Long-term Project

2017029916
양동해

1. 알고리즘 요약

training dataset(base 파일)을 통해 user별 item에 대한 rating matrix를 만들고, 이 matrix의 correlation coefficient(피어슨 상관 계수)를 구한다. 그런 뒤, test dataset(test 파일)에 있는 user들의 rating을 추정하여, 해당 user가 그 item에 어떤 rating을 줄지 예측하는 프로그램이다. 코드는 총 3가지 파트로 구분되며 각 파트당 알고리즘은 다음과 같다.

- CollaborativeFiltering 클래스
  - 생성자 (model creation)
  - predict() (model usage)
- 파일 읽기 및 파일 쓰기
- rating matrix 만들기, 모델 생성 및 실행

2. 주요 코드 상세

코드를 위에서 언급한 3가지 파트로 나누어, 각 파트가 무엇을 하는지 기술하였다.

[CollaborativeFiltering 클래스]

- 생성자 (model creation)

```
class CollaborativeFiltering():
  def __init__(self, rating_matrix, threshold):
    self.rating_matrix = rating_matrix
    self.threshold = threshold
    self.num_of_users, self.num_of_items = self.rating_matrix.shape

    # user_id와 매트릭스의 인덱스, item_id와 매트릭스의 인덱스 매핑
    self.user_id_dict = {}
    for i, user_id in enumerate(rating_matrix.index): self.user_id_dict[user_id] = i
    self.item_id_dict = {}
    for i, item_id in enumerate(rating_matrix.columns): self.item_id_dict[item_id] = i

    print("Creating model...")
    self.user_rating_mean = np.zeros([self.num_of_users]) # 유저별 rating 평균
    for i in range(self.num_of_users): self.user_rating_mean[i] = np.mean(self.filtered_user(rating_matrix.iloc[i, :].values))

    self.correlation_table = np.zeros([self.num_of_users, self.num_of_users]) # 피어슨 유사도
    ''' ver 1 '''
    # np_rating_matrix = rating_matrix.values
    # for i in range(self.num_of_users):
    #   for j in range(self.num_of_users):
    #     self.correlation_table[i][j] = self.pearson_sim(np_rating_matrix[i], np_rating_matrix[j])
    #   if (i % 10) == 0: print(i,'/',self.num_of_users)
    ''' ver 2 '''
    self.correlation_table = np.corrcoef(rating_matrix) # 피어슨 유사도
    print("Model creation completed!")
```

CollaborativeFiltering 클래스는 model을 creation하고 prediction을 하는 클래스이다. rating_matrix와 num_of_users, num_of_items는 변수명과 동일한 정보를 저장하는 변수이고, threshold는 prediction을 할 때, 해당 user와 성향이 비슷한 neighbor를 선정할 때 사용하는 변수이다. user_id_dict와 item_id_dict는 rating matrix의 index와 해당 id를 매핑해준 dictionary 변수이다. user_rating_mean은 모든 유저의 유저별 rating의 평균을 저장하는 변수이고, correlation_table은 rating matrix에 대한 피어슨 상관 계수를 저장하는 변수이다. user_rating_mean은 numpy의 mean() 함수를, correlation_table은 numpy의 corrcoef() 함수를 사용했다.

correlation_table 변수를 초기화하는 과정을 보면 ver1과 ver2가 존재하는데, ver1은 직접 피어슨 유사도를 계산하는 버전이고, ver2는 라이브러리를 이용한 버전이다. 직접 계산한 버전도 같은 결과를 도출하지만 학습시간이 오래 걸려 ver2를 사용했다.

- predict() (model usage)

```
def predict(self, test_dataset):
  result = []

  print("Predicting...")
  for sample in test_dataset.values.tolist():
    user_id = sample[0]
    user_idx = self.user_id_dict[user_id]
    predicted_rating = self.user_rating_mean[user_idx]

    item_id = sample[1]
    if item_id in self.rating_matrix.columns:
      neighbor_rating = self.rating_matrix[item_id].values

      mask1 = neighbor_rating > 0
      mask2 = self.correlation_table[user_idx] > self.threshold
      mask = mask1 * mask2
      # print(neighbor_rating[mask])
      # print(self.correlation_table[user_idx, mask])

      neighbor_rating = neighbor_rating[mask]
      neighbor_rating_mean = self.user_rating_mean[mask]
      neighbor_similarity = self.correlation_table[user_idx, mask]

      predicted_rating = neighbor_similarity * (neighbor_rating - neighbor_rating_mean)
      predicted_rating = self.user_rating_mean[user_idx] + (predicted_rating.sum() / (neighbor_similarity.sum() + 1e-9))

    if predicted_rating < 1: predicted_rating = 1
    elif predicted_rating > 5: predicted_rating = 5
    # else: predicted_rating = round(predicted_rating)
    result.append(predicted_rating)
  print("Prediction completed!")

  test_dataset = test_dataset.drop('time_stamp', axis=1)
  test_dataset['rating'] = result
  return test_dataset
```

predict() 함수는 만들어진 모델을 사용하는 함수이다. 주어진 test dataset에서 각 행마다 user와 item을 뽑아내고, 그 user가 해당 item에게 줄 rating을 예측한다. 예측하는 방법은 그 user와 성향이 비슷한 neighbor들을 뽑아내고, 그 neighbor들의 rating을 weighted average로 구해 자신의 rating 평균과 더하여 예측한다. neighbor는 correlation_table에서 threshold 이상의 값을 갖는 user들만 선택되며, test dataset으로 들어온 행 중 item이 존재하지 않는 경우는 그 user의 rating 평균 값으로 예측한다. 이후 나머지 과정은 공식을 그래도 코드로 구현 했기 때문에 자세한 설명은 생략한다. 마지막으로, 예측된 rating이 1보다 작다면 1로, 5보다 크다면 5로 세팅을 해주고 해당 rating들로 덮어 씌어진 test dataset을 다시 return한다.

[파일 읽기 및 파일 쓰기]

```
# 파일 읽기
train_filename = sys.argv[1]
test_filename = sys.argv[2]
output_filename = train_filename + "_prediction.txt"

dataset_header = ['user_id', 'item_id', 'rating', 'time_stamp']
train_dataset = pd.read_csv(train_filename, sep="\t", names=dataset_header)
test_dataset = pd.read_csv(test_filename, sep="\t", names=dataset_header)
```

```
# 파일 쓰기
result.to_csv(output_filename, header=False, index=False, sep="\t")
```

이 파트는 인자로 입력 받은 파일들을 읽고, 결과물을 쓰는 단계이다.

[rating matrix 만들기, 모델 생성 및 실행]

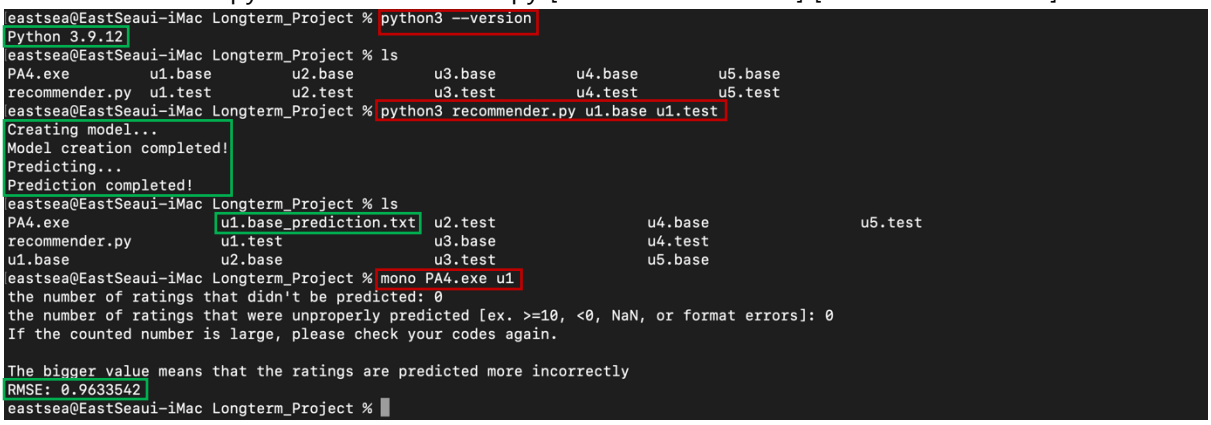
```
# rating matrix 만들기
tmp_train_dataset = train_dataset.drop('time_stamp', axis=1)
rating_matrix = tmp_train_dataset.groupby(['user_id', 'item_id'])['rating'].mean().unstack().fillna(0)

# Recommender System
cf = CollaborativeFiltering(rating_matrix, 0.03)
result = cf.predict(test_dataset)
```

이 파트는 rating matrix를 만들고, 모델을 생성 및 실행 하는 단계이다.

3. 컴파일 및 실행

- Python version: 3.9.12 + (numpy (1.22.3), pandas (1.4.2) library 사용)
- 프로그램 실행: python3 recommender.py [train data file name] [test data file name]



- 각 파일별 RMSE

u1	u2	u3	u4	u5
0.9633542	0.9541191	0.9479203	0.9447452	0.944097