Personalized Retrieval over Millions of Items

Hemanth Vemuri[*]
hvemuri@microsoft.com
Microsoft
India

Sheshansh Agrawal[*]
sheagraw@microsoft.com
Microsoft
USA

Shivam Mittal[*]
shimitta@microsoft.com
Microsoft Research
India

Deepak Saini
desaini@microsoft.com
Microsoft
USA

Akshay Soni
akson@microsoft.com
Microsoft
USA

Abhinav V. Sambasivan
abhinavv@microsoft.com
Microsoft
USA

Wenhao Lu
wenhlu@microsoft.com
Microsoft
USA

Yajun Wang[†]
yajwang@linkedin.com
LinkedIn Corporation
USA

Mehul Parsana[†]
mehulparsana@google.com
Google
USA

Purushottam Kar
purushot@cse.iitk.ac.in
IIT Kanpur
India

Manik Varma
manik@microsoft.com
Microsoft Research
India

ABSTRACT

Personalized retrieval seeks to retrieve items relevant to a user event (e.g. a page visit or a query) that are adapted to the user's personal preferences. For example, two users who happen to perform the same event such as visiting the same product page or asking the same query should receive potentially distinct recommendations adapted to their individual tastes. Personalization is seldom attempted over catalogs of millions of items since the cost of existing personalization routines scale linearly in the number of candidate items. For example, performing *two-sided* personalized retrieval (with both event and item embeddings personalized to the user) incurs prohibitive storage and compute costs. Instead, it is common to use non-personalized retrieval to obtain a small shortlist of items over which personalized re-ranking can be done quickly. Despite being scalable, this strategy risks losing items uniquely relevant to a user that fail to get shortlisted during non-personalized retrieval. This paper bridges this gap by developing the XPERT algorithm that identifies a form of two-sided personalization that can be scalably implemented over millions of items and hundreds of millions of users. Key to overcoming the computational challenges of personalized retrieval is a novel concept of morph operators that can be used with arbitrary encoder architectures, completely avoids the steep memory overheads of two-sided personalization, provides millisecond-time inference and offers multi-intent retrieval. On multiple public and proprietary datasets, XPERT offered upto 5% superior recall and AUC than state-of-the-art techniques. Code for XPERT is available at the following GitHub repository.

[*]These authors contributed equally to the paper
[†]Work done while the author was at Microsoft

SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591749

CCS CONCEPTS

• **Information systems → Personalization**.

KEYWORDS

personalized dense retrieval, personalization, channels, user embedding, morph operators

ACM Reference Format:
Hemanth Vemuri, Sheshansh Agrawal, Shivam Mittal, Deepak Saini, Akshay Soni, Abhinav V. Sambasivan, Wenhao Lu, Yajun Wang, Mehul Parsana, Purushottam Kar, and Manik Varma. 2023. Personalized Retrieval over Millions of Items. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3539618.3591749

1 INTRODUCTION

Personalized retrieval: The goal of personalization is to retrieve items relevant to a user event that is also adapted to the user's long-term preferences. This allows two users with distinct preferences to receive distinct retrievals even if they perform the same event e.g. visiting the same product page or asking the same query, thus enhancing user experience.

Retrieval for recommendation: Retrieving relevant items for a user from a large catalog of candidate items based on the user's past *events* such as browsing history is a well studied-problem with a lot of practical applications such as home-page recommendation [4, 14, 23]. A popular paradigm for retrieval is Per-Event Retrieval (PER), where items similar to a single user *event* are retrieved. Several tasks such as product-to-product recommendation [19] and

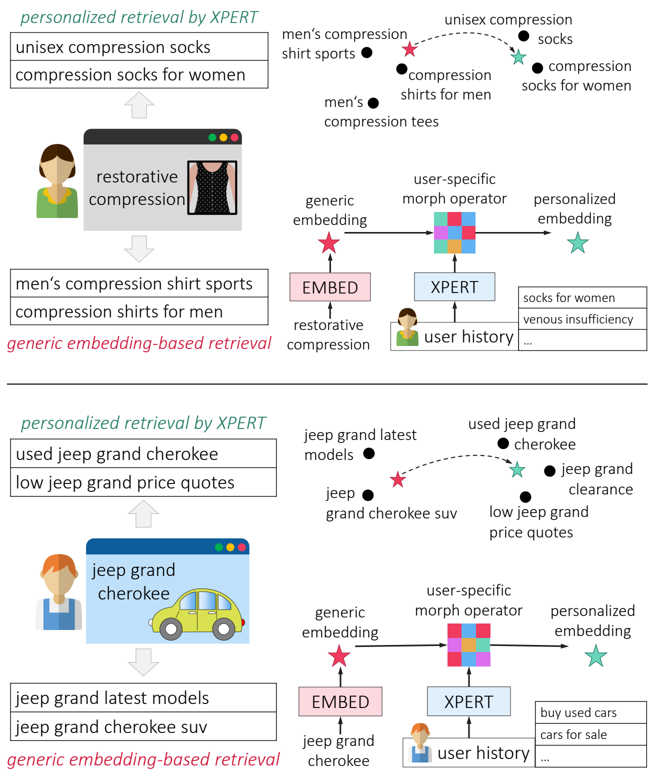

Figure 1: Actual personalized retrievals by XPERT. XPERT personalizes retrieval by morphing the generic embedding of a user event before MIPS retrieval. Each user's morph operator moves the embeddings to a region which captures the user's longer-term preferences better. (Top) A user visited a webpage titled "Restorative Compression" selling compression garments. Prior browsing history indicated the user's gender as well as their desire to purchase socks which XPERT correctly inferred. Suggestions using the generic event embedding focussed on shirts for men instead. (Bottom) A user visited a webpage selling a vehicle and browsing history revealed the user's desire to purchase discounted or low-cost products. XPERT successfully incorporated this into its suggestions but non-personalized generic embedding failed to do so.

search [33] are Per-Event Retrieval scenarios. A common way to perform retrieval for recommendation is to perform PER on a subset of user *events* and return the union as the retrieval result [14, 23]. When each individual event's retrieval is non-personalized and independent, we call it Non-Personalized Per-Event Retrieval (NP-PER). Personalized Per-Event Retrieval (P-PER) on the other hand, seeks to adapt each individual event's retrieval set to the user's long-term preferences.

The Key to Efficient NP-PER: Advances in dense retrieval such as Siamese networks [6, 11, 15, 32] have enabled efficient NP-PER. Consider a user u who performs event e and let a be a candidate item. Dense NP-PER techniques learn a deep architecture ϕ to embed user events and candidate items into a shared vector space so that the dot product $\langle\phi(a), \phi(e)\rangle$ is indicative of the relevance of item a to event e. Note that the embedding function ϕ does not depend on the user u i.e. it is non-personalized. At test time, the items most relevant to an event e_t can be retrieved simply by running a Maximum Inner Product Search (MIPS) query [16] with the test event embedding $\phi(e_t)$. This is usually orders of magnitude faster than explicitly computing $\langle\phi(a), \phi(e_t)\rangle$ for all catalog items a and can offer millisecond-time retrieval even when the catalog contains millions of items.

Challenges in P-PER: It is challenging to accelerate P-PER using the MIPS trick. One possibility is *two-sided* personalization that uses a personalized embedding function ϕ_u that is unique to every user i.e. use $\langle\phi_u(a), \phi_u(e)\rangle$ as the relevance score. Experiments in Section 5 and Figure 2 show that two-sided personalization can indeed offer flexible and accurate retrieval. This could offer accelerated retrieval if a separate MIPS structure over the entire candidate set is established separately for every user. For a million items and a billion users, creating these structures alone would require million $\times$ billion compute and storage. Another possibility is to use non-decomposable personalized scoring functions [18] of the form $f(a, e, u)$ indicating the relevance of a candidate item a for an event e performed by user u. Note that this score is personalized as it depends on the user u (e.g. their browsing history). Existing methods use nonlinear architectures e.g. MLP [10] or transformers [22] (see Section 2) that offer effective personalization but also preclude the MIPS trick and require applying f explicitly to each candidate item to find the most relevant ones. This does not require enormous storage but incurs infeasible inference times when candidate items are in the millions. The popular workaround is to shrink the candidate set by first performing NP-PER and then applying personalized ranking only to the shortlisted items. However, items that fail to get shortlisted by NP-PER step are irrevocably lost even if personalization could have later found them to be relevant to a particular user's tastes. Scalable P-PER over millions of items poses opportunities and computational challenges.

Contributions: This paper presents the XPERT method that makes personalized retrieval possible on catalogs of millions of items with millisecond-scale inference time. The key contributions of XPERT include:

(1) Introducing the concept of personalized *morph operators* that make personalized retrieval with a form of two-sided personalization possible with storage and inference costs similar to non-personalized retrieval.
(2) Integrating a scalable channel mechanism to select events for P-PER into XPERT to additionally improve diversity in the final retrieved set of items.
(3) Developing the XPERT technique that could train on a P-PER task with 1+ million catalog items and 1+ billion user-item interactions within 2 hours on a single P40 GPU and offer 2.1 millisecond inference time on a single CPU.
(4) 5% superior recall and AUC than state-of-the-art personalized retrieval methods.

Figure 1 shows two real-life examples where XPERT retrieved personalized items relevant to the user event. Note that, this paper focuses on improving retrieval through personalization and, as is usual in retrieval literature, other layers in a recommendation pipeline e.g. downstream re-ranking over shortlisted items, item auctions, display-layout selection etc. are beyond its scope.

2 RELATED WORKS

As noted in Section 1, several personalized ranking and recommendation methods can only be applied to small item shortlists. Since they do not have to deal with catalogs of millions of items, these methods implement the scoring function $f(a, e, u)$ using elaborate architectures such as MLP [10], transformers [22], personalized attention [30], graph convolutions on user-item interaction graphs [12] and meta-path variants thereof [9]. However, as discussed in Section 1, these methods if run on a non-personalized shortlist risk losing items, especially ones corresponding to rare user interests. This section focuses on methods which try to personalize this shortlist by doing personalized retrieval.

Single User Representation (SUR) Methods: SUR methods do not use a specific user event to trigger or seed the retrieval process i.e. do not perform "per-event" retrieval (PER) but rather embed the user itself as a vector and use a scoring function of the form $f(a, u) = \langle \phi(a), \xi(u) \rangle$ with a non-personalized item embedding function ϕ. Diverse models have been used to implement ξ as a function of user profile and browsing history including MLPs (feed-forward networks) [4], sequential models to encode browsing order such as GRU [1, 28] and LSTM [21], hierarchical attention networks [31] and transformers [26]. Although these methods offer scalable retrieval using the MIPS trick, they need to update the user embedding rapidly in response to browsing activities in order to stay relevant to the user's current intents which is a non-trivial overhead for real-time systems. They can also suffer from lack of diversity by relying on a single embedding. In experiments, XPERT could offer upto 15% better recall than SUR methods such as YouTube-DNN [4].

Channel-based Methods: A common technique [14, 23] to improve retrieval diversity, especially in SUR methods is to learn multiple user representations. User events are partitioned into *channels* each of which offers a distinct user representation. A variety of channeling techniques exist in literature including parametric ones such as Octopus [14] that learn multi-head architectures to effect channels as well as non-parametric ones such as PinnerSage [23] that simply cluster the event embeddings of a user with each cluster becoming a separate channel. In experiments, an NP-PER variant that used channels could offer as much as 8.7% higher recall than SUR methods such as YouTube-DNN [4]. XPERT itself incorporates a scalable channeling architecture. It is notable however that channels by themselves do not offer sufficient personalization. For instance, XPERT offered as much as 8% higher recall compared to its non-personalized variant with channels.

P-PER Methods: The DPSR method [33] uses a decomposed scoring function of the form $f(a, e, u) = \langle \phi(a), \psi(e, u) \rangle$ that uses a non-personalized item embedding ϕ but a personalized event embedding ψ. The event embedding is personalized by concatenating a non-personalized embedding of the event e with an average of embeddings of historical events from the users history and applying several MLP layers. However, this approach to event embedding personalization does not seem very effective as noted by DPSR itself [33]. In contrast, XPERT shows that by making use of personalized linear operators, one can effectively use a scoring function of the form $f(a, e, u) = \langle \psi(a, u), \psi(e, u) \rangle$ but in a manner that requires computational expense similar to those required by scoring functions of the form $f(a, e, u) = \langle \phi(a), \psi(e, u) \rangle$. This implicitly allows personalization of not just the event embedding but candidate item embeddings as well and can offer upto 9% higher recall and millisecond-time inference at the same time. The use of linear operators to affect personalization has been studied in the past, such as the COT method [13] that implements personalization by using a linear (tensor) operator. However, the method works in the matrix completion setting and faces challenges in incorporating new items and new users inherent to all matrix completion-style methods.

3 PROBLEM SETTING AND NOTATION

Let $\mathcal{U}$ denote the set of users and $\mathcal{A}$ denote the catalog of items available for recommendation. $\mathcal{A}$ can have millions of items and both $\mathcal{U}$ and $\mathcal{A}$ may dynamically evolve over time. For any user $u \in U$, let $\mathcal{H}_u$ denote the (ordered) list of their historical events. We will use the term *event* to denote user activity such as visiting a webpage, clicking on an ad, submitting a query, etc. Note that an event can be naturally identified with an item as well, say by considering the product for an ad click event or the webpage title for a page visit event. Events and items will be represented using their textual descriptions such as webpage title or query text. A shared encoder $\mathcal{E}$ will be used to obtain generic (non-personalized) embeddings of events and items as D-dimensional unit norm vectors i.e. for any event e or item $a \in \mathcal{A}$, we have $\mathcal{E}(e), \mathcal{E}(a) \in S^{D-1}$ where S^{D-1} is the surface of the D-dimensional unit sphere. As is standard in dense retrieval applications, we assume access to a maximum inner product search (MIPS) structure such as HNSW [16] that offers log-time retrieval. A MIPS structure over a set of unit vectors $W \subset S^{D-1}$ can take a query vector say $\mathbf{v} \in S^{D-1}$ and $k \in \mathbb{N}$ and return the k vectors from W with the largest dot product $\langle \mathbf{w}, \mathbf{v} \rangle$ with $\mathbf{v}$ (equivalently the nearest neighbors of $\mathbf{v}$ since the vectors are unit norm). This $O(kD \log |W|)$ time and milliseconds in practice even when $|W|$ is in the millions. MIPS structures can also be updated to ingest new items. Given this, P-PER requires the following two subtasks to be solved - 1. This corroborates an observation of DPSR [33] that leveraging user history in their formulation resulted in only marginal gains. 2. For each seed event $e \in \hat{\mathcal{H}}_u$ retrieve a set of items from $\mathcal{A}$ relevant to e and personalized to u. Seed events are expected to be historical events that reflect the unique content/product preferences of the user and can, for example, include the most latest event performed by the user. Also note that the items (implicitly) associated with historical events may or may not be a part of the candidate set $\mathcal{A}$ (e.g. a historical product item may no longer be on sale) but these seed events are key to initiating the retrieval process as personalized embeddings of seed events will be used by XPERT to retrieve candidate items most relevant to the user's preferences.

4 XPERT: E<u>X</u>TREME <u>PE</u>RSONALIZED <u>RE</u>T<u>R</u>IEVAL

[1] **Motivation**: The key desirables of personalized retrieval include:

(1) *Computational Efficiency*: given a new user event, retrieval must happen within milliseconds.
(2) *Storage Overheads*: the model should not require more than $O(|\mathcal{A}|)$ storage. Note that $O(|\mathcal{A}|)$ storage is required for a MIPS structure over the set $\mathcal{A}$ of candidate items even in NPR.
(3) *Long-short Term Adaptation*: the model should adapt to long-term user preferences e.g. those based on demographics but not neglect current (short-term) interests.
(4) *Ease of Maintenance*: a trained model should not require frequent re-training.

P-PER naturally addresses points 3 and 4 as choosing the last few events of a user as seed events to trigger the retrieval process is expected to cover the short-term interests of the user as well as offers multi-intent retrieval. Thus, personalization need only capture the long-term preferences of a user. However, these are by definition not expected to change rapidly and thus the model need not be re-trained frequently. To address points 1 and 2, we notice that a decomposed function of the form $f(a,e,u) = \langle \phi(a,u), \phi(e,u) \rangle$ that performs two-sided personalization using personalized embeddings of both the event e and candidate item a does allow MIPS-based retrieval. As shown in Figure 2, two-sided personalization can adapt to user tastes very flexibly. However, this would also require maintaining a separate, personalized MIPS structure for each user which is infeasible. This is why existing techniques such as DPSR [33] settle for a simplified scoring function $f(a,e,u) = \langle \phi(a), \psi(e,u) \rangle$ that uses personalized event embeddings but non-personalized item embeddings that requires a single MIPS structure and $O(|\mathcal{A}|)$ storage. XPERT's key novelty is a technique that enables a form of two-sided personalization i.e. using scoring functions of the form $f(a,e,u) = \langle \phi(a,u), \phi(e,u) \rangle$, but at $O(|\mathcal{A}|)$ storage cost and millisecond retrieval time.

Morph Operators: XPERT considers personalized embedding functions of the form $\phi(a,u) = P_u \cdot \mathcal{E}(a)$ and $\phi(e,u) = Q_u \cdot \mathcal{E}(e)$. Here $\mathcal{E}$ is a text embedding model shared by all users that embeds events and items to D-dimensional unit vectors and user-specific *morph operators* $P_u, Q_u \in SO(D)$ for $u \in \mathcal{U}$. These morph operators are expected to encode user preferences e.g. preference for low-priced products, and personalize generic embeddings along those directions. For now, let P_u, Q_u be orthonormal matrices since they ensure that for any unit-norm vector $\mathbf{v}$, $M_u\mathbf{v}$ is also unit norm. We will relax this requirement soon. Thus, $f(a,e,u) = \langle \phi(a,u), \phi(e,u) \rangle = \mathcal{E}(a)^\top P_u^\top Q_u \mathcal{E}(e)$. However, since the set of orthonormal matrices $SO(D)$ is closed under transposition and product it must be that $P_u^\top Q_u = L_u$ for some $L_u \in SO(D)$. Thus, $f(a,e,u) = \langle \mathcal{E}(a), \psi(e,u) \rangle$ where $\psi(e,u) = L_u \cdot \mathcal{E}(e)$ and L_u is the user-specific morph operator. Note that this re-parameterized scoring function uses non-personalized item embeddings and requires a single MIPS structure but actually embodies a two-sided personalization model. Since it is expensive to perform optimization over the rotation group $SO(D)$ directly [7], XPERT relaxes the orthonormality requirement by reparameterizing ψ as $\psi(e,u) = \mathfrak{N}((R_u + I_D) \cdot \mathcal{E}(e))$ with $R_u \in \mathbb{R}^{D \times D}$ as the morph operator (that need not be orthonormal) and $\mathfrak{N} : \mathbf{v} \mapsto \mathbf{v}/\|\mathbf{v}\|_2$ being normalization. The skip connection with the identity matrix I_D provides regularization and also allows training to commence with non-personalized embeddings by initializing with $R_u = 0$.

Prediction Pipeline: XPERT uses a 3-segment pipeline (see Fig. 3):
Segment S1: For a user $u \in \mathcal{U}$, embeddings of events in the user history $\{\mathcal{E}(e) : e \in \mathcal{H}_u\}$ are aggregated using a 2-layer transformer with 8 attention heads [27] without positional encoding to obtain an *intermediate user embedding* $\hat{\mathbf{z}}_u \in \mathbb{R}^D$.
Segment S2: The intermediate user embedding $\hat{\mathbf{z}}_u$ is passed through a single feed-forward layer with ReLU non-linearity and the D^2-dimensional output is reshaped into a $D \times D$ matrix that serves as the morphing operator R_u for user u. XPERT uses $D = 64$ for speed. Mild accuracy boosts were observed by using a multi-head architecture that passed $\hat{\mathbf{z}}_u$ through multiple feed-forward layers and averaged the resulting matrices to obtain R_u.
Segment S3: Using a seed selection strategy discussed below, a set $\hat{\mathcal{H}}_u \subset \mathcal{H}_u$ of s seed events is selected from user history and the personalized embedding $\psi(e,u) = \mathfrak{N}((R_u + I_D) \cdot \mathcal{E}(e))$ of each seed event $e \in \hat{\mathcal{H}}_u$ is used to query a MIPS structure over (generic embeddings of) the candidate items $\{\mathcal{E}(a) : a \in \mathcal{A}\}$ to obtain a set $\hat{\mathcal{A}}(e,u)$ of items. These are combined $\bigcup_{e \in \hat{\mathcal{H}}_u} \hat{\mathcal{A}}(e,u)$ to make the final prediction. Note that all events in $\mathcal{H}_u$ are used to learn R_u and not just the seed events $\hat{\mathcal{H}}_u$.
XPERT Training: The model was trained to predict the next event of a user $u \in \mathcal{U}$ based on their user history $\mathcal{H}_u$. To compute loss over a user $u \in \mathcal{U}$, the item involved in the next event $a_u^* \in \mathcal{A}$ was taken as a ground truth positive, a set $\hat{\mathcal{H}}_u$ of s seed events was chosen from the user history and a set $\mathcal{N}_u \subset \mathcal{A}$ of n hard-negative items was identified (negative mining discussed below). s and n were tuned as hyperparameters (see Appendix D).
Next, the seed event whose personalized embedding most closely resembled the clicked item/ad was chosen as the *most promising* seed event $e^* = \arg\max_{e \in \hat{\mathcal{H}}_u} \langle \mathcal{E}(a_u^*), \psi(e,u) \rangle$. The hard negative item most similar to the most promising seed event $b^* = \arg\max_{b \in \mathcal{N}_u} \langle \mathcal{E}(b), \psi(e^*,u) \rangle$ was also chosen. Then, using margin hyperparameters λ_+, λ_-, loss on a user is computed as

$$\ell(u) = \max\left\{\lambda_+ - \left\langle \psi(e^*,u), \mathcal{E}(a_u^*) \right\rangle, 0\right\} + \max\left\{\left\langle \psi(e^*,u), \mathcal{E}(b^*) \right\rangle - \lambda_-, 0\right\}.$$

The loss is averaged over all training users i.e. $\mathcal{L} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \ell(u)$ and is used to train the transformer layers in segment S1 and the feed-forward layer in S2 using the Adam optimizer.
Hard Negative Mining: Items that seem deceptively likely to get clicked/viewed as the next event but were not a part of the ground truth are termed hard negatives. Negative sampling is necessary [3, 5, 8, 32] with large output spaces since evaluating $\mathcal{L}$ with respect to *all* negative items in $\mathcal{A}$ would take $\Omega(|\mathcal{U}| \cdot |\mathcal{A}|)$ time. XPERT considered two types of negative mining: in-batch negative mining and global negative mining (see details in Appendix A in the supplementary [link]).
Seed Selection and Channels: XPERT explored two seed selection strategies: recency- and channel-based. The latter offered moderate

[1]Additional details for XPERT are provided at supplementary [link]. Code and datasets can be found in the following GitHub repository.

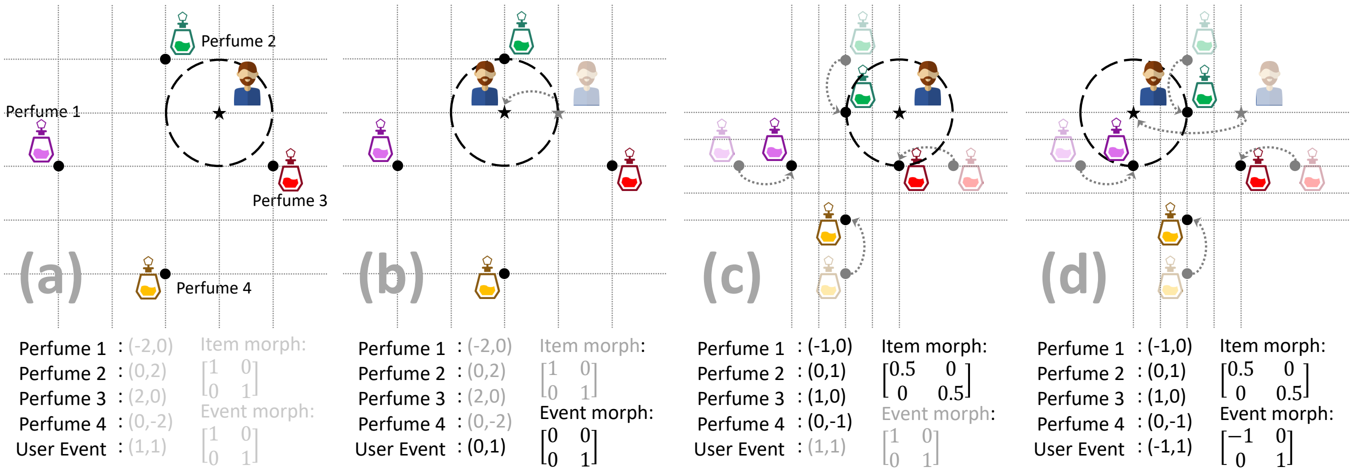

Figure 2: Illustrating the power of two-sided personalization, never mind its exorbitant cost. Figure 2(a) shows non-personalized retrieval, 2(b,c) represent event and item side personalization respectively, and 2(d) represents full two-sided personalization. In each of Figures 2(a,b,c,d), star denotes the user event embedding, black circles denote candidate item embeddings, and the grid lines illustrate the action of morphing embeddings. Only items within a unit radius of the user event are retrieved. Coordinates of embeddings and morph operators values are listed below each figure. Non-personalized embedding coordinates and identity morph operators are grayed out. Figure 2(a) shows that no perfume is retrieved if generic embeddings are used. If we just do event side personalization, we are able to move the event embedding close to just one perfume as is shown in example in Figure 2(b). We aren't able to retrieve more than 1 perfume even if it is relevant to the user. Item side personalization allows to move items close to the user, and as shown in Figure 2(c) allows retrieval of Perfumes 2 and 3 together or any 1 of those, but cannot retrieve any other subset. However, 2-sided personalization provides flexibility to retrieve any subset of items. Figure 2(d) shows retrieving perfume 1 and 2, and the complete example in Figure 5 in the supplementary [link]shows how two-sided personalization can retrieve any subset (pair or triplet or all) of perfumes. By implementing personalization using orthonormal morph operators, XPERT is able to offer the benefits of two-sided personalization without incurring the prohibitive storage cost.

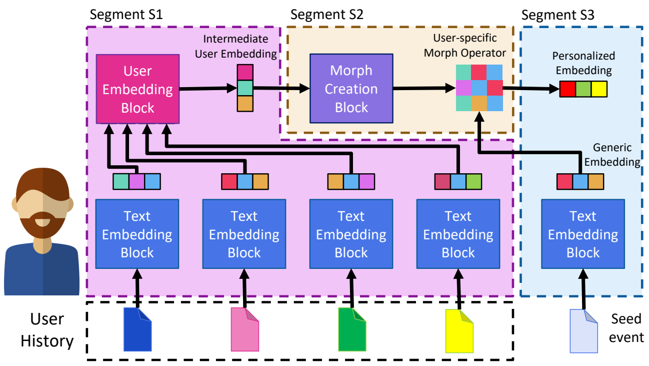

Figure 3: XPERT's pipeline consists of 3 segments. Segment S1 aggregates user history to obtain an intermediate user embedding that is used by S2 to extract a user-specific morph operator. S3 takes a seed event from user history and offers a personalized embedding that is used to perform retrieval by making a MIPS call. These three light-weight segments offer personalized retrieval within milliseconds.

boosts in retrieval accuracy but required an online algorithm for channel maintenance (see Appendix B). Channel-based architectures are popular [14, 23] and increase diversity but do not offer significant personalization by themselves.

Table 1: Dataset Statistics. ‡ indicates data redacted for proprietary datasets. U2A datasets are propreitary, AmazonReviews datasets are available with code at the following GitHub repository.

Dataset	$\|\mathcal{U}\|$	Total # of interactions	$\|\mathcal{H}_u\|$	$\|\mathcal{A}\|$
U2A-4M	3.96M	1.093B	‡	1.02M
U2A-300M	316M	23B	‡	19.4M
AmazonReviews-1M	920K	9.67M	10.42	286K
AmazonReviews-10M	9.71M	156M	16.06	3.7M

5 EXPERIMENTAL RESULTS

Datasets: Experiments were conducted on multiple public (AmazonReviews) and proprietary (U2A) personalized retrieval datasets (see Table 1) on a 24-core Intel Xeon 2.6 GHz machine with a single Nvidia P40 GPU.

• **AmazonReviews** - The AmazonReviews-(1M/10M) datasets were created out of the Amazon Review Data dump [20] and are publicly available with code at the following GitHub repository. The task in these datasets is to predict the product a user will review next, given the past products they have reviewed. The $n-2$ history items (where n is the user history length) of a user were used for training while the last two items were used for evaluation. Each history item was represented by the 768-dimensional embeddings of the product title from a 6-layered transformer model trained using NGAME [6, 25, 29]. Two versions of this dataset were created, namely AmazonReviews-1M and AmazonReviews-10M. For the smaller dataset, reviews from the following 5 categories were considered: CDs and Vinyl, Games, Electronics, Beauty, Grocery and Gourmet Food whereas all categories were considered for the AmazonReviews-10M dataset.

- **U2A** The U2A datasets were created from ad-click logs mined for two consecutive months from Microsoft's ad-serving network. For each click, user's browsing history from previous month was used to predict the final ad clicked by them. We note that browsing history was used and not just ad-click history since ad-clicks are scarce and browsing history offers a more informative signal to model user's preferences. Clicks with no accompanying historical browsing activity were filtered out. First month's data was used for training and second month's data was used for testing. Final data was randomly subsampled to form U2A-300M and U2A-4M datasets. Each ad and webpage was represented by 64-dimensional embeddings of the ad/page title from a 6-layered NGAME model [6, 25, 29].

Implementation and Hyperparameter Details are discussed in Appendix D of supplementary [link]
Baselines: Several SUR/P-PER baselines were used. Appendix E gives implementation details for these baselines.

- **NP-PER-recentS**: This method retrieves using the generic event embeddings of the most recent-s events of the user, where s was tuned as a hyperparameter.
- **SUR**: These use a single user embedding and do not use a seed event by the user to trigger the retrieval process. Various architectures have been used to implement SUR methods e.g. transformers (BERT4Rec [26]), feedforward networks (YouTube-DNN [4]) and GRUs ([28]) – see Section 2. We compare XPERT with the SUR-BERT and SUR-DNN baselines that closely resemble BERT4Rec and YouTube-DNN.
- **PinnerSage** [23]: This is NP-PER based retrieval method that clusters a user's history into multiple channels that are used for retrieval.
- **DPSR** [33]: This method was adapted to P-PER tasks by seeking retrieval with respect to all events in a seed event set. DPSR was offered advantages such as channel-based seed event selection and transformer-based user history aggregation (Original implementation [33] use a simple average instead).
- **XPERT w/o channels**: This variant used morph operators for personalizaton but only the last *s* events were chosen as seed events.
- **XPERT w/o morph operators**: This NP-PER variant offered channel-based seed event selection and closely resembles the PinnerSage method [23] but for differences in clustering algorithm.

Note that NP-PER variants offer retrieval which is not personalized while SUR variants offer personalized retrieval but may suffer from diversity and coverage issues since they only use a single embedding. In contrast, XPERT, PinnerSage and DPSR offer personalized retrieval that is diverse and offers good coverage of user interests since multiple user events are used to seed retrieval. Certain baselines were implemented by closely following their corresponding publications due to lack of publicly available code.

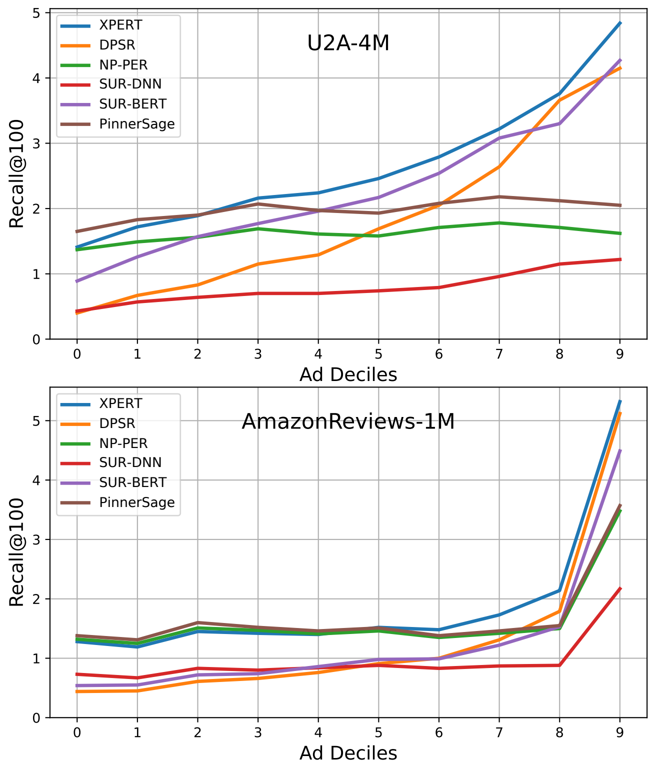

Figure 4: Decile-wise breakdown of Recall@100. Equi-voluminous item deciles were created on the basis of item popularity with decile 0 containing the most rarely clicked items and decile 9 containing the most popularly clicked items. XPERT leads on all deciles, often by a margin. In contrast, other methods such as DPSR do well on popular deciles but not so on rare deciles or the other way round.

Text Encoder Training: XPERT used a 6-layered DistilBERT base architecture [25, 29] pretrained using NGAME [6] as its text encoder $\mathcal{E}$. For U2A, a query-to-ad recommendation dataset mined from search engine click-logs was used to pretrain $\mathcal{E}$ on the (non-personalized) task of predicting the ad that would be clicked in response to a search engine query. For AmazonReviews, $\mathcal{E}$ was pretrained on the LF-AmazonTitles-1.3M dataset [2] on a product-to-product recommendation task. $\mathcal{E}$ was not fine-tuned while training XPERT although doing so could yield minor improvements.
Evaluation Metrics: Performance was evaluated using standard retrieval performance measures: Recall@k, nDCG@k, MRR@k ($k \in \{10, 50, 100\}$) and AUC (see Appendix C for metric definitions).
Results: Table 2 presents results on all datasets. XPERT was found to beat all baselines across all metrics on both public and proprietary datasets. XPERT's Recall@100 was at least 10% higher than NP-PER-recentS, which is a widely adopted retrieval method in commercial settings [17, 19, 23, 24]. In addition to this, XPERT outperforms the personalized retrieval baselines by up to 5% in terms of Recall@100. XPERT could also be upto 4%, 7.5% and 5% better in terms MRR@100, AUC@100 and nDCG@100 respectively compared to DPSR on U2A-4M dataset. Similar trends were observed on the two AmazonReviews datasets as well.

Table 2: Performance of XPERT vs baseline and competing algorithms across datasets and metrics.XPERT was found to offer up to 5% superior recall and 5% superior AUC compared to state-of-the-art in personalized retrieval techniques.

Method	Recall@10	Recall@50	Recall@100	nDCG @10	nDCG @50	nDCG @100	AUC@100	MRR@100
U2A-4M dataset								
NP-PER-recentS	6.450	15.502	16.155	3.726	5.762	5.872	13.451	3.364
SUR-DNN	4.257	9.149	11.036	2.346	3.422	3.730	8.320	2.023
SUR-BERT	12.635	20.042	22.864	8.487	10.116	10.575	18.686	7.592
PinnerSage	8.633	17.859	19.855	4.535	6.597	6.929	16.027	3.790
DPSR	8.371	17.040	18.352	4.687	6.617	6.836	15.147	4.016
XPERT (w/o channels)	12.009	22.482	22.962	7.651	10.031	10.113	20.023	6.865
XPERT w/o morph operators	8.504	17.879	19.715	4.617	6.713	7.019	15.988	3.938
XPERT	**14.852**	**25.778**	**27.189**	**9.318**	**11.790**	**12.026**	**23.390**	**8.212**
AmazonReviews-1M dataset								
NP-PER-recentS	3.467	7.620	9.034	2.115	3.224	3.509	11.997	2.468
SUR-DNN	2.616	4.591	5.413	1.702	2.242	2.406	7.327	1.973
SUR-BERT	4.285	6.227	7.213	3.684	4.208	4.404	10.333	4.888
PinnerSage	3.638	7.707	9.342	2.268	3.355	3.683	12.254	2.713
DPSR	3.879	6.539	7.401	3.000	3.725	3.899	10.590	3.932
XPERT	**5.505**	**9.344**	**10.70**	**4.422**	**5.453**	**5.724**	**14.849**	**5.799**
AmazonReviews-10M dataset								
NP-PER-recentS	2.383	4.793	5.449	1.454	2.106	2.239	7.484	1.662
SUR-DNN	1.368	2.039	2.270	0.930	1.115	1.161	3.182	1.054
SUR-BERT	3.277	4.329	4.710	2.665	2.956	3.032	7.017	3.316
PinnerSage	2.499	5.131	6.027	1.512	2.218	2.399	8.063	1.742
DPSR	2.795	4.240	4.426	2.114	2.516	2.554	6.688	2.568
XPERT	**3.943**	**6.106**	**6.550**	**3.136**	**3.724**	**3.815**	**9.740**	**3.992**
U2A-300M dataset								
NP-PER-recentS	7.140	15.901	17.819	3.413	5.891	6.190	15.225	3.478
XPERT	**9.682**	**20.010**	**22.291**	**6.013**	**9.348**	**10.001**	**20.119**	**8.012**

Analysis of Results: Fig 4 shows the contributions of each item decile to the overall recall of various baselines. On U2A-4M, NP-PER-recentS offered similar performance on both rare and popular items whereas SUR-BERT, SUR-DNN and DPSR all performed worse than NP-PER-recentS on rare items. PinnerSage was better than NP-PER-recentS across all deciles. However, XPERT offers the best of both worlds and performs better than NP-PER-recentS on rare items and better than SUR-BERT/DPSR/PinnerSage for popular items. On AmazonReviews-1M, XPERT again leads on all decides whereas baselines underperform either on popular items or rare items or both. XPERT w/o morph operators and XPERT w/o channels were found to offer almost 4% and 7% gains over NP-PER-recentS on U2A-4M revealing the benefits of morph operators and careful seed selection. It is notable that gains for XPERT w/o channels are attributable to the personalization enabled by morph operators, whereas those for XPERT w/o morph operators are attributable to channel based seed event selection.

Scalability and Inference Time: XPERT could train on the largest U2A-300M dataset with 316M users and 23 billion user-item interactions within 48 hours on a single P40 GPU. XPERT supports scalable inference. For live deployment, XPERT needs 256 bytes of additional memory per user to store the 64 dimensional vectors $\hat{\mathbf{z}}_u$. Upon any user activity, updating the seed events and applying the user-specific morph operator to obtain personalized seed event embedding takes 0.1 ms and the MIPS call takes upto 2 ms. All latency results are reported on single core CPU.

Benefits of Channels: Appendix F presents a discussion. Table 9 shows examples of channels created by Algo 1 that evidently capture distinct product categories. Table 10 shows that using channels allows XPERT to use superior seed events to increase retrieval diversity compared to NP-PER. Note that Algo 1, Table 10 and Table 9 are all available in the supplementary [link].

Illustrative Examples: As noted earlier, XPERT draws its gains from personalization by the user-specific morph operators and diversity offered by channels. Table 11 (available in supplementary [link]) shows two examples where ads retrieved by XPERT were better aligned to user tastes than NP-PER. In the first example, XPERT could retrieve the ad actually clicked by the user. In the first example in Table 11, user history indicated searches such as "compression socks for women". However, the seed event namely "Restorative Compression - Aqua Confetti - Primes Compression" was actually related to compression socks for men although the title does not reveal this. Thus, NP-PER recommended "compression shirts for men". In contrast, XPERT accurately captured both the gender and the intent of the user and recommended ads such as "unisex compression socks" that indicate a far superior alignment to user preferences. It is notable that the candidate set of items did not contain any web page such as "compression socks for women". The second example in Table 11 shows a use interested in used or low-price cars. However, the seed event page selected for retrieval, "2019 Jeep Grand Cherokee Limited RWD - $27,995 - CarGurus" did not contain this information and hence NP-PER recommended ads for new cars such as "new Jeep Grand Cherokee". However XPERT successfully latched on to the user itent and made recommendations such as "Used Jeep Grand Cherokee".

Ablation Experiments: Table 3 explores the impact of various design choices by XPERT:

- **seg1-mean**: replacing transformer layers in S1 with simple tf-idf weighted mean caused drop of 1.4% in Recall@100 and other metrics, indicating benefits of accurate user history aggregation.
- **seg3-no-residual**: removing the skip connection in S3 led to a drop in recall@100, AUC@100 and MRR@100. Residual connections stabilize training and prevent overpersonalization that could lead to a morphed embedding completely unrelated to the seed event.
- **NP-PER-BoE** & **XPERT-BoE**: In this ablation, the DistilBERT text embedding model $\mathcal{E}$ was replaced with a simpler BoE (bag-of-embeddings) model from SiameseXML [5]. Note that although overall accuracy does decrease, XPERT-BoE continues to offer similar gains over NP-PER-BoE.
- **Neg-inbatch-hard** & **userwise-batching**: Using in-batch-hard negatives in place of global-hard negatives and user-wise batching instead of item-wise batching caused noticeable drop in metrics suggesting their importance.
- **loss-triplet**: A triplet loss with margin 0.3 instead of a contrastive loss led to a drop in all metrics.

Table 4 reports additional ablations performed on the DPSR method:

- **mean**: Replacing the transformer aggregation (offered to DPSR as an advantage) with mean aggregation as suggested in original paper [33] reduced accuracy consistent with the seg1-mean ablation for XPERT.
- **no channels**: Removing the channel advantage offered to DPSR led to a drop of 2% in recall@100 indicating the utility of channeling irrespective of the (N)P-PER method
- **DPSR***: This variant was offered advantages over the base DPSR method such as additional MLP layers, residual connection, hard-negative mining in addition to the transformer based aggregation as well as channel-based seed event selection. This meant that the most prominent distinction between DPSR and XPERT was the personalization mechanism with XPERT using morphing operators and DPSR using its MLP architecture. DPSR* achieves noticeable gains over DPSR but was still worse than XPERT by 4%, 3% in Recall@100 and MRR@100 respectively.

Two-sided vs One-sided personalization: Given its exorbitant storage costs, two-sided personalization was explicitly implemented on a randomly subsampled subset of the U2A-4M dataset with 10k candidate ads and 150k users. Two-sided version of DPSR method, where both event and item embeddings were personalized achieved a training loss of 0.183 which is significantly smaller than one-sided version's 0.435.

Re-ranking after P-PER: as noted in Section 1, XPERT focuses on improved retrieval via personalization, and other layers in a typical recommendation pipeline such as re-ranking, display-layout are beyond its scope. To gain insights into the benefits of personalization in retrieval, further experimental results are presented using the U2A-4M dataset. Table 5 shows that even when comparing recall at large values of k such as recall@200 or recall@1000, XPERT continues to outperform NP-PER by a large margin i.e. merely increasing the shortlist size cannot overcome the drawbacks of NP-PER. Table 7 indicates that XPERT outperforms NP-PER even if NP-PER results are re-ranked using XPERT itself indicating the irreversible loss of items risked by non-personalized retrieval. Table 6 offers ranking metrics such as precision@k values for XPERT and various competitors where XPERT continues to offer gains.

6 CONCLUSION AND FUTURE WORK

This paper presented XPERT which introduces the concept of morphing operators that offer scalable yet effective personalization. Notably, this enables personalization at the retrieval stage itself without requiring a prior shortlisting to have taken place. The notion of morphing operators opens up several possibilities and whereas the XPERT algorithm only explores linear morphing operators, more powerful non-linear operators must be explored given the substantial improvements in retrieval yielded by XPERT's frugal architecture. XPERT can also be augmented to perform collaborative learning (e.g. via graph convolutional networks) by exploiting relational information in the form of user-user or item-item graphs. Alternate training strategies that improve gains on rarer items would be interesting and ensure that more items enjoy the benefits of P-PER.

REFERENCES

[1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural news recommendation with long-and short-term user representations. In *ACL*.

[2] K. Bhatia, K. Dahiya, H. Jain, A. Mittal, Y. Prabhu, and M. Varma. 2016. The Extreme Classification Repository: Multi-label Datasets & Code. http://manikvarma.org/downloads/XC/XMLRepository.html

[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*.

[5] K. Dahiya, A. Agarwal, D. Saini, K. Gururaj, J. Jiao, A. Singh, S. Agarwal, P. Kar, and M. Varma. 2021. SiameseXML: Siamese Networks meet Extreme Classifiers with 100M Labels. In *ICML*.

[6] Kunal Dahiya, Nilesh Gupta, Deepak Saini, Akshay Soni, Yajun Wang, Kushal Dave, Jian Jiao, Prasenjit Dey, Amit Singh, Deepesh Hada, et al. 2023. NGAME: Negative Mining-aware Mini-batching for Extreme Classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 258–266.

[7] Alan Edelman, Tomas A. Arias, and Steven T. Smith. 1998. The Geometry of Algorithms with Orthogonality Constraints. *SIAM J. Matrix Anal. Appl.* 20, 2 (1998), 303—-353.

[8] F. Faghri, D.-J. Fleet, J.-R. Kiros, and S. Fidler. 2018. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *BMVC*.

[9] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *KDD*.

[10] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*.

[11] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *CIKM*.

[12] Houyi Li, Zhihong Chen, Chenliang Li, Rong Xiao, Hongbo Deng, Peng Zhang, Yongchao Liu, and Haihong Tang. 2021. Path-based Deep Network for Candidate Item Matching in Recommenders. In *SIGIR*.

[13] Qiang Liu, Shu Wu, and Liang Wang. 2015. COT: Contextual Operating Tensor for Context-Aware Recommender Systems. In *AAAI*.

Table 3: Ablation experiments for XPERT on the U2A-4M dataset

Ablation Name	Recall@10	Recall@50	Recall@100	nDCG @10	nDCG @50	nDCG @100	AUC@100	MRR@100
XPERT	14.852	25.778	27.189	9.318	11.790	12.026	23.390	8.212
seg1-mean	12.783	23.088	24.644	7.643	9.969	10.229	20.888	6.628
seg3-no-residual	14.773	24.879	26.002	9.335	11.631	11.819	22.621	8.217
NP-PER-BoE	5.235	12.184	12.712	3.135	4.702	4.791	10.632	2.854
XPERT-BoE	12.339	19.600	21.024	7.612	9.270	9.506	18.120	6.570
neg-inbatch-hard	13.546	24.494	25.942	8.149	10.624	10.865	22.111	7.089
userwise-batching	13.065	23.784	25.340	7.746	10.167	10.428	21.484	6.700
loss-triplet	14.424	24.785	26.081	9.008	11.356	11.573	22.507	7.910

Table 4: Ablation experiments for DPSR on the U2A-4M dataset

Method	Recall@10	Recall@50	Recall@100	nDCG @10	nDCG @50	nDCG @100	AUC@100	MRR@100
DPSR	8.371	17.040	18.352	4.687	6.617	6.836	15.147	4.016
mean	7.806	15.409	16.645	4.379	6.073	6.279	13.763	3.735
no channels	7.385	14.734	15.624	4.186	5.828	5.977	13.074	3.593
DPSR*	10.899	21.522	23.201	6.144	8.533	8.813	19.282	5.267

Table 5: Recall@k values for NP-PER and XPERT for large values of $k = 200, 1000$.

Method	Recall@200	Recall@1000
NP-PER	20.5	31.2
XPERT	**31.9**	**42.5**

Table 6: Precision@K metric for various methods on the U2A-4M and AmazonReviews-1M datasets. Despite being focused on retrieval, XPERT offers superior precision numbers than competing methods.

	U2A-4M dataset		AmazonReviews-1M	
Method	P@1	P@3	P@1	P@3
XPERT	**5.10**	**2.84**	**3.99**	**2.33**
NP-PER	1.70	1.09	0.84	0.95
SUR-DNN	0.48	0.38	0.82	0.85
DPSR	2.10	1.45	2.57	1.55
PinnerSage	1.65	1.21	1.06	1.01

Table 7: Re-ranking after NP-PER vs XPERT on the U2A-4M dataset. 1000 ads were retrieved using NP-PER and re-ranked using XPERT by computing the cosine similarity with personalized embeddings of a user's 10 recent-most events. The top 100 ads were chosen. Even at 10x cost, this retrieve-then-rerank pipeline is outperformed by XPERT all by itself .

Method	Recall@100	P@1
NP-PER -> XPERT	21.9	4.10
XPERT	**27.1**	**5.10**

[14] Zheng Liu, Jianxun Lian, Junhan Yang, Defu Lian, and Xing Xie. 2020. Octopus: Comprehensive and Elastic User Representation for the Generation of Recommendation Candidates. In *SIGIR*.

[15] W. Lu, J. Jiao, and R. Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured Compressed BERT Models for Large-Scale Retrieval. In *CIKM*.

[16] A. Y. Malkov and D. A. Yashunin. 2020. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *TPAMI* (2020).

[17] Tharun Kumar Reddy Medini, Qixuan Huang, Yiqiu Wang, Vijai Mohan, and Anshumali Shrivastava. 2019. Extreme classification in log memory using count-min sketch: A case study of amazon search with 50m products. In *NeurIPS*.

[18] Bhaskar Mitra and Nick Craswell. 2019. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval* 13, 1 (2019), 1–126.

[19] A. Mittal, K. Dahiya, S. Agrawal, D. Saini, S. Agarwal, P. Kar, and M. Varma. 2021. DECAF: Deep Extreme Classification with Label Features. In *WSDM*.

[20] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP*.

[21] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *KDD*.

[22] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. arXiv:1910.14424.

[23] Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. 2020. PinnerSage: multi-modal user embedding framework for recommendations at pinterest. In *KDD*.

[24] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. 2021. GalaXC: Graph neural networks with labelwise attention for extreme classification. In *WWW*.

[25] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

[26] 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 *CIKM*.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*.

[28] Tian Wang, Yuri M Brovman, and Sriganesh Madhvanath. 2021. Personalized Embedding-based e-Commerce Recommendations at eBay. arXiv:2102.06156.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP: System Demonstrations*.

[30] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. Npa: Neural news recommendation with personalized attention. In *KDD*.

[31] Chuhan Wu, Fangzhao Wu, Junxin Liu, Shaojian He, Yongfeng Huang, and Xing Xie. 2019. Neural demographic prediction using search query. In *WSDM*.

[32] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

[33] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In *SIGIR*.