Integrating Optimized Item Selection with Active Learning for Continuous Exploration in Recommender Systems

Serdar Kadioglu
serdar.kadioglu@fmr.com

Fidelity Investments (United States)
Bernard Kleynhans
Fidelity Investments (United States)
Xin Wang
Fidelity Investments (United States)

Research Article

Keywords: Recommendations, Exploration-Exploitation, Active Learning

Posted Date: August 25th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3280704/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Annals of Mathematics and Artificial Intelligence on April 5th, 2024. See the published version at https://doi.org/10.1007/s10472-024-09941-x.

Integrating Optimized Item Selection with Active Learning for Continuous Exploration in Recommender Systems

Serdar Kadıoğlu[1,2*], Bernard Kleynhans[1] and Xin Wang[1]

[1]AI Center of Excellence, Fidelity Investments, Boston, USA.
[2]Dept. Computer Science, Brown University, Providence, USA.

*Corresponding author(s). E-mail(s): serdar.kadioglu@fmr.com;
Contributing authors: bernard.kleynhans@fmr.com;
xin.wang@fmr.com;

Abstract

Recommender Systems have become the backbone of personalized services that provide tailored experiences to individual users, yet designing new recommendation applications with limited or no available training data remains a challenge. To address this issue, we focus on selecting the universe of items for experimentation in recommender systems by leveraging a recently introduced combinatorial problem. On the one hand, selecting a large set of items is desirable to increase the diversity of items. On the other hand, a smaller set of items enables rapid experimentation and minimizes the time and the amount of data required to train machine learning models. We first present how to optimize for such conflicting criteria using a multi-level optimization framework. Then, we shift our focus to the operational setting of a recommender system. In practice, to work effectively in a dynamic environment where new items are introduced to the system, we need to explore users' behaviors and interests continuously. To that end, we show how to integrate the item selection approach with active learning to guide randomized exploration in an ongoing fashion. Our hybrid approach combines techniques from discrete optimization, unsupervised clustering, and latent text embeddings. Experimental results on well-known movie and book recommendation benchmarks demonstrate the benefits of optimized item selection and efficient exploration.

Keywords: Recommendations, Exploration-Exploitation, Active Learning

1 Introduction

Recommender Systems have become central in our daily lives and are widely employed in the industry. Prominent examples include online shopping sites (e.g., Amazon.com [1]), music and movie services (e.g., YouTube [2], Netflix [3, 4] and Spotify [5, 6]), mobile application stores (e.g., iOS App Store and Google Play), and online advertising [7]. The primary goal of recommender systems is to help users discover relevant content such as movies to watch, articles to read, or products to buy. From the user's perspective, this creates a tailored digital experience. From the business' perspective, it drives incremental revenue.

These systems learn users' preferences from historical observations to select the right content, at the right time, for the right channel. The classical setting is composed of a set of users, U, and a set of items, I, from which top-k items are chosen (e.g., items with the highest probability to be clicked) and shown to the user at time t. For each recommendation, the reward is observed, e.g., whether the user clicked. The feedback is incorporated into the next decision at time $t+1$, and the system proceeds. However, data sparsity in feedback is a challenge faced commonly, especially in newly launched recommender systems.

In this setting, the important realization is that there is an apriori decision to determine the *universe of items* I that can be recommended, at Day–0 when the system is launched, and also at Day–1, and onward, as new items are periodically added to the system.

To collect the necessary feedback data, recommender systems take advantage of randomized experimentation (i.e., *exploration*) to build personalization models (i.e., *exploitation*). However, randomized exploration incurs unwanted cost. From the users' perspective, randomized exploration is not the desired digital experience, in fact, the exact opposite of it. From the business perspective, randomization leads to missed opportunities. It has a cost on business KPIs from engagement metrics such as click-through rate to take-action rate for positive outcomes such as opening an account or purchasing a product. It is therefore critical to speed-up exploration and for the exploration policy to efficiently explore users' behaviors and interests as time proceeds.

We recently introduced a multi-level optimization approach to select items to be included in the initial randomized experimentation at the inception of a recommender system on Day–0 [8]. Our selection procedure is designed to maximize knowledge transfer between user responses and minimize the time-to-market for personalization. To achieve that, we jointly optimized the cardinality of the item universe and the diversity of items in the selection.

In this paper, we take optimized item selection [8] a step further and show how to use it beyond initial experimentation. This is a necessary and important direction since recommender systems run continually and require exploration beyond Day–0 to combat drift in user preferences and accommodate newly introduced items. To bridge the static item selection on Day–0 with the need for ongoing exploration, we show how to integrate the multi-level optimization approach with *Active Learning*. This hybridization enables item selection for exploration on Day–1+ in an ongoing fashion.

As the recommender system proceeds, optimized item selection and active learning work hand in hand to provide a principled approach for continuous exploration and to determine the frequency of randomized item recommendations. Overall, we extend our original approach [8] significantly with the following contributions:

1. We present an improved warm-start procedure based on item-to-item similarity that refines the approach in [8] to include model uncertainty. This allows for phase transition between trained and untrained items in the transfer learning procedure that more accurately reflects uncertainty in the collected data (§4).
2. We embed the multi-level optimization in an active learning framework to continuously update the selected set of items for exploration and adjust the exploration policy dynamically. This enables efficient randomized exploration as the system evolves beyond the initial selection of items (§5).
3. We provide further empirical evidence to demonstrate the effectiveness of active learning for selecting items beyond Day–1+ in an ongoing fashion. We show that the effectiveness of randomized exploration is improved through our active learning framework (§6). As such, we advocate for a principled approach for item selection not only at the inception of a recommender system but also throughout the life-cycle of its operation.

More broadly, as demonstrated in our experiments (§6), our hybrid approach serves as an integration block between modern recommender systems, classical discrete optimization techniques, unsupervised clustering, latent item embeddings and active learning.

2 The Item Selection Problem

Let us start with a formal description of our problem definition from [8].

Definition 1 (**Item Selection Problem (ISP)**) Given a set of items I, the goal of the Item Selection Problem (ISP) is to find the minimum subset $S \subseteq I$ that covers a set of labels L_c within each category $c \in C$ while maximizing the diversity of the selection S in the latent embedding space of items $E(I)$.

We use a Movie Recommender system as a concrete example to illustrate a situation where we encounter the ISP problem. We let *items* I correspond to all available *movie titles* that could be recommended. Collecting data on user ratings for all available movies is time-consuming. Therefore, we are interested in finding a subset S of movies to initialize the system for collecting training data. We need to ensure that selected movies cover a wide range of variety. The *categories* of interest, C can include the genre, language and producer. Within each category $c \in C$, we can have a set of *labels*, such as action, comedy, thriller in the genre category, and English and French in the language category. Such metadata is commonly available in recommender systems.

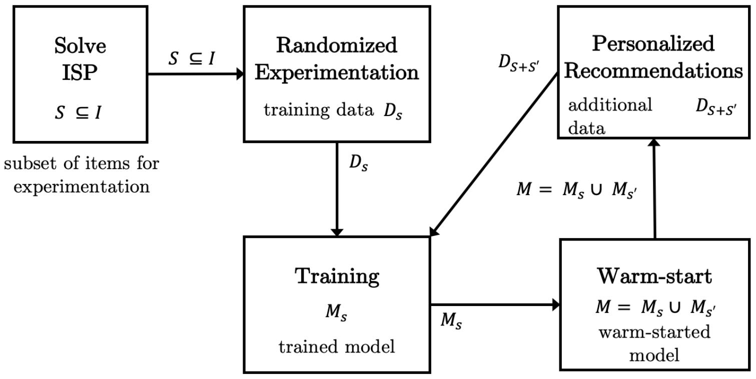


Fig. 1 Recommender System components from Item Selection Problem to Personalization.

The ISP seeks to include at least one movie from each label L_c within the different categories $c \in C$. Additionally, we want to maximize the diversity of selected movies in the *latent embedding space* $E(I)$. The latent representation can be based on textual data (e.g., synopses, movie reviews), image data (e.g., cover art), audio data (e.g., soundtracks), or video data (e.g., trailers), and the representation can be created by various existing feature engineering techniques [9–14]

The ISP is most relevant for recommender systems in new customer experiences with no historical data. As illustrated in Figure 1, randomized experimentation is employed to collect training data D_S. This training data is later used to build personalization models M_S. The longer the exploration phase takes, the worse the customer experience and business outcomes are. To mitigate this, our strategy focuses on solving the ISP to guide the randomized exploration which is later augmented with warm-started models $M_{S'}$. In the remainder, we focus on solving the ISP and the warm-start procedure.

3 Solving the ISP

The Item Selection is a Multi-Objective Optimization Problem where the goal is to *maximize* the diversity among selected items while *minimizing* the size of the selection that can cover all predefined labels (or the maximum possible coverage when the subset size is fixed). Notice also the underlying constraint satisfaction problem to ensure label coverage.

The solution methodology for ISP, as introduced in our previous work [8], is closely related to the classical Set Covering Problem (SCP) [15] which is embedded in a multi-level optimization framework. The solution consists of three levels addressing different objectives; finding the minimum subset size (§3.1), maximizing diversity (§3.2) and maximizing coverage within a fixed bound (§3.3). For brevity, we only include the most relevant components for solving the ISP below and refer to [8] for details.

3.1 Minimizing The Subset Size

Selecting a subset of items that cover all predefined labels is a standard covering formulation. Let $M_{l,i}$ be the incident matrix where rows correspond to all predefined labels, $l \in L_c$, for each category $c \in C$, and columns correspond to item $i \in I$. We define $L_{c,i}$ as the label in category c for item i and set $M_{l,i}$ to 1 only if $M_{l,i} = L_{c,i}$. Let X be the set of decision variables where x_i is a binary variable denoting whether item $i \in I$ is included in the selection. Assume each selected item incurs a cost of 1 and let *cost* represent the unit cost vector. Then formulating the unicost item selection problem, $P_{unicost}$, is straightforward:

$$\begin{aligned} & min \sum_{i}^{I} cost_i x_i \\ & \sum_{i \in I} M_{l,i} x_i \geq 1 \qquad \forall l \in L_c, \forall c \in C \\ & x_i \in \{0,1\}, cost_i = 1 \qquad \forall i \in I \end{aligned} \tag{$P_{unicost}$}$$

Assume $unicost\_selection \subseteq I$ is the solution to $P_{unicost}$ where $k = |unicost\_selection|$ is the number of selected items.

3.2 Maximizing Diversity

Our simple mapping from ISP to SCP so far does not take diversity into account. To that end, we turn to the latent representation of items. The variety of the selected subset can be captured as the separation in the item embedding space $E(I)$. We turn to a clustering algorithm, such as k-means, to cluster the embedding space of items into k clusters, leveraging the minimum subset size k from the solution of $P_{unicost}$. Using a clustering approach allows us to efficiently partition items into k clusters that have similar items within each cluster and larger variation between clusters. We reformulate $P_{unicost}$ by changing its cost structure: the inclusion of item i incurs $cost_i$ which is the minimum distance to centroids to maximize the diversity among selected items.

$$cost_i = \min \quad distance(i,k) \quad k \in K \quad \forall i \in I \tag{$P_{diverse}$}$$

3.3 Bounded Subset Size

While solving $P_{unicost}$ and $P_{diverse}$ successively leads to the smallest and most diverse set with coverage guarantees, it provides no control on the cardinality of the selection. However, remember that, the time it takes to run randomized experiments is directly proportional to the number of items. Therefore, it is desirable to control the subset size and time window of randomization using a predefined bound, t. To this end, our final formulation, $P_{max\_cover@t}$ maximizes the number of unique labels covered while limiting the total number of selected items with the given upper bound t.

Algorithm 1 Multi-Level Optimization for the Item Selection Problem (ISP).

```
**Multi-Level Optimization for ISP(I, M, E, t)**
**In:** Items: $I$
**In:** Incident Matrix: $M[label][item]$
**In:** Embedding Space: $E(I)$
**In:** Maximum Subset Size: $t$
**Out:** Selected Items: $S \subseteq I$

// First Level: Minimize the subset size
// Find the minimum set of items with full coverage
**Formulate** $P_{unicost}(I, M)$
$unicost\_selection \leftarrow \textbf{solve}(P_{unicost})$

// Second Level: Maximize diversity
// Find the minimum set of items with full coverage that maximizes diversity
$k \leftarrow |unicost\_selection|$    ▷ Use the first level to decide number of clusters
$K \leftarrow \textbf{cluster}(E(I), num\_clusters = k)$ ▷ Find clusters in embedding space
**Initialize** $cost \leftarrow$ zeros($|I|$) ▷ Set closest centroid distance as diversity cost
**for all** item $\in I$ **do**
    $cost_{item} \leftarrow \textbf{min}(\textbf{distance}(item, centroids \in K))$
**end for**
**Formulate** $P_{diverse}(I, M, cost, unicost\_selection)$
$diverse\_selection \leftarrow \textbf{solve}(P_{diverse})$    ▷ Solve for coverage and diversity

// Third Level: Maximize bounded coverage
// Find the maximum coverage within the diverse set subject to the bound
$t \leftarrow |diversity\_selection|$
**Formulate** $P_{max\_cover@t}(diverse\_selection, M, t)$
$S = max\_coverage \leftarrow \textbf{solve}(P_{max\_cover@t})$

**return** $S$
```

3.4 Multi-Level Optimization

Bringing these components together, Algorithm 1, as shown in our previous work [8], depicts the multi-level optimization framework that consist of solving $P_{unicost}$, $P_{diversity}$ and $P_{max\_cover@t}$ successively.

4 Warm-Start Procedure

Given the solution of ISP we start with randomized experimentation. As illustrated in Figure 1, this yields the training data D_S which is used to build personalization model M_S. As shown in [8], we propose a warm-start procedure that uses transfer learning [16, 17] to leverage M_S such that, when the personalization phase starts, it is not restricted to the initial subset of items but can expand beyond the trained model M_S. We explain this in detail next.

In general, machine learning models, by definition, are designed to generalize to unseen instances, and can be applied on unseen instances. In that sense, we can apply M_S to new/unseen items to get a prediction without a warm-start procedure. However, in the recommendation setting, new items (and users) lead to the notorious cold start problem, and not all recommendation algorithms have the capability to address the cold-start problem. For example, the traditional collaborative filtering algorithm lack explicit ratings for new items added to the system, hence cannot deal with the cold-start problem. A common approach to deal with this cold-start scenario is to consider latent item embeddings, denoted as $E(I)$ in our notation, to find similarities and enable the knowledge transfer between items.

4.1 Trained vs. untrained items

To be more precise about trained, S, and untrained items, S', we need a criteria, $\mathcal{G}$, to enable us differentiate between the two. We define it as follows.

The easy case is on Day-0 for items that are not included in the ISP selection, hence was not part of the randomized experimentation, have no training data, and consequently not part of the trained model. These unselected items are clearly part of the untrained set S'. The other easy case are the new items added to the system in subsequent days.

The next natural question is, of the items that are included in the ISP solution, $S \subseteq I$, whether all can declared as sufficiently trained. The answer to that question relies on application specific design decisions. Depending on the duration of the randomized experimentation (short vs. long), the number of selected items (small vs. large), the volume of traffic the newly launched application generates (slow vs. fast) and the expected rate of engagement (passive vs. active), some items, even though they are part of the randomized experimentation, might still not collect enough interaction labels necessary for training. This happens when an item in S is under-sampled, in which case the recommendations would be highly uncertain and persist large prediction errors. Transfer knowledge from such items is not necessarily beneficial to warm-start the items in S'. It is therefore reasonable to treat such items as *untrained* and prevent knowledge transfer from them. Analogously, these so-far-inadequately trained items can still benefit from warm-starts by other models until enough training data has been accumulated and their learning becomes stable.

To avoid this situation, a simple measure to define a criteria $\mathcal{G}$ is to use volume, i.e., the number of interactions an item has obtained in the training data D_s. While this is a simple criterion, it serves as a quick filtering condition to treat items with relatively low labels (in our experiments below 100 interactions) to be considered untrained as part of S'.

Beyond these easy cases and simple count-based measurement, we need a systematic approach to quantify how confident the model is when making predictions, which we study next.

Algorithm 2 Identifying items that have large uncertainty in M_S and marking these items in S as untrained.

```
Refined Definition for Trained and Untrained (S, M_S, D_S)
In: Selected Items: $S$
In: Trained data for $S$: $D_S$
In: Trained Model(s) for $S$: $M_S$
Out: Items that should remain untrained: $S'$

// Find model uncertainty of $M_S$ on each item $p$
for all $p \in S$ do
    $Uncertainty(M_{S_p}) \leftarrow$ UncertaintyEstimate$(M_S, D_S, p)$
end for

// Collect untrained items based on the criteria $\mathcal{G}(Uncertainty(M_S))$
derived from uncertainty estimates over all items in $S$
$S' \leftarrow \emptyset$
for all $p \in S$ do
    if $Uncertainty(M_{S_p}) > \mathcal{G}(Uncertainty(M_S))$ then
        $S' \leftarrow S' \cup p$
    end if
end for

return $S'$
```

4.2 Model uncertainty for warm-starts

As the system proceeds, we expect a transition from untrained items to their trained version, which we can monitor by evaluating the model uncertainty [18] of M_S for different items.

In the literature, several approaches have been studied to capture estimates on the model uncertainty [19–21]. Primarily, the uncertainty is either measured by the variance, entropy, or confidence interval of the model output [19, 21]. Alternatively, the uncertainty can be measured by the variance or the errors of learned model parameters [20].

Algorithm 2 presents our method to further refine our definition of trained and untrained items. We use the uncertainty estimate of M_S across all items, and on a particular item p, denoted by $Uncertainty(M_{S_p})$ where $p \in S$. Let $\mathcal{G}(Uncertainty(M_S))$ be a criteria found by considering several uncertainty estimates such as variance, entropy and confidence intervals over all uncertainty measures. When a particular item fails to satisfy the criteria $\mathcal{G}$, more precisely, when $Uncertainty(M_{S_p}) > \mathcal{G}(Uncertainty(M_S))$, the item is treated as still-untrained, and becomes unavailable for transfer learning in warm-start.

Based on the above, we now have access to a set of trained items, S, and their counterpart, untrained items, S'. Next, let's revisit our overall warm-start procedure and present an example using a concrete recommendation algorithm; namely, multi-armed bandits.

4.3 The overall procedure

Given the set of trained items, S, and untrained items, S', we warm-start items $s' \in S' : I \backslash S$ to build $M_{S'}$ sharing information from M_S. We take advantage of the item embedding $E(I)$ to calculate item-to-item similarity. Given pairwise distances, we find the closest item $s \in S$ for each item s'. To use s for warm-starting s', we enforce $distance(s, s') \leq w$ for $w > 0$ to ensure that the items are sufficiently similar. This can be considered as the 1-nearest neighbour [1]. A neat practical trick to obtain the distance threshold w in a data-driven fashion is to tie it to the distribution of all pairwise distances within a certain quantile, e.g., the top decile $q = 10\%$.

The next natural question is what does transfer learning between s and s' mean. Multiple options exist. We can transfer all or a subset of the interaction labels D_s to the untrained item. Similarly, we can directly use the trained parameters of model M_s to replicate it for $M_{s'}$. The important observation is, after the initial warm-start, the training data for s and s' grows separately on their specific user interactions. This allows models to continue learning independently from each other after the warm-start.

4.3.1 Example: Contextual Multi-armed bandits

Let us finish our treatment of the warm-start procedure with a concrete example using a specific recommendation model. The contextual multi-armed bandits (CMAB) are commonly used in recommendation settings thanks to their principled approach to balance exploration and exploitation trade-off [22, 23] with regret guarantees.

In CMAB, each arm corresponds to an item to be recommended. In our setting, the ISP decides the universe of arms to include in CMAB. By selecting a subset of arms in Day-0, we train our bandit learning policy faster, which yields the trained arms (items). In this scenario, M_S is not a single model but a collection of models corresponding to each arm. The initially excluded or newly added items (arms) are now untrained. Our warm-start procedure finds the closest arms, based on $E(I)$, to share the bandit models among trained vs. untrained arms. Solving ISP beyond Day–0, helps decide the next batch of new items (arms) to include in the system, which further boosts the subsequent warm-starts, and the system iterates. Notice that the recommender system can continue to exploit vs. explore in its choices among the arms, *once* the set of arms (items) are defined by our optimized selection.

5 ISP with Active Learning in Exploration

In the previous sections, our focus was on the inception of a new recommender system to speed up experimentation using our multi-level optimization framework to solve the item selection problem. We also considered a soft assignment based on model uncertainty to distinguish between trained vs. untrained items.

[1]Thanks to our anonymous reviewer for making this connection

In general, for a recommender system to operate effectively in a dynamic environment, we need to address two main problems. Initially, on Day–0, a large number of items have no or inadequate feedback data. Subsequently, on Day–1+, new items are periodically added to the system with no historical training data to learn from. Solving the ISP, addresses the first problem on Day–0. It provides an offline selection that helps minimize the cardinality of the item universe while maximizing the diversity of items for experimentation.

Taking this a step further to address the second problem, there is an ongoing need for experimentation in a recommender system to explore users' behaviors and interests. The continued exploration is a more challenging task than solving the one-shot ISP. In the beginning, all items are candidates for the initial ISP selection. As time progresses, the system needs to distinguish between items that are trained effectively vs. items for which further interaction data is still necessary, hence our further refinement in Algorithm 2 in the previous section. The two types of items require different levels of exploration for effective training. When that is the case, item selection need to be re-purposed to facilitate the ongoing exploration and exploitation trade-off.

Our ISP formalism and the solution algorithm continue to be relevant beyond Day–0. As new items are added to the system, there is a continued need to experiment with items that cover all predefined labels and that are as varied as possible. In our earlier bandit example, this corresponds to deciding the new set of extended arms to recommend from, which should still be most diverse in the embedding space and cover all predefined labels.

Furthermore, using the same ISP formalism ensures consistency between the selection on Day–0 and Day–1+. Solving the ISP in an ongoing fashion and incorporating optimized item selection dynamically provides further value to an operational system. This leads to the idea of merging the ISP into an Active Learning (AL) framework to select items continuously and guide the exploration strategy in a principled manner.

5.1 Active Learning in Recommender Systems

The main idea behind Active Learning [24] is that a machine learning algorithm can achieve greater accuracy with fewer labeled training instances if it is allowed to choose the training data from which it learns. In the context of recommender systems, this is accomplished by letting the system influence which items a user is exposed to learn users' preferences [21] more efficiently.

Two common cold-start problems addressed by active learning in the context of recommender systems are the *new user* [25] and *new item* [26]. Although a recommender system can extrapolate the preferences of new users, new users expect to see relevant results almost immediately. It is therefore essential to suggest items to a user that will speed up understanding their preferences. Similarly, for the *new item* problem, as new content is introduced to the system, it is important to gather relevant information, so that item exposure is optimized quickly.

Our ISP formalism provides a principled approach to alleviate the *new item* problem by steering the experimentation toward the latest content and balancing the exploration between the existing, hence trained to a certain degree, and the new content, which requires further user interaction data.

Active learning methods used in recommender systems can broadly be categorized as either *instance-based* or *model-based* methods [21]. Instance-based methods select points based on their properties in an attempt to predict a user's (or item's) rating by finding the closest match to other users (or items) in the system without explicit knowledge of the underlying model. Model-based methods select points to construct a model that best explains data supplied by the user to predict user ratings and maximize the model's expected error.

Our framework makes no assumptions about the underlying recommendation model, and we, therefore, focus on instance-based methods. Although model-based methods can achieve better performance than instance-based methods, strategies designed for one model are often incompatible with different systems since they rely on model-specific parameters.

In the next section, we show how to integrate ISP into an active learning framework and utilize the optimized item selection strategy.

5.2 Integrating ISP and Active Learning

Figure 2 shows how we can integrate the ISP into an active learning framework. The high-level idea is to solve the initial ISP on Day–0, and then again recursively on Day–1+ and onward as new items are added to the system.

On Day–0, the ISP yields a subset of items $S \subseteq I$. Active learning associates items with weights W_S to influence randomized experimentation. The randomization policy selects items proportionally to their assigned weights.

This yields training data D_S that is used to train our model M_S. By using the warm-start procedure described, we find a subset of remaining untrained items in $I \setminus S$ that can be warm-started. Next, we generate personalized recommendations for the trained and warm-started items, $T = S \cup S'$. Here, items in S are those with training data collected from the initial experimentation and are trained items, while items in S' are warm-started based on similarity in item embeddings. The personalized recommendations on T yields further training data D_T that is later used to update the model M_S.

On Day–1+, new items I' are added to the system, and we update I to be the set of existing and new items. To integrate ISP and Active Learning in the ongoing system, the continuous randomized experimentation is carried out in a subgroup of users. The remaining users are receiving personalized recommendation (which might have its inherent exploration policy, e.g., if bandit policies are used as in our earlier example). We perform item selection recursively to find a subset of items to be included in randomized experimentation, in order to boost the exploration as well as optimizing the cardinality and diversity of explored items. Unlike the selected items in ISP on Day–0, now the selected items contain new items plus any untrained or not warm-started ones previously. This yields the set of items $S \subseteq I \setminus T$ and item weights W_S

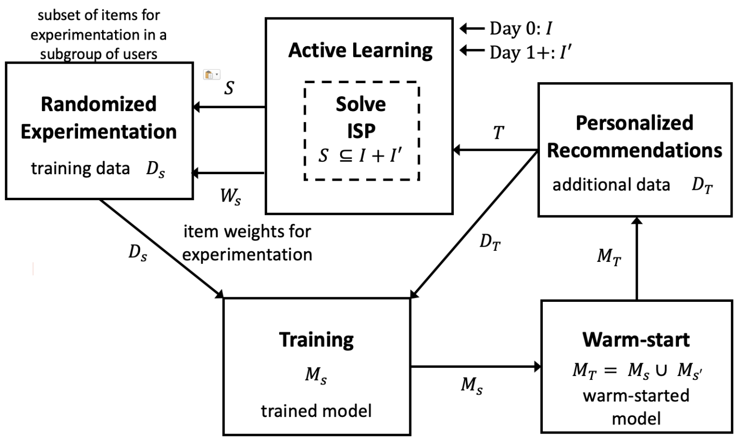


Fig. 2 Integrating Item Selection Problem with Active Learning to guide exploration.

to be used for continuous experimentation. Using the collected training data from the randomized experiment, model M_S is updated and used to warm-start remaining untrained items. The new personalized recommendations are generated as the system proceeds and learning continues.

Let us note that the proposed framework depicted in Figure 2 is flexible and can incorporate different selection methods with active learning. We consider the following two baselines for experimentation with active learning while still respecting the selection size t obtained by ISP:

- **AL-Random:** Simple baseline method that selects t untrained items using uniform random selection and uses uniform exploration weights.
- **AL-KMeans:** This method clusters the latent space into $k = |P_{unicost}|$ clusters and selects t items that are closest to the cluster centroids. Exploration weights W_S are set equal to the closest centroid distance for each item.

For ISP-based selection methods, we propose two different ISP methods within the active learning framework: one that is oblivious to the solution of the optimized item selection in the previous iteration and one that exchanges information in the cover formulation between subsequent solves.

- **AL-ISP:** Selects untrained items $S \subseteq I \setminus T$ with subset bound t using ISP as in Algorithm 1. The exploration weights W_S for each item is set to the closest centroid distance, which corresponds to the diversity cost in the multi-level optimization.
- **AL-ISP+:** Same as AL-ISP, except that the ISP formulation uses pre-fixed decision variables $x_i = 1$ for $i \in T$.

Both methods increase the diversity of the selected items and ensure that exploration collects more training data on items different from other items in the latent space.

On Day–0 with no trained items, the two methods are identical. From Day–1+, AL-ISP+ favors exploring untrained items with uncovered labels aside from those covered by T in the previous step.

6 Experiments

We experiment with well-known datasets from book and movie recommendations. The main goal of our experiments is to demonstrate the speed-up in the random experimentation phase enabled by our multi-level optimization framework while ensuring diversity and transfer learning capacity, both on Day–0 when there exists no historical data and on Day–1+ as new items are periodically added to the system.

6.1 Evaluation Metrics & Questions

The exact time window, in the number of days/weeks required for experimentation, depends on several factors such as the expected interaction volume, the engagement level of users (e.g., average click-through rates) and the complexity of the learning algorithm to train (e.g., linear regression vs. wide&deep networks [27]). While these remain application-specific, to assess the effectiveness of our approach, we focus on the following evaluation metrics measured before and after warm-start:

- **Before warm-start:** The *number of items*, which serves as a proxy of exploration time (the lower, the better) and the *number of labels* covered, which measures the scope of exploration (the higher, the better).
- **After warm-start:** The *number of items*, which measures the capacity of transfer learning (the higher, the better) and *number of labels* covered, which is a proxy for the diversity of items that can be recommended (the higher, the better).

To demonstrate the potential speed-up in random experimentation and effectiveness of the warm-start procedure at Day–0 we consider the following specific questions:

Q1: What is the minimum number of items required to cover all labels in comparison with the original number of items?
Q2: How much speed-up is enabled in exploration phase when using optimized item selection to collect response data for training?
Q3: How effective is the warm-start procedure in increasing the number of trained items and the resulting coverage?
Q4: How sensitive is the ISP formalism to the choice of latent embedding space of items?

Furthermore, to demonstrate the effectiveness of integrating ISP with active learning within the exploration component of a recommender system from Day–1+, we consider the following specific questions:

Q5: How effective is the active learning procedure in increasing trained items and the resulting coverage over time?
Q6: How diverse is the randomized experimentation data obtained by the active learning procedure?

Table 1 Summary statistics for Book and Movie Recommendation datasets.

Dataset	# Items	Categories	# Labels
GoodReads	1,000 10,000	{Genre, Publisher, Genre × Publisher }	574 1,322
MovieLens	1,000 10,000	{Genre, Producer, Language, Genre × Language }	473 1,011

6.2 Datasets: Book & Movie Recommendations

We use two well-known datasets from the recommender systems literature: the GoodReads Book Reviews [28, 29] with 11,123 books (items) and the MovieLens (ml-25m) Movie Recommendations [30] with 62,423 movies (items). We consider two randomly selected subsets, small and large versions with 1,000 and 10,000 items, respectively. We note that the selected sizes of the datasets are arbitrary and were selected to make the two datasets more comparable. The full datasets or alternative datasets with more instances could have been used. These datasets provide category and label metadata used in our ISP formulations. Table 1 summarizes our datasets.

For book recommendations, there are 11 different genres (e.g., fiction, non-fiction, children), 231 different publishers (e.g., Vintage, Penguin Books, Mariner Books), and genre-publisher pairs. This leads to 574 and 1,322 unique book labels for the small and large datasets, respectively.

For movie recommendations, there are 19 different genres (e.g., action, comedy, drama, romance), 587 different producers, 34 different languages (e.g., English, French, Mandarin), and genre-language pairs. This leads to 473 and 1,011 unique movie labels for the small and large datasets, respectively.

In the ISP, we are interested in selecting movies (books) for exploration that cover all (or maximum) genres, producers (publishers), languages, and genre-language (genre-publisher) combinations.

6.3 Setup and Parameters

All our experiments were run on a machine with Linux RHEL7 OS, 16-core 2.2GHz CPU, and 64 GB of RAM. For optimization, we use the Python-MIP [31] with the COIN-OR CBC Solver[32]. For clustering, we employ the default k-means from sklearn [33]. To generate text embeddings, we utilize TextWiser [34][2]. For the warm-start procedure we use a distance quantile of $q = 0.1$. For relevant experiments, results are averaged over $n = 50$ seeds.

6.4 Embedding Space

The embedding space is based on textual descriptions of movies and books. For vector representation, we use Term Frequency Inverse Document Frequency (TFIDF) [9], ignoring terms with a document frequency lower than the cut-off threshold of $min\_df = 20$. To reduce dimensionality, we transform these vectors

[2] http://github.com/fidelity/textwiser

using non-negative matrix factorization [13] and generate 30-dimensional feature vectors for each item. In Section 6.9, we experiment with other strategies to understand the sensitivity of ISP as a function of the embedding space.

6.5 Comparisons

We compare $P_{unicost}$, $P_{diverse}$, $P_{max\_cover@t}$ on each dataset against the following challenger algorithms:

1. *Random*: Uniform random selection as a simple baseline. The *Random* method uses the subset size k from the solution for $P_{unicost}$.
2. *Greedy*: The classical greedy heuristic for set covering that adds items iteratively, whereby at each step, the item with the best $\frac{cost}{coverage}$ ratio is selected. This is a competitive baseline with a polynomial-time approximation scheme with worst-case guarantees [35].
3. *KMeans*: Unsupervised clustering approach that operates on the same embedding space. As in *Random*, it uses the subset size k from the solution of $P_{unicost}$ as the number of clusters. This method first clusters the latent space into k centers and then selects items closest to the centroids.

Notice these challenger algorithms are better informed than their pure random, greedy, clustering counterparts since they inherent the size of the selection k from ISP. While *Greedy* maximizes coverage, it does not take diversity into account. This helps us assess the effectiveness of $P_{diverse}$. Analogously, while *KMeans* maximizes diversity, it omits label coverage. This in turn helps us determine the effectiveness of coverage constraints. Both *Random* and *KMeans* select k items for which we have an optimality certificate from $P_{unicost}$ that covers all labels.

6.6 Analysis of Coverage [Q1]

To answer Q1 and find the minimum set of items covering all labels, we solve $P_{unicost}$ and compare the number of selected items, the resulting label coverage before and after the warm-start procedure.

6.6.1 Book Recommendations:

Table 2 summarizes our results for the GoodReads dataset. Solving $P_{unicost}$ before warm-start returns 374 items covering all 574 labels in the small dataset and 1,080 items that cover all 1,322 labels in the large dataset. This represents reductions of 63% and 89% compared to selecting all items. We then use $|P_{unicost}|$ for *Random* and *Greedy*. The *Greedy* algorithm is also competitive on both datasets in terms of label coverage. As expected, the number of labels covered by *Random* and *KMeans* is markedly lower. The solution for $P_{diverse}$ only requires 72 and 85 more items than $|P_{unicost}|$ demonstrating the slight pay-off to maximize the diversity of the selected content. After warm-start, *KMeans* yields the highest number of warm-started items. This is expected since clustering purely targets the diversity of the space, but unfortunately, its label coverage is no different than *Random*.

Table 2 [Q1] Comparison of our solution to ISP and challenger approaches on the GoodReads dataset before and after warm-start in terms of the number of items selected and label coverage.

		Before Warm-Start			After Warm-Start		
Dataset	Method	Items	Labels	Coverage	Items	Labels	Coverage
GoodReads 1K Items 574 Labels	*Random*	374	325	57%	463	367	64%
	Greedy	374	574	100%	460	574	100%
	$P_{unicost}$	374	574	100%	470	574	100%
	$KMeans$	374	333	58%	741	431	75%
	$P_{diverse}$	446	574	100%	523	574	100%
GoodReads 10K Items 1,322 Labels	*Random*	1,080	606	46%	2,226	771	58%
	Greedy	1,080	1,322	100%	2,227	1,322	100%
	$P_{unicost}$	1,080	1,322	100%	2,433	1,322	100%
	$KMeans$	1,080	589	45%	2,834	838	64%
	$P_{diverse}$	1,165	1,322	100%	1,602	1,322	100%

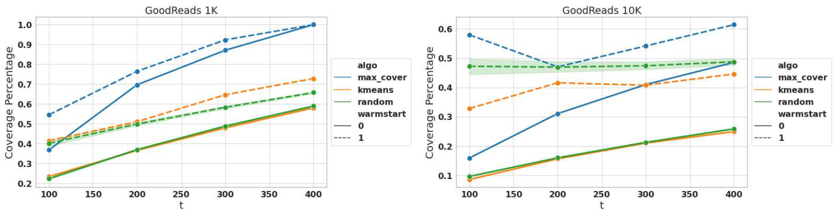
Table 3 [Q1] Comparison of our solution to ISP and challenger approaches on the MovieLens dataset before and after warm-start in terms of the number of items selected and label coverage.

		Before Warm-Start			After Warm-Start		
Dataset	Method	Items	Labels	Coverage	Items	Labels	Coverage
MovieLens 1K Items 473 Labels	*Random*	243	220	46%	624	274	58%
	Greedy	249	473	100%	648	473	100%
	$P_{unicost}$	243	473	100%	647	473	100%
	$KMeans$	243	206	43%	659	276	58%
	$P_{diverse}$	248	473	100%	652	473	100%
MovieLens 10K Items 1,011 Labels	*Random*	523	298	29%	2,479	561	55%
	Greedy	703	1,011	100%	3,031	1,011	100%
	$P_{unicost}$	523	1,011	100%	2,659	1,011	100%
	$KMeans$	523	317	31%	1,801	542	54%
	$P_{diverse}$	558	1,011	100%	1,971	1,011	100%

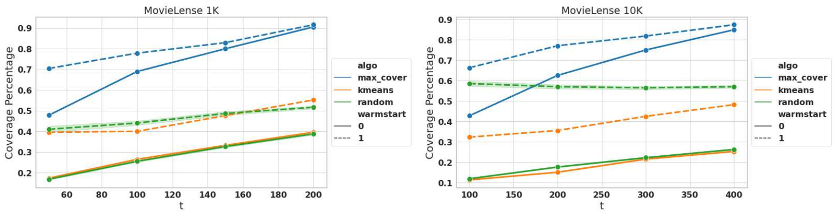
6.6.2 Movie Recommendations:

Table 3 summarizes our results on the MovieLens datasets. $P_{unicost}$ achieves complete coverage with almost a 90% reduction compared to selecting all items. In this dataset, *Greedy* cannot achieve the quality of the optimum solution. Its optimality gap (249 vs. 243) for the small dataset is 2% and is significantly worse (703 vs. 523) at 34% for the large dataset. *Random* and $KMeans$ continue performing poorly in terms of coverage before and after warm-start.

Lastly, in terms of runtime, solving the multi-level optimization with $P_{unicost}$, $P_{diversity}$ and $P_{max\_cover@t}$ takes 20 minutes at most. This shows the efficiency of optimization technology when faced with recommendation benchmarks.



(a) GoodReads 1K (b) GoodReads 10K



(c) MovieLens 1K (d) MovieLens 10K

Fig. 3 [Q2] Bounded coverage of labels with varying number of selected items t.

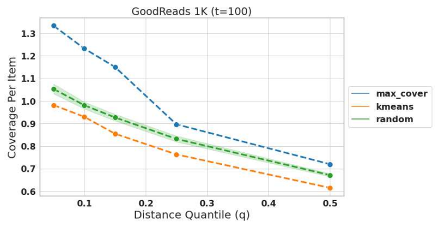
6.7 Analysis of Bounded Coverage [Q2]

To answer Q2 and demonstrate potential speed-up in random experimentation, we vary the subset bound t and analyze the label coverage before and after warm-starts for $P_{max\_cover@t}$, $KMeans$ and $Random$. We keep the range of t the same between datasets, with the exception of MovieLens 1K due to the smaller number of required items to cover all labels. In practice, the bound t is application-driven governed by time constraints, expected volumes, and user engagement. Figure 3 presents our results.

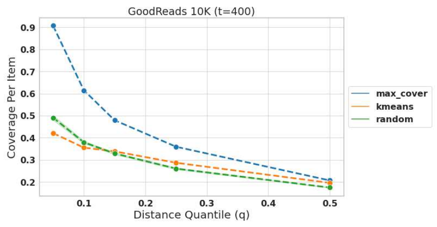
Before the warm-start, we see that, for each method, coverage increases consistently as t increases. Critically, for a given coverage level, the required number of items t is always lower for $P_{max\_cover@t}$ compared to other methods, indicating potential speed-up. For example, on the small GoodReads dataset, a coverage of 50% can be achieved at $t = 140$, while $t = 320$ is required to obtain the same coverage using the $KMeans$ and $Random$ methods. There is a 2X reduction in the number of items, again demonstrating time savings in exploration.

After the warm-start, coverage increases for each method at each t, and notably, the coverage for $P_{max\_cover@t}$ continues to rank highest in both datasets. $KMeans$ and $Random$ results in similar coverage, but neither is capable of passing 50% (60%) with 200 (400) items on small and large sets whereas $P_{max\_cover@t}$ reaches 80% (85%) within the same bound.

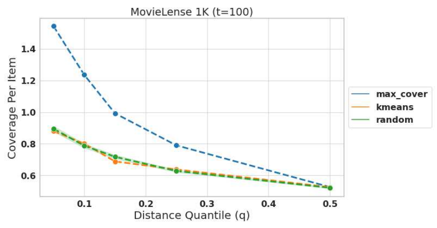
It is worth noting that the number of items warm-started is not the same for different methods. In the next section, we analyze the efficiency of the warm-start procedure in terms of the number of labels covered per item.



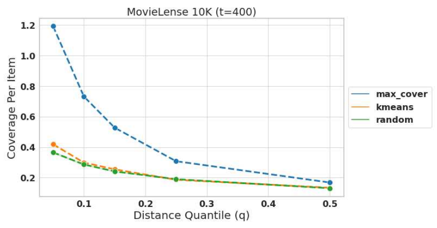
(a) GoodReads 1K



(b) GoodReads 10K



(c) MovieLens 1K



(d) MovieLens 10K

Fig. 4 [Q3] Warm-Start analysis of unit coverage with varying distance quantile q from the top semi-decile, $q = 0.05$, to $q = 0.5$.

6.8 Analysis of Warm-Start [Q3]

To answer Q3 and assess the effectiveness of the warm-start procedure we perform sensitivity analysis with varying thresholds of the distance quantile q and evaluate the average number of labels covered per item after warm-start.

We keep the number of selected items fixed at $t = 100$ for small dataset and $t = 400$ for large dataset and find the selected items using $P_{max\_cover@t}$, $KMeans$ and *Random*. Using the selected items, we run the warm-start procedure at different values of q from the top semi-decile, $q = 0.05$ to $q = 0.5$. As q increases, the distance constraint to warm-start an item is relaxed, thereby increasing the number of items that can be feasibly warm-started. In parallel, this possibly reduces the relevance of these items given the already collected training data.

Figure 4 presents the results for GoodReads and MovieLens datasets. For each method, the charts show the unit coverage (the number of covered labels divided by the number of items) after the warm-start. Notice that, as q increases, unit coverage decreases across the board for all methods and datasets. This clearly demonstrates the diminishing returns in label coverage as more items are included. Consistent with the coverage analysis, $P_{max\_cover@t}$ is the most effective approach in terms of the number of labels covered per item, significantly better than *Random* and $KMeans$ especially for the top (semi-) decile, i.e., $q \leq 0.1$.

Table 4 [Q4] Comparison of different item embeddings in terms of coverage and capacity to warm-start unseen items on the GoodReads datasets.

Dataset	Embedding	Items	Labels	Coverage	Unit Coverage
GoodReads 1K Items 574 Labels	TFIDF	254	313	54%	1.2
	Word2Vec	233	318	55%	1.4
	GloVe	208	299	52%	1.4
	Byte-Pair	235	300	52%	1.3
GoodReads 10K Items 1,322 Labels	TFIDF	1,743	723	55%	0.4
	Word2Vec	900	608	46%	0.7
	GloVe	1,098	641	48%	0.6
	Byte-Pair	940	553	42%	0.6

6.9 Analysis of Embedding Space [Q4]

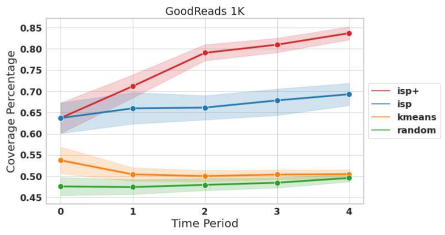
To answer Q4 and evaluate the sensitivity of ISP with respect to the underlying item embeddings, we solve $P_{max\_cover@t}$ on the books dataset with a fixed $t = 100$ and $q = 0.1$. We experiment with several complementary embeddings using TextWiser [34]. Besides our baseline TFIDF, we employ FastText Word2Vec to learn word vectors [10, 36], GloVe [37] embedding to learn global word representations, and Byte-Pair [38] embedding to learn character level information. In all cases, we apply Singular Value Decomposition (SVD) [14] to generate a fixed size 30-dimensional latent representation.

Table 4 reports the total number of items and percentage of label coverage after warm-start. The coverage is similar for the different embeddings hinting at the robustness of our multi-level framework. Nevertheless, more complex embeddings provide better unit coverage compared to TFIDF. In particular, the Word2Vec embedding achieves the best unit coverage in both datasets, closely followed by the GloVe embedding. This is thanks to the recent advances in NLP and the efficiency of pretrained models in capturing text semantics.

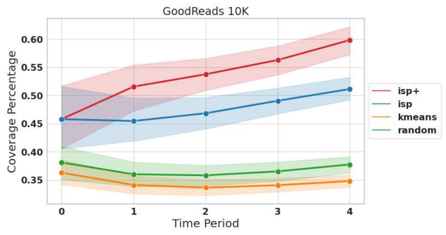
6.10 Analysis of Active Learning Coverage [Q5]

To answer Q5 and assess the effectiveness of the active learning procedures described in Section 5.2, we perform an ablation study on the GoodReads dataset. First, we randomly split each dataset into 5 batches representing periods. Then, at each time step, we select $t = 50$ items from the small dataset and $t = 80$ from the large dataset to simulate the arrival of new items, and subsequently warm-start a subset of the remaining untrained items. Then, at the end of each time step, we measure the label coverage of the trained items T based on the total items I active in the system at that time.

Figure 5 presents the results that show the effectiveness of combining ISP and AL together. For each method, the chart shows coverage after warm-start at each time period. Notice that coverage for *AL-ISP* and *AL-ISP+* are the same at time period zero and that this also corresponds to the coverage



(a) GoodReads 1K



(b) GoodReads 10K

Fig. 5 [Q5] Coverage percentage and unit coverage for ISP + Active Learning.

that ISP without AL would have generated. Beyond time period zero, AL-ISP methods consistently provide better coverage than *AL-KMeans* and *AL-Random*. Moreover, through simple information sharing, *AL-ISP+* provides better coverage compared to *AL-ISP*. Coverage for ISP without AL remains the same as at time period zero, or can even deteriorate as more items are added, demonstrating the effectiveness of integrating ISP with AL.

6.11 Analysis of Active Learning Diversity [Q6]

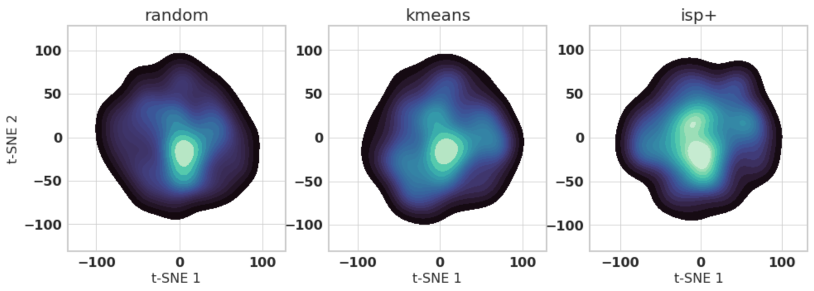
To answer Q6 and assess how diverse the randomized experimentation obtained from different methods, we follow the same evaluation setup in Q5. Performing item selection at each period yields a list of selected items and item weights representing the amount of randomized experimentation, which we visualize.

Figure 6 presents the results for the GoodReads datasets, and we omit other experiments with similar findings. Using a 2-dimensional t-SNE embedding [39] of the latent item representations $E(I)$, we visualize the distribution of experimentation data collected for each method. We see that the data distribution is more dispersed for *AL-KMeans* and *AL-ISP+*, as shown in larger highlighted areas and multiple centroids, compared to *AL-Random* showing a single centroid where most of the training data is located. The result suggests that *AL-KMeans* and *AL-ISP+* yield a more diverse training dataset, again, showing the added value of integrating item selection into active learning. Note that *KMeans* approach neglects the label coverage completely. Overall, *AL-ISP+* stands out as the best mechanism in terms of diversity, label coverage, and handling incoming items in an ongoing fashion for exploration.

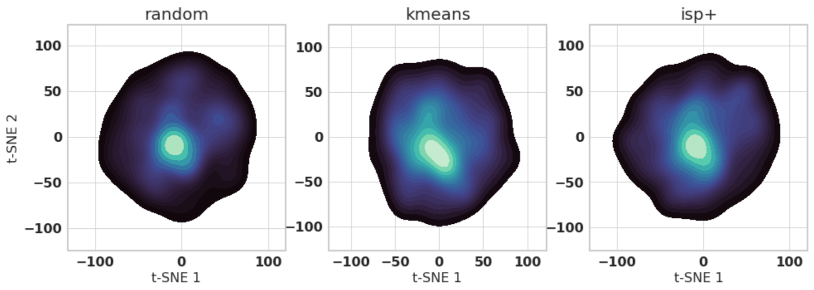
7 Related Work

Our hybrid work, at the intersection of Operations Research (OR), Natural Language Processing (NLP), Active Learning, and Recommender Systems, relates to several approaches.

From the OR perspective, while cover formulations are standard in the literature, we show that optimization solvers can tackle problems derived from widely used recommendation datasets. Our multi-level optimization framework can be seen as an example of Hybrid Optimization [40] as it combines strengths of the cover formulation with unsupervised clustering on the embedding space.



(a) GoodReads 1K



(b) GoodReads 10K

Fig. 6 [Q6] Diversity of trained items using active learning visualized using t-SNE embedding of item latent representations.

From NLP and Transfer Learning perspective, we take advantage of the recent advances in pre-trained word embeddings such as FastText [36], GloVe [37], and Byte-Pair [38] and use them in a downstream task.

From the Recommenders perspective, our framework leaves the choice of personalization algorithm open to a wide range of options, such as matrix factorization, collaborative filtering [41, 42], nearest-neighbors [43], factorization machines [44], and deep learning models including Wide&Deep[27], DeepFM [45], and DCN [46] (see [47, 48] for a survey).

From the Active Learning perspective, our active learning procedure that integrates ISP is an instance-based [21] method with the goal of increasing label coverage and a decreasing output uncertainty [49] by increasing the diversity of recommended items. It shares similarities with diversity-based [50] active learning methods, but these do not simultaneously optimize for external constraints (e.g., label coverage). Once the reward feedback is collected, error-based [51, 52] active learning methods could also be considered.

While we considered an approach that starts with exploration followed by exploitation, these steps can be blended together [7]. For instance, multi-armed bandit learning policies [22] such as ϵ-Greedy, Thompson Sampling [53], and Upper Confidence Bounds [54] mixes exploration and exploitation. These methods neglect the item metadata and the label coverage problem.

Traditionally, statistical power analysis [55] and experimental design [56] methods offer a formal treatment to identify significant effects for a given statistical power. However, given the combinatorial nature of the problem and limited data context for new applications, these methods are rarely suited for item selection.

Our warm-start procedure shares similarities with [16, 17] which builds ensembles based on item similarity. A more involved approach would be to transfer models between different applications as in cross-system recommendations [57, 58], e.g., cross-referencing between different systems such as book and movie recommendations.

Finally, let us mention that item selection is not purely an algorithmic problem. As we advocated in [8], experimentation design and item selection is a human-in-the-loop optimization problem, where we interface with non-technical partners. To that end, offering interactive decision-support tools [8] to motivate and explain the rationale behind the addition and removal of items plays a critical role in customer facing real-world applications.

8 Discussions

In this paper, we focus our evaluations on the maximized coverage of selected items' category labels and diversity. In practice, the performance of a "good" recommender system is more complicated to quantify, which depends on not only algorithmic factors but also the user experience design and the quality of items to be recommended. These aspects cannot be recovered by better recommendation algorithms or optimized item selection.

Recommenders systems need to be continuously trained with online feedback to capture the dynamic nature, and often changing behavior of users. Our framework leaves the choice of recommendation algorithm open. As systems evolve with continuous training, a recommender that uses standard exploration vs. using our ISP+Active Learning to boost exploration, given the same choice of underlying algorithm to build machine learning models, would eventually converge to similar performance as sufficiently large training data becomes available. The question is "how quickly" each variant starts delivering a better *personalized* experience for users, hence bring business value. In our experiments, we measure the quality of exploration using the cardinality and diversity of the item set selected on Day-0 and Day-1+. The results have shown that the time-to-personalization is shortened and we ensure a better diversity in the selected items for exploration.

9 Conclusion

We extended our previous work on optimized item selection [8] with active learning. This is especially relevant for new and data-sparse recommendation applications. Our interdisciplinary work combines techniques from OR, NLP, Unsupervised and Active Learning. With significant speed-ups in exploration, we alleviate alleviate the negative impact of randomization in customer experience and business outcomes. We hope that our ISP+AL formalism facilitates further integration between complementary fields and helps practitioners design new recommendation system experiences.

Declarations

- **Funding.** No funding was received other than the support of the authors' employer.
- **Competing interests.** The authors have no competing interests or conflicts to declare that are relevant to the content of this article.
- **Ethics approval.** Not applicable.
- **Consent to participate.** Not applicable.
- **Consent for publication.** All authors declare that we express our consent for the publication of this article.
- **Availability of data and materials.** The dataset used in our experiments are publicly available benchmarks that are commonly used in this line of research.
  1. The Goodreads datasets [28, 29] are available at https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home.
  2. The MovieLens datasets [30] are available at https://grouplens.org/datasets/movielens/.
- **Code availability.** Most of the work presented in this paper builds on top of open-source libraries. Specifically, the open-source TextWiser library is used for generating item embeddings from text which is available at http://github.com/fidelity/textwiser. The optimization formulations presented in Algorithm 1 is solved using the open-source Python-MIP library, which is available at https://github.com/coin-or/python-mip. The source code to reproduce the tables and figures in the paper can be made available online upon publication.
- **Authors' contributions.** All authors contributed to the design of the overall algorithm presented in this paper. The initial study conception and the design of the multi-level framework are composed by Serdar Kadıoğlu which is later extended to the active learning setting by Bernard Kleynhans and Xin Wang. The preparation of the material, data collection and analysis were performed by Bernard Kleynhans and Xin Wang. All authors contributed to the write-up of the manuscript. All authors read and approved the final manuscript.

References

[1] Lake, T., Williamson, S.A., Hawk, A.T., Johnson, C.C., Wing, B.P.: Large-scale collaborative filtering with product embeddings. CoRR **abs/1901.04321** (2019) https://arxiv.org/abs/1901.04321

[2] Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Sen, S., Geyer, W., Freyne, J., Castells, P. (eds.) Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016, pp. 191–198 (2016)

[3] Wu, C., Alvino, C.V., Smola, A.J., Basilico, J.: Using navigation to improve recommendations in real-time. In: Sen, S., Geyer, W., Freyne, J., Castells, P. (eds.) Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016, pp. 341–348 (2016)

[4] Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Cremonesi, P., Ricci, F., Berkovsky, S., Tuzhilin, A. (eds.) Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017, pp. 130–137 (2017)

[5] Hansen, C., Hansen, C., Maystre, L., Mehrotra, R., Brost, B., Tomasi, F., Lalmas, M.: Contextual and sequential user embeddings for large-scale music recommendation. In: Santos, R.L.T., Marinho, L.B., Daly, E.M., Chen, L., Falk, K., Koenigstein, N., de Moura, E.S. (eds.) RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020, pp. 53–62 (2020)

[6] Mehrotra, R., Shah, C., Carterette, B.A.: Investigating listeners' responses to divergent recommendations. In: Santos, R.L.T., Marinho, L.B., Daly, E.M., Chen, L., Falk, K., Koenigstein, N., de Moura, E.S. (eds.) RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020, pp. 692–696 (2020)

[7] Ricci, F., Rokach, L., Shapira, B. (eds.): Recommender Systems Handbook, (2015)

[8] Kadioglu, S., Kleynhans, B., Wang, X.: Optimized item selection to boost exploration for recommender systems. In: Proceedings of the 18th International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Vienna, Austria (2021)

[9] Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation (1972)

[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)

[11] Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)

[12] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv

preprint arXiv:1810.04805 (2018)

[13] Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562 (2001)

[14] Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. In: Linear Algebra, pp. 134–151 (1971)

[15] Beasley, J.E.: An algorithm for set covering problem. European Journal of Operational Research **31**(1), 85–93 (1987)

[16] Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Brodley, C.E. (ed.) Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004. ACM International Conference Proceeding Series, vol. 69 (2004)

[17] Caruana, R., Munson, A., Niculescu-Mizil, A.: Getting the most out of ensemble selection. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China, pp. 828–833 (2006)

[18] Wang, X., Kadioglu, S.: Bayesian deep learning based exploration-exploitation for personalized recommendations. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1715–1719 (2019). IEEE

[19] Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 441–448. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)

[20] Chan, N.N.: A-optimality for regression designs. Journal of Mathematical Analysis and Applications **87**(1), 45–50 (1982)

[21] Rubens, N., Elahi, M., Sugiyama, M., Kaplan, D.: Active learning in recommender systems. In: Recommender Systems Handbook, pp. 809–846 (2015)

[22] Strong, E., Kleynhans, B., Kadioglu, S.: Mabwiser: A parallelizable contextual multi-armed bandit library for python. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI 2019), pp. 885–890 (2019). IEEE. https://github.com/fidelity/mabwiser

[23] Strong, E., Kleynhans, B., Kadıoğlu, S.: MABWiser: parallelizable contextual multi-armed bandits. Int. J. Artif. Intell. Tools **30**(4), 2150021–1215002119 (2021). https://doi.org/10.1142/S0218213021500214

[24] Settles, B.: Active learning literature survey (2009)

[25] Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, pp. 127–134 (2002)

[26] Huang, Z.: Selectively acquiring ratings for product recommendation. In: Proceedings of the Ninth International Conference on Electronic Commerce, pp. 379–388 (2007)

[27] Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H.: Wide & deep learning for recommender systems. In: Karatzoglou, A., Hidasi, B., Tikk, D., Shalom, O.S., Roitman, H., Shapira, B., Rokach, L. (eds.) Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016, pp. 7–10 (2016)

[28] Wan, M., McAuley, J.J.: Item recommendation on monotonic behavior chains. In: Pera, S., Ekstrand, M.D., Amatriain, X., O'Donovan, J. (eds.) Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, pp. 86–94 (2018)

[29] Wan, M., Misra, R., Nakashole, N., McAuley, J.J.: Fine-grained spoiler detection from large-scale review corpora. In: Korhonen, A., Traum, D.R., Màrquez, L. (eds.) Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pp. 2605–2610 (2019)

[30] Harper, F., Konstan, J.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems **5**(4) (2015)

[31] Toffolo, T.A.M., Santos, H.G.: Python-mip: Version 1.9.1. https://www.python-mip.com/

[32] johnjforrest, Vigerske, S., Santos, H.G., Ralphs, T., Hafer, L., Kristjansson, B., jpfasano, EdwinStraver, Lubin, M., rlougee, jpgoncal1, h-i-gassmann, Saltzman, M.: coin-or/Cbc: Version 2.10.5

[33] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R.,

VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)

[34] Kilitcioglu, D., Kadioglu, S.: Representing the unification of text featurization using a context-free grammar. Proceedings of the AAAI Conference on Artificial Intelligence (2021)

[35] Vazirani, V.V.: Approximation Algorithms, (2001)

[36] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)

[37] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)

[38] Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (2016)

[39] van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (2008)

[40] Hooker, J.N.: Integrated Methods for Optimization. International series in operations research and management science, vol. 100 (2007)

[41] Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)

[42] Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, pp. 1257–1264 (2007)

[43] Goldberg, D., Nichols, D.A., Oki, B.M., Terry, D.B.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992)

[44] Rendle, S.: Factorization machines. In: Webb, G.I., Liu, B., Zhang, C., Gunopulos, D., Wu, X. (eds.) ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, pp.

995–1000 (2010)

[45] Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: A factorization-machine based neural network for CTR prediction. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp. 1725–1731 (2017)

[46] Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. CoRR **abs/1708.05123** (2017) https://arxiv.org/abs/1708.05123

[47] Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 73–105 (2011)

[48] Zhang, S., Yao, L., Sun, A.: Deep learning based recommender system: A survey and new perspectives. CoRR **abs/1707.07435** (2017) https://arxiv.org/abs/1707.07435

[49] Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. Int. Conf. on Machine Learning. Morgan Kaufmann (2001)

[50] McGinty, L., Smyth, B.: On the role of diversity in conversational recommender systems. In: International Conference on Case-Based Reasoning, pp. 276–290 (2003). Springer

[51] Rubens, N., Tomioka, R., Sugiyama, M.: Output divergence criterion for active learning in collaborative settings. IPSJ Online Transactions **2**, 240–249 (2009)

[52] Danziger, S.A., Zeng, J., Wang, Y., Brachmann, R.K., Lathrop, R.H.: Choosing where to look next in a mutation sequence space: Active learning of informative p53 cancer rescue mutants. Bioinformatics **23**(13), 104–114 (2007)

[53] Thompson, W.R.: On the Likelihood that one Unknown Probability Exceeds Another in View of the Evidence of Two Samples. Biometrika **25**, 285–294 (1933)

[54] Valko, M., Korda, N., Munos, R., Flaounas, I., Cristianini, N.: Finite-time analysis of kernelised contextual bandits. In: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, pp. 654–663 (2013)

[55] Cohen, J.: Statistical Power Analysis for the Behavioral Sciences, (2013)

[56] Ryan, T.P., Morgan, J.: Modern experimental design. Journal of Statistical Theory and Practice **1**(3-4), 501–506 (2007)

[57] Zhao, L., Pan, S.J., Xiang, E.W., Zhong, E., Lu, Z., Yang, Q.: Active transfer learning for cross-system recommendation. In: desJardins, M., Littman, M.L. (eds.) Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA (2013)

[58] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)