Building the optimal Book Recommender and measuring the role of Book Covers in predicting user ratings

Cécile Logé
ceciloge@stanford.edu

Alexander Yoffe
ayoffe@stanford.edu

ABSTRACT

Many companies today, such as Amazon, Netflix, BestBuy, rely on Recommender Systems to launch targeted marketing campaigns and make customized suggestions to their users. According to a McKinsey study [8], 35% of purchases on Amazon and 75% of content watched on Netflix are driven by algorithmic recommendations. In this study, we use several approaches (Naïve Bayes, Item Affinity, Matrix Factorization) to build a Book Recommender based on data from Goodreads, and experiment with Neural Networks to measure the effect of the book cover on user ratings. Our best Recommender result was obtained with our optimized Matrix Factorization ALS model (described in section 3.3), with a resulting RMSE score of 0.834 (a 4.3% improvement when compared to the 2007 Netflix Prize winners RMSE score).

INTRODUCTION

Goodreads is a social platform combined with a book database quite similar to IMDb for movies: users can search for books, tag them, share their thoughts and discuss. Most importantly, they can rate books they have read on a scale from 1 to 5, and discover new books to read. Our goal is to build a Book Recommender such that for any given Goodreads user, we are able to predict future ratings on books they have not read yet, and use the predictions to surface 10 books we think they will love, possibly books they have never heard of. The input to our models ranges from book features (author, genre, length, year, snippet, but also the book cover) to previous ratings from the user and their peers. We use different methods (Naïve Bayes, Item Affinity, Matrix Factorization, Neural Networks) to output predicted ratings and when possible, to derive individual predicted rankings of books to recommend.

1. RELATED WORK

Most Recommender methods rely on content-based filtering (grouping items together based on their attributes) and collaborative filtering (clustering users into peer groups) to predict ratings and interest. The state of the art is a Collaborative Filtering technique known as Matrix Factorization and was made famous by Y. Koren, R. Bell, C. Volinsky (2007 & 2009) [1][2] winners of the 2007 Netflix Prize competition with a RMSE score of *0.8712* for predicting movie ratings (from 1 to 5): the existence of latent features in the model makes it possible to account for implicit variables and leads to higher accuracy and better scalability than standard clustering or k-nearest neighbors. Given the number of variables this technique implies, MF can be approached and optimized in many different ways: creative solutions such as combining ALS with Tikhonov regularization as done by Zhou et Al. (2008) [3] in their approach to the Netflix challenge, can lead to improved results.

When it comes to predicting values (ratings, revenues, prices) from a large set of features, Neural Networks are known to obtain superior results to Linear Regression or GDA[5]. In our case, we are particularly interested in the impact of images on the model's robustness, and were inspired by E. H. Ahmed, M. N. Moustafa (2016) [6] who used a 3-layer Neural Network to predict house prices, and managed to reduce RMSE by 99% by adding images to their previous text feature-only model. It's interesting to note however that using images alone led to poorer results. In this project, we build on these various methods to design new optimized models, and experiment with predicting ratings from features and book covers.

2. DATASET, FEATURES & EVALUATION METRICS

2.1 DATASET & FEATURES

We use data from Goodreads, and after a necessary cleaning (harmonizing, removing outliers, merging different datasets together), we have at our disposal:

- **finalbooks.csv:** book_id x metadata with the following fields: authors, title, year of publication, isbn, book snippet, genre. We let B be the total number of books (8,000).
- **finalratings.csv**: user_id x book_id x rating (a total of ≈ 2,000,000 ratings (score from 1 to 5) from 15,000 users). We have an average of 140 ratings per user (from min 62 to max 200), and 164 ratings per book (from min 31 to max 7280). We let U be the total number of users (15,000).
- **train/cross validation/test sets**: 65-15-20% on finalratings.csv making sure all users and books are in all three as some of the techniques (affinity) require this. Average number of ratings per user in the train set is 112 / in the test set is 28.
- **cover images:** for a subset of the books (5928), we have access to cover images in color and of size 50x74. When experimenting with covers and their relation to ratings, we shrink the images to a 32x32 resolution and flatten them inside the model creation sequence.

2.2 EVALUATION METRICS

Fundamentally, the goal of a Recommender is to **(1) predict a user's ratings** on books they have not read yet, and **(2) surface a ranked list of top *k* books** we believe they would love to know more about. In essence, this means the predicted ranking is just as key as the predicted rating itself when it comes to evaluating the relevance of our model.

- **(1) RMSE** (Root Mean Squared Error) on the predicted ratings themselves
- **(2) nDCG** (Normalized Discounted Cumulative Gain) [7] as each recommendation strategy generates a ranked list, and for a given list of k books, we compute the average **nDCG** among users as follows:

$$DCG_u = \sum_{i=1}^{k} \frac{2^{p_{ui}}-1}{\log_2(i+1)} \quad iDCG_u = \sum_{i=1}^{k} \frac{2^{r_{ui}}-1}{\log_2(j+1)} \quad nDCG = \frac{1}{U}\sum_{u=1}^{U} \frac{DCG_u}{iDCG_u}$$

where r_{ui} is the actual rating at the actual rank *i*, and p_{ui} the actual rating at our predicted rank i.

However, another goal of a Recommender is to **(3) help users discover relevant items they would not have found otherwise**. Focusing on nDCG or RMSE alone could lead us to neglect that aspect entirely by surfacing only popular/frequently rated items the user actually already knows, and potentially contributing to a vicious cycle of under-representation of less known items. Which is why we introduce another metric:

- **(3) Diversity score** defined as the average proportion of items in the Top k recommendations coming from the long tail of the dataset (less popular/less known books), and computed as follows:

$$Div_k = \sum_{u=1}^{U}\sum_{i=1}^{k} \frac{1\left\{b_i^{(u)} \in T\right\}}{kU}$$

where T is the predefined list of long tail items, and $b_i^{(u)}$ the predicted item for rank i. We find that 71% of the books in the dataset are behind 95% of the user ratings, and defined the Tail T as the other 29% (2298 books). We look specifically at the Top 10 (Div10) to evaluate the Diversity score of our different models.

3. METHODS & EXPERIMENTS

3.1 BASELINE MODELS: Popularity & Naïve Bayes

The ***Popularity-based model*** is a straight-forward and intuitive approach consisting in ranking items by their popularity among users (in our case, the number of ratings the books generated) and suggesting the top *k* to all users - excluding the ones they have already read. The predicted ratings are also the same for all users and match the popular vote (overall average ratings per book).

Each book in the dataset comes with a snippet - an excerpt or synopsis of the book taken directly from the back cover - giving us a very high-level idea of the key themes and topics addressed in the book, as well as possibly a first taste of the writing style. The ***Naïve Bayes approach*** allows us to break these book snippets into lists of words, remove punctuation and stop words (using the NLTK Corpus), and derive rating probabilities for each user (1 as should be recommended and likely to get a rating > 3, 0 as should not be recommended and likely to get a rating <= 3) based on the book vocabulary. We train our model on *each user individually* so as to get customized parameters. The resulting probabilities are then ranked, and rescaled into a predicted 1-5 score (3 being equivalent to the neutral probability 0.5).

3.2 ITEM x ITEM AFFINITY: TF-iDF

TF-iDF is an information retrieval technique to estimate the importance of a word w appearing in a book snippet b. It combines Term Frequency (tf_{bw} as # of times a word w appears in a snippet b divided by total words) with Inverse Document Frequency ($idf_w = log(B/df_w)$ where df_w is the number of snippets containing w, so the more frequent across snippets, the lower the score): $F_{bw} = tf_{bw} * idf_w$.

Using this concept, we represent our Book dataset as a TF-IDF matrix F of size B x W where W is the number of words in our dictionary (selection done based on a minimum idf of 2.9 leading to $W = 7623$). From there, we compute an affinity factor between books i and j in the form of Cosine Similarity s_{ij} and obtain a B x B Affinity Matrix. We then use these similarities as weights to predict user ratings as well as derive rankings with the idea that if book X was rated 5 by a specific user and is very similar to Y, chances are the user will rate it high as well.

$$s_{ij} = \frac{F_i^T F_j}{\|F_i\|_2 \|F_j\|_2} \qquad \hat{r}_{ub} = \frac{\sum_{i \in B_u} r_{ui} s_{ib}}{\sum_{i \in B_u} s_{ib}}$$

where B_u is the list of books user u already rated, meaning predicted ratings are weighted averages of known ratings from the train set.

3.3 COLLABORATIVE FILTERING: Matrix Factorization

Going deeper into Affinity and Collaborative Filtering, and using the concept of ***Matrix Factorization***, we are able to assume the existence of d latent features such that our U x B rating matrix R can be represented as the product of two lower-dimension matrices: Q of size U x d and P of size d x B. More specifically, the predicted rating from User u on Book b can be represented as:

$$\hat{r}_{ub} = q_u^T p_b$$

The challenge behind Matrix Factorization comes down to estimating the latent features. As we want to optimize the RMSE between actual and predicted ratings with regards to Q and P, we obtain the following Loss function to minimize, including regularization terms for both matrices:

$$L(P,Q) = \|R - \hat{R}\|_{\mathrm{F}} + \alpha\|Q\|_{\mathrm{F}} + \beta\|P\|_{\mathrm{F}} = \sum_{b=1}^{B}\sum_{u=1}^{U}(r_{ub} - \hat{r}_{ub})^2 + \alpha\sum_{u=1}^{U}\|q_u\|_2 + \beta\sum_{b=1}^{B}\|p_b\|_2$$

This problem is very similar to Linear Regression, except we have two sets of variables to estimate and two ways to look at the data (books or users). We are also faced with missing values (as books have not been individually rated by all users even though they've all been rated at least once, and the same holds for users). To tackle this, we look at $L(P,Q)$ as two different loss functions and use the Alternating Least Square method as an EM-like approach to our problem: the idea is to alternate between holding Q fixed and computing the minimum w.r.t P and holding P fixed and computing the minimum w.r.t Q.

In each iteration, we choose to use the Normal Equation, knowing the regularization terms will guarantee invertibility. First, we update P for each book using the update equation on the left, then we update Q for each user using the update equation on the right:

$$p_b := (\tilde{Q}^T\tilde{Q} + \beta I_d)^{-1}\tilde{Q}^T\tilde{R}_b \qquad q_u := (\tilde{P}\tilde{P}^T + \alpha I_d)^{-1}\tilde{P}\tilde{R}_u^{\ T}$$

where the tilde indicates a restriction to the books already rated by user u (for P) or to the users who already rated book b (for Q). The sizes of the resulting vectors, p_b and q_u are both dx1 for all books and users, and all elements are updated at each iteration.

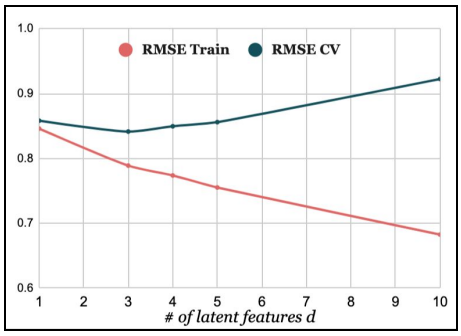
We experimented with parameters via Cross Validation with the goal of finding the optimal model w.r.t RMSE and overall nDCG:

(1) **Varying number of latent features:** we found $d = 3$ as the optimal number of latent features with regularization held fixed at 0.05. Beyond $d = 3$, our model was overfitting the Train set and RMSE on the CV set went back up as RMSE on the Train set went down (Fig.1).

(2) **Effect of regularization:** (Tables 1, 2) $\alpha = \beta = 0.1$ gave us the best results on the CV set after fixing $d = 3$

(3) **Effect of initialization:** (Table 3) Inspired by previous work in the field [1][3], and once we had settled on the parameters, we chose to experiment with different initialization techniques. We found that initializing P & Q with random

small numbers did not lead to improved results, and setting the first row of P as average ratings for the books did not help either. Instead, we went back to initializing them as standard matrices of $1/\sqrt{d}$ for all elements.



beta >	d = 3 / RMSE CV				
v alpha	0.075	**0.10**	0.125	0.15	0.20
0.075	0.8405	0.8416	0.8431	0.8442	0.8423
0.10	0.8386	**0.8383**	0.8396	0.8410	0.8417
0.125	0.8429	0.8458	0.8440	0.8454	0.8477
0.15	0.8432	0.8436	0.8430	0.8449	0.8503
0.20	0.8420	0.8407	0.8419	0.8423	0.8582

beta >	d = 3 / nDCG CV				
v alpha	0.075	**0.10**	0.125	0.15	0.20
0.075	0.9841	0.9839	0.9838	0.9837	0.9839
0.10	**0.9843**	**0.9842**	**0.9842**	0.9841	0.9841
0.125	0.9838	0.9836	0.9838	0.9836	0.9835
0.15	0.9838	0.9838	0.9839	0.9837	0.9836
0.20	0.9840	**0.9842**	0.9841	0.9840	0.9835

d = 3 , alpha = beta = 0.1			
initialization	**RMSE T**	**RMSE CV**	**DGC CV**
Random	**0.7956**	**0.8456**	**0.9838**
1/sqrt(d)	**0.7927**	**0.8383**	**0.9842**
1/sqrt(d) & avg ratings	**0.7997**	**0.8564**	**0.9830**
Random & avg ratings	**0.7905**	**0.8445**	**0.9839**

Left. Fig.1 RMSE as a function of d with regularization fixed at 0.05
Middle. Effect of Regularization on RMSE (Table 1. Top) & overall nDCG (Table 2. Bottom) on the CV set
Right. Table 3 Effect of Initialization on RMSE & nDCG on the Train & CV sets

3.4 NEURAL NETWORKS: Convolutional Neural Network & Multi-Layer Perceptron

In this section, we use ***Neural Networks*** to predict the average rating of a book from its features: author, title, number of pages, year of publication, and genre. We investigate the added effect of incorporating the image of the book cover into the features with three different models:
(1) A Multi Layer Perceptron for the numerical and categorical data only (8- and 4-neuron hidden layers + 1 output neuron with linear activation), (2) A Convolutional Neural Network for the images only, and (3) A combined Network from MLP and CNN for an improved model (concatenation of MLP and CNN outputs Fig.2).

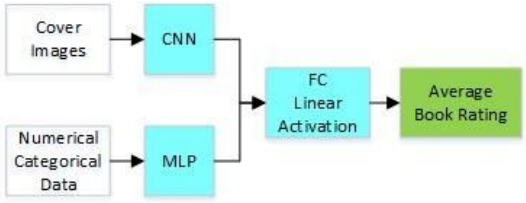


Fig.2 Combined NN Structure (3)

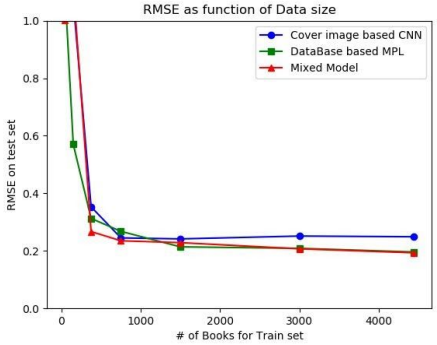


Fig.3 NN RMSE as a function of data size

As we observe a string sensitivity to the amount of input data, we plot the results of the predictions from each model as a function of size of data, i.e., number of books in the train set (Fig.3):

- From 50 books to the full train data set, we note an improvement of approximately 75% in RMSE on the test set. RMSE flattens after 1000 books.
- As we intuitively expect, the CNN model based on cover images only performs poorly compared to the other two, and confirms that a book cannot be fully judged on its cover.
- Finally, we see that initially the mixed model coincides with the image based model, giving less weight to the text features. However, starting at 1500 books, we observe a shift in performance and a slight overlap between the MLP model and the combined model, indicating less weight is given to images as data adds up. The combined model still benefits from the image input and gives the best results out of the three.

These results are consistent with the House Pricing experiment [6] mentioned earlier.

4. RESULTS, DISCUSSION & FUTURE WORK

Model	Set	RMSE	nDCG	nDCG Median	% nDCG = 1	Div10
Popularity	**Test**	**0.969**	**0.864**	**0.864**	**1.10%**	**0**
N. Bayes	Train	1.243	0.856			
	Test	**1.356**	**0.814**	**0.819**	**1.00%**	**0.285**
TF-iDF	Train	0.589	0.998			
	Test	**0.860**	**0.900**	**0.909**	**1.19%**	**0.300**
Matrix Fact.	Train	0.796	0.929			
	Test	**0.834**	**0.906**	**0.916**	**1.11%**	**0.189**

Neural Net.	Set	RMSE
CNN	Train	0.242
	Test	**0.247**
MLP	Train	0.045
	Test	**0.196**
Mixed	Train	0.038
	Test	**0.193**

Table 4. Summary of Results for Recommender models and Neural Networks

4.1 RESULTS & DISCUSSION

In our result table (Table 4), we focus on RMSE, nDCG and Div10, as well as the nDCG median across users and the proportion of users obtaining an ideal ranking (nDCG = 1). The idea is to confirm that the model's performance does not overly vary across users.

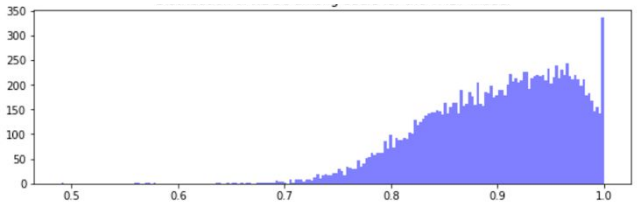


Fig.4 Distribution of nDCG (TFiDF model)

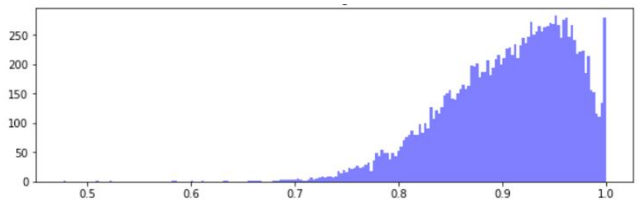


Fig.5 Distribution of nDCG (Matrix Factorization model)

Matrix Factorization turns out to be the best model to predict individual ratings (RMSE) and obtain the most ideal ranking. nDGC is also well-distributed among users with the highest median out of all the models. We did not expect to get such strong results with as few as $d = 3$ latent variables, and without a proper Cross Validation exercise, we would have intuitively gone for $d = 20$ or $d = 100$. However, with Div10 at 0.189, Matrix Factorization is also the least diverse model besides the Popularity baseline. Div10 = 0.189 is below the proportion of the Tail in our dataset at 0.29, which implies a popularity bias in the model.

TF-iDF gives better results than expected, with a nDCG very close to Matrix Factorization (and the best proportion of ideal rankings at 1.19%). This model also presents the second-best RMSE score. Given Div10 is 59% better than Matrix Factorization, we consider this model has the best in terms of overall performance.

Neural Networks gave strong results as well on predicting average ratings overall, with the Mixed Model combining both categorical/numerical features and images obtaining the lowest RMSE. It confirmed the intuition that including book cover images into the model could improve our results and help better predict future ratings, albeit only by an edge (1.5% better when compared to the MLP). The Convolutional Neural Network using only images did not perform as well with a RMSE score 26% lower than the MLP.

4.2 CONCLUSION & FUTURE WORK

Overall, we were able to achieve encouraging results, and we even surpassed the Netflix Prize winner RMSE score of *0.8712* by 4.3% with the optimized Matrix Factorization ALS model, and by 1.3% with the TF-iDF Affinity model. Our work led to interesting directions for future research with a solid potential to improve our results, in particular by focusing on:

- Optimizing the next version of our Matrix Factorization model by focusing on Diversity as well as the distribution of the performance across users, and on trying stronger regularization methods to test performance at a higher number of latent features,
- Building a hybrid model from our different approaches to combine each model's strength into a more robust recommender system,
- Leveraging the performance of Neural Networks and adapting them to individual users, making sure to take the book cover into account in the model.

Contributions:

Cécile and Alexander contributed equally. Everything was achieved jointly by both project partners.

Link to Project Code: https://github.com/cecileloge/cs229-RecoBook

References:

[1] Y. Koren, R. Bell, C. Volinsky ,"Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems" AT&T Labs Research, 2007

[2] Y. Koren, R. Bell, C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", IEEE, 2009

[3] Y. Zhou, D. Wilkinson, R. Schreiber, R. Pan, "Large-scale Parallel Collaborative Filtering for the Netflix Prize" AAIM, 2008

[4] G. Adomavicius, Y. Kwon, "Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques" IEEE, 2012

[5] R. Sharda, D. Delen "Predicting box-office success of motion pictures with neural networks", Expert Systems with Applications 30, 2006

[6] E. H. Ahmed, M. N. Moustafa "House price estimation from visual and textual features", NCTA 2016

[7] Jarvelin, Kekalainen, "Cumulated Gain-Based Evaluation of IR Techniques", ACM, 2002

[8] McKinsey "How retailers can keep up with consumers" (2013)

[9] Libraries and resources: Pandas, Numpy, Sklearn, Scipy, Matplotlib, NLTK Corpus, Keras, cv2 (open cv), urllib, pyimagesearch.com, fastml.com