Query-Aware Sequential Recommendation

Zhankui He
UC San Diego
zhh004@eng.ucsd.edu

Handong Zhao*
Adobe Research
hazhao@adobe.com

Zhaowen Wang
Adobe Research
zhawang@adobe.com

Zhe Lin
Adobe Research
zlin@adobe.com

Ajinkya Kale
Adobe Research
akale@adobe.com

Julian McAuley
UC San Diego
jmcauley@eng.ucsd.edu

ABSTRACT

Sequential recommenders aim to capture users' dynamic interests from their historical action sequences, but remain challenging due to data sparsity issues, as well as the noisy and complex relationships among items in a sequence. Several approaches have sought to alleviate these issues using *side-information*, such as item content (e.g., images), action types (e.g., click, purchase). While useful, we argue one of the main contextual signals is largely ignored—namely users' *queries*. When users browse and consume products (e.g., music, movies), their sequential interactions are usually a combination of queries, clicks (etc.). Most interaction datasets discard queries, and corresponding methods simply model sequential behaviors over items and thus ignore this critical *context* of user interactions.

In this work, we argue that user queries should be an important contextual cue for sequential recommendation. First, we propose a new *query-aware* sequential recommendation setting, i.e. incorporating explicit user queries to model users' intent. Next, we propose a model, namely *Query-SeqRec*, to (1) incorporate query information into user behavior sequences; and (2) improve model generalization ability using query-item co-occurrence information. Last, we demonstrate the effectiveness of incorporating query features in sequential recommendation on three datasets.[1]

CCS CONCEPTS

• **Information systems** → **Personalization**.

KEYWORDS

personalization, sequential recommendation

ACM Reference Format:
Zhankui He, Handong Zhao, Zhaowen Wang, Zhe Lin, Ajinkya Kale, and Julian McAuley. 2022. Query-Aware Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557677

*Correspondence to Handong Zhao <hazhao@adobe.com>.
[1]Query-SeqRec code is released in https://github.com/AaronHeee/Query-SeqRec.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557677

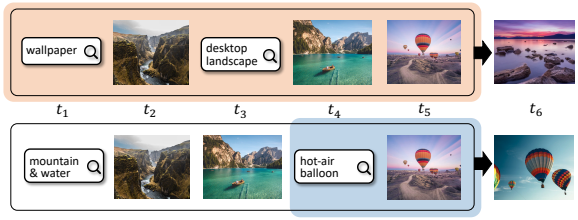

Figure 1: Motivating Examples. Given the same *item* sequence with different users' *queries*, the recommendation results are different, and 'boundaries' of useful historical interactions (shaded) also differ.

1 INTRODUCTION

Sequential recommender systems play an essential role in personalized online services (e.g. e-commerce, streaming media) on the basis of users' historical action sequences, but extracting relevant and accurate signals remains challenging; for example, user intent may gradually evolve—or change suddenly—leading to an erosion of the sequential context among items.

To mitigate such problems and capture users' intent among complex and noisy behavior sequences, various methods have been proposed that target different aspects, including model architectures [6, 9, 20] and side information [11, 12, 14]. Various side information has been exploited, such as item *content* [7, 12, 24] (e.g. reviews, images), user *action* types [14] (e.g. clicks, downloads, purchases) as well as *temporal* information [1, 11] (e.g. time intervals).

Despite the success of models that leverage rich side-information, some important signals remain under-explored. In this paper, we are specifically interested in users' *queries* that punctuate their interaction sequences. In many recommendation scenarios (e.g. e-commerce, music, photo-sharing), users interact with the system by alternately posing queries and browsing relevant items. However, such informative signals are usually discarded from sequential recommendation datasets, as demonstrated in Figure 1. We seek to investigate the use of explicit queries in sequential recommendation settings, namely *query-aware sequential recommendation.*

Queries can be an important contextual clue to reflect and predict users' evolving intent. The benefits of using queries are three-fold: (1) Queries reflect intent *granularity*. For example, Figure 1 shows queries such as 'wallpaper', 'hot-air balloon' suggest not only the recommendation target but also the desire for content diversity in a particular context. (2) Queries provide connections among interactions, which can be used to enrich item representations, especially for items that are rarely interacted with. (3) Queries help to detect user intent 'boundaries'. Figure 1 shows that a query of 'wallpaper'

followed by 'desktop landscape' may indicate a refinement of interests, whereas 'mountain & water' followed by 'hot-air balloon' would indicate unrelated intent; both scenarios have different semantics in terms of how we should regard relationships among sequential interactions (e.g. clicks or purchases).

In this work, we argue user queries should be considered in sequential recommendation, and propose a model for the *query-aware* sequential recommendation setting. First, we organize query and item information as heterogeneous query- and item-sequences. Second, we use query-item co-occurrence to improve the model generalization ability via *graph-based sequence augmentation*. Furthermore, we show how to technically handle large item embedding tables (e.g. 10 million items) in model training, at a scale rarely discussed in sequential recommendation papers. Our main contributions are summarized as follows:

- We propose a new *query-aware* sequential recommendation setting, i.e. incorporating explicit user queries as an important contextual cue to reflect and predict user intent.
- We propose a query-aware sequential recommender *Query-SeqRec* using *heterogeneous* user sequences and *graph-based sequence augmentation*. We also introduce a self-attentive model under this framework.
- We consider two existing datasets[2] for the new *query-aware sequential recommendation* setting, and use a new industrial dataset with millions of items. Experiments show the impact of incorporating explicit user queries and how our method outperforms state-of-the-art baselines.

2 METHODOLOGY

2.1 Input Sequence Formulation

Conventional Item Sequences. In conventional sequential recommendation, we are given a user set $\mathcal{U}$, an item set $\mathcal{I}$, and a set of user *item interaction* sequences $\mathcal{S} = \{S_1, \dots, S_{|\mathcal{U}|}\}$. Each sequence S_u consists of user u's (chronologically ordered) item interactions:

$$S_u = \left[i_1^{(u)}, i_2^{(u)}, \dots, i_{T_u}^{(u)} \right], \tag{1}$$

where $S_u \in \mathcal{S}$, $u \in \mathcal{U}$. $i_t^{(u)} \in \mathcal{I}$ is the item that the user clicked at the timestep t. T_u is the sequence length.

Query-aware Heterogeneous Sequence. We consider user queries by introducing an additional query set $\mathcal{Q}$ and word vocabulary $\mathcal{V}$, where a query $q \in \mathcal{Q}$ consists of a list of words $[v_1, \dots, v_{|q|}]$, $v \in \mathcal{V}$. We enrich the sequence S_u to a heterogeneous sequence $\hat{S}_u$, containing user u's queries and item interactions in chronological order:

$$\hat{S}_u = \left[\hat{s}_1^{(u)}, \hat{s}_2^{(u)}, \dots, \hat{s}_{\hat{T}_u}^{(u)} \right], \tag{2}$$

where $\hat{T}_u$ is the length of this query-aware heterogeneous sequence. $\hat{s}_t^{(u)}$ can be an item interaction or a query action. We use δ to indicate whether $\hat{s}_t^{(u)}$ at t-th step is a query or an item interaction:

$$\hat{s}_t^{(u)} \in \begin{cases} \mathcal{I}, & \text{if } \delta(\hat{s}_t^{(u)}) = 0, \\ \mathcal{Q}, & \text{otherwise.} \end{cases} \tag{3}$$

[2]For these datasets, the queries and clicks are collected for broader applications, but the user queries are discarded in conventional sequential recommendation settings.

We also examine other ways [13, 24] to incorporate user queries into item interaction sequences in our empirical studies (see Section 3.3).

Recommendation Goal. Given the query-aware heterogeneous sequence $\hat{S}_u$, the model predicts the next item for user u, which is formalized as modeling the probability over all possible items for this user's next item interaction, i.e.:

$$P\left(\hat{s}_{\hat{T}_u+1}^{(u)} = i^* \mid \hat{S}_u,\ \delta(\hat{s}_{\hat{T}_u+1}^{(u)}) = 0 \right). \tag{4}$$

$\delta(\hat{s}_{\hat{T}_u+1}^{(u)}) = 0$ assumes the next step is an item interaction. $\hat{s}_{\hat{T}_u+1}^{(u)} = i^*$ denotes that $i^* \in \mathcal{I}$ is the item the user interacts with at step $\hat{T}_u + 1$. We omit the user identifier u to simplify notation below.

2.2 Graph-Based Sequence Augmentation

Query-item co-occurrence is unique information unavailable in conventional sequential recommendation, which can be used to construct a query-item graph and improve model generalization ability via input sequence augmentation. Our intuition is a sequence $\hat{S}$ can be augmented as K sequences $\hat{S}^{(1)}, \dots, \hat{S}^{(K)}$ by stochastically replacing item i (query q) with similar items (queries), where query-item co-occurrence provides hints as to semantic similarities.

Graph Construction. We denote the query-item graph as $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A} = \mathcal{Q} \cup \mathcal{I}$ represents item and query nodes. The edge set $\mathcal{E}$ denotes all linkages between queries and items (i.e., (q, i) or $(i, q) \in \mathcal{E}$). We define the neighbors of item i as $\mathcal{N}(i) = \{q \mid (i, q) \in \mathcal{E}\}$, and define $\mathcal{N}(q)$ similarly. The initial edge set $\mathcal{E}_1$ is constructed by connecting item i with its latest query q. Then, to further reduce noises and retain confident linkages, we set a threshold α over $\mathcal{E}_1$ to retain the top $\lceil \alpha |\mathcal{N}(i)| \rceil$ linkages for item i and top $\lceil \alpha |\mathcal{N}(q)| \rceil$ linkages for query q. Thus we have $\mathcal{E} = \mathcal{E}_\alpha$. Note that α trades off the coverage and confidence of query-item linkages, where $0 < \alpha \leq 1$.

Graph-Based Sequence Augmentation. We augment input sequences by adopting a stochastic shared embedding (SSE) idea [22] based on our constructed query-item graph. For $\hat{s}_t \in \hat{S}$, we replace $\hat{s}_t$ with probability β following:

$$i \sim \hat{s}_t, j \nsim \hat{s}_t \rightarrow p(i, \hat{s}_t)/p(j, \hat{s}_t) = \rho, \text{ if } \delta(\hat{s}_t) = 0,\ i, j \in \mathcal{I} \tag{5}$$

$$q \sim \hat{s}_t, k \nsim \hat{s}_t \rightarrow p(q, \hat{s}_t)/p(k, \hat{s}_t) = \rho, \text{ if } \delta(\hat{s}_t) = 1,\ q, k \in \mathcal{Q}. \tag{6}$$

Here $p(\cdot, \cdot)$ is the replacement probability, and ρ is a constant greater than 1. We use $\sim$ ($\nsim$) to denote whether two nodes are similar or not. Given a graph $\mathcal{G}$, we define similar queries $q \sim k$ when q and k have some common neighbour(s), i.e., $\mathcal{N}(q) \cap \mathcal{N}(k) \neq \emptyset$. This is motivated by the fact that nodes (e.g. "cafe" and "coffee") which connect to many common neighbors are potentially similar, so are more likely to be replaced by each other for data augmentation. We can augment sequences 'on-the-fly' for each training epoch rather than generating all augmented sequences in advance.

2.3 Transformer-Based Model Backbone

Query-aware sequential recommendation is a new setting where various sequential recommendation backbones can build; here, we propose a Transformer-based [9, 21] model for query-aware setting, namely *Query-SeqRec*. Then we show model training, including some practices to handle large item pool sizes (e.g. 10 million items).

Representation. We use embeddings $\mathrm{M}^0 \in \mathbb{R}^{|\mathcal{I}|\times d}$, $\mathrm{M}^1 \in \mathbb{R}^{|\mathcal{Q}|\times d}$, $\mathrm{P} \in \mathbb{R}^{\hat{T}\times d}$, $\mathrm{B} \in \mathbb{R}^{2\times d}$ to represent d-sized *items*, *queries*, *timestep*, *interaction type*, respectively. We represent the inputs: **(1) Item, timestep:** Given item i and timestep (position) t, we directly look up corresponding embeddings $\mathrm{M}^0_i, \mathrm{P}_t$ respectively. **(2) Query:** For query $q = [v_1, \ldots, v_{|q|}]$, we retrieve corresponding word embeddings and adopt an average pooling operation to get our query representation $\mathrm{M}^1_q \in \mathbb{R}^{1\times d}$. Note that other methods (e.g. a hidden vector from an LSTM [8], sentence vector from BERT [2]) can also be used to obtain M^1_q. Note that the length of user query words is often short (e.g. less than 3 on average in our datasets), and usually lack strong sequential patterns. So average pooling that follows a bag-of-words paradigm can be a simple yet effective way to represent queries. **(3) Interaction type:** We look up $\mathrm{B} \in \mathbb{R}^{2\times d}$ to get embeddings for different interaction types (i.e., item or query).

Query-Aware Transformer Layer. Given a heterogeneous sequence $\hat{S}$ described in Section 2.1, we retrieve the input embedding matrix from the embedding layer `Emb` as:

$$\mathrm{E}^{(0)} = \texttt{Emb}(\hat{S}) = \begin{bmatrix} \mathrm{M}^{\delta(\hat{s}_1)}_{\hat{s}_1} + \mathrm{B}_{\delta(\hat{s}_1)} + \mathrm{P}_1 \\ \cdots \\ \mathrm{M}^{\delta(\hat{s}_{\hat{T}})}_{\hat{s}_{\hat{T}}} + \mathrm{B}_{\delta(\hat{s}_{\hat{T}})} + \mathrm{P}_{\hat{T}} \end{bmatrix}. \tag{7}$$

$\mathrm{E}^{(0)} \in \mathbb{R}^{\hat{T}\times d}$ is the input embedding matrix and + denotes element-wise addition. Here item and query representations (with interaction type embeddings B) are learned in a joint embedding space and are aware of sequential order by positional (timestep) embeddings. We build L Transformer [21] blocks on top of the embedding layer `Emb`, which works as sequential encoder to generate $\mathrm{E}^{(L)} \in \mathbb{R}^{\hat{T}\times d}$ as the output embedding matrix. The details of the stacked transformer block construction refer to [9, 19].

Predictor Layer. Given the output $\mathrm{E}^{(L)}_t \in \mathbb{R}^{1\times d}$ at timestep t (i.e., the t-th row in matrix $\mathrm{E}^{(L)} \in \mathbb{R}^{\hat{T}\times d}$), we follow BERT4Rec [19] to calculate output probability over a target i as:

$$P\left(\hat{s}_{t+1} = i \mid \hat{S}, \delta(\hat{s}_{t+1}) = 0\right) = \texttt{softmax}_{\mathrm{i}}\left(\mathrm{E}^{(L)}_t \mathrm{M}^{0^\top}\right), \tag{8}$$

where $\texttt{softmax}_{\mathrm{i}}$ denotes the i-th probability from the softmax layer and the logits are interpreted as inner product similarities between the output $\mathrm{E}^{(L)}_t$ with the original item embeddings from M^0.

2.4 Handling Large Item Vocabularies

Loss with Sampled Softmax. Technically, a large item embedding matrix $\mathrm{M}^0 \in \mathbb{R}^{|\mathcal{I}|\times d}$ due to item vocabulary size $|\mathcal{I}|$ (e.g. 10 million items [15]) may be prohibitive in terms of GPU memory with the softmax layer in Equation (8) in backpropagation (e.g. in the order of 100 GiB). Previous models like BERT4Rec [19] did not encounter this problem because the experimental datasets are small (e.g. 30 thousand items). In such cases, we use sampled softmax to reduce the memory cost in backpropagation, and revise Equation (8):

$$P_n\left(\hat{s}_{t+1} = i \mid \hat{S}, \delta(\hat{s}_{t+1}) = 0\right) = \texttt{softmax}_1\left(\mathrm{E}^{(L)}_t \mathrm{M}^{(n)\top}\right). \tag{9}$$

$\mathrm{M}^{(n)} \in \mathbb{R}^{n\times d}$ denotes sampled item embeddings. P_n is the probability that item i should be the target rather than the other $n-1$ candidates. This cross-entropy loss with sampled softmax also unifies the widely used BPR loss [17] when $n = 2$. We use the same full / sampled softmax for baselines and our models for fair comparison.

Multi-GPU Embedding. To feed a large item embedding table $\mathrm{M}^0 \in \mathbb{R}^{|\mathcal{I}|\times d}$ into GPU memory, the embedding table M^0 is split along the hidden size dimension (rather than item dimension), i.e. loading $\mathrm{M}^0_1 \in \mathbb{R}^{|\mathcal{I}|\times d_1}, \ldots, \mathrm{M}^0_{\mathrm{m}} \in \mathbb{R}^{|\mathcal{I}|\times d_m}$ onto m GPUs respectively; then we retrieve and concatenate needed item embeddings onto a single GPU during training in the form of mini-batches.

Table 1: Data Statistics. *Inter* for item interaction; *S* for sequence; *I* for item; *Q* for query; *A-I* for average number of interactions per item; *A-S* for average sequence length and *A-Q* for average number of query occurrences.

	#Inter	#I	#S	#Q	A-I	A-S	A-Q
Diginetica	52,164	22,587	8,020	5,870	2.31	6.50	1.92
Unsplash	1,623,566	22,517	240,993	56,634	72.10	6.74	9.63
Stock	25,731,635	8,633,462	987,173	1,516,020	2.98	26.07	1.95

3 EXPERIMENTS

3.1 Experimental Setting

Datasets. We use three datasets (see statistics in Table 1) for query-aware sequential recommendation. **(1) Diginetica**[3] is released in the CIKM 2016 CUP, containing user search and browsing logs on *diginetica.com*. This dataset is commonly used in session-based or sequential recommendation, only using transaction data and ignoring user queries. But our experiments use both users' clicks (on items) and queries (in sessions). **(2) Unsplash**[4] is a dataset from the freely-usable photography website *unsplash.com* with users' search and download logs. We use the *lite*-version data. **(3) Stock-Industrial** is the largest dataset we constructed for experiments. It is collected from Adobe Stock Image platform [5] from Oct. 16—31, 2020. We use users' search and click logs.

Metrics. We follow [9, 19] to conduct a *leave-last-out* data split. We use truncated Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (N@K) [9, 19] ($K = 20$) to measure ranking quality.

Baselines. The first group of baselines are item-only sequential recommenders: **(1) FPMC [18]** combines Markov chains with matrix factorization. **(2) GRU4Rec$^+$ [5]** is a improved RNN-based model for users' item interaction sequences for session-based recommendation [5, 6]. **(3) SASRec [9]** is a uni-directional self-attentive sequential recommender. **(4) BERT4Rec [19]** is a BERT-like [2] sequential recommender capturing bi-directional contextual information. **(5) SSE-PT [23]** extends SASRec by using explicit user representations. The second group includes context-aware baselines incorporating query information but do not consider the sequence order of item interactions: **(1) Non-personalized Search (NS)** projects query and item representations into a joint embedding space and defines similarities using inner product. We use co-occurrence of queries and items in the data (i.e., non-personalized). **(2) QBPR:** We adopt VBPR [3] to incorporate query (instead of

[3] https://competitions.codalab.org/competitions/11161
[4] https://unsplash.com/data
[5] https://stock.adobe.com

Table 2: Method Comparison. Highest/second highest scores are bolded/underlined. Here Δ_1 represents the relative improvement from SASRec to Query-SeqRec, Δ_2 represents the relative improvement from the best baselines to Query-SeqRec. * denotes sampled-item-ranking (1k negatives) rather than all-item-ranking metrics (which is infeasible for the industrial-scale dataset).

		Item-Only Sequential Baseline					Query-Aware Baseline				Query-Aware Seq. Rec.		
Dataset	**Metric**	FPMC	GRU4Rec$^+$	SASRec	BERT4Rec	SSE-PT	NS	QBPR	FM	NeuFM	Query-SeqRec	Δ_1	Δ_2
Diginetica	HR@20	0.2996	0.2174	0.3508	0.3221	0.3425	0.2948	0.1438	0.3571	0.3359	**0.4037**	+15.1%	+13.0%
	N@20	0.1953	0.1160	0.1979	0.1714	0.2315	0.1760	0.0986	0.2323	0.2245	**0.2361**	+19.3%	+01.6%
Unsplash	HR@20	0.5307	0.5874	0.5881	0.5912	0.5912	0.5317	0.2723	0.5199	0.5499	**0.6796**	+15.6%	+15.0%
	N@20	0.2669	0.2924	0.2972	0.2697	0.2985	0.2039	0.1221	0.1984	0.2109	**0.3439**	+15.7%	+15.2%
Stock*	HR@20	0.3832	0.4284	0.4527	0.4472	0.4549	0.2215	0.2153	0.1749	0.2625	**0.4831**	+06.7%	+06.2%
	N@20	0.2993	0.3412	0.3404	0.3445	0.3541	0.1677	0.1129	0.1319	0.1955	**0.3708**	+08.9%	+04.7%

Table 3: Ablation study for the effectiveness of query information and sequence augmentation. Here *Q* represents incorporating query information as Section 2.1; *A* represents our sequence augmentation method as Section 2.2; *R* means using a uniform random replacement strategy to replace *A*.

Dataset	Metric	Query-SeqRec	w/ R	w/o A	w/o Q
Diginetica	HR@20	**0.4037**	0.3996	0.3908	0.3508
	N@20	**0.2361**	0.2351	0.2287	0.1979
Unsplash	HR@20	**0.6796**	0.6672	0.6698	0.5881
	N@20	**0.3439**	0.3335	0.3403	0.2972
Stock*	HR@20	**0.4831**	0.4802	0.4758	0.4527
	N@20	**0.3708**	0.3686	0.3653	0.3404

visual) information. **(3) FM [16]** is a classic context-aware recommendation technique. We use the same 'bag-of-words' query representations as Section 2.3. **(4) NeuFM [4]** is a deep architecture for context-aware recommendation. We use the same features as FM and adopt MLPs for higher-order feature interactions.

Implementation Details. All models are trained with Adam [10] (initial lr=1e-3). We set d=64, and select l2 from {0, 1e-6, 1e-4, 1e-2, 1, 10} and dropout probability from {0, 0.2, . . . , 0.8}. We search α for sequential augmentation from {0.1, 0.2, . . . , 1} and ρ from {1, 1.1, . . . , 2}. For three datasets, we set the maximum length of query words as 5 and the maximum length of user sequences as 50.

3.2 Model Performance

General Performance Improvement. Table 2 shows our model outperforms all baselines. Specifically, Δ_1 is the relative improvement against the backbone model (SASRec). It shows our model gains 12.5% HR@20 and 14.6% N@20 against SASRec on average, which shows the effectiveness of incorporating query information and sequential augmentation strategies. Δ_2 represents the relative improvement against the best baselines for each dataset. For example, our method gains 11.4% HR@20 and 7.2% N@20 on average.

Improvement across Datasets. The benefits of incorporating queries varies across different datasets. The relative improvements on Diginetica and Unsplash are relatively more than for the Stock dataset. For example, Δ_1 shows 15.6% HR@20 gain for Unsplash but 6.7% HR@20 gain for Stock. Presumably, this is mainly because the average sequence lengths of Diginetica and Unsplash are shorter than Stock (e.g. 6.74 for Unsplash *vs.* 26.07 for Stock from Table 1). Shorter sequences bring insufficient information and more uncertainty about user intent, so that user queries help more.

Incorporation	HR@20	N@20
Heterogeneous	**0.3908**	**0.2287**
Early	0.3719	0.2169
FDSA [24] (Late)	0.3697	0.2081
NOVA [13]	0.3594	0.2031

Table 4: Query Incorporation.

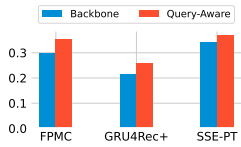

Table 5: More Backbones (HR@20).

3.3 Ablation Study

Effectiveness of Each Component. Table 3 shows the ablation studies of our essential components: (1) For incorporating queries, compared with *w/o Q*, our model and *w/o A* show that incorporating query information can significantly improve recommendation accuracy. (2) For the sequential augmentation, ours *vs. w/o A* shows that our augmentation strategy can improve the query-aware sequential recommenders; ours *vs. w/ R* indicates that introducing the query-item graph for augmentation outperforms uniform random replacement (similar to SSE-PT [23]).

Different Query Incorporation Methods. Table 4 shows the exploration of different query exploration methods on Diginetica. To exclude the influence of data augmentation, we report results without any augmentation strategies. Table 4 show empirically *Heterogeneous* (in Section 2.1) achieves better performance than the other three ways of organizing the query- and item-sequence, which are Early fusion, FDSA [24] fusion (i.e., late fusion) and NOVA [13] fusion. For example, *Heterogeneous* achieves 0.3908 HR@20 against 0.3719 from *Early*. 0.3697 from *FDSA* and 0.3594 from *NOVA*.

More Backbone Models. To show the generalization of the query-aware sequential recommendation setting, we experimented with other sequential recommenders as backbones. Table 5 shows that though model architectures are different (e.g. FPMC is Markov-Chain-based, GRU4Rec$^+$ is RNN-based, SSE-PT is Transformer-based), incorporating user query information under our framework can consistently improve the ranking performance. For example, on Diginetica, HR@20 of query-aware FPMC outperforms the corresponding backbone by 17.7% (relative improvement).

4 CONCLUSION

User queries are overlooked in sequential recommendation but can be an important contextual clue to predict users' evolving intent. We propose a *query-aware* sequential recommendation setting and a sequential recommender, *Query-SeqRec*, to incorporate query information, and examine different incorporation designs, showing the effectiveness of using user queries in sequential recommendation.

REFERENCES

[1] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks.. In *RecTemp@ RecSys*. 57–59.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[3] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[4] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.

[5] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[6] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[7] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 241–248.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 322–330.

[12] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. *arXiv preprint arXiv:2103.03578* (2021).

[13] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Noninvasive Self-attention for Side Information Fusion in Sequential Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4249–4256.

[14] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1091–1100.

[15] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. 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)*. 188–197.

[16] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[18] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[19] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[20] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. https://arxiv.org/pdf/1706.03762.pdf

[22] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2019. Stochastic shared embeddings: data-driven regularization of embedding layers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 24–34.

[23] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*. 328–337.

[24] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation.. In *IJCAI*. 4320–4326.