Tastecliq

Problem: User taste preferences are scattered on the internet

- movies on netflix.com
- music on rhapsody.com
- books on goodreads.com ...

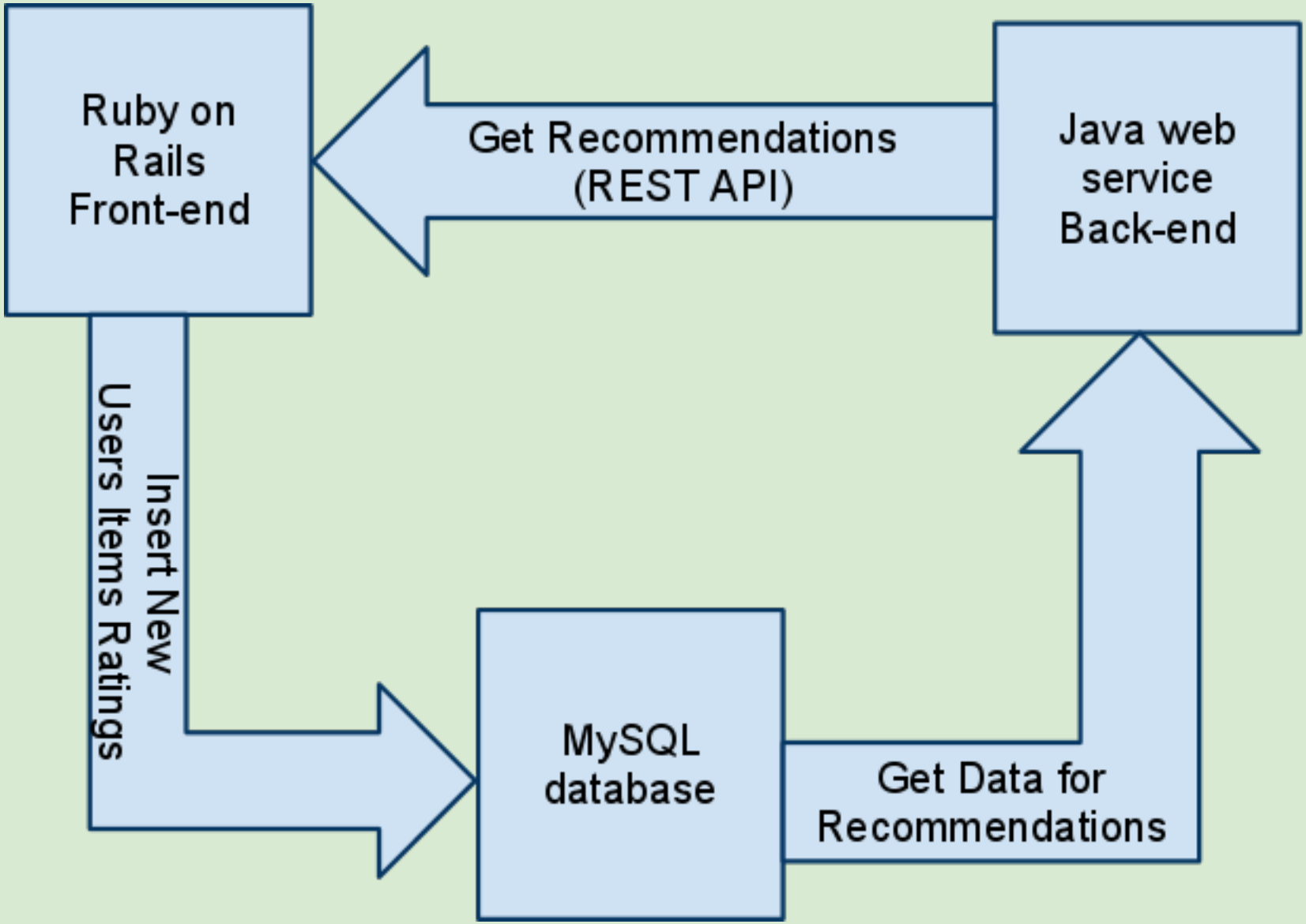
Solution: A one-stop-shop for organizing your personal libraries for different media types

Benefits:

1. People can track all taste preferences in one website

2. Website can make better recommendations since it knows more about users

Demo


Ruby on Rails Front-end
Get Recommendations (REST API)
Java web service Back-end
Insert New Users Items Ratings
MySQL database
Get Data for Recommendations

Recommendation Algorithm Choices

SlopeOne for general user recommendations

- Example:
  - If Sam gave the Beatles 5/5 and Aerosmith 4/5
  - Soheil gives the Beatles a 5/5
  - What do we think Soheil would give Aerosmith?
  - SlopeOne answers this!
- Our algorithm considers all user ratings on all items to predict the highest rated items for a single user

Recommendation Algorithm Choices (cont.)

Pearson Correlation Coefficent for similar items

- What if we just wanted items similar to another item?
  - Use an item based recommender!
- Pearson coefficent to find a good matching of items.

- Point of an item based recommender: Items are static in their similarities(i.e. they don't change much) so we can precompute values.

Other Algorithms Considered

Tanimoto Coefficient For Item Similarity:

- Computes a simple yes or no (did they buy it or not?) item similarity coefficent.
- Data we had was based on ratings, so we could not use.

Dynamic Item Similarity Computations:

- Keeps things up-to-date, very accurate.
- Misses the point of an item based recommender. Not static and very slow for computations.

So you want to make a Recommender?

- **What are you trying to do?**
  - More profit by suggesting relevant items to users.
  - Expose more *relevant* site content to users.

Recommendations do the searching for the user. Recommenders aren't limited to weird algorithms. Top 100 lists are also recommendations.

- **What information do you have available?**
  - Ratings, Page/Item View logs, buying history, static data you bought, item metadata (song characteristics, movie genres).

What if you don't have any good data available?

Case Study
YouTube's Related Videos

- **Goal:** Give users related videos (or next video in series)
- **With What Data?:** Collective users' viewing logs, text relevance (similar clip titles), user ratings.
- **Other Ideas:** Adjust recommendations based on what people *actually* click. More clicks => move up in ratings, less clicks => move down.

Testing and Adjustment

1. Implement a recommender that "should" work.
2. Does it meet our requirements? (Probably not.) Why not?
3. Make adjustments.
    - Recs too varied?
        - *Restrict by genre, or improve data quality.*
    - Recs too similar to original item?
        - *Change algorithm/feature. Hybridize your recommender.*
    - Recs not actually relevant?
        - *Change feature, or learn from failures based on user click behavior.*
4. Return to step 2 until you run out of time and money.

Things Learned:

- Open source packages are messy sometimes.
  - Ex. Broken build of Taste Mahout package.
- There is no "perfect" recommendation algorithm.
  - Different ones have different uses. It all depends on what your goal is.
- String matching is a problem. Can be a problem for recommender, but beyond the scope of our project.
  - Simple example "The Matrix" vs. "Matrix, The". Can be much more complicated.
- Ruby on Rails is awesome! Fast development and generally easy to learn.