A Comparative Study of 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 one of the most common Artificial Intelligence applications used to aid users in finding relevant items efficiently. A recommender system's performance, in a practical view, is coupled with the satisfaction of users and providers. Satisfaction, however, is not easily transformable into a mathematical optimization problem and thus, not represented by offline metrics such as accuracy and diversity. Although most state-of-the-art studies assess their accuracy with offline tests and metrics, many scholars have claimed that in today's marketplace, online tests such as A/B testing are more suitable in this regard. On the other hand, most recent studies only emphasize the utilization of recommender systems in circumstantial environments, discerning them from the wide-reaching nature of the present study. 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. In an endeavor to understand the behavior of these algorithms in various environments, we avoid discrimination against any algorithm on any platform and advocate taking advantage of what distinguishes each from the others. We utilize multiple offline and online analysis in an attempt to provide insight as to where these algorithms perform most efficiently and which method best fits our specific needs, given datasets, or scale of work. This work is a comparative study on the process of analyzing a large-scale nationwide recommender system that has been operating in the previous year on about 100 websites, with a load of approximately 300 requests per second. This system demonstrates user-based and item-based recommendations employing algorithms structured upon content-based (CB), collaborative filtering (CF), trend-based, and hybridization of those. 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) and page per session (PPS). It is worth mentioning that this recommender system addresses complications such as cold starts and popularity bias.**

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

I. Introduction

Recommender Systems 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 recommender systems to display content more efficiently to their users. [14, 23, 15] Recommender systems 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. The past decade has witnessed reasonable maturity in recommendation system algorithms. We observe the use of more traditional methods such as CB (exercising contextual similarity) and CF (exercising the deployment of methods such as matrix factorization on feedback datasets) algorithms as well as RL and deep-learning-based methods in state-of-the-studies as a continuous effort for the development of more complex algorithms, also applicable in academic environments with suitable performance. 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. Academic studies make a habit of assessing one algorithm's superiority to the other, while a practical context intends to satisfy user and provider needs. Parallel to "Recommendation with a Purpose" [16], our eventual goal is to contribute to the efficiency of methods in a practical environment. Another significant distinction between academic and practical circumstances is the evaluation method. More commonly used in the academic setting, offline tests depend on accuracy and diversity metrics. In contrast, online tests rely on clicks on recommendations (CTR) and the time users spend on the website (PPS). It has been detected that offline metrics are not able to adequately predict the performance of algorithms in the real world [24]. Another component contributing to the discrepancy between practical and academic environments is the narrowness of academic research datasets when it comes to variety in topic and scale. In the real world, attributes of recommended items and user motivation and intention are fluid within different contexts, making the feedback unpredictable. Different types of online tests come with their own set of complications. One type being the A/B test in which different recommendations will be presented to the users, and one group may receive better recommendations and others may receive inferior recommendations. Furthermore, Online evaluations should be

done continuously, As declared in [6], comparisons must be expressed through time as the performance of recommender systems will not stay the same.

A. Yektanet Co.

1-1 Yektanet co. The present study has been conducted in Yektanet co., the most prominent digital advertising establishment in Iran. This company provides recommender systems 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 100 websites. Yektanet co. delivers user-based and item-based recommendation systems engaging content-based, collaborative filtering, trend-based, and some hybrids. Yektanet's clients include e-commerce, media streaming platforms, news broadcasting websites, and digital publishing networks. 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 click-through-rate (CTR) and page-per-session (PPS) metrics. Yektanet also enlists visual analyses and manual tests in the first stages of system development for assessment.

II. Litrature Review

Over the last few years, many studies have been conducted concerning RS evaluation, most of which [13, 7] have concluded that offline tests can not report the system's efficiency on their own. However, certain studies attempted to perform these offline tests and enhance them in the process. Examples of these studies are [26, 28], using diversity and novelty alongside accuracy metrics, or [17], which implied that we may use part of the matrix that occurs at the latest timely occasion instead of a random percentage of the matrix to evaluate the collaborative filtering methods (in CF methods). They introduced SW-eval (sliding window evaluation) as a superior substitute for LOOCV (Leave-One-Out Cross-Validation) in their work. Works such as [24] evaluated recommender system performance by designing questionnaires. [22, 25, 19, 20] are amongst the studies that made an effort to extract the correlation between offline and online tests and predict CTR via offline metrics. [11] envisaged the recommender system as a blackbox to gain insight into the model behavior and improve evaluation methods aiding in model selection. It was quite surprising that in the 2017 RecSys challenge which was focused on job recommendations, traditional methods predicted user preferences on unknown items more adroitly [4, 27]. In this challenge, offline performance was more or less similar amongst different methods. The winning method, however, was a content-based approach. As has been discussed in [23, 6], not only is it better for recommender systems to be evaluated in online tests, but online metrics must be calculated over time. The performances of two distinguished algorithms may change over time in an online environment. An illustration of this conundrum is that when a recommender system is loaded onto a website, a period of time has to pass before users rely on this recommender system, as proven on the CTR w.r.t time diagrams. This is known as a trust-related complication. [16] has appraised recommender systems from two separate perspectives: the users' point of view and the providers'. Users want to discover new items and alternatives in their area of interest; they also want their needs fulfilled. Providers aspire to create new demands in the user, introduce new services, improve user retention and eventually increase revenue. We must have factors that catch whether these needs are being met in our recommender systems. Alongside this, [16] familiarizes us with different scenarios the users might find themselves in which we may use for improved accuracy in our recommendations. The user might be just browsing, he might be after similar items or he might solely want to find popular items. Our work attempts to be more general, comprehensive of what we consider overlooked in studies analyzing and evaluating the performance of recommender systems. We have factored in the differences between websites varying in type and scale of data; different data subsets of interest differ in performance carried on different algorithms. We have examined different algorithms, made comparisons, and addressed the circumstantial nature of these algorithms. We have ultimately carried out both online and offline tests.

The use of each algorithm is circumstantial as they each come with their own set of costs and benefits. Collaborative filtering algorithms have popularity biases and tend to recommend popular items more often. The website's landing page may expose a set of items more than the others that cause an unfairly higher number of views for those items. Implicit feedback datasets [9] are widely impaled with a popularity bias since they solely commit to clicks and views, as opposed to explicit feedback datasets, which rely on rankings and like/dislike buttons. [1, 18] confer that items grouped under "long tail", which happen to be better drivers for a provider's growth, are recommended much less than a small number of items that are grouped as popular. Metrics such as gini-index and group-average-popularity are applied to showcase these intricacies. Studies like [2, 3] have exercised reranking methods to appoint an individual user's interests in the production of recommender systems to subdue popularity bias and control the effects of item exposure. In the present study, we have tried to minimize these repercussions via the tuning of hyperparameters in collaborative filtering algorithms. CF algorithms also contain a subordinate fill rate and coverage. The portion of users/items that the recommender is able to create recommendations for is called fill-rate and the portion of items that the system recommends is defined as coverage. An element contributing to CF's lower fill rate value is the cold start problem which materializes when we do not have sufficient data about some users or items. Websites with heightened item churn face this issue quite often as hundreds of news articles may be published daily. [12] titled "when collaborative filtering's data is not enough" deliberates this area closely. State-of-the-art studies such as [5] are prone to utilizing hybrid methods in an attempt to battle the cold start problem. In Yektanet co, when there is not enough data on the user or the item, we retreat to a self-organized fallback

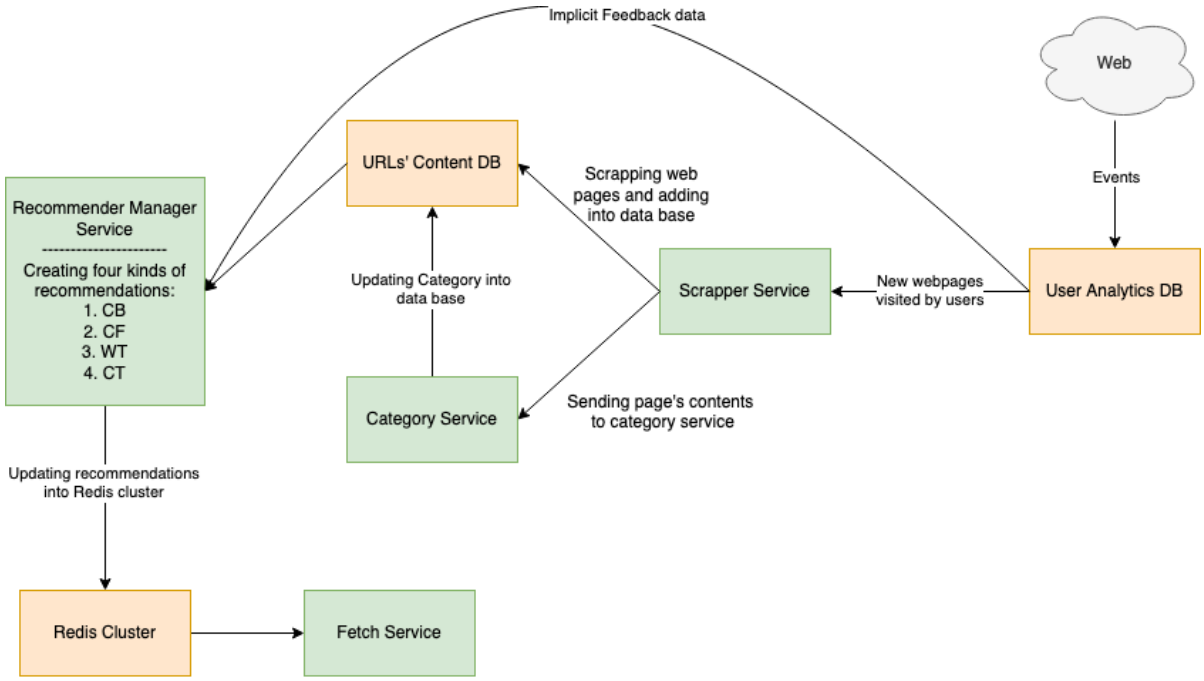

Fig. 1: architectural view of Yektanet's recommendation system

strategy that presents less personalized and more generalized recommendations. When CF recommendation is not present, for instance, the recommender system generates content-based, category-trend, or website-trend recommendations. We also rerun our algorithms more frequently for websites with higher item churn. Regularly, the primary metrics for evaluating recommender systems are accuracy-based (e.g., MAP, NDCG, and AUC). [21] has declared that an appraisal of accuracy is not enough and may even hurt RS performance. Users prefer more exciting, surprising, novel, and diverse recommendations. Novelty is defined as how unique the item appears to the user, and diversity is achieved by the recommended items differing from each other [8]. [10] suggests that recommendations may be produced inside a filter bubble, limiting their novelty. To mitigate this issue, we imposed some thresholds on cosine similarity between recommended items.

III. Methodology

This section provides a detailed description of Yektanet's recommender system. This system provides recommendations for around 100 websites. We set cookies for our users' and track their views and use this data to build an implicit feedback matrix. Contents of all pages are also crawled for the content-based algorithms. A variety of algorithms have been implemented, which will be introduced at length. 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. For the content-based 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. Collaborative filtering 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 [9], 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

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

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 content-based 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.

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 collaborative filtering 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

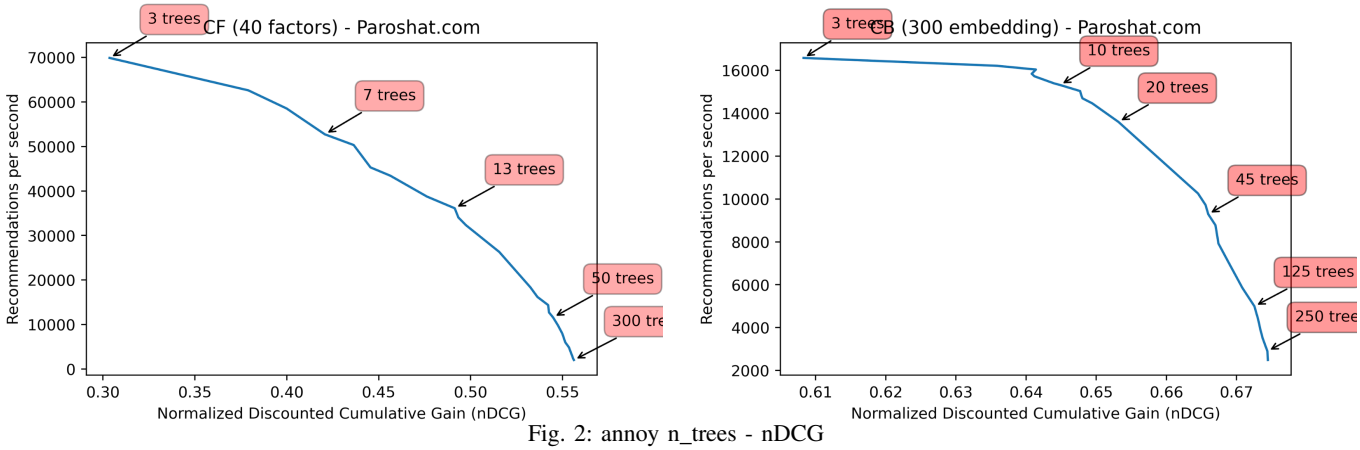

Fig. 2: annoy n_trees - nDCG

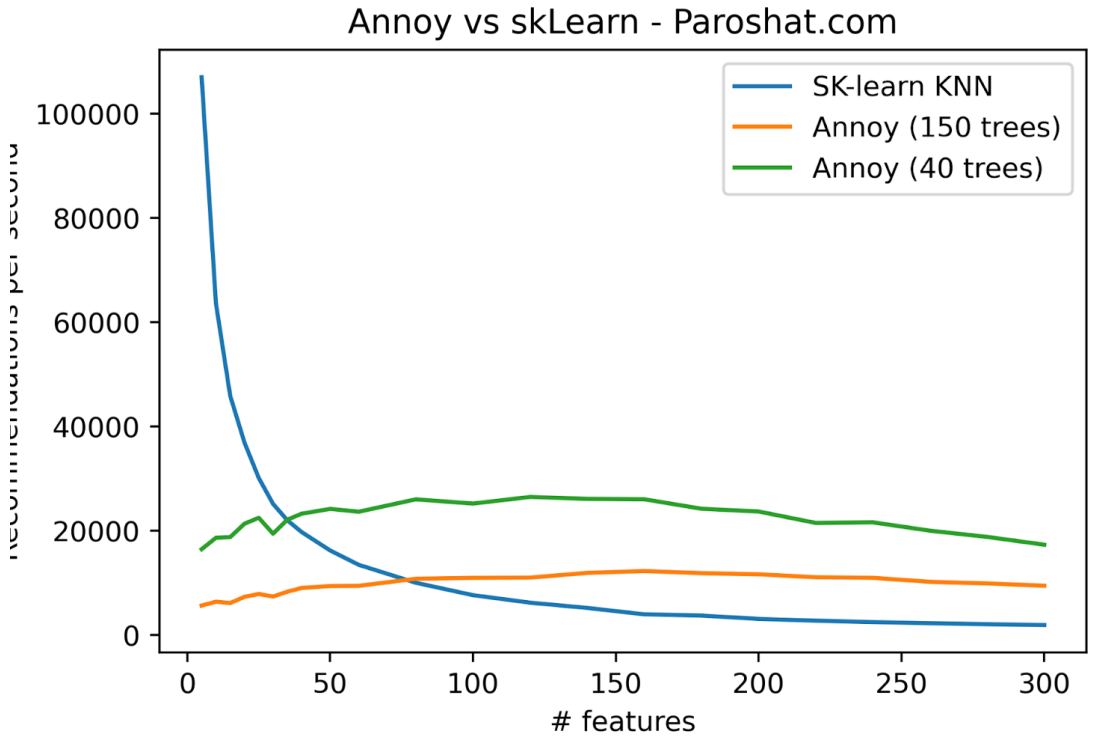

Fig. 3: annoy vs sklearn on paroshat

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

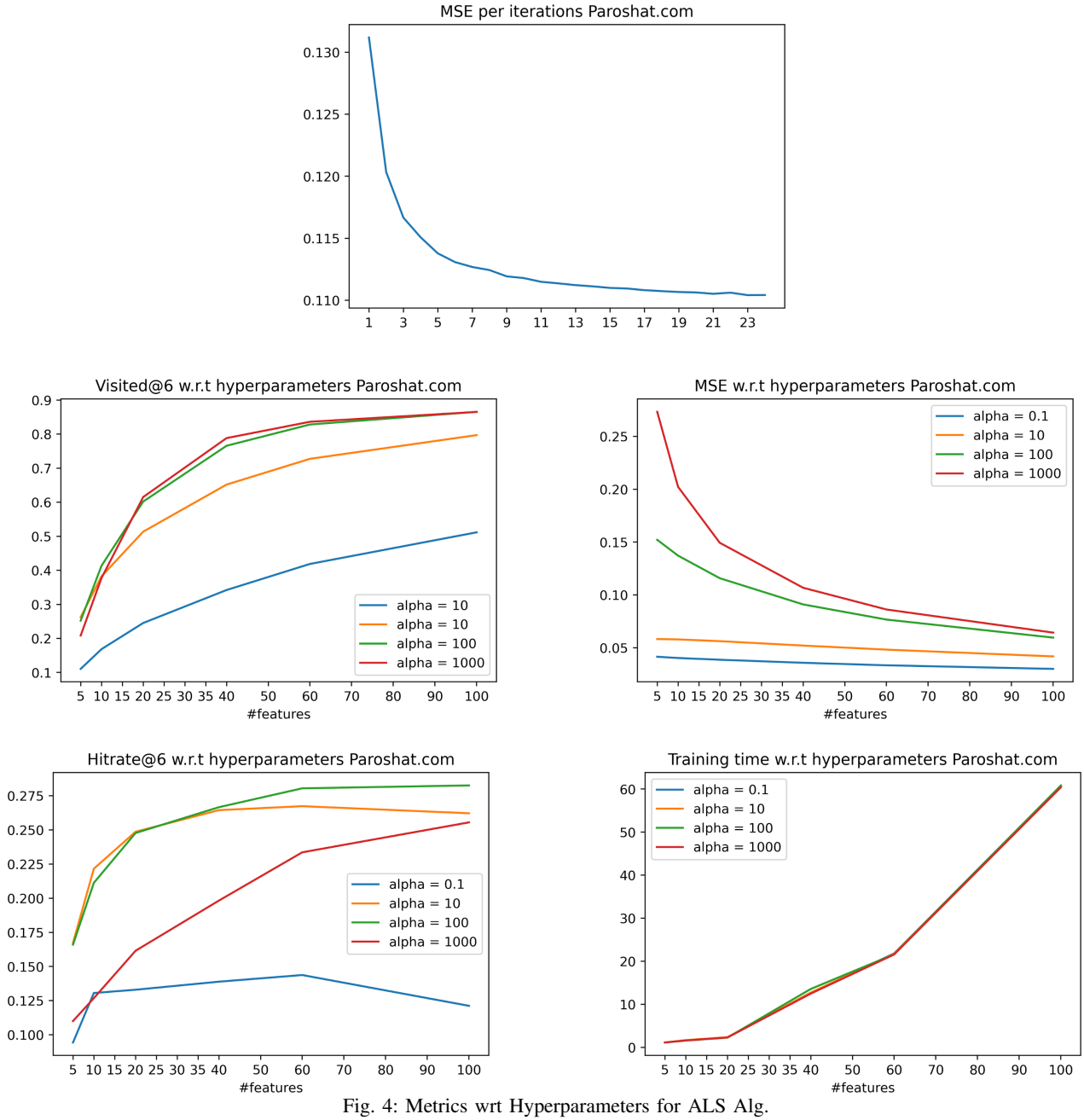

Fig. 4: Metrics wrt Hyperparameters for ALS Alg.

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

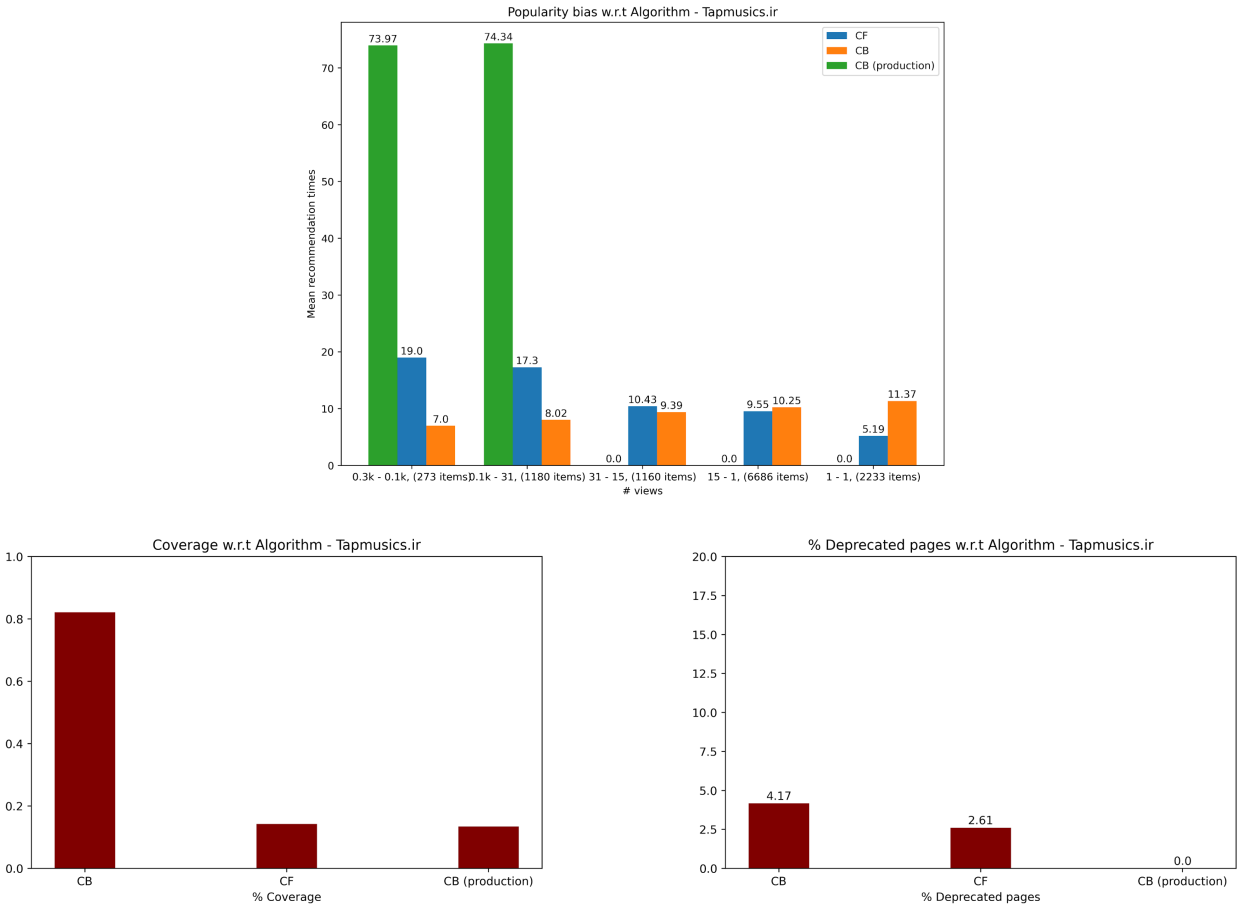

Fig. 5: Popularity Bias

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 collaborative filtering 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 content-based 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 (content-based algorithm, pre-filtered content-based algorithm and collaborative filtering 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 content-based, collaborative filtering, website-trend,

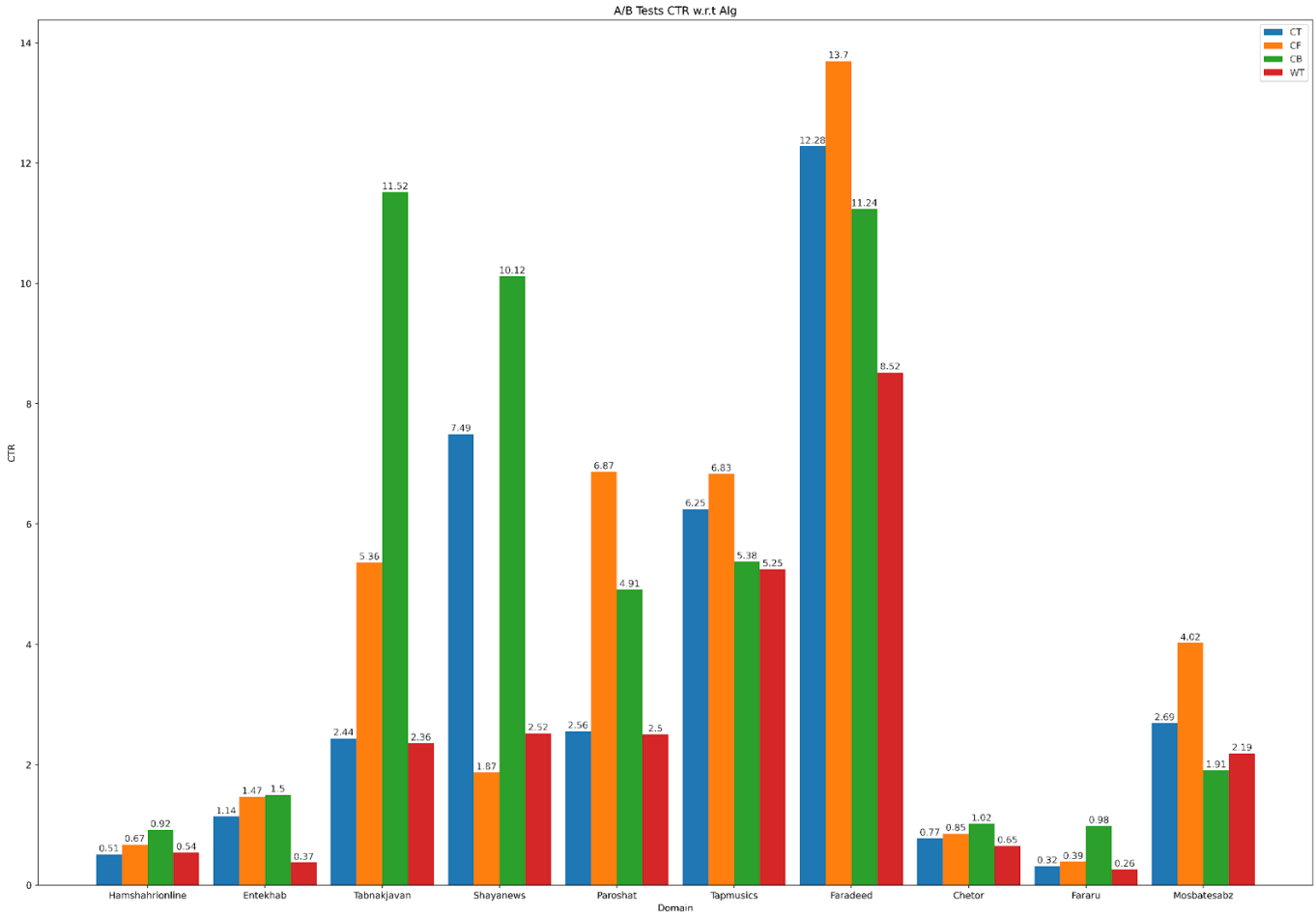

Fig. 6: CTR Metric per Algorithms

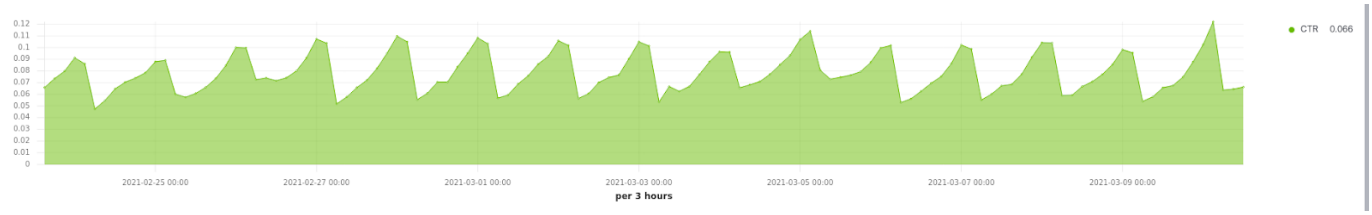

Fig. 7: System Requests in Two Weeks

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.

V. Refrences

[1] Abdollahpouri, H. et al. (2021) "User-centered evaluation of popularity bias in recommender systems," in Pro-

ceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization. New York, NY, USA: ACM.

[2] Abdollahpouri, H., Burke, R. and Mobasher, B. (2017) "Controlling popularity bias in learning-to-rank recommendation," in Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17. New York, New York, USA: ACM Press.

[3] Abdollahpouri, H., Burke, R. and Mobasher, B. (2019) "Managing popularity bias in recommender systems with personalized re-ranking," arXiv [cs.IR]. Available at: http://arxiv.org/abs/1901.07555.

[4] Abel, F. et al. (2017) "RecSys Challenge 2017: Offline and Online Evaluation," in Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17. New York, New York, USA: ACM Press.

[5] Barkan, O. et al. (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. New York, NY, USA: ACM.

[6] Beel, J. (2017) "It's time to consider 'time' when evaluating recommender-system algorithms [proposal]," arXiv [cs.IR]. Available at: http://arxiv.org/abs/1708.08447.

[7] Beel, J. and Langer, S. (2015) "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. Cham: Springer International Publishing, pp. 153–168.

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

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

[10] Chen, L. et al. (2019) "How serendipity improves user satisfaction with recommendations? A large-scale user evaluation," in The World Wide Web Conference on - WWW '19. New York, New York, USA: ACM Press.

[11] De Pauw, J. (2020) "Exploratory methods for evaluating recommender systems," in Fourteenth ACM Conference on Recommender Systems. New York, NY, USA: ACM.

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

[13] Garcin, F. et al. (2014) "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.

[14] Gutiérrez, F. et al. (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. New York, NY, USA: ACM.

[15] Huang, T., Zhang, Z. and Zhang, J. (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.

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

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

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

[19] Ktena, S. I. et al. (2019) "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.

[20] Maksai, A., Garcin, F. and Faltings, B. (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] McNee, S. M., Riedl, J. and Konstan, J. A. (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] Peska, L. and Vojtas, P. (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] Ray, B., Garain, A. and Sarkar, R. (2021) "An ensemble-based hotel recommender system using sentiment analysis and aspect categorization of hotel reviews," Applied soft computing, 98(106935), p. 106935. doi: 10.1016/j.asoc.2020.106935.

[24] Rossetti, M., Stella, F. and Zanker, M. (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] Tahmasebi, H., Ravanmehr, R. and Mohamadrezaei, R. (2021) "Social movie recommender system based on deep autoencoder network using Twitter data," Neural computing applications, 33(5), pp. 1607–1623. doi: 10.1007/s00521-020-05085-1.

[26] Vargas, S. and Castells, P. (2011) "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.

[27] Volkovs, M., Yu, G. W. and Poutanen, T. (2017) "Content-based neighbor models for cold start in recommender systems," in Proceedings of the Recommender Systems Challenge 2017 on ZZZ - RecSys Challenge '17. New York, New York, USA: ACM Press.

[28] Vrijenhoek, S. et al. (2021) "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.