Don't Lose Your Head(s): Model Pre-Training and Exploitation for Personalization

Pablo Zivic
MercadoLibre, Inc.
Buenos Aires, Argentina
pablo.rzivic@mercadolibre.com

Jorge Sanchez
MercadoLibre, Inc.
Buenos Aires, Argentina
jorge.sanchez@mercadolibre.com

Rafael Carrascosa
MercadoLibre, Inc.
Buenos Aires, Argentina
rafael.carrascosa@mercadolibre.com

ABSTRACT

Many tasks in e-commerce involve providing users with relevant and personalized information that reflects their particular interests and affinities. Although tasks such as search and recommendation are commonplace in most large-scale e-commerce platforms nowadays, there is a growing need in providing new features and capabilities, most of which would benefit from personalized user-centric strategies. However, designing custom solutions takes a considerable amount of time and effort, so the availability of generic behavioral representations that can be used in a variety of tasks, is of great practical importance

In this work, we propose a generic feature extraction model that effectively reduces the time and costs of building personalized solutions by leveraging a diverse and general set of tasks suitable to e-commerce. During pre-training, the model predicts attribute embeddings of the next and purchased items in a navigation session. At run-time, instead of discarding the output prediction heads, we use them to build a novel item representation that incorporates user preferences observed in the navigation session.

Equipped with these representations, we perform experiments on two different datasets, demonstrating competing performance and a high degree of complementary with other approaches.

ACM Reference Format:
Pablo Zivic, Jorge Sanchez, and Rafael Carrascosa. 2023. Don't Lose Your Head(s): Model Pre-Training and Exploitation for Personalization. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'23).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Being able to generate personalized responses is key for e-commerce, as it improves the overall user experience by presenting results that align with their interests and affinities. Applications of personalization range from generating customized product recommendations [6, 27] and showing personalized responses to search queries [1, 15], to the prediction of size and fit of fashion [19]. These applications require a way to encode user behavior and preference dynamics concisely. These are variables that are difficult to grasp, and can not be captured from historical data only, e.g. user purchase and/or recommendation history. They are modulated by external factors such as trends, social and seasonal events, etc. [25, 27]. This makes session-based personalization [18] appealing, as sessions account for all these variables implicitly by looking at the actual interaction behavior of users in a continuous period of time [25]. Session-based methods have several advantages over more traditional ones [12], namely: they do not assume the availability of user preference data, they can be readily applied to new and/or anonymous users, and reflect the dynamics of user preferences over time.

Besides the modeling capabilities of these designs, another important factor is *scale*, especially in high-traffic scenarios like large auction platforms/marketplaces, and streaming service providers. In these cases, the large number of items that change over time (items being constantly added or removed) or the real-time nature of the interactions (low latency response), make the actual design face constraints that go beyond downstream performance. Furthermore, given the variety of problems that involve deciding what to present to a specific user in a given context, there is a practical need for developing generic approaches that reduce the effort required by designing custom solutions.

In this work, we tackle these problems and propose a generic learning framework that leverages user navigation data to learn a generic model that allows us to compute session-based representations that perform well on various tasks. We learn the model by leveraging supervision signals well suited for e-commerce applications. Importantly, generating such signals does not require any additional effort since they can be inferred automatically from historic navigation logs.

Our contributions can be summarized in the following:

- We propose a multi-task self-supervised pre-training approach from user activity logs. Our approach is designed to be independent of the size of the product catalog.
- A feature extraction mechanism that leverages the trained sequential model and enables its use across multiple applications. This mechanism can be easily integrated into existing solutions and incorporate problem-specific features in a straightforward manner.
- We show that the model and the new representation can be computed incrementally with a minimum effort, making them suitable for online personalization settings.
- We report experiments on two different e-commerce datasets and show consistent improvements w.r.t a variety of models. We also provide a disaggregate analysis of the session data that provide new insights into the particularities of the problem.

2 RELATED WORK

In this section, we review relevant work on personalization and representation learning in the context of e-commerce. We refer the interested reader to [25] for a thorough survey of the subject.

SIGIR eCom'23, July 27, 2023, Taipei, Taiwan

In [14], the authors propose a user representation learned across multiple tasks. The model consists of an LSTM network and an attention mechanism trained on user sequences that include a variety of interaction events (clicks, bookmarks, purchases). The model is learned over five different tasks. Some tasks are specific to Taobao (shop preference prediction, fashion icon following prediction) while others are relevant to any e-commerce platform (click-through rate prediction, ranking, and price preference prediction). Although the authors demonstrate a gain from fine-tuning to a new task, it is not clear how a machine learning practitioner should incorporate problem-specific features such as placement for ads, or contextual information for push notifications.

Instead of fine-tuning the model for each downstream task, [29] proposes to "patch" the learned model in order to use it downstream. The approach is based on pre-training a convolutional architecture on a large set of user navigation sequences, using a context-based masked item prediction objective (single task). During fine-tuning, the authors propose to insert small sub-networks across the architecture in order to reduce the number of parameters needed to be fine-tuned for a given new task. Note, however, that the convolutional nature of the model makes it difficult to apply to an online/streaming scenario, where incremental computation capabilities are desirable.

[28] use a contrastive learning strategy and different data augmentations to train a transformer architecture from sequence data. In this model, input sequences are transformed by two randomly sampled augmentations and the model has to predict if a pair correspond to the same underlying sequence or not. The negative pairs correspond to other randomly sampled sequences from the training dataset. The authors focus exclusively on the recommendations, and it is not clear how could be applied to other e-commerce tasks.

[11] adapts the BERT model [5] to e-commerce data by self-supervision. The model adopts an architecture that explicitly differentiates between short- and long-term interactions. This bidirectional transformer model is trained using self-supervision on different masked-language modeling tasks specially formulated for e-commerce.

[21] trains a recurrent model by applying two different augmentation strategies, namely: considering all prefix sequences with a training instance as new samples and applying random item deletions across training sequences. The use of recurrent models for session-based recommendation allows for incremental data processing. This is an important aspect in online and streaming scenarios [25].

A particularity of most of the models described above is that they are trained on some variation of the "next-item prediction" task. This task is usually formulated as a classification problem over the items in the catalog. In an e-commerce setting, this number can be too large from a practical standpoint (number of parameters in the last layer of the model). Moreover, with new items being added and removed constantly, the effectiveness and traceability of such a system over time becomes an issue. An alternative formulation, proposed by [21] is to reformulate the problem as an embedding regression task. In this way, the output of the model becomes independent of the number of entries in the catalog. We take this approach and extend it across different tasks in a unified manner.

Lastly, none of the described models provide a mechanism to incorporate problem-specific features when using them for new downstream tasks. That means that all downstream tasks must rely solely on the user session and not on other features that may be relevant to the specific problem [17, 20, 25].

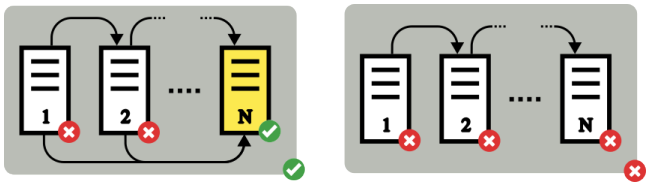

Figure 1: Schematic view of the auxiliary tasks for a session that ends in a purchase (left) and a session that does not (right): next item attributes (upper arrows), purchased item attributes (bottom arrows), an item being purchased (✓) or not (✗), the session ends in a purchase or not. We say a session is a purchase if the last event in the session is a purchase.

3 PROBLEM SETTING

Let $\mathcal{Z}$ be the set of items[1] available to the user. An item $z \in \mathcal{Z}$ can be described by a collection of M different attributes $a(z) = \{a_i(z)\}_{i=1}^{M}$, e.g. its price, title, product description, etc. Let $\mathcal{Q}$ be the set of *events* that capture the different ways a user can interact with an element in $\mathcal{Z}$, e.g. viewing it, buying it, marking it as a favorite, etc. A user session can be modelled as a sequence $S = \{(z_i, q_i)\}_{i=1}^{N}$, with $(z, q) \in \mathcal{Z} \times \mathcal{Q}$ of item-event pairs. We use (z, q) or $q(z)$ instinctively to denote the event q acting on item z.

In our work, we aim at learning a representation that is efficient (can be computed incrementally) and generic (exhibits good off-the-shelf performance on a variety of tasks), and which encodes information about the behavior of users exploring $\mathcal{Z}$ based on their preferences.

We focus on two types of events that are particularly relevant in our setting: viewing an item and/or buying it. In what follows, we present a model and learning formulation that account for these goals.

4 MULTI-TASK PRE-TRAINING OVER USER NAVIGATION DATA

Let us assume the availability of a large set of user session data $\mathcal{S} = \{S_i\}_{i=1}^{n}$. As mentioned above, we restrict the event set to $\mathcal{Q} = \{q_{view}, q_{buy}\}$. Let us further assume an item $z \in \mathcal{Z}$ can be encoded based on any given attribute by a mapping $\phi_a : \mathcal{Z} \to \mathbb{R}^d$, $a \in \{a_1, \ldots, a_M\}$, i.e. embeddings for the publication title, price and/or product description. These representations are item-centered and capture generic knowledge about the type of data (modality) being represented. They do not account for anything related to user behavior during navigation. We propose to train a model on top of these representations in order to capture the dynamics induced by the user during a navigation/shopping session.

We choose a recurrent network (GRU) [3] as our base model. This choice is motivated by the following. First, predictions have to be carried out on-the-fly (as the user interacts with the system) and fast (low latency). In this context, a representation that can be

[1]While a *product* represents a manufactured object or commodity, an *item* depicts something being sold at a given price and conditions.

computed incrementally is thus desirable. Second, user sessions can be arbitrarily long and spread over time. Having to store the whole data for each user and session becomes impractical.

To train our model we define a series of tasks tailored to the type of events we mentioned above. Concretely, given a large collection of user session data, we train a model to predict if an item is being viewed and/or purchased. At a given time step, the query item can be the current, the next, or the last one along the sequence. If the last element in the sequence is being purchased, we say the whole session corresponds to a purchase event. Identifying these types of sessions early is important from a business perspective. Sessions and events types are shown schematically in Fig. 1.

Regarding view events, instead of trying to predict the id of the following item in a navigation session, we seek to predict each of its attributes. The rationale behind this approach over the more usual of casting the problem as a classification task is as follows. First, many items in e-commerce correspond to the same product being sold by different users, each of which has a different identifier. Casting the problem as a classification over the set of item ids not only makes the number of free parameters grow but also makes training such a model difficult due to the long-tailed distribution of item interactions, i.e. most items in the catalog would not have enough samples to properly train the classification layer. Our formulation, on the other hand, is independent of the catalog size and it can be easily adapted to accommodate new attributes and tasks.

The set of pre-training tasks and associated losses our model is trained on will be discussed in the sections below. The tasks correspond to predicting whether the current item is purchased or not, whether the session ends in a purchase or not, the attributes of the next item viewed by the user, and, if applicable, the attributes of the item being purchased.

In what follows, we use $h_i = g(\phi(z_i))$ to denote the hidden state of the recurrent network g at the i-th step of the sequence.

4.1 (This) Item Purchase

Given an item z we would like to predict if the associated event is a purchase or not. This task is posed as a binary classification problem with the loss:

$$\mathcal{L}_{buy\ item} = \sum_{S \in \mathcal{S}} \sum_{i=1}^{|S|} \ell_{ce}(f_1(h_i), q_i) \tag{1}$$

where $f_1 : \mathbb{R}^d \rightarrow Q$ is a classifier and $\ell_{ce}(\cdot, \cdot)$ the cross entropy loss.

4.2 Session Purchase

This is similar to the above but instead of predicting if the current item is being purchased, we try to anticipate if the session will end on a purchase based on the navigation history up to the current step.

$$\mathcal{L}_{buy\ session} = \sum_{S \in \mathcal{S}} \sum_{i=1}^{|S|} \ell_{ce}(f_2(h_i), q_{|S|}). \tag{2}$$

Here, $q_{|S|}$ denotes the event associated with the last item in S and f_2 a classifier as before.

4.3 Next Item Attributes

Given an item z, we aim at predicting the attributes (their embeddings) of the next item in the sequence. The loss term associated with attribute a_m can be written as:

$$\mathcal{L}_{a_m} = \sum_{S \in \mathcal{S}} \sum_{i=1}^{|S|-1} \ell_a(f_{a_m}(h_i), a_m(z_{i+1})). \tag{3}$$

where $a_m(z_{i+1})$ is the m-th attribute of the $(i+1)$-th item.

The overall loss for this task, $\mathcal{L}_{view\ next}$, is the combination of the following partial terms:

- Item title, $\mathcal{L}_{title}$: regression to the average word embedding of the next item publication title. We use the cosine loss, i.e. one minus the cosine similarity between the attribute embedding and its prediction, as the regression loss for this task.
- Product ID, $\mathcal{L}_{id}$: same as before, but regressing towards an embedding of the product ID. Note that different items can be mapped to the same ID, e.g. same product offered by different sellers. By regressing towards (an embedding of) the product ID, we make the model independent of the number of products in the catalog [3], a number that can be rather large and change over time.
- Item price, $\mathcal{L}_{price}$: regression towards the log of the item price using a mean square error loss. We also explore a variation of this term based on the classification of the "price bucket" in which the product falls. In this case, the size and number of buckets can be set from statistics computed over the training set or guided by the application, e.g. pre-defined price ranges.

4.4 Purchased Item Attributes

This task is the same as the above but instead of predicting the attributes of the next item in the sequence, we try to predict those of the one being purchased. In this case, we consider only sequences that end in a purchase. Eq. (3) can be modified as follows:

$$\mathcal{L}'_{a_m} = \sum_{S \in \mathcal{S}} \mathbb{1}_{q_{|S|}=q_{buy}} \sum_{i=1}^{|S|-1} \ell_a(f'_{a_m}(h_i), a_m(z_{|S|})). \tag{4}$$

where $\mathbb{1}_p$ is an indicator variable that takes the value one if p y true and zero otherwise. As before, the overall loss for this task, $\mathcal{L}_{buy\ last}$, will be the sum of the $\mathcal{L}'_{a_m}$s corresponding to the attributes of the purchased item.

There are many ways in which we can aggregate Eqs. (1)–(4) into a single loss to be optimized. Aggregation methods vary both in their technical details as well in the intuitions they rely on. For linear aggregation, losses can be weighted by considering each task uncertainty, learning speed, or per-task performance [4]. Other methods balance losses by modifying gradients by imposing constraints, doing re-normalization, or performing gradient surgery to avoid interference between tasks [31]. However, results are mixed and while some of these techniques lead to improvements in the final performance, results depend not only on the task but also on the specifics of the architecture and model design [23, 30].

In light of these results, we opt for the following strategy. We combine losses linearly. We first balance each term by multiplying it by a constant value so that all loss terms have roughly the same magnitude. During training, we follow [9] and learn importance weights along with the parameters of the model. In preliminary

experiments, we observed this automated approach leads to a very similar pre-training performance compared to the case of setting these values manually using cross-validation.

5 TOWARDS PERSONALIZED ITEM REPRESENTATIONS

Training a model on the tasks described above allows us to capture important yet generic user behavior patterns from data. We can think of these representations as user-generic as they originate from learning from a large collection of real user interactions within an e-commerce context. Some applications, however, would benefit from building a more personalized characterization of the user, based on their particular interests.Examples are search personalization, where we have to reorder search results in relation to the recent navigation behavior, or push notification selection, where we have to select a set of candidates, dates, and times to send a push notification to a user.

To build such representations, first note that most prediction heads in our model correspond to regression heads, i.e. their output account for the model's best guess for the attribute embeddings of the next/purchased item in a navigation sequence. We leverage such predictions and build a representation that contrasts the predicted embeddings with those of the attributes of each item in a (short) list of candidates. Formally, let $f_{m,i} = f_{a_m}(h_i)$ denote the embedding predicted by the model for attribute a_m at the i-th step. Let z_k denote the k-th product in a list of candidates (e.g. items returned by an initial search query) and consider the following set of descriptive statistics:

$$\max_{i=1,\dots,n} s(f_{m,i}, \phi_{m,k}) \tag{5}$$

$$\frac{1}{n}\sum_{i=1}^{n} s(f_{m,i}, \phi_{m,k}) \tag{6}$$

$$s(f_{m,n}, \phi_{m,k}) \tag{7}$$

where $\phi_{m,k} = \phi_m(p_k)$ is the representation for attribute m of product z_k and $s(A, B)$ denotes a similarity measure between A and B, e.g. cosine similarity for vector embeddings or 1 minus a distance for scalar attributes (e.g. price).

Eq. (5)–(7) compute simple descriptive statistics that seek to capture generic patterns from the items the user has recently interacted with, in relation to a candidate item. Eq. (5) computes the maximum similarity of the current model prediction with all the items in the navigation history. Eq. (6) aims at capturing z_k's average relevance to the current session. Finally, (7) computes the similarity between z_k and the prediction for the last item visited by the user.

In this context, the computation of $s(f_{m,i}, \phi_{m,k})$ can be thought of as a simple attention mechanism with item-to-item similarity at its core, and can retrieve properties about the relationship between a given candidate and all past interactions. This formulation can be extended with more problem-dependant statistics as long as the final vector has a fixed size.

All these coefficients can be computed efficiently with a minimum of caching, i.e. by maintaining a running max and running average of the similarities. Using the regression heads presented in Sec. 4 we compute a total of 18 coefficients: 9 correspond to heads predicting the next event, and 9 to session purchases. In both cases, we extract

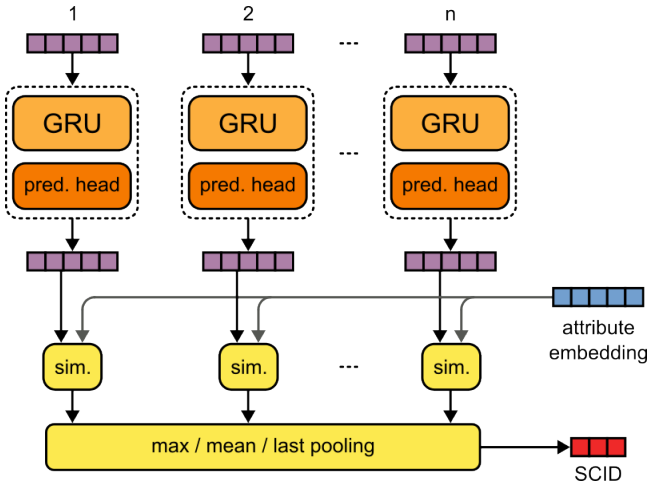

Figure 2: Computing SCID features. We take advantage of the prediction heads for pre-training and compute similarity statistics with (the attributes of) a candidate item (in blue). Each attribute provides us with a set of coefficients (in red) that we stack into a vector to form a personalized representation for that item.

3 features for each of the 3 attributes of the item. This is a compact representation that can be computed for any new item given a user session.

We term this representation *session conditioned item descriptor* (SCID). Figure 2 illustrates the processes to compute the SCID coefficients for a given attribute and a session of length n. In the experiments, we also consider replacing the GRU with a simple moving average model, while pre-training using the same heads and tasks configurations. The computation of this SCID variant remains the same.

6 EXPERIMENTAL SETUP

In this section, we present our experimental setup, show results on different datasets, and discuss our findings.

6.1 Datasets and Tasks

We run experiments on two different datasets: Coveo Data Challenge (CDC) [20] and MeLi-Sessions.

CDC is a publicly available session-level dataset reflecting 10M interactions over 57K different products in an e-commerce website. It accounts for more than 4M user sessions. Interactions in this dataset include *add to cart*, *remove from cart*, *view*, *purchase*, and *search* events. The associated challenge considered two different sets of tasks: session-based recommendation and cart abandonment. We focus on the former, i.e. given the first n elements of a session, the task is to predict future interactions. We consider the following problems: *next item prediction* and *search personalization*. For the first, we set the target as the last item in each sample sequence. For the second, we use the available annotations and ask the model to predict the item clicked by the user from the list shown after a search query. In both cases, we use mean reciprocal rank (MRR)

Property	CDC	MeLi-Sessions
# of unique items	57483	4036255
# of sessions	4934699	1019704
# of interactions	10431611	14924167
# of purchases	77848	78235
25/50/75 pct. of session length	2/3/8	7/12/21
Language	English	Spanish

Table 1: Statistics for the training set of the CDC and MeLi-Sessions datasets.

as the performance metric. Preliminary results showed that other ranking metrics such as NDCG, precision, and recall at k were highly correlated with MRR, thus we decided to report only MRR for simplicity.

MeLi-Sessions is an in-house dataset that we use to evaluate different scenarios relevant to our application domain. It consists of 1M user sessions over more than 4M items. Interactions include view and purchase events. Samples in the dataset were collected between April 2021 and July 2021. Sessions in this dataset are longer than CDC, with a median of 12 interactions (vs. 3 for CDC). For evaluation, we consider the following tasks: *search personalization*, *session-based recommendation*, and *product ads*. The recommendation task is similar to search personalization with the difference that the trigger is an item instead of a search query. The list of candidates, in this case, is obtained from co-occurrence statistics w.r.t the item we use as trigger. As before, we use the MRR as our metric. The *product ads* task corresponds to estimating the conversion rate of a sponsored item. In this case, we use the area under the ROC curve as our metric.

Table 1 shows summary statistics computed on the training set of both CDC and MeLi-Sessions. From the table, we see that although the number of session samples in MeLi-Sessions is four times smaller than CDC, sessions are longer and the number of unique items is considerably larger. Also, an important aspect of MeLi-Sessions is that it accounts for e-commerce tasks in Spanish.

6.2 Architecture and Training Details

We split the datasets chronologically into two disjoint subsets. For CDC, we use sessions occurring before May 20th for pre-training (~80% of the total) and those after this date for evaluation, unless specifically stated otherwise. We follow similar criteria for the MeLi-Sessions dataset. We use data from April to July for pre-training and the subsequent month for evaluation. In both cases, there is no overlap between train and test subsets, and the test sets are posterior to the training sets in each case.

We pre-train a GRU network (single layer, 256-dimensional hidden state) using the losses described in Sec. 4. Prediction heads consist of two-layer MLPs with ReLU non-linearities for CDC and single-layer MLPs for MeLi-Sessions. (Pre-)training is run using a learning rate of 0.001 and the Adam optimizer. We use an early stopping heuristic with a *patience* parameter of 4. As mentioned before, we follow [9] and combine Eqs. 4.1–4.4 linearly and learn the importance weights along with the model parameters. During preliminary experiments, we observed no significant difference from setting these weights manually by cross-validation.

7 EXPERIMENTS

In the following, we present and discuss the results for the different downstream tasks for both datasets.

7.1 Experiments On CDC

In this section, we show experimental results and discussions on the CDC dataset.

7.1.1 Item and Attribute Encoders. We build a representation using the embeddings provided with the dataset. We concatenate the description and category embeddings, adding the normalized price range as an additional scalar feature. Description embeddings correspond to the ones provided with the dataset for textual metadata. We compute category embeddings as the average description embedding for the items in each category.

7.1.2 Model Pre-Training. We adapt the attribute prediction tasks in Sec. 4.3-4.4 to those available in the dataset. Price estimation is approached as a 10-way classification problem using the provided annotations. Instead of title embeddings, we use embeddings of the item description provided by the authors of the dataset. Finally, instead of category IDs, we regress towards an embedding of the product category computed as the average of the description embeddings for the items of each category. We pre-train our model as described above and use the trained model to extract the features that feed a downstream predictor. We do not perform any task-specific fine-tuning, i.e. we freeze the model weights and use it as a simple feature extractor.

7.1.3 Next Item Prediction. For this task, we train a gradient boosting model as implemented in LightGBM [8] on top of different feature combinations. The use of LightGBM offers numerous advantages in a production environment such as avoiding the need for feature normalization, outlier handling, and explicit treatment of missing values and categorical feature encoding. We consider the following features: a baseline (BL) consisting of the concatenation of the item description embedding provided by the authors of the dataset and an additional feature that encodes the expected click-through rate, given the ranking position within a list of candidates derived from co-counts, a simple sequential model consisting of a moving average (MA) of input embeddings, using the hidden state of the GRU network (GRU), and two variations of the SCID features computed using either a moving average or GRU network as the base sequential model (SCID). Feature combinations consist of simple concatenations, without any normalization or pre-processing. We show disaggregated results for cases in which the target item was already seen by the user during navigation and cases in which it has not. These two cases reflect different expected *behaviors*. In the first case, the model should provide a confirmation of the user interests as shown during navigation (e.g. by revisiting the same item multiple times) while in the second, the model should bring to the user's attention a more diverse set of items that match hers/his immediate interests. We report MRR scores for each case. For these

experiments, we report results in the challenge test set[2]. We use the last item of each sequence as the target item for that session. Although the evaluation setup might differ from that used during the competition, it provides us with a sensible reference.

When reporting results using SCID features (first block in the table), we show two different scores. The first corresponds to the case of SCID features computed using GRU embeddings, and the second to SCID features computed based on moving average embeddings.

Results are shown in Table 2. From the table, we see that combining either MA or GRU with the BL features gives a slight but consistent improvement. If we consider the combination of BL with either of the SCID features (based on MA or GRU models), we observe an additional improvement mostly coming from sessions in which the target item was already seen. The improvement brought by adding SCID is higher for the GRU-based model. If we compare the effect of adding the GRU-based SCID to the BL baseline, the relative improvement for the "Seen" case is 9.6% while for the "Not Seen" case is only 1.2%. This can be attributed to the fact that when the target item was seen during navigation, its similarity with the items in the navigation session is a strong indicator of user interest, i.e. the max similarity statistic is a maximum (for a cosine similarity) and the average similarity increases with the number of times the user visited the target item during navigation. From the table, we also observe little improvement for the combination of more than 2 descriptors.

As a reference, we show results obtained by the top-performing teams during the competition. Teams `DeepBlueAI` and `NVIDIA Merlin` [13] obtained almost the same results. The first relied on complex heuristics and the former used a transformer-based solution. The `tsotfsk` team [10] relied on a graph-based approach and learned node embeddings over interaction graphs. Although the results are not strictly comparable, we observe that our approach is highly competitive on average.

7.1.4 Search Personalization. We leverage the data provided with the CDC dataset and evaluate the performance of our model in search personalization. We use the data before May 20th for training and validation and the rest for testing. We consider only sessions containing search events and crop them so that they are the last in the sequence. For each such entry, we are also provided with a list of candidates and the item being clicked. We approach this problem as a binary classification problem and use the classifier's score to rank the items in the candidate list. Table 3 show the results for different combinations of input embeddings. Since all entries in this dataset are already processed and embedded, methods such as TF-IDF or BM25 [16] cannot be applied. However, we observed that shortlists in the dataset are already sorted by their relevance w.r.t the query. If we compute the MRR score of the data as is, we observe a score of 0.469. This score drops to 0.350 if we shuffle the list of candidates for each query. We include this default CDC baseline for reference. We also include a custom baseline model built from the concatenation of the description and query embeddings along with the price of the query item. As before, we show disaggregate results for sessions in which the item clicked by the user was previously seen or not.

[2]Challenge organizers kindly provided us with the test data. Since target labels were not available, we followed their recommendation and used the last item in each session as the prediction target.

BL	MA	GRU	SCID	Next Item Prediction		
				All	Seen	Not Seen
✓				0.279	0.490	0.234
✓	✓			0.281	0.493	0.235
✓		✓		0.281	0.495	0.235
✓			✓	0.286/0.288	0.534/0.537	0.233/0.235
✓	✓		✓	0.287	0.534	0.234
✓		✓	✓	0.289	0.540	0.235
DeepBlueAI / NVIDIA				0.277	-	-
tsotfsk				0.271	-	-
scitator				0.228	-	-

Table 2: Results for the *next item prediction* task on the CDC dataset, as measured by the MRR metric. For SCID, we show results for the descriptors built on top of MA/GRU models. Results were computed on the challenge test data using the last item as the target. We also show results obtained by the top-performing teams during the challenge. These results are not disaggregated within the seen/not seen categories since these were obtained from the challenge leaderboard.

MA	GRU	SCID	Search		
			All	Seen	Not Seen
✓			0.456	0.478	0.443
	✓		0.463	0.488	0.447
		✓	0.576/0.605	0.734/0.800	0.477/0.483
✓		✓	0.570	0.730	0.470
	✓	✓	0.602	0.798	0.480
Baseline (CDC default)			0.469	0.494	0.452
Baseline (ours custom)			0.449	0.465	0.439

Table 3: *Search personalization* results on the CDC dataset. Results are shown for a baseline model, moving average, and GRU-based models, both variants of SCID, and their combination.

From the table, we see that GRU performs on par with MA, showing an improvement for sessions where the target item has already been seen. All feature combinations perform better than the baseline model alone. The performance of MA and GRU models gets surpassed by a large margin by either of the corresponding SCID. In this case, the model that uses a GRU as the base sequential model shows the best performance overall. It brings relative improvements of 63.9% and 8% for the "Seen" and "Not Seen" cases, respectively, over the GRU-only counterpart. In this case, in contrast to the *next item prediction* task, we see an increase in performance for sessions that do not include the target item. When combined, either MA+SCID or GRU+SCID seems to saturate over their SCID variants alone, showing the expressiveness of the SCID on this task. Also, besides the large gain brought by the SCID, we observe that CDC default is a competitive baseline compared to both our sequential variants (MA and GRU).

7.2 Experiments on MeLi-Sessions

In this section, we show experimental results and discussions on the MeLi-Sessions dataset.

7.2.1 Item and Attribute Encoders. We use a concatenation of FastText [2] and Meta-Prod2Vec [24] embeddings as input to our model for pre-training. They encode the item's title and product, respectively. These embeddings were trained internally on historical data collected previous to the construction of the dataset. Once we have a pre-trained model on session data, we use it as a monolithic feature extractor as before, i.e. without performing any task-specific fine-tuning.

7.2.2 Search Personalization and Session-based Recommendation. We approach both of these problems by training a binary classifier to predict the item clicked by the user from the list of candidates shown after a search query (Search) or retrieved using co-occurrence statistics with the last item the user interacted with (Recommendation). We use the classification score as a relevance measure to rank the items in this list. This model is trained on top of different feature combinations, and results are shown in the "Search" and "Recommendation" groups of Table 4. We consider a set of problem-specific features (BL) used internally for each task (concatenation of the embeddings for the search query / last item, and the item being ranked in the candidate list) and MA, GRU, and SCID features as before. We also consider different combinations by concatenation and show disaggregate results for the search and recommendation tasks.

When comparing the performances of individual models, we see the problem-specific strategy without personalization performs on par (search) or better (recommendation) than the sequential models alone. Although there seems to be no difference between MA and GRU when considered alone, the SCID descriptors computed from them observe a clear improvement, with the GRU-based model performing better than the MA-based one. For the SCID features, if we consider the overall metrics ("All"), we observe a marked improvement for both descriptor variants in search and recommendation. The performance gain observed on these tasks is due mainly to a large gain in sessions for which the target has already been seen. For sessions in which the target was not previously seen, there is a noticeable decrease in performance, possibly due to the intrinsic difficulty of the task. The good complementarity between SCID and the other representations allows us to recover and boost this loss by a margin. We observe that, when combining the BL features with either MA or GRU, there is no gain in the overall performance. It is only after combining these models with the SCID that we observe a large increase on all sides, especially for the GRU-based SCID. These improvements are more noticeable for sessions in which the target item has already been seen. For instance, in the search personalization task, while there is a 21% relative improvement after adding the SCID features to the BL+MA combination, the improvement is 30% for sessions in which the target has been seen and 1.6% for those in which has not. For session-based recommendation, the overall/seen/not seen improvements are 14.6%/16.1%/4.5%, respectively. For the combination of BL+GRU, the improvements after adding the SCID features are even more noticeable: 28.8%/45.2%/4.4% for search and 21.2%/23.8%/4.1% for recommendations.

7.2.3 Product Ads. This task is similar to session-based recommendation, but where the candidates originate from a list of advertised products. The goal here is to predict if an item within this list was finally bought or not. The last column of Table 4 shows the results using the ROC-AUC metric. As before, we consider a baseline model using problem-specific features describing the placement and past performance of the ad, and two models that combine session embeddings with the SCID features from Sec. 5. We observe similar trends as before but with lower relative improvements overall. In this case, we do not show disaggregated results since we observed very similar behavior in both cases. Interestingly, although there is a switch in the best-performing SCID model when considered alone, the GRU-based SCID seems to exhibit better complementarity compared to the one based on a moving average, showing a 2.7% relative improvement compared to BL alone or combined with either of the sequential models.

7.3 Qualitative Analysis

Figure 3 show qualitative examples of the effect of our model on search personalization for the query "*bts kpop*" (left) and "camperas de abrigo dama" (lady coat jackets, right) in the MeLi-Sessions dataset. The top row shows the last 5 items the user interacted with, with the most recently viewed on the right. The middle and bottom rows show the top-5 results as ranked (ordered in decreasing order from left to right) by the baseline model and our BL+GRU+SCID, respectively. The item clicked by the user is highlighted in green. In both cases, we can observe that the results provided by our model are more consistent with the user navigation history, regarding both product type and price range. In the first example, results shown by the baseline, although relevant to the search query, ignore the user intent as reflected in hers/his recent navigation history. In the second, the price range of the top-ranked results by the baseline is one order of magnitude higher than what the user actually searched for.

8 CONCLUSIONS

In this paper, we proposed a multi-task feature learning and extraction model in the context of e-commerce. Our approach is based on the pre-training of a sequential model using a set of auxiliary tasks relevant to our application domain, for which supervision is easily obtained from user navigation logs.

Instead of training a set of prediction heads that are subsequently discarded, we leverage them and for each new item, we compute a discriminative descriptor that shows consistent performance improvements w.r.t other alternatives on a variety of tasks. The presented approach allows its application on a broad set of e-commerce problems using standard modeling techniques, thus lowering the difficulty of deploying session-based solutions.

Our analysis considers two different scenarios that account for two different user navigation/shopping behaviors, i.e. sessions for which the target item was already seen by the user and sessions for which it has not. We observe that the second is the more challenging case, as it calls for a greater degree of diversity of the model responses, as there is no proxy (item revisits) that helps the model to succeed on the final predictions. However, the former option is relatively easier, which raises the question of whether it may introduce a bias in the training procedure towards simpler solutions.

BL	MA	GRU	SCID	Search			Recommendation			Product Ads
				All	Seen	Not Seen	All	Seen	Not Seen	All
✓				0.400	0.485	0.306	0.397	0.401	0.375	0.724
	✓			0.403	0.489	0.307	0.389	0.392	0.366	0.647
		✓		0.403	0.489	0.307	0.389	0.392	0.366	0.649
			✓	0.507/0.559	0.730/0.826	0.260/0.264	0.433/0.476	0.455/0.500	0.297/0.321	0.646/0.634
✓	✓			0.402	0.486	0.310	0.397	0.401	0.374	0.724
✓		✓		0.402	0.487	0.307	0.397	0.401	0.376	0.724
✓	✓		✓	0.523 (+30.0)	0.710 (+47.9)	0.315 (+1.6)	0.455 (+14.6)	0.465 (+16.1)	0.391 (+4.5)	0.738 (+1.9)
✓		✓	✓	0.565 (+40.5)	0.783 (+62.5)	0.324 (+5.5)	0.482 (+21.2)	0.496 (+23.8)	0.392 (+4.1)	0.743 (+2.7)

Table 4: Search personalization, session-based recommendations, and product ads results on the MeLi-Sessions dataset. Results are shown for a baseline model, moving average, and GRU-based sequential models, both variants of SCID, and their combination. Relative improvements after incorporating the SCID descriptors to the combination of baseline and MA/GRU are shown between parentheses. We use the MRR metric for the search and recommendation tasks and ROC-AUC for product ads.

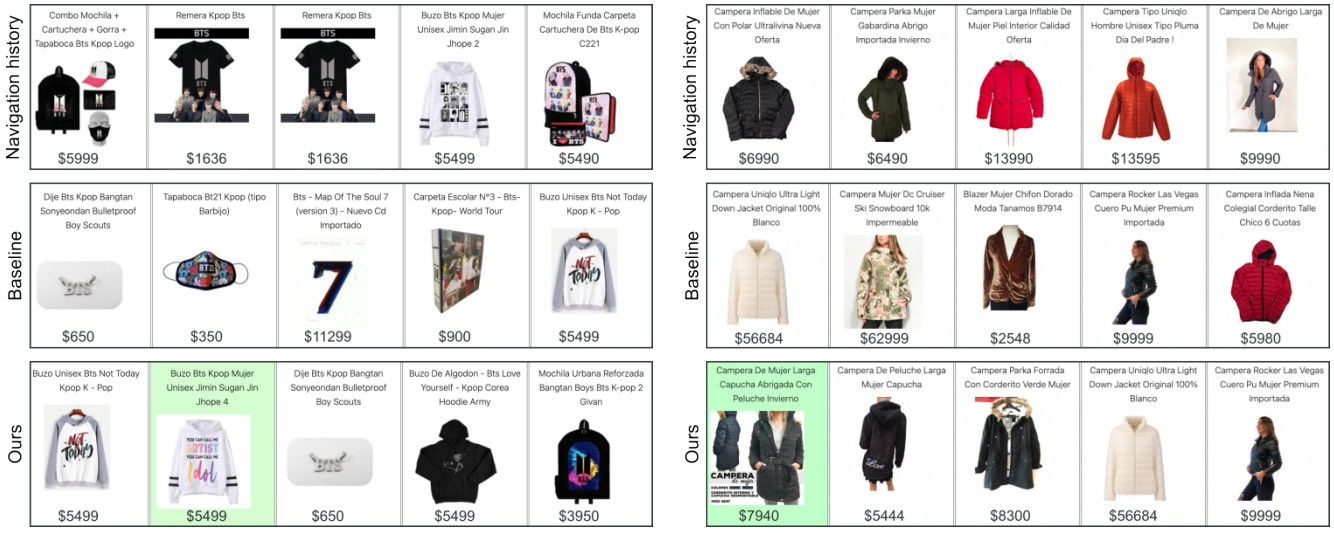

Figure 3: Qualitative example on the search personalization task for the queries "bts kpop" (left) and "camperas de abrigo dama" (*lady coat jackets*) (right). The figure shows the navigation history (top), the results shown by the baseline system (middle), and the results of our approach (bottom). The clicked (target) item is highlighted in green.

The findings and insights gained from this work can serve as a foundation for further advancements in the following directions: *a) parametrizing SCID statistics*, results have demonstrated the importance of SCID in session-based recommendation systems. Moving forward, future research can focus on parametrizing SCID, enabling its learning instead of relying on heuristics; *b) comparing with efficiency-optimized transformer architectures*, in this study, we examined our GRU model's performance in session-based recommendation. Future research can explore different optimized transformer architectures like Block-Recurrent transformer [7], Linformer [26], and Sinkhorn Transformers [22] for assessing computational complexity and predictive performance trade-offs; and *c) incorporating long-range signals and user context*, the integration of long-range signals, such as a user's purchase history, can further improve accuracy beyond users' immediate interests.

ACKNOWLEDGMENTS

We thank the authors of the CDC dataset for providing us with the data and guidelines for the experiments of Sec. 7.1.

REFERENCES

[1] Qingyao Ai, Daniel N Hill, SVN Vishwanathan, and W Bruce Croft. 2019. A zero attention model for personalized product search. In *Proc. of the 28th ACM intl. conf. on Information and Knowledge Management*. 379–388.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[3] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proc. of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.

[4] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proc. of the 12th ACM conf. on Recommender Systems*. 451–454.

[7] DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. *arXiv preprint arXiv:2203.07852* (2022).

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. of the IEEE conf. on computer vision and pattern recognition*. 7482–7491.

[10] Kaiyuan Li, Pengfei Wang, and Long Xia. 2021. A Session-aware DeepWalk Model for Session-based Recommendation. *SIGIR Workshop On eCommerce* (2021).

[11] Tianyu Li, Ali Cevahir, Derek Cho, Hao Gong, DuyKhuong Nguyen, and Bjorn Stenger. 2022. UserBERT: Modeling Long-and Short-Term User Preferences via Self-Supervision. *arXiv preprint arXiv:2202.07605* (2022).

[12] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 31, 1 (2021), 149–181.

[13] Gabriel de Souza P Moreira, Sara Rabhi, Ronay Ak, Md Yasin Kabir, and Even Oldridge. 2021. Transformers with multi-modal features and post-fusion context for e-commerce session-based recommendation. *arXiv preprint arXiv:2107.05124* (2021).

[14] 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 *Proc. of the 24th ACM SIGKDD intl. conf. on Knowledge Discovery & Data Mining*. 596–605.

[15] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proc. of the Eleventh ACM conf. on Recommender Systems*. 130–137.

[16] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[17] Michail Salampasis, Alkiviadis Katsalis, Theodosios Siomos, Marina Delianidi, Dimitrios Tektonidis, Konstantinos Christantonis, Pantelis Kaplanoglou, Ifigeneia Karaveli, Chrysostomos Bourlis, and Konstantinos Diamantaras. 2023. A Flexible Session-Based Recommender System for e-Commerce. *Applied Sciences* 13, 5 (2023), 3347.

[18] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *Proc. of the 1st ACM conf. on Electronic commerce*. 158–166.

[19] Abdul-Saboor Sheikh, Romain Guigourès, Evgenii Koriagin, Yuen King Ho, Reza Shirvany, Roland Vollgraf, and Urs Bergmann. 2019. A deep learning system for predicting size and fit in fashion e-commerce. In *Proceedings of the 13th ACM conference on recommender systems*. 110–118.

[20] Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Bingqing Yu, Patrick John Chia, Federico Bianchi, and Giovanni Cassani. 2021. SIGIR 2021 e-commerce workshop data challenge. *arXiv preprint arXiv:2104.09423* (2021).

[21] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proc. of the 1st workshop on deep learning for recommender systems*. 17–22.

[22] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*. PMLR, 9438–9447.

[23] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. 2021. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3614–3633.

[24] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proc. of the 10th ACM conf. on recommender systems*. 225–232.

[25] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[26] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[27] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2020. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395* (2020).

[28] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th intl. conf. on data engineering (ICDE)*. IEEE, 1259–1273.

[29] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proc. of the 43rd intl. ACM SIGIR conf. on research and development in Information Retrieval*. 1469–1478.

[30] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. 2021. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision* 129 (2021), 3069–3087.

[31] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* 34, 12 (2021), 5586–5609.