Evaluation Metrics of Recommender Systems

Estimated Time : 15 Minute

Objectives

- Describe the concept of recommender systems
- Explore the need of evaluation matrics
- Explain various evaluation metrics applicable to recommender systems
- Demonstrate how to compute these metrics using a sample dataset

Recommender systems

A recommender system is a type of information filtering system that predicts and suggests items or content that a user might be interested in. It utilizes data analysis techniques, algorithms, and user feedback to make personalized recommendations.
The primary objective of recommender systems is to enhance user experience by providing personalized recommendations tailored to individual preferences and interests. These systems aim to:

- **Increase user satisfaction:** By offering relevant and personalized recommendations, recommender systems aim to satisfy users' diverse needs and preferences, leading to higher user satisfaction.

- **Enhance engagement:** Recommender systems help users discover new and interesting items, leading to increased user engagement and interaction with the platform.

- **Drive Business Goals:** Personalized recommendations can drive sales, increase user retention, and boost revenue by promoting relevant products or content to users.

These are widely used in various domains, including e-commerce platforms, streaming services, social media platforms, news websites, and online learning platforms. They play a crucial role in improving user engagement, increasing user satisfaction, and driving business revenue by facilitating personalized recommendations tailored to individual user preferences and interests.

Need for evaluation metrics

Evaluation metrics are essential for assessing the effectiveness and performance of recommender systems. They serve several purposes:

- **Measure system performance:** Evaluation metrics provide quantitative measures of how well a recommender system is performing in terms of accuracy, relevance, and other key aspects.

- **Validate algorithm performance:** Metrics help validate the effectiveness of different recommendation algorithms and techniques, enabling researchers and developers to compare and select the most suitable approaches.

- **Identify areas for improvement:** By analyzing evaluation metrics, developers can identify weaknesses or limitations in the recommender system and prioritize areas for improvement.

- **Ensure user satisfaction:** Evaluation metrics help ensure that recommended items are relevant and aligned with user preferences, ultimately leading to higher user satisfaction and engagement.

Different evaluation metrics

To showcase the functionality and evaluation of recommender systems, we have provided a simulated code and a sample dataset for testing and analyzing various recommendation algorithms and evaluation metrics.

Let's see what the different evaluation metrics that can be applied to the unsupervised learning recommender systems:

1. Precision and recall

- **Precision**: In our content-based recommendation system example, precision represents the percentage of courses recommended to users that are relevant to their profiles out of all the courses suggested. For instance, if a user is interested in database, Python, and data analysis, precision would measure how many of the recommended courses match these preferences.

$$Precision = \frac{\text{Number of relevant recommended items}}{\text{Total number of recommended items}}$$

- **Recall**: Recall measures the percentage of relevant recommended courses that the system successfully retrieves out of all the relevant courses available in the dataset. In our example, recall would indicate how many of the courses the user likes are actually recommended to them.

$$Recall = \frac{\text{Number of relevant recommended items}}{\text{Total number of relevant items}}$$

- **F1 Score**: The F1 score combines precision and recall into a single metric, providing a balanced measure of the recommendation system's performance. It's the harmonic mean of precision and recall, giving equal weight to both metrics.

$$F1Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2. Diversity metrics

- **Intra-list diversity:** Intra-list diversity evaluates the diversity of recommended courses within a single recommendation list for a user. It ensures that the recommended courses cover various genres and topics, catering to diverse user preferences.

$$\text{Intra-List Diversity} = \frac{\text{Number of unique genres in recommended list}}{\text{Total number of recommended items}}$$

- **Inter-list diversity:** Inter-list diversity measures the diversity of recommendations across multiple users. It assesses whether the recommendation system can provide diverse recommendations to different users with varied preferences.

$$\text{Inter-List Diversity} = \frac{\text{Number of unique genres across all recommended lists}}{\text{Total number of recommended lists}}$$

3. Novelty metrics

- **Novelty:** Novelty measures the degree of uniqueness or unfamiliarity of recommended courses to users. In our example, it encourages the system to recommend courses that users may not have encountered before, thus promoting exploration and discovery.

4. Coverage

- **Catalog coverage**: Catalog coverage measures the proportion of unique courses recommended to users over the entire catalog of available courses. It indicates how well the recommendation system explores the entire range of available courses.

$$\text{Catalog Coverage} = \frac{\text{Number of unique recommended items}}{\text{Total number of items in the catalog}}$$

5. User engagement metrics

- **Click-through rate (CTR)**: CTR measures the ratio of users who click on recommended courses to the total number of users who receive recommendations. It reflects the effectiveness of the recommendation algorithm in capturing user interest and engagement.

$$\text{CTR} = \frac{\text{Number of users who click on recommended items}}{\text{Total number of users who received recommendations}}$$

6. Serendipity

- **Serendipity:** Serendipity measures the unexpectedness or surprise factor of recommended courses. It assesses whether the recommendation system can suggest courses that are not only relevant to the user's profile but also introduce new and interesting topics outside the user's typical preferences.

7. Efficiency

- **Scalability:** Scalability evaluates the efficiency of the recommendation algorithm in handling large datasets and increasing numbers of users and courses while maintaining reasonable response times.

Data Set Overview

The data set provides essential user profiles, course genres, and interaction data to simulate recommendation scenarios.
Here's a brief overview:

1. **Users:** Two users, user1 and user2, are included, each with distinct preferences in various fields such as database, Python, and machine learning.
2. **Courses:** The data set contains two courses, Course A and Course B, categorized into different genres like database, Cloud computing, and machine learning.

3. **Interactions:** User interactions with courses are recorded through ratings, indicating user interest or preference for specific courses.

Let's see how we can calculate the evaluation matrix of recommender system in unsupervised learning.

```
from collections import defaultdict
# Sample data
user_profile_data = {
    'user1': {'Database': 1, 'Python': 1, 'CloudComputing': 0, 'DataAnalysis': 1, 'Containers': 0, 'MachineLearning': 1, 'ComputerVision': 0, 'DataScience': 1, 'BigData': 0, 'Chatbot': 0, 'R': 1, 'Ba
    'user2': {'Database': 1, 'Python': 0, 'CloudComputing': 1, 'DataAnalysis': 1, 'Containers': 0, 'MachineLearning': 1, 'ComputerVision': 0, 'DataScience': 0, 'BigData': 1, 'Chatbot': 0, 'R': 1, 'Ba
}
course_genre_data = {
    'course1': {'Database': 1, 'Python': 0, 'CloudComputing': 1, 'DataAnalysis': 1, 'Containers': 0, 'MachineLearning': 1, 'ComputerVision': 0, 'DataScience': 0, 'BigData': 1, 'Chatbot': 1, 'R': 0, '
    'course2': {'Database': 0, 'Python': 1, 'CloudComputing': 0, 'DataAnalysis': 1, 'Containers': 1, 'MachineLearning': 0, 'ComputerVision': 1, 'DataScience': 0, 'BigData': 1, 'Chatbot': 0, 'R': 1, '
}
test_data = {
    'user': ['user1', 'user1', 'user2', 'user2'],
    'item': ['course1', 'course2', 'course1', 'course2'],
    'rating': [1, 0, 1, 1]
}
def precision_recall_f1(test_data, user_profile_data, course_genre_data):
    precision_sum = 0
    recall_sum = 0
    f1_score_sum = 0
    for user, item, rating in zip(test_data['user'], test_data['item'], test_data['rating']):
        if user in user_profile_data:
            relevant_courses = [key for key, val in user_profile_data[user].items() if val == 1]
            recommended_genres = [key for key, val in course_genre_data[item].items() if val == 1]
            true_positives = len(set(relevant_courses) & set(recommended_genres))
            precision = true_positives / len(recommended_genres) if len(recommended_genres) > 0 else 0
            recall = true_positives / len(relevant_courses) if len(relevant_courses) > 0 else 0
            precision_sum += precision
            recall_sum += recall
            f1_score_sum += 2 * ((precision * recall) / (precision + recall)) if (precision + recall) > 0 else 0
    precision_avg = precision_sum / len(test_data['user'])
    recall_avg = recall_sum / len(test_data['user'])
    f1_score_avg = f1_score_sum / len(test_data['user'])
    return precision_avg, recall_avg, f1_score_avg
def diversity_metrics(test_data, course_genre_data):
    unique_genres_per_user = defaultdict(set)
    total_unique_genres = set()
    for user, item, rating in zip(test_data['user'], test_data['item'], test_data['rating']):
        recommended_genres = [key for key, val in course_genre_data[item].items() if val == 1]
        unique_genres_per_user[user].update(recommended_genres)
        total_unique_genres.update(recommended_genres)
    intra_list_diversity = {user: len(genres) / len(test_data['item']) for user, genres in unique_genres_per_user.items()}
    inter_list_diversity = len(total_unique_genres) / len(test_data['item'])
    return intra_list_diversity, inter_list_diversity
def novelty(test_data, user_profile_data, course_genre_data):
    novelty_scores = []
    for user, item, rating in zip(test_data['user'], test_data['item'], test_data['rating']):
        if user in user_profile_data:
            relevant_courses = [key for key, val in user_profile_data[user].items() if val == 1]
            recommended_genres = [key for key, val in course_genre_data[item].items() if val == 1]
            novel_courses = [course for course in recommended_genres if course not in relevant_courses]
            novelty_score = len(novel_courses) / len(recommended_genres) if len(recommended_genres) > 0 else 0
            novelty_scores.append(novelty_score)
    return sum(novelty_scores) / len(test_data['user'])
def coverage(test_data, course_genre_data):
    unique_recommendations = set(test_data['item'])
    total_unique_courses = set(course_genre_data.keys())
    coverage_score = len(unique_recommendations) / len(total_unique_courses) if len(total_unique_courses) > 0 else 0
    return coverage_score
def click_through_rate(test_data):
    num_clicks = sum(test_data['rating'])
    ctr = num_clicks / len(test_data['user'])
    return ctr
# Compute evaluation metrics
precision, recall, f1_score = precision_recall_f1(test_data, user_profile_data, course_genre_data)
intra_list_diversity, inter_list_diversity = diversity_metrics(test_data, course_genre_data)
novelty_score = novelty(test_data, user_profile_data, course_genre_data)
coverage_score = coverage(test_data, course_genre_data)
ctr = click_through_rate(test_data)
# Print results
```

```
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1_score)
    print("Intra-list Diversity:", intra_list_diversity)
    print("Inter-list Diversity:", inter_list_diversity)
    print("Novelty Score:", novelty_score)
    print("Coverage Score:", coverage_score)
    print("Click-through Rate:", ctr)
```

Let's go through the code step by step:

1. Data Structures

- user_profile_data: Dictionary containing user profiles where keys are user IDs and values are dictionaries representing user profiles.
- course_genre_data: Dictionary containing course genres where keys are course IDs and values are dictionaries representing course genres.
- test_data: Dictionary containing test data with keys 'user', 'item', and 'rating', where each key corresponds to a list of user IDs, course IDs, and ratings respectively.

2. Precision, Recall, and F1 Score Calculation (precision_recall_f1 function)

- Iterate through the test data.
- For each user-item pair:
- Extract relevant courses from the user profile data.
- Extract recommended genres from the course genre data.
- Calculate precision, recall, and F1 score based on the overlap between relevant courses and recommended genres.
- Compute averages of precision, recall, and F1 scores across all user-item pairs.

3. Diversity Metrics Calculation (diversity_metrics function)

- Iterate through the test data.
- For each user:
- Collect unique recommended genres for that user.
- Calculate intra-list diversity as the ratio of unique recommended genres to the total number of recommended courses for each user.
- Calculate inter-list diversity as the ratio of unique recommended genres to the total number of unique courses.

4. Novelty Calculation (novelty function)

- Iterate through the test data.
- For each user-item pair:
- Extract relevant courses from the user profile data.
- Extract recommended genres from the course genre data.
- Calculate novelty score as the ratio of recommended genres that are not in the user's relevant courses to the total number of recommended genres.

5. Coverage Calculation (coverage function)

Calculate coverage as the ratio of unique recommended courses to the total number of unique courses in the catalog.

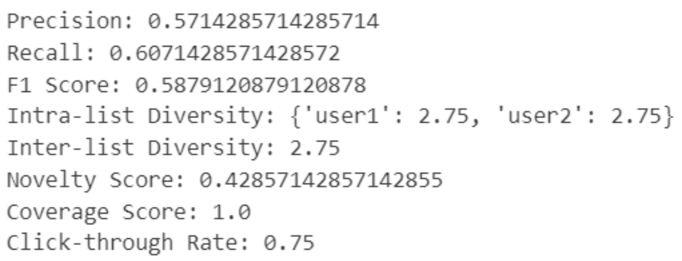
6. Click-through Rate Calculation (click_through_rate function)

Calculate click-through rate as the ratio of the total number of users who clicked on recommended courses to the total number of users.

7. Execution:

- Call each function to compute the corresponding evaluation metric.
- Print the results.

Output



```
Precision: 0.5714285714285714
Recall: 0.6071428571428572
F1 Score: 0.5879120879120878
Intra-list Diversity: {'user1': 2.75, 'user2': 2.75}
Inter-list Diversity: 2.75
Novelty Score: 0.42857142857142855
Coverage Score: 1.0
Click-through Rate: 0.75
```

Conclusion

In this reading, we delved into various evaluation metrics essential for assessing the performance of unsupervised learning recommender systems. It's essential to choose appropriate evaluation metrics based on the specific goals and characteristics of the recommender system and interpret the results carefully to make informed decisions for further improvement. These metrics collectively enable comprehensive evaluation and optimization of unsupervised learning recommender systems to enhance user experience and system efficiency.

Author(s)

Akansha Yadav