Integrating OLAP and Recommender Systems: An Evaluation Perspective

Artus Krohn-Grimberghe
Information Systems and Machine Learning Lab
University of Hildesheim, Germany
artus@ismll.de

Alexandros Nanopoulos
Information Systems and Machine Learning Lab
University of Hildesheim, Germany
nanopoulos@ismll.de

Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim, Germany
schmidt-thieme@ismll.de

ABSTRACT

The integration of OLAP with web-search technologies is a promising research topic. Recommender systems are popular web-search mechanisms, because they can address information overload and provide personalization of results. Nevertheless, the evaluation of recommender systems is a challenging task. In this paper, we propose a novel framework for evaluating recommender systems, which is multidimensional and takes into account for the multiple facets of the recommendation algorithms, data sets and performance measures. Emphasis is placed on supporting business applications of recommender systems, notably e-commerce, by allowing analysts to perform ad-hoc analysis and use popular online analytical processing (OLAP) operations. Combined with support for visual analysis, action such as drill-down or slice/dice allow assessment of the performance of recommendations in terms of business objectives. We describe a detailed methodology for designing and developing the proposed multidimensional framework, and provide insights about its applications. Our experimental results, using a research prototype, demonstrate the ability of the proposed framework to comprise an effective way for evaluating recommender systems.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms

Algorithms, Experimentation

Keywords

Integration, Recommender Systems, Recommendation, Multidimensional, OLAP, Exploratory Data Analysis, Performance Analysis, Data Warehouse

DOLAP'10, October 30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Recommender systems are popular web-search mechanisms, which are used to address information overload and provide personalized results. Recommender systems can act as an electronic sales assistant and may help exploring an online-shop's portfolio or increase revenue by offering cross-selling opportunities. Classical recommender tasks are, thus, item prediction [12] where the recommender suggests items to a user based on a personalized profile and rating prediction [4] where the task at hand is predicting how much something is liked by a user.

The vast majority of related research evaluates recommender algorithms usually with aggregated numeric scores [8]. Root mean square error (RMSE) or mean absolute error (MAE) are the standards for rating prediction, precision / recall / F-measure and AUC are their counterparts for item prediction. Evidently, such measures can indicate the performance of algorithms regarding some perspectives of recommender systems' applications and they are well suited to determine the winners of most of today's challenges. But generally, business analysts will also be interested in breaking down those aggregate numbers by dimensions known to them (e.g., time, product, customer, ...) for more detailed insight into the algorithms' performance. Furthermore, they are most certainly interested in testing recommender algorithms according to their own criteria using ad-hoc queries and in examining the effect of different algorithm parameters over selected subsets of the data, too.

To address this, we propose a multidimensional framework for evaluating recommender systems. Extending a sketch of this idea [11], we describe a detailed methodology for designing and developing the proposed multidimensional framework, and provide insights about its applications.

Organizing the data in a principled way as facts, measures, dimensions and hierarchies enables two paths of analysis. One is the exploratory analysis of the data set itself, leading, among others, to customer and item segmentation, coverage analysis, insights into the business entities and a breakdown of evaluation criteria such as the average rating by the possibly interesting criteria. The other path is a detailed performance analysis of the recommender algorithms' past and present predictions broken down by valuable categories identified in the first step. One such analysis might be F-measure vs. customer segment, another may be RMSE of algorithm A compared to RMSE of algorithm B vs. category.

Another key aspect from our evaluation framework is the

ease of use of the resulting tool. Though predefined reports can be used, the strength lies in the ability to swiftly create ad-hoc queries in a graphical way. This enables business analysts to satisfy their own information needs and speeds up the work of researchers. Together with the ability to drill down along defined hierarchies in the data and the option to refine queries by slicing and dicing, interactive work with the data is encouraged. Besides that, it still remains possible to programmatically query the framework and to extend queries with custom functions for deeper analysis.

To assess the benefits of the proposed framework, we implemented a research prototype and present experimental results that demonstrate its ability to comprise an effective way for evaluating recommender systems. In particular, our experimental results are able to provide insight regarding already identified types of behavior by existing recommender systems. Our main contributions are summarized as follows: i) A flexible multidimensional framework for evaluating recommender systems that can serve as a template for researchers using it with their own recommender datasets. ii) A comprehensive procedure for efficient development of the framework in order to support analysis of both, data set facets and algorithms' performance using ad-hoc OLAP queries (e.g., drill-down, slice, dice). iii) The consideration of a much extended set of evaluation measures, compared to standards such as the RMSE. iv) Experimental results with interesting findings that relate to the recent popular $1M Netflix prize (netflixprize.com).

2. RELATED WORK

The integration of OLAP and recommender systems was introduced by Adomavicius et al. [2, 1], who treat recommender systems together with their common dimensions of users, items, and ratings as inherently multidimensional. These works focus on the multidimensionality of the generation of recommendations and suggest that recommenders themselves are multidimensional entities that may be queried like OLAP cubes. In contrast, our work acknowledges the multidimensional nature of recommender systems but focuses on the multidimensional evaluation of their recommendations.

Existing frameworks for recommender systems analysis usually focus on the automatic selection of one recommendation technique over another. E.g., [7] researches an API that allows retrieval and derivation of user satisfaction with respect to the recommenders employed. The AWESOME system by Thor and Rahm [14], the closest approach to ours, shares the data warehousing approach, the description of the necessary data preparation (ETL), and the insight of breaking down the measures used for recommender performance analysis by appropriate categories. But contrary to the approach presented here, the AWESOME framework is solely focussed on website performance and relies on static SQL-generated reports and decision criteria. Furthermore, it incorporates no multidimensional approach and does not aim at easing end-user-centric analysis or providing dynamic analysis at all.

Recent interesting findings with respect to data set characteristics that have a noteworthy effect on recommender performance, when modeled appropriately, are e.g. the results obtained during the Netflix challenge [3, 10] on user and item base-effects and time-effects in data. The long time it took to observe these properties of the data set might be an indicator that with currently available tools proper analysis of the data at hand is more difficult and tedious than it should be. This motivates the creation of easy to use tools enabling thorough analysis of the data sets and the recommender algorithm's results and presenting results in an easy to consume way for the respective analysts.

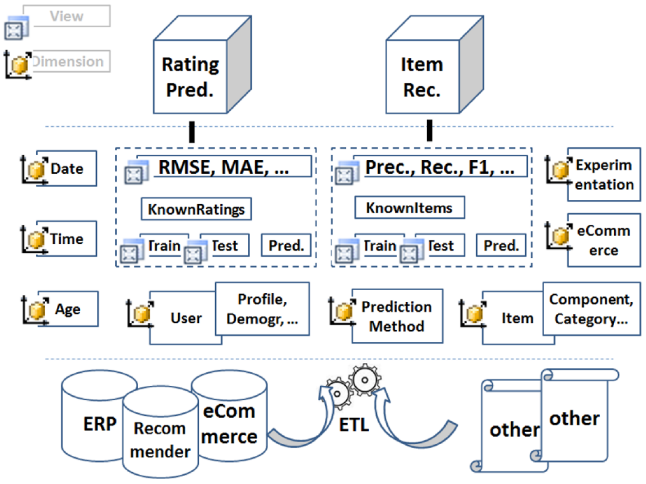


Figure 1: The three tiers of the framework. The bottom tier consists of source data and the necessary extract-transform-load process. The middle tier represents the relational data warehouse structure and the top tier consists of the OLAP cubes available to the end users.

3. PROPOSED FRAMEWORK

Analysis of recommender algorithms works by calculating an (error) score per prediction given a ground truth. Usually, those scores are then aggregated over the whole dataset or each $< prediction, groundtruth >$ pair, resulting in a single condensed number.

Considering these (error) scores as fact information regarding the recommender algorithms' performance, we aim at relating the relevant dimensions to these facts (measures). This paves the way for multidimensional analysis, where aggregate scores still are available, but slicing, dicing, drill-down, and combining different measures in ways meaningful to business analysts becomes feasible, too. Thus, we naturally reach an OLAP approach, which is based on linking factual information to dimensions used for interpretation.

Our framework will be helpful to business users and recommender algorithm researchers, who can use it as a template for their own datasets without changing anything but the data load (ETL) component. Business analysts already employing multidimensional analysis tools can use our framework as a guide to extending analytic capabilities to recommender systems. Extending our framework and adding existing dimensions from corporate OLAP cubes for extended analysis capabilities is possible and encouraged.

3.1 Overview of the Framework

The proposed multidimensional framework (figure 1) follows a three-tier design approach.

The foundation of the framework is the data gathered for the analysis of recommender systems. This data, stemming from the algorithms' predictions and from external

ItemKey	UserKey	DateKey	HourKey	Rating
2658	6034	4/26/2000	1	5
2919	6034	4/26/2000	1	3
3062	6034	4/28/2000	3	3
3266	6034	4/28/2000	3	3
3367	6034	5/16/2000	1	4
3	6035	5/16/2000	1	1
7	6035	5/16/2000	2	3
11	6035	5/16/2000	2	4

UserKey	Gender	OccupationKey	ZIP	State	Country
6034	Female	2	90210	CA	USA
6035	Male	1	10026	NY	USA

UserKey	FirstRating	LastRating	DaysActive	CountRating
6034	4/26/2000	5/16/2000	3	5
6035	4/26/2000	4/26/2000	1	3

Figure 3: Core data and its enrichment through derived data (increase-insight data)

sources (e.g., ERP, CRM), resides on tier 1 together with the dataset-specific Extract-Transform-Load (ETL) prepared it for the above tiers. From the respective source, the master data, the transactional data, and the algorithm predictions are cleaned, transformed, and subsequently imported into a data warehouse, while referential integrity between the elements is maintained.

The second tier is a relational data warehouse and it is used to connect the fact information present in the framework with the available dimensions and to enforce referential integrity. Additionally, derived information is made available via views.

The top tier, tier 3, contains the multidimensional calculations, aggregations and partitions the cubes for rating prediction and item recommendation comprise of. This is the layer the users of the framework connect and interact with. Together with the base data, the dimensions and core measures used in the proposed framework in more detail in the following subsection.

3.2 Architecture of the Framework

Following the flow of the data, we will start with presenting in more detail what data is loaded into the framework.

First, the algorithm's training data is loaded into the relational data warehouse. For most of today's algorithms this is so-called explicit or implicit feedback data consisting of previous ratings, shopping baskets, click-logs or similar. Together with the training data, the test data (test set) used for evaluation of the recommender and the predicted information (i.e. ratings or items) are loaded into the respective tables for train, test and prediction. Additionally, the respective dimensions for analysis are loaded into the framework. They are the metadata for analysis and to a large extend should be available as corporate master data; see table 1 for a reference of the dimensions.

Fact information and the regular dimension information

Framework Dimensions
User
Occupation
Item
Category
Component
PredictionMethod
Date
Time
Age
Experimentation
eCommerce

Table 1: The dimensions proposed for our framework.

forms what we will call *core data* as it is present in every recommender setting.

Additionally, in a recommender scenario it is often meaningful to use facts as dimension members: i.e., it is certainly interesting to explore recommender performance against customer classification or against item sales or training instances per user or item. This information, which we will call *increase-insight data* is present in the respective fact tables in and can usually only be used as a slicer with complex and slow-running query statements. In the framework, this information is added to the dimensions it belongs to, such as User or Item, via views.

Exemplarily, figure 3 shows rating fact data that is related to the respective user dimension to provide some increase-insight data (here: the number of active days as DaysActive, the number of ratings spent as CountRatings, and information about the first and the last rating). Table 2 shows available information for the user dimension. The item dimension share analogous members.

Core data represents the algorithms' training data, their persisted predictions and the related dimensions. Common examples for training data include, e.g., past ratings, purchase transaction information, click streams from an online shop and audio listening histories. The metadata used for analysis of the fact data is added to the framework by means of dimensions and their hierarchies.

Increase-insight data Especially in business scenarios not all users or items are equal. This leads to the fact that algorithm prediction errors may have different weight depending the subject in question. Many performance indicators for importance with respect to a business scenario are easily derived but they constitute fact information instead of dimension information. Increase-insight data describes former fact (measure) data that is pivoted to dimension members for additional analysis capabilities. Examples include dynamic abc-analysis information, rating or purchase counts, days of activity and activity co-efficients.

Users should only connect to OLAP cubes for Rating Prediction and Item Recommendation. Furthermore, the dimensions in the multidimensional space mostly parallel their relational cousins. Thus, we now proceed with the description of the multidimensional space as presented in figure 2.

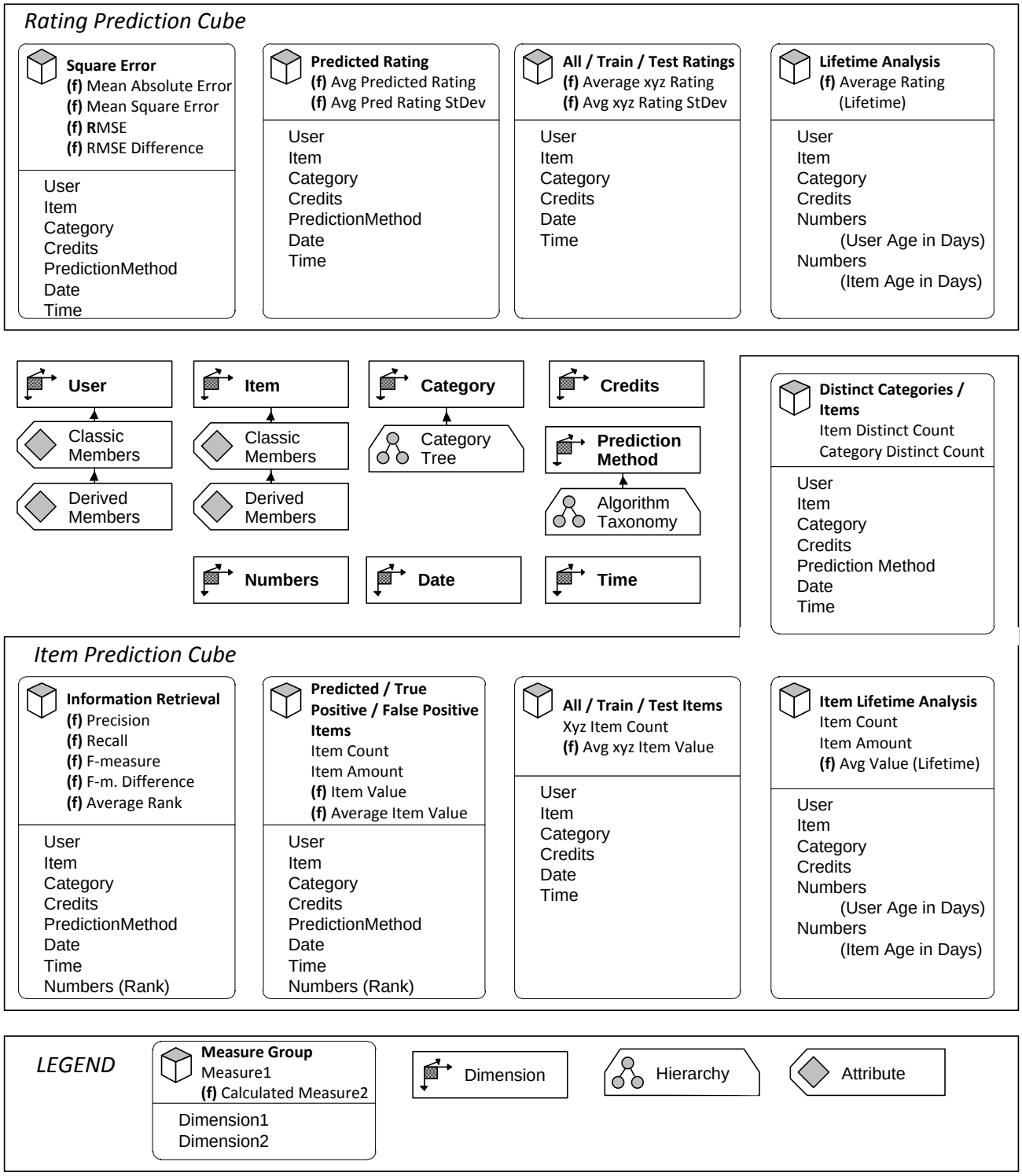


Figure 2: The two cubes, their measure groups, and the related dimensions.

Sample increase-insight data
ABC-classification
Count of rated items
Count of items
Date of first rating
Date of first item
Number of active days
Ratings per active day
Items per active day

Table 2: User increase-insight data added for train, test and the union of both.

The Date, Time, and Age dimensions are for temporal analysis of both, the recommender algorithms' predictions and the base data. Age refers to the relative age of the user or item at the time the rating is given / received and allows for extended temporal analysis (c.f. section 4.2). User and the related dimensions such as UserProfile and UserDemographics allow for analysis by user master data and by dynamically derived information such as activity related attributes described above under increase-insight data. Item and the related dimensions such as ItemCategory and Credits (alternatively: Components) parallel the user-dimensions. The PredictionMethod dimension allows the OLAP user to investigate the effects of the various classes and types or recommender systems and their respective parameters. As recommender algorithms usually accompany a commercial or scientific application (e.g., eCommerce) that will have dimensions on its own, these dimensions can easily be integrated into and be used by our framework. In case this framework is used in an experimentation-driven scenario [5], such as an online or marketing setting, ExperimentationMethod related dimensions should be added. They are analogous to the PredictionMethod dimension but are more specific to their usage scenario. All dimensions are used in both cubes as presented in figure 2.

The measures being analyzed in both cubes can be grouped by basic statistical and information retrieval measures. Among the basic statistical measures are counts and distinct counts, averages and their standard deviations (stdev), ranks, (running) differences and (running) percentages of various totals for each dimension table, train ratings, test ratings and predicted ratings. Several of them are directly exposed, while others—such as the percentages and ranks—are provided via MDX. A listing of the directly exposed measures for the Rating Prediction and the Item Recommendation cubes is found in tables 3 and 4, respectively. Among the information retrieval measures are the popular MAE and (R)MSE for rating prediction, plus precision, recall and F-measure for item prediction. We also support novelty, diversity, and coverage measures, which improve the user experience. Furthermore, for comparative analysis, the differences in the measures between any two chosen (groups of) prediction methods are supported as additional measures.

SQUARE ERROR
Mean Absolute Error (MAE)
Mean Square Error
Root Mean Square Error (RMSE)
RMSE Difference (A minus B)
PREDICTED RATINGS
Average Predicted Rating
Average Predicted Rating StDev
Predicted Rating Count
ALL RATINGS
Average Rating
Average Rating StDev
Rating Count
TRAIN RATINGS
Average Train Rating
Average Train Rating StDev
Train Rating Count
TEST RATINGS
Average Test Rating
Average Test Rating StDev
Test Rating Count
LIFETIME ANALYSIS
Average Rating (Lifetime)

Table 3: Measure Groups in capital letters and their respective Measures for the Rating Prediction cube

The Item Recommendation Cube
INFORMATION RETRIEVAL
Precision
Recall
F-measure
F-measure Difference (A minus B)
Average Rank
PREDICTED ITEMS, TRUE POSITIVE ITEMS,
FALSE POSITIVE ITEMS
Item Count
Item Amount
Item Value
Average Item Value
ALL ITEMS, TRAIN ITEMS, TEST ITEMS
Item Count
Average Item Value

Table 4: Measure Groups in capital letters and their respective Measures for the Item Recommendation cube.

4. RATING PREDICTION PROTOTYPE

This section describes the implementation of a research prototype for the proposed framework and its application to a standard recommender dataset. The prototype was implemented using Microsoft SQL Server 2008 and the presented results could all be obtained graphically using Microsoft Excel 2007 and later.

4.1 Prototype Implementation

In our evaluation, the prototype considers the Movielens 1m data set [6], which is freely available and a common benchmark for recommender systems. It consists of 6.040

users, 3.883 items, and 1.000.209 ratings received over roughly three years. Each user has at least 20 ratings and the metadata supplied for the users is userid, gender, age bin, occupation, and zip-code. Metadata for the item is movieid, title, and genre information.

For the ETL process, SQL Server Integration Services were used loading the Movielens 1m data set from text files into the framework's relational data warehouse. Data types were adapted, surrogate keys created, and the respective lookups performed.

The tables used in the warehouse of the prototype are Date, Time, Genre (instantiation of Category), Item, ItemGenre (table needed for mapping items and genres), Numbers (a helper table for the age / lifetime analysis), Occupation, PredictedRatings, PredictedItems, PredictionMethod, TestRatings, TestItems, TrainRatings, TrainItems, and User. The Item and User table are in fact views over the master data provided with the Movielens data set and dynamic information gathered from usage data (c.f. section 3.2. Further views are SquareError, AllRatings, and AgeAnalysis.

On top of the warehouse prototype, the Rating Prediction OLAP cube was created using Microsoft SQL Server Analysis Services. The Item Recommendation cube was not implemented.

As measures we created the respective counts and sums, and further derived measures such as distinct counts, averages, stdevs, ranks, (running) differences and (running) percentage. From square error the core measures MSE, RMSE, and MAE are derived as presented in section 3.2.

4.2 Performance evaluation

We evaluate the usefulness and effectiveness of the proposed framework by using our prototype. Our focus is on demonstrating the flexibility and ease with which we can answer important questions for the performance of recommendations. Due to lack of space, we present only the most characteristic results. Since our prototype performs movie recommendations, we show findings that relate to noteworthy conclusions drawn from the $1 Million Netflix prize, which attracted large popularity far beyond the recommender systems community.

As discovered during the Netflix prize, it is useful to explicitly model the effects describing changes in the rating behavior over the various users (user base-effect), items (item base-effect), and age of the respective item or user (time effects) [3, 9, 10]. For this reason, we choose to demonstrate the benefits of the proposed framework by setting our scope on those effects followed by an example of comparative performance analysis. The framework allows an easy visualization of the item effect described e.g. in [9], namely that there usually is a systematic variation of the average rating per item. More than that, when bucketing the rating counts by roughly equal number of items, a trend of heavier rated items being rated higher can be observed (figure 4). Further research shows there are two aspects to this effect: first, that movies receiving more ratings per day are on average rated higher (blockbuster effect; see figure 5) and, second, that, in accordance with [10], the average rating increases with increasing item age (all-time classics effect, figure 6). From a line of business application point of view, the speed-up in decision time that can be achieved when using the ratings-per-day effect is crucial.

Running the same analysis on the users (results omitted

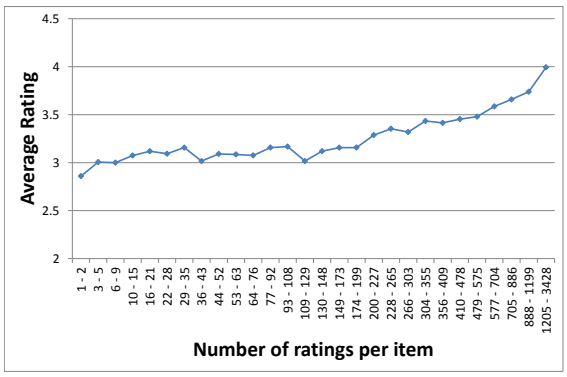


Figure 4: The effect of the number of ratings per item on the average rating.

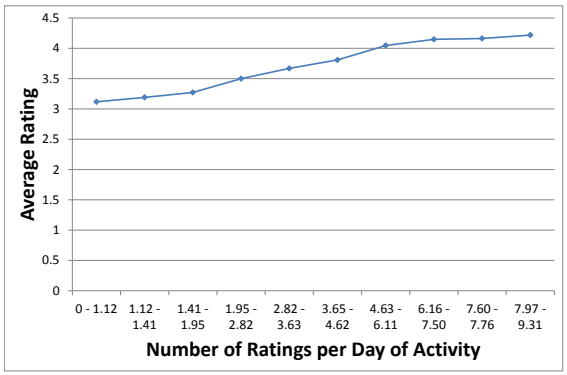


Figure 5: The effect of the number of ratings per day an item receives on the average rating.

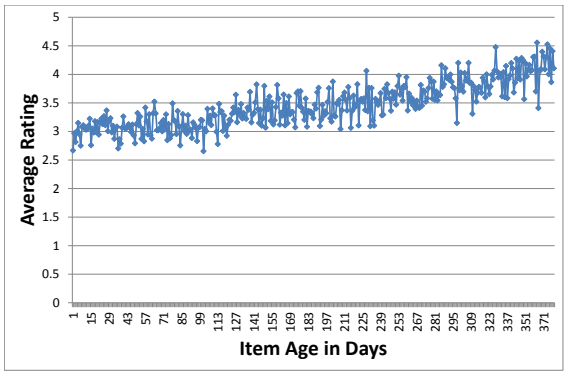


Figure 6: The all-time classics effect. Ratings tend to increase with the age of the movie at the time the rating is received. Age is measured in days since the first rating on the respective item is recorded.

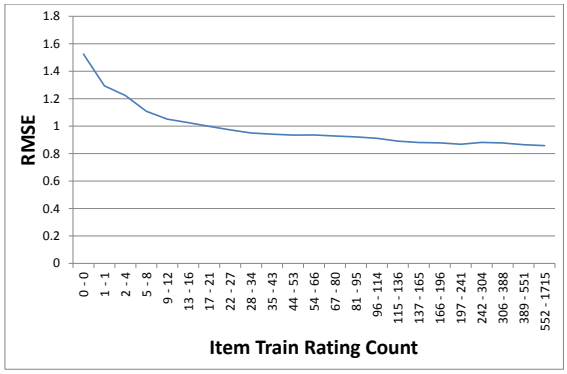


Figure 7: RMSE of the Matrix Factorization algorithm vs. ratings available per item on the train data set.

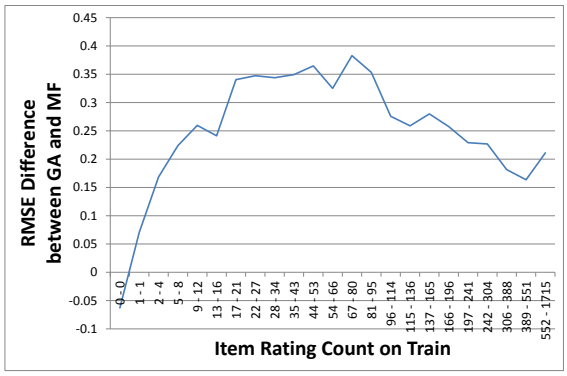


Figure 8: Difference in RMSE between Matrix Factorization (MF) and Global Average (GA) vs. ratings available per item on the train data set.

for brevity), it is interesting to note that the user rating count effect is inverse to the item rating count effect described above: the higher the amount of ratings spent by a given user, the lower his or her average rating. One explanation to this behavior might be that real heavy raters encounter a lot of rather trashy or at least low quality movies.

For algorithm performance comparison, the Movielens 1m ratings were randomly split in two nearly equal size partitions, one for training (500103), and one for testing (500104 ratings). As an example, a vanilla matrix factorization [13] (RMSE of 0.8831 given the presented train-test-split) was compared to the global average as baseline method. Algorithm parameter estimation was conducted on the training samples only using 5-fold cross validation. Figure 8 reveals that for this factor model, as expected, more ratings on train do increase the relative performance. But beyond a certain point the static baseline method will regain lost ground. Investigation into this issue might be interesting for future recommender models.

In order to demonstrate how easily the previously shown figures can be generated with the presented framework, figure 9 show the process of generating figure 7: Starting with an OLAP client (Microsoft Excel 2010) connected to the Rating Prediction cube (figure 9a) we see the characteristic pivot table. Note that with Excel you have the opportunity to only display dimensions and measures related to a specific measure group, which was employed here—Square Error is selected. The first step then is adding a dimension to the pivot table by clicking on it or dragging it to the "Values" box. This yields figure 9b, presenting the aggregate RMSE of 1.007 over all predictions by all algorithms. The next action is limiting the report to show only the results of the matrix factorization algorithm which is done by adding the PredictionMethod dimension to the "Report Filter" field and selecting the appropriate value from the filter drop down list (figure 9c). Finally, we can add the desired dimension member, i.e. Item.[Train Count Rated], on rows (leaving RMSE values as the column dimension) and, technically, we are done (figure 9d). Pressing the "Pivot Chart" button and tidying up the resulting graphic will lead to figure 7.

5. CONCLUSIONS

We have proposed a novel multidimensional framework for integrating OLAP with the challenging task of evaluating recommender systems. We have presented the architecture of the framework and described the implementation of a research prototype which can serve as a template for other datasets. Our evaluation considered the popular task of movies recommendation, and our results indicate the suitability of the prototype to easily provide conclusions whose importance has been indicated during the recent Netflix prize. Moreover, we demonstrated the ease of obtaining the presented results and highlighted the reusability of the framework due to its template character for recommender datasets.

In our future work, we will consider the extension and generalization of our research prototype as well as developing a web-based implementation that will promote its usage.

6. ACKNOWLEDGMENTS

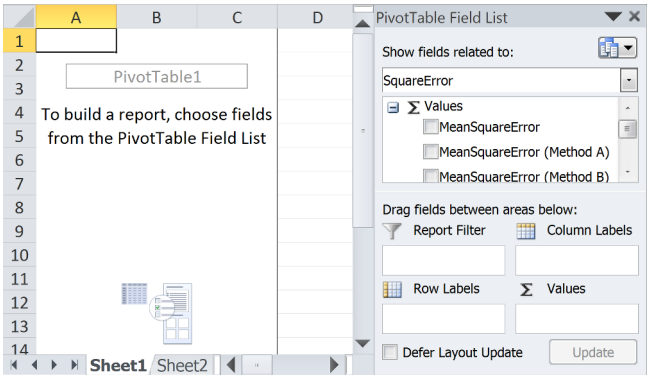
The authors gratefully acknowledge the partial co-funding of their work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006 and through the European Regional Development Fund project LEFOS (www.ismll.uni-hildesheim.de) under the grant agreement no. 80028934. Furthermore, the corresponding author thanks Vladimir Stepa for discussions and advice regarding Microsoft SQL Server Analysis Services.

7. REFERENCES

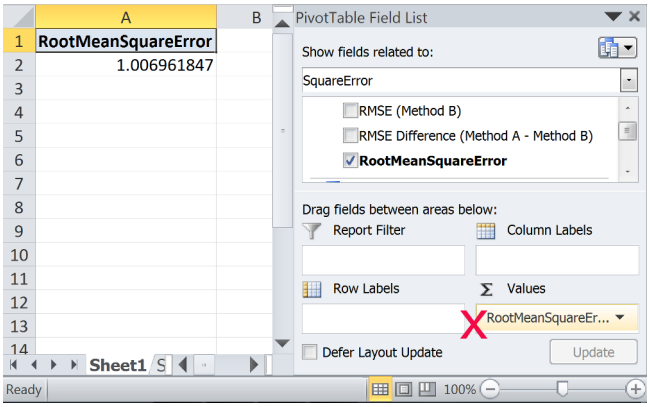
[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin. Multidimensional recommender systems: A data warehousing approach. In *WELCOM '01: Proceedings of the Second International Workshop on Electronic Commerce*, pages 180–192, London, UK, 2001. Springer-Verlag.

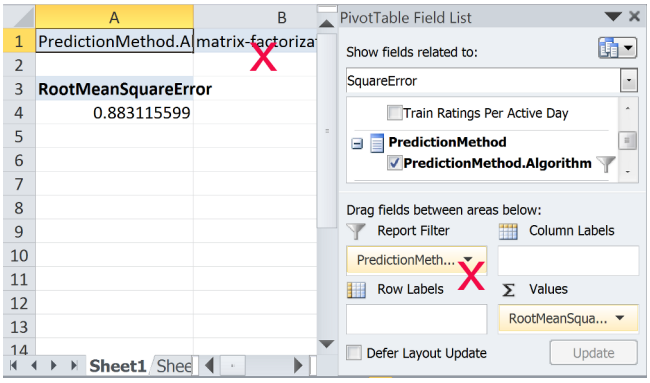
[3] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104, New York, NY, USA, 2007. ACM.



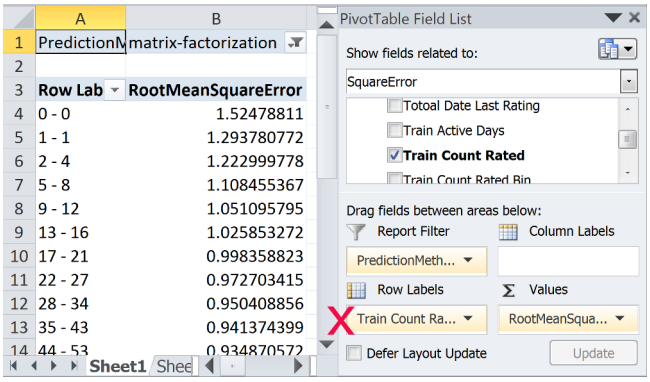
(a) An empty pivot table...



(b) Added the measure "Root Mean Square Error"



(c) Filtered by "matrix-factorization" as prediction algorithm.



(d) Drilled down by Item Rating Count on Train.

Figure 9: The making of figure 7.

[4] J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[5] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham. Seven pitfalls to avoid when running controlled experiments on the web. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, New York, NY, USA, 2009. ACM.

[6] GroupLens. Movielens data sets. http://www.grouplens.org/node/73.

[7] C. Hayes, P. Massa, P. Avesani, and P. Cunningham. An on-line evaluation framework for recommender systems. In *In Workshop on Personalization and Recommendation in E-Commerce (Malaga*. Springer Verlag, 2002.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[10] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[11] A. Krohn-Grimberghe, A. Nanopoulos, and L. Schmidt-Thieme. A novel multidimensional framework for evaluating recommender systems. In *Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI)*. CEUR-WS, to appear.

[12] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *1994 ACM Conference on Computer Supported Collaborative Work Conference*, pages 175–186, Chapel Hill, NC, 10/1994 1994. Association of Computing Machinery, Association of Computing Machinery.

[13] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *NETFLIX '08: Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8, New York, NY, USA, 2008. ACM.

[14] A. Thor and E. Rahm. Awesome: a data warehouse-based system for adaptive website recommendations. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 384–395. VLDB Endowment, 2004.