Recommender Systems

Prof. Gheith Abandah

Reference: *Artificial Intelligence with Python*, by Prateek Joshi, Packt Publishing, 2017.

Introduction

- YouTube Video: *Recommendation Systems - Learn Python for Data Science #3* by Siraj Raval

https://youtu.be/9gBC9R-msAk

Outline

1. Introduction
2. The MovieLens dataset
3. Similarity scores
4. Building a collaborative recommendation system
5. Open source Python packages
6. Summary

1. Introduction

- A **Recommender System** predicts the likelihood that a user would prefer an item and it recommends items to the user.
- **Examples**:
  - Facebook — “People You May Know”
  - Netflix — “Other Movies You May Enjoy”
  - LinkedIn — “Jobs You May Be Interested In”
  - Amazon — “Customer who bought this item also bought …”
  - Google — “Visually Similar Images”
  - YouTube — “Recommended Videos”

1. Introduction

- **Recommender System Types**:
  1. A **collaborative filtering** algorithm works by finding a set of people with preferences or tastes similar to the target user. Using this smaller set of “similar” people, it constructs a ranked list of suggestions.
  2. **Content-based filtering** is based on a description of the item and a profile of the user’s preferences to recommend items that are similar to those that a user liked.
  3. **Hybrid**

2. The MovieLens DataSet

- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Includes users data and ratings data

Users

(943, 5)

	user_id	age	sex	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

Ratings

(100000, 4)

	user_id	movie_id	rating	unix_timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

3. Similarity Scores

1. **Euclidean score** (Euclidean distance, lower is better)

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2. **Pearson score** (1 is best)

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}$$

4. Building a Collaborative Recommendation System

- Function to recommend movies for a user
- For each other user:
  - Find the Pearson score of commonly rated movies, ignoring dissimilar users.
  - Extract a list of movies that have been rated by this user but haven't been rated by the input user.
  - For each item in this list, keep a track of the weighted rating based on the similarity score.
- Finally, sort the scores and extract the movie recommendations.

4. Building a Collaborative Recommendation System

```python
# Get movie recommendations for the input user
# Assume the input user is in the dataset and there is at lease one
#   recommendation
def get_recommendations(dataset, input_user):
    overall_scores = {}
    similarity_scores = {}

    for user in [x for x in dataset if x != input_user]:
        similarity_score = pearson_score(dataset, input_user, user)

        if similarity_score <= 0:
            continue

        filtered_list = [x for x in dataset[user] if x not in \
                         dataset[input_user] or dataset[input_user][x] == 0]

        for item in filtered_list:
            overall_scores.update({item: dataset[user][item] * similarity_score})
```

4. Building a Collaborative Recommendation System

```
# Generate movie ranks
movie_scores = np.array([[score, item]
                         for item, score in overall_scores.items()])

# Sort in decreasing order
movie_scores = movie_scores[np.argsort(movie_scores[:, 0])[::-1]]

# Extract the movie recommendations
movie_recommendations = [movie for _, movie in movie_scores]

return movie_recommendations
```

5. Open Source Python Packages

- [LightFM]
- [GraphLab]
- [Crab]
- [Surprise]
- [Python Recsys]
- [MRec]

Summary

1. Introduction
2. The MovieLens dataset
3. Similarity scores
4. Building a collaborative recommendation system
5. Open source Python packages
6. Summary