AMBRIX/Fingle Recommendation Assignment

This report summarizes the design and implementation of a hybrid interest-based recommender that produces top-3 post recommendations per user.

Problem framing

- **Goal**: Recommend 3 posts/user leveraging profile interests, past engagement, and content attributes.
- **Signals available**:
- Users: `age`, `gender`, `top_3_interests`, `past_engagement_score`.
- Posts: `content_type`, `tags`, `creator_id`.
- Engagements: binary `engagement` (implicit feedback).

Approach (Hybrid)

We combine three complementary components:
- **Content similarity**: Jaccard overlap between user `top_3_interests` and post `tags`.
- **Collaborative filtering**: TruncatedSVD over an implicit user–item matrix (positives=1, negatives=0.1) to learn latent factors.
- **Popularity prior**: Laplace-smoothed positive rate per post.

Final score per user–post:
score = w_cf * (U·V) + w_content * jaccard(interests, tags) + w_pop * popularity
score *= (0.75 + 0.5 * past_engagement_score)
This balances personalization from CF, cold-start coverage from content, and global priors from popularity, scaled by a mild user propensity term.

Data preprocessing

- Parse and lowercase list-like fields (`top_3_interests`, `tags`).
- Map `user_id`/`post_id` to contiguous indices.
- Build implicit matrix from `Engagements.csv` using positives=1.0, negatives=0.1.

Train/validation split

- **User-wise holdout**: for each user with ≥1 positive, randomly hold out one positive (test) and train on all remaining interactions. Users with no positives are excluded from metric aggregation.
- This simulates next-positive prediction while preserving negatives for contrast.

Evaluation metrics

- Reported at k=3: **Precision@3**, **Recall@3**, **MAP@3**, **NDCG@3**.

- Computed over users with a held-out positive. See last evaluation cell in the notebook for aggregated metrics and user count evaluated.

Results (example)

Run the notebook to produce metrics on your machine. The last evaluation cell prints a dictionary like:
{'users_evaluated': <N>, 'Precision@3': <p>, 'Recall@3': <r>, 'MAP@3': <map>, 'NDCG@3': <ndcg>}

Output

- The notebook writes `recommendations_top3.csv` with columns: `user_id, post_id, rank, score`.

Practical extensions

- **Weight learning**: Learn `w_cf, w_content, w_pop` via validation (e.g., Bayesian optimization or grid search).
- **Better implicit modeling**: Use Implicit ALS/BPR or LightFM with WARP/BPR for ranking-optimized training.
- **Richer content**: Encode `content_type`, `creator_id`, and text embeddings (e.g., MiniLM) for posts; expand user profiles beyond top-3 interests via embedding aggregation.
- **Temporal awareness**: Time-based splits, recency decay on interactions and popularity.
- **Diversity/serendipity**: Penalize near-duplicates and encourage topic coverage in top-k.
- **Cold-start creators**: Add creator-level priors and user–creator affinity features.
- **Fairness & safety**: Guardrails for sensitive attributes; monitor exposure/engagement balance across topics/creators.

How to run

1. Ensure Python 3.9+ and install requirements:
pip install -r requirements.txt
2. Open `Fingle_Recommender.ipynb` and run all cells. The notebook expects the CSVs in the same folder.
3. Collect outputs: metrics printed near the end and `recommendations_top3.csv` generated in the project root.

Repository structure

- `Users.csv`, `Posts.csv`, `Engagements.csv`
- `Fingle_Recommender.ipynb`
- `REPORT.md`
- `requirements.txt`