Project Proposal:

Matteo Gläser, Sonja Dagn, Ilyas Satik

Question 1: *(alternative question at the end)*

Improving Recommendations by including User-Specific Constraints, based on Domain Knowledge.

There are certain fields where it is conceivable users have soft/hard constraints, on properties of items, that if violated, immediately disqualify an item as a recommendation. Some examples could be:

- Climbing Routes, Sports (Current Skill level Constraints)
- Travel/Booking (Budget Constraints)
- Legal Constraints
- **Physical Constraints in Rhythm Games like osu!**

Recommenders may inherently capture such constraints reasonably well. But could this be improved if we calculate/estimate such constraints on a user-specific level, and then include them explicitly into the recommendation's algorithm? Masking already seen/consumed items is in a way a basic version of this and used often.

- How are such constraints best implemented (filtering, weighting, cost function, constraint solving) and does this differ for different recommendation algorithms?
- Does this improve the precision of the algorithm, since we avoid items, that we assume can only be total misses?

Motivation:

In all honesty, we wanted to explore recommendations within the domain of rhythm-games, because it is almost unexplored but very interesting. This came up as one of the questions that applies particularly well here, but one that can also be tested nicely. We assume, that this could be a simple technique to achieve Improvements, with the potential to make it more sophisticated. We want to test this hypothesis and determine the best ways of implementing something like this.

Dataset:

Context:

The dataset we are going to use is player data/player Scores, in the popular rhythm game **osu!.** Players play *Beatmaps,* where a song plays, and circles and other objects appear in rhythmic patterns. Players must aim at the objects and tap/hold them while keeping rhythm. Performance points are awarded; The game has a competitive element and leaderboards.

Source:

We sourced the dataset ourselves. Most of the data was acquired through a link to a datadump from the main developer and creator of the game. It was then organized into a MySql database and further extended through calls to a public API of the game.

Structure:

Within the database, organized in several tables, but conceptually:

Users: [10.000 random; 10.000 top ranked]

- Country
- Performance-, leaderboard-, activity/playtime-features
- Liked Beatmap list (not yet) (valuable)

Items (Beatmaps) [200.000 – all available "ranked" maps]:

- .osu file, meta data and full hit-object data
- Around 20 relevant set-, musical-, leaderboard- and mapping-features.
- Several more dynamically calculated gameplay features: difficulty rating, pp, aim, speed, strain, ...
- Full table of how mod combinations alter the features above.
- fail/exit heatmap
- Mp3? (not yet)

Interactions (Scores) [50-100 million] + [5-10 million for the random users]

Best Score for each User on each Beatmap (+ Mod Combination)

- < 10 performance/leaderboard features for best attempt
- total play counter
- timestamp

Previous Work

Constraint-based recommendation has been explored in diverse domains by encoding hard requirements directly into the retrieval process. Le et al. (2023) introduce a vehicle-purchase recommender in which user requirements such as budget, compatibility rules, and legal constraints are modeled as variables in a constraint satisfaction problem and solved via SPARQL queries over an RDF knowledge graph[i]. Moreno et al. (2019) apply a similar idea to smart tourism itinerary planning, combining collaborative filtering with a constraint solver to enforce time windows, walking-distance limits, and venue opening hours so that every proposed route is physically and temporally feasible[ii]. Work by Felfernig and Burke (2008) formalizes how to capture arbitrary "hard" constraints, such as mandatory product features or budget limits, for recommender in an e-commerce setting[iii].

Within the osu! community, some recommender systems have been developed to support beatmap discovery.

Tillerino recommends maps drawn from a player's top ten scores[iv], often biasing toward "farm" maps. Dynam1cBOT generates a "skill fingerprint" from a user's best performances and suggests one easier, one matched, and one harder map via a hybrid collaborative-filtering and rule-based approach. AlphaOsu![v] focuses exclusively on maximizing performance points (pp) for competitive players. More recently, m1tm0 has developed a Bayesian Personalized Ranking–based system (discussed on Reddit) that incorporates implicit enjoyment metrics and mod sensitivity, and other gameplay related filters. (incorporating some User-Set Constraints into recommendations)

Some of these work quite well and demonstrate the feasibility of recommender systems in osu!, but they have severe limitations as well. None incorporate physical constraints, potentially automatically estimated on a per-user-basis.

Methodology:

- Establish a solid "Rating/Enjoyment" metric from multiple interaction signals.
- Develop a recommender system (maybe multiple), overcoming the domain specific challenges (detailed below). Test this as a baseline.
- Develop a method of calculating constraints around approach rate, maximum (sustained) hitrate, bpm... for each user.
- Incorporate into the recommendations in different ways. Does it improve the baseline?
- Can we make the constraints more sophisticated, what works what doesn't?
- Find other fields where user-specific constraints might intrinsically exist and test the approach.

Challenges/Potential Solutions

Domain specific:

- Mod – Beatmap interaction - "Exploding Items"
  - ➢ Collapse preference combinations, exclude gimmick combinations, treat competitively relevant combinations as separate items.
- Heavy Popularity and Farm Bias
  - ➢ Algorithmic Solutions
- Sequential Nature of Data
  - ➢ Filter for Scores in a timeframe of comparable performance point level
- **How to capture Enjoyment of an item, (basic metrics will not suffice)**
  - ➢ Crucial for evaluation as well.
  - ➢ Exploit richness of features
  - ➢ Rely on personal experience, try to come up with something smart.

Question 2:

Recommending to users with fundamentally different motivations.

Motivation: This applies particularly well to osu! again, where users fall on a spectrum between recreational and competitive. This leads to the immediate problem, that implicit signals for some users might indicate enjoyment while for others they might indicate the exact opposite.

Example: A recreational player might replay a map a lot of times, because they really love it, while for a competitive player tons of replays indicate a painful/unsuccessful grind. What they are looking for are maps, where they can get a high amount of performance points with as few attempts as possible.

Assumption: Any "Rating"-metric we come up with, maybe a combination of tons of implicit feedback features, probably won't fit all users very well, if their motivations are so different.

Methodology: Research first, then: How can we address that? Let's try if a ML-Based RecSys**,** which learns weights of our combined metric on a per user basis, captures this better (Or something similar).

Challenge: Mainly the evaluation. We would probably need some "true enjoyment" metric, that we just define very smartly based on domain knowledge, or through community sourcing. Then test, whether a simple combined metric or the one with user-specific weights performs better.

---

[i] https://arxiv.org/abs/2307.10702
[ii] https://doi.org/10.1080/10941665.2019.1592765
[iii] https://dl.acm.org/doi/10.1145/1409540.1409544

[iv] https://github.com/Tillerino/Tillerinobot/wiki
[v] https://alphaosu.keytoix.vip/self/pp-recommend