Generalizing User Expertise in Recommender Systems

James Liu **Rahul Pandey** **Hongxia Zhong**
{yuqiliu,rkpandey,hzhong62}@stanford.edu

Abstract

We develop a recommender system based on learned expertise level associated with users and reviews, given a community of users and their product ratings. Users can evolve over time as they write more reviews and as they spend more time within the community. The performance of recommender system improves significantly by modeling in user expertise, reducing the Root Mean Square Error (RMSE) as much as 33.9% over the traditional latent factor model. This paper presents the implementation of the expertise-based algorithm, together with an analysis of how the entire community gains expertise over time. On average, users spend the majority of time in the lowest and highest experience levels, rapidly gaining experience in the middle. Finally, we study if the rate of learning on a certain product category is more dependent on the user or the community.

1 Problem Description

Interest in developing and improving recommender systems has grown in recent years given the rise of ecommerce sites such as Amazon, CNN, and Netflix. These online marketplaces have plentiful access to user historical data and an ability to recommend a long tail of products that physical stores could not offer [1]. Most recommender systems attempt to identify similar users or similar items (collaborative filtering and item-item similarity, respectively) to offer products that a user would likely purchase or consume. However, most recommender systems are unable to capture the evolution of a user who learns and adapts as he/she changes and as the community evolves. McAuley and Leskovec first studied the idea of modeling user evolution into recommender systems based on data from the domains of beers, wines, and movies [2]. They capture the idea that certain products may not be appreciated by a user with little expertise on the products, but the user's expertise may evolve over time, and the user will begin to appreciate products more sophisticated and beloved by experts. Alternatively, certain attributes of a product may change over time, such as if a movie has been out long enough that it is considered a classic.

As discussed in the McAuley and Leskovec paper, the idea of personal development maps well to our intuitive understanding of beer consumption. The paper uses data from RateBeers, an online portal to rate different types of beers. We generally think of users as progressing from introductory beers like Bud Light to more nuanced or refined beers like Indian Pale Ales (IPAs). Few beginner beer drinkers would initially find the bitterness of IPAs palatable, yet expert drinkers would not want to go back to Bud Light. McAuley's work was to capture this notion of acquired taste and user evolution, and in this paper we study this further. It is important to note two different factors at play here: first is the experience of the user in terms of their true age. This distinguishes between the tastes of a teenager and someone middle-aged. The second factor is the experience of the user within the community. This distinguishes between a user who has just recently joined a community and another who has been part of a community for several years (and has thus become acclimated to the terminology and norms).

In his paper, McAuley demonstrated the value in modeling user expertise in a domain by showing significant improvements in recommender system performance. One of the questions that arises from McAuley's paper is how well the concept of user expertise generalizes to other product categories. There is a well-defined, passionate com-

	Books	Music	Movies	Gourmet Foods
Total User Count	2.59M	1.13M	1.22M	113K
Users w/ many reviews	103K	44.4K	65.9K	258
# reviews same day	208K	55.9K	276K	1385
# reviews diff. days	1.26M	3.43M	4.22M	6350

Table 1: Basic statistics about selected dataset. The threshold to qualify for many reviews is 10 for the smaller datasets (Gourmet Foods and Video Games), and 20 for the larger ones (Movies and Music).

munity with the beer drinking community, and one would expects beer tasters to follow a similar maturation process into becoming experts. However, users in other product areas such as Books, Arts, or Music are likely different from the RateBeers community. These groups are broader in terms of their appeal, and there is no intuitive progression of user development. For example, users do not converge to liking a common type of Music, even after being involved in the community for many years.

Another question which arises is to better understand how users gain expertise. Is the biggest determination of user expertise the number of reviews written, time involved in the community, or some combination of the two? Finally, how does the ability to acquire expertise in one domain translate to acquiring expertise in another? Or to rephrase it, if someone rapidly gains expertise in one domain, whether or not this 'rate of learning' to some extent translates to other domains. These questions could all have a meaningful impact on recommender systems. With a better understanding of user progression patterns in expertise, we are able to produce even better recommendations.

2 Prior Work

The task of building a recommender system has been well studied in the last 15 years following the rise of recommendations for online marketplaces [3, 4]. Traditional papers discuss recommender system approaches such as content-based recommendations and collaborative filtering. In general, the task is to recommend various items to users based on an utility matrix which should capture some notion of user preferences.

Only recently, however, have they captured the idea of individual user's evolution of taste over time. McAuley's work was one of the first to do so [2]. Rashid et. al. deal specifically with the task of how to cater to new users in recommender systems [5]. For new users in the community, they propose adopting information theory techniques to extract as much information as possible. This allows them to make better recommendations by learning about the initial experiences of users in the overall community. While they showed significant improvement for new users in the community, the applicability of this work is limited since it only applies to a small percentage of the users. This does not address the issue of users joining the community at different expertise levels, or the evolution of the entire community over time.

Lathia et. al. study temporal diversity in recommender systems, whereby they capture the idea that recommendations for users involved in the community eventually get stale [6]. Therefore, they propose a new evaluation metric for collaborative filtering systems in which they maximize temporal recommendation diversity without extensively penalizing accuracy. This work does address the idea of past user experiences in the community, but the main shortcoming is that user and product parameters evolve along a single timescale. That is, Lathia's model does not allow for the flexibility for users to learn at different rates. Like many other works which attempt to personalize at the user level, this work views the entire community moving as one entity.

3 Data

Our data source was the e-commerce store Amazon. We had access to anonymized review data from a variety of Amazon categories. The categories on which we performed analysis were Gourmet Foods, Video Games, Music, Movies and Books. The data format of the reviews was a text file containing a star rating, review text, and user/product identifying information.

One advantage of using only Amazon data was consistency among users across each product category. The same user leaving a review in Video Games could be identified in the Music category since the users would share the same ID. This

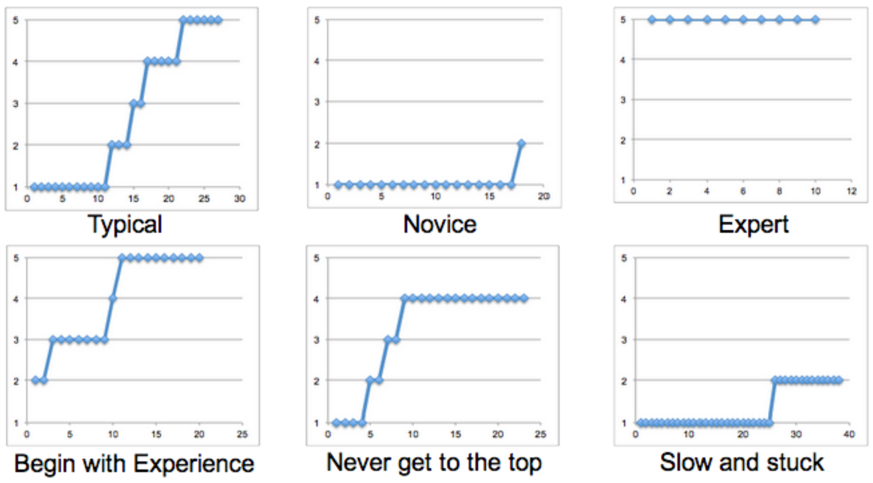


Figure 1: Experience was modeled as a monotonically increasing discrete integer value from E = 1 to 5. Most users spent the majority of their time on the lowest or highest experience level (labelled Typical).

allowed us to correlate experience level growth across datasets.

The relevant fields in the Amazon data are user ID, product ID, users' rating on the product, and the timestamp. One interesting aspect to the data, not found in the RateBeers data, was the large fraction of users that left dozens of reviews in a category in a single day. This behavior indicates that many users may simply be dumping a backlog of reviews they have in a short time span, as opposed to 'learning' more about the product category between reviews. Some quick statistics of reviews from such users are also shown in Table 1. We will briefly discuss the effect of this data characteristic in the Results section.

4 Methodolgy

4.1 Overview

The first step in our project was to build a similar system as described in the McAuley paper, with the ability to tune the recommender system built in order to train over other datasets.

We implemented our algorithm in Python and believed that our program would run reasonably fast if we took advantage of the efficient high-dimensional computing capabilities provided by Numpy and the sophisticated L-BFGS implementation in Scipy. Although an optimized C++ program can easily outperform a Python implementation, we decided to use Python as we hope to benefit from the fast iteration cycles when developing in scripting languages.

When we were in the process of developing the algorithm, we manually constructed small artificial datasets to test with. Then we trained our models on computation optimized Amazon EC2 instances. Thanks to the monstrous computing power offered by Amazon AWS, our Python scripts were only 5x-10x slower than the highly optimized C++ program written by McAuley.

In hindsight, it was a right call to use Python rather than C++ because it accelerated our development cycle, making it possible to complete this project within limited time this quarter. In the meantime, the amount of runtime performance sacrificed was also under a tolerable limit.

4.2 Runtime Performance Optimization

Considering that we planned to tackle some of the largest Amazon review datasets (the Books dataset is 6GB and 12.9 million reviews), we were thorough in optimizing our algorithm. Otherwise, it would have been almost impossible to perform any meaningful analysis if the training time was 5 days.

Other than some of the most obvious optimizations such as using Numpy arrays and matrices for high-dimensional calculations and providing our gradient calculations for Scipy's L-BFGS to replace its numerical estimation, we worked to improve the parallelization of the L-BFGS step once we identified this step as our performance bottleneck. Unfortunately, Scipy does not provide any multi-threading support to its L-BFGS library, so the only option for us was to parallelize the ob-

	Books	**Music**	**Movies**	**Gourmet Foods**
Expertise Model	0.723	0.904	0.993	1.02
Latent factor	1.09	1.25	1.29	1.21
% improved	**33.9%**	**27.6%**	**23.2%**	**15.7%**
McAuley % improved	32.9%	35.0%	29.2%	20.5%

Table 2: Performance of expertise-based recommender system compared to traditional latent factor model, as measured by Root Mean Squared Error (RMSE). The 3rd row shows how our results are comparable to the algorithm produced by Julian.

jective function and gradient function for L-BFGS calculations – they are the only parts exposed to us. The first thing we tried is Python's native multithreading pool. Although it immediately brought us a 2x improvement on performance when we tested it locally, it did not scale as we expected when we ran it on our EC2 instance with 32 virtual cores. Later, we realized that due to the presence of GIL (Global Interpreter Lock), Python's multithreading mechanism is not designed to parallelize CPU intensive tasks. Then we decided to rewrite the multithreading program as a multiprocessing program, which brought us a significant performance boost as we expected, but we ran into new problems. The most challenging problem was decomposing calculation into independent modules so that the worker processes could cooperatively calculate the object function and its gradients with limited shared memory. Although we devised an algorithm that distributes objective function calculation to independent worker processes, we failed to extrapolate this algorithm for gradients calculate: the partial gradients returned by each process have extremely high dimensionality so that it is impossible to efficiently push intermediary results into a shared queue. Fortunately, the EC2 instances had enough memory for us to allocate the entire gradient in shared memory so that the worker processes could then update gradients using their partial results collectively.

4.3 Evaluation metric

To train our expertise-based model and evaluate its performance, we divide each Amazon Review dataset into three parts: a training set, a validation set, and a testing set. For each user, we randomly place 80% of his/her reviews in the training dataset, 10% in validation and the remaining 10% in testing. We chose the reviews at random because the training set should ideally include reviews of all experience levels for each user. Otherwise, it is possible that our trained model has no knowledge of the experience level of the reviews in the testing set.

Evaluation consisted primarily of calculating the RMSE on the testing sample. We compute the difference in a user's predicted rating on a product (based on the recommender system output) and the true rating given. As described in McAuley's paper, the RMSE is a metric for the accuracy of a model prediction.

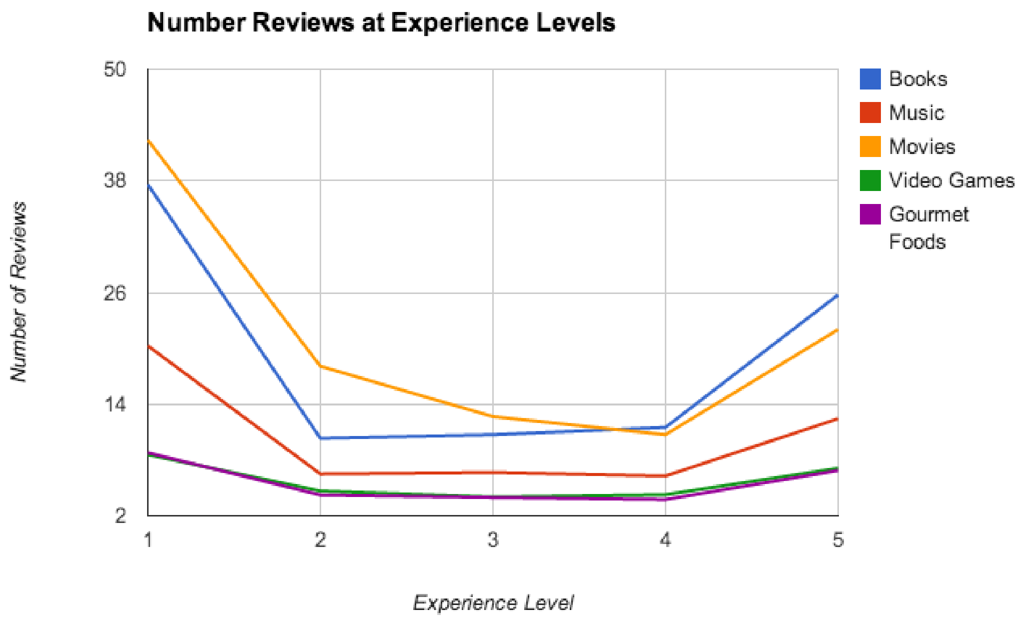
One major concept we tried to understand was user expertise progression across datasets. After understanding the breakdown of how long users are in each experience level, we aimed to determine the primary driver for user expertise growth. Does it depend primarily on the user ability to learn about the product category, or is it more dependent on the community's rate of learning?

It is difficult to evaluate the correctness of user experience growth since there is no ground truth answer. The concept we are trying to model is roughly characterized as user expertise, but this is not a measurable quantity. In the Future Work section, we discuss collecting user self-assessment of their expertise in order to gauge the correctness of the learned experience level.

5 Results

5.1 Replicate and Generalize the Expertise-Based Recommender System

The first step we made was to implement the expertise-based system proposed in McAuley, and study how well it generalizes to a wider range of datasets including music, gourmet foods, video games, and books, compared to the baseline traditional latent factor model. We still use average root mean square error (RMSE) of the prediction on a small test dataset withdrawn randomly from the entire dataset as the criterion on how well the recommender systems are performing.



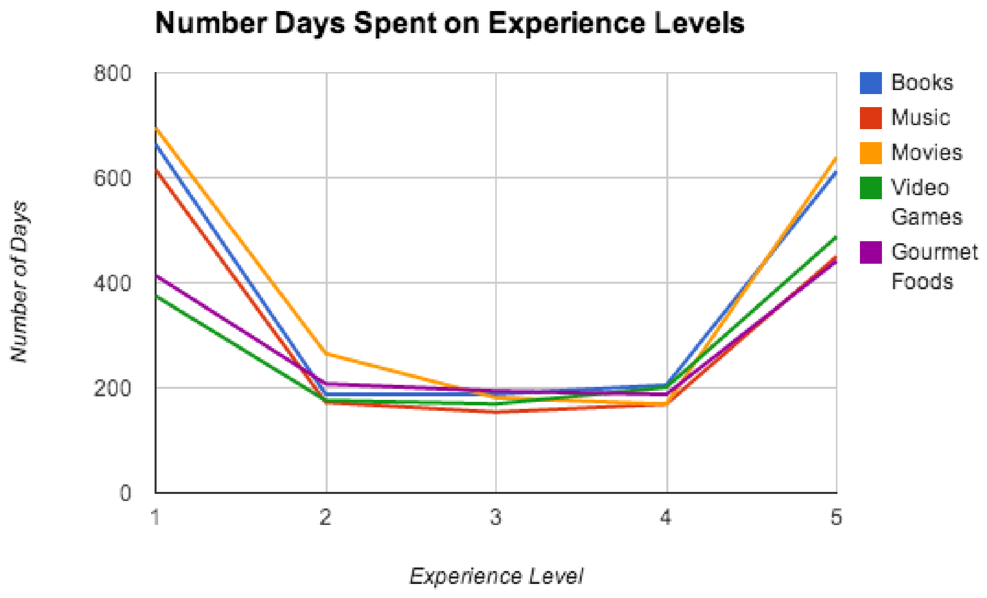


Figure 2: Users spend most of their time on the minimum and maximum experienced levels. Time can be measured in terms of number of reviews written (left) or number of days (right).

One point to note is in McAuley's original paper, he applied two different methods to create the test datasets: the first took the final ratings of each user as the test dataset, and the second method is the same random selection one mentioned above. We only considered the random selection method in our implementation because we did not have information to classify the final ratings into experience levels when the immediate prior votes are not at the highest level. For example, if a user has 50 ratings in total, and if we take out the last 5 ratings as the test dataset, and training the recommender system gives out the result that within the 45 reviews, the first 15 are in level 1, second 15 are in level 2, and the last 15 are in level 3, then we cannot tell whether the last 5 votes are in level 3, 4 or 5.

Also different from the originally proposed approach to draw a certain number of reviews from each user as test dataset, we instead drew a random 10 percent of all reviews from each user, given the fact that users with not that many reviews might not have enough training data points if certain number of reviews are drawn out. The results produced by our implementation are presented in Table 2. From the table we can see that the expertise-based model greatly outperformed the basic latent factor model by 15 to 33 percent, in terms of reduction in RMSE. One interesting fact to note is that the expertise-based system produce higher improvement on larger datasets, like Movies, Music, and Books, all with over 23 percent improvement, and does not do as well on smaller dataset, namely gourmet foods and video games. One explanation is large datasets have a significantly more ratings per user, and the model can thus provide a more accurate view for an individual's actual learning curve, which allows the expertise-based model to work the way as designed.

The table also presents results we obtained from running McAuley's implementation on the new datasets, and the results from the two implementations are mostly in agreement. The discrepancies are suspected to be due to the randomness in test dataset selection and different model specification, like the regularization parameter.

5.2 Descriptive Analysis of User Expertise Progression Over Time

After obtaining the fully trained expertise-based model, we also obtain as a byproduct the entire history of each user's expertise level. As a result, we have information on at what level a user entered a product group, how long they spent on a certain expertise level, how many reviews they wrote during that period of time, and what expertise level they eventually get to.

Based on this information, we did a descriptive analysis to understand how users in a certain product group progress on expertise levels. The results are shown in Figure 1 and Figure 2.

In our current model of five expertise levels for each datasets, users on average spent one to two years on the first level, around half a year on the subsequent three levels, and then stayed on the highest level. The pattern also shows consistency across all datasets. Similar patterns hold if we analyze the number of reviews people write within each expertise level, although smaller datasets had fewer reviews in total. Another interesting fact is for people who attain the highest level, they on average have also spent one to two years on that level, although they write reviews less frequently than users on level 1. For example, in the movies dataset a user on average spends 695.7 days in level 1 and writes 42.4 reviews; and users who reached level 5 have spent 639.0 days on that level, yet only wrote 22.0 reviews while on that level.

5.3 User Across Dataset Analysis

To determine the drivers of user expertise evolving, we first isolated users which had a significant number of reviews across multiple datasets. Based on a user's learning history on one product category, we predicted the expertise level of that same user on another dataset in two ways. First, we assume the same shape of the learning curve, namely, a user spends the same percentage of reviews on all experience levels. The second approach assumes the user have the community average learning curve. Two sets of predictions are obtained to compare with the user's actual expertise progression on the other dataset. Finally, we compute the difference between these two quantities across all users. The two numbers outputted provide a summary of whether the user is driving experience growth more or the community. This process is shown in Figure 3.

Figure 4 shows that the predictions from learning curves from different product of the same user perform better than the predictions obtained from the community average learning curves. If a user's

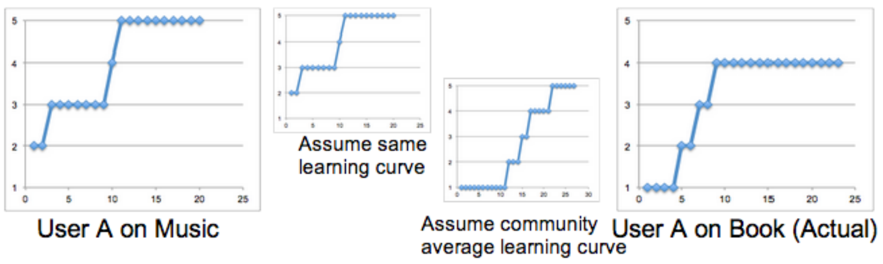


Figure 3: To evaluate if a user's learning ability transfers across communities, we first predict the user's ratings based on their progression in another dataset. Then we again predict the user's ratings based on the average user in the other dataset.

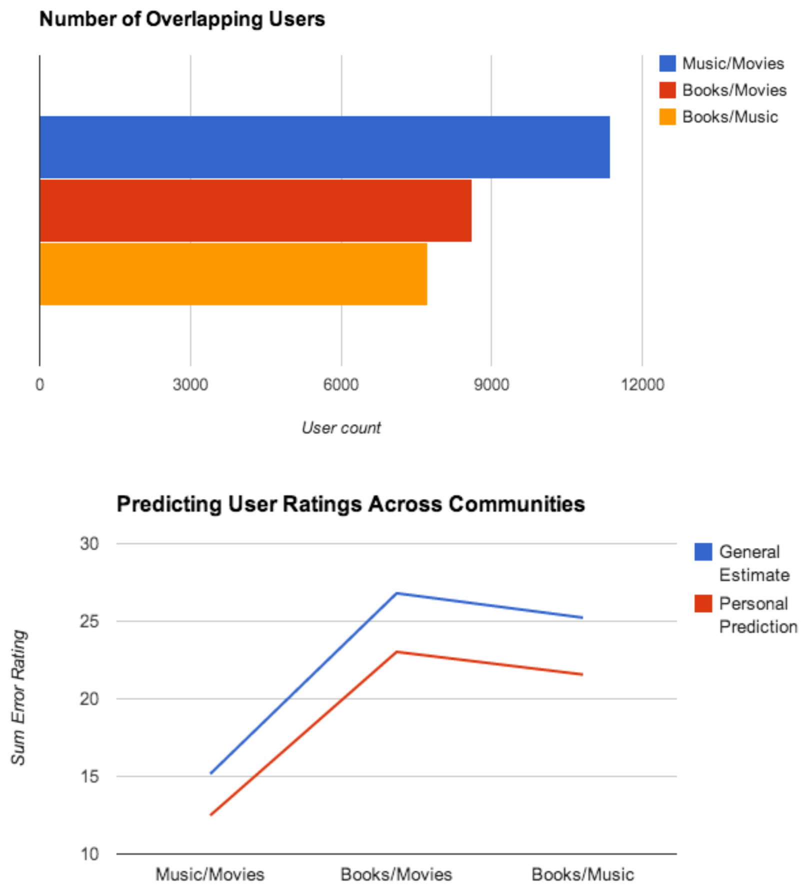


Figure 4: Left: The number of users in overlapping communities. Right: rating prediction improved by using taking the user's past experience progression compared to the general community progression.

learning curve on one dataset is uncorrelated with the user's learning curve on another dataset, the error produced by the individual learning curve approach should perform worse or at most the same as the community average approach. Otherwise, it would show that an individual's learning curves are positively correlated across product categories.

Based on our results from the three large datasets, movies, music, and books, the individual learning curve predictors actually outperform those from the community average method. The outperforming implies that learning speed has some association with individual learners, namely, one user's learning rate in a different product category correlates to his or her learning rate in another dataset.

5.4 Future Work

One area that calls for significant amount of future work is a more delicate system to predict a user's learning curve and expertise level at a given point of time (probably in the future, for the next review). In the section above, our findings reveal that users' learning curves across different datasets are positively correlated. Consequently, it would be useful to combine a user's rating history of the target product category with rating histories in other product categories to produce better prediction on when will users advance on their expertise levels and give recommendations accordingly.

Another interesting area of study is to educate the recommender system based on initial user ratings, namely, what products a user begin with and what ratings he or she gives. If the user see himself or herself as an expert prior to joining the community, clearly the recommendations for an inexperienced user (which is the default assumption for users new to a community) would not suffice. By collecting user preferences and their self-assessment, the recommender system can further reduce the time to learn expertise level and progression patterns on certain users, and this can be effective in situations where only a few of user reviews are seen, namely providing a solution to the classic cold-start problem of recommender systems.

Another area of work is to dynamically learn the number of experience levels. In our experiments, we set E level to be five following the original expertise model specification. However, the number of stages that a user evolves through on a product category is likely dependent on the depth of the product. For example beer intuitively seems to be a more complex product group than bottled water, and this should be reflected in the user progression. Being able to learn the optimal number of experience levels could also lead to recommendation improvements.

6 Conclusion

In summary, this project successfully implemented an expertise-based recommender system and performed meaningful analysis that lead to three important findings. It proves the effectiveness and advantages of expertise-based model compared to latent factor model on a variety of datasets the expertise-based algorithm had not been tested on, and therefore claims with reasonable confidence that the expertise-based model is a superior recommender system when dealing with massive number of reviews on a product category with timestamp associated with each review. It also generalizes a typical expertise growth trajectory for average users that is common in different datasets. Lastly, our results suggest that individual expertise growth patterns are correlated across different product categories, and this correlation leaves hope for more accurate predictions on individual expertise gaining behavior and thus an even better performing expertise-based recommender system.

Acknowledgments

We thank the Spring 2014 Stanford CS341 course staff for providing the data we worked with, along with their guidance. In particular, we are indebted to both Jeff Ullman and Julian McAuley for their mentorship. We give permission for this document to be distributed to Stanford-affiliated students taking future courses.

References

[1] Ullman, Jeff, Jure Leskovec, Anand Rajaraman. 2012. *Mining of Massive Datasets.* Cambridge University Press. http://infolab.stanford.edu/ ullman/mmds/ch9.pdf.

[2] McAuley, Julian and Jure Leskovec. 2013. *From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews.* http://i.stanford.edu/ julian/pdfs/www13.pdf.

[3] Resnick, Paul, and Hal R. Varian. 1997. *Recommender systems.* Communications of the ACM 40.3 56-58.

[4] Adomavicius, Gediminas, and Alexander Tuzhilin. 2005. *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.* Knowledge and Data Engineering, IEEE Transactions on 17.6 734-749.

[5] Rashid, Al Mamunur, et al. 2002. *Getting to know you: learning new user preferences in recommender systems.* Proceedings of the 7th international conference on Intelligent user interfaces. ACM

[6] Lathia, Neal, et al. 2010. *Temporal diversity in recommender systems.* Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval.