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
  - Emulates ‘users who liked this product also liked’ feature
  - Uses **cosine similarity** search to find related products
  - Produces instant results from millions of reviews
- Based on 8 million reviews of 1 million books and albums
- Search for books, albums, artists, or author

Demo!

Let’s see Presto in action…

01 The Data

8 million user reviews for 1 million books and albums

McAuley Lab Dataset

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

- **Over 500 million** Amazon reviews from 34 categories (!)
  - 5 million music reviews from 2 datasets (Digital Music, Audio CDs)
  - 228K artists
    - 30% of eccentric niche artists had 10 or more reviews
    - 10% coverage of worldwide total
- Requires extensive treatment, contains many misclassified products
- No genre data

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
- Widely used by commercial products (Spotify, Amazon, etc)
- 100% success rate on stress testing against rare artists (!)
- Organizationally related to the Internet Archives / Way Back Machine

Lesson Learned + Pivot

A commercial-grade music recommender for niche artists requires more data than is publicly available

Roughly **tens of millions** of reviews comprising dozens of gigabytes

Time to pivot by focusing on data instead of music recommendations:

- Create a general-purpose product and review database
- Validate general-purpose data by including books and music
- Switch primary focus to books

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

Future Possibilities

- **Tagging System**
  - Each product can have any number of tags with arbitrary type and value
  - Solves multi-value searches (e.g. books with multiple authors)
  - Solves category-specific attributes (e.g movies by actor)
  - Could be the foundation for search 2.0 and natural language processing
- **Podcasts**
- **Movies**

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** (important everywhere except USA)
  - 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 recommendations

Review Quality control

How and why Presto considers review metadata

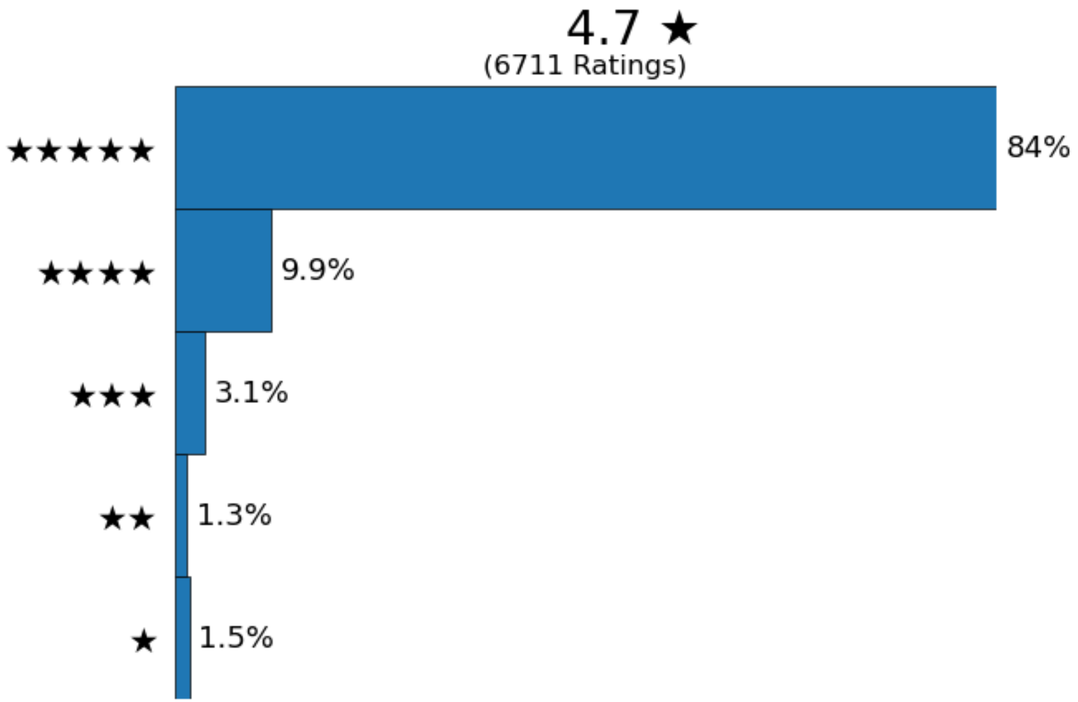
Filtering low-quality reviews helps:

- 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

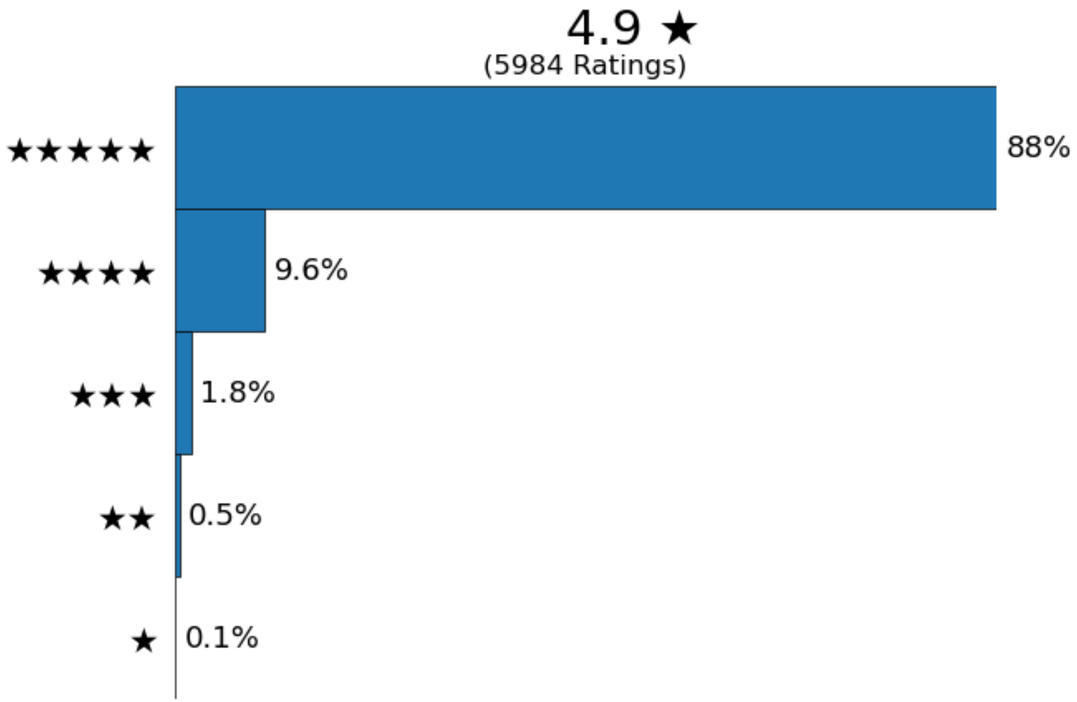
J. R. R. Tolkien's The Hobbit

1 star ratings decrease 15x when adding quality control:



Raw Ratings

Vs



Helpful Ratings

Review Quantity control

About 10% of products have enough data for recommendations.

04 Recommenders

Using user ratings to find related products

The Presto! Process

How Presto! turns reviews into recommendations

- Filter low-quality reviews (using upvotes / downvotes)
  - Pad each review’s upvotes/downvotes for optimal results
- Find all related products and users
  - Products are ‘related’ if they have reviewers in common
- Filter low-quality products and users
  - Based mainly on number of common reviews in this case
- Find products with user review patterns similar to the starting product

Presto!’s Emergent Behavior

Even though Presto! Only considers reviews, it finds works with similar:

- Author or artist (first and foremost)
- Aliases
  - e.g Tubeway Army = Gary Numan
- Level of popularity vs obscurity
- Genre / Sound
- Time

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 editions of the same one or two mostly identical products.

4. Recommenders are Magic

A recommendation system 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 Thanks

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