An Aggregator for Recommendation Engines

Max Kirker Burton

2260452b

Proposal

Motivation

Recommendation engines are increasingly used in a variety of services. However, current effectiveness is far from optimal, and users often do not get recommended things they want. These recommendations could be made more effective by aggregating existing algorithms.

Aims

This project will develop an aggregator that compares several recommendation algorithms and creates a new, ideally more effective list of recommendations. This will take a user's profile as input and return a ranked list as output. The effectiveness of the project will be measured against a 'golden list' and compared with non-aggregated algorithms.

Progress

- Language chosen: project will be implemented in Python (many recommendation libraries exist for Python)
- Experimentation and research on the libraries and the concepts behind recommender systems
- Data handling implemented (including methods that split up a dataset)
- First algorithm implemented (K-Nearest Neighbours)
- Basic prototype created (a simple federator that works on split up datasets)
- Basic metric measurement method added, used to compare non-aggregated and aggregated algorithms to a 'golden list'

Problems and risks

Problems

- Pandas library does not perform well/at all on large data (i.e. >int32) which this project requires. As such, many workarounds and custom helper methods were needed.
- Many different algorithms/libraries/aggregation methods to consider, so it is time-consuming to experiment with them all to find ideal ones.

Risks

- Unsure of the best way to map and scale the results of each algorithm together **Mitigation:** Have planned a meeting with Richard McCreadie to discuss this
- Finding a good 'golden list' is very difficult and would often be hand-picked with expert knowledge. **Mitigation:** Research more on existing 'golden lists' and in the meantime, use non-aggregated lists as a baseline.

Plan

- Week 1: Improve metric measurements using MAP@K, MAR@K and NDCG metrics
    - **Deliverable:** modular method that can be used as a scoring method for scikit-learn's gridsearch
- Week 2-3: Have algorithms take in a user profile by generating recommendation lists based on their highest rated movies.
    - **Deliverable:** algorithms that now generate recommendations on several movies but still only output one recommendation list
- Week 4-5: Implement at least 2 more algorithms.
    - **Deliverable:** well-tested algorithms that take in the same input (user profile) and returns a list in the same format as the KNN algorithm
- Week 6-7: Create final federator
    - **Deliverable:** a federator that takes in one user profile, gets several recommendation lists and returns one, ideal list
- Week 8: Run evaluation experiments
    - **Deliverable:** figures showing comparative performances of the federator
- Week 9-10 Write Up
    - **Deliverable:** first draft submitted to supervisor 2 weeks before final deadline