k+1}, x_{k+2}, \{mask\}, ..., x_{2k-1}, \{mask\}$]
Target 2 = [x_{k+3}, x_{2k}]

Training (GLAD):
Sub-sequence 1 = [$x_1, x_2, ..., x_{k-1}, \{mask\}$]
Target 1 = [x_k]
Sub-sequence 2 = [$x_{k+1}, x_{k+2}, , x_{2k-1}, \{mask\}$]
Target 2 = [x_{2k}]

Validation: [$x_{k+2}, x_{k+3}, ..., x_{2k}, \{mask\}$]
Targets: [x_{2k+1}]

Test: [$x_{k+3}, x_{k+4}, ..., x_{2k+1}, \{mask\}$]
Targets: [x_{2k+2}]

Implementation Details: We use a 2-layer Bert4Rec model as the backbone sequence encoder along with a 2-layer GraphSAGE network as GNN for the graph memory. The embedding dimension size e is chosen as 256. The chunk size is set to 50 for E-COM and 20 for Electronics and Home & Kitchen. We use k-means clustering to create the set of clusters for the clustered graph memory. We choose a clustering factor of 100 for E-COM dataset, which means approximately 100 nodes are clustered together to form one cluster node. Since both Electronics and Home & Kitchen datasets are smaller in size, we do not cluster the graph memory for these datasets. The model is trained with a Cross Entropy loss with Adam [15] optimiser, Cosine Annealing Scheduler [20] and a negative sampling strategy. For hyperparameter tuning of all the models (including our model), we

start with the optimal hyperparameters suggested by the authors, and leveraged a mix of manual tuning and grid search. Except for the FRUM model, we use Recbole [42,41,40] implementations of the baselines. Note that for some of the models, like Bert4Rec, the choice of hyperparameters and the number of training epochs is critical (as was also observed in [24]). Accordingly, to address these shortcomings of Bert4Rec implementation in Recbole, we use a custom Transformer implementation on top of the RecBole training framework. We perform all our experiments on a distributed setup of 4 Nvidia V100 GPUs. We use Deep Graph Library (DGL)[36] for implementation of the graph component in our model. In our graph implementation, each of the operations are batched to ensure maximum training efficiency. Since processing entire graph at once is expensive, while creating user node representations from the graph memory, we sample all the users in the batch along with their 1-hop and 2-hop neighbors to create a sub-graph and apply GNN over the subgraph to get user embeddings.

Table 1: hit@k and mrr@k metrics on next-item prediction task. For E-COM, we report relative gains with respect to FRUM baseline for dataset anonymity. Abbreviations used: E-COM (EC), Electronics (EL), Home & Kitchen (H&C).

		FRUM	GRU4Rec	STAMP	SRGNN	Bert4Rec	CL4SRec	GCSAN	CORE	**GLAD**
EC	hit@1	—	0.0693	0.1277	0.1325	0.1523	0.1413	0.1441	0.1548	**0.1609**
	hit@5	—	0.1335	0.1694	0.1740	0.1767	0.1824	0.1847	0.1780	**0.1889**
	hit@10	—	0.1491	0.1705	0.1770	0.1737	0.1844	0.1856	0.1857	**0.1965**
	mrr@10	—	0.0939	0.1426	0.1473	0.1555	0.1561	0.1586	0.1631	**0.1727**
EL	hit@1	0.1728	0.2054	0.2485	0.2264	0.2524	0.2350	0.2347	0.2073	**0.3639**
	hit@5	0.3438	0.4559	0.4912	0.4783	0.5020	0.4840	0.4862	0.4347	**0.5714**
	hit@10	0.4207	0.5786	0.5949	0.5874	0.6142	0.5939	0.5977	0.5471	**0.6665**
	mrr@10	0.1719	0.3123	0.3509	0.3335	0.3594	0.3410	0.3416	0.3047	**0.4525**
H&C	hit@1	0.1569	0.1845	0.2439	0.2018	0.2056	0.2013	0.2029	0.2098	**0.3426**
	hit@5	0.3098	0.4270	0.4337	0.4362	0.4376	0.4369	0.4387	0.4348	**0.5118**
	hit@10	0.3981	0.5236	0.5435	0.5467	0.5446	0.5476	0.5491	0.5493	**0.5888**
	mrr@10	0.1590	0.2886	0.3014	0.3020	0.3049	0.3017	0.3036	0.3051	**0.4145**

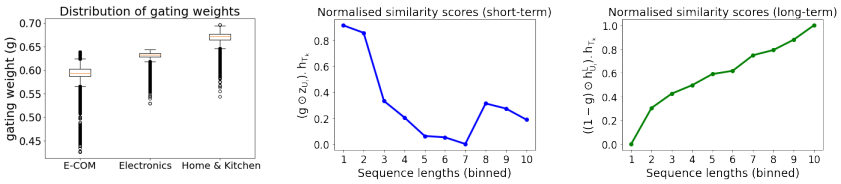
4.1 Discussion

The performance metrics for our proposed model along with popular sequential recommendation baselines are summarized in Table 1. In this section, we discuss the results in detail, and examine various qualitative aspects of the model.

RQ1. *How does the proposed model (GLAD) perform in comparison to the representative sequential recommendation models?*

It can be observed from Table 1, that our model outperforms the state of the art baselines by a significant margin. We observe clear gains across all metrics for the proposed GLAD model over the existing baselines. As expected, among the other models, it can be seen that transformer based architectures perform better in general compared to the memory/attention only models. The observed gains of the proposed GLAD over other baselines can be attributed to: 1) incorporating

the graph structure, which helps our model learn from similar users, 2) target based attention mechanism (eq. 5) that selectively attends to node-representations from long-term memory optimally, and 3) the gating mechanism (eq. 6) that strikes a delicate balance between the long term and short term component. On the ECOM dataset, our model outperforms the closest baseline by 0.32% - 1.18% on hit-rate@k. For large industry scale datasets, this is a significant improvement, given the millions of impressions on daily basis. For instance, with an average purchase value of 100 USD [2], a conversion-rate of ∼0.2%, and attribution of conversion to recommendation of ∼25% [29], 1.18% improvement in hit@10 roughly translates to incremental gain of USD 600 per million impressions.



(a) (b) (c)

Fig. 3: Visualisation of gating weights. Here, (a) Distribution of gating weights, (b) Normalised similarity scores of encoder representation, and (c) Normalised similarity scores of graph memory with increasing sequence lengths.

RQ2. *Does the proposed dynamic graph-based memory effectively leverage long-term dependencies in user interactions?*

We analyse the performance of GLAD model with graph memory on different segments of users (top x% in terms of sequence lengths) based on the overall number of interactions (Table 2). It can be observed that not only does the addition of graph based memory gives consistent improvements over the backbone model (Bert4Rec), the performance gains are greater for users with larger number of interactions. This reinforces the argument that the graph based memory component is able to efficiently retrieve long term user preferences to make relevant recommendations. In order to understand the model's behavior with respect to users with different sequence lengths, we plot and analyse the gating weights g. About $35\% - 50\%$ weight is given to long term memory highlighting the need for using memory network to gain additional performance improvements (Fig. 3a). To gain further insights into contribution of short term vs. long term memory component, we calculate the dot-product similarity scores for these components with the target items, and plot the similarity scores. To maintain dataset anonymity, we do not disclose the actual sequence lengths. Sequences are binned into equidistant ordinal bins based on sequence lengths, and we report the average similarity scores (normalised) for each bin. These are calculated as $(g \odot z_{U_i}).h_{T_k}$ for interaction encoder output (Fig. 3b) and $((1-g) \odot h^L_{U_i}).h_{T_k}$ for the graph memory component (Fig. 3c). It can be observed that moving from users with smaller interactions histories to users with longer histories, the impact

Table 2: Relative gain of the GLAD model over Bert4Rec for users with varying length of historical interactions on the ECOM dataset. Sequence length is given as top % of the population for dataset anonymity.

Seq. len	Δhit@10	Δmrr@10
top 0.05%	**+3.38%**	**+4.51%**
top 1%	**+3.87%**	**+2.73%**
top 5%	**+4.47%**	**+3.46%**
top 15%	+2.31%	+1.17%
top 50%	+1.03%	-0.26%
all users	+1.73%	+0.39%

Table 3: Impact of different model components of the proposed model. Results on ECOM dataset are relative to FRUM baseline.

Dataset	Model	hit@10	mrr@10
E-COM	Bert4Rec	0.1737	0.1555
	GLAD(G)	0.1749	0.1427
	GLAD(G+C)	0.1878	0.1594
	GLAD(G+C+A)	**0.1965**	**0.1727**
Electronics	Bert4Rec	0.6142	0.3594
	GLAD(G)	0.6107	0.3608
	GLAD(G+A)	**0.6665**	**0.4525**
Home & Kitchen	Bert4Rec	0.5446	0.3049
	GLAD(G)	0.5653	0.3230
	GLAD(G+A)	**0.5888**	**0.4145**

of graph memory increases, while the contribution of short-term representation from the encoder keeps decreasing, which shows that the proposed graph structure effectively leverages long-term user preferences for users with long histories.

RQ3. *What is the incremental benefit of various model components in improving the quality of the final recommendations?*

We experiment with four different model variants and report the incremental gains in performance in Table 3. All the models use Bert4Rec as the interaction sequence encoder. GLAD(G) represents the proposed model with graph memory without clustering, GLAD(C) represents the clustered version of this model and GLAD(C+A) represents the final clustered model with target conditional attention. It can be observed that the variant with clustering and target attention model performs the best highlighting the importance of higher order information sharing between different users, and selective memory read operations.

RQ4. *Is the proposed graph-memory a generic memory module that can be used with other sequential recommendation models?*

To establish the proposed memory component as a general extension which can be used with any of the sequential recommendation models, we experiment with different backbone models and tabulate the results in Table 4. It can be observed that addition of graph based memory results in a significant boost in the performance of the backbone models in both the cases.

Table 4: Performance of **G**LAD with different backbones (G+) on the E-COM dataset.

Model	hit@5	hit@10	mrr@10
STAMP	0.1694	0.1705	0.1425
G+STAMP	**0.1852**	**0.1871**	**0.1559**
Bert4Rec	0.1767	0.1737	0.1555
G+Bert4Rec	**0.1889**	**0.1965**	**0.1727**

Table 5: Comparison of inference latency of representative models. Results correspond to a set of ∼1MM customers.

Time	FRUM	GRU4Rec	Bert4Rec	GCSAN	GLAD
Total	7500s	5520s	1260s	1500s	2025s
Avg.	7.14 ms	5.25ms	1.19ms	1.42ms	1.92ms

RQ5. *Is the model suitable for deployment for low latency applications?*

While complex models bring in higher capacity, it is important for a model to be within acceptable latency bounds for good user experience in real world recommendation systems. Table 5 shows the inference latency comparison of our model with the other baselines on sampled E-COM dataset. It can be observed that unlike conventional memory networks, the increase in inference time due to proposed memory component is insignificant relative to the baseline models.

RQ6. *How does the memory footprint of GLAD compare against traditional memory networks?*

We estimate and compare the memory requirements of our proposed model for the memory component with that of two representative memory networks.

$$Mem(GLAD) = N_{seq} \times e/c_f + N_U \times e \tag{8}$$

Here, c_f is the clustering factor (100 here), and N_{seq} is the number of sequences. N_{seq} can be re-written as $l_{seq} \times N_U$, where l_{seq} is the average number of sequence nodes connected to a user, which is approximately 1.2 for the E-COM dataset (chunk-size = 50), and N_U represents the number of users. The effective number of parameters thus roughly reduces to:

$$Mem(GLAD) = l_{seq} \times N_U \times e/c_f + N_U \times e = 1.01 \times N_U \times e \tag{9}$$

For FRUM model with f memory slots , the relative memory requirement is:

$$Mem(FRUM) = f \times N_U \times e \approx f \times Mem(GLAD) \tag{10}$$

Whereas, the number of parameters in the memory component for an L layered DMAN model is:

$$Mem(DMAN) = L \times f \times N_U \times e \approx f \times L \times Mem(GLAD) \tag{11}$$

5 Conclusion

In this work, we present a novel sequential model for personalized recommendations. The proposed architecture encodes the most recent interactions of the user to form a highly predictive short-term memory representation, and at the same time has the ability to query a larger set of past interactions through a shared graph-based memory formed using past user interaction subsequences. The graph component allocates memory for the user based on the size of the interaction history, thus making efficient use of the available memory, without having noticeable increase in the inference time. We improve the scalability of the model via clustering the graph nodes, resulting in further reduction in the number of sequence nodes to be stored for the graphs while enabling information flow between similar users. Finally, we propose a target conditioned read operation on the clustered graph memory for more accurate memory retrieval. The proposed model outperforms representative sequential recommendation baselines. Moreover, the graph-based memory is modular by design, and can be added to any sequential recommendation model to act as a store for historical interactions.

References

1. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (2018)
2. Chevalier, S.: U.S. online shopping order value by device 2022 | Statista — statista.com. https://www.statista.com/statistics/439516/us-online-shopping-order-values-by-device/ (2022)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)
4. Cook, P.: Sequential k-means (2008), https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/C/sk_means.htm
5. Covington, P., Adams, J.K., Sargin, E.: Deep neural networks for youtube recommendations. Proceedings of the 10th ACM Conference on Recommender Systems (2016)
6. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. p. 173–182. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2017). https://doi.org/10.1145/3038912.3052569, https://doi.org/10.1145/3038912.3052569
8. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. Proceedings of the 26th International Conference on World Wide Web (2017)
9. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. p. 843–852. CIKM '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3271761, https://doi.org/10.1145/3269206.3271761
10. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. CoRR **abs/1511.06939** (2016)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**, 1735–1780 (1997)
12. Hou, Y., Hu, B., Zhang, Z., Zhao, W.X.: Core: Simple and effective session-based recommendation within consistent representation space. Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022)
13. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM). pp. 197–206 (2018). https://doi.org/10.1109/ICDM.2018.00035
14. Kang, W., McAuley, J.J.: Self-attentive sequential recommendation. CoRR **abs/1808.09781** (2018), http://arxiv.org/abs/1808.09781
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
16. Koren, Y., Bell, R.: Advances in Collaborative Filtering, pp. 77–118. Springer US, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_3, https://doi.org/10.1007/978-1-4899-7637-6_3
17. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009). https://doi.org/10.1109/MC.2009.263

18. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. p. 1419–1428. CIKM '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132847.3132926, https://doi.org/10.1145/3132847.3132926
19. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: Stamp: Short-term attention/memory priority model for session-based recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and amp; Data Mining. p. 1831–1839. KDD '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3219819.3219950, https://doi.org/10.1145/3219819.3219950
20. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=Skq89Scxx
21. Ma, C., Ma, L., Zhang, Y., Sun, J., Liu, X., Coates, M.: Memory augmented graph neural networks for sequential recommendation. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 5045–5052 (Apr 2020). https://doi.org/10.1609/aaai.v34i04.5945, https://ojs.aaai.org/index.php/AAAI/article/view/5945
22. MacQueen, J.: Some methods for classification and analysis of multivariate observations (1967)
23. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 188–197. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1018, https://aclanthology.org/D19-1018
24. Petrov, A., Macdonald, C.: A systematic review and replicability study of bert4rec for sequential recommendation. In: Proceedings of the 16th ACM Conference on Recommender Systems. p. 436–447. RecSys '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3523227.3548487, https://doi.org/10.1145/3523227.3548487
25. Ren, K., Qin, J., Fang, Y., Zhang, W., Zheng, L., Bian, W., Zhou, G., Xu, J., Yu, Y., Zhu, X., Gai, K.: Lifelong sequential modeling with personalized memorization for user response prediction. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 565–574. SIGIR'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3331184.3331230, https://doi.org/10.1145/3331184.3331230
26. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. pp. 811–820 (2010). https://doi.org/10.1145/1772690.1772773
27. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: The Web Conference (2001)
28. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**, 2673–2681 (1997)
29. Sharma, A., Hofman, J.M., Watts, D.J.: Estimating the causal impact of recommendation systems from observational data. In: Roughgarden, T., Feldman, M., Schwarz, M. (eds.) Proceedings of the Sixteenth ACM Conference on Economics

and Computation, EC '15, Portland, OR, USA, June 15-19, 2015. pp. 453–470. ACM (2015). https://doi.org/10.1145/2764468.2764488, https://doi.org/10.1145/2764468.2764488
30. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. in Artif. Intell. **2009** (jan 2009). https://doi.org/10.1155/2009/421425, https://doi.org/10.1155/2009/421425
31. Sukhbaatar, S., Szlam, A.D., Weston, J., Fergus, R.: End-to-end memory networks. In: NIPS (2015)
32. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. p. 1441–1450. CIKM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3357384.3357895, https://doi.org/10.1145/3357384.3357895
33. Tan, Q., Zhang, J., Liu, N., Huang, X., Yang, H., Zhou, J., Hu, X.: Dynamic memory based attention network for sequential recommendation. In: AAAI (2021)
34. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (2018)
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
36. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., Zhang, Z.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv: Learning (2019)
37. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1410.3916
38. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 346–353 (Jul 2019). https://doi.org/10.1609/aaai.v33i01.3301346, https://ojs.aaai.org/index.php/AAAI/article/view/3804
39. Xu, C., Zhao, P., Liu, Y., Sheng, V.S., Xu, J., Zhuang, F., Fang, J., Zhou, X.: Graph contextualized self-attention network for session-based recommendation. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. p. 3940–3946. IJCAI'19, AAAI Press (2019)
40. Xu, L., Tian, Z., Zhang, G., Wang, L., Zhang, J., Zheng, B., Li, Y., Hou, Y., Pan, X., Chen, Y., Zhao, W.X., Chen, X., Wen, J.R.: Recent advances in recbole: Extensions with more practical considerations (2022)
41. Zhao, W.X., Hou, Y., Pan, X., Yang, C., Zhang, Z., Lin, Z., Zhang, J., Bian, S., Tang, J., Sun, W., et al.: Recbole 2.0: Towards a more up-to-date recommendation library. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. pp. 4722–4726 (2022)
42. Zhao, W.X., Mu, S., Hou, Y., Lin, Z., Chen, Y., Pan, X., Li, K., Lu, Y., Wang, H., Tian, C., Min, Y., Feng, Z., Fan, X., Chen, X., Wang, P., Ji, W., Li, Y., Wang, X.,

Wen, J.: Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In: CIKM. pp. 4653–4664. ACM (2021)
43. Zhou, C., Ma, J., Zhang, J., Zhou, J., Yang, H.: Contrastive learning for debiased candidate generation in large-scale recommender systems. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (2020)
44. Zhou, G., Song, C.N., Zhu, X., Ma, X., Yan, Y., Dai, X.W., Zhu, H., Jin, J., Li, H., Gai, K.: Deep interest network for click-through rate prediction. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2017)