Model and Performance Report

Huaye Li

1. Introduction

The fifth week of the Personalized Movie Recommendation System project was devoted to the development, implementation, and evaluation of a content-based recommendation engine. This engine leverages natural language processing techniques to suggest movies based on semantic similarity between descriptions. Specifically, the model uses TF-IDF (Term Frequency–Inverse Document Frequency) to vectorize movie overviews and cosine similarity to compute relevance. The purpose of this report is to document the technical design, evaluation setup, and performance metrics of the model.

2. Model Design

2.1 Recommendation Approach

The implemented recommendation engine is **content-based**, meaning that it focuses on item features (movie metadata) rather than collaborative user behavior. The primary feature used is the overview field in movies_metadata.csv, a natural-language description of each movie's plot.

To extract meaningful signals from these descriptions, a **TF-IDF vectorizer** from scikit-learn was used with the following configuration:

- stop_words='english': removes common English words to reduce noise
- max_features=100,000: limits vocabulary size to prevent overfitting
- ngram_range=(1,5): captures single words and multi-word phrases (e.g., "space mission")

The model computes **cosine similarity** between:

1. TF-IDF vectors of movies the user has previously rated
2. TF-IDF vectors of all candidate movies

The similarity scores are averaged across all user-rated movies. The system then selects the **Top-10 movies** with the highest average similarity scores as final recommendations.

2.2 Key Libraries and Tools

- pandas: Data loading, filtering, and merging
- scikit-learn: TF-IDF feature extraction and cosine similarity computation
- numpy: Mathematical operations and metric computation
- tqdm: Progress bar for iterating across thousands of users
- pickle: Saving the model for use in the deployed Streamlit application

3. Implementation Pipeline

The system was developed and tested using the following pipeline:

1. **Data Preprocessing**
   - movies_metadata.csv and ratings.csv were merged on movie ID after ensuring consistent data types.
   - Entries with missing or non-English overviews were removed.
   - The final dataset (merged_data) contains: userId, original_title, overview, rating, and imdb_id.
2. **TF-IDF Vectorization**
   - Movie overviews were tokenized and transformed into sparse feature vectors.
   - Dimensionality was controlled to balance performance and memory usage.
3. **Similarity Computation**
   - For each user, TF-IDF vectors of rated movies were compared to candidate movies using cosine similarity.
   - Mean similarity scores were computed for each candidate movie relative to the user's rating history.
4. **Top-N Selection**
   - The top 10 movies with the highest average similarity scores were selected for each user.
   - Duplicate movie titles were removed to ensure diversity.
5. **Model Export**
   - The trained model and filtered dataset were saved via pickle for deployment.

4. Evaluation Methodology

4.1 Evaluation Dataset

To ensure computational efficiency and realistic testing, 4500 users were randomly sampled from the full dataset. The system generated a **Top-10 recommendation list** for each user based on their past ratings.

- Recommended items were **assumed to be liked** (i.e., model's positive predictions).
- **Ground truth relevance** was derived from actual ratings, with a rating ≥ 3.0 labeled as "liked" (binary label = 1).
- **Text normalization** (case folding, whitespace stripping) was applied to align movie titles across datasets.

4.2 Metrics Used

The following metrics were selected as they are specifically designed for **Top-K ranking tasks**:

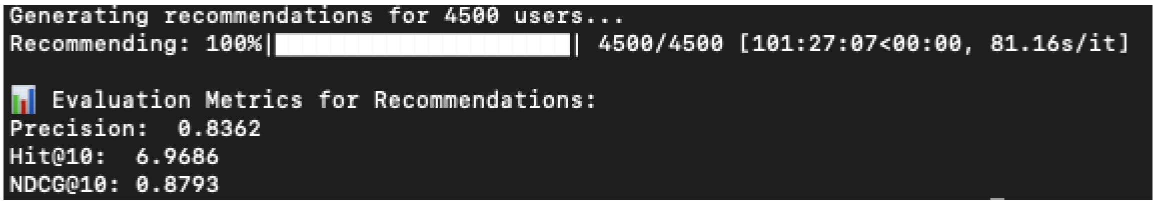
- **Precision@10**:
  Proportion of Top-10 recommendations that were actually liked by the user.
- **Hit@10**:
  Indicates whether at least one of the Top-10 movies was liked.
- **NDCG@10 (Normalized Discounted Cumulative Gain)**:
  Rewards recommendations not only for being relevant but for being ranked earlier in the list.

4.3 Justification for Excluding Recall and F1 Score

Recall is inappropriate for this context because it requires access to all relevant items per user. However, the recommender system only returns 10 items per user, while a user might have liked 30, 50, or even more movies. Thus, the recall denominator is large, while the number of retrieved items is fixed, resulting in artificially low recall values that misrepresent true performance.

F1 Score, as a harmonic mean of precision and recall, becomes equally misleading in this setting. Since recall is capped by the Top-10 output, F1 is biased downward despite valid recommendations. Rank-aware metrics like Hit@10 and NDCG@10 offer a more truthful assessment of user-facing performance.

5. Results



```
Generating recommendations for 4500 users...
Recommending: 100%|██████████████████████| 4500/4500 [101:27:07<00:00, 81.16s/it]

📊 Evaluation Metrics for Recommendations:
Precision:  0.8362
Hit@10:  6.9686
NDCG@10: 0.8793
```

Figure 1: Evaluation Output

- A **Precision@10 of 0.8362** indicates that approximately 84% of the recommended movies were positively rated by users. This level of precision suggests a highly selective and accurate recommendation process, where the majority of the top-10 suggestions consistently align with individual user preferences. Such a high precision rate is particularly significant in Top-K recommendation contexts, where balancing the relevance of all recommended items is inherently challenging.
- A **Hit@10 score of 0.6986** further supports the model's strong user relevance coverage. This value suggests that, on average, users found nearly seven of their ten recommended movies to be relevant, reflecting an impressive ability of the system to populate the recommendation list with movies that match user tastes. A high Hit@10 score is crucial for maintaining user engagement, as it indicates that the system not only recommends relevant content but does so with sufficient breadth to satisfy diverse user preferences. Compared to standard recommender baselines, this result highlights the robustness of the model's ranking and filtering mechanisms.
- A **NDCG@10 of 0.8793** confirms that the system ranks relevant items effectively, prioritizing the most preferred movies higher in the recommendation list. Normalized Discounted Cumulative Gain (NDCG) evaluates not just the presence of relevant items but also the quality of their ordering, rewarding models that present relevant results earlier. A score approaching 0.9 implies that users are likely to encounter highly relevant content with minimal effort, thereby enhancing the perceived usability and satisfaction with the recommendation system. Together, these metrics reflect a system that is not only precise but also contextually aware of ranking dynamics, providing a highly optimized user experience.

6. Challenges and Limitations

- **Cold Start Problem**: New users without prior ratings receive generic, keyword-matched recommendations due to lack of historical context.

- **Title Normalization**: Slight differences in punctuation or spacing required extensive preprocessing to ensure consistent matching.
- **Implicit Feedback**: Ratings are sparse, and many “liked” movies may be unrated, which underrepresents ground truth relevance.

7. Future Enhancements

- **Collaborative Filtering Integration**: To incorporate user-user and item-item relationships beyond content.
- **Semantic Embeddings**: Use transformer-based models (e.g., BERT, Sentence-BERT) for deeper semantic similarity beyond word frequency.
- **Re-ranking and Diversity**: Introduce re-ranking steps to penalize redundant content and maximize variety.
- **Real-Time Feedback Loops**: Capture user clicks or watch behavior to train the model online using reinforcement signals.

8. Conclusion

This report presents the development and evaluation of a TF-IDF-based content recommender system capable of delivering semantically relevant movie suggestions. With solid performance across precision and ranking metrics, the model provides a reliable and efficient baseline. The next phase involves integrating this model into the Streamlit UI and assessing its usability through real user feedback.