Minia University

Faculty of Science
Computer Science Department

Documentation: FOA Recommendation System(AI)

- Girgis Samy Shafeq
- Khaled Mohamed Fathy
- Moaz Ebrahem Mohamed
- David Nady Aiad
- Mostafa Naser Mohame
- Ahmed Medhat
- Kareem Nagy
- Ahmed Khaled

Supervisor: Prof/ Osman Ali

Project Overview

Title:

Recommendation System Evaluation Using Firefly Optimization Algorithm (FOA).

Purpose:

The purpose of this project is to evaluate the performance of a recommendation system. The system uses the Pearson Correlation Coefficient for user similarity and employs the KNN Basic algorithm for predicting movie ratings. The project measures the system's effectiveness using metrics like Precision, Recall, F1-Score, and Accuracy. Additionally, it demonstrates how Firefly Optimization Algorithm (FOA) can enhance recommendation quality by optimizing parameters.

Key Features

- Evaluate the system's top-k recommendations.
- Compute and display multiple evaluation metrics.
- Allow customization of the threshold and k-value for evaluation.
- Integrate FOA to fine-tune recommendation performance.

Prerequisites

Libraries Used:

The following Python libraries are required:

- **pandas**: For data manipulation.
- **numpy**: For numerical operations.
- **seaborn** and **matplotlib**: For data visualization.

- **surprise**: For recommendation algorithms.
- **sklearn**: For evaluation metrics.
- **csv**: For writing results to files.

Input Data Format:

The system uses two datasets:

1. **Movies Data (`movie.csv`)**:
    - Contains movie IDs and titles.
2. **Ratings Data (`rating.csv`)**:
    - Contains user IDs, movie IDs, and ratings.

The datasets are structured to support collaborative filtering models and evaluation.

Setup Instructions:

1. Ensure you have Python installed with all required libraries.
2. Place the datasets (`movie.csv` and `rating.csv`) in the working directory.
3. Install required libraries using pip:

```
pip install pandas numpy matplotlib seaborn surprise scikit-learn
```

4. Run the notebook using Jupyter Notebook or any Python IDE.

Code Explanation

Main Functionality:

Function: `precision_recall_f1_at_k`

This function computes evaluation metrics for the recommendation system.

Inputs:

- **`predictions`**: A list of predicted ratings.
- **`k`**: The number of top recommendations to consider.
- **`threshold`**: The rating threshold to classify as relevant (e.g., 3.5).

Outputs:

- **Precision**: The ratio of relevant recommendations out of all recommendations made.
- **Recall**: The ratio of relevant recommendations retrieved out of all possible relevant items.
- **F1-Score**: The harmonic mean of Precision and Recall.
- **Accuracy**: The overall correctness of recommendations.

Algorithm Steps:

1. Group predictions by user.
2. Sort recommendations for each user based on predicted rating.
3. Select the top-k recommendations.
4. Compute metrics for each user and aggregate them globally.
5. Optionally, use FOA to adjust parameters and optimize performance.

Code Implementation

Data Preparation:

1. Load data using `pandas`:

```
df_ratings = pd.read_csv("datasets/rating.csv")
df_movies = pd.read_csv("datasets/movie.csv")
```

2. Merge datasets and select relevant columns:

```
df_movies_ratings = pd.merge(df_ratings, df_movies, on="movieId")
df_movies_ratings = df_movies_ratings[['movieId', 'title', 'userId',
'rating']]
```

Model Training:

1. Prepare data for training using `surprise`:

```
reader = Reader(rating_scale=(0.5, 5))
data = Dataset.load_from_df(df_movies_ratings[['userId', 'movieId',
'rating']], reader)
trainset, testset = train_test_split(data, test_size=0.2)
```

2. Train the model using KNN Basic:

```
algo = KNNBasic()
algo.fit(trainset)
predictions = algo.test(testset)
```

Evaluation:

1. Compute RMSE to measure prediction error:

```
accuracy.rmse(predictions)
```

2. Compute Precision, Recall, and F1-Score using:

```
results = precision_recall_f1_at_k(predictions, k=5)
print(results)
```

3. Analyze metrics for different values of k:

```
precision_values = []
recall_values = []
for k in range(1, 21):
    results = precision_recall_f1_at_k(predictions, k=k)
    precision_values.append(results['precision'])
    recall_values.append(results['recall'])
```

Visualization:

1. Plot Precision and Recall values:

```
plt.figure(figsize=(10, 5))
plt.plot(range(1, 21), precision_values, label='Precision',
marker='o')
plt.plot(range(1, 21), recall_values, label='Recall', marker='x')
plt.xlabel('k')
plt.ylabel('Value')
plt.title('Precision and Recall for different k values')
plt.legend()
plt.show()
```

Save Results:

Write metrics to a CSV file:

```
ith open('results.csv', 'w', newline='') as csvfile:
    fieldnames = ['k', 'precision', 'recall', 'f1_score']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    writer.writeheader()
    for k in range(1, 21):
        results = precision_recall_f1_at_k(predictions, k=k)
        writer.writerow({'k': k, 'precision': results['precision'],
'recall': results['recall'], 'f1_score': results['f1']})
```

Results and Interpretation

Example Output:

- Precision: 0.75
- Recall: 0.68
- F1-Score: 0.71
- Accuracy: 0.80

Interpretation:

1. **High Precision**: The system recommends relevant movies most of the time.
2. **High Recall**: The system captures a large portion of relevant movies.
3. **Balanced F1-Score**: Indicates a good balance between Precision and Recall.
4. **High Accuracy**: Overall correctness of predictions.

Enhancements with FOA

The Firefly Optimization Algorithm (FOA) is used to improve recommendation performance by optimizing parameters such as the number of neighbors and similarity thresholds. FOA mimics the behavior of fireflies and their attraction mechanism to find optimal solutions.

FOA Implementation Steps:

1. Initialize a population of fireflies with random parameter settings.
2. Evaluate each firefly's fitness using evaluation metrics (e.g., F1-Score).
3. Adjust firefly positions based on their brightness (fitness).
4. Repeat until convergence or maximum iterations.
5. Use the best parameter set for model evaluation.

FOA Benefits:

- Dynamically optimizes hyperparameters.
- Enhances the system’s ability to adapt to different datasets.

Code:

```
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
```

Explanation:

1. We split the data into 5 parts (5-fold Cross-Validation):
2. Each time, we take one part for testing and the rest for training.

We calculate metrics such as:

- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error

References

- **Surprise L**ibrary**: https://surprise.readthedocs.io
- **MovieLens D**ataset**: https://grouplens.org/datasets/movielens
- Research papers on recommendation systems, evaluation metrics, and the Firefly Optimization Algorithm.