Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms

Martha Larson[1,2], Alessandro Zito[3], Babak Loni[2], Paolo Cremonesi[3]
[1]Radboud University, Netherlands
[2]Delft University of Technology, Netherlands
[3]Politecnico di Milano, Italy
m.larson@cs.ru.nl,zito.ales@gmail.com,b.loni@tudelft.nl,paolo.cremonesi@polimi.it

ABSTRACT

This paper states the case for the principle of minimal necessary data: If two recommender algorithms achieve the same effectiveness, the better algorithm is the one that requires less user data. Applying this principle involves carrying out training data requirements analysis, which we argue should be adopted as best practice for the development and evaluation of recommender algorithms. We take the position that responsible recommendation is recommendation that serves the people whose data it uses. To minimize the imposition on users' privacy, it is important that a recommender system does not collect or store more user information than it absolutely needs. Further, algorithms using minimal necessary data reduce training time and address the cold start problem. To illustrate the trade-off between training data volume and accuracy, we carry out a set of classic recommender system experiments. We conclude that consistently applying training data requirements analysis would represent a relatively small change in researchers' current practices, but a large step towards more responsible recommender systems.

ACM Reference format:
Martha Larson[1,2], Alessandro Zito[3], Babak Loni[2], Paolo Cremonesi[3]. 2017. Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms. In *Proceedings of Workshop on Responsible Recommendation at ACM RecSys'17, Como, Italy, 31 August 2017 (FATREC'17),* 6 pages.
https://doi.org/10.18122/B2VX12

1 INTRODUCTION

Conventionally, recommender algorithms are developed to exploit all available training data. Although there is wide-spread awareness of the downsides of such *data greed* during algorithm training and deployment, the convention stands largely unquestioned. In other words, researchers generally know that prediction performance saturates after a certain amount of data has been collected from users, and additional data only increases training times. However, this knowledge is currently not translated into best practice for the development of recommender systems algorithms.

In this paper, we state the case for the practice of analyzing training data requirements during the development and evaluation of recommender system algorithms. Such an analysis implements the principle of minimal necessary data: If two recommender algorithms achieve the same effectiveness, the better algorithm is the one that requires less user data. Our position is that any new recommender algorithm should be judged by the way in which it trades off between accuracy and amount of training data used. Beyond a certain point, additional training data will not have a meaningful effect on predictions. Effectively, the extra training data will have an "invisible" impact on user experience. We argue that pushing the collection and use of user data beyond this point should be discouraged. In short, a responsible recommender system takes no more from users than it needs to. The case for training data requirements analysis is closely related to the 2013 idea of *Differential Data Analysis* [6], which creates characterizations of which data contributes most to the accuracy of a recommender algorithm. The extended arXiv version of [6] emphasizes that data is a liability: services providers need to protect it, and they need to respond to subpoenenas. Data breaches are a serious worry for companies storing data. Considerations of privacy and data security are becoming increasingly important as Europe continues to emphasize users controlling their own personal data (cf. the EU General Data Projection Regulation[1], which goes into force in 2018).

With this paper, we build on the motivation of [6], and also echo the question, "Is all this data really necessary for making good recommendations?" We first argue for the importance of training data requirements analysis in recommender system research. Then, we report on classic experiments showing that lengthening the history-length of the training set does not necessarily improve prediction accuracy. The picture that emerges is that recommender systems have much to gain, and actually nothing to lose, in moving towards minimal necessary data.

2 BACKGROUND AND MOTIVATION

This section looks at aspects of the current state of recommender system research that motivate minimal necessary data.

Addressing the Data Greed Habit Looking at the field of recommmender system research and development as a whole, unquestioned data greed is quite surprising. We point to the work on cold-start recommendation, and in particular to [8], as evidence that researchers are well aware that after a certain saturation point more data does not necessarily translate into better performance. We suspect that data greed is simply a bad habit developed when standard, static data sets are used for evaluation. With such data sets the assumption that "more is always better" does not lead to any obvious negative consequences. On the contrary, comparison of results on standard data sets requires standardized test/training

FATREC'17, 31 August 2017, Como, Italy

DOI: 10.18122/B2VX12

[1]http://ec.europa.eu/justice/data-protection

splits. In other words, using less than all available data is actually associated with faulty methodology. Adopting training data analysis as a best practice would maintain comparability between research results, while at the same time allowing application of minimal necessary data.

Fulfilling Non-Functional Requirements Recent years have seen a push towards evaluating recommender systems with respect to not only functional, but also non-functional requirements [26]. During this time, analysis of resource use has become more common in the literature, and the development of algorithms with unnecessary computational complexity or high response time has been discouraged. Training data requirements analysis is another form of resource analysis that supports understanding of the practical usefulness of recommender algorithms in real-world settings. Seen in this way, minimal necessary data is a continuation of an existing evolution.

Ensuring User-centered Recommendation Recently, research studies have demonstrated that algorithm accuracy does not necessary play a dominant role in the reception of a recommender system by users [7, 10]. If performance improvements achieved by using more data to train a recommender system are too slight or subtle for users to notice, the additional data is adding no value, and should not be used. We understand responsible recommendation as recommendation that serves the people whose data it uses. Conscientious service of users requires formulating an explicit definition of success that characterizes the goals of the recommender system. The definition should contain a specification of the trade-off between accuracy levels and user experience. Such a definition throws a spotlight on where recommender systems are collecting, storing, and using data that is not needed. Using more data than needed imposes on users' privacy, and, collecting user data that does not serve a specific goal cannot be justified.

In sum, if the convention of data greed has no principled justification, and the recommender system community is already focusing on non-functional requirements and user experience, it is an obvious and relatively small step to focus on minimal necessary data.

3 RELATED WORK

Here, we overview previous work related to trade-offs between training data volume and recommender system prediction performance.

3.1 Analyzing Training Data Requirements

Papers analyzing training data requirements are scattered throughout the recommender system literature. In 2008, [27] evaluated the performance of algorithms on the NetFlix Prize dataset against the number of users in the training data. There is a clear saturation between 100,000-480,000 users, i.e., the algorithm does not achieve continued improvement. The plot is on log scale, and the authors are focused on what can be achieved by 0-100,000, and do not mention the saturation effect. Also in 2008, [21] analyzed the impact of the number of using ratings on news item recommendation. In 2010, [22] analyzed the number of weeks of training data on the recommendations of seminar events at a university. On the whole, we find that attention to minimal necessary data has been the exception rather than the rule.

3.2 Doing More with Less

In addition to work that looks at the impact of training data volume on specific algorithms, other work is dedicated to actually developing algorithms that do more with less. In the general machine learning literature, there is clear awareness that certain algorithms are better suited than others for performing under conditions of limited data, e.g., [11]. Here, we mention some other examples of work that we are closely connected to. Cold start is the classic case in which recommender system algorithms must be capable of doing more with less. Different sizes of datasets have been studied in order to investigate different levels of cold start [8, 9]. Further [8] shows that there is a difference between algorithms with respect to data requirements. The idea of minimal necessary data can be seen as the proposal to take the ability of algorithms designed to address cold start conditions and applying it as broadly as possible.

In [23], we touched on the privacy benefits of algorithms that do not need to store data in association with user IDs for long periods. Explicit attention to minimal necessary data will promote the development of such algorithms. We note that algorithms that use minimal personal data are useful to address news recommendation, where user IDs might be unstable or unavailable [16].

3.3 Timed-based Training Data Analysis

The closest work to the experiments presented in this paper is work on time-aware recommender systems. The survey article [4] discusses techniques that weight ratings by freshness and mentions that the more extreme version of such an approach is *time truncation*, i.e., actually dropping ratings older a specified threshold. They authors cite only two time-truncation papers. The first is [5], which demonstrates that using information near the recommendation date improves accuracy on the CAMRa 2010 Challenge. The second is [13], which reports interesting results using a time-window filtering technique intended to capture fluctuations in seasonal demand for items. Perhaps the most well-known work on time-aware recommendation is Collaborative Filtering with temporal dynamics [18]. Here, we adopt [18] as a baseline to demonstrate the effect of time truncation above and beyond time-based weighting.

4 EXPERIMENTAL SETUP

Next, we turn to a set of experiments that illustrate the trade-off between data volume and prediction accuracy using a timed-based training data requirements analysis. In this section, we describe our data sets, recommender algorithms, and analysis methodology. Our experiments support the position that this trade-off should not be considered a a tweak to be taken care of by engineers at deployment time. Rather, training data size has substantial measurable impact in common experimental set ups used by recommender system researchers. Here, we study time truncation since it is a well-established method for identifying training data that is less valuable. We emphasize that other approaches, such as sampling, are important for training data requirements analysis.

4.1 Data sets

We choose to experiment on three data sets. The data sets were chosen because of their long temporal duration, and the fact that they are widely used, which supports reproducibility. The first two,

MovieLens 10M and NetFlix were selected because they are 'classic', in the sense that they are well understood by the community. The third, a dataset from Amazon, is representative of a highly sparse recommender problem.

Basic statistics of the datasets are shown in Table 1. We briefly mention the temporal ranges and further details about each. The MovieLens 10M dataset [14] has a data range from January 1995 to January 2009 (14 years). The Netflix dataset[3] was collected between October 1998 and December 2005 (7 years). To make the dataset size manageable we randomly selected 10% of users. We observed that our sample is big enough to cover almost all the movies and that the distribution of ratings has the same shape in the sample and in the original dataset. Furthermore, the temporal window of the sample is almost as long as the original dataset.

The Amazon dataset [19] consists of ratings collected from June 1995 to July 2005 (10 years). We use only the products that belong to the four main product groups: books, DVDs, music and videos.

Dataset	#Users	#Items	#Ratings	Density
ML-10M	69.878	10.681	10M	4,47%
Netflix	480.180	17.770	100,5M	1,2%
Amazon	1.553.447	401.961	7,5M	0,001%

Table 1: Datasets statistics

4.2 Recommender framework

We use four different algorithms to train our models. The experiments are implemented using WrapRec [20], an open source evaluation framework for recommender systems. The experiment were run on a machine with 16 CPU cores with clock speed of 2.3 GHz and 16 GB of memory. The following algorithms are used in this work where the first three are used for rating prediction and the last one is used for the top-N ranking task.

Biased Matrix Factorization (BMF): This method [18] is the most widely-used model-based algorithm for rating prediction problems. This method is the standard Matrix Factorization model with user, item and global biases. In this work, we used the MyMediaLite [12] implementation of BMF with its default hyper-parameter values. The optimization algorithm is Stochastic Gradient Descent (SGD) with a learning rate of 0.01. The latent factors are initialized with a zero-mean normal distribution with standard deviation of 0.1. The number of latent factors, however, is varied. Our experiments demonstrate the effect of latent factors.

Factorization Machines: Factorization Machines [24] are state-of-the-art models for rating prediction problems. In this work, we used the more advanced optimization method of Markov Chain Monte Carlo (MCMC), that is implemented in LibFm [24]. The only hyper-parameter of the MCMC algorithm, i.e., the standard deviation of the initializer distribution, is set to 0.1, the default value in the LibFm implementation [24].

Time-Aware Factor Model: This method [17] is also a latent factor model for rating prediction problems. The temporal effect of user preferences is modeled with a time-dependent bias function. This method yielded top performance in the Netflix prize. The hyper-parameters are the default values of the MyMediaLite implementation of Time-Aware model.

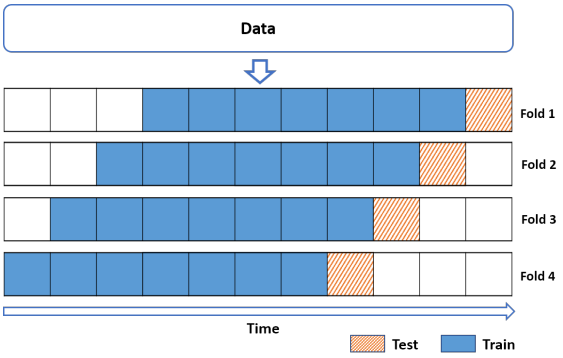

Figure 1: An Overview of the sliding process.

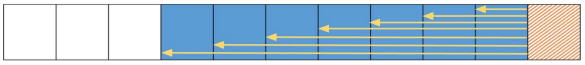

Figure 2: Representation of sliding window for one fold.

Bayesian Personalized Ranking (BPR): This method [25] is an state-of-the-art method for ranking problems where the learning involves optimization for ranking. Since this method is designed for datasets with unary positive-only feedback, we consider the ratings above user average rating as a positive feedback. BPR uses SGD for optimization. The learning rate is set to 0.05 and the standard deviation of the initializer is set to 0.1.

4.3 Sliding-window Cross-validation

Our experiments use *sliding window cross-validation*, which allows us to maintain the temporal ordering of the data (also referred to as 'forward chaining'). We start by partitioning the data into 11 temporal segments. Each fold of the cross-validation consists of a data window that is split into test and training data. The test data consists of the temporally most recent segment. To create multiple folds, the data window is slid backwards in time by one segment, such that the test data is different for each fold. The sliding process is illustrated in Figure 1. We vary the size of the training dataset by increasing its *history length*, i.e., the length of time that the training dataset extends into the past. We test seven history lengths, indicated by the arrows in Figure 2. Each history length is created by adding one segment to the next-shortest history length. Since our initial split created 11 segments, increasing the history length by one segment means increasing the training data by 10%.

Our data partitioning method makes it possible to validate the results using a training set up to the length of seven segments preceding the test set in each fold. We could have extended the training set with additional segments, but, as we will see in the next section, seven are sufficient to illustrate the phenomenon of saturation that motivates our research. We also noted that the different datasets have different trends in density development as the history length of the training set grows longer. Although we do not measure it formally here, this gives us confidence that the effects we observe are not caused by density trends.

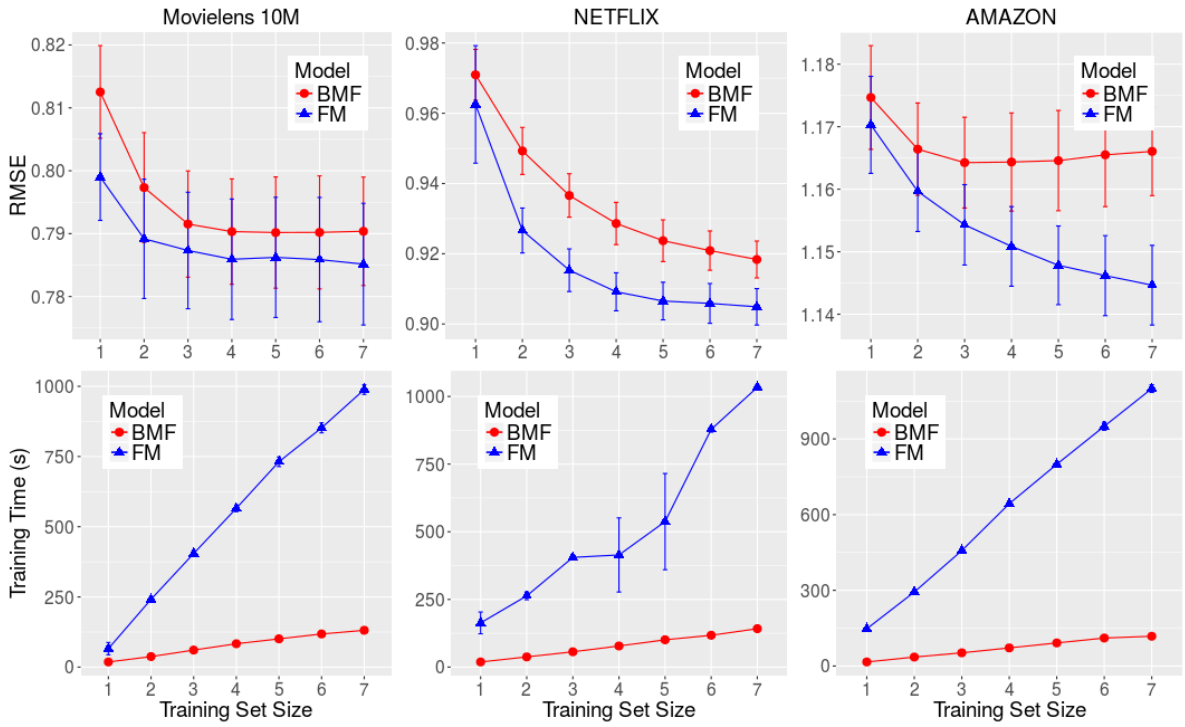

Figure 3: Empirical comparison of the performance and the training time of the two methods of BMF and FM on our three datasets with respect to the training set size (history-length of training set measured in segments).

5 EXPERIMENTS AND RESULTS

We perform three different experiments in order to observe the effect of increasing the history length based on different models.

5.1 Impact of Training Data History-Length

In this experiment, we apply our recommender algorithms while increasing the training set size by extending the history-length of the training window (see Figure 1). The experiment has a relatively a naïve formulation: we simply observe what happens when we apply time-truncation when training classic recommender system algorithms out of the box. For this experiment, we used Biased Matrix Factorization (BMF) and Factorization Machines (FM). The performance of the models are evaluated using the Root Mean Squared Error (RMSE) metric. We also measured the training time of the two models based on different number of segments. Both models were trained with 30 iterations and 10 latent factors. As can be seen in Figure 3, as the history length of the training dataset increases, a certain saturation effect can be observed with all three datasets. At the same time, the training time increases linearly with the history length. The saturation is quite dramatic with MovieLens 10M. However, in all cases there is a clear fall off in the added value of extra data once the training set reaches a certain size. These results show that a large reduction of training data requires a relatively small trade-off of prediction accuracy.

Next, we dive more deeply to investigate whether the choice of the number of latent factors explains the saturation effect. The left column of Figure 4 shows the influence of the number of latent factors. For this experiment, we use the BMF model and two data sets, MovieLens 10M and Netflix. The figures confirm that the saturation effect dominates the impact of the choice in the number of factors. In other words, increasing the number of latent factors does not necessarily cause the model to benefit from more data.

5.2 Exploiting Temporal Dynamics

In this section, we look more closely at temporal effects. The purpose of this experiment is to eliminate the possibility that the observations in the previous section can be attributed to time-truncation acting as a primitive method for incorporating temporal dynamics into a model. We use the time-aware factor model, introduced in [17], where the temporal aspect of user preferences are exploited using a time-dependent bias function. We use same procedure as in previous experiments to increase the size of the training set. The middle column of Figure 4 reports results on the MovieLens 10M and Netflix datasets. The fact that we find saturation effects using an algorithm that models temporal dynamics, suggests that time-truncation of training data should be used in addition to exploiting temporal dynamics.

5.3 Top-N Recommendations

Next, we turn to Top-N Recommendation and explore the effect of training set size on a learning-to-rank method. We used Bayesian Personalized Ranking (BPR) [25] to train our model, and report results in terms of recall at three different cut-off levels N. We used same number of iterations and latent factors as the naïve experiment (Section 5.1). To calculate recall, we apply a procedure

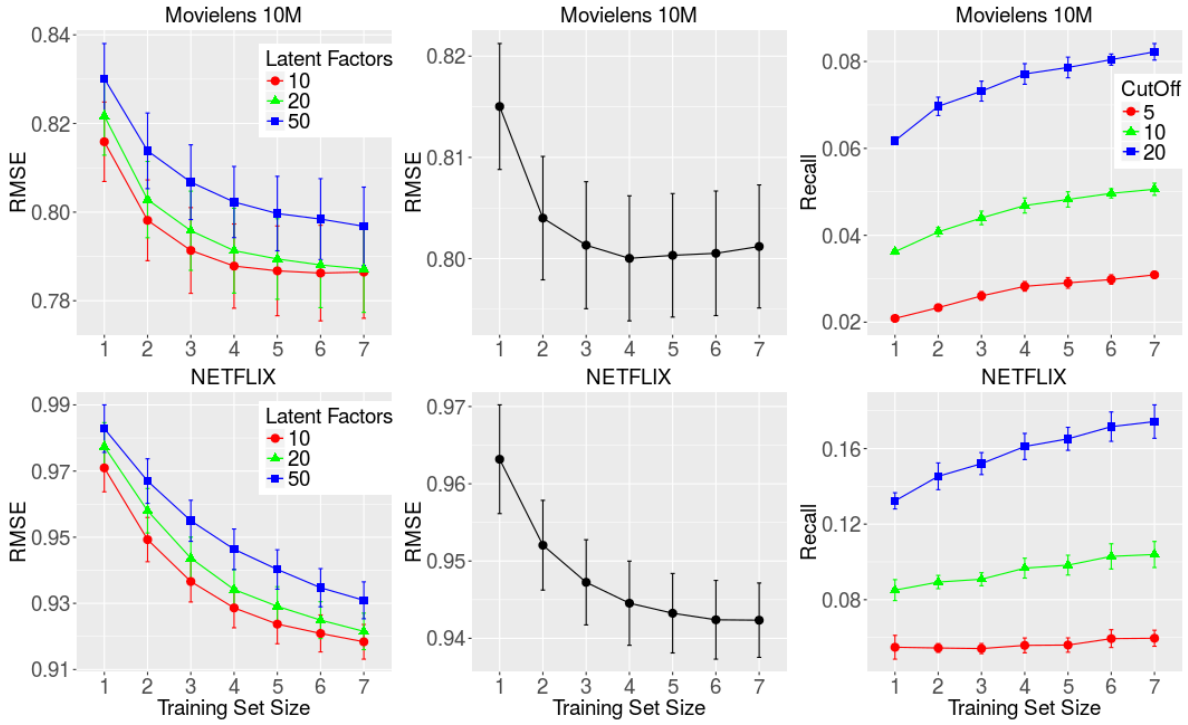

Figure 4: The effect of number of latent factors (left column), saturation effect on the time-aware latent factor model (middle column) and the effect of history-length size on a ranking model (right column)

known as *one-plus-random* [9]: For each test point, 1000 random items that are not rated by the user are sampled and the target item is added. This set of 1001 items are ranked using the trained model. If the target item appears in one of the top-N positions of the ranked list, we count that test point as having achieved a hit. As can be seen in the right column of Figure 4, a smaller training set benefits more from additional data than a larger training set. The results are in this way comparable to what we found in our rating prediction experiments in Sections 5.1 and 5.2. We also calculated performance with respect to Mean Reciprocal Rank (MRR), which is not depicted here for space reasons. For both recall and MRR, we observe diminishing returns effects.

5.4 Discussion

Our experiments illustrate saturation effects as the size of the training dataset increases, but also reveal aspects of data reduction that are not yet thoroughly understood. Following the idea of Differential Data Analysis [6], we would like to have insight into when and why we observe saturation, i.e., diminished returns from additional data. Users in the test set are likely to be represented in the segments temporarily closest to the test set. Ideally, we would like to understand how much of the effect can be attributed to pruning inactive users, and how much is related to taste/item shift, or its opposite, information redundancy. A detailed understanding of these effects would make it possible to designs schemes for data collection and retention that have minimal impact on user privacy. For example, if inactive users are no longer contributing to improving predictions, their data should simply be deleted.

When to apply time-truncation is not easy to predict. During our exploratory experiments, we found that prediction accuracy using the smaller data set MovieLens 1M, with 1M ratings and a time span of 3 years (leading to much shorter segments than with ML 10M), does not saturate. This effect suggests that further investigation is needed into the relationship between training data history length and performance for shorter history lengths. We believe, however, that very recent history is very valuable. For example, [15] demonstrates the value of adding information on the most recent history items that the user has interacted with to the prediction for the current item using ML 1M. To better support privacy, we would like to give further consideration to user-specific data dropping, i.e., truncating specific user histories when certain conditions hold. For example, future research could focus on optimizing algorithms that exploit only the very most recent interactions of the user, and delete older interactions. Our initial experiments in this area revealed that it is not trivial. User truncation, could, however, ultimately lead to recommender systems that are not only privacy-sensitive, but also more even handed, and do not favor active users.

6 CONCLUSION AND OUTLOOK

In this paper, we have made a case for recommender systems research to adopt training data requirements analysis as a best practice when developing and evaluating new algorithms. Specifically, researchers developing a recommender system should explicitly analyze the trade-off between the amount of data that the system requires, and the performance of the system. When the improvement in prediction performance becomes negligible, more data

should not be used. If two algorithms achieve the same prediction performance, the algorithm that uses less data should be preferred.

We have presented experimental evidence that trade-offs between objective metrics and the amount of data used deserve increased attention in recommender research. We argue that the recommender system community is well aware of results of this sort, and implicitly already understands the disadvantages of *data greed* and also of the benefits of doing more with less. Carrying out an analysis that demonstrates that an algorithm uses minimal necessary data represents a straightforward application of this awareness. A relatively small shift in research practices represents a large step towards more responsible recommender systems.

As mentioned in the introduction, there is a connection between algorithms that determine the usefulness of data, and user privacy [6]. Obfuscation can protect users and does not necessarily impact recommender performance. Techniques involving obfuscation have been used to anonymize data sets, enabling their release for research purposes, as in [2]. Moving forward, we feel that the idea of minimal necessary data can provide an entry point for researchers in becoming interested in developing obfuscation techniques.

We close with a warning about adopting the position that 'someone else is doing it'. A metareviewer of a previous version of this paper commented, "How to obtain good recommendations from a minimal amount of data is an interesting problem. At the same time, the idea the the predictive modeling performance improves as the training data grows but eventually tends to level off has been well established in machine learning and is quite well understood (i.e., the concept of learning curves is machine learning reflects exactly that)." We agree with this statement. A recent article in The Economist [1] quotes Google's chief economist commenting on the "decreasing returns to scale" of data. However, we are left wondering why a well-understood idea in machine learning remains apparently so severely underexploited in recommender system research. When it comes to questioning in the assumption of data greed in recommender systems, it appears that 'someone else is *not* doing it', and that more effort needs to be made to move the community towards minimal necessary data.

Writing this paper gave us a direct experience of how easy it is to overlook the wider implications of data use. A reviewer pointed out that the NetFlix data set, used here, has been removed from public availability, citing its deanonymizability. Ironically, this consideration escaped us during our experimentation. We must count ourselves among the researchers who face the challenge of understanding the full implications of a commitment to best practices including minimal necessary data.

In sum, we argue that recommender system research must look at how much data is really necessary to accomplish a given recommendation task. However, we find that moving towards minimal necessary data represents a relatively small change in current practices. Recommender system researchers have acquired years of experience addressing cold start. It is time to shift our perspective to realize that cold start is not only a problem, it is also a solution.

7 ACKNOWLEDGEMENTS

This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

REFERENCES

[1] The data economy: Fuel of the future. *The Economist, 6–12 May*, pages 13–16, 2017.
[2] F. Abel, A. Benczúr, D. Kohlsdorf, M. Larson, and R. Pálovics. RecSys Challenge 2016: Job recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 425–426, 2016.
[3] J. Bennett, S. Lanning, et al. The Netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.
[4] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. 24(1-2):67–119, 2014.
[5] P. G. Campos, A. Bellogín, F. Díez, and J. E. Chavarriaga. Simple time-biased KNN-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 20–23, 2010.
[6] R. Chow, H. Jin, B. Knijnenburg, and G. Saldamli. Differential data analysis for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 323–326, 2013.
[7] P. Cremonesi, F. Garzotto, and R. Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *ACM Trans. Interact. Intell. Syst.*, 2(2):11:1–11:41, June 2012.
[8] P. Cremonesi and R. Turrin. Analysis of cold-start recommendations in IPTV systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 233–236, 2009.
[9] P. Cremonesi and R. Turrin. Time-evolution of IPTV recommender systems. In *Proceedings of the 8th European Conference on Interactive TV and Video*, EuroITV '10, pages 105–114, 2010.
[10] M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, 2014.
[11] G. Forman and I. Cohen. *Learning from Little: Comparison of Classifiers Given Little Training*, pages 161–172. Springer Berlin Heidelberg, 2004.
[12] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: a free recommender system library. RecSys '11. ACM, 2011.
[13] S. Gordea and M. Zanker. Time filtering for better recommendations with small and sparse rating matrices. In *Proceedings of the 8th International Conference on Web Information Systems Engineering*, WISE '07.
[14] F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.
[15] A. Karatzoglou. Collaborative temporal order modeling. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 313–316, 2011.
[16] B. Kille, A. Lommatzsch, F. Hopfgartner, M. Larson, and A. P. de Vries. A stream-based resource for multi-dimensional evaluation of recommender algorithms. SIGIR '17 to appear, 2017.
[17] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, Apr. 2010.
[18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.
[19] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), 2007.
[20] B. Loni and A. Said. Wraprec: An easy extension of recommender system libraries. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 377–378, 2014.
[21] V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon. Evaluation of an ontology-content based filtering method for a personalized newspaper. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 91–98, 2008.
[22] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. Collaborative future event recommendation. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 819–828, 2010.
[23] R. Pagano, P. Cremonesi, M. Larson, B. Hidasi, D. Tikk, A. Karatzoglou, and M. Quadrana. The contextual turn: From context-aware to context-driven recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, 2016.
[24] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3), May 2012.
[25] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.
[26] A. Said, D. Tikk, K. Stumpf, Y. Shi, M. A. Larson, and P. Cremonesi. Recommender systems evaluation: A 3D benchmark. In *ACM RecSys 2012 Workshop on Recommendation Utility Evaluation: Beyond RMSE*, 2012.
[27] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, NETFLIX '08, pages 6:1–6:8, 2008.