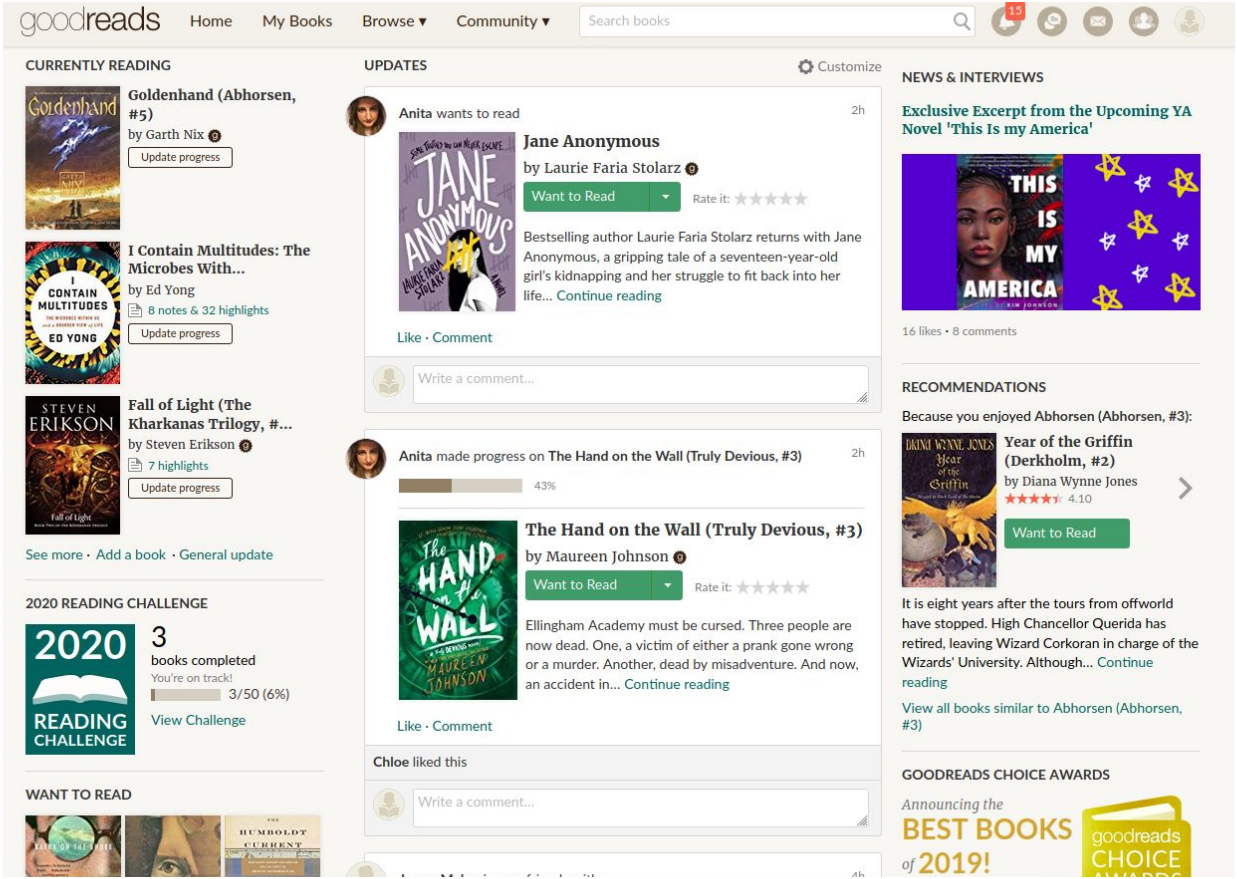
NoveList

Find your next page turner

Meghan Thommes
Health Data Science Fellow

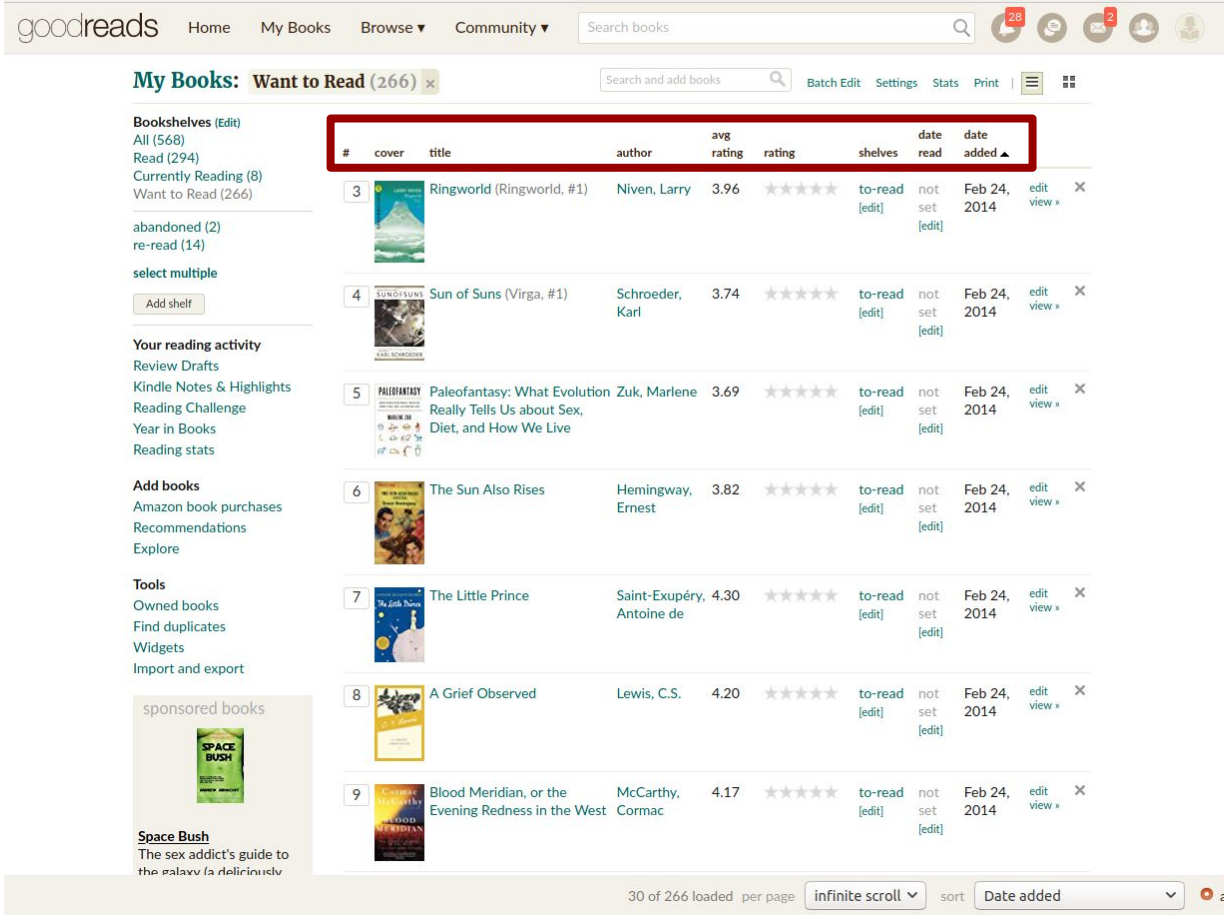
goodreads: A website for book lovers



Which book should a user read next?

Can sort by

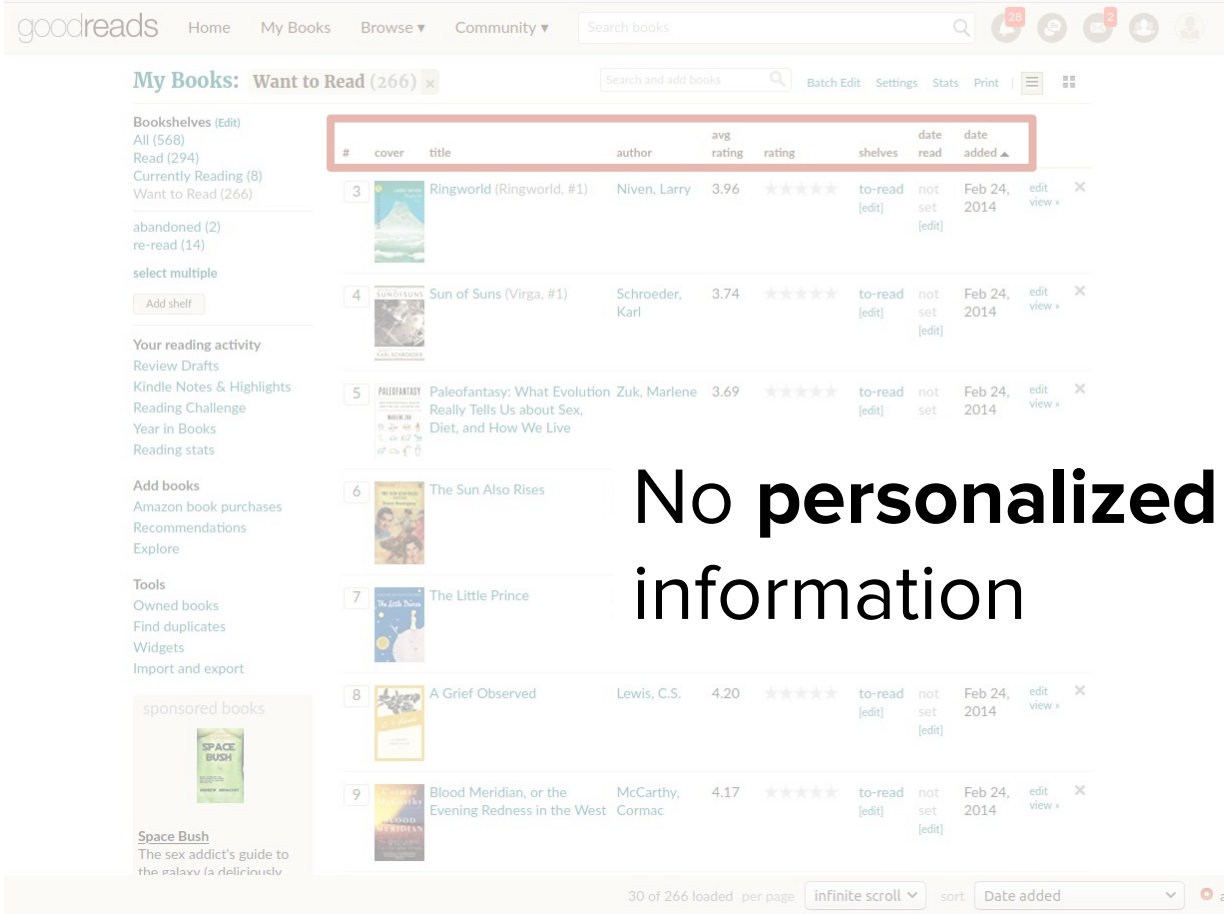
- Average Rating
- Date Added
- Author
- Title
- *etc*



Which book should a user read next?

Can sort by

- Average Rating
- Date Added
- Author
- Title
- *etc*



No **personalized** information

NoveList

https://insight-novelist.herokuapp.com/

Find your next page turner

This app ranks your "To-Read" books on Goodreads

Would you like to upload a CSV of your exported Goodreads library?

Upload your Goodreads data?

- (•) Yes, upload my own Goodreads data
- ( ) No, use pre-loaded data

Upload your Goodreads library:

Upload your exported Goodreads Library CSV file

Drop files here to upload
or
browse files

Obtain data on how **users** rate **books**

Data cleaning

goodreads
UCSD Book Graph
15.7M reviews

{JSON}
16.7GB

2GB

UCSD Book Graph

Scraped Goodreads users' public shelves in late 2017

- 2.1M books
- 465k users

Most books are rated **3 stars** or more

Data cleaning → Feature extraction

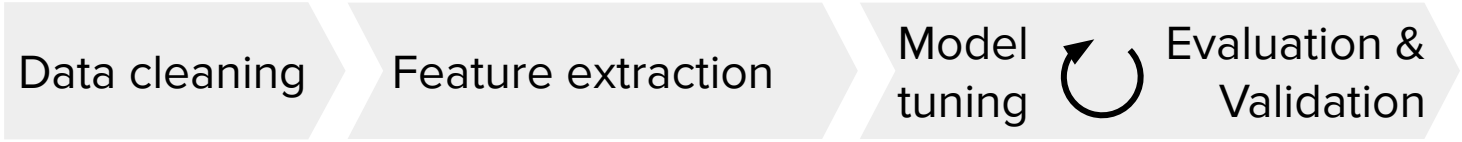
goodreads
UCSD Book Graph

15.7M reviews

↓

9.3M reviews → Book Ratings

Use **collaborative filtering** to predict book ratings



goodreads

UCSD Book Graph

15.7M reviews

9.3M reviews → Book Ratings → ?

Collaborative-based

Sometimes the simplest method is the best

Model	Root Mean Square Error (# stars)
Baseline (Alternating Least Squares)	0.988
Baseline (Stochastic Gradient Descent)	0.990
Matrix Factorization (Singular Value Decomposition)	1.027

The error is not great…but neither was Netflix's

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

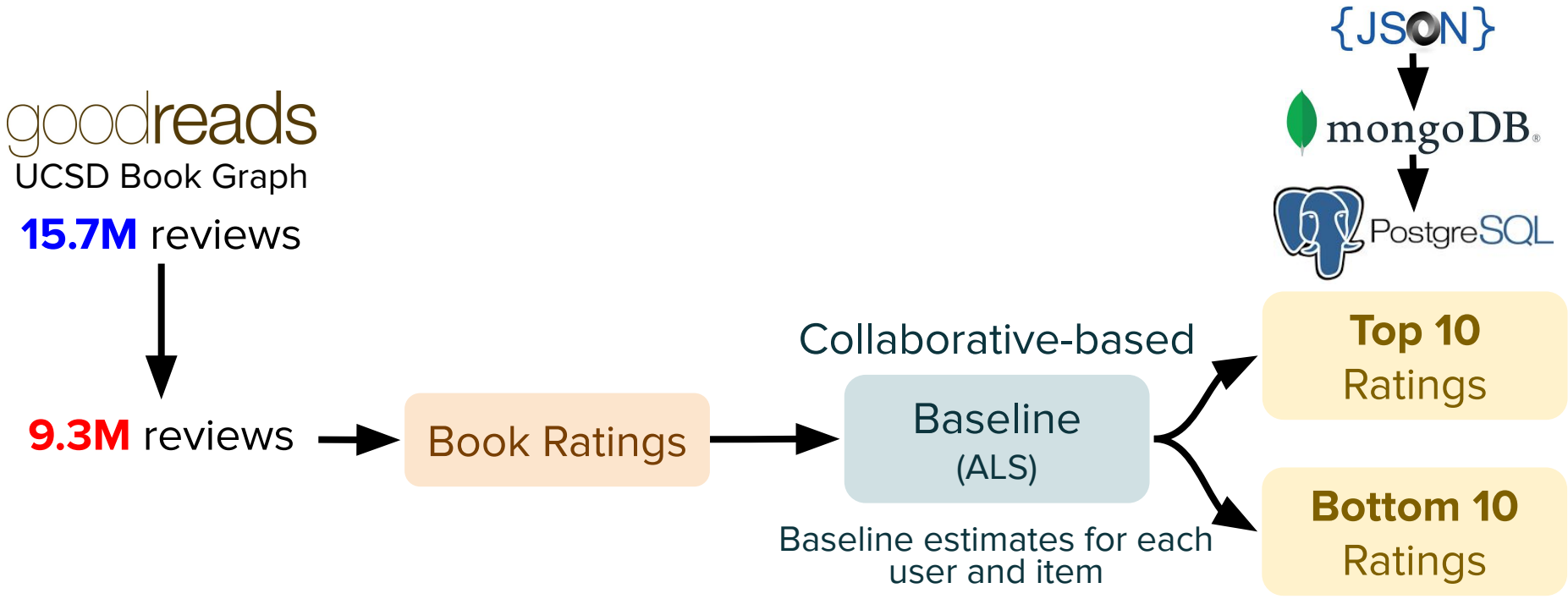
Best RMSE = 0.95 → 0.86

…but the additional accuracy gains did not justify the effort needed to bring them into production

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40

Use **collaborative filtering** to predict book ratings

Data cleaning → Feature extraction → Model tuning ⟳ Evaluation & Validation → Use model to predict ratings



NoveList gives users a **personalized** experience

User A

Your top 10 ranked books are:

	Title	Author
0	When Breath Becomes Air	Paul Kalanithi
1	Homo Deus: A History of Tomorrow	Yuval Noah Harari
2	Algorithms to Live By: The Computer Science of Human Decisions	Brian Christian
3	Never Let Me Go	Kazuo Ishiguro
4	Seveneves	Neal Stephenson
5	The Sixth Extinction: An Unnatural History	Elizabeth Kolbert
6	The Winds of Winter (A Song of Ice and Fire, #6)	George R.R. Martin
7	Anne of Green Gables (Anne of Green Gables, #1)	L.M. Montgomery
8	The Little Prince	Antoine de Saint-Exupéry
9	Animal Farm	George Orwell

★★★★

User B

Your bottom 10 ranked books are:

These books may be ranked low because you have not read similar books

	Title	Author
0	The Relic Master	Christopher Buckley
1	You're Never Weird on the Internet (Almost)	Felicia Day
2	The Savage Detectives	Roberto Bolaño
3	The Stone Sky (The Broken Earth, #3)	N.K. Jemisin
4	Love, Nina: Despatches from Family Life	Nina Stibbe
5	A Face Like Glass	Frances Hardinge
6	Sacred Games (Sacred Games)	Vikram Chandra
7	American War	Omar El Akkad
8	Human Voices	Penelope Fitzgerald
9	Love & Gelato	Jenna Evans Welch

★★★★★

Additional Slides

Collaborative filtering to make personalized predictions

👤	5		4			4
👤		3		3		2
👤			5		5	5
👤	4		5		4	
👤	4				2	

Personalized predictions are **hard**

Every person is **unique** with a variety of interests

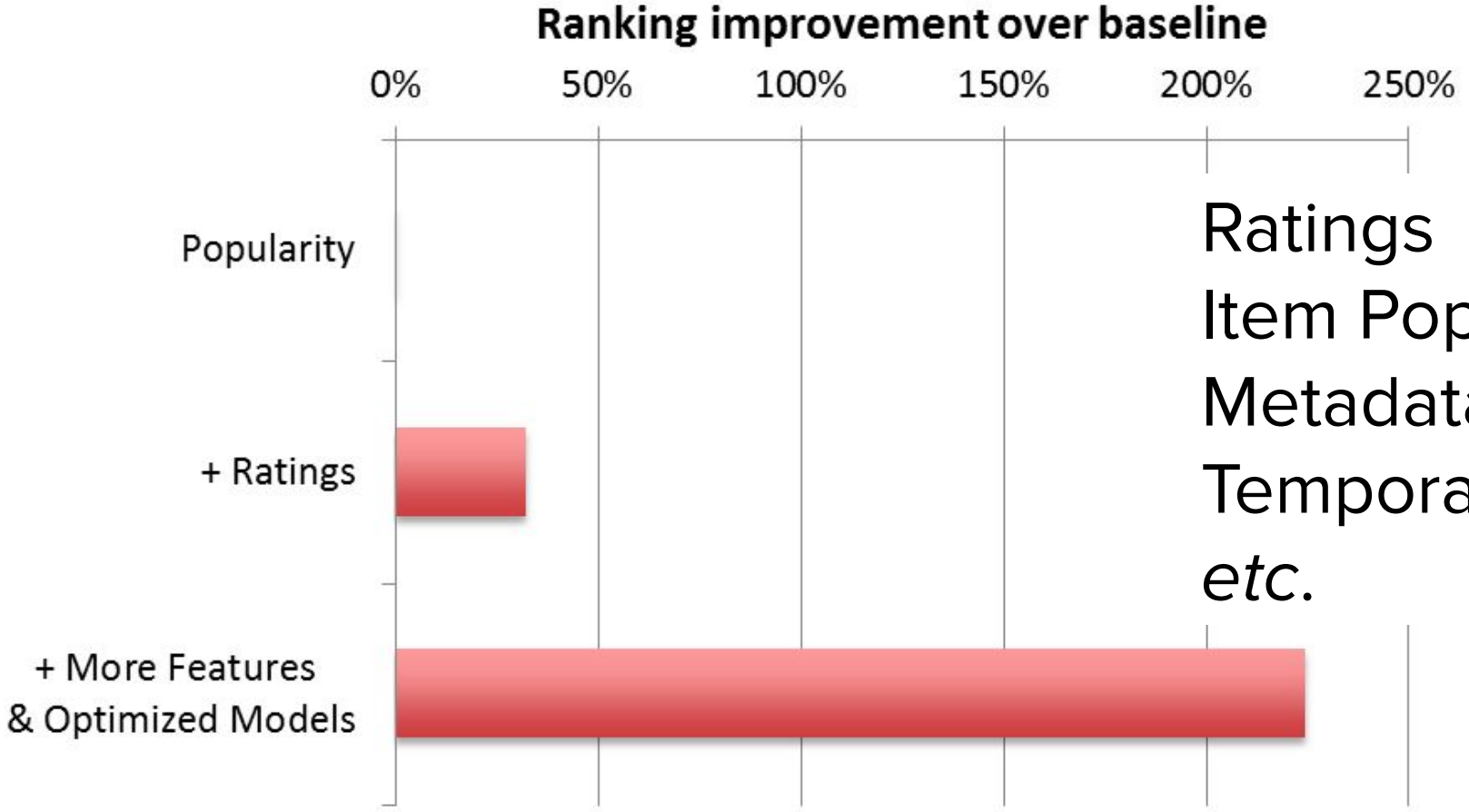
Data Sparsity: Have **large datasets**, but a small amount of **data per user**

Cold-start: Cannot make a prediction if a user or book does not have **enough ratings**

Enjoyment depends on mood, context, ...

...or a if user just wants something new, fresh, etc

Adding **features** is the best way to improve rankings



Ratings
Item Popularity
Metadata
Temporal Data
etc.

Potential features for **content-based** filtering

goodreads

Author(s)

Genre(s)

Publisher

Year Published

Number of Pages

Duration (*how long it takes to read*)

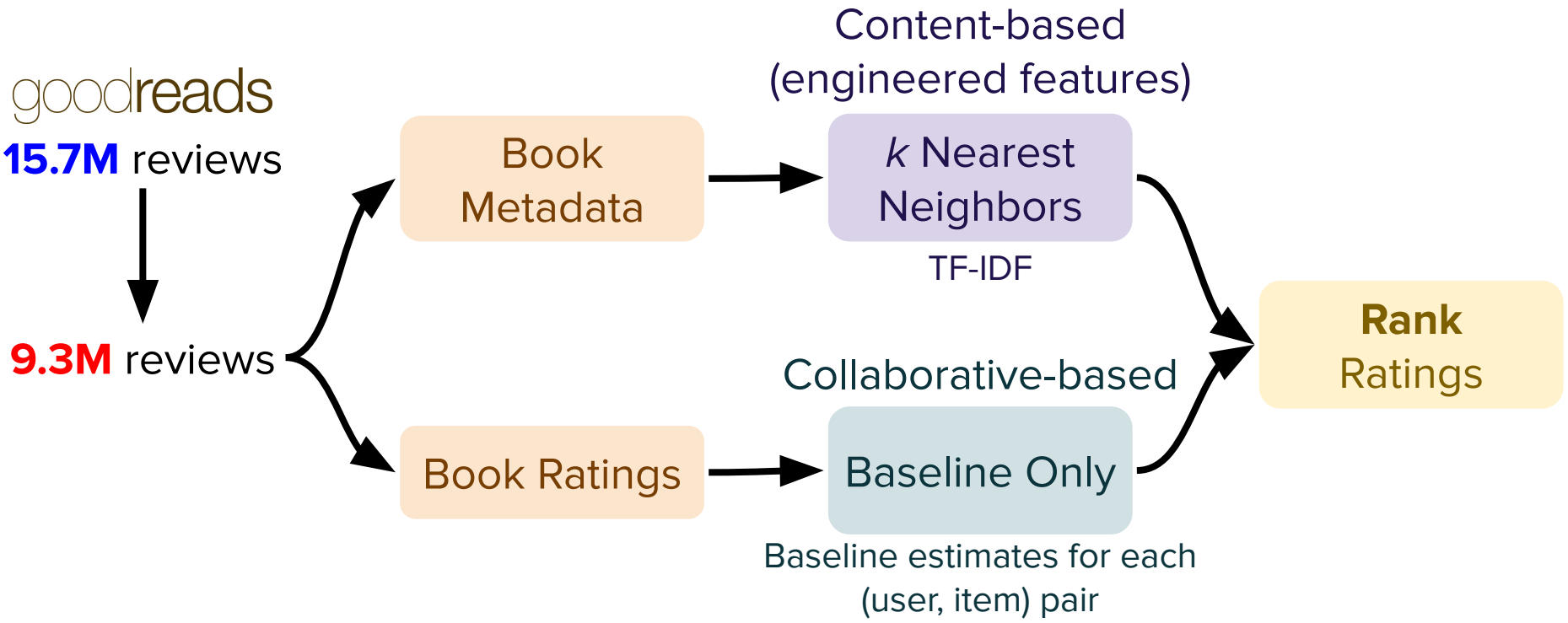
Aggregates bibliographic information into subject terminology schemas

FAST subject headings

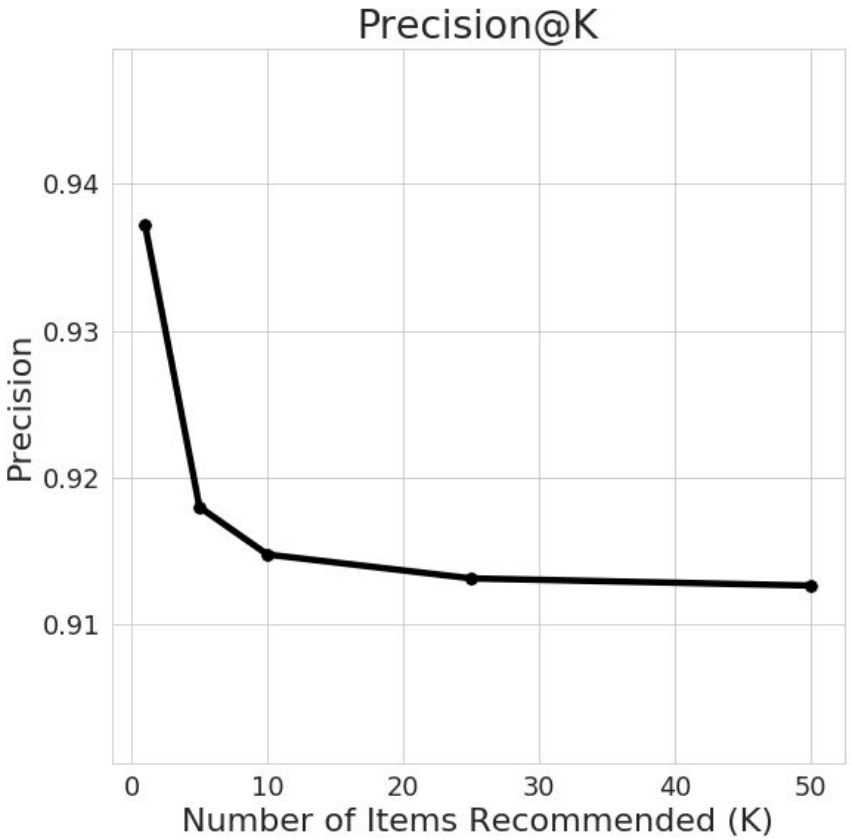
- Faceted Application of Subject Terminology

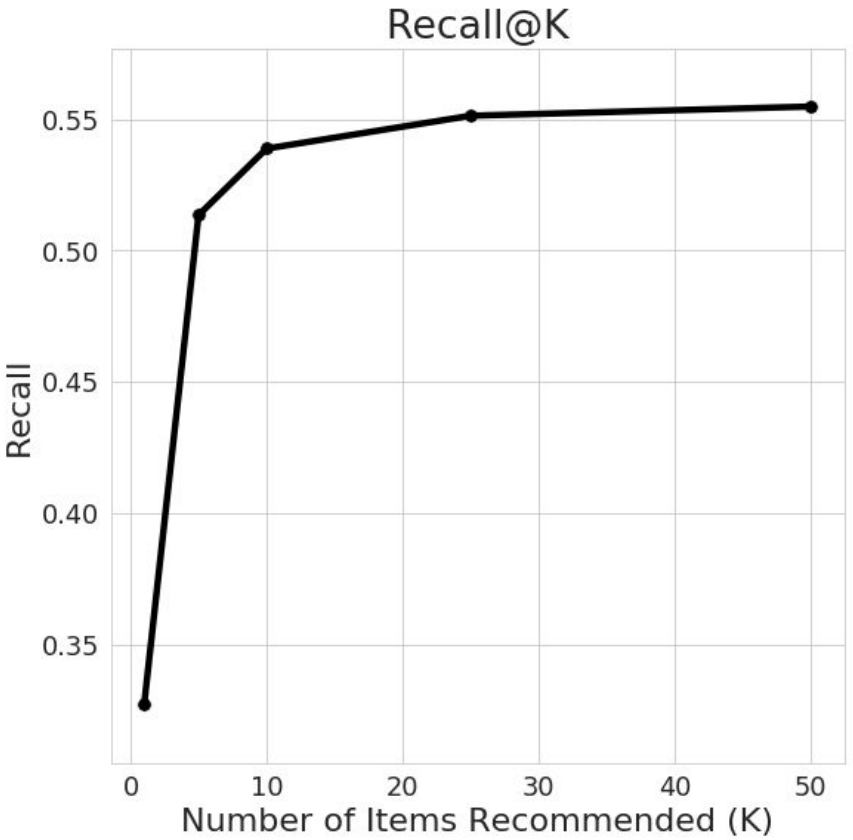
Recommender system to predict book ratings

Data cleaning → Feature extraction → Model tuning ⟳ Evaluation & Validation → Use model to predict ratings

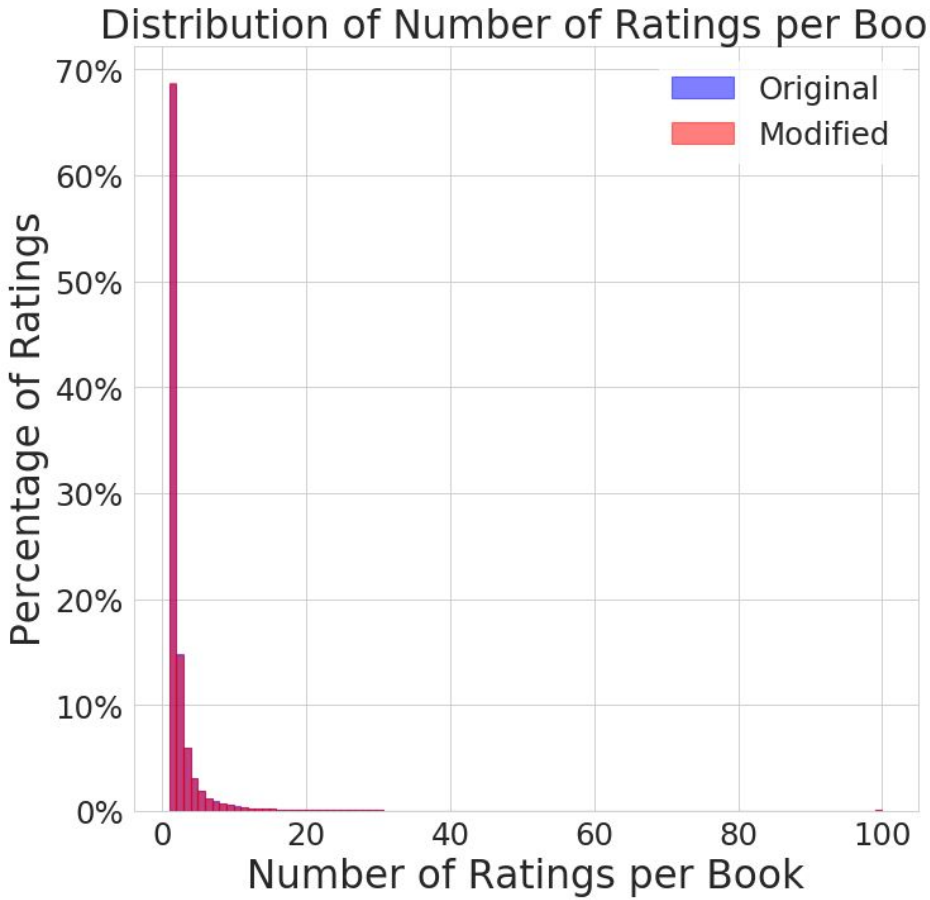


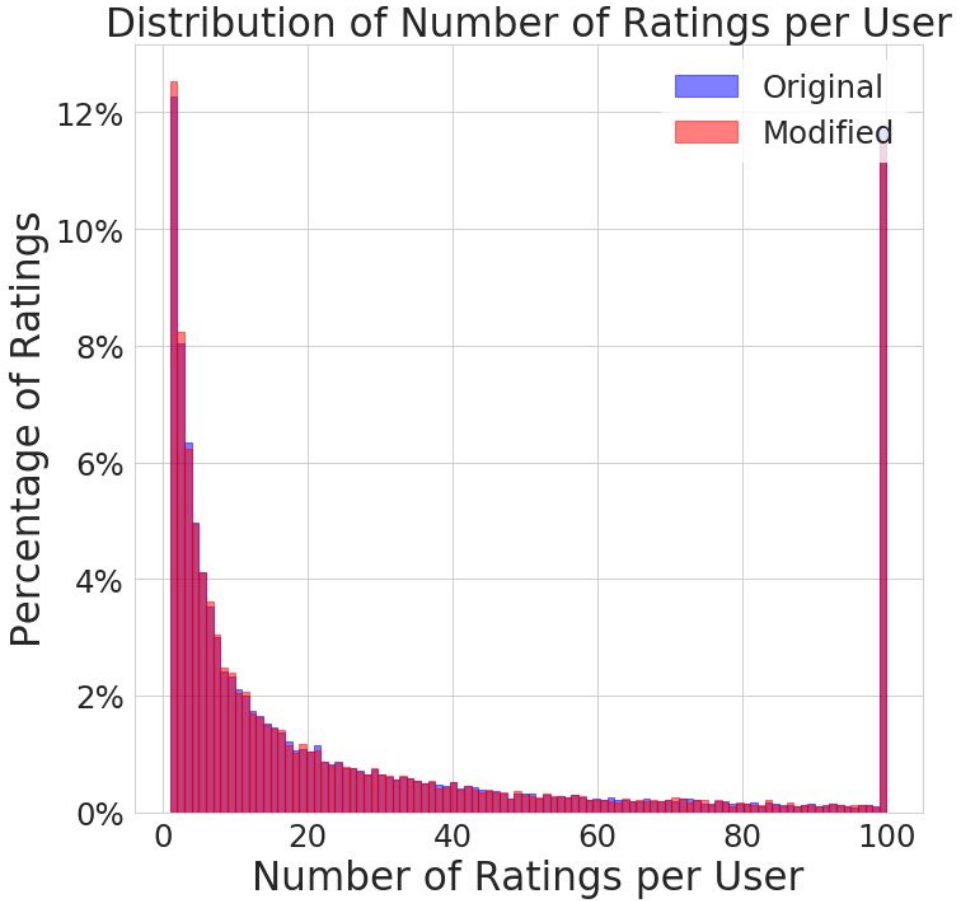
Precision and recall at cutoff *K*



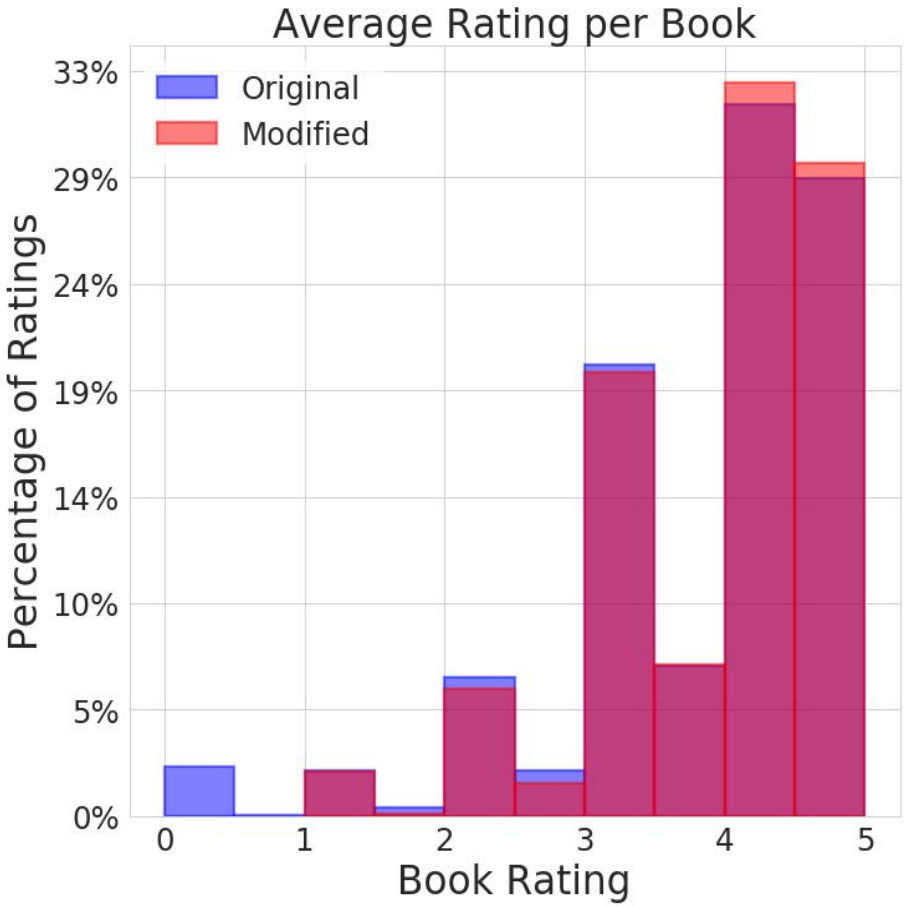


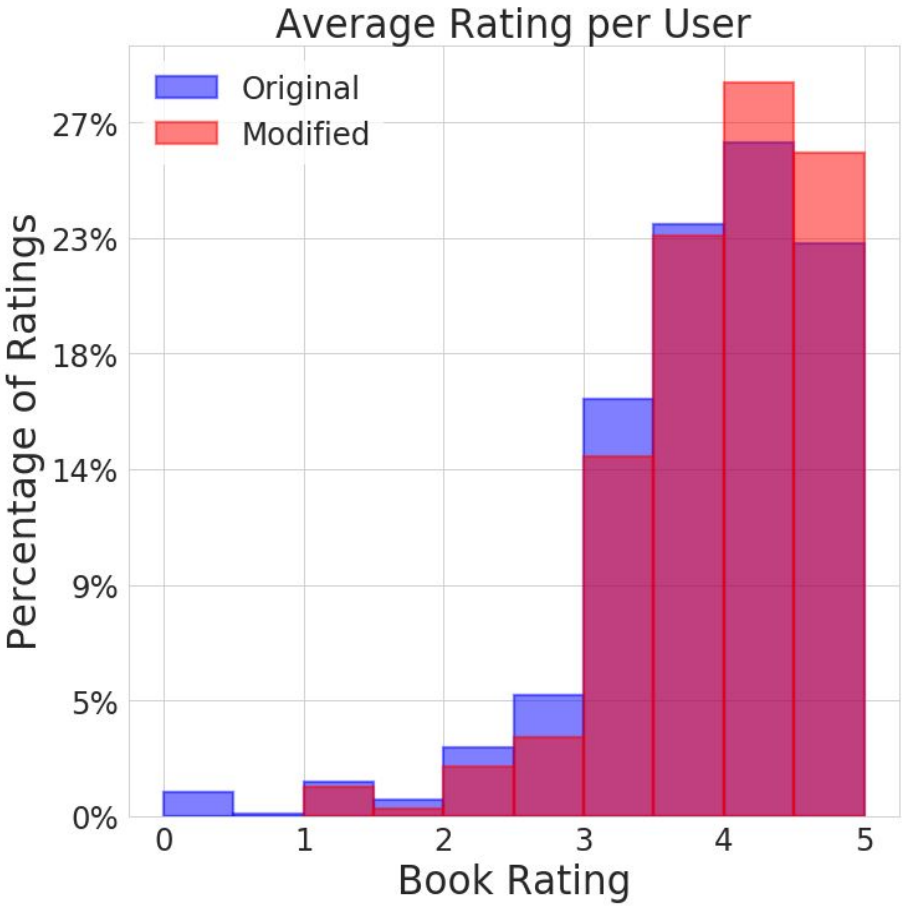
Histograms: Number of Ratings





Histograms: Average Ratings





Why not the Goodreads API?

✔️ Rest API

✖️ Opens up users to some security vulnerabilities

✖️ Python wrappers are not well supported (developed as a side project by non-employees)

✖️ OAuth integration is not well documented (and runs into errors)

Method Overview: Baseline

Rating is predicted based on the baseline estimate for each user and book:

$$\hat{r}_{user,book} = b_{user,book} = \mu + b_{user} + b_{book}$$

Minimizing the regularized square error:

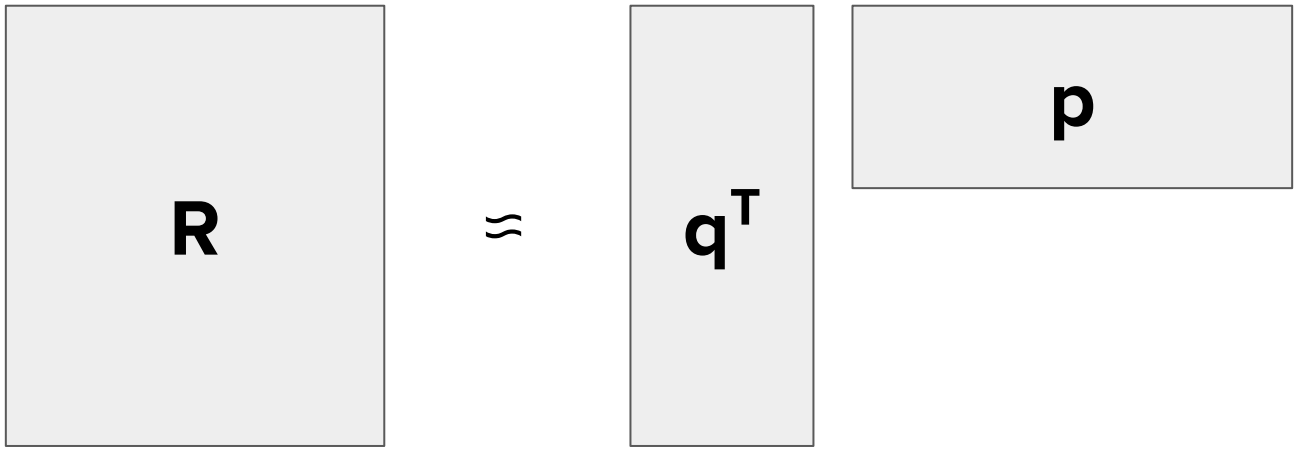
$$\sum_{r_{user,book} \epsilon R_{train}} (r_{user,book} - (\mu + b_{user} + b_{book}))^2 + \lambda_{user} b_{user}^2 + \lambda_{book} b_{book}^2$$

Can use stochastic gradient descent (SGD) or alternating least squares (ALS) to minimize the error

surprise

Method Overview: Matrix Factorization

Find **latent features**



surprise

Method Overview: Matrix Factorization

Rating is predicted based on the baseline estimate for each user and book and the latent book and user factors:

$$\hat{r}_{user,book} = b_{user,book} = \mu + b_{user} + b_{book} + q_{book}^T p_{user}$$

Minimizing the regularized square error:

$$\sum_{r_{user,book} \epsilon R_{train}} (r_{user,book} - \hat{r}_{user,book})^2 + \lambda(b_{book}^2 + b_{user}^2 + \|q_{book}\|^2 + \|p_{user}\|^2)$$

Use stochastic gradient descent (SGD) to minimize the error

surprise

Method Overview: Matrix Factorization

Singular Value Decomposition (SVD)

$$p_{user} \leftarrow p_{user} + \gamma(e_{user,book} \cdot q_{book} - \lambda p_{user})$$

$$q_{book} \leftarrow q_{book} + \gamma(e_{user,book} \cdot p_{user} - \lambda q_{book})$$

Non-Negative Matrix Factorization (NMF)

- Factors are non-negative

$$p_{user,factor} \leftarrow p_{user,factor} \cdot \frac{\sum_{book \epsilon B_{user}} q_{book,factor} \cdot r_{user,book}}{\sum_{book \epsilon B_{user}} q_{book,factor} \cdot \hat{r}_{user,book} + \lambda_{user} |B_{user}| p_{user,factor}}$$

$$q_{book,factor} \leftarrow q_{book,factor} \cdot \frac{\sum_{user \epsilon U_{book}} p_{user,factor} \cdot r_{user,book}}{\sum_{user \epsilon U_{book}} p_{user,factor} \cdot \hat{r}_{user,book} + \lambda_{book} |U_{book}| q_{book,factor}}$$

surprise

Method Overview: Clustering (*k*-NN)

Use similarity between users or items

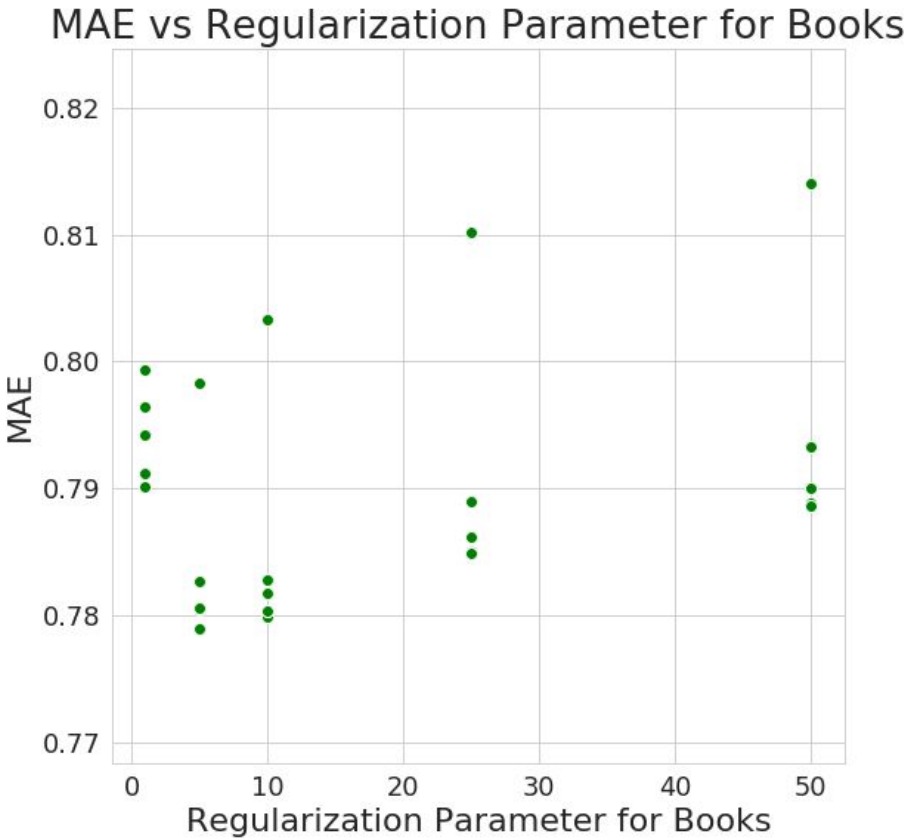
User-centric: $\hat{r}_{user,book} = \frac{\sum_{v \epsilon N^k_{book}(user)} \text{sim}(user, v) \cdot r_{v,book}}{\sum_{v \epsilon N^k_{book}(user)} \text{sim}(user, v)}$

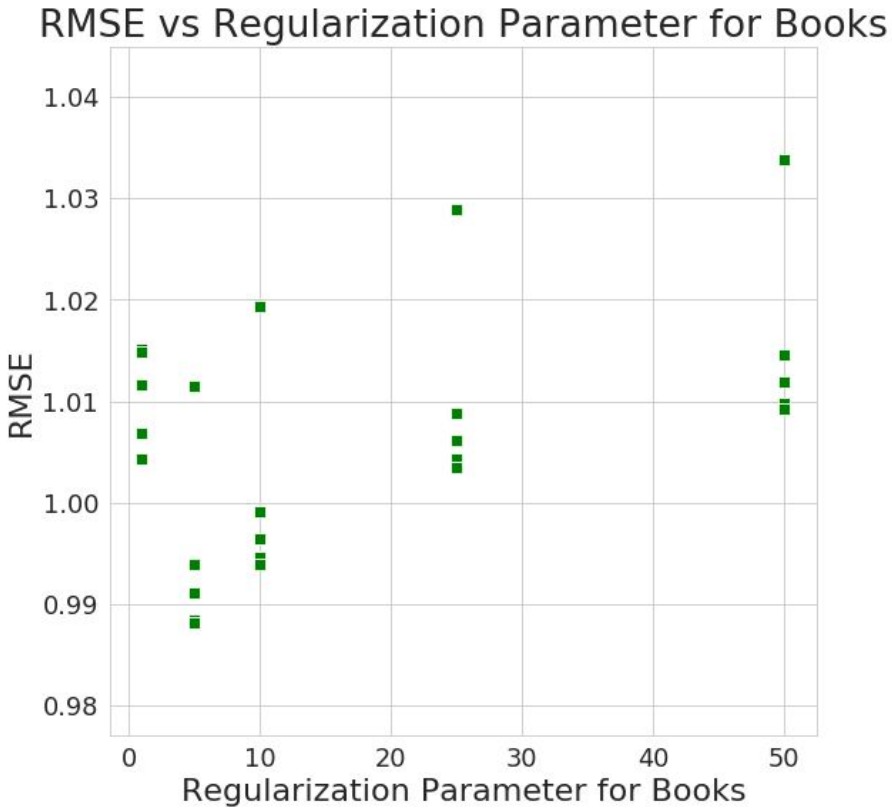
Item-centric: $\hat{r}_{user,book} = \frac{\sum_{j \epsilon N^k_{user}(book)} \text{sim}(book, j) \cdot r_{user,j}}{\sum_{j \epsilon N^k_{user}(book)} \text{sim}(book, j)}$

Can also offset by mean or baseline

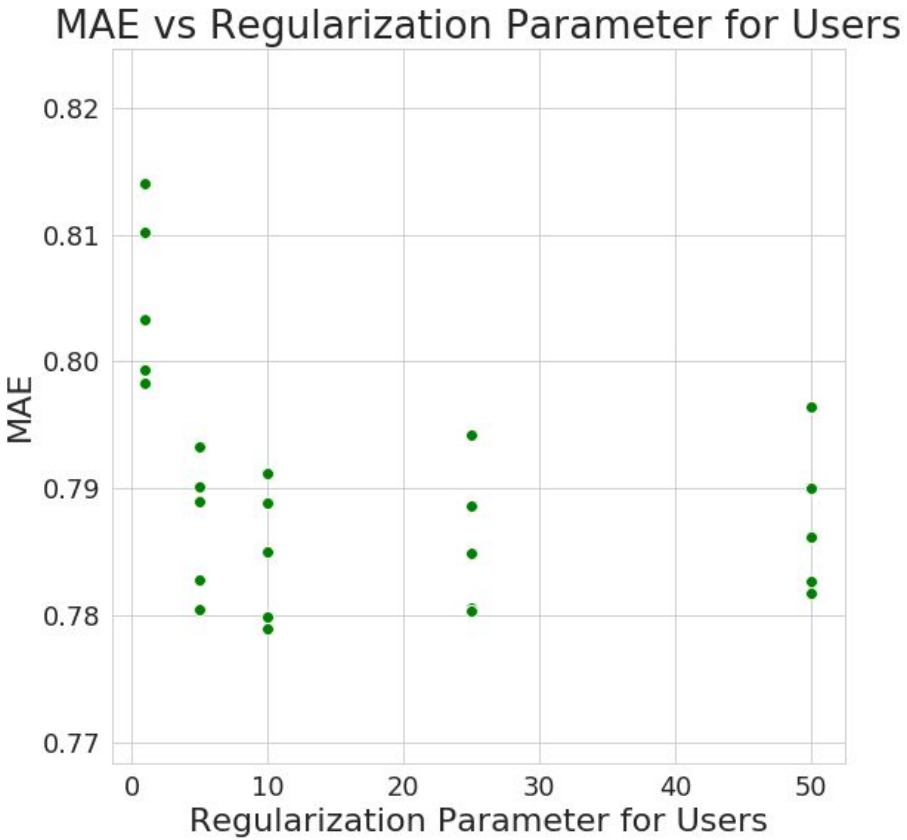
surprise

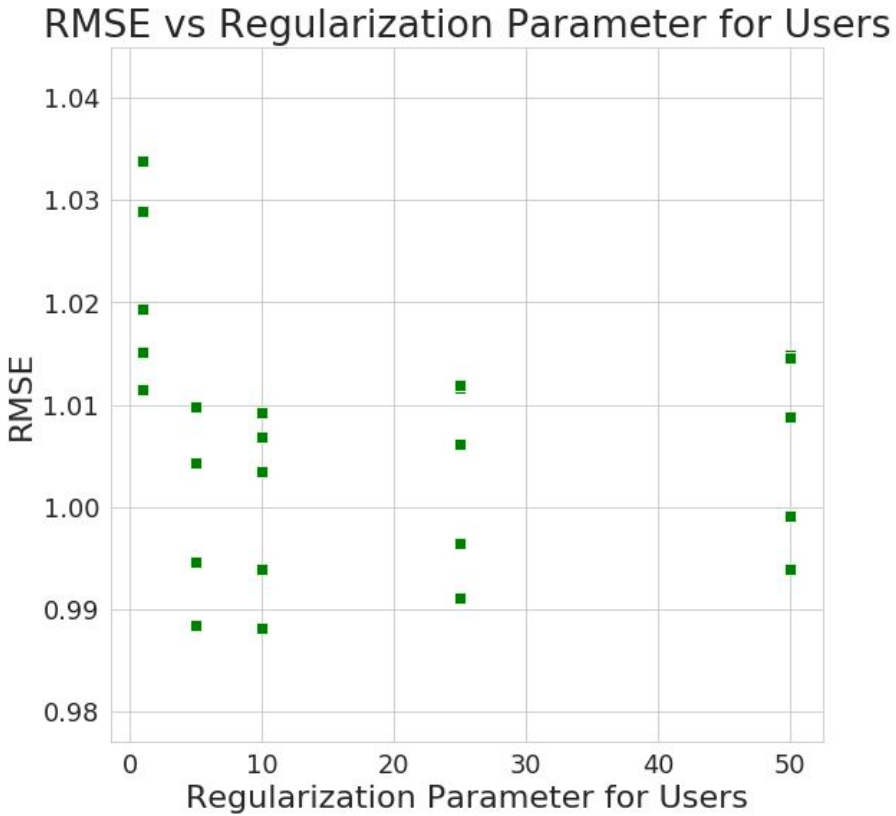
Parameter Tuning: Baseline (λ_{books})



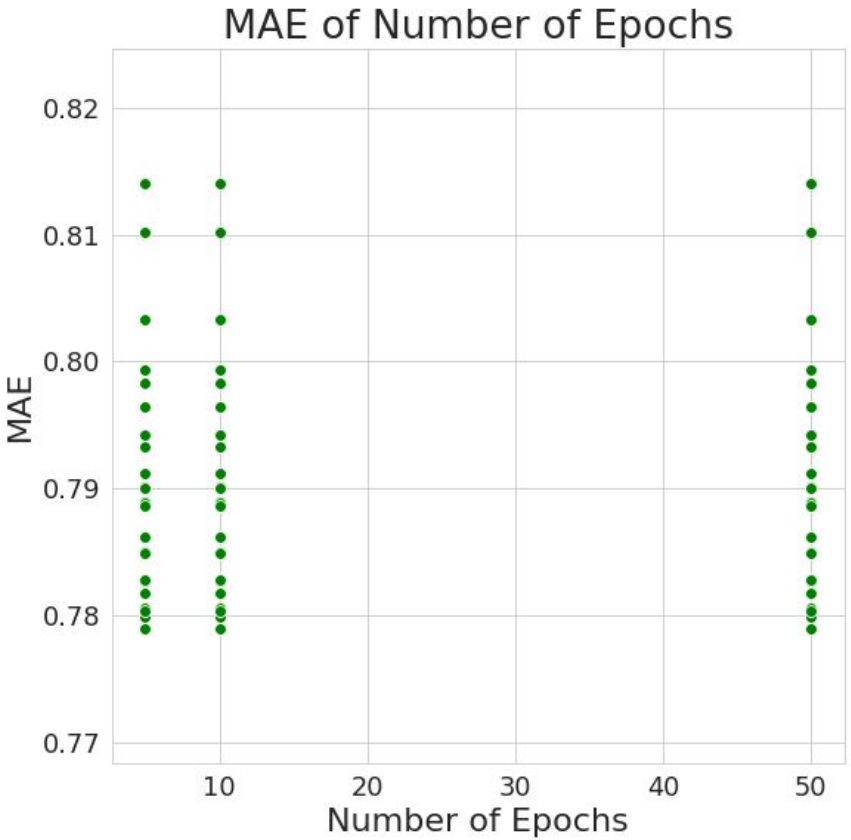


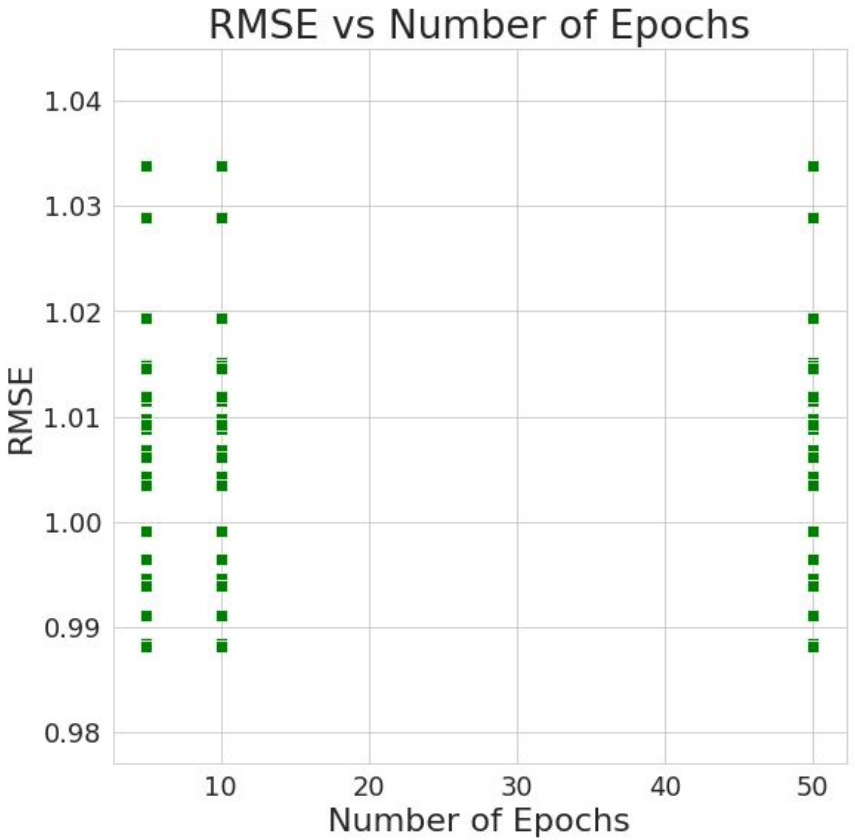
Parameter Tuning: Baseline (λ_{users})





Parameter Tuning: Baseline (n_{epochs})





Assessing Models