When Volatility Reigns: Learning to Rank Recommendations in Second Hand Marketplaces

Jose San Pedro*
Jordi Esteve†*
jose.sanpedro@adevinta.com
j.esteve.sorribas@gmail.com
Adevinta
Barcelona, Spain

Victor Codina
victor.codina@adevinta.com
Adevinta
Barcelona, Spain

Sandra Garcia Esparza
sandra.garcia@adevinta.com
Adevinta
Barcelona, Spain

ABSTRACT

Second-hand marketplaces exhibit unique features that make them a challenging problem setting for recommender systems. Users interact with the system to buy and sell items, which in turn makes the inventory highly volatile: cold start is not only a problem, but rather an organic characteristic of these systems. The lifespan of items is measured in hours or days, instead of months or years. And the catalog abounds with near-duplicate items, as many different users try to sell their own instance of a product (e.g. a specific model of smartphone). This paper presents the results obtained from testing different recommendation strategies in this challenging context. Both collaborative filtering and content-based methods are evaluated using a three-fold methodology: offline evaluation, user study, and finally A/B test. The results found have relevant implications for the design and evaluation of recommender systems in the application domain of second-hand marketplaces.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

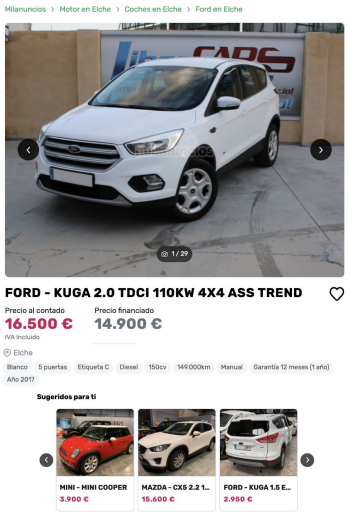
marketplace, recommender systems, near-duplicate, volatility

ACM Reference Format:
Jose San Pedro, Jordi Esteve, Victor Codina, and Sandra Garcia Esparza. 2022. When Volatility Reigns: Learning to Rank Recommendations in Second Hand Marketplaces. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Recommendations are a popular tool used in many e-commerce applications. Their goal is to facilitate the discovery of relevant content to potential buyers and ultimately increase user conversion and satisfaction. While the catalog of purchasable products is organically dynamic, most have a minimum lifespan of several years where users can interact with, add to basket and purchase them over many different sessions. In this process, users interact with similar and complementary products, that enable the successful use of Collaborative Filtering (CF) methods to learn effective recommendation models.

Figure 1: Related items recommendation in the detail page of an ad in Milanuncios.



This paper presents a case study of recommender systems applied to second-hand marketplaces (or just, marketplaces). While well within the e-commerce domain, marketplaces present a number of unique challenges that limit the application of traditional recommender systems literature in e-commerce. Marketplaces are fully user-driven: supply and demand of goods respond to the needs of users wanting to obtain a monetary incentive for items they no longer need (i.e. sellers) and users that prefer to satisfy their purchasing needs with pre-owned items (i.e. buyers).

In such a type of marketplace, items emerge and vanish as a response of users listing items for sale and finally transacting. Each item listed is unique: regardless of the actual product listed (e.g. iPhone 12), each item has been used by a different user and in a different way. Some may have light use, some others heavy. Some may feature a mint condition, while others may present visible blemishes. Beyond the differences that the many instances of a single product may have, the main challenge for creating effective recommendations is the individuality of such items within the system: each listed item is uniquely identified by its own id. This

*Both authors contributed equally to this research.
†Author contributed to this work while being affiliated with Adevinta

SIGIR eCom'22, July 15, 2022, Madrid, Spain

vastly increases the sparsity of the item space. Moreover, once sold it is no longer available in the platform. This hampers the ability to leverage from the knowledge collected about items during their lifetime, which depending on their category and popularity, can span from hours to a few weeks.

As a result of these characteristics, marketplaces exhibit an organic and permanent cold-start scenario. Ideally, if the marketplace can create matches between buyers and sellers efficiently, then items would be sold and removed from the platform before enough data to train an effective collaborative filtering model can be collected. Moreover, the item catalog is populated by near-duplicates that pose difficulties not only for training relevance models, but also for conducting offline evaluation, because the relevance of near-duplicates cannot be easily generalized. Hence, in this setting offline accuracy measurements tend to underestimate the potential of a model to recommend relevant items: relevance is assessed as a function of the items interacted with by users (i.e. viewed, saved, ...), thus excluding unseen but probably relevant items.

In this work, we focus on one of such marketplaces: Milanuncios[1], one of Adevinta's largest platforms in Spain. As in many other marketplaces, Milanuncios offers a wide variety of item categories to organize its catalog, which are grouped into four main verticals: Generalist, Motor, Real Estate and Jobs. Each vertical presents unique characteristics; ads are not created and browsed equally for buying and selling houses or smartphones. Solutions for recommending items should cater for such differences to optimize the expected impact per vertical. We consider the specific case of the Motor vertical, and in particular focus on the cars category. This category is one of the primary growth pillars for Adevinta [2], and provides an interesting problem setting where structured metadata related to car characteristics is available, enabling learning opportunities that can be leveraged for building effective content-based recommender systems. We implemented such a recommender, and present a deep dive study comparing it to several baselines.

Our study considers the "related-items" recommendation scenario, where the item currently being viewed is used as a query to retrieve similar related items, aiming at helping users navigate the catalog and find relevant items (see Fig 1). This is the most common application of recommenders in Milanuncios, and is expected to have the maximum impact for its users.

The contribution of the paper is two-fold. First, the paper compares the effectiveness of different recommendation strategies, including collaborative-filtering and content-based, in the domain of second-hand marketplaces. The evaluation is performed using both offline and online methods. Second, the paper provides experimental evidence of the offline-online gap in the recommenders problem setting, by comparing the results of two offline evaluation methods (standard evaluation using a held-out dataset and user study) with the results obtained in an online controlled experiment. Both contributions have relevant implications for the design and implementation of recommender systems in the context of marketplaces.

The paper is organized as follows. We review previous relevant literature in Section 2. Section 3 formalizes the recommendation problem and describes the baseline strategy, based on CF, currently used in our production environment. Section 4 introduces a content-based recommender suitable to our application domain. Section 5 and 6 provide offline evaluation results of all the strategies. The results of comparing the strategies in an A/B test are reported in Section 7. We conclude with Section 8.

[1] http://milanuncios.com

[2] https://www.adevinta.com/stories/articles/adevinta-launches-growing-at-scale-its-new-strategic-plan-designed-to-accelerate-its-profitable-growth

2 RELATED WORK

Cold-start. Cold-start, where interaction data about a user or item is not available in the system, is a common problem in recommender systems, specially for CF methods. The item cold-start problem can be solved with two different approaches: i) hybrid solutions that incorporate both behavioural and content information from the item (e.g. price) [1] and ii) pure CF that dynamically updates the item representations as more clicks are gathered [2, 3]. However, these two strategies do not consider cold-start as the norm; their goal is to expose cold-start items to users to collect enough data to generate a stationary representation of them. In our setting, items are transient: once sold, they leave the catalog. Our goal is to increase the liquidity of the marketplace: increase the rate of transactions, and hence decrease the lifetime of items. Aiming at collecting data for the purpose of generating a stationary representation introduces a conflicting interest.

Methods purely based on content for generating recommendations (CB) present their own drawbacks, with some authors claiming that content alone does not provide enough information to generate relevant recommendations [4]. In this work, we build a recommender that relies on a learning to rank approach using only content features but leveraging behavioural signals during training.

Learning to rank for recommender systems. The goal of learning to rank (LTR) is to improve the ranking of documents by applying machine learning techniques [5]. Traditionally, learning to rank has been applied to Information Retrieval (IR) problems, commonly to rank search results given a search query. In the last years learning to rank has also been successfully applied to recommender systems in many domains such as in music [6], fashion [7] and marketplaces [8]. In the context of recommendations, a query often corresponds to a user or an item for which we want to provide recommendations. To alleviate the computational complexity of ranking the full catalog during the inference stage, production systems are generally designed so that predictions are only computed for a subset of documents, called *candidates* [8, 9].

In this paper, we built a content-based pairwise ranking approach following the LTR paradigm. This ranking model uses structured and unstructured content features, generated from the ads metadata, to compute the similarity between pairs of items. Our underlying hypothesis is that, by effectively leveraging user generated content, we can overcome the limitations of CF in this particular domain.

Evaluation. Before deploying or testing a new algorithm in production it is common to conduct an offline evaluation to assess its potential without the productionization cost and the risk of harming the user experience [10]. Recent work in the recommenders area has shown the existence of a significant difference between offline and online performance, *offline-online gap*, which arises from different factors, such as biases that may unfairly favour certain algorithms [11]. This problem is especially significant in

domains with high sparsity, where exact matches do not necessarily correlate with relevance as highly as in other domains.

To address these limitations, some authors conduct user studies, which have shown to provide a closer approximation to the quality of recommendations than offline methods [12]. An advantage of user studies is that they are not affected by biases commonly found in offline evaluation. For example, in [13] a user study is presented where different CF and CB algorithms for related item recommendations are benchmarked, concluding that CB approaches provided a better match of item similarity with regards to the user's expectations. Our work is similar to [14], in which the authors evaluate different algorithms for the home improvement domain. The limitations of offline evaluation in their application setting to properly assess similar, but not exact, products motivates them to conduct a user study in addition to the offline evaluation to choose the algorithm to use in production. In our case, this problem is magnified by the extreme volatility and abundant presence of near-duplicate items. We provide a comparison of the performance of several recommendation strategies using these two evaluation methods to assess their suitability in our context by comparing their outcomes with the result of an online controlled experiment.

To the best of our knowledge, this is the first work that provides an in-depth study of recommenders in the marketplaces application domain, where volatility, item cold-start and near duplication are dominating factors. This work debates the effectiveness of different strategies in this domain.

3 PROBLEM DEFINITION AND CURRENT SOLUTION

We solve the related-items recommendation task as a surrogate problem of estimating the similarity between a pair of items. This similarity can be inferred from behavioural and/or content data. Formally, given a source item that belongs to catalog I, we derive a function that retrieves the most similar items $T \subset I$ and ranks them in decreasing order of similarity, according to a specific definition of similarity. Our goal is to provide the top-20 recommendation list for every item in the catalog, which is the maximum size of the related items widget in the application.

3.1 Baseline Recommender

Our baseline recommender is a *mixed* hybrid strategy [15]. This hybrid model combines the recommendations generated by two item-to-item similarity models: a CF approach where similarity is based on item co-ocurrences, and a CB similarity approach that only relies on text matching. Our hybridisation strategy consist of appending the new items retrieved by the CB recommender to the CF recommendations. We call our baseline recommender *backfill-hybrid*.

CF approach. Our method builds on top of the Adamic-Adar's index measure [16] that calculates item proximity by looking at the amount of shared neighbors, i.e. users that clicked on them in our case. A correction is applied based on users' activity, as to make the contribution to the score of users with too many neighbors less significant than users with fewer connections. We also apply a time decay function that reduce the contribution of older item co-occurrences to favor more recent interactions. Finally, we filter out recommended items with scores below a minimum threshold for controlling the trade-off between relevance and coverage. We will refer to this model as *Adamic*. We have experimented with more sophisticated CF methods, including matrix factorization-based approaches, e.g. lightFM [1], and sequence-based like Prod2Vec [17]. We did not observe a significant enough gain in our online metrics to justify the added complexity of training and serving these models.

CB approach. The method we use is BM25F [18], a widely used text similarity algorithm in the IR field with the ability to exploit document structure via content field weighting. Specifically, we calculate the BM25F score for each pair of items using the following three fields: title, description and price. To this end, we preprocess text fields by applying standard tokenization and stopword removal. Price is bucketized using a fixed bin size to enable a more effective matching. Finally, we only score item pairs in the same geographical region to improve the scalability of the solution given the large size of our item inventory. We will refer to it as *BM25F*.

3.2 Implementation Details and Limitations

Currently the two aforementioned candidate generation approaches are implemented as batch Spark jobs that generate recommendations in an hourly fashion. Recommendations are then stored in a in-memory key-value store for fast online retrieval. The backfilling aggregation is performed in real time. The main advantage of our baseline hybrid recommender is primarily its low complexity and satisfactory accuracy-coverage trade-off. However, this simplicity comes with some limitations.

On the one hand, any CF approach is very sensible to the cold start problem, short item lifetime and abundance of near-duplicate items. We heuristically mitigate these limitations by hourly updating our CF model. On the other hand, our CB approach is based on text matching and cannot capture complex relationships at feature level from the data, e.g. users might be willing to travel further for a good deal. Additionally, BM25F deals poorly with numerical features (e.g. car price) and categorical features (e.g. car model), and our hypothesis is that they are crucial to provide relevant recommendations. Although, *hybrid-backfill* is robust to cold-start, the top positions of the recommender are populated with CF recommendations, and so the aforementioned limitations still hold.

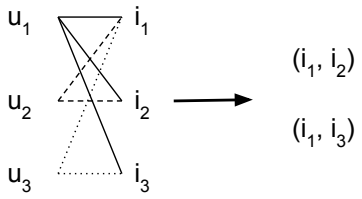
4 LEARNING TO RANK RECOMMENDATIONS

To mitigate the aforementioned limitations, we built a content-based ranking model that leverages both structured and unstructured metadata to learn similarities at a product level using LTR.

By learning similarities at the product level, we alleviate the issues caused by high catalog volatility: items are unique and can just be bought by one user, but products are stationary. To learn such similarities, we need to be able to infer complex relationships among metadata fields of different types, which simple models such as BM25 are unable to. An example of such interplay between metadata fields would be the relation between price and distance: a good deal would encourage users to travel further.

Formally, our solution is as follows: given a catalog I with $|I|$ items, we define the nth item $i_n \in I$ as a vector of content features $[f_{1n}, ..., f_{Mn}], \forall f_{mn} \in \mathbb{R}$ where M is the number of features. Then, our goal is to find a non-linear function g that best sorts a set of

Figure 2: This image depicts the process to extract similarity labels from the data. We generate a user-item (u_i-i_i) bipartite graph, from which we extract co-clicked items. In this example, just items that are co-clicked at least by two users are considered similar.



items $T \subset I$ for a given source item $s \in I$ using to this end the feature representation built, which is independent of the unique item identifier.

4.1 Defining Similarity

Learning to rank requires having a clear notion of similarity (or relevance) between queries and documents. In our problem setting (related items recommendations), both queries and documents are items of the marketplace. Relevance for each pair of items can be obtained from explicit assessor judgments or inferred from implicit data [19]. This work considers the latter scenario.

As already stated, we assume that users tend to interact with similar and related items during their sessions. The more users that co-interact with a given pair of items, the stronger the evidence to consider these items relevant to each other. This process is visually described in Figure 2. With this assumption, we can use item co-occurrences across users as our similarity proxy. We can adjust the level of agreement required to consider pairs of items relevant by defining a minimum co-ocurrence threshold. In our setting, we found empirically that 2 users gave enough evidence of the similarity between pairs of items.

Negative signals, i.e. pairs of items that weren't co-clicked, are more common than positives and need to be subsampled. Negatives were sampled uniformly at random with a ratio of 50:1. We found this ratio to be a good trade-off between training complexity and task learning in this scenario.

4.2 Feature Processing and Generation

This section introduces the feature space used to represent our sample items. Given that the goal is to endow the model with the ability to abstract individual items into classes (or products), it is crucial that the representation chosen captures as many facets as possible of the items characteristics.

4.2.1 Feature processing. Table 1 lists the metadata available to represent items in our marketplace for the specific category of cars. The list includes a variety of categorical, numerical and unstructured textual variables. It is worth noting that all these item metadata are user generated and, as such, are subject to noise and inconsistencies resulting from the lack of a controlled data acquisition process. For instance, the presence of typos and capitalization differences, artificially increase the sparseness of categorical variables. Also, the data is inherently unbalanced, as some brands and price ranges are largely more popular than others.

Table 1: List of content data and their type. For a snapshot of the catalog, we show the type the numeric data and the percentage of missing values for each content feature, the cardinality of the categorical and ordinal data and the inter quantile range (IQR) of the numerical data. IQR for geolocation and publication date are not disclosed for confidentiality.

Name	Type	Cardinality/ IQR	% missing
raw model	free text	36,268	4
model	categorical	36,268	4
brand	categorical	70	0
fuel type	categorical	10	5
seller type	categorical	2	0
seller unique identifier	categorical	129,459	0
color	categorical	8,326	7
transmission type	categorical	6	6
price	numerical	4,300-20,000	0
mileage	numerical	46,000-168,344	5
registration year	numerical	2008-2018	5
geolocation	numerical	-	0
publication date	numerical	-	0
number of doors	ordinal	7	6

We carried out a feature processing stage specifically for categorical and text variables consisting of two steps: normalization and encoding. In the normalization step, accents were removed and text was converted to lower case.

During the encoding step, we aimed at reducing the sparsity of categorical variables. There are many methods, of varying complexity, to produce effective and compact categorical features from user generated content: target encoding, hash encoding, entity embeddings [20], etc. In the context of our problem setting, where only car brand and color had to be re-encoded, we decided to run this process manually to preserve the well known semantics of these two variables. We conducted an analysis of the distribution of car brands in the marketplace catalog. We found that car brand follows a long-tail distribution, so we decided to keep the most representative brands, by popularity, in the catalog (a total of 18) and clustered the remaining brands into six additional classes with clear associated semantics (e.g. luxury cars, off-road 4x4, ...). A similar process was followed for the color variable, resulting in 11 disjoint color classes.

4.2.2 Feature generation. Learning a ranking model requires capturing the similarity between queries and documents in the feature space used for representing instances. In this section we describe the features generated using the metadata fields at our disposal:

Similarity features are a function of items pairs (namely source and candidate) and aim at capturing different dimensions of the similarity between them. We use the item metadata listed in Table 1 to compute the following set of features for each pair of items:

Table 2: Configuration of the top 3 best models and their MAP@5 on the validation set. The first four hyper-parameters are of structural type and the remaining are learning parameters. The number of trees (num_tree) was controlled with early_stopping_round, which was set to 10. We set scale_pos_weight equal to the the negative sampling ratio to control class imbalance. Hyper-parameters names respect the name in the documentation.

num_tree	num_leaves	max_depth	min_data_in_leaf	learning_rate	subsample	colsample_bytree	reg_lambda	MAP@5
-	[5-3000]	[3-30]	[200-1000]	[0.01-1.0]	[0.4-1.0]	[0.4-1.0]	[0.01-100.0]	-
143	1140	24	509	0.22	0.67	0.74	1.87	0.811
195	525	25	568	0.15	0.65	0.50	15.23	0.810
93	**290**	**30**	**269**	**0.34**	**0.68**	**0.62**	**64.27**	**0.807**

- Geodistance with Haversine distance
- Absolute and relative distance (as a percentage) for price, mileage and registration year
- Car model similarity: cosine similarity between the items model represented as TF-IDF vectors.
- Boolean similarity for: unique seller identifier, seller type, number of doors, transimission type, fuel type and color.

Standalone features are functions of individual items and aim at enriching the model with the ability to incorporate user preferences and biases towards specific cars and car characteristics. We include the freshness of the ad (i.e. the number of days since it was published), and the categorical brand and model of both source and candidate items, also looking at the possibility to find interactions between the brands themselves (e.g. users searching for Renault may be willing to buy Peugeot if there is a good deal) and also between brands and other features (e.g. geographical distance may not be as important for buying a luxury car as it is for a more conventional vehicle).

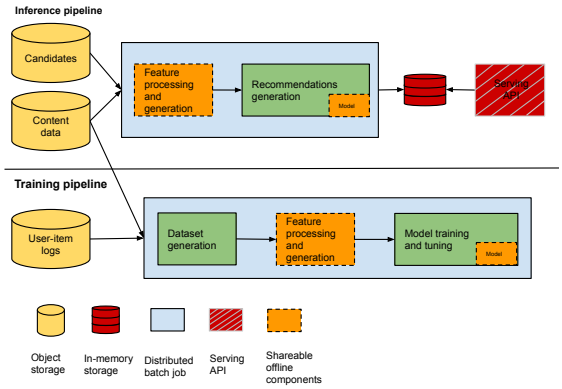
As a result of this feature generation processes, each pair of source and candidate item was represented by a vector of 20 features, 15 similarity features and 5 standalone features.

4.3 Ranking Model, Training and Tuning

As described in Section 4.2, the input of the model is represented as tabular data. Gradient boosting Decision Trees (GBDT), which have been successfully used for ranking [6, 21], are the state-of-art for this type of data and easier to train than other more complex models [22, 23]. Among the several GBDT algorithms, we selected LightGBM [24] because it handles categorical features and missing values (see Table 1), as well as for its efficiency and scalability.

To train the model, we collected a dataset using one full week of interaction data from the cars category of Milanuncios, which amounted to several million pairs of co-occurring items. We used the processes presented in Section 4.1 and 4.2 to build the item pairs representation. This dataset was split at the query level into 70% training, 20% validation and 10% test sets. We tuned the model hyper-parameters using the standard hold out approach and we considered two types of hyper-parameters: structural, which define the physical structure and complexity of the model, and learning parameters. Table 2 lists the hyper-parameters that we tuned, including the considered ranges. The remaining hyper-parameters were left at their default value[3]. We used the Tree Parzen Estimator tuning strategy [25] to optimize the search process, and the Katib framework to orchestrate the whole process [4]. The objective function used for the lightGBM model was LambdaRank [26] and the evaluation metric optimized during the tuning process was Mean Average Precision at a cutoff level of 5 (MAP@5).

Figure 3: Ranking system architecture. The inference runs hourly to update the recommendations according to the latest catalog snapshot.



The top 3 best models, in terms of MAP@5, found during the tuning process are presented in Table 2. We selected the third best model because of its complexity-performance trade-off: it has substantially fewer trees at the expense of a marginal decrease in MAP@5 (< 0.5%). This reduced model complexity allows to run inference at a fraction of the time it would take for the top 2 models, reducing the latency of generating recommendations. The performance of this model in the test set was MAP@5=0.768.

To analyze the contribution of each feature to the prediction model, we used Shaply values [27]. We found that the most important features are related to price, registration year, car model and geolocation. These findings help to validate our hypothesis that numeric and categorical features carry relevant information for modeling item similarity, which limits the ability of simpler methods (i.e. BM25F) to effectively perform this task.

4.4 Architecture and Candidate Generation

The system architecture used for generating recommendations is depicted in Figure 3. The training pipeline works offline and periodically updates the ranking model to capture seasonality and broad

[3]https://lightgbm.readthedocs.io/

[4]https://www.kubeflow.org/docs/components/katib/

behavioural shifts in user preferences. The inference pipeline combines an offline and online components. The offline side computes recommendations for all items in the catalog using the pairwise similarity model in a batch process, and saves them in a fast in-memory storage. The online side serves client requests for recommendations, which are retrieved by directly querying this storage.

Because of the quadratic complexity of the pairwise inference process, the inference pipeline makes use of a preliminary candidate generation stage. Many candidate generation strategies have been proposed; for instance, combining heuristic business rules with models based on text matching algorithms and click aggregations [8], or methods to encode the catalog in a latent space and retrieve candidates with approximate nearest neighbours queries [9]. Our candidate generation strategy is similar to the former, where we use as candidates the items retrieved by the two baseline recommender strategies, Adamic and BM25F. We did not pursue any additional candidate generation strategies as the main focus of this work is on the ranking phase. Using this candidate generation model, we obtained an average of over 80 candidates for every source item. These were then scored by the ranking model, and the resulting top 20 finally stored to be returned by the API (section 3). We will refer to this full solution as *ltr-car*.

5 OFFLINE EVALUATION

5.1 Experimental Design

To mimic as much as possible our production setting, in which recommendations are updated in hourly fashion, we follow a sliding window evaluation method using test sets of 1-hour size and a strict time-dependent condition to ensure that all the test interactions are always more recent than any training interaction [28]. For this experiment we used all the hours of a given day, i.e. 24 test splits.

Our evaluation protocol is based on a next item prediction scheme, similar to recent work for evaluating session-based recommenders [29], since we believe it resembles closely our production setting[5]. In this evaluation protocol, user sessions are treated as the ground truth, and the task of the evaluated system is to predict, for a given item in the session, the immediate next k items. Differently from the original scheme, where items are iteratively revealed one after another to simulate the user journey throughout a session, at each iteration we only consider the current item as source. We empirically found that using 10 iterations per session and the 10 subsequent items (k) per source as maximum produces the most reliable results in our test data in terms of variance across splits. After applying the aforementioned splitting protocol we obtained in the order of tens of thousands distinct test source items per split.

5.2 Results and Discussion

Table 3 shows the results of the evaluated algorithms in terms of Mean Average Precision at 5 (MAP@5) as a ranking metric, and F1 score (*F1@5*) as a relevance metric.

Comparing the candidate generation approaches individually, we observe that BM25F vastly underperforms Adamic (-84% in both metrics), supporting the case that CF outperforms CB in offline evaluations. As expected, the difference between Adamic and backfill-hybrid is minimal, given that Adamic is always placed at the top of the ranking. Opposite to our intuition, the proposed ltr-car model features a substantial performance decrease (-57% MAP@5 and -46% F1@5) compared to backfill-hybrid.

[5]We also tried a non-session based protocol but resulted in metrics with higher variance, possibly caused by irrelevant items from different sessions

Table 3: Metrics correspond to the mean value across all test splits along with their 95% confidence interval.

Model	MAP@5	F1@5
Adamic	0.0480 ±0.0018	0.0648 ±0.0011
BM25F	0.0077 ±0.0003	0.0107 ±0.0003
backfill-hybrid	0.0494 ±0.0019	0.0617 ±0.0011
ltr-car	0.0211 ±0.0005	0.0329 ±0.0005

The offline evaluation results did not match the expectations we had for the proposed ltr-car model. Although we were aware of the limitations of offline evaluation, there was a real concern that the new method may harm the user experience during an A/B test. We conducted a blind user study to collect additional information about the performance of the proposed ltr-car method.

6 USER STUDY

6.1 User Study Design

The study aimed at understanding the preference of users for different recommendation strategies. In particular, we considered the two baselines described in Section 3.1 as well as our proposed method.

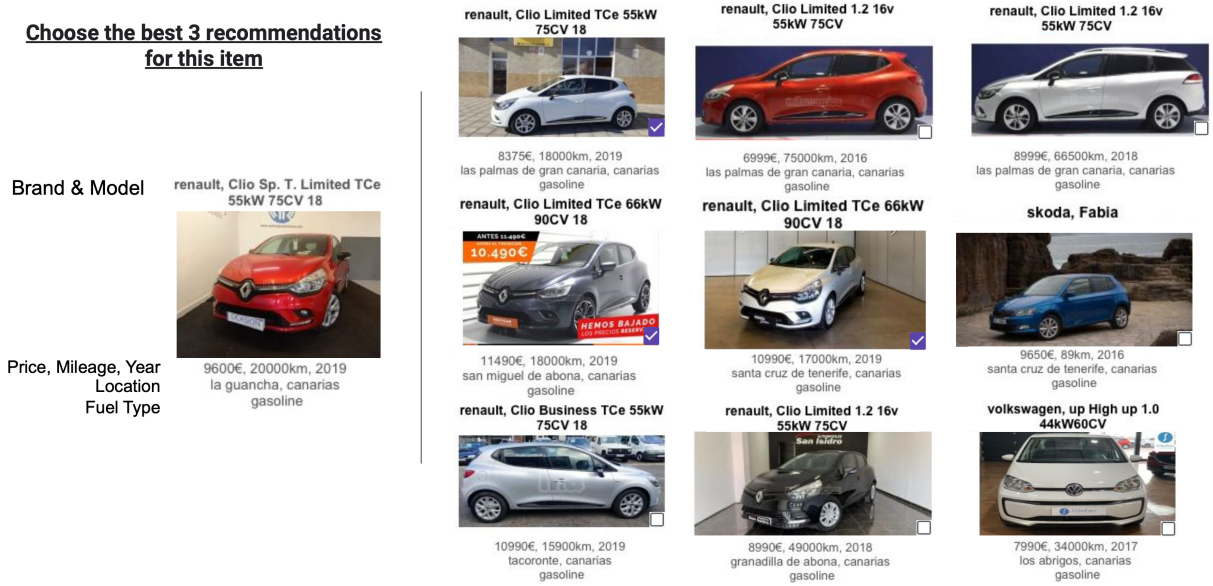
In the study, we presented tasks to participants to collect evidence of these preferences. Each task corresponds to a single source item (i.e. a car for sale in the marketplace), and shows the three top recommendations for Adamic, BM25F and ltr-car in a 3x3 grid. Notice that for the top 3 recommendations the backfill-hybrid is essentially the same as Adamic (see Section 3.1). We selected the top 3 recommendations to simplify the tasks to participants, and is in consonance with the UI design of the marketplace where only the first 3 recommendations are visible as shown in Fig 1.

The participants were agnostic to the strategy used to generate each recommendation of the grid. For each task, they were asked to select the 3 best recommendations using a checkbox next to each recommendation. Figure 4 depicts the web interface used to collect the responses. To aid in the task, participants could readily inspect some relevant metadata fields (car brand and model, mileage, …) as well as a photograph for both the source and recommended items.

To minimize the potential influence of different biases in the results of this study we took the following design decisions. First, we established that each task should be assessed by two participants independently. The combined set of judgments per task was used to determine the final score for each strategy. Furthermore, task assignment was performed uniformly at random by pairs of participants, to promote pair-based diversity in the collection of ratings. Second, the 3x3 grid presented to participants was randomly shuffled to mitigate the effect of the presentation bias.

An important point in the study design was the source selection strategy, i.e. what source items and recommendations would be used to compile the set of tasks. Ideally, we want to assess as many recommendations as possible, but the cognitive effort required to solve each task adds up quickly, so sub-sampling was required. We

Figure 4: Web interface used for the tasks of the User Study. Participants had access to metadata to help them find the best recommendations.



decided to stratify the item catalog by brand to generate a representative sample: for the top 10 most prevalent car brands in the catalog we randomly sampled 10 vehicles, for a total of 100 unique source items. The assignment process took the brand stratification into account to distribute brands evenly across participants.

A total of 12 participants (3 female) whose ages ranged from 27 to 48 years old took part in the study. All of the participants were workers of the company, and held a variety of occupations, including scientists, engineers, and analysts. The main limitation of this study is that this small group of participants is not representative of the overall population. The study was implemented as a website accessible to participants online. All of the participants completed the assigned tasks (between 16 and 17) for a total number of 200 tasks, corresponding to 100 sources assessed by 2 participants.

6.2 Results and Discussion

Participants' selections were used as relevance indicators to compute a performance score for each recommendation strategy. Firstly, we combined the responses of the two assessments given per task to obtain a binary relevance judgment per recommendation. With these relevance judgments, we could then compute the average precision for each task, and ultimately the Mean Average Precision (MAP@3) for each strategy averaging the metric across all tasks.

Because each of the three recommendation strategies is independent of the other two, it was possible that two would simultaneously choose one or more identical items in their top 3. In these cases, when an item that was recommended by two strategies was chosen by a user, it was considered as relevant for both strategies.

We show the MAP metric obtained for each strategy in Table 4. We can observe that participants of our user study had a clear preference for the recommendations generated by the ltr-car strategy.

Table 4: Results of the user study. MAP is significantly higher for the ltr-car model, contradicting the findings of the offline evaluation. Significant differences are marked with ""**

		Pairwise differences (p-value)		
	MAP@3	BM25F	Adamic	ltr-car
BM25F	0.304	-	-	-
Adamic	0.311	0.067 (1.0)		
ltr-car	**0.638**	0.334 (0.000**)	0.327 (0.000**)	-

We carried out a Kruskal-Wallis test to establish the statistical significance of the differences between strategies, which resulted in a strong rejection of the null hypothesis (H=69.44, p<0.001). After finding significance in the results, we computed a series of post-hoc Dunn's tests [30] to establish significance at the pairwise level. This test showed a significant difference between ltr-car and the two baseline methods.

These findings are not aligned with the results of the offline evaluation, indeed leading to the exact opposite conclusions. The ltr-car model, despite achieving a poor MAP performance in the offline evaluation process, vastly outperforms the baselines using the same metric when users directly compare the recommendations generated by each strategy. Although the user study considers a small set of recommendations, which limits the ability to generalize these results, it provides enough support to conduct an online test to measure its performance when used by real users in the wild.

7 ONLINE EVALUATION

The user study results provided us enough confidence to carry out an A/B test to compare the two approaches studied: our production baseline (backfill-hybrid) and the proposed solution (ltr-car).

7.1 Online Metrics and Experiment Design

Measuring online performance is not always straightforward. While the ultimate goal is to drive a higher revenue for the marketplace, previous literature suggest that recommenders may not be able to increase short-term revenue [31], as users can achieve their purchasing goals using alternative navigational tools. In our domain, we have the added difficulty that transactions do not normally happen in the context of the marketplace; buyers and sellers often meet face to face to this end, limiting our ability to observe recommender-induced transactions. For all these reasons, measuring a meaningful change in revenue in the span of a few days is unfeasible.

As a usability and navigational tool, recommender systems have a two-fold objective for our users and marketplaces: first, improve the user satisfaction with the overall experience with our marketplaces, by making relevant content readily available and easy to browse; second, increase the efficiency of the marketplace to make successful matches between buyers and sellers.

Given the absence of transactional information, the stronger signal of interest in an item comes from users contacting each other using the marketplace-provided tools (e.g. via phone call, email or direct message). These interactions between users, which we refer to as *lead*, can be used to measure interest and success in our users sessions. In particular, we can define success at the session level by computing the *percentage of sessions with at least one lead*. Analogous metrics, such as percentage of sessions ending in a transaction, are commonly used in the e-commerce domain for measuring success [32]. In our context, user sessions are defined as periods of uninterrupted activity in the marketplace. We use a threshold of 30 minutes of inactivity to split sessions. This metric (Session Success Ratio) captures the ability of the marketplace to effectively match buyers and sellers in every visit, and is influenced by the relevance of recommendations showed to users.

In addition to this success metric, we also computed two additional secondary metrics. First, we are interested in the lead to view ratio (i.e. average number of views that users need to do before they *lead*) as a way to measure the effort needed by users to find compelling items. Second, we also want to measure user engagement with the content clicked, and we used dwell time to this end. Dwell time was directly computed from server-side events, which has been shown to be as reliable as client-side measurements in specific applications [33], which we capped at 150 seconds to remove outliers and increase metric sensitivity [34].

The experiment was conducted on the ad detail page (see Figure 1) in the cars category of the Milanuncios marketplace. The target population were logged in users of our iOS platform application that visited the ad detail page of a car during the duration of the experiment. Users that met this criteria were split between control and treatment uniformly at random with an even 50:50 split, so they would be exposed to one of the recommendation strategies consistently. We only logged events coming from the cars vertical for the subsequent analysis and computation of the aforementioned metrics. The experiment lasted for 15 days. We used a standard two-tailed t-test to compare the results between control and treatment. With the given experiment duration, and the traffic volume and population size of the marketplace (which we can't disclose for confidentiality reasons), we could compute the statistical significance of the difference for all the metrics considered at $\alpha = 0.05$ level with a statistical power of $1 - \beta = 0.8$.

7.2 Results and Discussion

The experiment showed that the differences between both strategies were significant for all the metrics considered. The primary metric, session success ratio, was increased by 5.4% ($p < 0.001$) which shows the ability of ltr-car to expose users to more relevant content, leading to a higher average number of successful sessions. This conclusion was further supported by the uplift in the lead to view ratio, which was increased by a 7.6% ($p < 0.001$), indicating a significant reduction in the effort taken by users to generate a lead. Finally, dwell time was increased by 2.0% ($p < 0.001$), indicating a higher engagement of users with the content browsed.

The results obtained in the online experiment are in consonance with the outcome of the user study. We can conclude that, for this problem setting characterized by a permanent cold-start scenario, short item lifespan, and abundance of near-duplicates, offline evaluation is not able to provide an assessment of relevance useful for effectively comparing different recommendation strategies. Furthermore, the proposed content-based approach is able to learn a similarity model for the cars domain using a limited set of metadata which outperforms collaborative filtering in this domain.

8 CONCLUSIONS AND FUTURE WORK

In this work, we studied the performance of different recommendation strategies in a highly volatile environment with vast amounts of near-duplicate items. These conditions limit the effectiveness of collaborative filtering methods. We proposed a learning to rank solution that successfully models item similarity at the product level using a purely content-based approach, but leveraging behavioural signals during training. We conducted an A/B test where the proposed method significantly outperformed our existing baseline, a hybrid algorithm combining CF and CB recommendation strategies, highlighting the limitations of methods relying on a the accumulation of user-item interactions in this domain. Finally, we provided additional empirical evidence of the findings of previous work in the recommenders area, where online and offline evaluations results led to opposite conclusions [14].

There are three main areas of this work that we plan to continue working on. First, we want to extend our comparative analysis to consider more sophisticated CF methods, such as [35], to further validate the results found during our study. We also want to improve our offline evaluation methodology to reduce the offline-online gap, both by considering alternative weighting schemes (e.g. inverse propensity scoring) as well as off-policy evaluation methods. Finally, we want to explore aspects of the recommendations beyond relevancy, including seller and buyer fairness as well as diversity.

ACKNOWLEDGMENTS

We want to thank all the current and former members of Adevinta's Personalization and UX teams that made this work possible.

REFERENCES

[1] M. Kula, "Metadata embeddings for user and item cold-start recommendations," in *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.* (T. Bogers and M. Koolen, eds.), vol. 1448 of *CEUR Workshop Proceedings*, pp. 14–21, CEUR-WS.org, 2015.

[2] J. Xu, Y. Yao, H. Tong, X. Tao, and J. Lu, "Rapare: A generic strategy for cold-start rating prediction problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 10 2016.

[3] P. Gupta, T. Dreossi, J. Bakus, Y.-H. Lin, and V. Salaka, "Treating cold start in product search by priors," *Companion Proceedings of the Web Conference 2020*, 2020.

[4] Z. Nazari, C. Charbuillet, J. Pages, M. Laurent, D. Charrier, B. Vecchione, and B. Carterette, "Recommending podcasts for cold-start users based on music listening and taste," *Proceedings of the 43rd International ACM SIGIR*, Jul 2020.

[5] H. Li, "A short introduction to learning to rank," *IEICE Transactions on Information and Systems*, vol. 94, no. 10, pp. 1854–1862, 2011.

[6] M. Volkovs, H. Rai, Z. Cheng, G. Wu, Y. Lu, and S. Sanner, "Two-stage model for automatic playlist continuation at scale," pp. 1–6, 10 2018.

[7] C. Lynch, K. Aryafar, and J. Attenberg, "Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 541–548, 2016.

[8] Y. M. Brovman, M. Jacob, N. Srinivasan, S. Neola, D. Galron, R. Snyder, and P. Wang, "Optimizing similar item recommendations in a semi-structured marketplace to maximize conversion," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 199–202, 2016.

[9] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, (New York, NY, USA), 2016.

[10] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*, pp. 257–297, Springer, 2011.

[11] Z. Zhu, Y. He, X. Zhao, and J. Caverlee, "Popularity bias in dynamic recommendation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2439–2449, 2021.

[12] J. Beel and S. Langer, "A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems," in *International conference on theory and practice of digital libraries*, pp. 153–168, Springer, 2015.

[13] Y. Yao and F. M. Harper, "Judging similarity: a user-centric study of related item recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 288–296, 2018.

[14] P. Kouki, I. Fountalis, N. Vasiloglou, X. Cui, E. Liberty, and K. Al Jadda, "From the lab to production: A case study of session-based recommendations in the home-improvement domain," in *Fourteenth ACM conference on recommender systems*, pp. 140–149, 2020.

[15] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, p. 331–370, nov 2002.

[16] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[17] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, (New York, NY, USA), p. 1809–1818, Association for Computing Machinery, 2015.

[18] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, pp. 333–389, 01 2009.

[19] A. Karatzoglou, L. Baltrunas, and Y. Shi, "Learning to rank for recommender systems," pp. 493–494, 10 2013.

[20] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," 2016.

[21] M. Volkovs, G. W. Yu, and T. Poutanen, "Content-based neighbor models for cold start in recommender systems," in *Proceedings of the Recommender Systems Challenge 2017*, pp. 1–6, 2017.

[22] D. Jannach, G. de Souza P. Moreira, and E. Oldridge, "Why are deep learning models not consistently winning recommender systems competitions yet? a position paper," in *Proceedings of the Recommender Systems Challenge 2020*, pp. 44–49, 2020.

[23] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," in *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

[24] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[25] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[26] C. Burges, R. Ragno, Q. Le, and C. J. Burges, "Learning to rank with non-smooth cost functions," in *Advances in Neural Information Processing Systems 19*, MIT Press, Cambridge, MA, January 2007.

[27] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017.

[28] P. Campos, F. Rubio, and I. Cantador, "Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols," *User Modeling and User-Adapted Interaction*, vol. 24, 02 2014.

[29] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *User Modeling and User-Adapted Interaction*, vol. 28, p. 331–390, dec 2018.

[30] O. J. Dunn, "Multiple comparisons using rank sums," *Technometrics*, vol. 6, no. 3, pp. 241–252, 1964.

[31] H.-H. Chen, C.-A. Chung, H.-C. Huang, and W. Tsui, "Common pitfalls in training and evaluating recommender systems," *SIGKDD Explor.*, vol. 19, pp. 37–45, 2017.

[32] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '18, (New York, NY, USA), p. 311–320, Association for Computing Machinery, 2018.

[33] Y. Kim, A. Hassan, R. W. White, and I. Zitouni, "Comparing client and server dwell time estimates for click-level satisfaction prediction," in *Proceedings of the 37th International ACM SIGIR*, SIGIR '14, (New York, NY, USA), p. 895–898, Association for Computing Machinery, 2014.

[34] R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing.* Cambridge University Press, 2020.

[35] H. Steck, "Embarrassingly shallow autoencoders for sparse data," in *The World Wide Web Conference on - WWW '19*, ACM Press, 2019.