A Comparative Study on Recommendation Algorithms: Online and Offline Evaluations on a Large-scale Recommender System

Ali Elahi
Department of Computer Science
University of Illinois at Chicago
Chicago, Illinois
aelahi6@uic.edu

Armin Zirak
Schulich School of Engineering
University of Calgary
Calgary, Canada
armin.zirak@ucalgary.ca

***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 the Persian (Farsi) language, allowing us to focus on recommendations tailored to this specific linguistic and cultural context.

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 analysed and proposed methods to address cold-start and popularity bias.

***Index Terms*—Information Retrieval, Recommender Systems**

I. 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, 21, 13] RS filter streams of information to present the user with related or personalized content. This filtering process is either focused on item similarity or user profiles, the first being item-based and the latter being user-based.

More traditional methods, use CB and CF algorithms analysing 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 simplify 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. [22] 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 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, Additionally, the performance comparisons should be assessed over time, as the effectiveness of RS is not static and can change.[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 subsets 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 hyperparameter tuning. Our online metrics consist of CTR.

II. Litrature Review

A. 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.[19] 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.[11, 6] In response to this limitation, several researchers have explored ways to enhance offline testing methodologies. Some studies [24, 25] 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[21, 5]. 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 [22]. Other research has focused on correlating offline and online tests to predict CTR using offline metrics, demonstrating the relationship between these evaluation methods [20, 23, 17, 18]. Furthermore, the concept of viewing the recommender system 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 in our RS. 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.

B. Problems of RS

The use of each algorithm is 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, 16]. 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 recommender 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.

III. 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 all pages are also crawled for the CB algorithm. A variety of algorithms have been implemented, which will be introduced at length.

a) Trend-based: Trend-based algorithms include website-trend and category-trend, which generate item-to-item recommendations. Website-trend and category-trend recommendations are items with the most views on the website and most popular items within the original item's category, respectively. The category for the item is defined using its textual content. The website-trend recommendation is randomly chosen from a list of popular pages, determined by each pages' number of views.

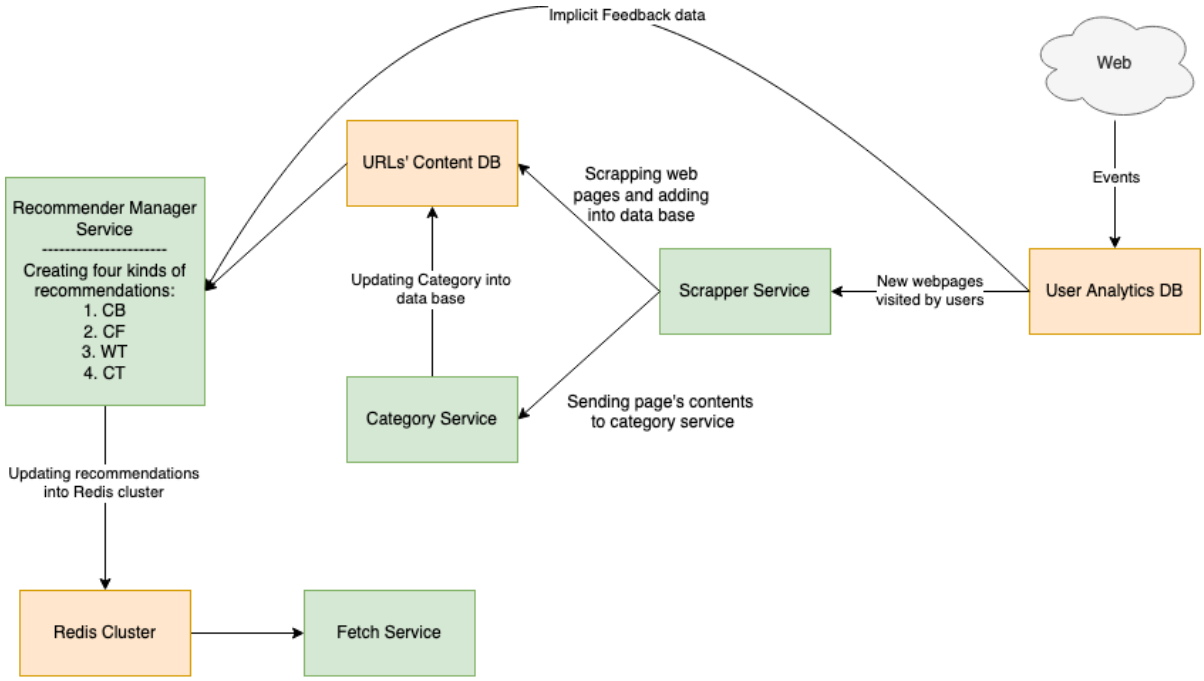

Fig. 1: architectural view of Yektanet's recommendation system

website	# items	# users	kind
Tabnakjavan.com	16.4K	8.388M	News Website
Tapmusics.ir	16.38K	1.048M	Movie Criticize Website
Paroshat.com	8.19K	4.194M	Movie Criticize Website
Entekhab.ir	32.67K	2.097M	News Website
Faradeed.ir	14.2K	1.4M	News Website
Chetor.com	8.19K	2.1M	Online Publishment
Academicfiles.ir	624	16.38K	Online Shop: Academic Books

TABLE I: Numbers of Users and Items for each website

For the CB algorithms, a word2vec mapping is calculated for each item using its title and the first L-words of the text. The word2vec model in Farsi was trained on a 50-gigabyte corpus in various contexts, using the Gensim library. In the Experiments segment of Manual Analysis, the function of this model was assessed; The model witnessed better functionality in comparison to ParsBERT model in these analyses and displayed improved mobility for large-scale projects. The MIRn mapping is thus created. The nearest neighbor algorithm was finally engaged to suggest similar pages. Engaging the Latent Semantic Analysis (LSA) and word2vec methods, tf-idf vectors are computed founded on item contents' unigrams and bigrams, embeddings were generated and applying clustering methods such as K-means and DBSCAN, the MIC mapping was formed. For the purpose of forming recommendations for item I in category C, the most popular items from the same category are randomly selected. CF algorithms utilize matrix factorization to generate both item-to-item and item-to-user (user-based) recommendations. The ALS algorithm implemented by the Implicit library employed the implicit feedback matrix [8], loaded with numbers of page views. Ultimately, the MIRmand MURm mappings were devised. The nearest neighbor for each item in this embedding are presented as item-based recommendations. The user and item matrices are multiplied to find the approximated implicit feedback matrix. User-based recommendations are then generated, displaying unseen items with the highest approximated values. The approximate nearest neighbor implemented by Spotify in the Annoy library was put to work, proven to be a more time-efficient option for us as compared with exact nearest neighbor algorithms, such as those implemented by the SK-Learn library. A comparison evaluating accuracy and time between exact and approximate models has been made in the Offline Evaluations section, under Experiments. Web pages have been post-filtered and pre-filtered to provide higher-quality recommendations. A percentage of pages with less views have been omitted from the CB algorithm's recommendations to avoid displaying deprecated pages. After finding our nearest neighbors, pages exceeding a threshold of cosine similarity have also been eliminated to avoid excess similarity. In the CF algorithm, users with less than two views and pages with less than five are taken out of the matrix to make it more

dense.This deletion helps avoid the production of inadequate recommendations for the user, contributing to the cold-start issue.

Figure 3 demonstrates an architectural view of Yektanet's recommendation system. A database called User Analytics holds data such as which items each user has previously browsed. The Scrapper service crawls items viewed by users that have not yet been saved in Yektanet's database and places them in the URL-content database. The Category service then proceeds to update URL categories. The Recommender Manager service conclusively implements different algorithms using available data and administers item-based and user-based recommendations for each item and user, respectively, and then loads them onto the REDIS dataset. The Fetch service receives the page-appropriate recommendations from REDIS as the page is about to load. Recommendations for domains' new pages are delivered intermittently by the recommender manager service, updating the REDIS dataset. The frequency for these repetitions vary, depending on how often the website comes up with newer pages. Websites with higher item churns or turnovers, such as news websites, get more frequent recommendation deliveries.

In our algorithms, to avoid low fill-rate/coverage when we lack sufficient data on a user or item, we implement a self-organized fallback strategy that provides less personalized, more generalized recommendations. In the absence of CF recommendations, the recommender system may generate CB, category-trend, or website-trend suggestions. Additionally, we run our algorithms more frequently for websites experiencing higher item churn.

IV. Experiments

A. Dataset

The data for this study has been collected from websites using our RS. approximately 100 websites use Yektanet's RS and we tried to include different types of platforms with different scales in our research. The Table 1, demonstrates numbers of users and items in each of these websites.

B. Manual analysis

We devised a type of manual analysis for interpreting the functionality of different sections of the algorithm. A number of recommender system experts, product managers and domain experts have rounded up to grade recommendations from different algorithms with disparate hyperparameters. Some hyperparameters, namely the ones affecting semantic similarity, are not represented with a convenient offline tuning metric, neither could they be tuned suitably in an online evaluation environment, and so we decided on this particular type of testing. Significant questions that came up and were resolved through these tests 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?

We tuned the algorithm hyperparameters in our offline tests. The performances of annoy library and implicit, which are respectively used for deploying the approximated nearest neighbour algorithm and the ALS algorithm, were optimized according to our needs. The final stage of our offline test scrutinizes the popularity bias complication and measures it in different algorithms and finally tuning our hyperparameters so that this issue is minimized.

C. Offline

We tuned the algorithm hyperparameters in our offline tests. The performances of annoy library and implicit, which are respectively used for deploying the approximated nearest neighbour algorithm and the ALS algorithm, were optimized according to our needs. The final stage of our offline test scrutinizes the popularity bias complication and measures it in different algorithms and finally tuning our hyperparameters so that this issue is minimized.

a) Annoy library: The Annoy library is a C++ library with python bindings developed by Spotify, which searches for points in space which are close to a given query point. We use this library to find similar items in our own embedding spaces. Our embeddings can be CF embeddings or word2vec embeddings. The approximated nearest neighbour's performance was analyzed time-wise and accuracy-wise. Accuracy is how the items which were found to be similar to the primary item (by ANN) correlates with the exact nearest neighbours, enlisting the nDCG measure which is a ranking method. Due to the large-scale disposition, time is critical. We consider time (query time) to be the throughput, which is the number of recommendations per second produced by the algorithm in the embedding space. The annoy library comes with two roughly independent hyperparameters, n_trees and search_k. We resorted to the default version of search_k and tuned the n_trees hyperparameter in accordance with nDCG and time-throughput measures. For the majority of websites, Figure 3 indicates that the nDCG value reaches no higher than 0.5 and this value somewhat sustains satisfaction as excess similarity is undesired. In the trade-off between time and accuracy, 50 was chosen for the n_trees hyperparameter of our algorithm. In Figure 5, how the number of embedding features (embedding size) affects the time performances of the nearest neighbour algorithms has been contemplated. The nearest neighbor algorithms from the scikit-learn library and the annoy library were compared, with different n_trees of 50 and 150. The diagram presents that the annoy library's time performance is enhanced positively once feature count reaches a certain number, which itself depends on the number of items in the domain. We established that the scikit-learn library should be used for websites with lower numbers of

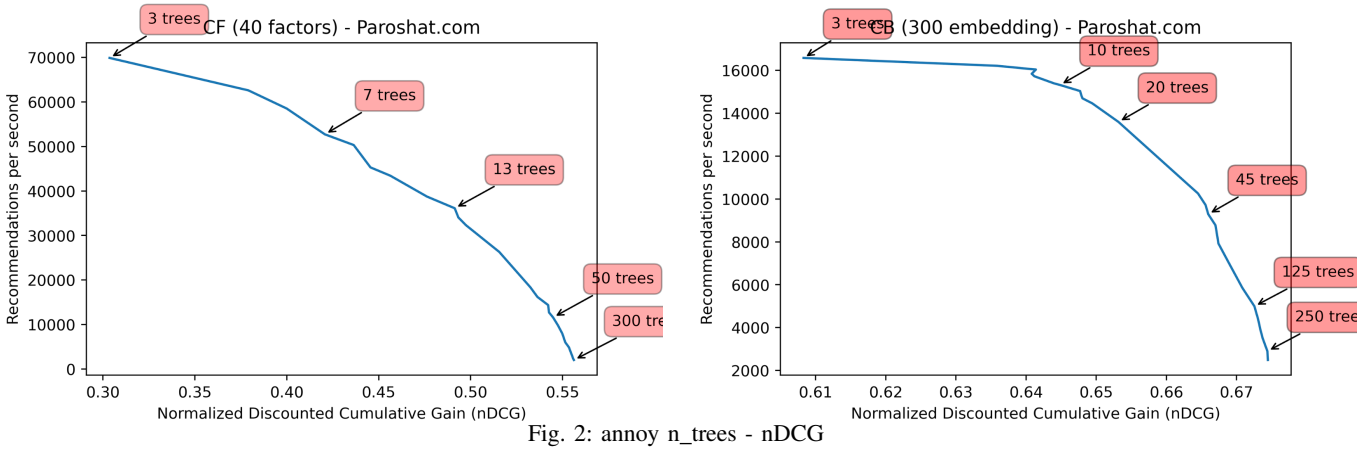

Fig. 2: annoy n_trees - nDCG

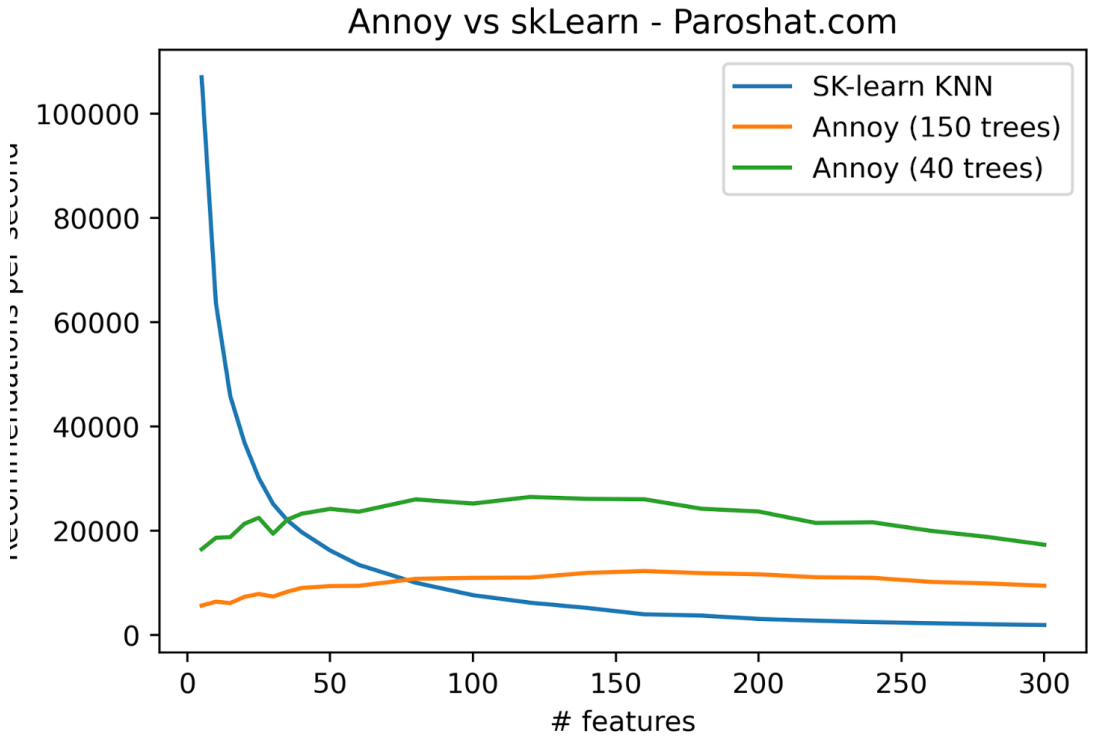

Fig. 3: annoy vs sklearn on paroshat

items and users, while the annoy library is arranged for other websites which make up the majority of our publishers.

b) ALS: The Implicit library has been engaged to deploy the CF algorithm, using ALS (alternative least square) method, which itself adopts the stochastic gradient descent to create user and item matrices. The two main hyperparameters being alpha and embedding size have been tuned. 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 hyperparameter is the number of features (the dimension of embedding space or embedding size). The higher the value for this hyperparameter, 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, as we are aware

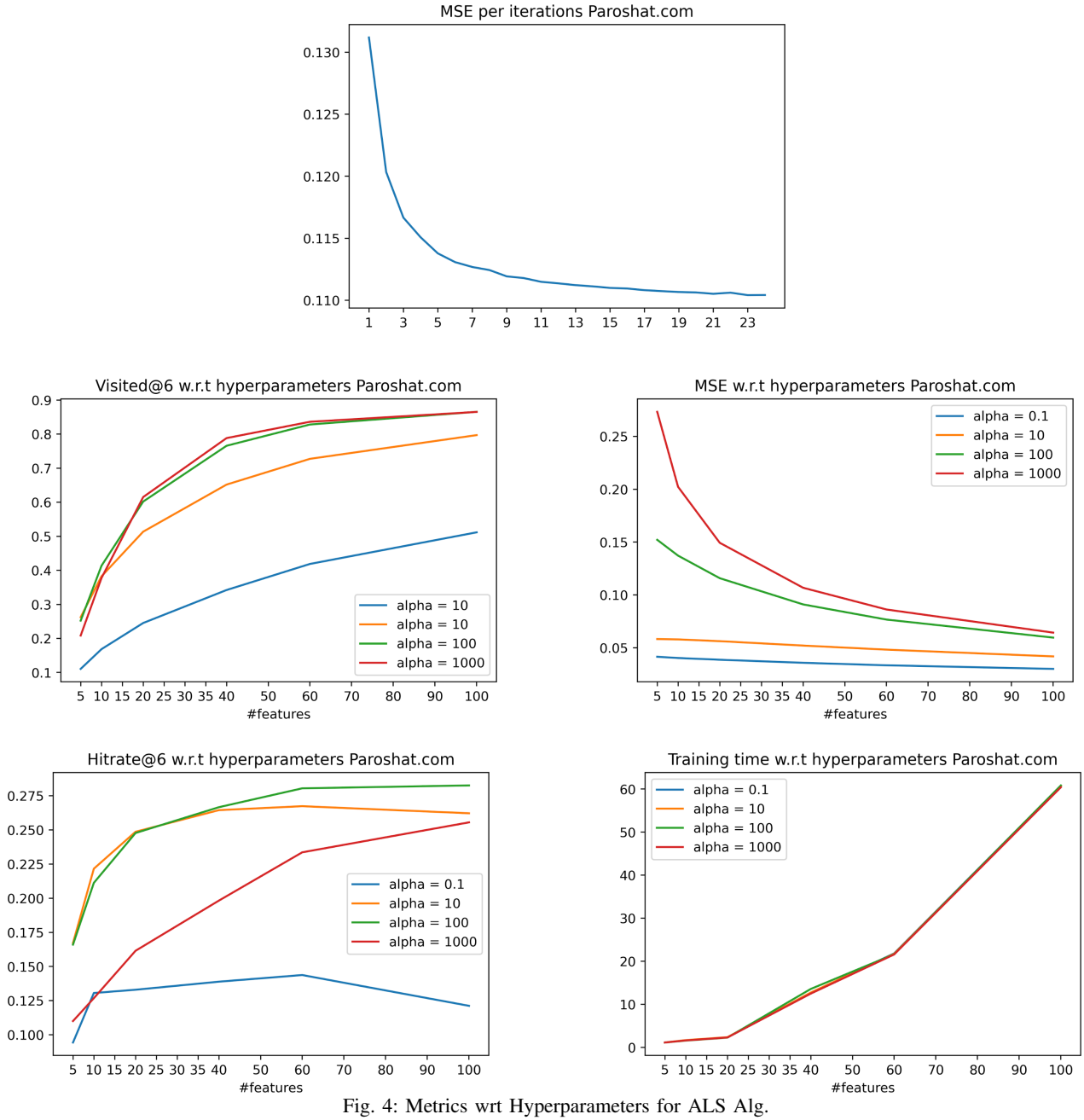

Fig. 4: Metrics wrt Hyperparameters for ALS Alg.

that if matrix density is not improved upon, CF algorithms will be facing issues; for instance, only spawning popular recommendations for users. Afterwards, ALS is trained on the matrix. HR@k and Visited@k criteria were employed for hyperparameter 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. /HR and Visited definitions/ 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 hyperparameter appears in Figure 4.

c) 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.

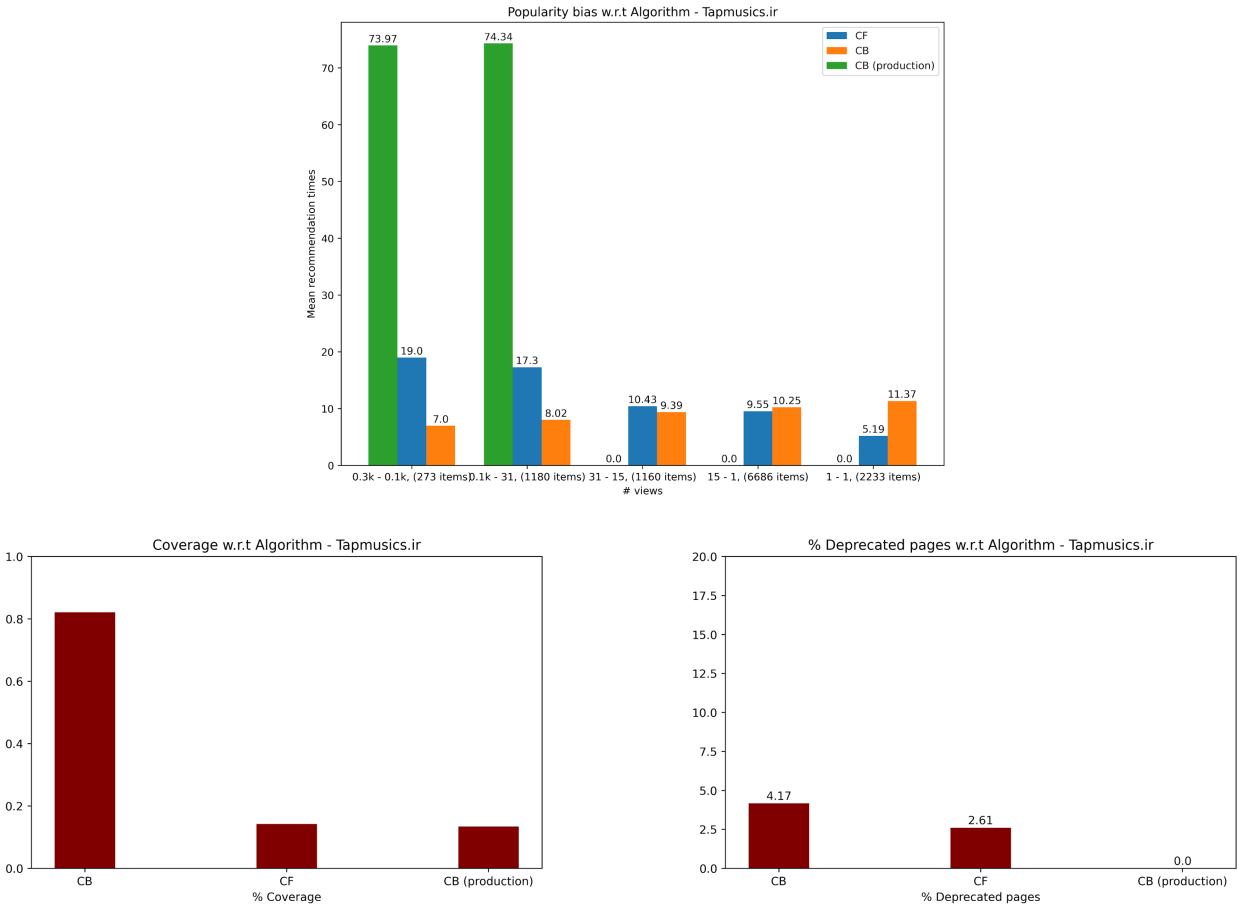

Fig. 5: Popularity Bias

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.

D. Online

We examined algorithm performance through online metrics such as CTR and PPS in our online evaluation section. The number of users and items of websites on which we conducted online tests can be seen in Table 1. These tests were organized in October 2021. Recommendation boxes are located below the item by default. These boxes include four to six recommendations generated by one or more algorithms such as CB, CF, website-trend, category-trend and user-based. Yektanet Co. has its own logging mechanism; Yektanet's fetch and click services perform event logging whenever a recommender box is fetched or when a recommendation is clicked. We have subsequently saved these logs in an elastic search database via Fluent Bit tools and the Elastic Kibana interface was implemented for the visualization of results.

In Figure 6, the box CTR is shown per algorithm. Hybrid algorithms evidently have a better overall performance. It is also discernible that CB algorithms performed better on news websites and CF algorithms are optimal for music and movie platforms.

The present diagram displays algorithm fill rates as reported by the system. The CB algorithm found itself able to create recommendations for almost 100 percent of the pages while the CF algorithm generates recommendations for only half of the pages; this is due to the pre-filtering imposed before conveying pages and users to the CF algorithm which eliminates users with less than two views and pages with less than five views for a denser matrix. The second segment of this diagram displays algorithm fill rates for users which is the percentage of users for which the user based algorithm has been able to create recommendations.

REFERENCES

[1] H. Abdollahpouri, R. Burke, and B. Mobasher. "Controlling popularity bias in learning-to-rank recommen-

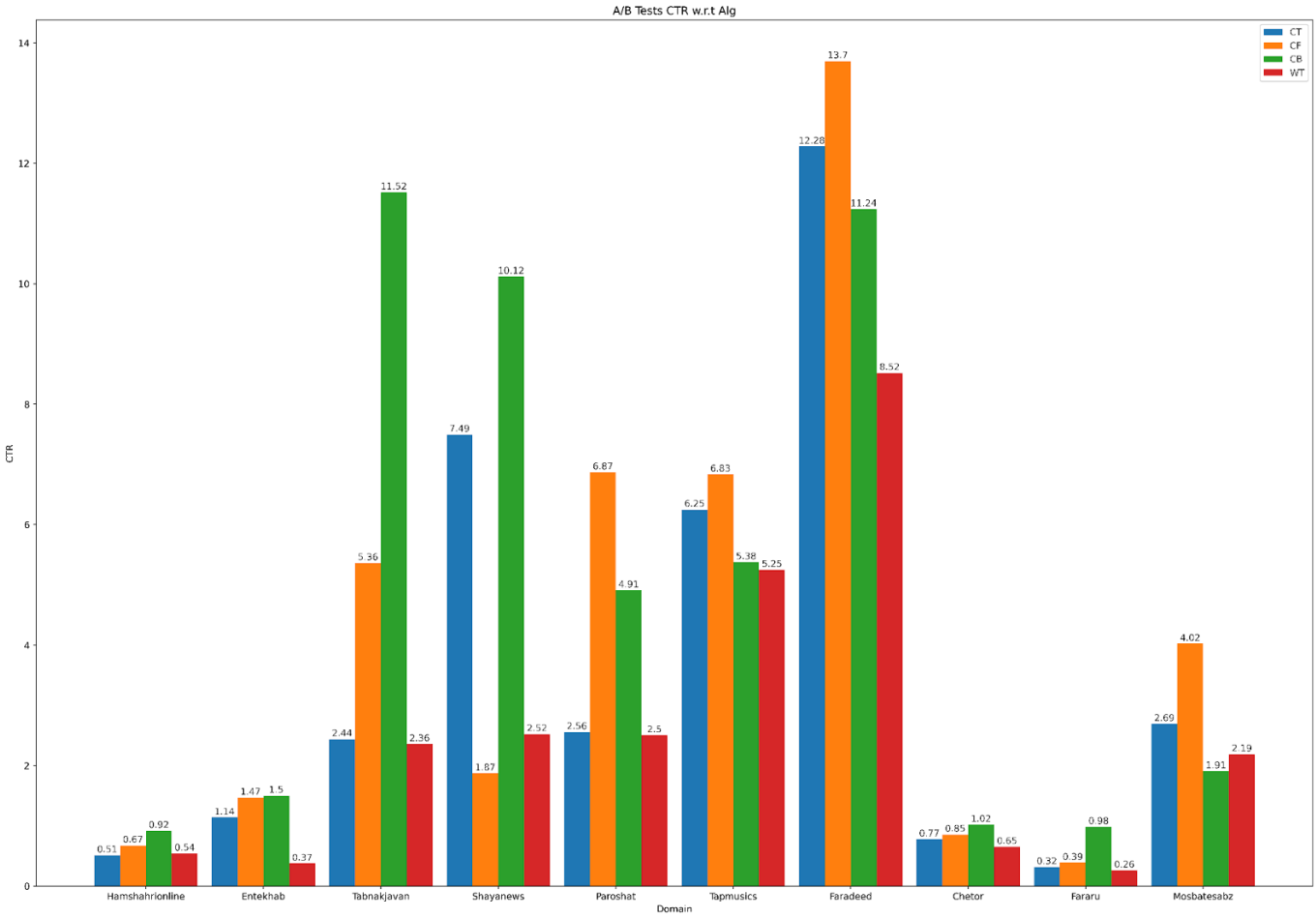

Fig. 6: CTR Metric per Algorithms

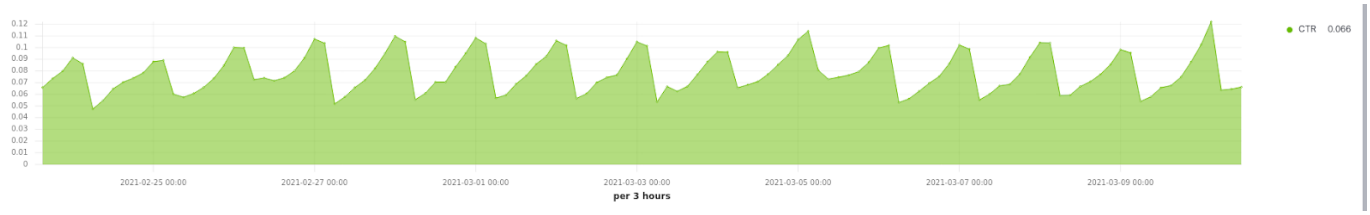

Fig. 7: System Requests in Two Weeks

dation". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17. New York, New York, USA: ACM Press*. 2017.

[2] H. Abdollahpouri, R. Burke, and B. Mobasher. "Controlling popularity bias in learning-to-rank recommendation". In: (2019). arXiv: 1901.07555 [cs.IR].

[3] H. Abdollahpouri et al. "User-centered evaluation of popularity bias in recommender systems". In: *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization. New York, NY, USA: ACM*. 2021.

[4] O. Barkan et al. "CB2CF: A neural multiview content-to-collaborative filtering model for completely cold item recommendations". In: *Proceedings of the 13th ACM Conference on Recommender Systems. New York, NY, USA: ACM*. 2019.

[5] J. Beel. "It's time to consider 'time' when evaluating recommender-system algorithms [proposal]". In: (2017). arXiv: 1708.08447 [cs.IR].

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

[7] P. Castells, S. Vargas, and J. Wang. *Novelty and diversity metrics for recommender*

systems: Choice, discovery and relevance. https://repositorio.uam.es/handle/10486/666094 (Accessed: January 16, 2022). 2011.

[8] C.-M. Chen et al. "Collaborative similarity embedding for recommender systems". In: *The World Wide Web Conference on - WWW '19. New York, New York, USA: ACM Press*. 2019.

[9] J. De Pauw. "Exploratory methods for evaluating recommender systems". In: *Fourteenth ACM Conference on Recommender Systems. New York, NY, USA: ACM*. 2020.

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

[11] F. Garcin et al. "Offline and online evaluation of news recommender systems at swissinfo.ch". In: *Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14. New York, New York, USA: ACM Press*. 2014.

[12] F. Gutiérrez et al. "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. New York, NY, USA: ACM*. 2019.

[13] T. Huang, Z. Zhang, and J. Zhang. "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*. 2019.

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

[15] O. Jeunen, K. Verstrepen, and B. Goethals. *Fair offline evaluation methodologies for implicit-feedback recommender systems with MNAR data*. https://www.semanticscholar.org/paper/1e5b3a5b1e1c60834f1542e41565ad81ad7980b4 (Accessed: January 16, 2022). 2018.

[16] D. Kowald, M. Schedl, and E. Lex. "The unfairness of popularity bias in music recommendation: A reproducibility study". In: *Lecture Notes in Computer Science* (2020), pp. 35–42.

[17] S. I. Ktena et al. "Addressing delayed feedback for continuous training with neural networks in CTR prediction". In: *Proceedings of the 13th ACM Conference on Recommender Systems. New York, NY, USA: ACM*. 2019.

[18] A. Maksai, F. Garcin, and B. Faltings. "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*. 2015.

[19] S. M. McNee, J. Riedl, and J. A. Konstan. "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*. 2006.

[20] L. Peska and P. Vojtas. "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*. 2020.

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

[22] M. Rossetti, F. Stella, and M. Zanker. "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*. 2016.

[23] H. Tahmasebi, R. Ravanmehr, and R. Mohamadrezaei. "Social movie recommender system based on deep autoencoder network using Twitter data". In: *Neural computing & applications* 33.5 (2021), pp. 1607–1623. DOI: 10.1007/s00521-020-05085-1.

[24] S. Vargas and P. Castells. "Rank and relevance in novelty and diversity metrics for recommender systems". In: *Proceedings of the fifth ACM conference on Recommender systems - RecSys '11. New York, New York, USA: ACM Press*. 2011.

[25] S. Vrijenhoek et al. "Recommenders with a mission: Assessing diversity in news recommendations". In: *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval. New York, NY, USA: ACM*. 2021.