Why Not Just Recommend Popular Books?

We tested collaborative filtering vs. popularity baseline:

Method	**Precision@10**
Pure collaborative filtering	2.16%
Improved collaborative (filtered)	2.45%
Popularity baseline	7.56%
Hybrid (50/50)	8.83%

Lesson: Pure collaborative filtering performs worse than just recommending popular books!
Why? 95.75% sparsity - too much missing data
Solution: Hybrid approach works best (8.83% vs 7.56%)

The Hybrid Method: Best of Both Worlds

The Solution: 50/50 Hybrid Approach

Combine two recommendation strategies:

$$\text{Final Score} = 0.5 \times \text{Popularity Score} + 0.5 \times \text{Collaborative Score}$$

- **Popularity:** Recommend books many users liked (safe baseline)
- **Collaborative:** Recommend based on user's learned preferences (personalization)

Result

8.83% precision - outperforms both individual methods! (17% better than popularity alone)

Data Filtering & Scalability

Data Filtering

Started with 1,000 Hardcover users and 45,203 books. Filtered to:

- **Users with $\geq$ 20 ratings:** 246 users
- **Books with $\geq$ 5 users:** 2,547 books
- **Total possible:** $246 \times 2,547 = 626,562$ ratings
- **Actual interactions:** 26,598 (only 4.25%)

Computational Performance

Training time and scalability (300 iterations, 20 features):

- **Current dataset (246 users):** 8.1 seconds
- **Projected 10,000 users:** 32.7 seconds
- **Projected 100,000 users:** 5.2 minutes
- **Time per user:** 33ms (scales linearly)

Implicit Feedback: Weighting Different Book Statuses

Not All Interactions Are Equal

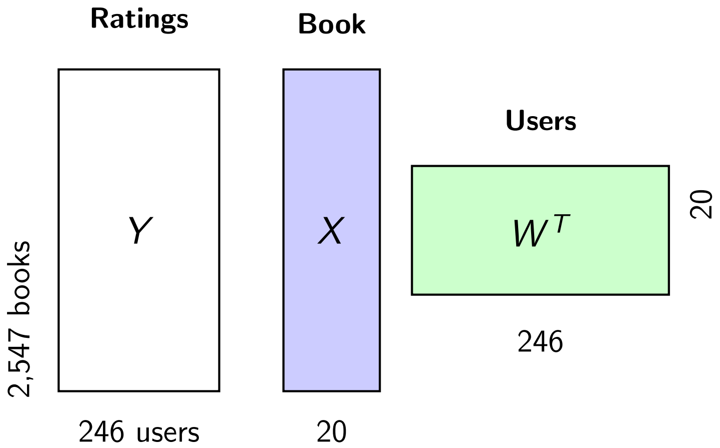
- **Read with rating $\geq$ 3:** 1.0 (strong positive - definitely liked)
- **Currently reading:** 0.7 (moderate positive - probably enjoying)
- **Want to read:** 0.3 (weak positive - interested)
- **Read with rating $<$ 3 / DNF:** 0.0 (negative - disliked)

What About Unrated/Unread Books?

- **During training:** Both unrated and unread books are masked out
- **After training:** Model predicts values for all unrated/unread books
- **Recommendations:** Books with highest predicted values

Impact: Added 11,424 extra signals (75% more data), improving precision from 2.45% to 5.31%

Matrix Factorization: Learning User Preferences



Each user gets a 20-dimensional feature vector describing their reading preferences

Clustering: Finding Your Reading Tribe

Goal: Group users with similar reading preferences

How It Works

1. Start with each user's 20-dimensional feature vector
2. **Normalize** the vectors (make them unit length)
3. Use K-means clustering to group similar users
4. Tested different cluster counts, found K=13 is optimal

Finding Friends within Your Cluster

- Calculate **dot product** (cosine similarity) with each cluster member
- Higher similarity = better friend match
- Show shared books as conversation starters

Result: 13 distinct reading groups