Recommender Systems Challenge 2023

Lorenzo Campana

Politecnico di Milano

February 21, 2024

Personal Code: 10605775

Overview

- $\approx 600k$ user-book interactions ($\approx 13k$ users, $\approx 22k$ books)
- Implicit ratings
- Slides results consider 80% training, 20% test
- Evaluation metric: MAP@10

$$\text{MAP@}K = \frac{1}{K}\sum_{u=1}^{N}\frac{1}{\min(K,m)}\sum_{k=1}^{K}P(k) * rel(k)$$

Baseline

Top Popular algorithm

Identify the most frequently occurring books

- Non Personalized
- MAP@10 $= 0.011742$
- How was it used?
  - To provide recommendation to new users

Hyperparameter tuning

- Optuna
- 5 Folds Cross Validation
- AWS free tier ≈ €2.25

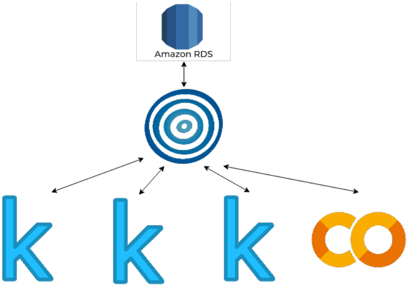

Figure: Optuna easy parallelization

Directory tree structure

```
...
├── Recommenders
│   .....
│   ├── XGboostRecommender.py
├── best_models.ipynb
├── best_models_info.pickle
├── Builder.py
├── dataset
│   └── book_dataset
│       ├── data_target_users_test.csv
│       └── data_train.csv
├── saved_models
│   ├── URM.npz
│   ├── URM_test.npz
│   ├── URM_train.npz
│   ├── URM_train_val.npz
│   ├── URM_val.npz
│   ├── cross_val
│   │   ├── train
│       ...
├── submissions
│   └── xgboost_final.csv
├── tuners
│   ├── KNN
│   │   ├── graph_tuners.py
│   │   ├── knn_ml.py
│   │   └── knn_tuners.py
│   ├── MF
│   │   └── mf_tuners.py
│   ├── hybrid
│   │   ├── hybrid_tuner.py
│   │   └── xgboost_tuner.py
│   ├── neural
│   │   └── neural_tuner.py
│   └── tuner.py
├── xgboost.ipynb
└── xgboost_tuner_run.py
```

Figure: Directory tree structure

Recommendation Algorithms Comparison

KNN Collaborative Filtering

- Item-Based
  - tversky
  - MAP@10 $= 0.04596$
- User-Based
  - TF-IDF
  - MAP@10 $= 0.03493$

Graph-based

- P3alpha
  - $k = 40$
  - MAP@10 $= 0.04793$
- RP3beta
  - $k = 28$
  - MAP@10 $= 0.04899$

Recommendation Algorithms Comparison

Matrix Factorization

- IALS
  - $k = 80$ latent factors
  - MAP@10 $= 0.03230$

Neural

- MultVAE
  - MAP@10 $= 0.037868$

Item-Based CF

- SLIMElasticNet
  - $k = 570$
  - MAP@10 $= 0.05003$
- SLIMBRP
  - MAP@10 $= 0.03893$

Hybrid Models

ScoresHybrid

- `SLIMElasticNet`, `RP3beta`
- $\alpha = 0.50282$
- $\text{MAP@10} = 0.05104$

$$\tilde{R} = \alpha \cdot \tilde{R_A} + (1 - \alpha) \cdot \tilde{R_B}$$

XGBoost (Preprocessing)

- Candidate generation selection
- Candidate generation cut-off selection
- Feature engineering
- Categorical features encoding

XGBoost (Candidate Generation)

- **ScoresHybrid**
  - `SLIMElasticNet`, `RP3beta`
  - Metric: `Recall`
  - Cut-off = 35
  - $\alpha = 0.36771$
  - RECALL@35 = 0.24885
- `SLIMElasticNet`
  - Metric: `Recall`
  - Cut-off = 35
  - RECALL@35 = 0.24221
- `RP3beta`
  - Metric: `Recall`
  - Cut-off = 35
  - RECALL@35 = 0.24156

XGBoost (Feature Engineering)

- Chosen scores:
  - `ScoresHybrid`
  - `SLIMElasticNet`
  - `RP3beta`
  - `ItemKNNCF`
  - `UserKNNCF`
  - `TopPop`
  - `IALS`
  - `MultVAE`
- Metric: `NDCG`
- Cut-off = 10

- k=[3, 10, 20]
  - `is_in_top_k()`
  - `count_agreement()`
- `score_stats()`
  - `mean`
  - `...`
  - `iqr`
- `normalize_scores()`
- `disagreement()`
- $n_{groups} = 20$
  - `user_profile()`
  - `item_profile()`

XGBoost (Categorical Features)

- naive `enable_categorical` = *True* (Used approach)
- Truncated SVD ($k = 100 \approx 10\%$ explained variance)
- What I did not tried:
  - Target encoding
  - Matrix Factorization
  - One-hot encoding
  - Embeddings

XGBoost (Feature Importance)

- MAP@10 = 0.051485

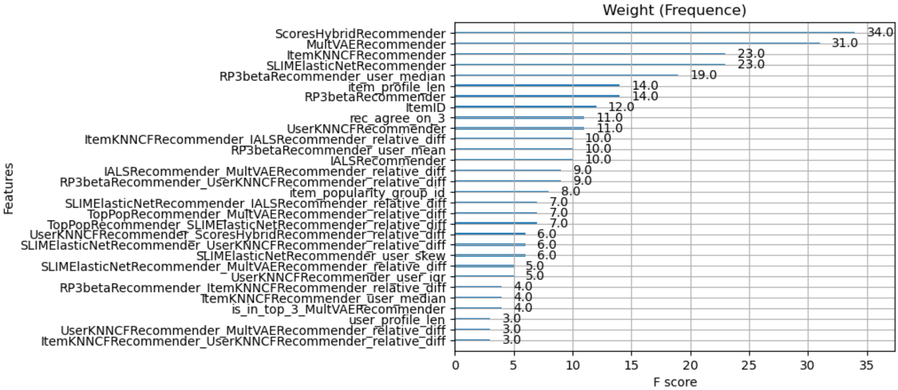

Figure: Feature Importance

Conclusion

- Thank you for your attention!
- Feel free to check out the project on GitHub:
  `https://github.com/lorecampa/RecSysChallenge2023`