Do Common Educational Datasets Contain Static Information? A Statistical Study

Théo Barollet, Florent Bouchez-Tichadou, Fabrice Rastello

Do Common Educational Datasets Contain Static Information? A Statistical Study

Théo Barollet
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
theo.barollet@inria.fr

Florent Bouchez Tichadou
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
florent.bouchez-tichadou@imag.fr

Fabrice Rastello
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble France
fabrice.rastello@inria.fr

ABSTRACT

In Intelligent Tutoring Systems (ITS), methods to choose the next exercise for a student are inspired from generic recommender systems, used, for instance, in online shopping or multimedia recommendation. As such, collaborative filtering, especially matrix factorization, is often included as a part of recommendation algorithms in ITS.

One notable difference in ITS is the rapid evolution of users, who improve their performance, as opposed to multimedia recommendation where preferences are more static. This raises the following question: how reliably can we use matrix factorization, a tool tried and tested in a static environment, in a context where timelines seem to be of importance.

In this article we tried to quantify empirically how much information can be extracted statically from datasets in education versus datasets in multimedia, as the quality of such information is critical to be able to accurately make predictions and recommendations. We found that educational datasets contain less static information compared to multimedia datasets, to the extent that vectors of higher dimensions only marginally increase the precision of the matrix factorization compared to a 1-dimensional characterization. These results show that educational datasets must be used with time information, and warn against the dangers of directly trying to use existing algorithms developed for static datasets.

Keywords

Knowledge tracing, Recommender systems, collaborative filtering, static models, matrix factorization

1. INTRODUCTION

Knowledge tracing tries to model the knowledge of students as they learn, and is a key component of Intelligent Tutoring Systems (ITS). In such systems, the aim is to recommend resources, such as exercises (or "problems"), to students in the most effective way, that is, to recommend resources which correspond to their learning needs. These resources can be of various forms, but in this article we focus solely on recommending new problems for the student to solve. In order to perform any recommendation, we believe that we should be able to predict the outcome of one particular student trying to solve one particular problem; we call this a *(student, problem)* pair. Ideally, we have perfect information for all such *(student, problem)* pairs, whether that information is actual (extracted from observation) or deduced (based on previous observation). This would allow us, for instance, to skip problems that are predicted as being too easy or too hard for a student.

Each *(student, problem)* pair could reflect a level of "difficulty" indicating the student's proficiency. In such a system, one would derive existing difficulty levels from known interactions, for instance through how much time was required for a student to solve a problem, or how many attempts it took to successfully solve it. A "good" system would then predict difficulty levels for interactions that did not happen, possibly with a confidence measure of the outcome prediction. It would also provide an understanding of the structure of the problem set. For example, it would enable the recognition of problems that train similar skills or use similar knowledge, without relying on expert knowledge components that require human expertise.

Historically, the field of knowledge tracing has been independent of recommender systems. With expert knowledge components, one can explicitly measure student proficiency with simple models like Item Response Theory and Bayesian Knowledge Tracing [2]. Using data mining on large datasets, it is possible to relax the knowledge components to be latent features that do not require human experts to partition the domain into explicit student skills [11]. Techniques in educational data mining are inspired from techniques used in collaborative filtering [1], such as factorization methods [20, 18], but also from techniques used in deep learning such as deep knowledge tracing [11, 19, 20, 13].

In this article, we will focus on matrix factorization methods. These are traditionally used in contexts where the available data is not very sensitive to time, for instance movie tastes and shopping habits. In contrast, students learn each time

they practice and should normally improve with time, so it would make sense to take history into account when analyzing datasets, making predictions and doing structural studies. However, we do not know how much impact this has on the results. The question that we would like to raise here is whether taking history into account is that important or if it is still possible to make good predictions when considering datasets as timeless. Recommending a good problem in terms of teaching is not an easy task, but it is even more difficult when we cannot reliably predict whether the student will succeed or fail, and how long it will take them to do so. In the rest of this article, we will study how the matrix factorization algorithm behaves in three datasets from the educational data mining community compared to one dataset from the traditional collaborative filtering community. We will often deliberately *leave out* chronological information in the educational datasets to see how much information can still be extracted, compared to a traditional dataset.

This article makes two main claims:

- Educational datasets contain much less static information than usual datasets found in multimedia recommendation. Hence, treating educational datasets without any dynamic method should be avoided;
- The little static information they contain amounts to a one-dimensional value per student or problem.

We also propose in Section 3 a pre-processing procedure for educational datasets meant to facilitate prediction, even though we could not find any variable that is accurately predicted among all tested datasets, as well as a filtering procedure to try to find clusters among students and problems that are particularly accurately predicted in Section 5.

2. RELATED WORK

2.1 Data Pre-processing

It is sometimes necessary or advantageous to perform preprocessing of the data before trying to extract information. For example, it was possible to improve the classification error in the MNIST database from 12% to 8%, keeping the same linear classifier, by using deskewing pre-processing [8]. Regarding our knowledge tracing problem, many corrections to the ASSISTment dataset [4] are proposed by Xiong et al. [19]. They are now included in the public dataset that we use later. We also propose some more pre-processing in Section 3.

2.2 Matrix Factorization

Matrix factorization (MF) is a widely used technique in recommender systems, as illustrated by its extgensive usage in the 2009 Netflix Prize Competition [7]. We consider a set U of N *users*, a set I of M *items*, and a set of *ratings* R. These sets are usually given as *records* $(u, i, r_{u,i})$, representing how much $(r_{u,i})$ a given user u likes item i. From these we can build a sparse rating matrix $X \in \mathbf{R}^{N \times M}$. The goal of matrix factorization is to find two matrices $W \in \mathbf{R}^{N \times k}$ and $H \in \mathbf{R}^{M \times k}$ (usually with low rank $k \ll N, M$) such that X is close to $WH^\intercal$. This is an optimization problem written as:

$$\underset{\substack{W \in \mathbf{R}^{N \times k} \\ H \in \mathbf{R}^{M \times k}}}{\operatorname{argmin}} \sum_{(i,j) \in \Omega} (X_{ij} - w_i h_j^\intercal)^2 + \lambda(\|W\|_F^2 + \|H\|_F^2) \quad (1)$$

Where λ is a regularization meta-parameter and $\|.\|_F^2$ is the Frobenius norm [21]. Equation 1 may vary in regularization terms (bias, sparsity penalty...) and can incorporate a loss function between X_{ij} and $w_i h_j^\intercal$. We can now estimate unknown ratings within the product $WH^\intercal$. In other words, we look for signatures for users and items in the same latent space of dimension k (i.e., vectors of rank k), such that the outcome of the user rating an item is close to the dot product of these signatures.

This optimization problem is non-convex in general, but different methods exist [7]. The Alternating Least Square (ALS) method is the most popular method as it converges better than the Stochastic Gradient Descent (SGD) method due to non-convexity. When large-scale data is needed, as ALS is not easy to parallelize, and Coordinate Descent is preferred [6, 21]. In a knowledge tracing setting, users are *students* and items are *problems* [20, 16, 18]. Problems are usually split into smaller components that are the *problem steps*. We will see in subsection 3.1 why we recommend a first regrouping pass in order to work with whole problems.

2.3 Cold Start Problem and Online Settings

The *cold start* problem is a typical problem in recommender systems that corresponds to the initial phase of a "nude" system (no data collected yet). The lack of data makes the prediction accuracy unreliable at that early stage. MF techniques are not designed to tackle the cold start problem but, some extensions seek to solve it partially [22, 10]. As we are not focused on prediction accuracy, we will not consider these extensions in this article but we will try to evaluate when the cold start problem ends in Section 4, that is, when there is enough data for MF to start giving results. Trivedi et al. [17] try to solve a cold start problem in an ITS environment with spectral clustering to help refine raw prediction, but they work on the raw features of datasets without student or item signatures.

Even after the cold start, the system usually benefits from new data in general. This is referred to as online recommendation, and MF is widely studied in such a context [5, 9, 12]. These works consider extremely large datasets, about the order of millions of users and items, but it is still feasible to redo a factorization after adding a few elements, as we will see for instance in Section 4 and 5.

3. FLAT PREDICTION AND AGGREGATION

In this section we studied matrix factorization (MF) on four different datasets, and found that not all datasets were directly usable without some pre-processing, compared to classical datasets. We use the basic version (L2 regularization) of Equation 1 with a fixed rank of 20 (apart from the last section where we measure the impact of rank variation). We use coordinate descent for the optimization [21] because some experiments in sections 4, 5, and 6 require numerous factorizations.

3.1 Educational Datasets and Pre-processing

We will use three common educational datasets for the rest of the article: Algebra I 2006–2007 [14], Bridge to Algebra I 2006–2007 [15] (both of these come from the Cognitive Tutor problem set) and ASSISTment09 (we use the corrected

Table 1: Raw data sets overview

Data set	Users	Problems	Steps	Steps occurring once	Mean samples per step	Samples
Algebra I 2006-2007	1338	5644	418 060	314 198	5.4	2 270 384
Bridge to Algebra I 2006-2007	1146	14 787	202 672	46 935	18.1	3 679 199
ASSISTment09	4217	17725	26 688	3123	13.0	346 660
ML-1M	6040	3706	N/A	N/A	269.9	1 000 209

Table 2: Preprocessed data sets overview

Data set	Users	Problems	Samples	Density	Mean samples per problem	Success percentage
Algebra I 2006-2007	1147	3111	152 709	0.043	49.1	0.79
Bridge to Algebra I 2006-2007	1068	8736	235 147	0.025	26.9	0.91
ASSISTment09	2025	12587	238 746	0.009	19.0	0.98
ML-1M	6040	3706	1 000 209	0.045	269.9	N/A

and collapsed version of the dataset) [4]. All three datasets record scaffolding problem statistics (also called *steps* in Cognitive Tutor datasets – we will use both terms here). For each record (also named *sample*), we extract:

1. A student and a *main problem* ID;
2. A *scaffolding problem* ID;
3. A *timestamp* when a student starts a step and the *duration* to complete the step;
4. If the student succeeded at his first attempt: *Correct-First-Attempt* (CFA);
5. The number of *hints* and *errors* of the student for this step.

To our surprise, these datasets are not very usable without pre-processing in comparison with well-established recommender system datasets like the MovieLens dataset. We will compare most experiments with the ML-1M version of this dataset, which will act as a "control" dataset: multimedia recommendation datasets being the canonical use of matrix factorization for recommender systems. The two main reasons for this poor usability that we try to mitigate with pre-processing are the following:

- The notion of scaffolding problem is not standardized between the datasets and is hard to use as is. Some of them are optional, which makes the number of steps for a main problem vary between students. The step order may also change between users, which makes matching between users more difficult at the problem level.
- There is no guarantee of the minimum number of occurrences for a student or a step. Moreover, many steps are done by a single student across the whole dataset, as seen in Table 1 (especially for the Algebra I dataset, where steps can be generated for a student from a template, and are thus unique. These constitute up to 3 quarters of the steps).

Our first pre-processing pass, which is motivated by the very low number of samples per step on average, corresponds to aggregating all the steps of a common main *(student/problem)* pair together. Aggregating timestamps and durations is straightforward (the beginning of a problem is the beginning of the first step and the total duration is the sum of the steps' durations). To aggregate "Correct-First-Attempt" we take the mean across a *(student/problem)* pair so we obtain a floating point value between 0 and 1 instead of a boolean value.

Simply aggregating hint and error counts by summing them is not satisfactory because ultimately we want to have an idea of how much a student struggled on a problem. Summing these quantities is not sufficient to access some basic information such as "*Has the given student reached the end of the problem or given up?*" This information is not provided in the datasets, so we had to build a proxy variable. To answer this question, we need to know, for a given problem, the number of basic steps it decomposes into. To find this quantity, which we call the *problem size*, we counted for each problem/student pair the number of samples. For a student who succeeded (possibly with hints and intermediate mistakes), the problem size and this number should match. We assumed that for a given problem, the most represented number (among all students) was the actual problem size. We believe that this high representativeness comes from the fact that the ITS providing the datasets give enough hints for most students to reach the end of the problem before giving up. This makes the number of hints and errors valuable information to measure the difficulty of a problem for a given student.

Once we have a boolean proxy indicating success by reaching the end of a problem, we can derive two variables: reaching the end without errors and reaching the end without hints. We can also build a difficulty variable to aggregate the hint and error counts: we sum the two counts with a 0.5 coefficient for hints. We represent failure by assigning a difficulty value of twice the maximum value.

After aggregation, we have six variables (called *target variables* or simply *targets* from now on) of interest for each student/problem interaction: *duration* (0-1 scale value), *difficulty* (0-1 scale value), *correct-first-attempt* (0-1 scale value) and *success-reached* (boolean value), *success-no-error* (boolean value), *success-no-hint* (boolean value). The first three are normalized per problem so that for each problem, the "worst" student gets a value of 0, and the best one a value of 1 (giving rise to what we called above a 0-1 scale value). ML-1M has a single target which is the movie rating (also normalized for comparison). After aggregation, we filter out users who have done fewer than 20 problems and problems that are done fewer than 5 times (same threshold than for ML-1M). Table 2 shows the size of the datasets after pre-processing.

3.2 Influence of Aggregation on Datasets

Figures 1, 2, and 3 report the ability of a factorization to accurately model the different target variables on the four

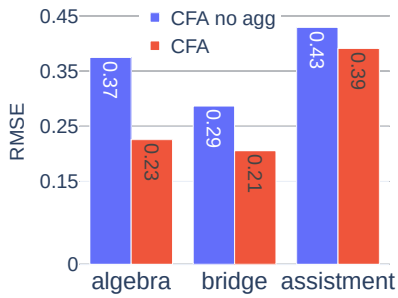


Figure 1: RMSE between no aggregation and aggregation for correct first attempt

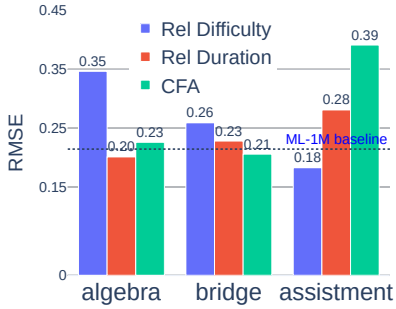


Figure 2: RMSE for duration, correct first attempt and difficulty

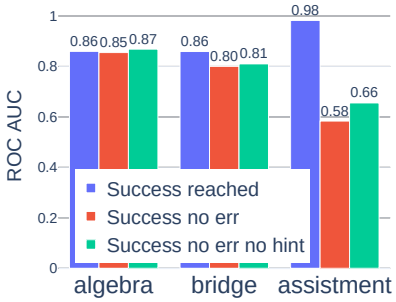


Figure 3: ROC AUC for success and derivatives (higher is better)

datasets and compare the effect of aggregation. For 0-1 scale variables we use the root mean square error (RMSE – the smaller the better) as the error metric, and for boolean variables we use Receiving Operating Curve Area Under Curve (ROC AUC – the closer to 1 the better). For ML-1M, the only possible target variable is the movie rating. We report it in Figure 2 as a horizontal line.

In Figure 1, we see that the aggregation and filter procedure improve the prediction quality of the Cognitive Tutor datasets in a notable way, but only by a small margin for the ASSISTment dataset. We believe that if the pre-processing removed about one third of the problems and half of the students, the density would still be very low compared to the others. In all the remaining experiments, we will use the aggregated versions of the datasets.

It is hard to find any trend regarding the RMSE differences in Figure 2. Variations seem to indicate that different datasets favor different target variables. MF can have about the same prediction capability for educational datasets and multimedia datasets if the target variables are chosen carefully, which suggests that situations call for pre-analyses in order to select the target variable which will be the most accurately predicted.

In Figure 3 we can see that accuracy on success classification is reasonably good. However, we cannot explain the difference in ASSISTment between success-reached and the two other success target variables. This might stem from the aggregation procedure that relies on approximated methods to obtain the number of steps in a problem. We will not do any further experiments on these target variables (which are boolean), as they are barely comparable with the 0-1 scale variable of the ML-1M dataset.

4. ONLINE PREDICTION

In this section, we will try to evaluate the point at which there is enough information to predict reasonably well with MF techniques. This allows the system to stop using whatever bootstrapping technique it was using to solve the cold start problem.

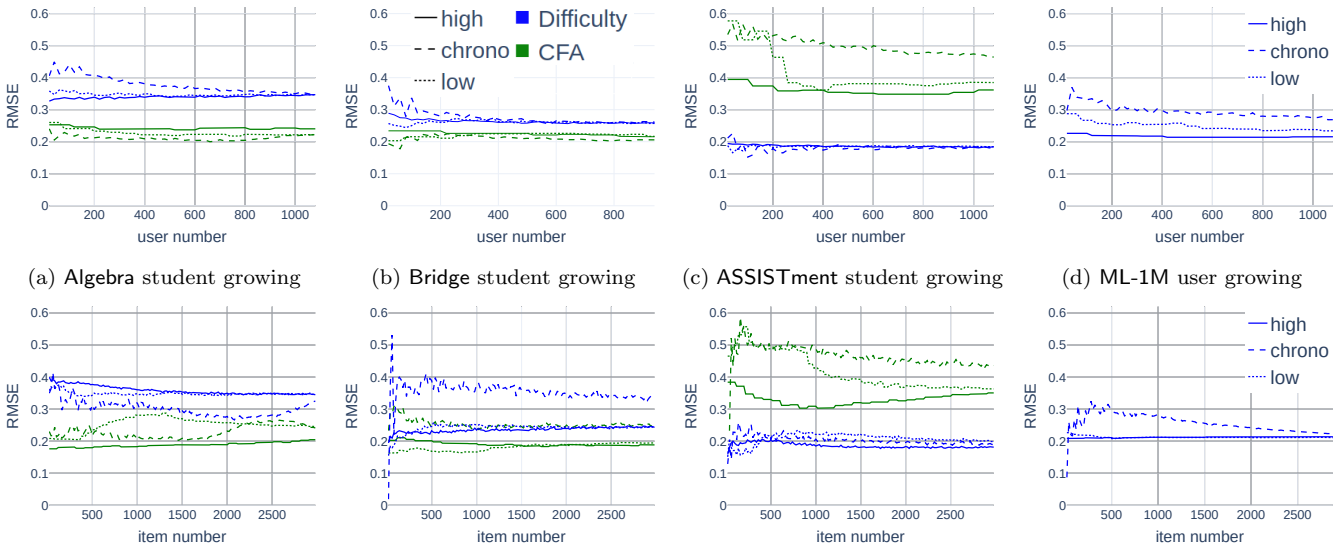
To evaluate this we start by getting either the full student set (and no problem) or the full problem set (and no student). We then progressively add new problems in the first case and new students in the second case, adding 20 new elements at each iteration. At each iteration we redo a full factorization and evaluation as if the system was complete.

We independently measure RMSE of correct-first-attempt and difficulty variables. To evaluate whether the order in which new elements are added makes a difference or not, we considered three orders: (i) elements sorted by their number of occurrences either in decreasing (*high* density first); (ii) or in increasing (*low* density first) order; (iii) and following the chronological order (*chrono*). Only the chronological order makes sense in an online context, but we still use the number of occurrence orders to evaluate whether or not we benefit from a higher density.

We report in Figure 4 the results of this experiment. We can see that for three out of four datasets (not ASSISTment), adding elements by highest density makes the system converge really fast (about 200 elements for Bridge), which was to be expected as those elements carry the most information. For all datasets, adding elements by lowest density, as we might expect, makes the system converge really slowly. We believe that the extremely accurate prediction on some of the curves for the first few iterations of the growing process is due to overfitting (recall that the factorization uses a rank of $k = 20$ in those experiments).

Still, there are some artifacts to these results. In Figure 4e, the previous claims are reversed for difficulty target. Maybe this is a hint that this aggregated variable may not be robust enough on all systems. Our advice is to systematically test target variables on a system to make sure that the ones we choose are consistent and can be trusted.

Finally, we do not observe any "dramatic" drop of the RMSE in curves representing the chronological order that we could clearly label as the "cold start" (although it sometimes takes a few "adds" to stabilize). Of course, the highest accuracy is obtained whenever all the data is used, but this suggests that MF accuracy starts to get close to the maximum early in the process. However, bear in mind that we only evaluated our ability to model existing data (we evaluate on the matrix we factorize), but did not evaluate our ability to predict (by evaluating on the remaining, not factorized, part of the matrix).



(a) Algebra student growing (b) Bridge student growing (c) ASSISTment student growing (d) ML-1M user growing

(e) Algebra problem growing (f) Bridge problem growing (g) ASSISTment problem growing (h) ML-1M item growing

Figure 4: RMSE evolution for student and problem set growing on the four data sets
If not specified the legend is the same as (b).

5. STUDENT OR PROBLEM PREDICTION KERNELS

In this section we search for a subset of students and problems where the prediction is more accurate than on the rest of the dataset. Having such a subset can be of interest in various ways: further analysis like signature clustering might work better on a subset with high accuracy prediction, or this can be a first step towards building a confidence measure for new predictions using a similarity measure with this accurate subset.

Note that we briefly tried some clustering algorithms on the student and problem signatures given by MF, but they were not promising. We will explain in Section 6 how the signatures we can obtain with MF may not be appropriate to such a study.

5.1 Iterative Filtering

We describe an iterative procedure to filter students and problems that have the least accurate predictions.

We alternately remove students and problems: at each iteration we remove the 8% of the considered set that are the least accurately predicted in terms of RMSE (or 15 elements if 8% is lower than 15). We report in Figure 5 and Figure 6 the evolution of the density of the rating matrix and RMSE for the difficulty target variable.

Figure 5 presents the variations in density as we progressively remove students and problems. Interestingly, removing items usually increases the density while removing students decreases it in the three educational datasets, meaning that the students that solved many problems are viewed as "problematic" by the system. This behavior is not observed in the reference ML-1M dataset. Figure 6 confirms the tendency that removing students in the beginning tends to improve prediction accuracy. This result is disturbing as it means that, for the educational datasets, MF prefers less dense matrices with regard to the users, i.e., less information for a given student. This suggests that MF performs best when a problem was done by many students, but when the students have done few problems. What is interesting here is that this scenario is the one that resembles most closely the ML-1M dataset: by having students that did fewer problems, we are indeed eliminating students that likely *progressed* during the experiment, hence whose behavior cannot be represented by a single vector across all their interactions. This is a first solid hint that MF alone does not seem suited to educational datasets, as it shuns chronological subtleties.

6. INFLUENCE OF RANK VARIATION

In this section we repeat the experiments from previous sections with different rank values. In addition to rank $k = 20$ that we already measured, we use rank 5 and a rank of 1. We deliberately choose a rank of 1 to mimic a Whole History Rating (WHR) [3]. Even though it is not an exact correspondence, we believe that the information extracted by a MF with rank 1 can also be extracted by a WHR.

We see in Figures 7, 8, 9 and 10 a clear difference between the educational datasets and ML-1M regarding the influence of ranks. This benefit from rank increase agrees with the intensive use of MF techniques in multimedia recommender systems. However, the benefit of such an increase for educational datasets is almost negligible. This is particularly apparent in Figure 10, where the ML-1M RMSE curves get lower with increasing rank while all other curves are nearly indistinguishable by rank. This shows that, when the chronological information is not used, vectors of size 5 or 20 do not improve accuracy compared to a simple vector of size 1, i.e., a single float. This suggests that we cannot do better than assign a single number to problems and students, which could be interpreted as having a "difficulty"

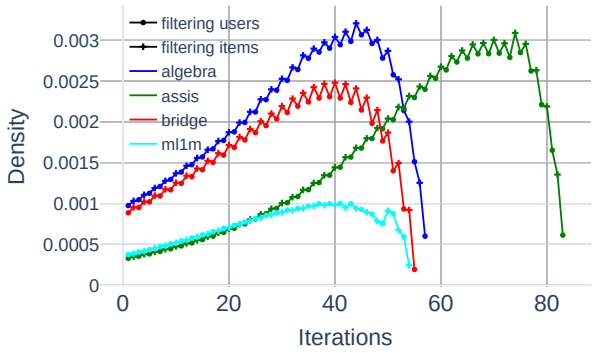


Figure 5: Evolution of density during filtering with target **Difficulty**

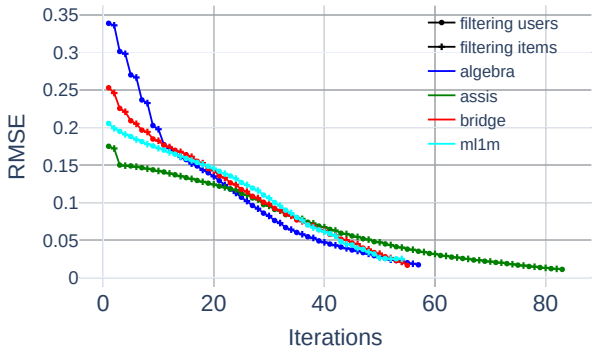


Figure 6: Evolution of RMSE during filtering with target **Difficulty**

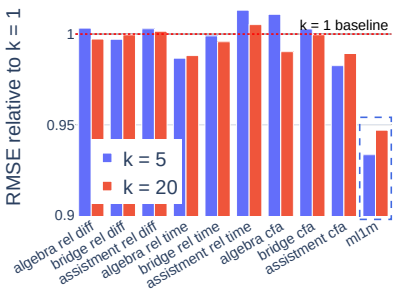


Figure 7: Evolution of flat factorization RMSE depending on rank relatively to k = 1

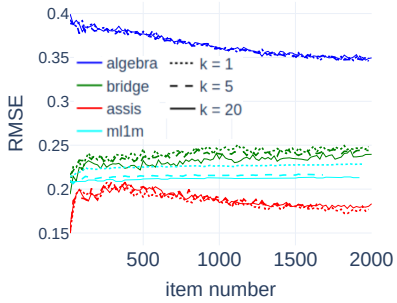


Figure 8: Evolution of RMSE while adding problems with target **Difficulty**

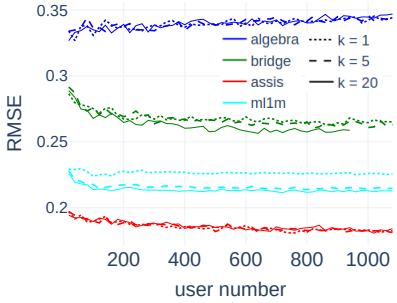
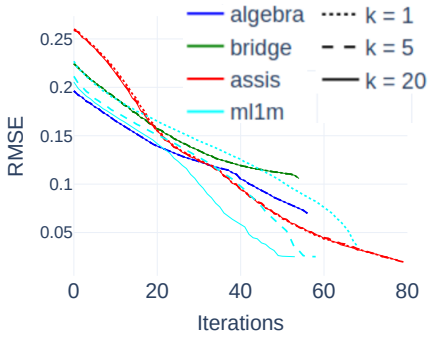
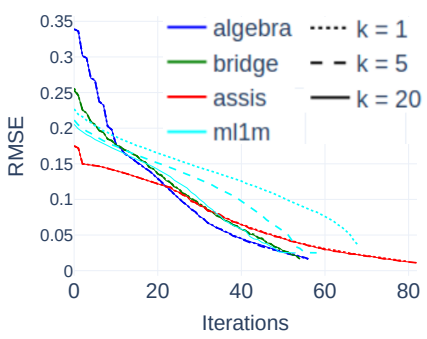


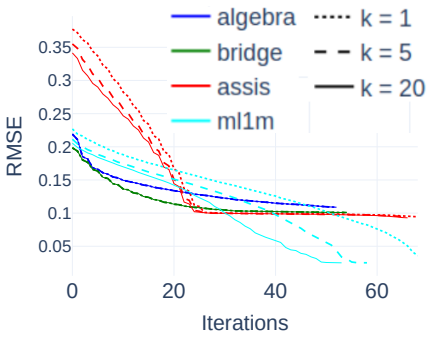
Figure 9: Evolution of RMSE while adding students with target **Difficulty**



(a) Duration



(b) Difficulty



(c) Correct First Attempt

Figure 10: Evolution of RMSE during filtering with various target

rating for problems and a "skill" rating for students, mimicking a WHR rating system. This is a second strong hint that educational datasets do not have the same structural properties as datasets from multimedia recommenders, and that if we want to extract more information and discover a better characterization of students and problems, it is necessary to consider chronological information.

7. CONCLUSION

We applied preprocessing to common educational datasets to try to improve the accuracy of MF techniques. While these did improve the results, we also showed that when MF techniques from the collaborative filtering community are directly applied, they do not benefit from having ranks higher than 1, meaning that the attribution of a single value to students and problems is about as effective as we can get. This seems to indicate that MF techniques might not be the most efficient model to extract static information from these datasets, or, more probably, that static information is scarce. We believe that this stems from the fact that, unlike users in multimedia recommender systems, students change over time as they are faced with new problems but also from outside interactions not recorded in ITS, hence chronological information needs to be taken into account in order to improve accuracy and make predictions. Still, in the eventual absence of more sophisticated analyses in a recommender system, MF can be used to extract a crude measure of what could be labeled as a level of difficulty of a problem and a level of proficiency or skill of a student.

8. REFERENCES

[1] Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. E. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. *International Educational Data Mining Society*, 2012.

[2] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[3] R. Coulom. Whole-history rating: A bayesian rating system for players of time-varying strength. In H. J. van den Herik, X. Xu, Z. Ma, and M. H. M. Winands, editors, *Computers and Games*, pages 113–124, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] M. Feng, N. Heffernan, and K. Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User Model. User-Adapt. Interact.*, 19:243–266, 08 2009.

[5] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. *ArXiv*, abs/1708.05024, 2016.

[6] C.-J. Hsieh and I. S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1064–1072, 2011.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2290, 1998.

[9] X. Luo, Y. Xia, and Q. Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271 – 280, 2012.

[10] U. Ocepek, J. Rugelj, and Z. Bosnić. Improving matrix factorization recommendations for examples in cold start. *Expert Systems with Applications*, 42(19):6784–6794, 2015.

[11] C. Piech, J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. *arXiv preprint arXiv:1506.05908*, 2015.

[12] S. Rendle and L. Schmidt-Thieme. Online updating regularized kernel matrix factorization models for large-scale recommender systems. 01 2008.

[13] T. Sergent, F. Bouchet, and T. Carron. Towards temporality-sensitive recurrent neural networks through enriched traces. In A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, editors, *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020*. International Educational Data Mining Society, 2020.

[14] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra i 2006-2007. development data set from kdd cup 2010 educational data mining challenge. find it at `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`, 2010.

[15] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Bridge to algebra 2006-2007. development data set from kdd cup 2010 educational data mining challenge. find it at `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`, 2010.

[16] N. Thai-Nghe, L. Drumond, T. Horváth, L. Schmidt-Thieme, et al. Multi-relational factorization models for predicting student performance. In *KDD Workshop on Knowledge Discovery in Educational Data (KDDinED)*, pages 27–40. Citeseer, 2011.

[17] S. Trivedi, Z. A. Pardos, G. N. Sarkozy, and N. T. Heffernan. Co-clustering by bipartite spectral graph partitioning for out-of-tutor prediction. *International Educational Data Mining Society*, 2012.

[18] J.-J. Vie and H. Kashima. Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 750–757, 2019.

[19] X. Xiong, S. Zhao, E. G. Van Inwegen, and J. E. Beck. Going deeper with deep knowledge tracing. *International Educational Data Mining Society*, 2016.

[20] L. Xu and M. A. Davenport. Dynamic knowledge embedding and tracing. In A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, editors, *Proceedings of the 13th International Conference on Educational Data Mining, EDM 2020, Fully virtual conference, July 10-13, 2020*. International Educational Data Mining Society, 2020.

[21] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *IEEE International Conference of Data Mining*, 2012.

[22] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 315–324, 2011.