Restaurant Recommendation System

Chaitanya Shashi Kumar, Sushmitha Muppa and Farzan Mirza

Drexel University

Abstract—This project presents a restaurant recommendation system designed to help users find dining options based on their preferences. Using the Entree Chicago Recommendation Data, which contains restaurant features such as cuisine type, ambiance, and service style, the system generates recommendations without explicit user interaction data. We implement three approaches: Item-Based Collaborative Filtering, which compares restaurants based on shared features; Content-Based Filtering, which uses feature similarity; and a Hybrid Approach that combines both methods for improved accuracy. Additionally, we incorporate an implicit feedback model using Singular Value Decomposition (SVD) to learn latent patterns from user behavior, even in the absence of explicit ratings. To evaluate performance, we use Precision & Recall, Jaccard Similarity Score, and Normalized Discounted Cumulative Gain (NDCG). Our goal is to provide accurate, diverse, and personalized recommendations that enhance user experience and restaurant discovery.

1. Introduction

Finding the right restaurant can be overwhelming, especially in cities with numerous dining options. This project aims to develop a restaurant recommendation system that assists users in discovering new dining places based on their preferences. Using the Entree Chicago Recommendation Data, which contains restaurant attributes but lacks explicit user ratings, we employ three main recommendation approaches: Item-Based Collaborative Filtering, which compares restaurants based on shared features; Content-Based Filtering, which represents restaurants as feature vectors and computes similarity scores; and a Hybrid Approach that integrates both techniques for improved accuracy. Additionally, we incorporate an implicit feedback model using Singular Value Decomposition (SVD) to learn hidden patterns from user interactions, even in the absence of explicit feedback. To ensure high-quality recommendations, we evaluate the system using Precision & Recall to measure relevance, Jaccard Similarity Score to assess feature overlap, and Normalized Discounted Cumulative Gain (NDCG) to evaluate ranking quality. The goal is to provide accurate, diverse, and personalized restaurant suggestions that enhance user experience and simplify the decision-making process.

2. Domain Definition

The domain of this project is **restaurant recommendations**, focusing on helping users find suitable dining options based on their preferences. The system operates in the broader domain of **recommender systems**, which suggest relevant items to users based on various factors. Within this domain, our project specifically targets **feature-based restaurant recommendations**, where recommendations are made using restaurant metadata such as cuisine type, ambiance, and service style rather than explicit user ratings. The goal is to assist users in discovering new restaurants that match their tastes while ensuring diversity and personalization in recommendations.

3. Intended Users & Use Case

The primary users of this restaurant recommendation system are individuals looking for dining options that match their personal preferences. This includes locals exploring new restaurants, tourists unfamiliar with the area, and food enthusiasts seeking specific cuisines or dining experiences. Users may have different requirements, such as dietary restrictions, budget constraints, or preferences for certain ambiance styles.

The system addresses the common challenge of restaurant selection by recommending options based on metadata such as cuisine type, service quality, and dining atmosphere. Users interact with the system by inputting their preferences or browsing suggestions, after which the system generates personalized recommendations. The recommendations help users save time, discover new restaurants, and make informed dining choices. By leveraging a feature-based approach, the system ensures that users receive relevant, diverse, and high-quality suggestions tailored to their needs.

4. User Interaction with the System

Users interact with the system by entering their preferences, such as cuisine type, budget, or ambiance, or by browsing recommended restaurants. The system processes this information and provides a ranked list of suitable options. Users can refine their search using filters or provide implicit feedback through clicks and selections, which helps improve future recommendations. The goal is to offer a simple and personalized experience for discovering new dining options.

5. Data Used in the Applications

The restaurant recommendation system is built using data from the **Entree Chicago Recommendation Data**, which includes information from multiple restaurant datasets. The key datasets used are:

- **Boston.txt and Chicago.txt**: These files contain a list of restaurants from Boston and Chicago, along with associated feature codes that represent different restaurant attributes.
- **Features.txt**: This file provides a mapping of feature codes to actual restaurant characteristics, such as cuisine type (e.g., Italian, Mexican, Japanese), dining options (e.g., Takeout Available, Fine for Dining Alone), price range (e.g., Below $15, $15-$30), and ambiance (e.g., Romantic, Quiet for Conversation).

Each restaurant in the dataset is represented by a set of categorical features that describe its characteristics. Since explicit user ratings are not available, the system relies on these attributes to generate recommendations based on feature similarity. The data is processed to extract meaningful patterns, allowing for personalized and relevant restaurant suggestions.

6. Algorithm Selection and Justification

Our restaurant recommendation system employs three primary algorithms: **Item-Based Collaborative Filtering**, **Content-Based Filtering**, and an **Implicit Feedback Model using Singular Value Decomposition (SVD)**. Below, we justify the selection of each algorithm and provide relevant code snippets.

6.1. Content-Based Filtering

Content-Based Filtering recommends restaurants based on similarity in features such as cuisine type, ambiance, and price range. Each restaurant is represented as a feature vector using TF-IDF (Term Frequency-Inverse Document Frequency), and recommendations are generated based on cosine similarity.

```
from sklearn.feature_extraction.text import
    ↪ TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Convert restaurant features into TF-IDF vectors
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['Features'])

# Compute similarity scores
cosine_sim = cosine_similarity(tfidf_matrix,
    ↪ tfidf_matrix)

```

Code 1. TF-IDF Feature Extraction and Similarity Computation

Justification: TF-IDF is used to weigh feature importance, ensuring that rare but significant features (e.g., "Michelin Star") contribute more to recommendations than common features (e.g., "Casual Dining"). Cosine similarity then helps in identifying the most similar restaurants.

6.2. Item-Based Collaborative Filtering

Item-Based Collaborative Filtering compares restaurants based on shared features to determine similarity. In this approach, we represent features as binary vectors and use similarity measures such as Jaccard similarity.

```
from sklearn.metrics import jaccard_score

# Function to compute Jaccard similarity
def jaccard_similarity(vec1, vec2):
    return jaccard_score(vec1, vec2, average='micro')

# Example: Compute similarity between two restaurants
similarity_score = jaccard_similarity(feature_matrix[0],
     ↪  feature_matrix[1])

```

Code 2. Jaccard Similarity Computation for Item-Based Filtering

Justification: Jaccard similarity is well-suited for categorical feature-based recommendation systems, as it measures the proportion of shared attributes between two restaurants.

6.3. Implicit Feedback Model with SVD

Since explicit user ratings are unavailable, we leverage Singular Value Decomposition (SVD) on an implicit feedback matrix derived from user interactions such as restaurant clicks, visits, or selections.

```
from scipy.sparse.linalg import svds
import numpy as np

# Create a user-restaurant interaction matrix
interaction_matrix = df.pivot(index='User_ID', columns='
     ↪ Restaurant_ID', values='Interaction').fillna(0)

# Apply SVD
U, sigma, Vt = svds(interaction_matrix, k=50)
sigma = np.diag(sigma)

# Compute predicted interaction scores
predicted_matrix = np.dot(np.dot(U, sigma), Vt)

```

Code 3. Singular Value Decomposition (SVD) for Implicit Feedback

Justification: SVD helps in discovering hidden patterns in user interactions, enabling personalized recommendations even without explicit ratings. The matrix factorization technique captures latent factors that influence restaurant preferences.

6.4. Hybrid Approach

To enhance recommendation accuracy, we combine Content-Based Filtering and Collaborative Filtering by weighting their scores and normalizing them.

```
# Compute final recommendation score as a weighted sum
alpha = 0.6  # Weight for content-based filtering
beta = 0.4   # Weight for collaborative filtering

final_score = alpha * cosine_sim + beta *
     ↪ jaccard_sim_matrix

```

Code 4. Combining Content-Based and Collaborative Filtering Scores

Justification: The hybrid model balances both approaches, ensuring that recommendations consider both feature similarity and collaborative patterns, leading to a more accurate and diverse restaurant recommendation system.

7. Experimental Setup & Evaluation

Our restaurant recommendation system was evaluated using Cosine Similarity, Jaccard Similarity, TF-IDF with Cosine Similarity, Hybrid Recommendations, and an Implicit Feedback Model using Singular Value Decomposition (SVD). The evaluation was conducted on the Chicago and Boston restaurant datasets. Metrics used for evaluation include Precision, Recall, F1 Score, RMSE, and NDCG.

7.1. Cosine Similarity-Based Recommendations

Cosine Similarity measures the closeness of restaurant feature vectors.

```
from sklearn.metrics.pairwise import cosine_similarity

# Compute cosine similarity scores
cosine_sim = cosine_similarity(tfidf_matrix,
     ↪ tfidf_matrix)
```

Code 5. Cosine Similarity Computation

Results:

- RMSE: 0.0000
- Precision: 0.9890
- Recall: 0.2582
- F1 Score: 0.4095
- NDCG: 0.9968

7.2. Jaccard Similarity-Based Recommendations

Jaccard Similarity evaluates the overlap in categorical features between restaurants.

```
import numpy as np

def jaccard_similarity(matrix):
    intersection = np.dot(matrix, matrix.T)
    row_sums = matrix.sum(axis=1)
    outer_sum = np.outer(row_sums, row_sums)
    union = outer_sum - intersection
    return intersection / (union + 1e-10)

jaccard_sim = jaccard_similarity(feature_matrix)
```

Code 6. Jaccard Similarity Computation

Results: Jaccard-based recommendations had a maximum similarity score of 0.1077, indicating lower feature overlap compared to Cosine Similarity.

7.3. TF-IDF with Cosine Similarity

TF-IDF assigns weights to restaurant features, enhancing content-based recommendations.

```
from sklearn.feature_extraction.text import
     ↪ TfidfVectorizer
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['Features'])
tfidf_sim = cosine_similarity(tfidf_matrix, tfidf_matrix
     ↪ )
```

Code 7. TF-IDF and Cosine Similarity Computation

Results:

- RMSE: 0.0000
- Precision: 0.8162
- Recall: 0.7091
- F1 Score: 0.7589
- NDCG: 0.9984

7.4. Hybrid Model Performance

The Hybrid Model combines content-based and collaborative filtering approaches.

```
# Compute final hybrid score as a weighted sum
alpha = 0.6  # Weight for content-based filtering
beta = 0.4   # Weight for collaborative filtering
hybrid_score = alpha * cosine_sim + beta * jaccard_sim
```

Code 8. Hybrid Model Score Calculation

Results:

- Precision: 0.9890
- Recall: 0.2582
- F1 Score: 0.4095
- NDCG: 0.9968

7.5. Implicit Feedback Model Using SVD

The implicit feedback model learns user-restaurant preferences from interaction data using Singular Value Decomposition (SVD). Unlike previous approaches that rely on feature similarity, this model uses a user-item interaction matrix, where interactions (e.g., clicks, visits, orders) are encoded implicitly.

```
from scipy.sparse.linalg import svds
import numpy as np

# Create a user-restaurant interaction matrix
interaction_matrix = df.pivot(index='User_ID', columns='
    ↪ Restaurant_ID', values='Interaction').fillna(0)

# Apply SVD
U, sigma, Vt = svds(interaction_matrix, k=50)
sigma = np.diag(sigma)

# Compute predicted interaction scores
predicted_matrix = np.dot(np.dot(U, sigma), Vt)
```

Code 9. Training an Implicit Feedback Model using SVD

Results for Different Test Set Splits:

Test Set	Precision	Recall	F1 Score	NDCG
10%	0.0060	0.0068	0.0064	0.0051
20%	0.0080	0.0052	0.0063	0.0065
30%	0.0100	0.0043	0.0060	0.0081

Table 1. Performance of Implicit Feedback Model (SVD) for Different Test Splits

Observations: - Precision increases with test set size, reaching 1.0% at 30% test data. - Recall decreases as the test size grows, suggesting that the model struggles with generalization. - F1 Score remains low across test splits, indicating that neither precision nor recall is strong enough to make confident recommendations. - NDCG scores are very low, implying poor ranking quality of recommendations.

7.6. Analysis of Results

- **Cosine Similarity** and **TF-IDF Similarity** are the most effective methods, offering high precision and recall.
- The **Hybrid Model** maintains high precision but does not significantly improve recall.
- The **Implicit Feedback Model using SVD** performed poorly, with low precision, recall, and NDCG scores across all test set sizes.

Overall, content-based and hybrid filtering approaches significantly outperform the implicit feedback model, which may require additional training, better implicit interaction data, or alternative matrix factorization techniques.

7.7. Comparison of Recommendation Quality

Method	Precision	Recall	F1 Score	NDCG
Cosine Similarity	0.9890	0.2582	0.4095	0.9968
TF-IDF Similarity	0.8162	0.7091	0.7589	0.9984
Hybrid Model	0.9890	0.2582	0.4095	0.9968
Implicit Feedback (SVD)	0.0100	0.0043	0.0060	0.0081

Table 2. Performance Comparison of Recommendation Methods

8. Conclusion & Future Work

8.1. Conclusion

This project developed a restaurant recommendation system using Content-Based Filtering, Collaborative Filtering, and an Implicit Feedback Model (SVD). Evaluation showed that Cosine Similarity and TF-IDF performed best, achieving high precision (98.9%) and strong ranking quality (NDCG close to 1.0). The Hybrid Model maintained high accuracy but did not improve recall, while the Implicit Feedback Model performed poorly due to limited user interaction data. The results suggest that feature-based and hybrid filtering methods are more effective than implicit feedback models in this dataset.

8.2. Future Work

Future improvements include enhancing the Implicit Feedback Model with Alternating Least Squares (ALS) or Neural Collaborative Filtering (NCF) to handle sparse interaction data. Incorporating user profiles and real-time feedback can improve personalization. Additionally, deep learning approaches like Graph Neural Networks (GNNs) and context-aware recommendations (e.g., location, time, weather) could enhance recommendation quality. With these refinements, the system can provide more adaptive and personalized restaurant suggestions.