Building a Book Recommender

Cécile Logé (ceciloge@stanford.edu) & Alexander Yoffe (ayoffe@stanford.edu) / CS 229 Fall 2019

Stanford University

MOTIVATION & DATASET

Goodreads is a social platform where users can discuss and rate books on a scale from 1 to 5. We want to build a Book Recommender and find an efficient way to predict book ratings.

Dataset:

- 8,000 books (B): snippet, genre, # of pages, year, authors, title, book cover
- 2m ratings from 15,000 users (U) (average of 140 ratings per user)
- 71% of the books in the dataset are behind 95% of the user ratings. We will define the Tail (T) as the other 29% (2298 books)

The dataset is split between train (65% + 15% for cross validation) and test (20%).

EVALUATION METRICS

Building a Book Recommender can be divided into three core goals each evaluated by a key metric:

(1) Predict a user's ratings on books they haven't read yet (**RMSE**)
(2) Surface a ranked list of top k books for each user (**nDCG**)[1]:

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

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

(3) Help users discover relevant items (*Top 10*) from the Tail, that they would not have easily found otherwise (**DivScore** / all books but train set):

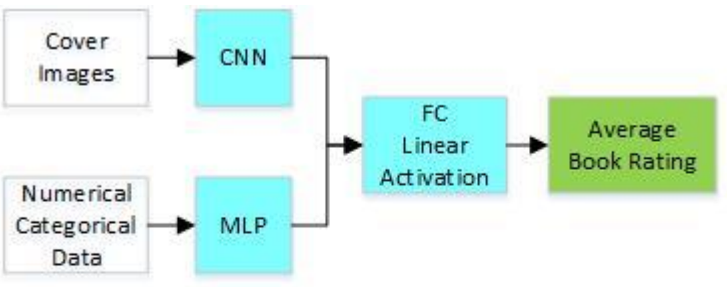
$$Div_k = \sum_{u=1}^{U}\sum_{i=1}^{k} \frac{1\left\{b_i^{(u)} \in T\right\}}{kU}$$

where $b_i^{(u)}$ is the book at *predicted* rank i for user u

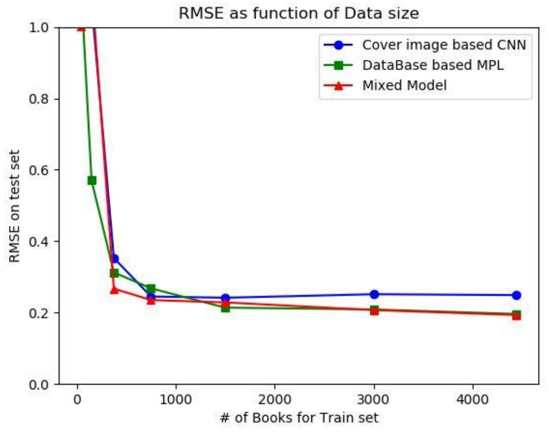
NEURAL NETWORKS

Neural Networks were used to **predict the average rating** of a book based on the following input: Cover Image, Author, Title, Genre, Year published, # of pages

- **CNN:** Convolutional Neural Network for a model based on cover images
- **MLP:** Multi-Layer Perceptron used to handle 3 discrete (Categorical) variables and 2 continuous (Numerical) variables(8 and 4 neuron hidden layers + output neuron with linear activation)
- **Mixed Model:** concatenated NN outputs combining both the CNN and MLP and using both images and data as input



NN Architecture / based on [4]



REFERENCES:
[1] Cumulated Gain-Based Evaluation of IR Techniques / Jarvelin & Kekalainen (2002)
[2] Matrix Factorization Techniques for Recommender Systems / Koren, Bell & Volinsky (2009)
[3] Large-scale Parallel Collaborative Filtering for the Netflix Prize / Zhou, Wilkinson, Schreiber & Pan (2009)
[4] pyimagesearch.com

BASELINE MODELS

- **Popularity Model**

Recommending the most popular books to all users. Efficient but contributes to a vicious cycle of under-representation of less known books.

- **Naïve Bayes Approach**

We broke Book snippets into lists of words after removing punctuation and stop words. This allowed us to use NB and derive probabilities for each user to like the book based on its vocabulary and deduce predicted rankings & ratings.

Average token weight among users			
Likely Good Rating		**Likely Bad Rating**	
harry	0.1925	irrevocably	-0.1034
one	0.1751	grey	-0.1030
world	0.1671	aspect	-0.0968
death	0.1599	twilight	-0.0957
boy	0.1562	agency	-0.0949

ITEM x ITEM AFFINITY MODEL & TF-iDF

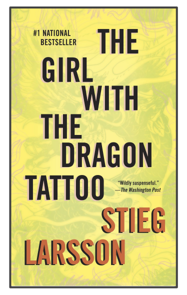
TF-iDF is an information retrieval technique to estimate the importance of a word w in a book b (snippet). It combines ***Term Frequency*** (tfw = # of times w appears in b divided by total words) with ***Inverse Document Frequency*** ($idfw = \log(B/dfw)$ where dfw is the number of books containing w):

$F_{bw} = tf_{bw} * idf_w$ ⟶ F is a B x W matrix ⟶ ***Cosine similarities*** s_{ij} between books i, j into a B x B matrix

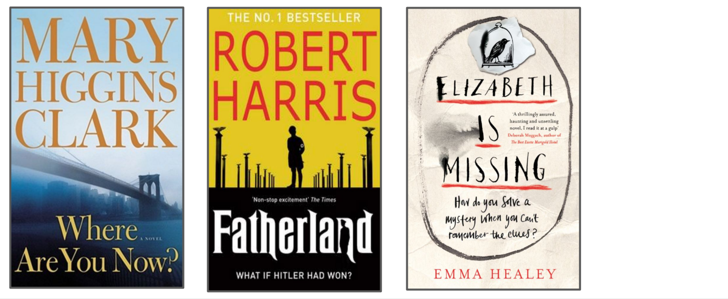
Item x Item Affinity: using similarities between books as weights to predict ratings as well as derive rankings with the idea that if book X was rated 5 by a specific user and is very similar to Y, chances are the user will rate it high as well. Predicted ratings as a weighted average of known ratings.

$$s_{ij} = \frac{F_i^T F_j}{\|F_i\|_2 \|F_j\|_2}$$

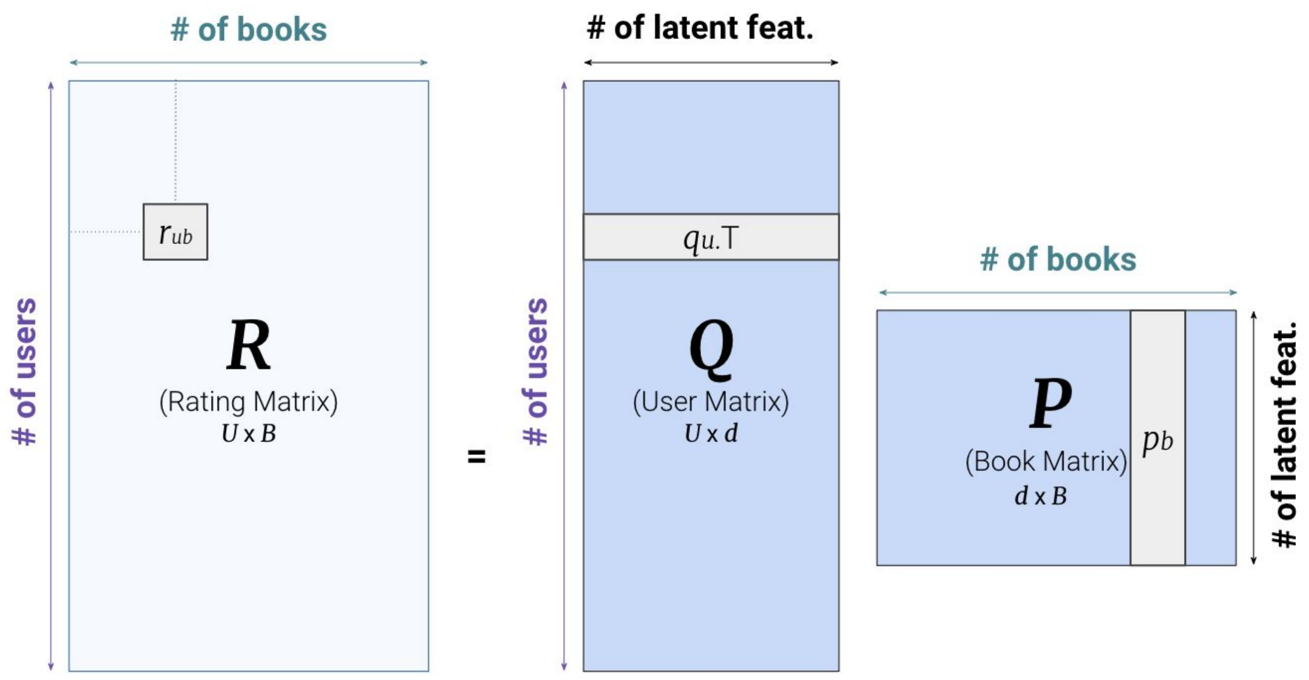
$$\hat{r}_{ub} = \frac{\sum_{i=1}^{k} r_{ui} s_{ib}}{\sum_{i=1}^{k} s_{ib}}$$

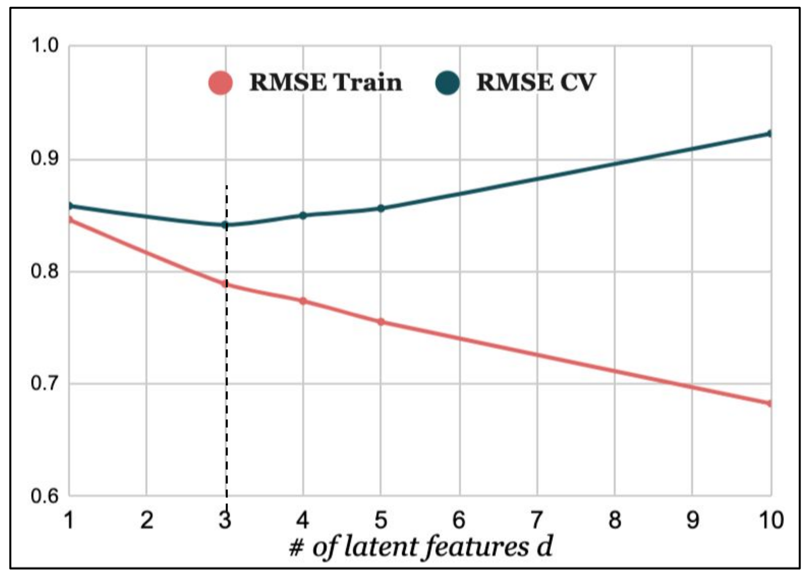


Top 3
Most Similar Books
⟶



COLLABORATIVE FILTERING & MATRIX FACTORIZATION





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 / DCG 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

Matrix Factorization [2] consists in assuming there exist d latent features that can allow us to approach our U x B rating matrix R as the product of two matrices: Q of size U x d (users) and P of size d x B (books).

- More specifically, the **predicted rating from User u on Book b is represented as:**

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

- **Loss Function:** we want to minimize the RMSE between ratings and prediction with regards to Q and P, adding regularization terms for both:

$$\arg\min_{Q,P} \|R - \hat{R}\|_F + \alpha\|Q\|_F + \beta\|P\|_F$$

- **ALS (Alternating Least Squares):** we optimized loss by approaching it as **two different cost functions**. We alternated between holding Q fixed and computing the minimum w.r.t P and holding P fixed and computing the minimum w.r.t Q (using the Normal Equation on each user/book).
- **Parameter Optimization:** we found through Cross Validation that $d = 3$ and $\alpha = \beta = 0.1$ were optimal for RMSE as well as overall DGC.
- **Initialization:** we tried several initialization techniques [3] for Q and P and found through Cross Validation that the most efficient one was to initialize them to standard matrices of $1/\mathrm{sqrt}(d)$.

RESULTS

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

DISCUSSION & FUTURE WORK

Matrix Factorization turned out to be the best model to predict individual ratings (RMSE) and obtain the most ideal ranking (highest nDGC with satisfying distribution among users). Div10 = 0.189 implies a more balanced model than the Popularity Baseline, but below the Tail proportion of our dataset (0.29).
Neural Networks results confirmed the intuition that the content of the book is a better predictor of ratings and book quality than the cover image alone. Results were still good for all three and the mixed model provided the best results.

Next steps: An hybrid model from all our different approaches could be an interesting way to combine each model's strength into a robust recommender system. We would also work on leveraging the Neural Network model to individual users.

Link to our Poster Video

Complete link: https://www.dropbox.com/s/egydg2dsir3o956/Poster%20CS229%20Book%20Reco.mov?dl=0