Online and Offline Evaluations of Collaborative Filtering and Content Based Recommender Systems

Ali Elahi
aelahi6@uic.edu
University of Illinois at Chicago
Chicago, Illinois, USA

Armin Zirak
aarmin.zirak@ucalgary.ca
University of Calgary
Calgary, Canada

Abstract

Recommender systems are widely used AI applications designed to help users efficiently discover relevant items. The effectiveness of such systems is tied to the satisfaction of both users and providers. However, user satisfaction is complex and cannot be easily framed mathematically using information retrieval and accuracy metrics. While many studies evaluate accuracy through offline tests, a growing number of researchers argue that online evaluation methods such as A/B testing are better suited for this purpose.

We have employed a variety of algorithms on different types of datasets divergent in size and subject, producing recommendations in various platforms, including media streaming services, digital publishing websites, e-commerce systems, and news broadcasting networks. Notably, our target websites and datasets are in Persian (Farsi) language.

This study provides a comparative analysis of a large-scale recommender system that has been operating for the past year across about 70 websites in Iran, processing roughly 300 requests per second collectively. The system employs user-based and item-based recommendations using content-based, collaborative filtering, trend-based methods, and hybrid approaches. Through both offline and online evaluations, we aim to identify where these algorithms perform most efficiently and determine the best method for our specific needs, considering the dataset and system scale.

Our methods of evaluation include manual evaluation, offline tests including accuracy and ranking metrics like hit-rate@k and nDCG, and online tests consisting of click-through rate (CTR). Additionally we analyzed and proposed methods to address cold-start and popularity bias.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Information Retrieval, Recommender Systems, Evaluation of Recommender Systems

1 Introduction

Recommender Systems (RS) are one of the most common and significant services provided by information systems. Music and media streaming platforms, e-commerce, employment websites, and online news publishers all employ RS to display content more efficiently to their users [12, 13, 23]. RS filter streams of information to present the user with related or personalized content. This filtering process is either focused on item similarity or users' previous views, the first being item-based and the latter being user-based.

More traditional methods, use CB and CF algorithms analyzing the contextual similarity and feedback datasets. Recently, reinforcement and deep learning-based methods are being utilized in the state-of-the-studies. A gap, however, is apparent between the practical and the academic environments, which this paper aims to cover via the evaluation of the algorithms in the practical environment.

Parallel to "Recommendation with a Purpose" [14], our ultimate objective is to enhance the efficiency of methods in real-world environments. A key distinction between academic and practical settings lies in the evaluation approach. While offline tests, often used in academic research, focus on metrics like accuracy and diversity, online evaluations measure performance through metrics like click-through rate (CTR) and time spent on the platform (PPS). Rossetti et al. [24] have shown that offline metrics often fail to accurately predict algorithm performance in real-world scenarios. Another factor contributing to the gap between academic research and practical applications is the limited variety and scale of datasets used in academic studies. In practical environments, the attributes of recommended items and user behavior are dynamic across different contexts, leading to unpredictable feedback.

A/B testing is a common online testing method. In A/B testing, users are presented with different sets of recommendations, which can result in some groups receiving superior suggestions while others receive inferior ones. The two groups are either exposed to the same algorithm with varying hyper-parameters or two distinct algorithms, typically with one key variant distinguishing the two groups. Online evaluations must be conducted continuously, with performance comparisons assessed over time, as the effectiveness of the recommendation system (RS) is dynamic and can fluctuate due to factors like users' growing trust in the system or cold-start challenges. [5].

This study was conducted in a company that provides RS as a service. It has been providing over 30 million users within the past year; processing with a load of 300 requests per second and maintains a clientele of approximately 70 websites. The service delivers user-based and item-based recommendation systems engaging content-based (CB), collaborative filtering (CF), trend-based, and some hybrids.

We tested our RS on a range of platforms, including e-commerce sites, media streaming services, news broadcasting websites, and digital publishing networks. In our analysis, we accounted for the differences in type and scale of data across these websites, recognizing that various data sets can yield differing performance levels with different algorithms. We thoroughly explored multiple algorithms, made comparisons, and considered the contextual factors that influence their effectiveness.

Our offline analysis includes ranking and accuracy metrics such as hit-rate@k and, nDCG, mainly used for hyper-parameter tuning. CTR is used as our online metric.

2 Literature Review

2.1 Evaluation of RS

Traditionally, the primary metrics for evaluating recommendation systems focus on accuracy, such as Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), and Area Under the Curve (AUC). However, relying solely on accuracy can be insufficient and may even hinder the performance of the recommendation system [21]. Users tend to prefer recommendations that are exciting, surprising, novel, and diverse. Novelty refers to how unique an item appears to the user, while diversity is achieved by ensuring that the recommended items differ from one another [7].

Numerous studies on RS evaluation have established that offline tests, when used in isolation, fail to accurately reflect a system's efficiency [6, 11]. In response to this limitation, several researchers have explored ways to enhance offline testing methodologies. Some studies [26, 27] have integrated diversity and novelty metrics alongside traditional accuracy measurements to provide a more comprehensive assessment.

Studies propose that RS online evaluations, should be done frequently [5, 23]. The performances of a RS may change over time in an online environment. Users often need time to develop trust in the system through their interactions.

Jeunen et al. [15] proposed evaluating CF methods by using the most recent segment of the dataset rather than a random subset. This led to the introduction of SW-eval (sliding window evaluation), which is presented as a superior alternative to Leave-One-Out Cross-Validation (LOOCV), offering improved insights into the performance of CF techniques.

Another evaluation method is using questionnaires, offering insights into user satisfaction and preferences [24]. Other research has focused on correlating offline and online tests to predict CTR using offline metrics, demonstrating the relationship between these evaluation methods [18, 20, 22, 25]. Furthermore, the concept of viewing the RS as a black box has been proposed, facilitating a better understanding of model behavior and improving evaluation methods that aid in model selection [9]. These approaches collectively enhance the evaluation and implementation of RS.

Jannach et al. [14] has evaluated RS both users and providers perspective. Users seek to discover new items and alternatives within their areas of interest, while also wanting their needs to be met. On the other hand, providers aim to generate new demands among users, introduce new services, enhance user retention, and ultimately increase revenue. It is essential to have metrics in place to assess whether these needs are being satisfied. Jannach et al. [14] introduces various scenarios that users may encounter, which can be leveraged to improve the accuracy of our recommendations. A user might be casually browsing, looking for similar items, or specifically searching for popular products.

2.2 Challenges in Recommender Systems

The effectiveness of recommendation algorithms are context dependent, as they each come with their own set of benefits and drawbacks. CF algorithms often exhibit popularity biases, frequently recommending popular items over others. This can be influenced by the website's landing page, which may highlight certain items more prominently, leading to an unfairly higher number of views for those items. Implicit feedback datasets [8] are particularly affected by this popularity bias, as they are based solely on clicks and views, whereas explicit feedback datasets rely on user rankings and like/dislike buttons. Items categorized as "long tail," which can be more effective drivers of growth for providers, are recommended significantly less than a select few popular items [3, 17]. Metrics such as the Gini index and group average popularity are used to illustrate these complexities. Studies have proposed filtering methods or re-ranking methods [1, 2] to mitigate the effect of item exposure.

CF algorithms also tend to have lower fill rates and coverage. The fill rate refers to the proportion of users or items for which the RS can generate recommendations, while coverage indicates the percentage of items that the system is able to recommend. One factor contributing to the lower fill rate in CF is the cold start problem, which occurs when there is insufficient data about certain users or items. This issue is particularly prevalent on websites with high item churn, such as news sites that publish hundreds of articles daily. The paper titled "When Collaborative Filtering's Data Is Not Enough" [10] discusses this challenge in detail. To address the cold start problem, state-of-the-art studies like [4] often utilize hybrid methods.

3 Methodology

Our RS provides service to 70 websites. We track the pages each user views and use this data to build an implicit feedback matrix. Contents of the pages are also crawled for the CB algorithm. A variety of algorithms have been implemented, which will be introduced at length.

3.1 Algorithms Overview

Website-Trend. Trend-based recommendation is an item-to-item algorithm that suggests pages currently trending on a website.

Content-based. For content-based (CB) algorithms, a word2vec embedding is generated for each item using its title and the first L words of its text. The word2vec model, specifically trained in Farsi on a 50-gigabyte corpus with diverse contexts using the Gensim library, provides these embeddings. This mapping allows items to be represented as vectors, and a nearest neighbor algorithm is then applied to recommend similar pages based on their proximity in the embedding space.

Category-Trend. Using Latent Semantic Analysis (LSA), we created mappings for the pages. We then applied the DBSCAN clustering method to group pages, and used the Bayesian Information Criterion (BIC) to determine the optimal number of categories. For recommending items related to a specific item I in category C, popular items from the same category are randomly selected.

Collaborative-Filtering. CF algorithm utilize matrix factorization to generate item-to-item and user-to-item recommendations. We used the Alternating Least Squares (ALS) algorithm [16] implemented by the Implicit library. Section 6.3 will analyses the performance of this algorithm. In our CF method, we used implicit feedback matrix [8], populated by page view counts. This approach

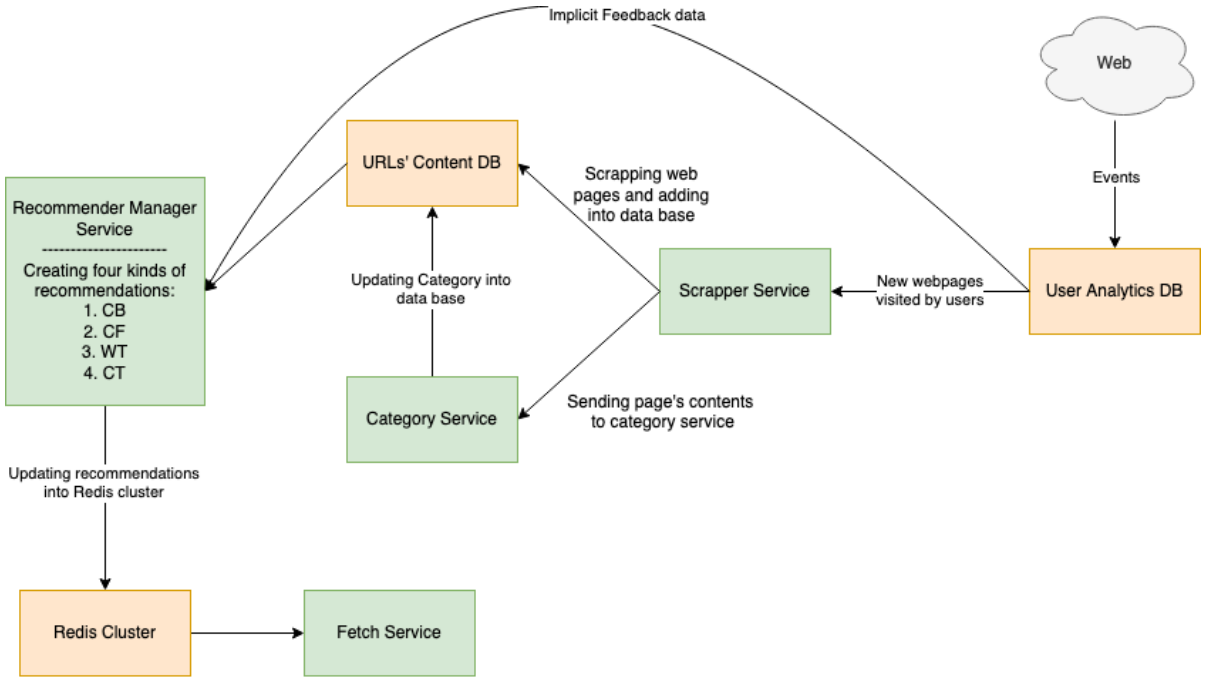


Figure 1: The RS architecture is shown above; The green boxes are the services and the orange boxes are data-bases.

generates embeddings for both items and users, with item-based recommendations determined by identifying each item's nearest neighbors in the embedding space. To approximate the implicit feedback matrix, the user and item matrices are multiplied, enabling user-based recommendations by suggesting unseen items with the highest predicted values.

3.2 System Architecture

Figure 1 illustrates the architecture of our RS. The User Analytics database stores user browsing data and creates the implicit feedback dataset, while the Scraper service collects item URLs viewed by users that haven't yet been saved, populating the URL-content database. The Recommender Manager service applies various algorithms using this data to generate item- and user-based recommendations, loading the results into the REDIS dataset. As a page loads, the Fetch service retrieves appropriate recommendations from REDIS. For new pages in a domain, the Recommender Manager periodically updates REDIS, with the frequency depending on the site's item turnover. High-churn websites, like news sites, receive updates more frequently. Additionally the Category service updates the URL categories accordingly every time the recommendations are being made. Every time a user log into one of the websites using our RS, the website will fetch the recommendations either for an item or for a user.

3.3 RS Components

Approximate Nearest Neighbor. We employed Spotify's Annoy library [19] for approximate nearest neighbor (ANN) searches, which proved to be significantly more time-efficient compared to exact nearest neighbor algorithms, such as those provided by the SK-Learn library. A detailed comparison of accuracy and processing time between the exact and approximate models is presented in the Offline Evaluations section under Experiments. In Section 6.2 we will compare the performance of the nearest neighbor and approximate nearest neighbor.

Post/Pre-Filtering. To enhance recommendation quality, web pages underwent pre- and post-filtering. In the content-based (CB) algorithm, a percentage of low-view pages were excluded to avoid displaying outdated content. After identifying nearest neighbors, pages exceeding a certain cosine similarity threshold were also removed to prevent recommending overly similar content, such as duplicate news coverage from different reporters. For the collaborative filtering (CF) algorithm, users with fewer than two views and pages with fewer than five views were omitted from the matrix to increase density. This adjustment helps reduce inadequate recommendations and addresses the cold-start problem.

Cold Start Fall-back Strategy. In our algorithms, to mitigate low fill-rate or coverage issues when data on a user or item is insufficient, we employ a self-organized fallback strategy that provides generalized recommendations. If CF recommendations are unavailable, the RS defaults to CB, category-trend, or website-trend suggestions. Additionally, for websites with higher item churn, we run our algorithms more frequently to maintain recommendation relevance.

4 Dataset

The data for this study was collected from various websites utilizing our RS. Approximately 70 websites currently implement Yektanet's RS, and we aimed to include a diverse range of platform types and

scales in our research. Table 1 presents the number of users and items across each of these websites.

Website	# Items	# Users	Type of Website
Tabnakjavan.com	16.4K	8.388M	News
Tapmusics.ir	16.38K	1.048M	Movie Criticize
Paroshat.com	8.19K	4.194M	Movie Criticize
Entekhab.ir	32.67K	2.097M	News
Faradeed.ir	14.2K	1.4M	News
Chetor.com	8.19K	2.1M	Publishing
Academicfiles.ir	624	16.38K	E-commerce

Table 1: Numbers of users and items for each website used in our analysis.

5 Manual Analysis

To interpret the functionality of different sections within the algorithm, we developed a manual analysis approach. A team of recommender system experts, product managers, and domain specialists collaborated to assess the quality of recommendations produced by various algorithms configured with distinct hyper-parameters. Certain hyper-parameters, particularly those affecting semantic similarity, lacked suitable offline tuning metrics and couldn't be effectively optimized within an online evaluation environment, prompting us to adopt this testing method. Key questions that arose and were addressed through this testing process include:

- How far along the text will the extracted string for our word2vec embedding follow?
- How many clusters are suitable for the category trend algorithm?
- Is our self-trained word2vec algorithm a good fit, or are we better off retorting to the Persian version of pretrained models such as Farsi-BERT or fastText, trained respectively by Google and Facebook?
- Between 100 and 300, both of which we tried, which one is a better size for our word2vec embedding?

6 Offline Evaluations

In our offline tests, we tuned the algorithm hyper-parameters to optimize performance. Specifically, we adjusted parameters in the Annoy library and the Implicit library, used for deploying the approximate nearest neighbor and ALS algorithms, respectively. In the final stage, we examined the impact of popularity bias across different algorithms and refined our hyper-parameters to minimize this effect as much as possible.

6.1 Metric

nDCG. Normalized Discounted Cumulative Gain (nDCG) is a metric used to evaluate the effectiveness of a ranking system. It considers the relevance of the ranked items and their positions in the result set. The nDCG is computed in two steps: first, the Discounted Cumulative Gain (DCG) is calculated using the relevance scores of the items, and then it is normalized by the Ideal Discounted Cumulative Gain (IDCG), which represents the maximum possible DCG for a perfect ranking. The formulation for DCG at rank p is given by:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)}$$

where rel_i is the relevance score of the item at position i. The nDCG is then computed as:

$$\mathrm{nDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}$$

where IDCG_p is the DCG for the ideal ranking of the items.

Hit Rate (HR@k). HR@k is a commonly used metric for evaluating the performance of recommendation algorithms. It measures the proportion of times that a user's true preference, or "hit," appears in the top k recommendations generated by the model. Formally, for a user u, a hit occurs if any of the items they have interacted with is present in the top k items recommended. Let R_u denote the set of items recommended to user u and T_u be the set of items that user u actually interacted with (e.g., viewed, clicked, or purchased). The Hit Rate@k for user u is defined as:

$$\mathrm{HR@k}_u = \begin{cases} 1 & \text{if } R_u \cap T_u \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

The overall Hit Rate@k is then calculated as the average hit rate across all users, which can be expressed as:

$$\mathrm{HR@k} = \frac{1}{|U|} \sum_{u \in U} \mathrm{HR@k}_u,$$

where $|U|$ is the total number of users. This metric reflects the effectiveness of the model in providing relevant items within a limited recommendation list. A higher HR@k score indicates that the model is more successful in delivering items of interest to users within the top k recommended items.

6.2 Annoy library

The Annoy library is a C++ library used to search for points in space that are close to a given query point, making it ideal for identifying similar items within our large embedding spaces. These embeddings may be generated through collaborative filtering (CF) or word2vec. We evaluated the Annoy-based approximate nearest neighbor (ANN) algorithm by examining both its time efficiency and accuracy. Accuracy refers to the correlation between items identified by ANN as similar to a primary item and the exact nearest neighbors, using the nDCG ranking measure to assess relevance. Given the large scale of our dataset, time efficiency, or query time, is crucial. We measure throughput as the number of recommendations generated per second within the embedding space.

The Annoy library features two largely independent hyper-parameters: n_trees and search_k. We utilized the default setting for search_k while tuning the n_trees hyper-parameter based on nDCG and time-throughput metrics. As shown in Figure 2, the nDCG values for most websites do not exceed 0.5, indicating a satisfactory level of similarity, as excessive similarity is not ideal. After evaluating the trade-off between time efficiency and accuracy, we settled on an n_trees value of 50 for our algorithm.

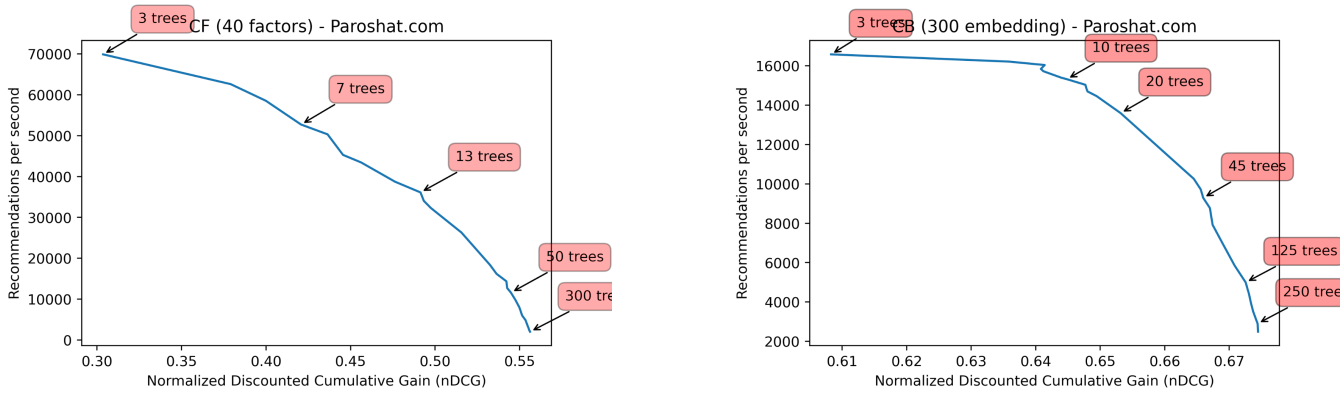


Figure 2: nDCG with respect to recommendation generation frequency by changing the number of trees of the Annoy library.

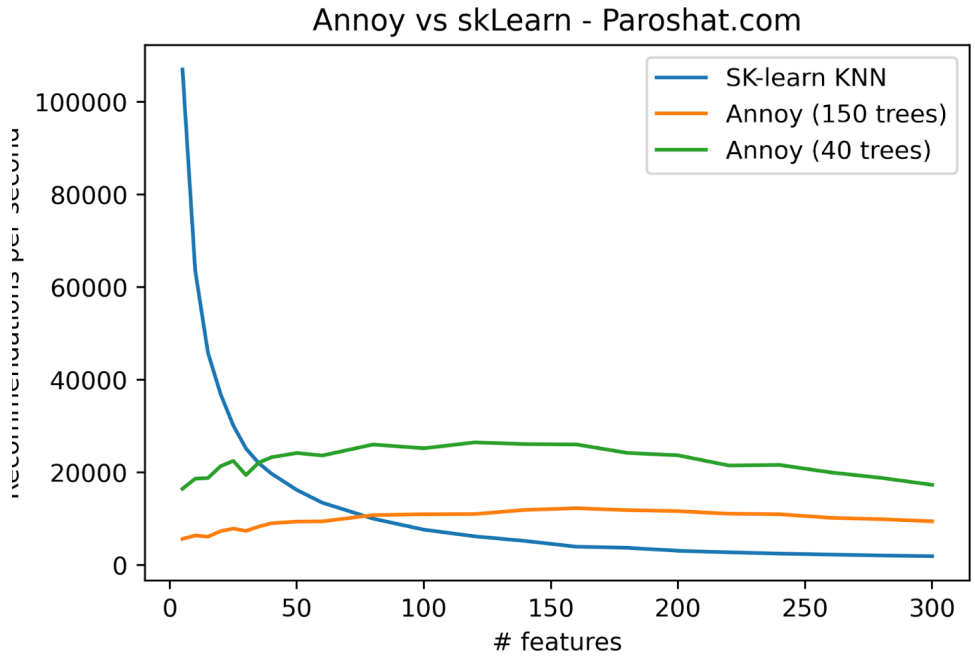
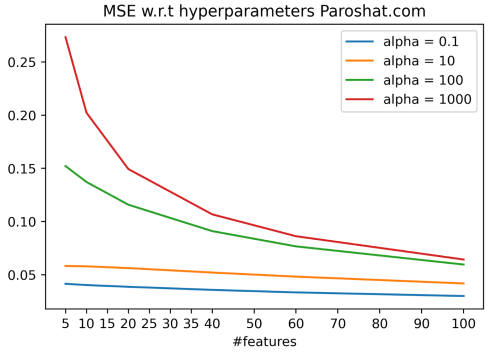


Figure 3: Annoy vs SK-Learn on Paroshat.com: The y-axis is the frequency (recommendations generated per second) and the x axis is the number of features in CF algorithm.

Figure 3 illustrates the impact of embedding feature count (embedding size) on the time performance of nearest neighbor algorithms. A comparison was made between the nearest neighbor algorithms from the SK-Learn library and the Annoy library, using n_trees values of 50 and 150. The diagram shows that the time performance of the Annoy library improves positively once the feature count exceeds a certain threshold, which is influenced by the number of items in the domain. Based on our findings, we determined that the SK-Learn library is more suitable for websites with fewer items and users, while the Annoy library is better suited for the majority of our publishers, which tend to have larger datasets.

6.3 Alternating Least Square

The Implicit library has been engaged to deploy the CF algorithm, using Alternative Least Square (ALS) method, which itself adopts the stochastic gradient descent to create user and item matrices. The two main hyper-parameters being alpha and embedding size have been tuned.



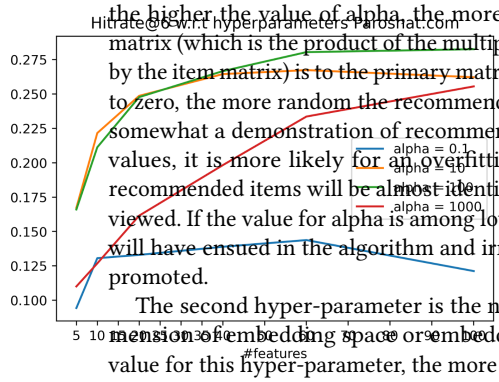
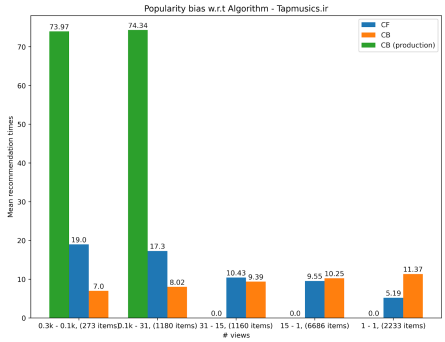
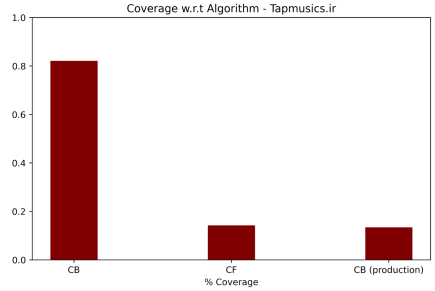


Figure 4: Metrics wrt hyper-parameters for ALS Alg.





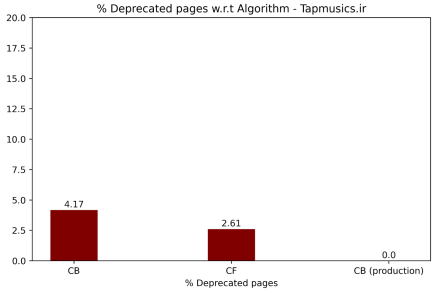


Figure 5: Popularity Bias

Alpha is the strictness of the simulation for the primary matrix; the higher the value of alpha, the more similar the approximate matrix (which is the product of the multiplication of the user matrix by the item matrix) is to the primary matrix. The closer alpha inches to zero, the more random the recommendations would be. Alpha is somewhat a demonstration of recommendation novelty; in higher values, it is more likely for an overfitting issue to arise as most recommended items will be almost identical to the items previously viewed. If the value for alpha is among lower numbers, randomness will have ensued in the algorithm and irrelevant items will also be promoted.

The second hyper-parameter is the number of features (the dimension of embedding space or embedding size). The higher the value for this hyper-parameter, the more the approximate matrix is similar to the primary matrix. In order to improve matrix density, less active users and non-recent items with less views are eliminated in the first stage. This step will be crucial to mitigate popularity bias. Afterwards, ALS is trained on the matrix.

HR@k criteria were employed for hyper-parameter tuning on train and test data. 20% of viewed pages were eliminated from the matrix to be used as test data; an attempt was made for the eliminated pages to be among the ones viewed not long ago by users. As observable in Figure 4, hit rate increases both in testing and training with the increase of alpha and number of features until they reach roughly static conditions.

We finally gathered that the value for alpha should be situated somewhere between 60-100 and feature count should be somewhere between 20-40. Outside of these latter limits, the query time computation for Annoy library would take too long and time throughput will decrease, resulting in less authenticity for the algorithm. MSE loss and hit rate for each hyper-parameter appears in Figure 4.

Popularity bias. The CF algorithm is known to be associated with popularity bias problem. As shown in Figure 5, the CF algorithm recommends popular items more frequently than non-popular items, while CB algorithms do not segregate non-popular items. A differentiation between items based on popularity is not necessarily undesirable; at very least, it aids with decreasing the prevalence of deprecated item previews. It is recognizable that a filtered CB algorithm does not show deprecated items (items with very few views over time).

We have divided items into groups based on popularity and calculated the average times an item is recommended with each of the three algorithms (CB algorithm, pre-filtered CB algorithm and CF algorithm). This has been done with two different alpha values in the CF algorithm in an attempt to minimize popularity bias. Figure 5, alternatively, displays deprecated page recommendations, suggesting that CF algorithms are generally not as prone to displaying deprecated items. A CB algorithm employed on a set of items in which deprecated pages are filtered beforehand, is likewise capable of the same thing.

7 Online

We examined algorithm performance through online metrics such as CTR. These tests were organized in October 2021. Figure 7 shows the CTR of the system over a week; The average CTR for the week is reported 6%.

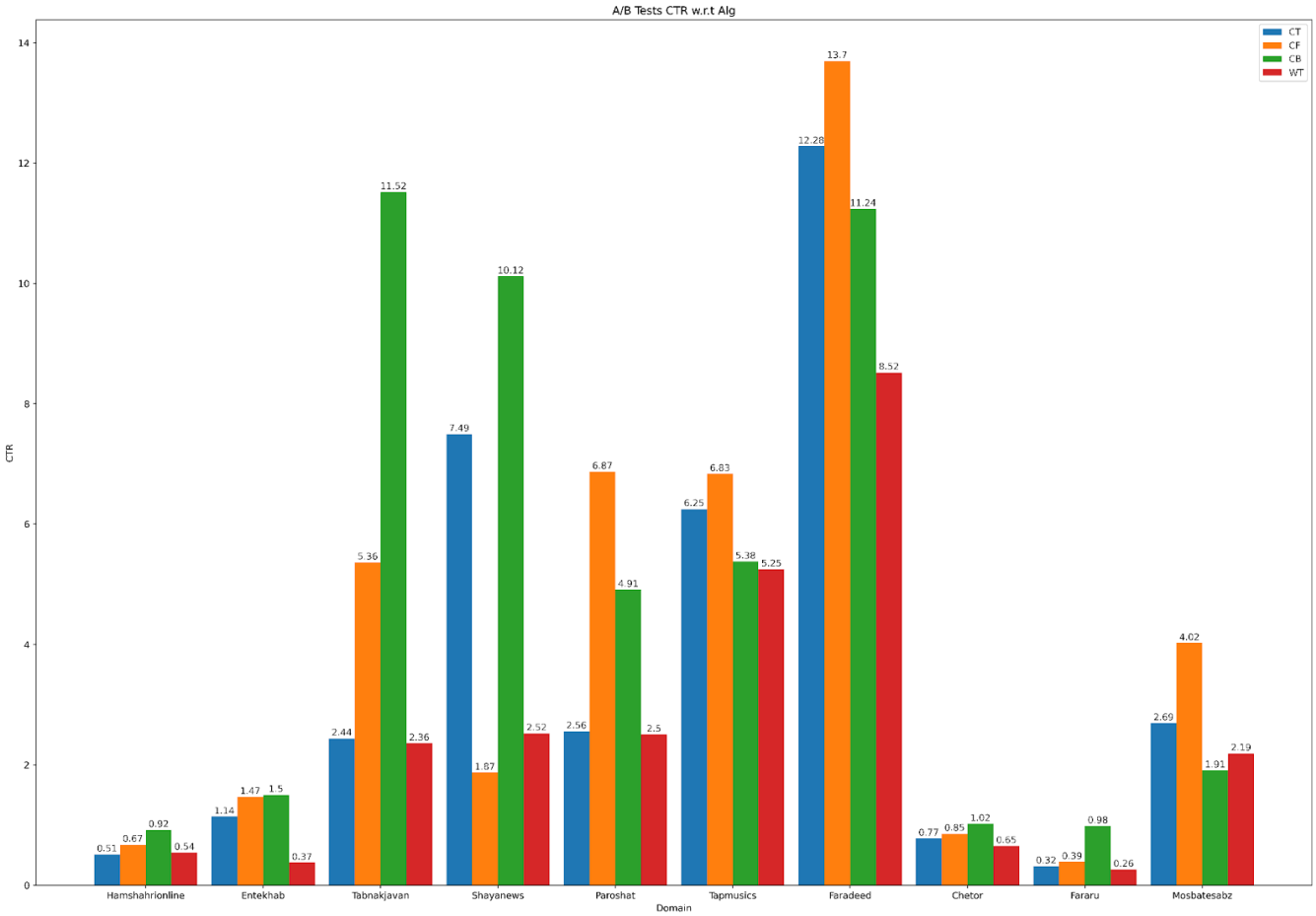


Figure 6: CTR Metric per Algorithms

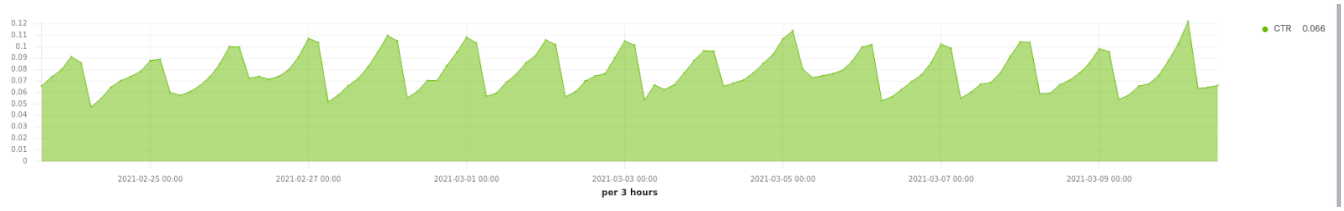


Figure 7: System Requests in Two Weeks

Recommendation boxes appear below each item and contain four to six suggestions generated by algorithms like CB, CF, website-trend, category-trend, and user-based models. Our fetch and click services log events whenever a recommendation box is displayed or a suggestion is clicked. These logs are stored in an Elasticsearch database using Fluent Bit, and we use the Elastic Kibana interface to visualize the results.

In Figure 6, the box plots show CTR for each algorithm, highlighting that hybrid algorithms achieve the highest overall performance. Additionally, CB algorithms perform particularly well on news websites, while CF algorithms are optimal for music and movie platforms.

The CB algorithm successfully generates recommendations for nearly 100% of pages, while the CF algorithm provides recommendations for only about half of the pages. This difference results from pre-filtering applied before sending pages and users to the CF algorithm, which excludes users with fewer than two views and pages with fewer than five views, creating a denser matrix. The second part of the diagram illustrates fill rates for users, representing the percentage of users for whom the user-based algorithm was able to generate recommendations.

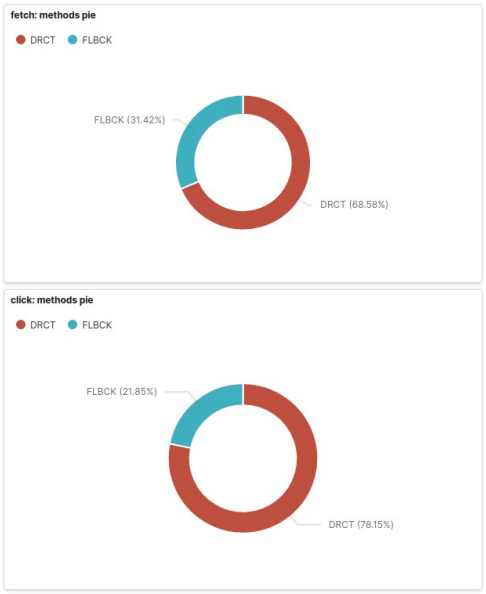


Figure 8: Fall-back Statistics

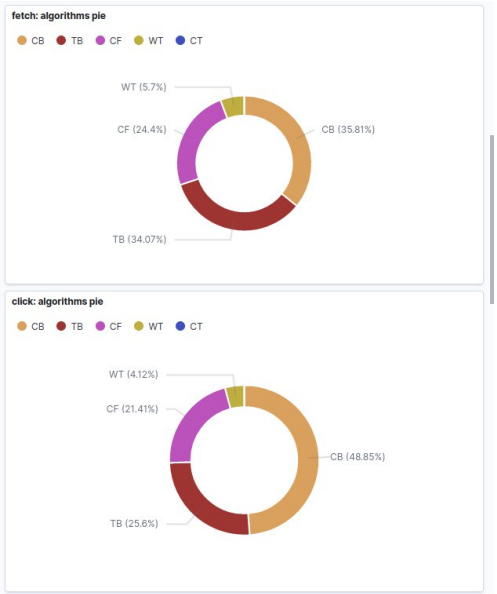


Figure 9: Fetch and Click Ratios for Each Algorithms

Figure 8 illustrates the system's fallback times. As described, a fallback strategy is used when there are insufficient recommendations for an item or user, prompting the system to switch to more general methods—for example, showing content-based (CB) recommendations instead of collaborative filtering (CF) when CF recommendations are unavailable. The fetch service is invoked when a user navigates to a page, and that page requests recommendations for either the user or the item. Similarly, the click service is activated when a recommendation is clicked, redirecting the user to the targeted page. Approximately 31.4% of fetch requests end up using this fallback strategy, with 21.85% of these resulting in a click.

Figure 9 displays the percentage of recommendations served by each algorithm. For instance, 31.81% of the recommendations shown to users were generated by the CB algorithm, while 48.85% of the clicks were on CB-generated recommendations.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 42–46. https://doi.org/10.1145/3109859.3109912

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 42–46. https://doi.org/10.1145/3109859.3109912

[3] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, Bamshad Mobasher, and Edward Malthouse. 2021. User-centered Evaluation of Popularity Bias in Recommender Systems. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization* (Utrecht, Netherlands) *(UMAP '21)*. Association for Computing Machinery, New York, NY, USA, 119–129. https://doi.org/10.1145/3450613.3456821

[4] Oren Barkan, Noam Koenigstein, Eylon Yogev, and Ori Katz. 2019. CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 228–236. https://doi.org/10.1145/3298689.3347038

[5] Jöran Beel. 2017. It's Time to Consider "Time" when Evaluating Recommender-System Algorithms [Proposal]. *ArXiv* abs/1708.08447 (2017). https://api.semanticscholar.org/CorpusID:32545283

[6] J. Beel and S. Langer. 2017. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. *Research and Advanced Technology for Digital Libraries* (2017), 153––168.

[7] P. Castells, S. Vargas, and J. Wang. 2011. Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. https://repositorio.uam.es/handle/10486/666094 (Accessed: January 16, 2022).

[8] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2637–2643. https://doi.org/10.1145/3308558.3313493

[9] Joey De Pauw. 2020. Exploratory Methods for Evaluating Recommender Systems. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 782–786. https://doi.org/10.1145/3383313.3411456

[10] E. Frolov and I. Oseledets. 2019. HybridSVD: When collaborative information is not enough. In *Proceedings of the 13th ACM Conference on Recommender Systems. New York, NY, USA: ACM.*

[11] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the 8th ACM Conference on Recommender Systems* (Foster City, Silicon Valley, California, USA) *(RecSys '14)*. Association for Computing Machinery, New York, NY, USA, 169–176. https://doi.org/10.1145/2645710.2645745

[12] Francisco Gutiérrez, Sven Charleer, Robin De Croon, Nyi Nyi Htun, Gerd Goetschalckx, and Katrien Verbert. 2019. Explaining and exploring job recommendations: a user-driven approach for interacting with knowledge-based job recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 60–68. https://doi.org/10.1145/3298689.3347001

[13] T. Huang, Z. Zhang, and J. Zhang. 2019. FiBiNET: Combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems. New York, NY, USA: ACM.*

[14] D. Jannach and G. Adomavicius. 2016. Recommendations with a purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16. New York, New York, USA: ACM Press.*

[15] Olivier Jeunen, Koen Verstrepen, and Bart Goethals. 2018. Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[17] Dominik Kowald, Markus Schedl, and Elisabeth Lex. 2020. The Unfairness of Popularity Bias in Music Recommendation: A Reproducibility Study. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II* (Lisbon, Portugal). Springer-Verlag, Berlin, Heidelberg, 35–42. https://doi.org/10.1007/978-3-030-45442-5_5

[18] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. 2019. Addressing delayed feedback for continuous training with neural networks in CTR prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 187–195. https://doi.org/10.1145/3298689.3347002

[19] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[20] A. Maksai, F. Garcin, and B. Faltings. 2015. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems. New York, NY, USA: ACM.*

[21] S. M. McNee, J. Riedl, and J. A. Konstan. 2006. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 extended abstracts on Human factors in computing systems - CHI EA '06. New York, New York, USA: ACM Press.*

[22] L. Peska and P. Vojtas. 2020. Off-line vs. On-line evaluation of recommender systems in small E-commerce. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media. New York, NY, USA: ACM.*

[23] B. Ray, A. Garain, and R. Sarkar. 2021. An ensemble-based hotel recommender system using sentiment analysis and aspect categorization of hotel reviews. *Applied soft computing* 98 (2021), 106935. https://doi.org/10.1016/j.asoc.2020.106935

[24] M. Rossetti, F. Stella, and M. Zanker. 2016. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16. New York, New York, USA: ACM Press.*

[25] H. Tahmasebi, R. Ravanmehr, and R. Mohamadrezaei. 2021. Social movie recommender system based on deep autoencoder network using Twitter data. *Neural computing & applications* 33, 5 (2021), 1607—-1623. https://doi.org/10.1007/s00521-020-05085-1

[26] Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 109–116. https://doi.org/10.1145/2043932.2043955

[27] Sanne Vrijenhoek, Mesut Kaya, Nadia Metoui, Judith Möller, Daan Odijk, and Natali Helberger. 2021. Recommenders with a Mission: Assessing Diversity in News Recommendations. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval* (Canberra ACT, Australia) *(CHIIR '21)*. Association for Computing Machinery, New York, NY, USA, 173–183. https://doi.org/10.1145/3406522.3446019

8 Acknowledgments

This project was conducted in Iran, at Yektanet Company. All data was gathered with user consent, and website content was crawled following notification of the site owners.