Presto!

A high-performance SQL-based recommendation system

Rolf Hendriks, December 4 2024

Overview

Presto! Features at a Glance

- Performant book/music recommendation system based on user ratings
- Traverses millions of items in milliseconds
- Find books, albums, artists, or authors
- Emulates ‘users who liked this product also liked’ feature
- Uses **cosine similarity** search to make item-based recommendations

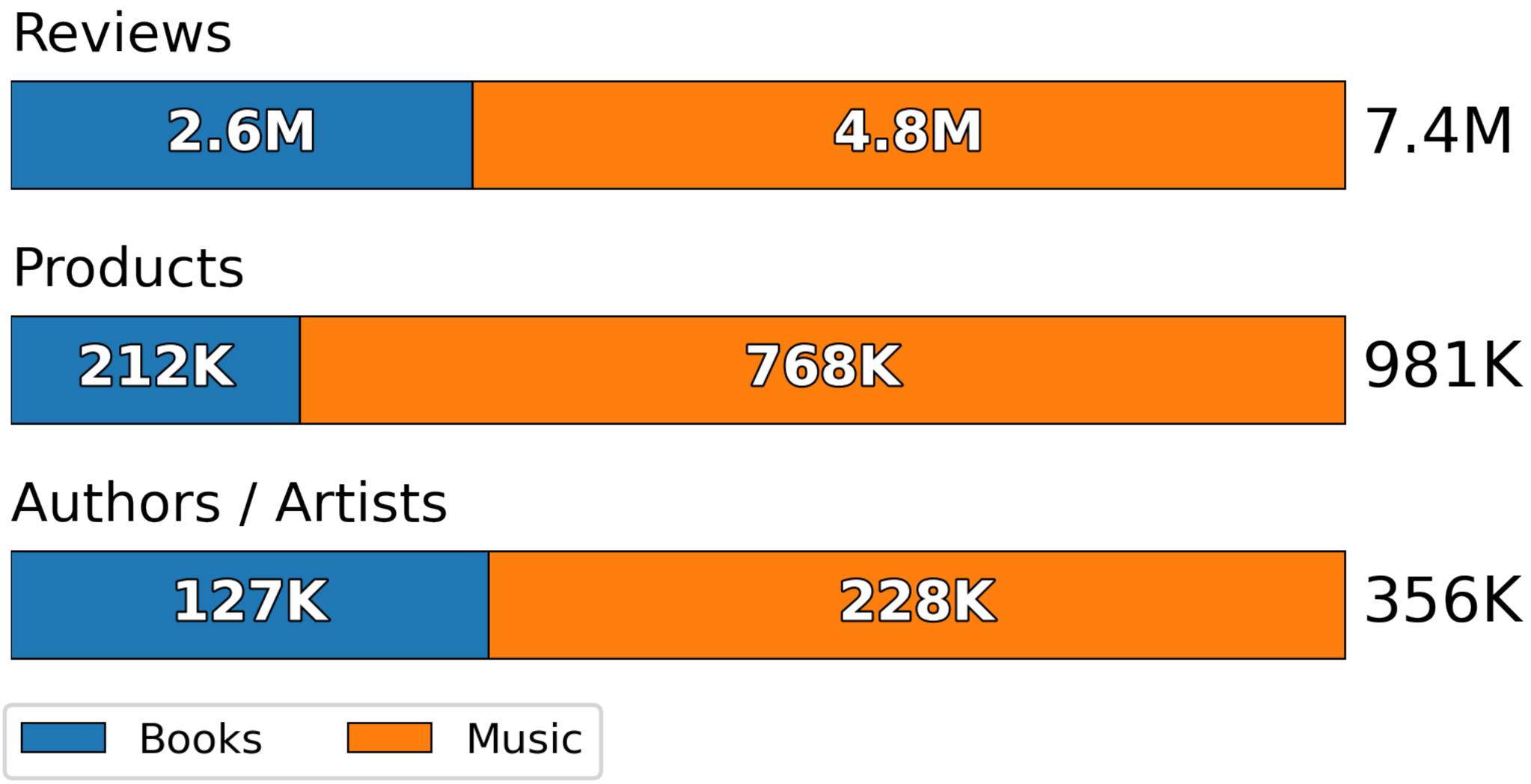
Demo!

Let’s see Presto in action…

01 The Data

7.4 million user reviews for 1 million books and albums

Presto! Product Data



McAuley Lab Dataset

https://amazon-reviews-2023.github.io/

- **Over 500 million** Amazon reviews from 34 categories (!)
- 5 million music reviews of 228K artists
- Requires extensive treatment, contains misclassified products
- Results:
    - 30% of eccentric niche artists had 10 or more reviews
    - 10% coverage of worldwide total artists

MusicBrainz Database

https://musicbrainz.org/doc/Development/JSON_Data_Dumps

- The wikipedia of music data, with seemingly **comprehensive** coverage of artists, albums, genres, songs
  - Over 2.2 million artists
- Useful for removing misclassified data from McAuley sets
- 100% success rate on stress testing against rare artists (!)
- Widely used by commercial products (Spotify, Amazon, etc)
- Organizationally related to the Internet Archives / Way Back Machine

Book Review Dataset

https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews

- 3M user reviews originating from GoodReads
- Over 200K books
- Simpler to process than McAuley dataset
- **Has review downvote metadata**

SQL Database Backend

From minutes to milliseconds thanks to SQL and indices

- Handles millions of records within milliseconds (!)
- Baseline: loading data without SQL takes minutes and can crash a machine
- SQL speed: 0.01 seconds + hundreds of thousands of records per second
  - Tens of seconds for full scans of millions of records
  - **Covering indices** enable millisecond-level performance for user-facing results

02 Search

From query to product(s)

Search Approach - v 1.0

Create separate search-facing vs user-facing texts for searchable fields

- Title
- Search Title
- Creator (artist or author)
- Search Creator

Cases Handled

- **Diacritic insensitivity** (critical in every language except English)
  - E.g: ‘Céline Dion’ matches ‘Celine Dion’
- ‘The’ insensitivity
  - E.g: ‘The Offspring’, ‘Hunger Games’, etc
- Convert ‘Lastname, Firstname’ to ‘Firstname Lastname’
  - E.g: ‘George Orwell’ matches ‘Orwell, George’
- Limited: **disambiguation** for multiple editions of the same book/album

Cases Not Handled

- Efficient **substring search** (!)
  - Database indices handle matches and prefixes, but not containment
  - Possible solutions: reverse search, sqlite FTS (full text search), tags
- Synonymous names / aliases
  - E.g: ‘Jackson Five’ vs ‘Jackson 5’
  - Recommender will pick up aliases very well though

03 Ratings and Reviews

The backbone of Presto! recommendations

Review Quality control

How and why Presto considers review metadata

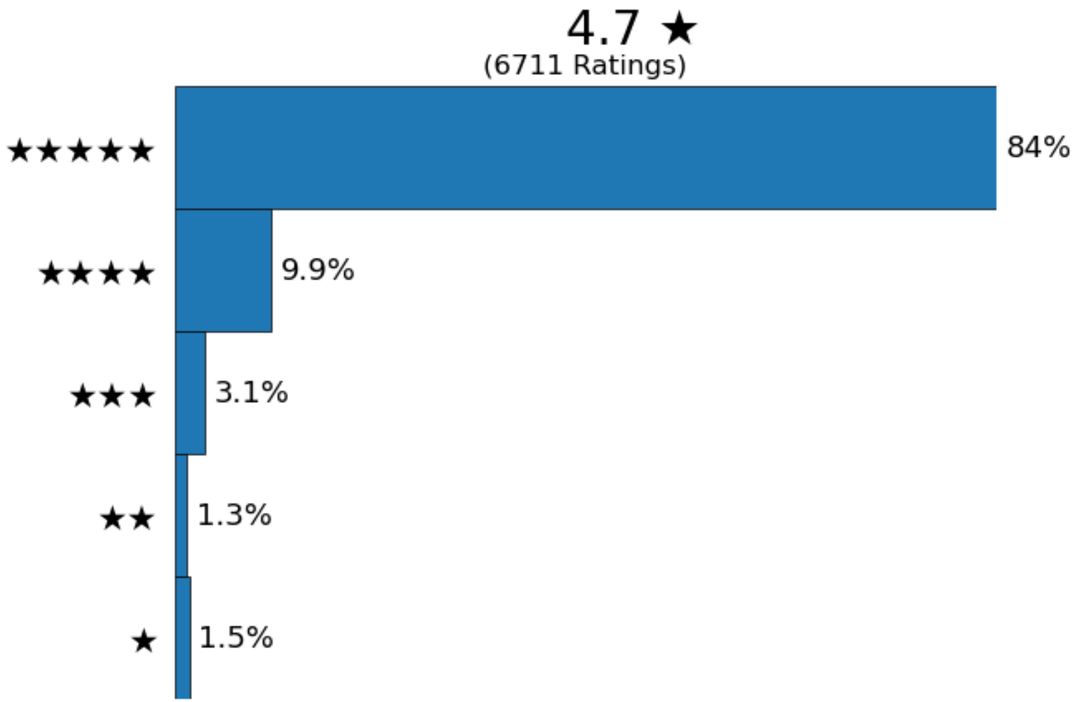
Filtering low-quality reviews helps:

- Improve recommendation quality
    - Low-quality data produces low-quality output (even for high-quality algorithms)
- Highlight high-quality products
- Provide recommendations quickly

Review Quality Example

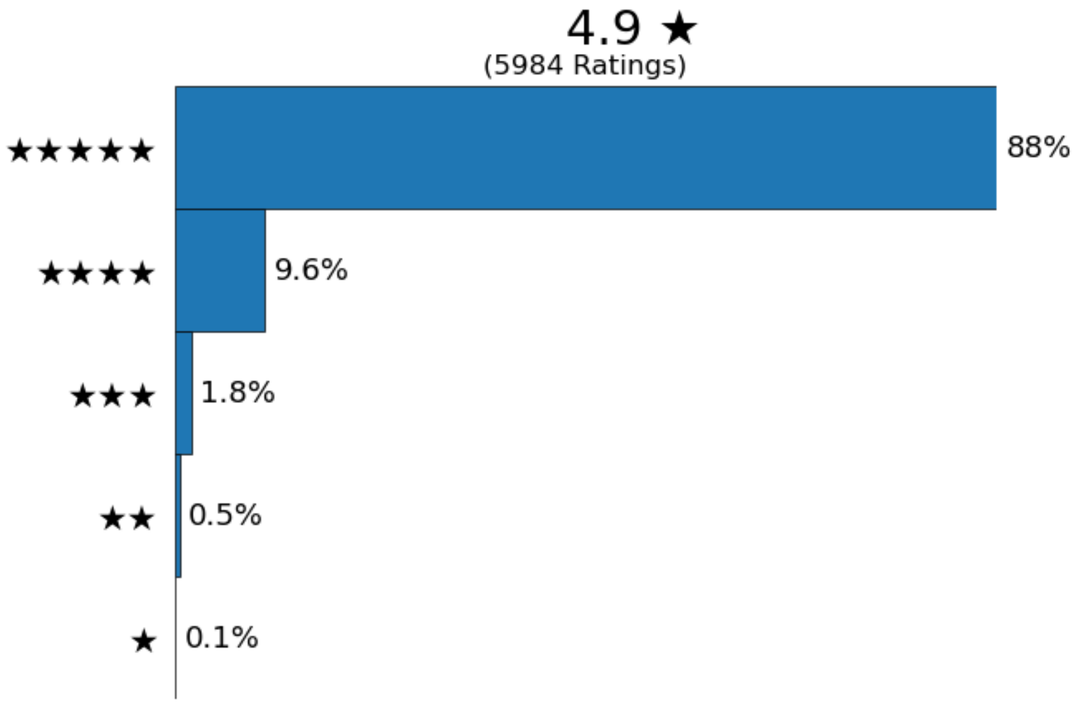
Harry Potter Reviews

1 star ratings decrease 15x when adding quality control:



Raw Ratings

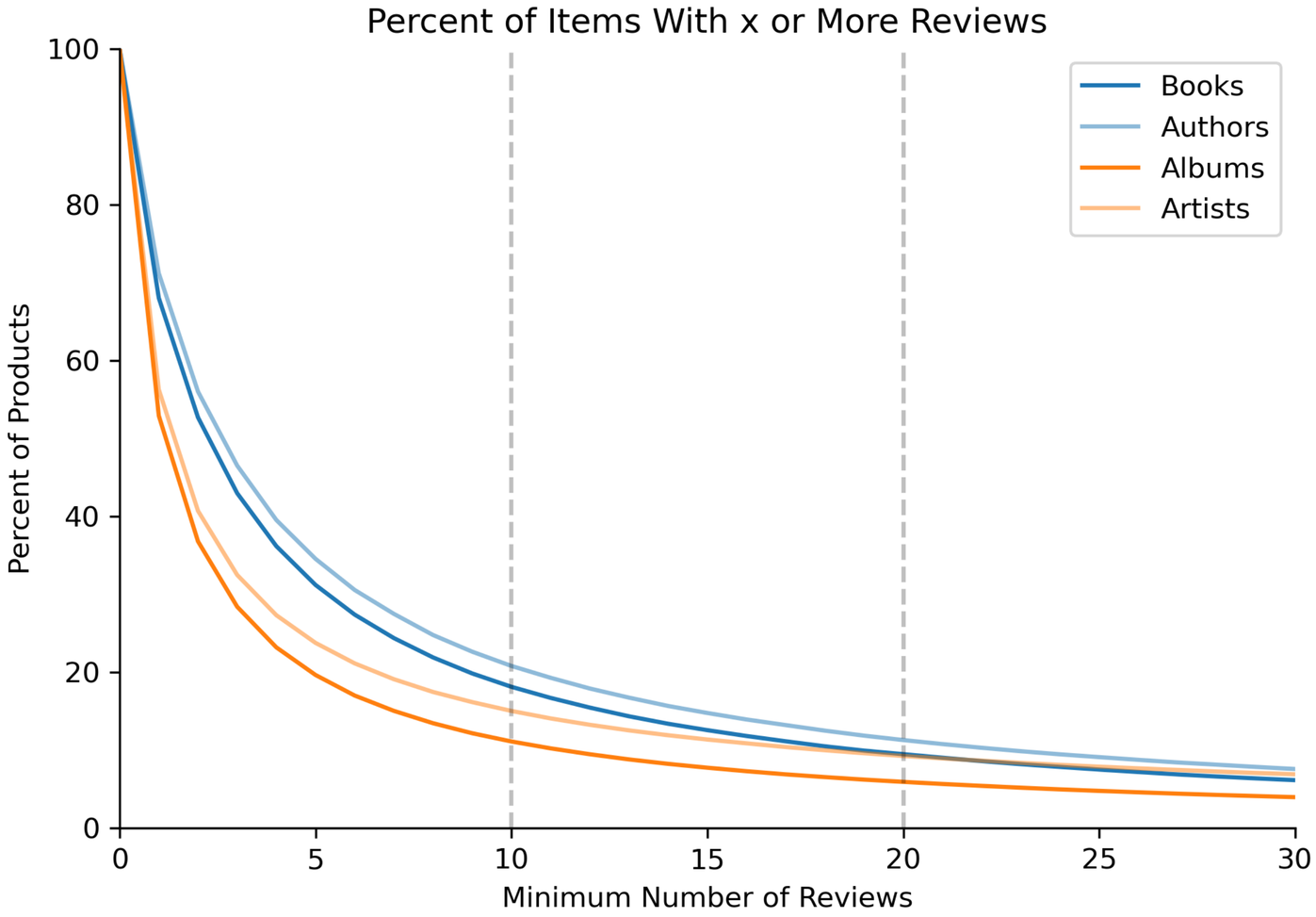
Vs

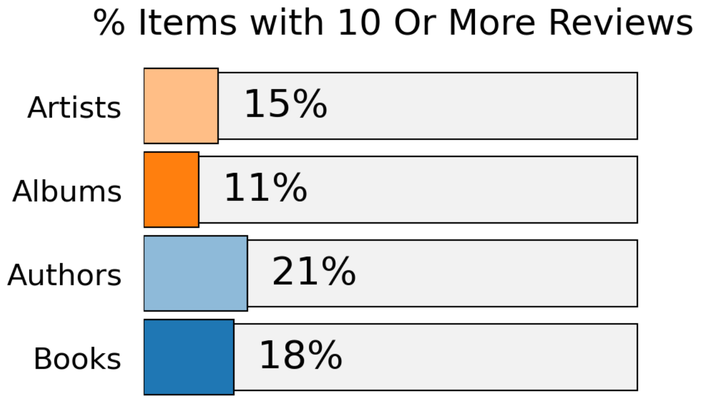


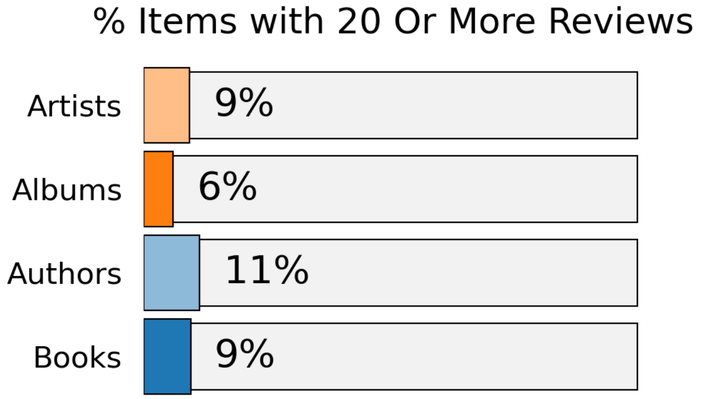
Helpful Ratings

Review Quantity control

Most products are ‘review starved’

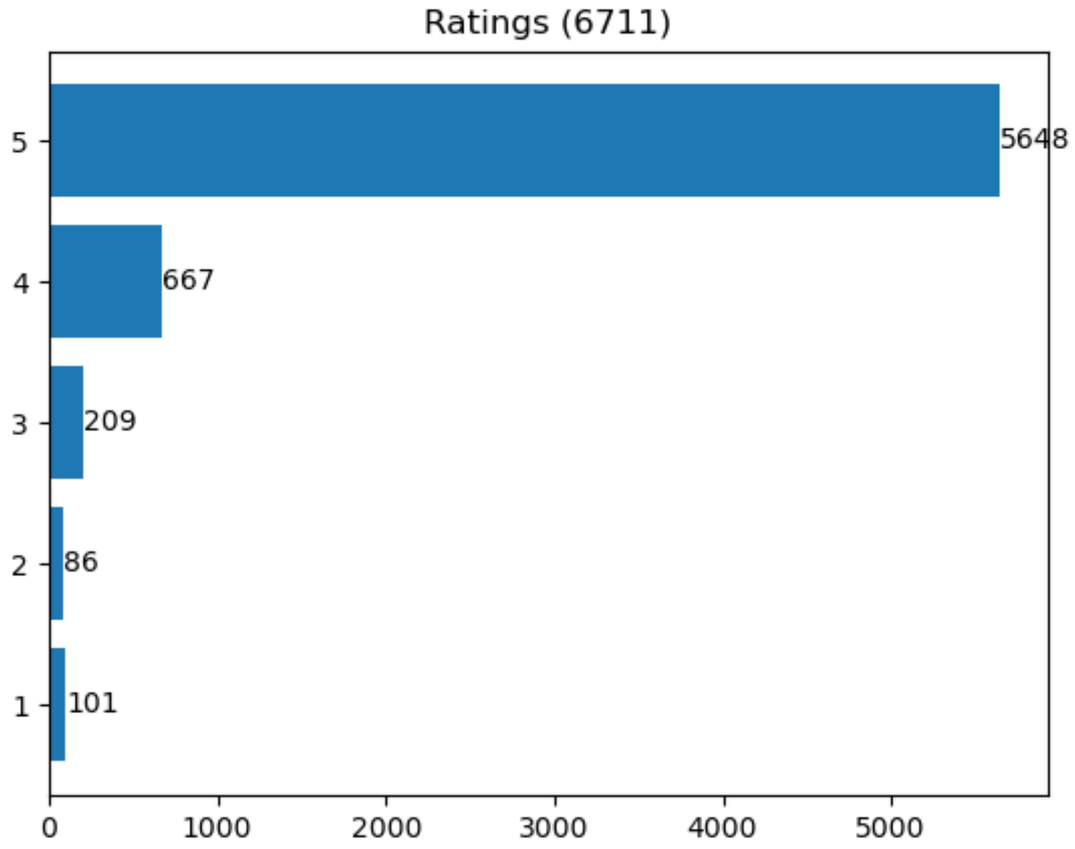






Review Visual Quality Control

This is what happens when we show ratings using matplotlib defaults:



04 The Presto! Process

How Presto! turns ratings into recommendations

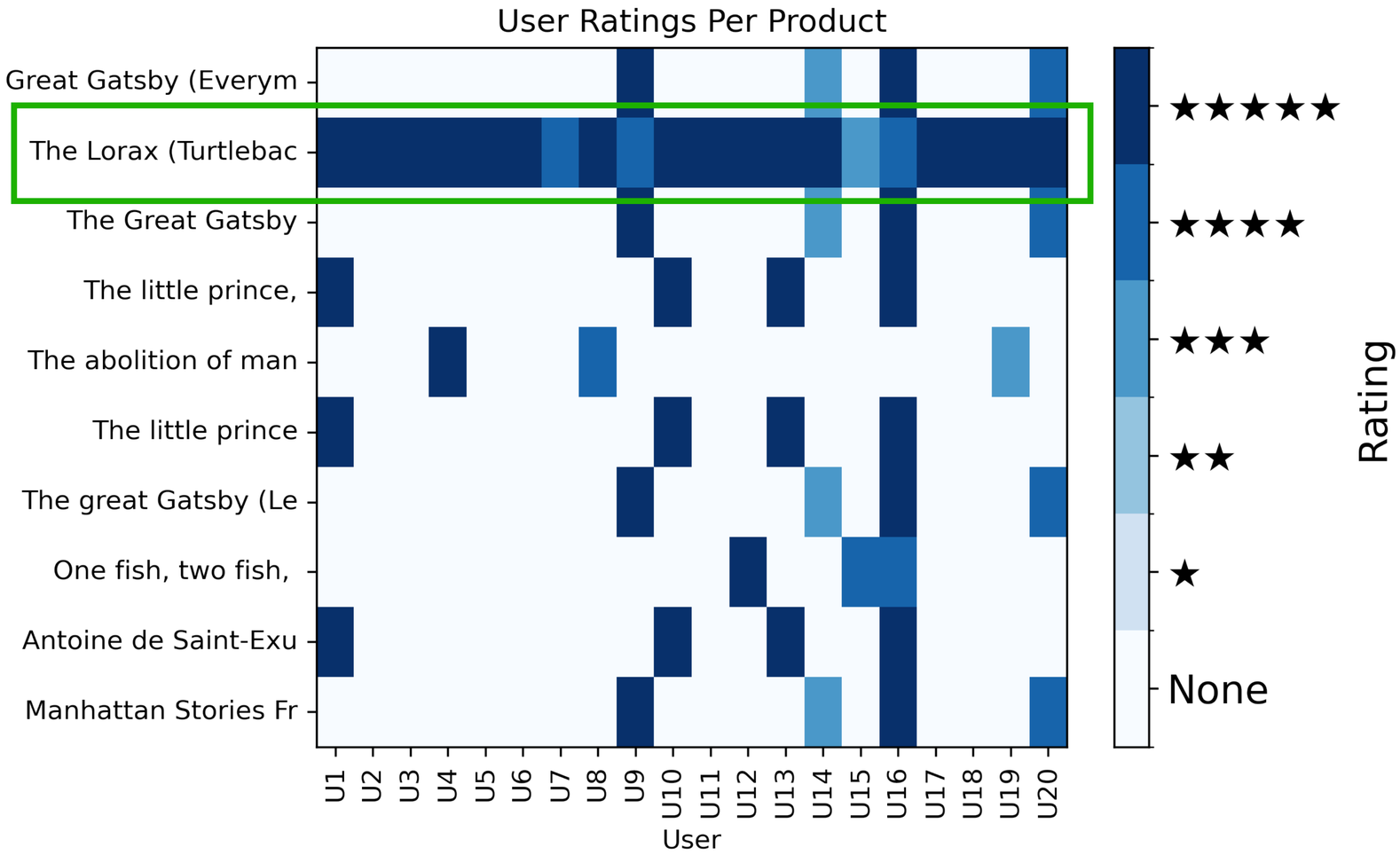
Step 1: Find Related Reviews

How Presto! turns reviews into recommendations

- Filter low-quality reviews (using upvotes / downvotes)
- Find all related products and users
  - Products are ‘related’ if they have any reviewers in common
- Results may be large and sparse
  - Harry Potter: 21K related books (of 212K)

Step 2: Create Rating Table

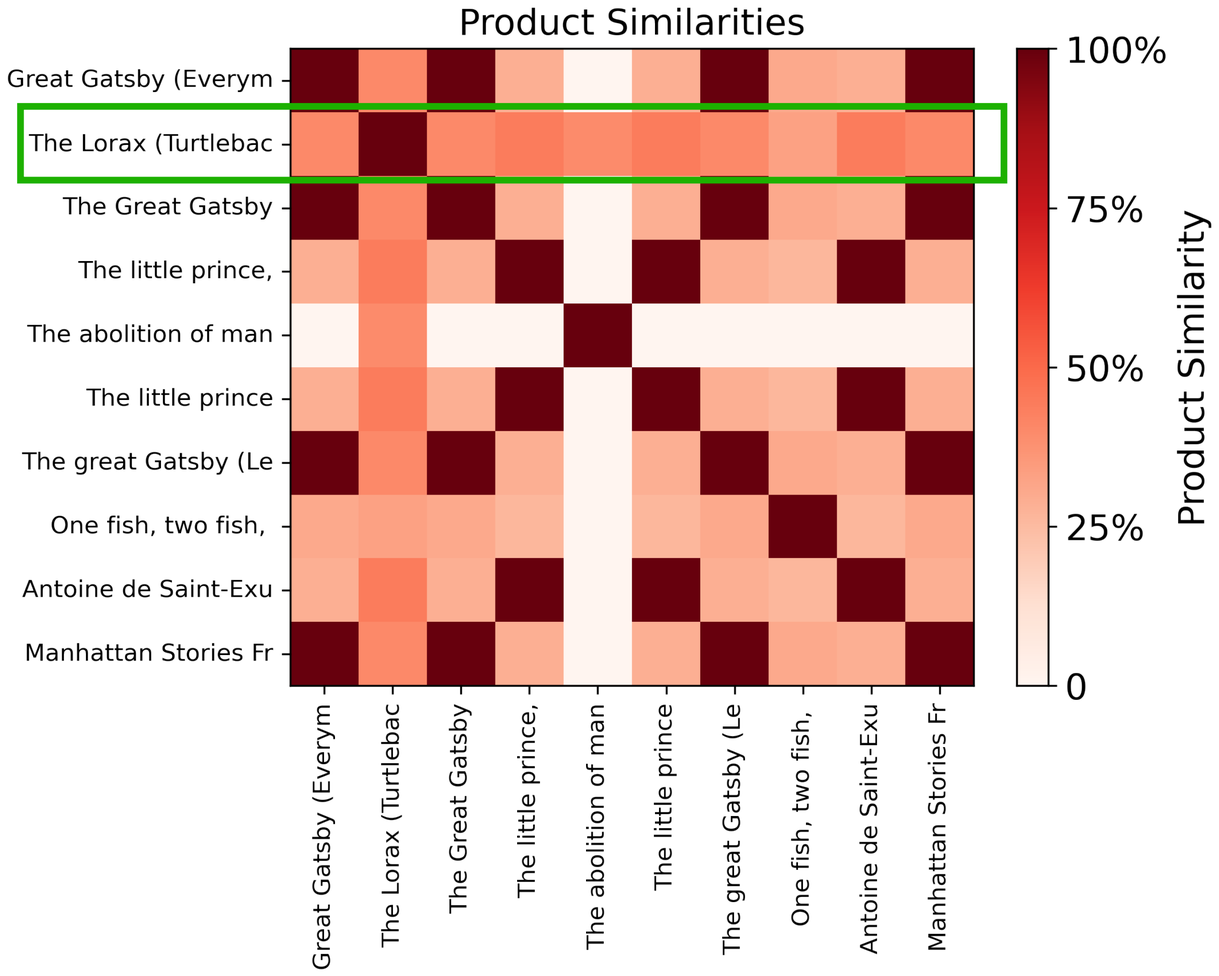
How Presto! turns reviews into recommendations



- Filter low-quality reviews
- Filter reviewers by review count
- Filter products by review count
- Create rating table

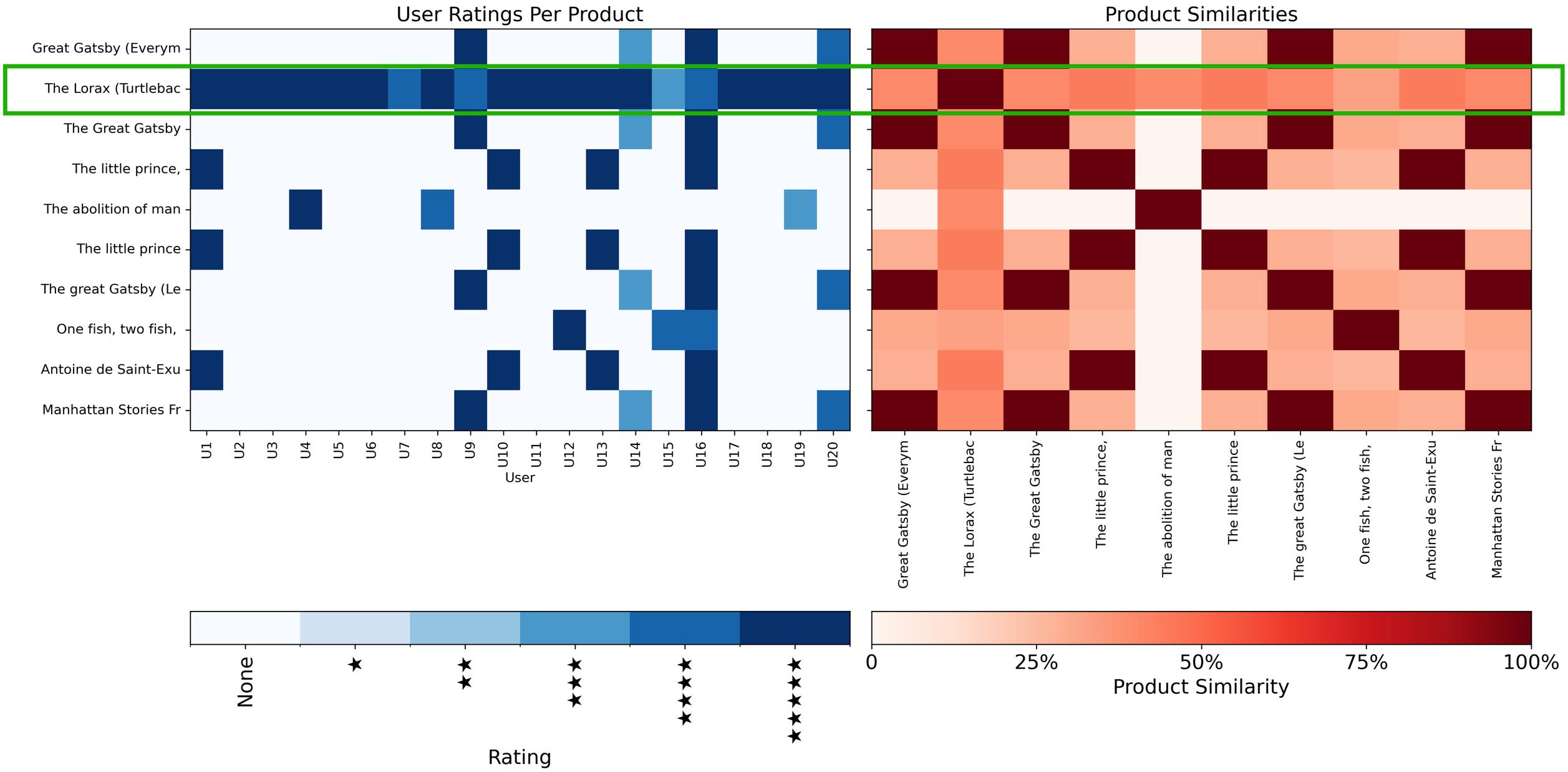
Step 3: Find Similar Rows

How Presto! turns reviews into recommendations



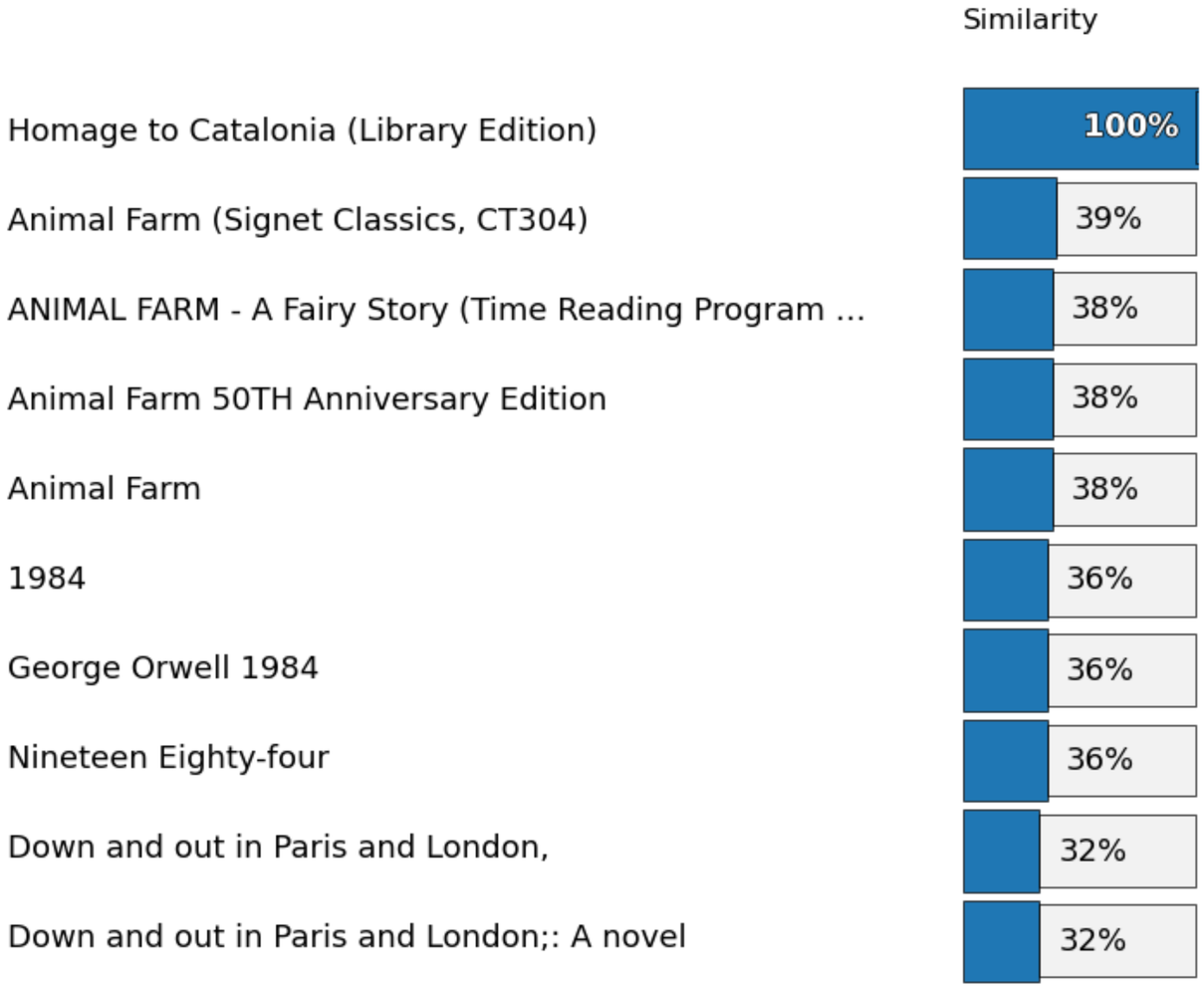
Each square maps the similarity of two rows in the rating table

How Product Recommendations Work



Results

People who liked Homage to Catalonia also liked:



05 Conclusions

Take-home lessons for recommender systems

1. Quantify Your Data Needs

A competitive recommender needs *very large* volumes of data. To gauge actual vs desired volumes of data:

- Monitor number of books / authors / etc with enough reviews to make recommendations
- Acquire a comprehensive list of all possible books / movies / etc (e.g. MusicBrainz database) to monitor universal coverage
- Monitor estimates for the percentage of user needs covered by products with enough user reviews
    - E.g maintaining records on terms entered that had no results or too few reviews for recommendations

2. Monitor Data Quality

Garbage in, garbage out

Invest in metadata and algorithms to remove low-quality data from your models

E.g. user upvotes / downvotes

E.g. pre-filtering candidate products and users to consider for recommendations

3. Invest in Disambiguation

Real-world product databases are saturated with duplicate editions of the same product. Invest in solutions that prevent recommending 10 variations of the same one or two products.

4. Recommenders are Magic

Presto! only receives:

- Ratings
- Product IDs
- User IDs

And yet, it picks up on bands with shared:

artists, genres, band members, musical styles, lyrical themes, historical influence, etc

06 The End?

‘A data science project is never done
- it only reaches its deadline’

Future Possibilities

- **Tagging System**
  - Each product can have any number of tags with arbitrary type and value
  - Solve multi-genre/author products, search, NLP, movies by actor, etc
- **Web app**
- **Podcasts, Movies**
- Better **duplicate removal**