What Happens When We Make the Wrong Assumptions? Exploring “Success” in Recommendation

Jessie J. Smith

Introduction

In personalized recommender systems, machine learning algorithms learn from past user preferences in order to predict their future interests. Personalized systems are often accuracy-based; they define and measure success based on how accurately they represent the inferred interests of their users. The assumption in many of these systems is that the more *accurately* a system performs, the better the system is. Thus, the heuristics that we use to define and measure accuracy are just as, if not more important than the results of these measurements. Realistically, when we measure the accuracy of a machine learning system, we are actually measuring how well that system solves a surrogate problem, not how well that system recommends items. The assumption is that if our performance metrics are good, our system is recommending items well -- however this is not necessarily always the case. There are unintended and unexpected consequences that occur as a result of our assumptions about user preferences. To explore these consequences, in this work I conducted an empirical study to measure the impact that our assumptions can have on the performance of a recommender system using data from TikTok, a short video recommendation platform.

Related Work

In a paper written by Covington, Adams, and Sargin on YouTube’s deep learning recommender system, the authors named a concept called the **Surrogate Problem** (Covington, Adams, and Sargin, 2016). A surrogate problem is described as the assumption that we make that fuels the problem that we are trying to solve with our recommender system. In the example of 2016 YouTube, their surrogate problem was if they could predict which videos users are more likely to watch more of, then they could make better recommendations. In order for surrogate problems to result in effective recommender systems, there is an underlying assumption that must be true. In this case of YouTube, the assumption made in their surrogate problem was that *users want to be recommended videos that they are more likely to finish* (or get close to finishing). The assumption that is made in a surrogate problem selection also informs which feature or features will be used as positive labels or indicators of success in our recommendation model. In this scenario, the positive signal for YouTube was to use “video finishes” as the feature that indicated a positive signal to their system.

Naturally, selecting an appropriate surrogate problem (and making an appropriate assumption about user preferences) is crucial to creating a robust recommender system for various reasons: (1) overfitting; (2) unintended consequences; and (3) wrong assumptions.

1. **Overfitting**

First, it is easy to overfit a model to the surrogate problem. GoodHart's law states that, *"when a measure becomes a target, it ceases to be a good measure,"* (Goodhart, 1984). Similarly, in the world of recommender systems, when our indicators of successful systems rely on assumptions that we make about user preferences, those assumptions can also cease to be good measures -- especially when those assumptions might be necessarily wrong. In the case of YouTube, when they previously made the assumption that "users clicking on a video means that they want to be recommended more of the same content" -- they selected "video clicks" as a positive signal in their system, and consequently recommended far more clickbait (Meyerson, 2012). Another example from Covington during his presentation at RecSys 2016 described how the assumption that "users watching more minutes of a video means they want to be recommended more of the same content," would lead to selecting "minutes of video watched" as a positive signal in their system, and would result in recommending longer videos over shorter videos regardless of if the shorter videos were more engaging or relevant.

2. **Unintended Consequences**

Second, there are unintended consequences of decisions made in any machine learning system, but especially for decisions regarding the surrogate problem and defining heuristics for when our system is or isn't successful. In the case of YouTube, choosing surrogate problems that result in recommended clickbait or longer videos might lower user satisfaction and experience, resulting in a loss of user retention. Another well-known example of an assumption for a surrogate problem in recommendation is that "accurately predicting ratings leads to effective movie recommendations," (Amatriain, 2012). In this example, the surrogate problem tends to work well for A/B testing in a live setting, but is very difficult to measure in offline experiments (Covington, Adams, and Sargin, 2016). Additionally, if video recommendation platforms focus on watch time as the best positive signal, then they are building a platform designed to keep users *engaged* and on the platform for longer. In a blog post by YouTube, they explicitly state how they have designed their system around this assumption: *"if viewers are watching more YouTube, it signals to us that they're happier with the content they've found. It means that creators are attracting more engaged audiences,"* (Meyerson, 2012). However, previous research has also shown that the goal of keeping users engaged and on the YouTube platform for longer can unintentionally be leading users down malicious content rabbit holes such as conspiracy theory content (Ribeiro et. al, 2020).

3. **Wrong Assumptions**

Finally, as mentioned previously, the assumption(s) that we make in our surrogate problem ultimately inform the features we select as positive signals from user data. It is these positive signals that are used to define and measure "success" in the recommender system. Going back to the clickbait example, we learned that using the signal of "user clicks" on a video led to a system that recommended more clickbait. In this scenario, if the goal of the system was to learn from previous "clicks" that users had made and to recommend more items to maximize this signal in the future, then a recommendation list full of clickbait would yield a positive accuracy result (e.g., a high precision). However, despite the fact that the system is reportedly performing well, the performance metric is measuring how successful the system was at solving the

surrogate problem, not necessarily how successful the system is at *providing good recommendations*. Thus, if we were to make a wrong assumption -- that is, if we were to incorrectly assume that an implicit or explicit signal from a user indicates that they want to be recommended items based off that signal -- then our system could appear as though it is providing good recommendations, when in reality it is doing the opposite.

One additional framework of relevance to this work is the concept of **Algorithmic Folk Theories**, which is the idea that users have their own personal beliefs and understanding about how an algorithm works (DeVito, 2021). On YouTube or other video recommendation platforms such as TikTok, you can see thousands of videos of content creators describing their own folk theories of how the recommender system works for that platform. One interesting thing to note is that despite the fact that most users actually do not know the specifics of how recommendation algorithms work, their personal beliefs about how they work drives their actions on the platform (Karizat et. al, 2021). For example, if someone truly believes that TikTok only recommends videos to them based on when they 'comment' on a video as a positive signal, then they are more likely to 'comment' on more videos and less likely to behave in the way that the platform was expecting them to behave. This means that they could be providing a very explicit signal to the algorithm about what they want to be recommended, but the algorithm could be ignoring it, because of the incorrect assumptions that we made about that user and their own preferences for the kind of content they want to be recommended.

Research Question

In order to address some of these concerns, I sought to explore the discrepancy between perceived success (are we solving the surrogate problem) and real success (are we providing good recommendations to users) in the domain of short video recommendation. Specifically, I was curious what would happen if we make a wrong assumption about the kinds of implicit or explicit signals users want to be utilized for their recommendation algorithm. For example, what if we assume that users want to be recommended videos that they are more likely to finish, but that isn't the case?

Methods

In this section, I describe the methods used in my study and experimental design and evaluation for this project.

Datasets

The dataset used for these experiments was adapted from the ICME 2019 competition on short video recommendation (Cheng et. al, 2019). After modifying the data to be compatible with librec-auto, the open-source software used to run the experiments (Sonboli et. al, 2021), I had approximately 19,622,340 instances of real, historical user interactions with TikTok videos. In order to use the data for experiments on my machine, I had to sample it to a smaller size. I decided to sample with a k-core sampling method using a k value of 180, which meant that my final dataset would only consist of users who had interacted with at least 180 videos, and videos

that had been interacted with by at least 180 users. This sampling method resulted in the final dataset used for experiments, which had 1,705,250 user-item interactions.

From this sampled dataset, I created two separate versions in order to run my experiments. In one version of the dataset, I gave a rating of “1” for videos that the user had liked and a rating of “-1” for every other kind of interaction. In the other dataset, I gave a rating of “1” for videos that the user had finished, and a rating of “-1” for every other kind of interaction. The goal was to create two datasets that used different implicit or explicit positive signals from users, to demonstrate two systems that were built off of different assumptions about user preferences.

Technical Ideas and Plan

I decided to run two separate rounds of experiments with the TikTok data. I ran the two rounds of experiments based on two assumptions:

1. The assumption that users liking a video means that they want to be recommended more of the same content
2. The assumption that users finishing a video means that they want to be recommended more of the same content

I wanted to know if we trained TikTok’s recommender system to perform well for one assumption, would it perform more poorly for users whose folk theories and actions on the platform aligned with the alternate assumption?

First, I trained two control recommender systems with two different positive signals. To train these models, I tested out 3 different algorithms and tuned the hyperparameters to try to find the model that performed best. The three algorithms and their hyperparameters are detailed in Table 1. I measured performance through precision and ndcg. Then, I ran my two control studies: (1) A model that was trained and tested on users ‘liking’ a video as the positive signal, and (2) a model that was trained and tested on users ‘finishing’ a video as a positive signal. Finally, I ran my experiments. For each of the models that I had tuned, I tested them with new data that used the alternative feature as the positive signal. This meant that for control model #1 that was trained on user ‘likes’ as the positive rating, I tested the model on user ‘finishes’ as the positive rating, and then for control #2 that was trained on user ‘finishes’ as the positive rating, I tested the model on user ‘likes’ as the positive rating. You can see these experiments in a visual representation in Figure 1.

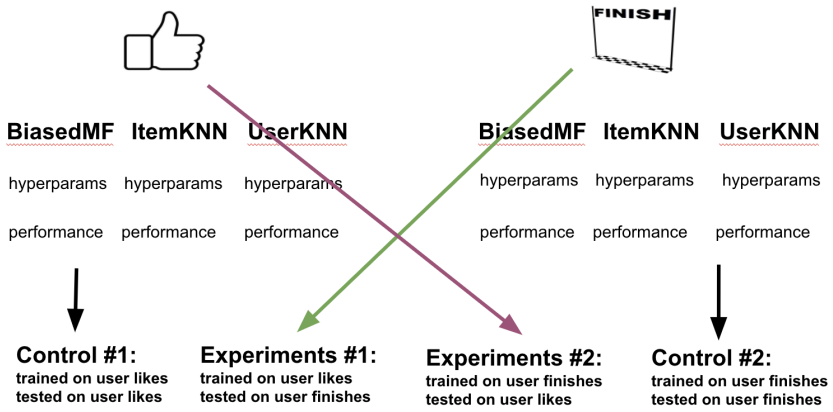

Figure 1: The control and experimental study setup

Algorithm	num factors	list size
biasedmf	60	100
biasedmf	80	100
biasedmf	100	100
biasedmf	60	200
biasedmf	80	200
biasedmf	100	200
biasedmf	60	400
biasedmf	80	400
biasedmf	100	400
Algorithm	**neighborhood size**	**list size**
itemknn	50	100
itemknn	50	200
itemknn	50	400
itemknn	100	100
itemknn	100	200
itemknn	100	400
itemknn	200	100
itemknn	200	200
itemknn	200	400
Algorithm	**neighborhood size**	**list size**
userknn	50	100
userknn	100	100
userknn	200	100
userknn	50	200
userknn	100	200
userknn	200	200
userknn	50	400
userknn	100	400
userknn	200	400

Table 1: The algorithms and hyperparameters that were tested for each of the control and experimental datasets. Each of these combinations was tested with control #1 (trained and tested on likes), control #2 (trained and tested on finishes), experiment #1 (trained on likes, tested on finishes) and experiment #2 (trained on finishes, tested on likes). The results of these experiments can be seen in the appendix.

Hypothesis

In order to compare the results of these two rounds of control studies and experimental studies, I decided to place myself into the role of a data scientist at TikTok who was attempting to build the best model they could, given a specific surrogate problem. For example, in Control #1 -- the data scientist would be given a dataset of historic user-item ratings on TikTok, where the “rating” was a “1” if the user had liked a video, and a “-1” if any other kind of interaction had happened. In this scenario, the data scientist would attempt to build the model that best recommends videos to users that are likely to receive this positive signal -- which in this example would mean recommending videos to users that they are more likely to explicitly click the “like” button on. The data scientist in theory would try out different kinds of algorithms and tune various hyperparameters on those algorithms, and then find the combination of algorithm and hyperparameters that resulted in the best performance. Then, that model would be chosen to deploy to production, because it performed the best on the training data.

My hypothesis was that a model that was trained to perform well for one surrogate problem would likely perform worse for users who did not conform to the assumptions of that surrogate problem. For example, if we were to discover that the BiasedMF algorithm had the best precision and NDCG with a factor size of 100 and a list size of 100 for the Control #1 data, then in Experiments #1, for the BiasedMF algorithm that was trained with these hyperparameters for user “likes” -- the model would perform more poorly for users whose positive signal was “finishes.”

In other words, if the best performing model in each round of experiments performs better with the control dataset than the experimental dataset, then my hypothesis is true. However, if the experimental dataset performs better than the control dataset, my hypothesis cannot be accepted. A visual representation of my hypothesis can be seen in Figure 2.

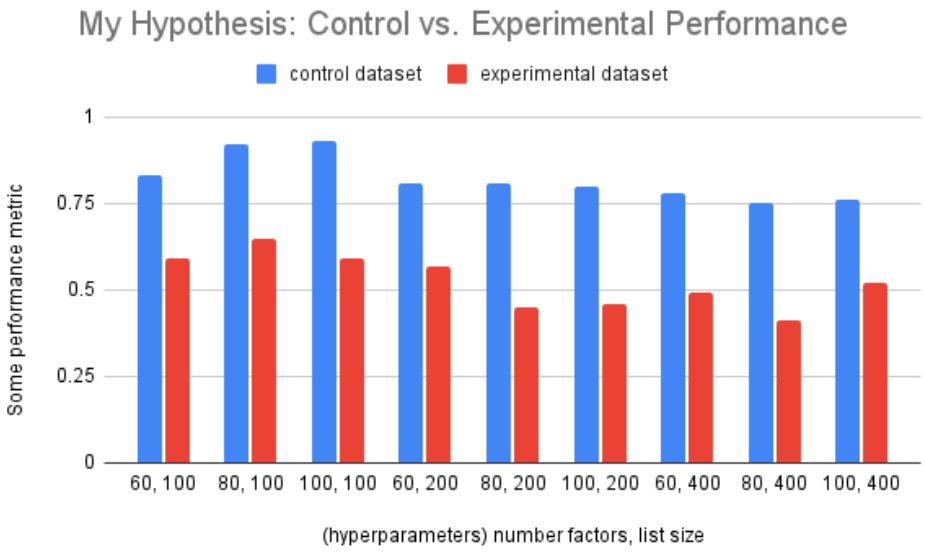

Figure 2

Results & Discussion

Control #1 and Experiments Round #1

For this set of experiments, the control models were trained on user “likes” as the positive signal, and tested on user “likes” as the positive signal. The experimental model was trained on user “likes” as the positive signal, and tested on users “finishing” a video as the positive signal. The surrogate problem for the control models was: “if we can predict what videos users will like, we can recommend better videos to them,” which was based on the assumption that users liking a video means that they want to be recommended more of the same content.

For this round of experiments, the experimental data performed better than the control data on average for both of the neighborhood based algorithms -- ItemKNN and UserKNN. For the BiasedMF algorithm, the results were mixed because I had to throw out several data points from the experimental data due to librec-auto giving negative NDCG and precision values, so there was no conclusive data from that algorithm. However, despite the fact that the experimental data performed on average better than the control data - in the specific tuned models that performed *best* with the control data, those models performed *much worse* for the experimental data. An example of this is shown in Figure 3, where the UserKNN model that performed best with the control data (with a neighborhood size of 50 and a list size of 400) was one of the models that performed worst for the experimental data.

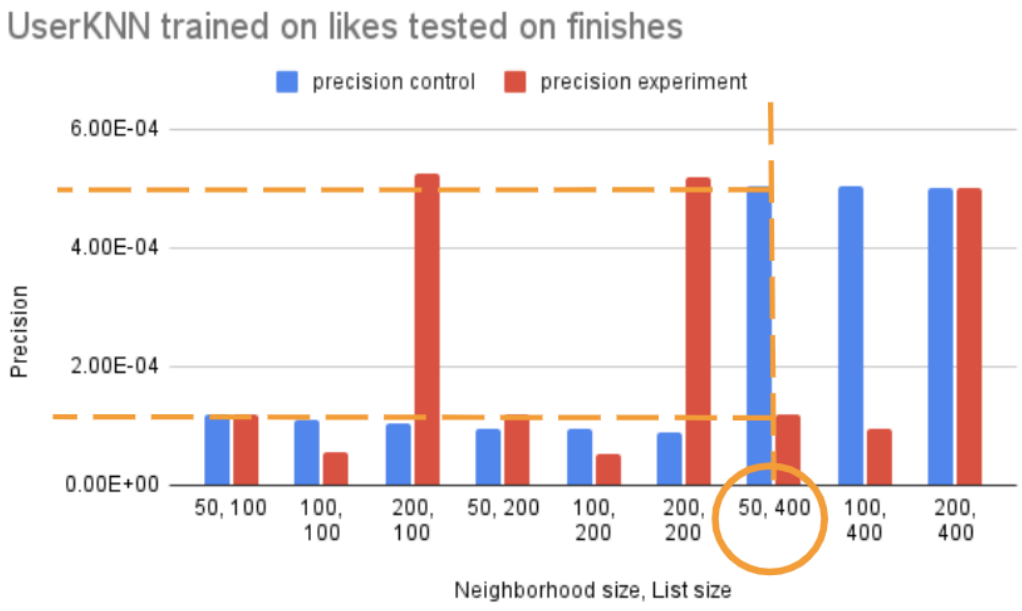

Figure 3

In this scenario, if the data scientist was to choose to deploy the model that performed best given their selected surrogate problem (using the control data), then that model would have performed significantly worse for users who do not ever ‘like’ videos as an indicator of their preferences on the TikTok platform.

Control #2 and Experiments Round #2

For this set of experiments, the control models were trained on user “finishes” as the positive signal, and tested on user “finishes” as the positive signal. The experimental model was trained on user “finishes” as the positive signal, and tested on users “liking” a video as the positive signal. The surrogate problem for the control models was: “if we can predict what videos users will finish, we can recommend better videos to them,” which was based on the assumption that users finishing a video means that they want to be recommended more of the same content.

For this round of experiments, the control data performed on average similarly or better than the experimental data, which aligned with my hypothesis. Specifically, the ItemKNN, UserKNN, and BiasedMF algorithms had higher NDCG and precision for the control data for a majority of the hyperparameter selections -- with the exception of precision on the ItemKNN algorithm. In this experiment, the precision was almost exactly the same for every hyperparameter for both control and experimental data, without any clear reason as to why. Additionally, there were several scenarios where librec-auto gave negative results for NDCG in this round of experiments, especially for the ItemKNN algorithm -- which resulted in omitting a lot of data from that round of experiments and is a limitation of this study.

However, there were two scenarios where the metrics produced from the control and experimental studies aligned very well with my hypothesis -- when measuring precision for the BiasedMF and UserKNN algorithms. These results can be seen in Figure 4. In both of these scenarios, the control data performed on average better or similar to the experimental data.

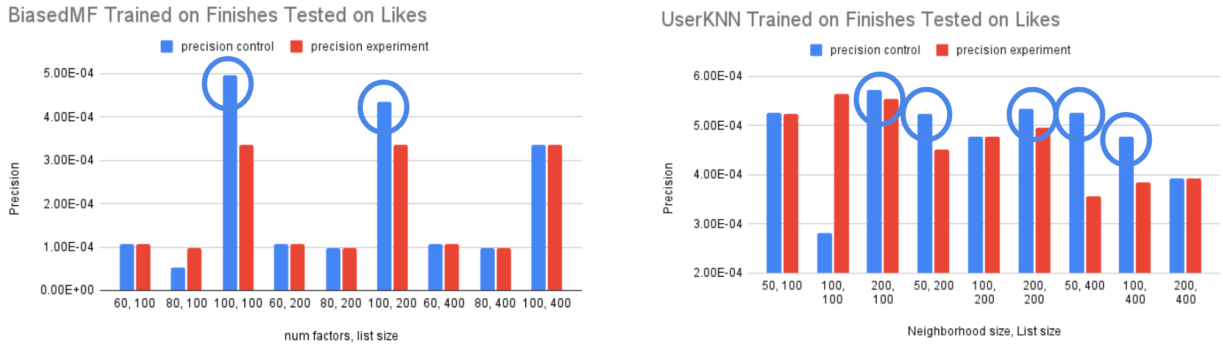

Figure 4

In this scenario, if the data scientist was to choose to deploy the model that performed best given their selected surrogate problem (the idea that users finishing a video means they want to be recommended more of that same kind of video), then that model would have performed significantly worse for users who do not want this implicit signal an indicator of their preferences on the TikTok platform.

Challenges and Limitations

Throughout this project, I encountered quite a few challenges and setbacks. First, my data was very sparse, and while running k-core sampling helped with this sparsity, it only did so when I made the k value significantly high, which meant that the only users and items that were being analyzed in my datasets were users that were very active and items that were very popular. I see this as a limitation of my study. I also ran into problems with sparsity using unary ratings, since librec-auto does not take into account a rating of “0” for the model. In order to confront this, I decided to change ratings of “0” to be “-1,” which helped with sparsity issues and improved performance for the baseline control studies. There were also a few bugs I encountered in librec-auto, such as performance metrics above 1 -- which I found the cause of, and performance metrics that were negative, which I never found the cause of and I had to throw out some data because of that. I see this missing data as a limitation of my work. I also needed to find a way to test a model with different data than what librec-auto had automatically split between training and testing sets -- so I had to come up with a way to swap in testing data with an alternative signal for librec to use during evaluation but after training. Other limitations I ran into during this study included the really low NDCG and precision values for both control and experimental data, which made it hard to find any statistically significant results or differences in performance between the controls and the experiments. In the future if I had more time I would have liked to find other ways to improve those baseline metrics as well.

Conclusions & Future Work

In conclusion, for this project I gathered, cleaned, and sampled from over 19 million user item interactions from TikTok and performed 2 control studies and 2 rounds of experiments that explored 3 algorithms each, with 9 hyperparameters for each algorithm, measuring 2 performance metrics for each hyperparameter. This means I collected a total of 216 metrics to explore and compare. I ran these studies in order to determine what the impact on users could be if we make a wrong assumption for our surrogate problem in the context of short video recommendations. Though my result data was not statistically significant enough to reject or accept my hypothesis, I did see some promising trends appear in my results. First, I noticed that in some of the experimental scenarios, user ‘likes’ and ‘finishes’ acted as proxies for one another, which would need to be addressed in future experiments on this data. I also noticed that for certain algorithms that are trained on certain assumptions, the models do perform worse for users who do not match the control assumptions. For example, in models that use UserKNN and BiasedMF algorithms that are trained on users finishing a video as the positive signal, those models appear to perform worse for users who like videos as their positive signal instead.

In the future, I would like to run these experiments with the full dataset on a machine that has a lot of computing power, to see if my sample was indicative of the population of users that I had access to. I also would have liked to run experiments with more combinations of signals to see if there were other assumptions that I could test that might be more realistic to what TikTok does in production for their own system, and also to check for other features that might be proxies for each other. I also would love to talk to real TikTok users about what sorts of folk theories they have about the algorithm, how they think that their actions are used as positive signals for the

recommender system, and also the signals they wish the algorithm used -- to incorporate their desires into future experiments with similar data. Overall, though the results of this study do not certify if my hypothesis is true or not, this study acts as an important first step into exploring the impacts that our assumptions have on the recommender models that we create, and on the users who interact with them.

Works Cited

1. Goodhart, Charles AE. "Problems of monetary management: the UK experience." Monetary theory and practice. Palgrave, London, 1984. 91-121.
2. Covington, Paul, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations." Proceedings of the 10th ACM conference on recommender systems. 2016.
3. Amatriain, Xavier. "Building industrial-scale real-world recommender systems." Proceedings of the sixth ACM conference on Recommender systems. 2012.
4. Meyerson, Eric. "Youtube now: Why we focus on watch time." YouTube Creator Blog (2012).
5. Ribeiro, Manoel Horta, Raphael Ottoni, Robert West, Virgílio AF Almeida, and Wagner Meira Jr. "Auditing radicalization pathways on YouTube." In Proceedings of the 2020 conference on fairness, accountability, and transparency, pp. 131-141. 2020.
6. DeVito, Michael Ann. "Adaptive folk theorization as a path to algorithmic literacy on changing platforms." Proceedings of the ACM on Human-Computer Interaction 5, no. CSCW2 (2021): 1-38.
7. Karizat, Nadia, Dan Delmonaco, Motahhare Eslami, and Nazanin Andalibi. "Algorithmic folk theories and identity: How TikTok users co-produce Knowledge of identity and engage in algorithmic resistance." Proceedings of the ACM on Human-Computer Interaction 5, no. CSCW2 (2021): 1-44.
8. Cheng, Guan Yu, Hui Qin Xiao, Jian Wei Li, Dong Wei Zhao, and Xiaosheng Wu. "ICME Grand Challenge on Short Video Understanding." In 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), pp. 617-620. IEEE, 2019.
9. Sonboli, Nasim, Masoud Mansoury, Ziyue Guo, Shreyas Kadekodi, Weiwen Liu, Zijun Liu, Andrew Schwartz, and Robin Burke. "librec-auto: A Tool for Recommender Systems Experimentation." In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4584-4593. 2021.

APPENDIX

Link to Datasets, Config File, and Data Cleaning Notebook

Final Results & Collected Data

					Control Round #2			Experiments Round #2	
Control 2	**Algorithm**	**num factors**	**list size**	**num factors, list size**	**ndcg control**	**precision control**		**ndcg experiment**	**precision experiment**
Experiment 1	biasedmf	60	100	60, 100	2.19E-04	1.07E-04		9.56E-05	1.07E-04
Experiment 1	biasedmf	80	100	80, 100	2.20E-04	5.37E-05		7.53E-05	9.66E-05
Experiment 1	biasedmf	100	100	100, 100	2.19E-04	4.95E-04		7.67E-04	3.36E-04
Experiment 1	biasedmf	60	200	60, 200	6.06E-05	1.07E-04		9.56E-05	1.07E-04
Experiment 1	biasedmf	80	200	80, 200	6.05E-05	9.66E-05		1.10E-04	9.66E-05
Experiment 1	biasedmf	100	200	100, 200	6.03E-05	4.34E-04		6.70E-04	3.36E-04
Experiment 1	biasedmf	60	400	60, 400	9.74E-04	1.07E-04		9.56E-05	1.07E-04
Experiment 1	biasedmf	80	400	80, 400	9.79E-04	9.75E-05		1.12E-04	9.66E-05
Experiment 1	biasedmf	100	400	100, 400	9.73E-04	3.36E-04		5.23E-04	3.36E-04
	Algorithm	**neighborhood size**	**list size**	**neighborhood size, lis**	**ndcg control**	**precision control**		**ndcg experiment**	**precision experiment**
Experiment 2	itemknn	50	100	50, 100	-2.96E-04	2.99E-04		3.33E-04	2.99E-04
Experiment 2	itemknn	100	100	100, 100	0.0066	2.99E-04		2.21E-04	2.99E-04
Experiment 2	itemknn	200	100	200, 100	0.00515	2.97E-04		-6.58E-04	2.98E-04
Experiment 2	itemknn	50	200	50, 200	-1.65E-04	1.71E-04		4.03E-04	1.71E-04
Experiment 2	itemknn	100	200	100, 200	0.00693	3.04E-04		2.81E-04	3.04E-04
Experiment 2	itemknn	200	200	200, 200	0.0047	3.03E-04		-6.10E-04	3.03E-04
Experiment 2	itemknn	50	400	50, 400	-6.23E-05	5.43E-04		4.60E-04	5.43E-04
Experiment 2	itemknn	100	400	100, 400	0.007471	4.94E-04		4.16E-04	4.94E-04
Experiment 2	itemknn	200	400	200, 400	0.00592	2.85E-04		-4.01E-04	2.85E-04
	Algorithm	**neighborhood size**	**list size**	**neighborhood size, lis**	**ndcg control**	**precision control**		**ndcg experiment**	**precision experiment**
Experiment 3	userknn	50	100	50, 100	3.82E-04	5.25E-04		5.30E-04	5.24E-04
Experiment 3	userknn	100	100	100, 100	4.00E-04	2.82E-04		5.48E-04	5.64E-04
Experiment 3	userknn	200	100	200, 100	0.00125	5.73E-04		5.48E-04	5.55E-04
Experiment 3	userknn	50	200	50, 200	9.10E-04	5.24E-04		4.95E-04	4.52E-04
Experiment 3	userknn	100	200	100, 200	4.93E-04	4.78E-04		5.21E-04	4.78E-04
Experiment 3	userknn	200	200	200, 200	-9.31E-04	5.34E-04		5.27E-04	4.96E-04
Experiment 3	userknn	50	400	50, 400	0.00362	5.25E-04		6.21E-04	3.56E-04
Experiment 3	userknn	100	400	100, 400	0.00202	4.78E-04		6.77E-04	3.85E-04
Experiment 3	userknn	200	400	200, 400	0.00246	3.92E-04		6.86E-04	3.92E-04

					Control Round #1			Experiments Round #1	
Control 1	**Algorithm**	**num factors**	**list size**	**num factors, list size**	**ndcg control**	**precision control**		**ndcg experimer**	**precision experiment**
Experiment 1	biasedmf	60	100	60, 100	3.53E-04	4.48E-06		0	0
Experiment 1	biasedmf	80	100	80, 100	4.18E-04	4.47E-06		5.06E-06	1.79E-06
Experiment 1	biasedmf	100	100	100, 100	4.19E-04	4.47E-06		0.001359	6.69E-04
Experiment 1	biasedmf	60	200	60, 200	3.03E-04	1.31E-03		0	0
Experiment 1	biasedmf	80	200	80, 200	3.53E-04	1.31E-03		1.54E-05	4.47E-06
Experiment 1	biasedmf	100	200	100, 200	3.53E-04	1.31E-03		0.001819	8.61E-04
Experiment 1	biasedmf	60	400	60, 400	3.03E-04	0.00E+00		0	0
Experiment 1	biasedmf	80	400	80, 400	3.03E-04	0.00E+00		5.57E-05	1.43E-05
Experiment 1	biasedmf	100	400	100, 400	3.03E-04	0.00E+00		0.0041176	0.0013108
	Algorithm	**neighborhood size**	**list size**	**neighborhood size, lis**	**ndcg control**	**precision control**		**ndcg experimer**	**precision experiment**
Experiment 2	itemknn	50	100	50, 100	3.64E-04	3.61E-04		-7.70E-04	3.61E-04
Experiment 2	itemknn	50	200	50, 200	4.34E-04	3.26E-04		-7.70E-04	3.61E-04
Experiment 2	itemknn	50	400	50, 400	5.57E-04	2.98E-04		-7.70E-04	3.61E-04
Experiment 2	itemknn	100	100	100, 100	3.32E-04	3.19E-04		7.50E-04	1.59E-04
Experiment 2	itemknn	100	200	100, 200	3.84E-04	3.16E-04		0.00239	3.16E-04
Experiment 2	itemknn	100	400	100, 400	5.14E-04	2.98E-04		0.00239	3.15E-04
Experiment 2	itemknn	200	100	200, 100	3.94E-04	3.65E-04		-0.001314	5.25E-04
Experiment 2	itemknn	200	200	200, 200	4.71E-04	3.58E-04		-6.80E-04	4.71E-04
Experiment 2	itemknn	200	400	200, 400	6.27E-04	3.33E-04		-2.04E-04	3.33E-04
	Algorithm	**neighborhood size**	**list size**	**neighborhood size, lis**	**ndcg control**	**precision control**		**ndcg experimer**	**precision experiment**
Experiment 3	userknn	50	100	50, 100	1.78E-04	1.20E-04		-5.61E-05	1.20E-04
Experiment 3	userknn	100	100	100, 100	1.68E-04	1.09E-04		-5.05E-05	5.46E-05
Experiment 3	userknn	200	100	200, 100	1.51E-04	1.04E-04		-5.69E-05	5.24E-04
Experiment 3	userknn	50	200	50, 200	1.05E-04	9.40E-05		-0.002185	1.20E-04
Experiment 3	userknn	100	200	100, 200	1.17E-04	9.40E-05		-0.0023	5.40E-05
Experiment 3	userknn	200	200	200, 200	1.09E-04	8.77E-05		-0.00229	5.18E-04
Experiment 3	userknn	50	400	50, 400	2.89E-04	5.04E-04		0.0013	1.20E-04
Experiment 3	userknn	100	400	100, 400	2.37E-04	5.03E-04		0.001117	9.40E-05
Experiment 3	userknn	200	400	200, 400	2.35E-04	5.00E-04		6.98E-04	5.00E-04