Full-Page Recommender: A Modular Framework for Multi-Carousel Recommendations

Jan Kislinger
jan.kislinger@gmail.com
Czech Technical University
Prague, Czech Republic

Abstract

Full-page layouts with multiple carousels are widely used in video streaming platforms, yet understudied in recommender systems research. This paper introduces a structured approach to generating such pages by recommending coherent item collections and optimizing their arrangement. We break the problem into subcomponents and propose methods that balance user relevance, diversity, and coherence. We also present an evaluation framework tailored to this setting. We argue that this approach can improve recommendation quality beyond traditional ranked lists.

CCS Concepts

• **Information systems** → **Recommender systems**; **Information retrieval**; • **Computing methodologies** → *Ranking*; • **Human-centered computing** → *User interface design.*

Keywords

Recommender systems, Full-page layout, Personalization, User behavior, Evaluation

ACM Reference Format:
Jan Kislinger. 2025. Full-Page Recommender: A Modular Framework for Multi-Carousel Recommendations. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25), September 22–26, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3705328.3748753

1 Introduction

Contemporary research in recommender systems has predominantly focused on linear recommenders, where a user is presented with a single ranked list of items. This paradigm, while effective in many domains, does not reflect the layout used in modern content platforms, particularly in video-on-demand services like Netflix, Hulu, or Peacock. These platforms utilize a full-page interface composed of multiple horizontally scrollable carousels, each containing a curated list of items grouped by a shared attribute, such as genre or popularity. This format reflects real user experiences and demands a rethinking of how recommendation algorithms are designed and evaluated.

Despite its prevalence in production systems, the carousel-based interface has received limited attention in the recommender systems literature. Most research in this area has concentrated on user interaction studies, such as eye tracking or click-through behavior, rather than on the algorithmic or evaluation challenges posed by this layout. Moreover, there is a lack of systematic frameworks for both generating and evaluating these full-page recommendations.

This work addresses the central research question: *How can we transition from a single ranked list to a full-page layout composed of multiple carousels in a principled and effective manner?*

Given a user with a known interaction history, our objective is to generate a personalized full-page recommendation layout. This layout is structured as a vertical list of carousels, each containing a fixed number of items. To ensure coherence, all items within a carousel should share a meaningful attribute that defines its label (e.g., *Romantic Comedies*).

The layout must strike a balance across several potentially conflicting goals:

- **User relevance**: recommendations should reflect the user's preferences.
- **Catalog coverage**: a broad and diverse set of items should be surfaced.
- **Specificity**: carousels should be focused and non-generic.
- **Consistency**: items within a carousel should share a clear common feature.
- **Deduplication**: minimize repetition of items and overlap of carousel themes.

These objectives often pull in different directions, requiring the algorithm to make careful trade-offs. For example, when a user has shown no interest in documentaries, including a generic *Documentaries* carousel may still be justified to highlight catalog diversity. In contrast, for a user interested in documentaries, the system should surface more specific collections like *History* or *Nature*.

Because the layout spans the entire page, we allow the importance of individual objectives to vary across different parts of the page. The top of the page should focus on high relevance, while the lower part of the page can emphasize novelty and broader catalog exposure. Additionally, deduplication constraints may be relaxed for carousels that are far apart, allowing more flexibility in showcasing content.

Although our primary focus is on generating general-purpose recommendation layouts, the same framework can be applied to other parts of the platform. By adjusting the item ranking component, the system can adapt to context-specific needs—for instance, combining query relevance with user preferences for search pages, recommending related items on detail pages, or tailoring carousels for genre-specific or category-specific subpages.

2 Related Work

Several papers have explored non-linear recommendation interfaces. For example, Rahdari et al. [12] proposed a *carousel interaction model* to analyze user navigation effort in carousels versus ranked lists. Using an offline dataset, they showed that users perform fewer interactions to find a desirable item in carousels than in ranked lists. While this study focuses on interaction modeling, our work addresses the upstream challenge of generating high-quality carousels in the first place.

Xi et al. [17] introduced a self-attention-based model to rerank items within carousels. Their analysis, based on a proprietary dataset from an app store homepage, revealed that an item's CTR is influenced by nearby items. This effect weakens with distance. Their algorithm uses inter-list and intra-list attention to optimize page layout. Our approach complements this by focusing on how to construct and select the carousels themselves, before any reranking is applied.

Singal et al. [13] developed a *collection recommender system* that clusters relevant songs for a user, assigning each item to exactly one collection. Unlike traditional systems, their method does not rely on shared, predefined collections. Collection labels were based on the three most common artists in each group. Our work extends this idea to more general-purpose collections and emphasizes flexible, overlapping, and semantically coherent grouping.

To represent collections, Elahi and Chandrashekar [2] proposed a method for aggregating item embeddings into a single embedding for a sorted list (a *slate*). The model incorporates (dis)similarities among items to improve representation quality. The resulting embedding is used to score recommendation slates. While their work focuses on slate embedding, our method treats collection construction and layout selection as distinct steps in a modular pipeline.

Some research has focused on grid-based recommendation layouts [16]. Unlike carousels, rows in a grid do not need to follow a shared topic. This changes how recommendations are presented and interpreted. Our framework specifically targets carousel-based interfaces where thematic coherence is a design constraint.

Behavioral studies also contribute valuable insights into user interaction with recommendation interfaces. Jugovac and Jannach [8] reviewed a range of recommender systems, including those with *multiple lists*. Our work builds on these behavioral insights by designing algorithms that aim to improve the quality and coherence of such lists.

Jannach et al. [7] conducted UI experiments with 775 users to compare multi-list, grid-based, and single-list interfaces. They found that multi-list interfaces increase the time to locate a desired item. However, carousel labels enhance perceived diversity and novelty. We take these UI considerations into account in our layout generation and evaluation strategy, particularly by incorporating label coherence and layout diversity.

Evaluation methods for multi-list recommendations remain limited. Some works (e.g., [17]) evaluate each carousel separately using metrics such as *MAP* and *DCG*. They then aggregate the results into a single score. Ferrari Dacrema et al. [3] proposed 2DCG, a metric for grid and list-of-lists layouts. It extends DCG by incorporating both horizontal and vertical positions. We build upon this line of work by proposing a user behavior-informed simulation model that accounts for user attention across carousels.

Alves [1] explored how to evaluate recommendation clusters using *inductive matrix completion*. This method assesses clustering quality when all items in a cluster share the same score for a user. We apply similar ideas to benchmark automatically generated collections, but extend them to evaluate layout-level metrics and interactions.

3 Proposed Framework

To tackle the complexity of the full-page recommendation problem, we decompose it into three interdependent subproblems. This decomposition is motivated by both practical and scientific considerations. By isolating key components—personalized item ranking, collection generation, and layout optimization—we enable focused research on each aspect. This modular approach not only supports systematic evaluation of each part independently but also allows for the flexible integration of different models and methods across subproblems.

3.1 Personalized Item Ranking

We require a scoring function that estimates the relevance of items for a given user. This component is not the primary focus of our research. Instead, we employ standard models such as matrix factorization [11], EASE [14], and SASRec [10] to generate these scores. Results will be demonstrated using multiple models to ensure robustness of downstream components.

This study does not aim to propose novel item scoring or ranking algorithms. Instead, our focus lies in evaluating how existing item scoring techniques interact with downstream components of the recommendation pipeline. For example, we analyze whether using the same item-item similarity matrix for both clustering and scoring—such as one derived from EASE—leads to improved performance or consistency across the system.

Figure 1 illustrates a typical recommendation produced by the EASE model, following a user's recent interaction history. This form of linear recommendation is widely used in traditional setups, but does not account for layout or diversity at the page level.

3.2 Collection Generation

In our setting, items are grouped into semantically coherent collections, each of which becomes a carousel displayed to the user. These collections are formed using two main sources of information:

- **Side information**, such as genres, actors, or production metadata, which offer direct signals about item similarity.
- **Interaction data**, derived from user behavior, capturing implicit signals of relatedness based on user engagement.

Each collection is assigned a descriptive *label*, and its items are internally ranked using *affinity scores*, which reflect how relevant the item is for inclusion.

Although our study emphasizes algorithmic generation, production systems often rely on a mix of heuristics and editorial strategies. Typical examples include:

- **Genre-based collections**, such as *Comedies* or *Action Movies*, which group items by metadata tags.

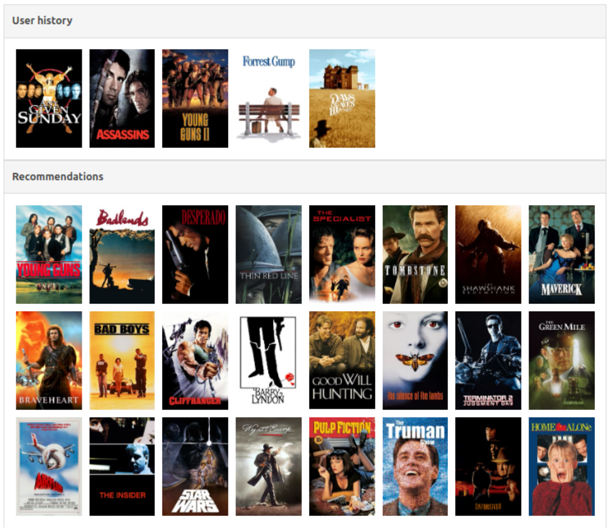

Figure 1: A user's viewing history (top) and linear recommendations generated by the EASE model presented in a grid layout (bottom).

- **Aggregated popularity lists**, like *Top 10 Movies in Czech Republic*, derived from recent user activity.
- **Curated selections**, for instance *Editor's Picks*, created manually by content editors.
- **Franchise groupings**, such as *Marvel Movies*, used to promote branded or serialized content.
- **Conditional lists**, like *Because you watched ...*, generated using relevance-based models.
- **Rule-based filters**, such as *Recently Added*, based on simple metadata internal to the system.
- **Hybrid strategies**, such as *Recently Added Comedies*, which combines multiple types of filtering.

Some collections, such as *Top 10 Movies*, require a fixed order that should not be changed by the recommendation model. Others, like *Trending Movies*, can be re-ranked to better fit the overall layout of the page. In those cases, we use a measure such as "item trendiness" as the affinity score to enable internal prioritization. The model producing trendiness scores (or similar metrics) is external to our framework and considered one of its inputs.

These varied strategies highlight that the problem of defining collections is not purely algorithmic. While we focus on automatic methods, our evaluation framework must account for this diversity. For example, we use TMDB-provided keywords [15] as reference groupings to benchmark collection quality on the MovieLens dataset [6].

To support automatic collection generation, we explored three families of clustering methods:

- **Community detection** on a bipartite graph of users and items. Edges represent interactions. While promising in theory, this approach proved too computationally expensive and yielded poor-quality clusters.
- **Item-item graph embeddings**, where nodes represent items and edge weights are based on co-engagement (e.g., co-ratings or outputs of collaborative filtering methods such as EASE). The resulting embeddings are clustered to form collections.
- **Model-based item embeddings**, obtained from a trained recommender model (e.g., attention-based networks). We extract intermediate representations (e.g., token embeddings) and apply clustering algorithms to produce the collections.

Importantly, real-world systems require items to belong to multiple collections. Overlapping groups must be semantically meaningful: either nested (e.g., *Historical Documentaries* within *Documentaries*) or orthogonal (e.g., *Adventure Movies* vs. *Movies Based on a Book*). Supporting this requires more nuanced techniques than flat clustering. We therefore experimented with:

- **Fuzzy clustering**, allowing soft assignment of items to multiple collections.
- **Hierarchical clustering**, where different levels of the tree can be interpreted as nested collections.
- **Iterative clustering**, where successive runs adapt based on previous outputs—for example, discouraging repeated groupings to produce diverse, non-overlapping collections.
- **Subspace clustering**, which splits the embedding space into subsets of dimensions under the hypothesis that each subset captures a distinct latent factor. Clustering on these subspaces encourages diversity.

3.3 Carousel Layout Optimization

Given a set of candidate collections and per-user item relevance scores, our final task is to assemble an ordered list of carousels. This is framed as an optimization problem with a composite objective that balances the goals described in section 1.

Since the space of feasible solutions is too large for exact optimization, we turn to heuristics that approximate the ideal solution. The first heuristic is inspired by user behavior patterns: users typically start browsing at the top left and move downward through the page and rightward within carousels. This *F-shaped* browsing pattern suggests that a greedy, sequential algorithm—selecting one carousel at a time—may be an effective strategy.

We begin by computing a personalized score for each collection-item pair, combining item-level rankings with collection-level affinity. Collection affinity can be derived from semantic similarity or user preference signals for the underlying concept or label. To prevent excessive duplication, we introduce an *item temperature*, which tracks how often—and how prominently—each item has been shown. Item temperature starts at zero and is updated after each carousel: all temperatures decay by a constant factor, and items selected in the new carousel receive a boost based on their position. At each iteration (i.e., for each new carousel), we update these item scores based on their current temperature and re-rank the items within each collection (excluding those already marked as sorted).

Next, we calculate a score for each collection by aggregating its item scores. Items near the top of the carousel contribute more to the score, using position-based weights that are passed as input to the algorithm. These weights can be tuned based on device type—for example, the importance of top-ranked items may drop off more steeply on mobile devices compared to TV screens, where more items are typically visible without scrolling.

We then select the collection with the highest aggregated score, append it to the final list of carousels, and remove it from the candidate pool. We also update item temperatures—increasing them for selected items and slightly decreasing them for others. This process repeats until we reach the desired number of carousels.

The sequential approach is practical in real-world systems because it supports lazy loading. This means we can serve recommendations to the user without needing to generate the entire page upfront.

While this method is efficient and suitable for online recommendation scenarios, it has clear limitations. Since the algorithm constructs the page one carousel at a time, it can overlook globally better configurations. For instance, placing a highly relevant item at the end of one carousel may prevent placing it in a more impactful position in the next. Likewise, the algorithm cannot decide whether to recommend a group of items as one collection or split them across several carousels for better diversity and coverage.

Alternative search strategies—such as beam search, Monte Carlo sampling, or lookahead scoring—may offer improved performance and are explored in section 6.

Figure 2 shows a final full-page recommendation for the same user presented in Figure 1. While the earlier figure displayed a simple ranked list, this layout highlights how multiple carousels combine to form a richer and more complex recommendation interface. It emphasizes the importance of evaluating not just individual item relevance, but also the layout and diversity of the full page.

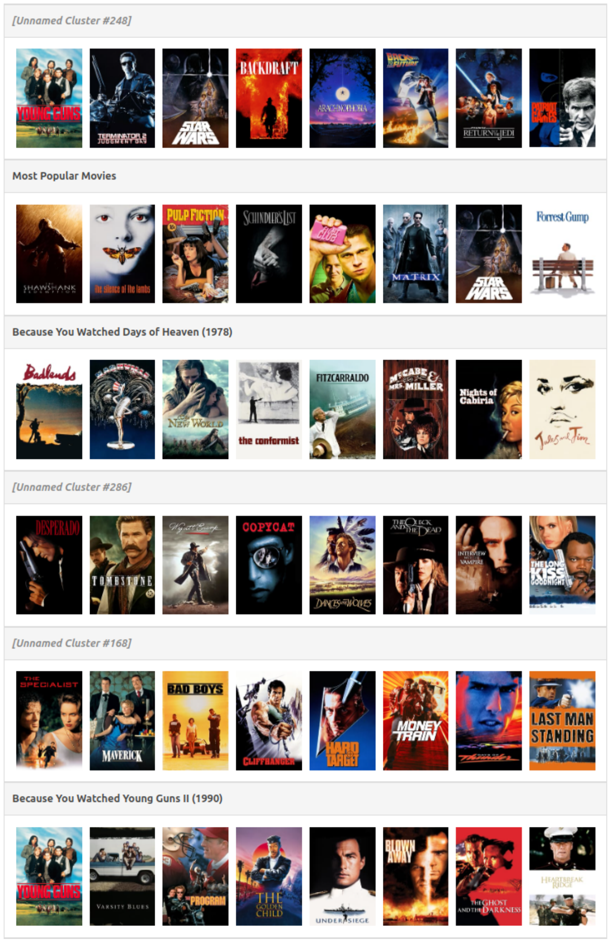

Figure 2: Recommended full-page layout with carousels generated by multiple models. The page combines algorithmic collections with popularity- and similarity-based collections.

4 Evaluation Methodology

There are many accuracy and beyond-accuracy metrics used to evaluate recommender systems. A detailed overview is provided in [9]. However, applying these metrics directly to a multi-carousel layout does not yield meaningful results. It is difficult to rely on offline datasets to accurately assess how satisfied users would be with the presented recommendations.

For instance, applying a metric like DCG to each carousel would favor placing the most relevant items at the top of every list. A system that duplicates the same top items across all carousels would be optimal according to DCG, but clearly suboptimal in practice. Similarly, the adjusted 2DCG metric from [3] can be gamed by placing items in positions with the highest gain, following an F-shaped scan path. While effective for grid layouts, this approach does not promote carousel coherence and is not suitable for our setting.

Evaluating the full-page layout requires a more nuanced approach. Standard metrics must be combined with other goals outlined in section 1.

Therefore, we evaluate individual components of the algorithm separately. This also allows us to study whether these components correlate with final user satisfaction metrics. Specifically, for the item scoring model, we apply standard evaluation techniques that are well-established in the literature.

4.1 Metrics for Collections

When evaluating collections, we must consider both the items included and the assigned labels. Label evaluation is particularly difficult—if not impossible—using offline datasets. It requires either live A/B testing in production or user studies in a controlled setting.

To evaluate collections as item sets, we can use both unsupervised clustering metrics and supervised methods that incorporate interaction data. For example, we may treat collection assignments as the only side information and analyze their effectiveness in cold-start scenarios.

A known issue with clustering evaluation is that it often favors solutions with more clusters, as these tend to better fit the data. This pattern also appears in our setting. To address it, we compare against collections created by randomly assigning items, which we can generate to match any collection size distribution. This provides a fair baseline for comparison.

4.2 User-Behavior Evaluation & Full-Page Metrics

Some studies use a flattening technique to convert the 2D layout of recommendations into a linear list to enable standard evaluation.

This can be done by traversing carousels row-by-row or following an F-shaped scan pattern. However, these strategies do not accurately reflect how users interact with full-page layouts.

We hypothesize that users skip carousels they find unappealing and explore those they find interesting. We propose a new evaluation procedure that mimics this behavior.

In our model, the user starts at the top of the page with some initial *interest* in each carousel. They examine items one by one, increasing their interest if they find relevant content (positive interaction) and decreasing it otherwise. They continue exploring until their interest fades.

This dynamic procedure flattens the page into a single list, which we then evaluate using standard linear metrics. We expect that coherent carousels encourage deeper exploration and result in higher relevance in the flattened list.

Put simply, the user initially sees just a few items in each carousel, and we aim to position the most relevant ones visibly. Remaining items serve as secondary recommendations, visible only if the user chooses to scroll.

We believe that more consistent carousels should perform better. Preliminary experiments comparing metadata-based collections with random ones support this hypothesis.

5 Implementation & Experimental Design

We developed a high-performance library, *Full-page recommender*[1], that efficiently computes near-optimal full-page recommendation layouts. The core algorithm is implemented in Rust for speed and is exposed to Python for easy integration. The library offers a single main method to generate the carousels. The key inputs to this function are a list of collections with titles, the items in each collection, and their respective scores. It supports combining collections from various sources—such as algorithmic, editorial, or popular content—and works independently of the chosen item ranking model. Other inputs act as hyperparameters, e.g. the cooling factor for deduplication or positional weights to reflect each position's prominence.

To evaluate our approach, we used the MovieLens 32M dataset [6], which contains explicit interactions in the form of user ratings. We filtered for all interactions with rating 4 or higher as positive feedback. We used two benchmark rankers: *Pop* (a non-personalized model based on item popularity) and *EASE* [14].

In addition, we implemented a clustering algorithm to group items into collections. To construct these clusters, we generated multiple collaboratively induced graphs representing item-item relationships. Each graph used a different method for assigning edge weights: co-occurrence of positive interactions, similarity scores from the EASE model, or lift between item pairs [5]. We then applied the Node2Vec algorithm [4] to each graph to obtain item embeddings, which were subsequently clustered using k-means.

Using the Full-page recommender library, we generated personalized layouts based on these clusters. We also implemented the evaluation metric described in subsection 4.2 and performed an initial analysis of the layouts. The metric values correlate with visual inspection of the results, although some variation is expected due to subjectivity.

[1]https://github.com/jankislinger/full-page-recommender

Finally, we conducted a sensitivity analysis to explore how input parameters influence the layout. For example, we studied how the cooling factor in item temperatures affects the number of duplicated items. These experiments are still in early stages and require further validation.

6 Conclusion & Future Work

We started by breaking down the complex problem of recommending multi-carousel layouts into three more manageable components. We then implemented several algorithms for the first two components—item scoring and collection generation—using the MovieLens dataset. Finally, we applied these results to our implementation of the Full-Page Recommender. Our initial analysis shows that this approach is viable in terms of both performance and perceived recommendation quality.

Future work includes evaluating generalization and comparing different methods for collection generation. We plan to expand our benchmark suite with additional item-ranking algorithms and datasets to ensure a robust evaluation of the full-page recommender. The evaluation will also incorporate beyond-accuracy metrics to better reflect user experience and recommendation quality.

As discussed in subsection 3.3, our current method adds one carousel at a time. We aim to explore layout strategies that consider multiple carousels jointly—either by evaluating partial layouts in depth, or by estimating the remaining collections' potential contribution.

One of the least explored areas is how to label the carousels. We plan to use language models to assist in generating collection titles. However, we currently lack a clear approach for evaluating the quality of these titles using offline datasets.

We are also interested in developing an end-to-end model for layout generation. Our goal is to train a generative model (e.g., LSTM or Transformer) to learn to approximate optimal layouts for real-time use. This approach would rely on running expensive optimization steps offline and training the model to mimic the resulting layouts. The trained model could then be deployed in production to generate layouts efficiently.

Lastly, we plan to test the system in a live A/B test on a video-streaming platform. This would allow us to connect offline evaluation results with real-world user engagement. Such results could also help us fine-tune model parameters offline to improve production performance.

Acknowledgments

This research was supported by the student grant project SGS23/210/-OHK3/3T/18 of the Czech Technical University in Prague.

References

[1] Rodrigo Alves. 2024. Regionalization-Based Collaborative Filtering: Harnessing Geographical Information in Recommenders. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 10, 2 (May 2024). https://doi.org/10.1145/3656641

[2] Ehtsham Elahi and Ashok Chandrashekar. 2020. Learning Representations of Hierarchical Slates in Collaborative Filtering. *CoRR* abs/2010.06987 (2020). arXiv:2010.06987 https://arxiv.org/abs/2010.06987

[3] Maurizio Ferrari Dacrema, Nicolò Felicioni, and Paolo Cremonesi. 2022. Offline Evaluation of Recommender Systems in a User Interface With Multiple Carousels. *Frontiers in Big Data* 5 (2022). https://doi.org/10.3389/fdata.2022.910030

[4] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining. ACM. https://doi.org/10.1145/2939672.2939754 arXiv:1607.00653

[5] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann, Waltham, MA.

[6] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4, Article 19 (Dec. 2015). https://doi.org/10.1145/2827872

[7] Dietmar Jannach, Mathias Jesse, Michael Jugovac, and Christoph Trattner. 2021. Exploring Multi-List User Interfaces for Similar-Item Recommendations. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '21)*. ACM, Utrecht, Netherlands. https://doi.org/10.1145/3450613.3456809

[8] Michael Jugovac and Dietmar Jannach. 2017. Interacting with recommenders in a user interface: presentation styles and interaction modalities. *User Modeling and User-Adapted Interaction* 27, 1 (2017).

[9] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7, 1 (December 2016), 42 pages. https://doi.org/10.1145/2926720

[10] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. *arXiv preprint arXiv:1808.09781* (2018). https://arxiv.org/abs/1808.09781

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[12] Behnam Rahdari, Branislav Kveton, and Peter Brusilovsky. 2022. The magic of carousels: Single vs. multi-list recommender systems. (2022). https://www.amazon.science/publications/the-magic-of-carousels-single-vs-multi-list-recommender-systems

[13] Sanidhya Singal, Piyush Singh, and Manjeet Dahiya. 2021. Automatic Collection Creation and Recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems (RecSys '21)*. ACM, Amsterdam, Netherlands. https://doi.org/10.1145/3460231.3478865

[14] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *Proceedings of the World Wide Web Conference*. ACM.

[15] TMDB. 2024. *The Movie Database (TMDB)*. https://www.themoviedb.org

[16] Liang Wu, Mihajlo Grbovic, and Jundong Li. 2021. Toward User Engagement Optimization in 2D Presentation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM '21)*. ACM, Virtual Event, Israel. https://doi.org/10.1145/3437963.3441749

[17] Yunjia Xi, Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Rui Zhang, Ruiming Tang, and Yong Yu. 2023. A Bird's-eye View of Reranking: From List Level to Page Level. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM '23)*. ACM, Singapore, Singapore. https://doi.org/10.1145/3539597.3570399