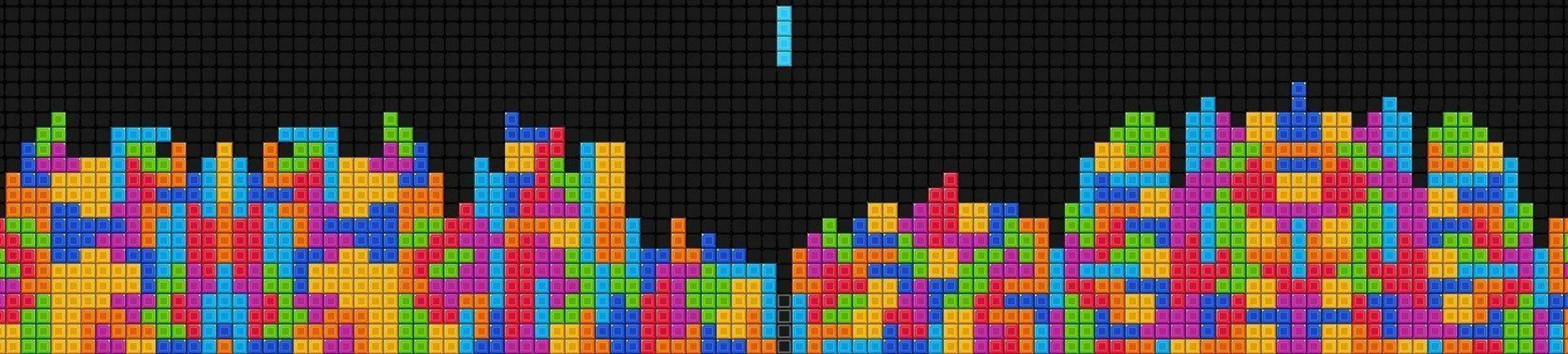
Steam Video Game Recommendation System

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

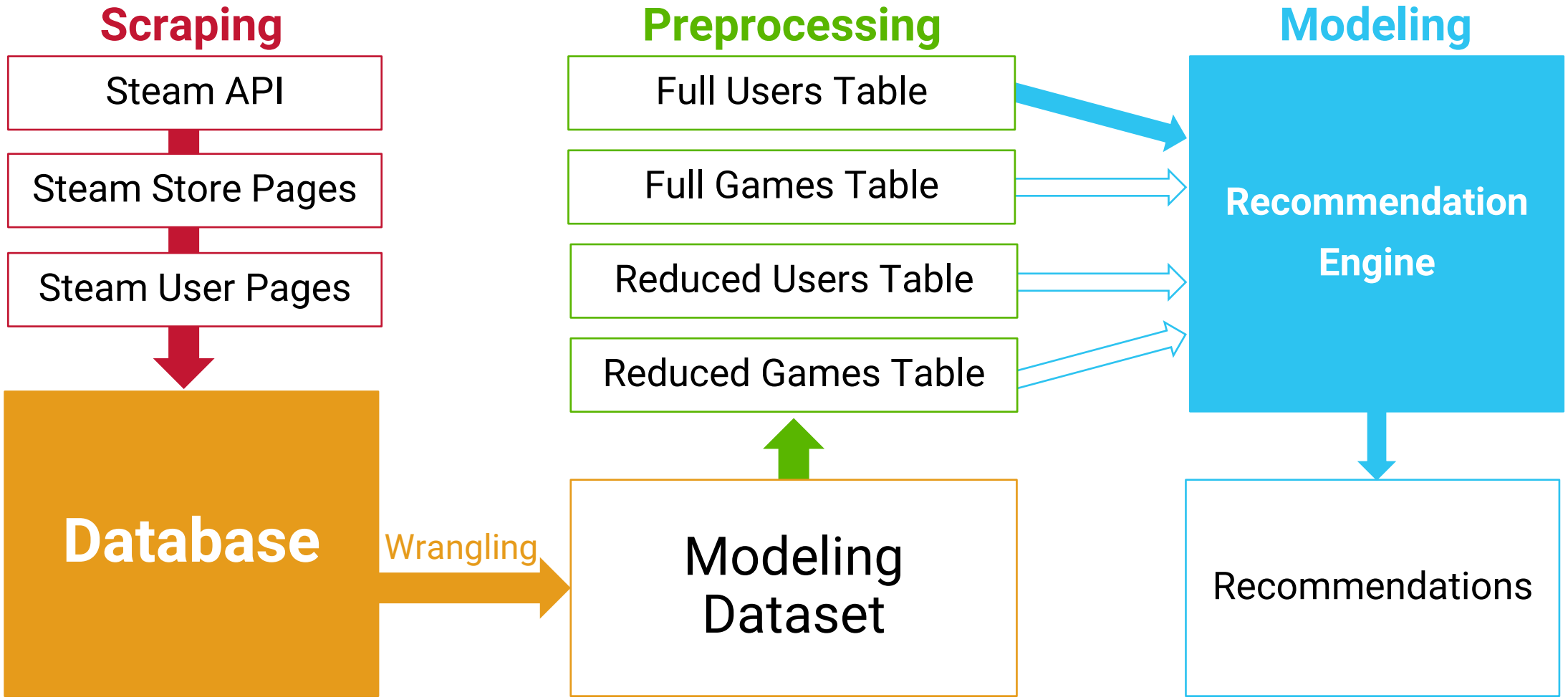
Web Scraper

· Create user profiles
· Create game profiles

Recommendation Engine

· Collaborative + Content
· Hybrid Filtering (custom)

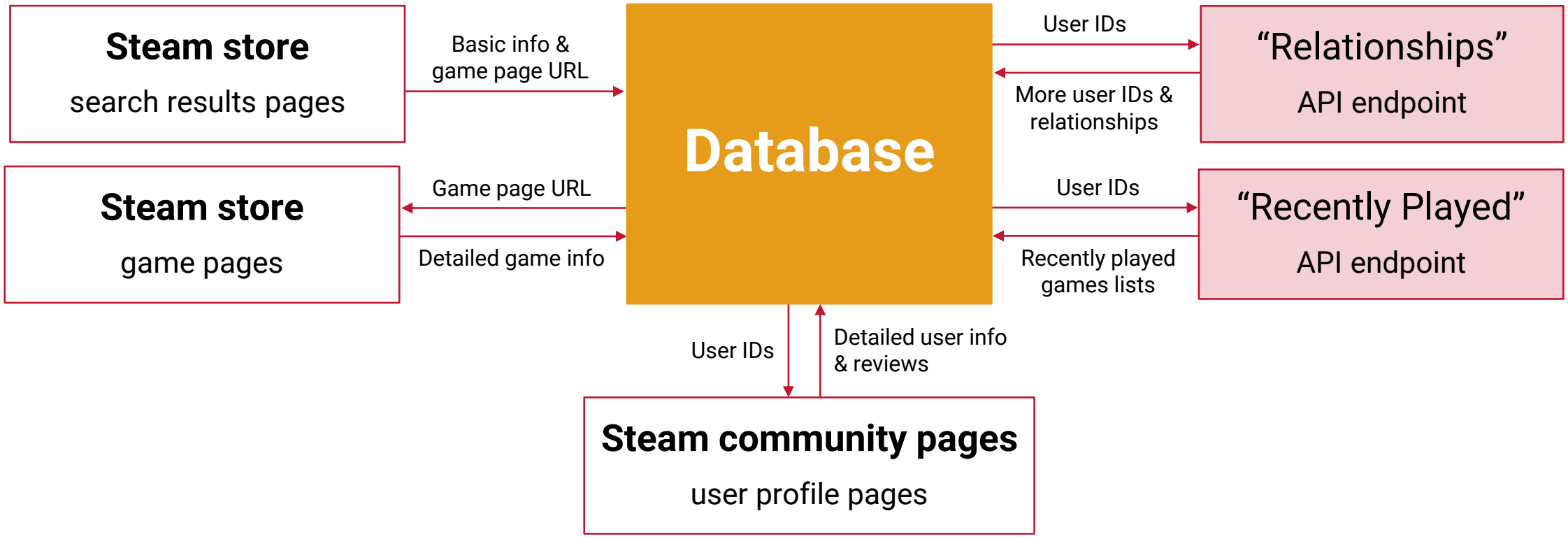
Overall Project Flow



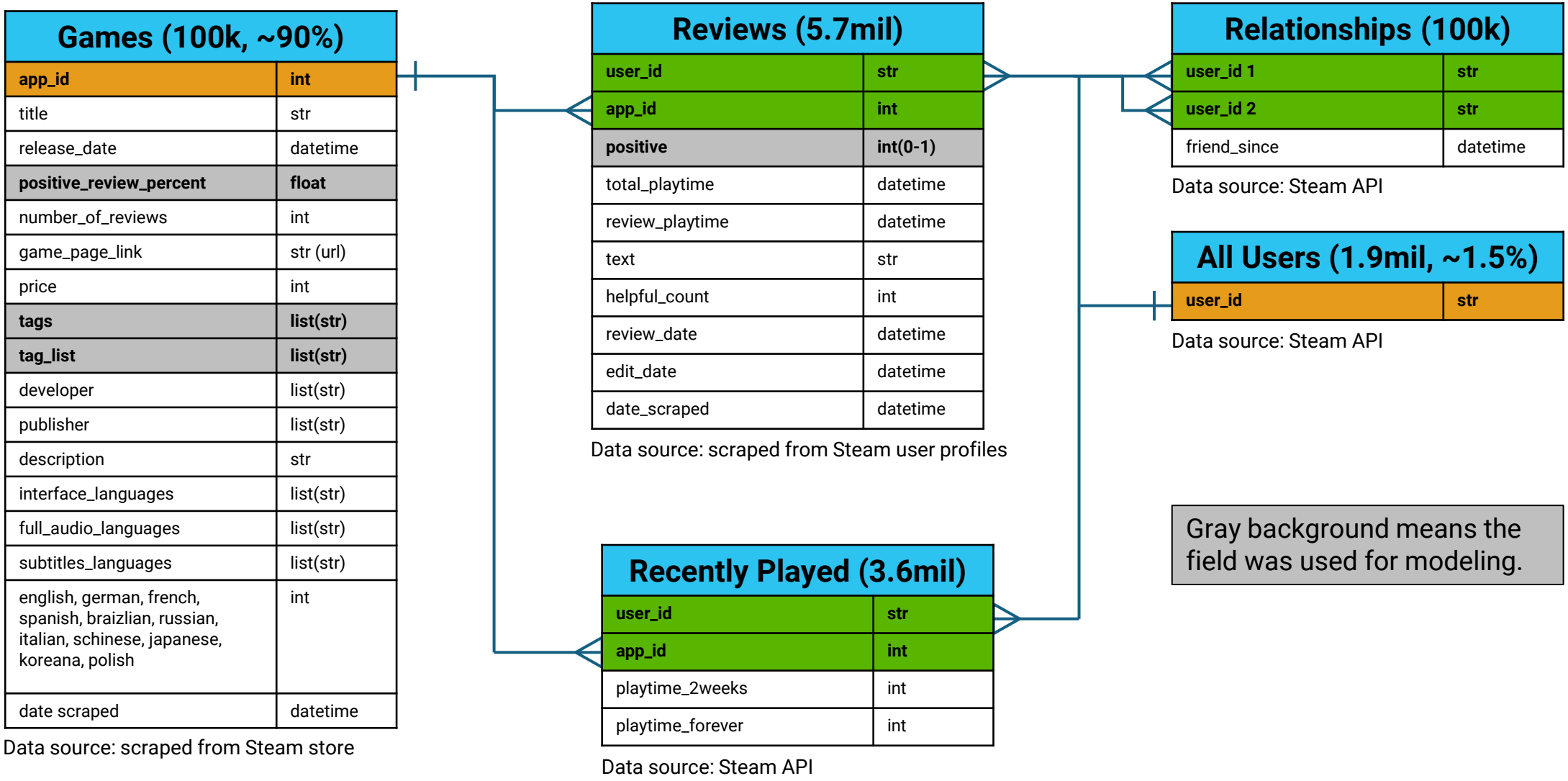
Scraping Flow & Sources

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)

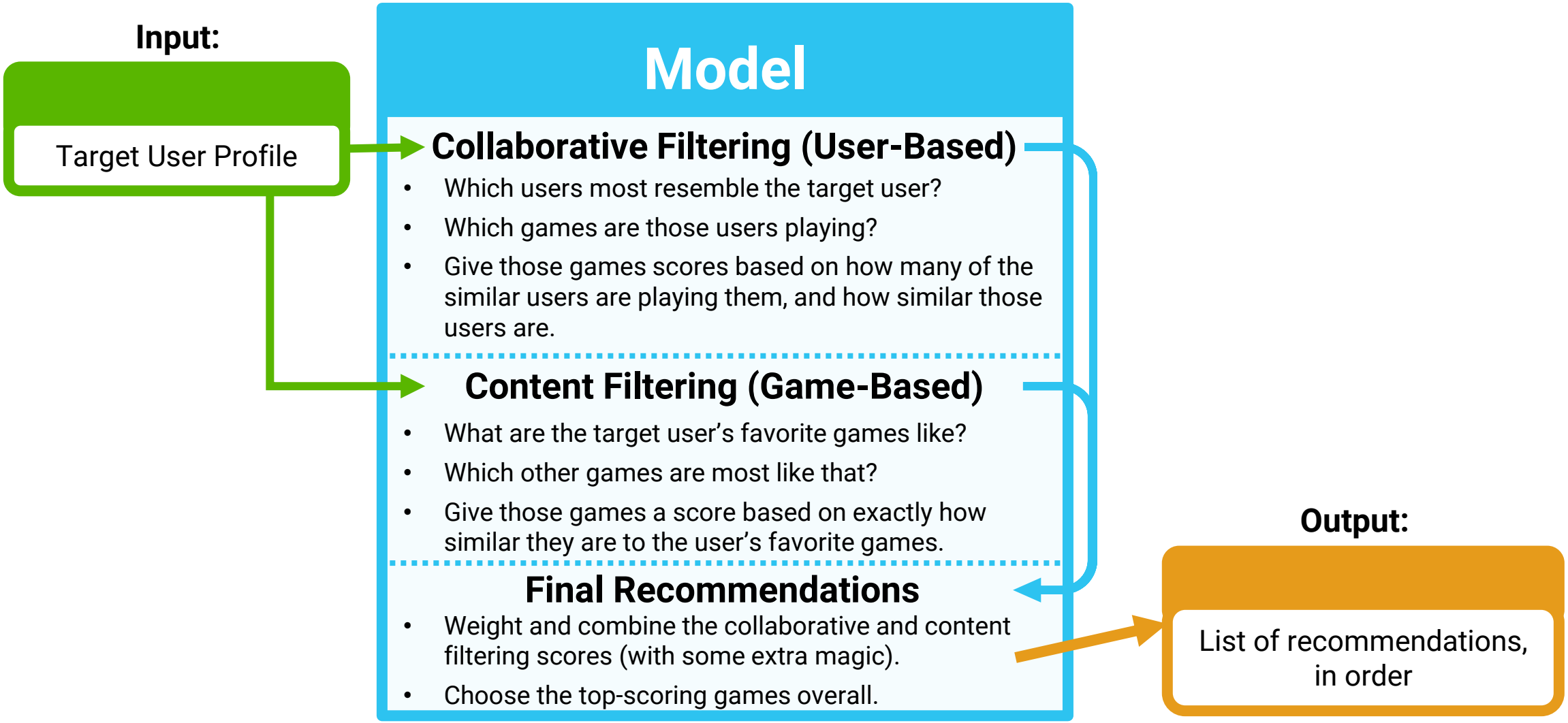
Preprocessed Users Tables

Condensed Sparse Row Format

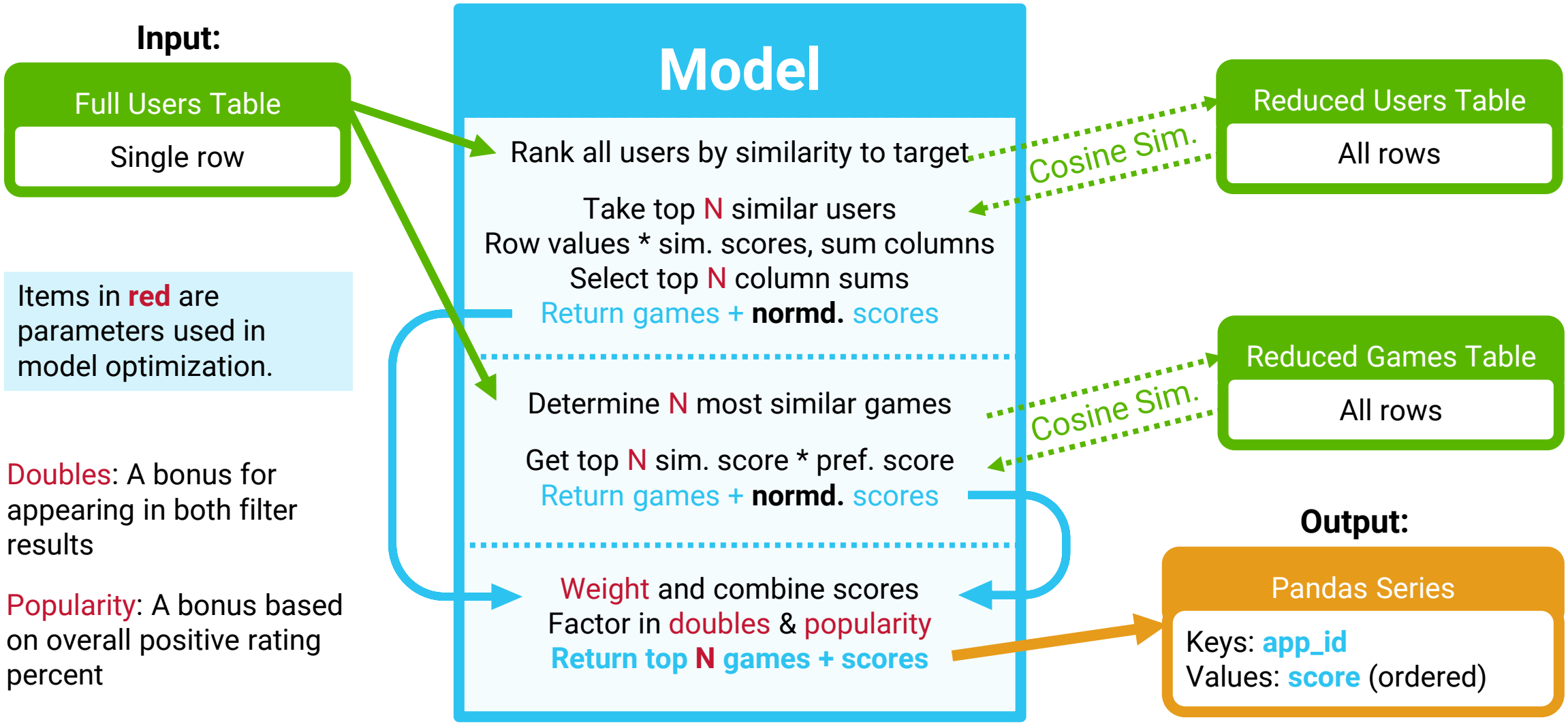
	game 1 col index	game 2 col index	game 3 col index	… (100k)
user index 1	1	0	0.2	…
user index 2	0.2	0.2	0	…
user index 3	0	-1	0.2	…
… (1.9mil)	…	…	…	…

- “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)

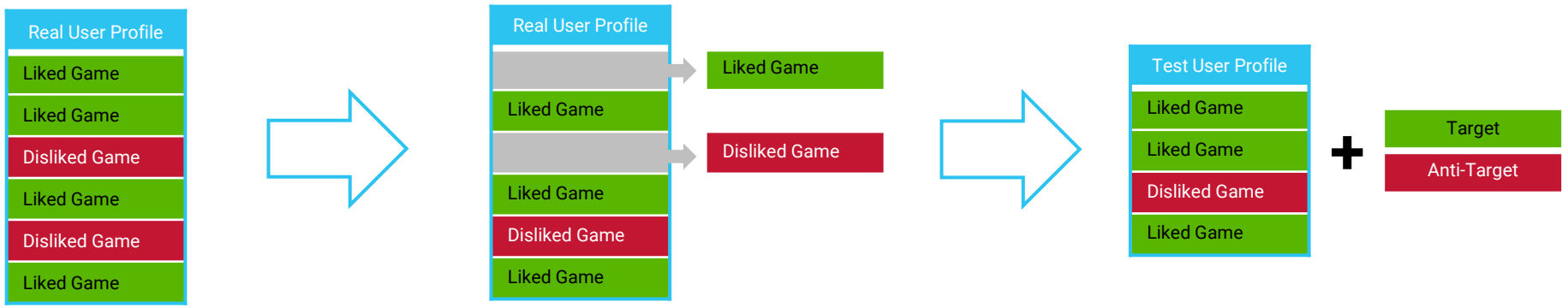


Modeling Flow (Complexified)



Testing & Evaluation Process

1. **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