Steam Video Game Recommendation System

Joshua Ogden-Davis
April 2024

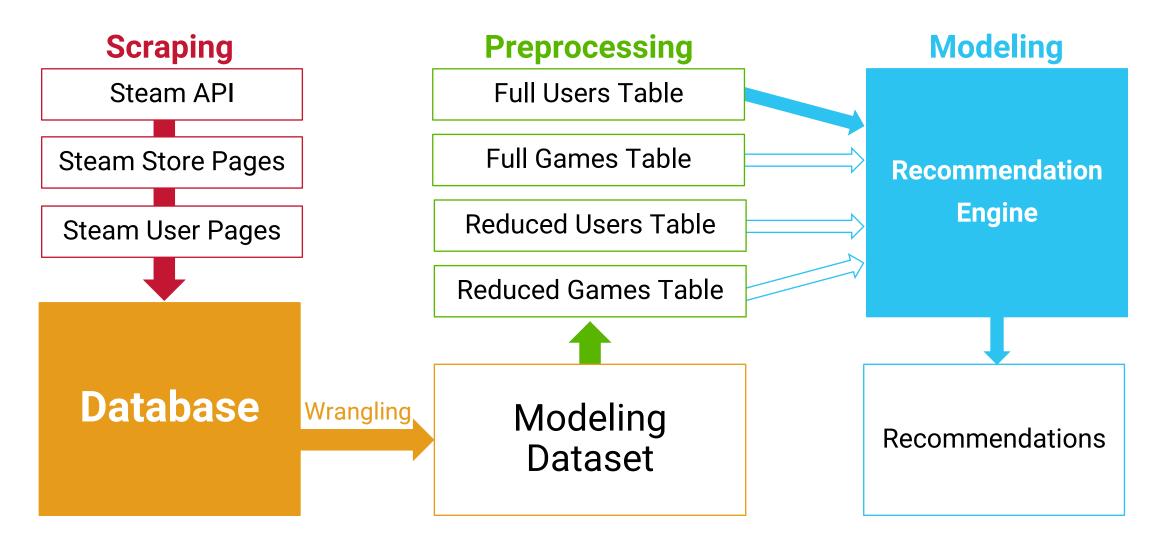
Purpose

- 1. Generate an extensive dataset from publicly-available data
- 2. Develop a game recommendation system for existing users

Two Main Sub-Projects

Web Scraper Create user profiles Create game profiles Hybrid Filtering (custom)

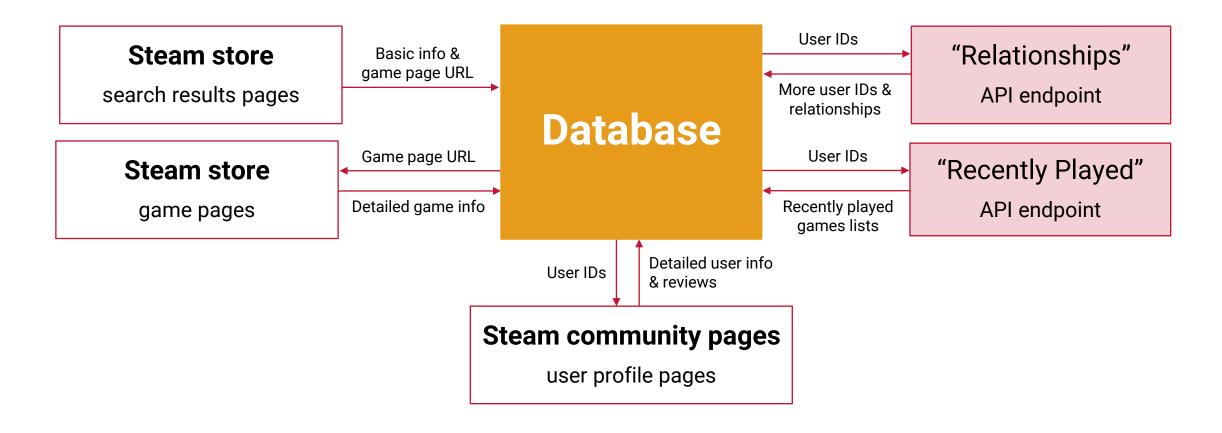
Overall Project Flow



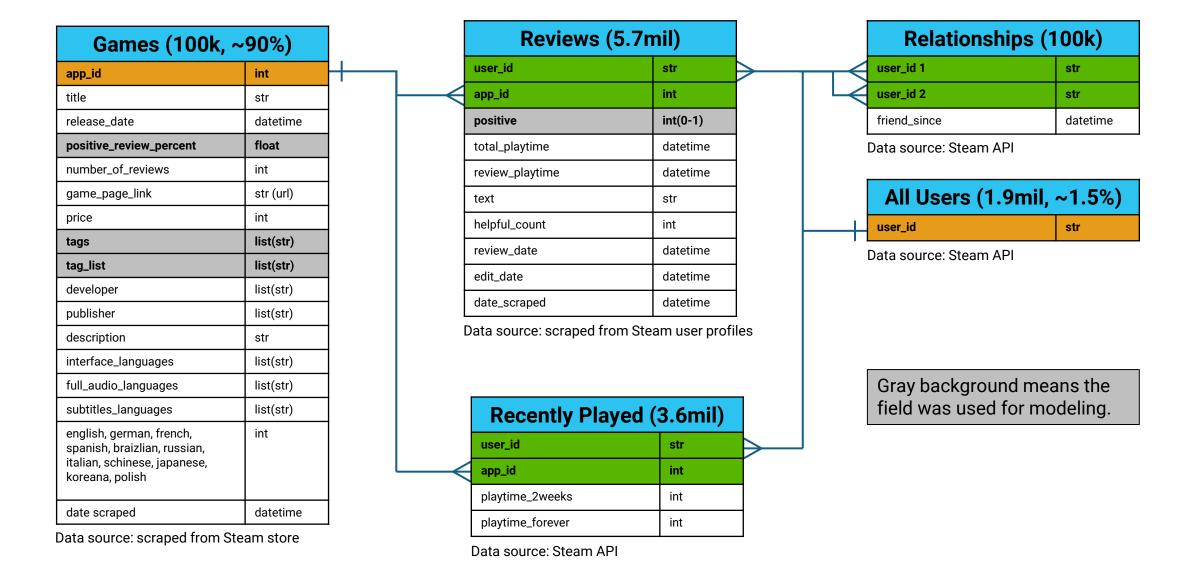
Scraping Flow & Sources

Over 10 million API & URL requests

Libraries: Requests + BeautifulSoup



Database Schema



Preprocessed Games Tables

Condensed Sparse Row Format

	Adventure	Indie	Fem. Protagonist	(446)
game index 1	0	1	0	•••
game index 2	0.8	0	1	•••
game index 3	0	0	0	•••
(100k)	•••	•••		

- "Reduced" games table omits games with <5 tags (arbitrary).
- Because CSR has no separate index, we must maintain:
 - Bidict of full table index <-> app_id
 - Bidict of full table index <-> reduced table index
 - (Steam already indexes tags starting at 1)

Preprocessed Users Tables

Condensed Sparse Row Format

	game 1 col index	game 2 col index	game 3 col index	(100k)
user index 1	1	0	0.2	
user index 2	0.2	0.2	0	
user index 3	0	-1	0.2	
(1.9mil)	•••			

- "Reduced" users table omits users with <10 games (arbitrary).
- Values indicate levels of preference (does not stack):
 - Recently played: 0.2
 - Positive review: 1
 - Negative review: -1

Modeling Flow (Simplified)

Input:

Target User Profile

Model

- Collaborative Filtering (User-Based)
- Which users most resemble the target user?
- Which games are those users playing?
- Give those games scores based on how many of the similar users are playing them and how similar those users are.

Content Filtering (Game-Based)

- What are the target user's favorite games like?
- Which other games are most like that?
- Give those games scores based on exactly how similar they are to the user's favorite games.

Final Recommendations

- Weight and combine the collaborative and content filtering scores (with some extra magic).
- Choose the top-scoring games overall.

Output:

List of game recommendations, in order

Modeling Flow (Complexified)

Input:

Full Users Table

Single row

Items in **red** are parameters used in model optimization.

Doubles: A bonus for appearing in both filter results

Popularity: A bonus based on overall positive rating percent

Model

Rank all users by similarity to target....

Take top N similar users

Row values * sim. scores, sum columns

Select top N column sums

Return games + **normd.** scores

Determine N most similar games

Get top N sim. score * pref. score Return games + **normd.** scores

Weight and combine scores
Factor in doubles & popularity
Return top N games + scores

Reduced Users Table

All rows

Reduced Games Table

All rows

Output:

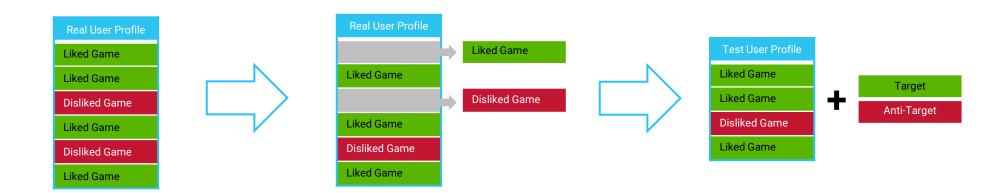
Pandas Series

Keys: app_id

Values: score (ordered)

Testing & Evaluation Process

- Create test users by removing one liked game and one disliked game from random existing user profiles.
- 2. Generate recommendations for test users.
- 3. Compare the recommendations to the removed games.
 - If we recommended the target game, score +1
 - If we recommended the anti-target game, score -1



Results

Best parameters*:

'similar_user_limit': 250

'collab_filter_limit': 103

'content_filter_limit': 32

'double_bonus': 1.79

'popular_bias': 1.84

'ratio': 0.71

'recs': 20

* via BayesianOptimization()

Best model performance:

Test users: 100

Good recs: 30

Bad recs: 17

Total score: 13

Next Steps

- Continuously increase dataset size
- Continuously refresh records
- Improve data quality (especially for developers and publishers)
- Include more features (developer, publisher, date released, etc)
- Control for new vs old users (only have "recently" played game info)
- Develop a higher-resolution evaluation function (utilize ranking, etc)
- Move to the cloud

Thank you!

Joshua Ogden-Davis

May 2024

GitHub: joshtod

Email: ogden.davis@gmail.com