Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Posgrados

6.2 Avance de proyecto 2: Sistema de Recomendación

Abril Yusely Cota Jaquez A01795114
Gabriel Paredes Garza A00797698
Carlos Daniel Villena Santiago A01795127

26 de mayo del 2024

Experimentación con un Algoritmo de Factorizacion Matricial

Descripción del Algoritmo

La factorización matricial es una técnica avanzada que se utiliza en sistemas de recomendación, especialmente útil para el filtrado colaborativo. Este método descompone una matriz grande de interacciones usuario-item en dos matrices de menor dimensión. Estas matrices latentes capturan las características ocultas de los usuarios y los items, lo que permite predecir las interacciones faltantes con alta precisión.

Proceso de Implementación

1. **Carga de Datos:**
   Se cargan los datos de los usuarios, items y revisiones desde archivos JSON y se convierten en DataFrames de pandas.
2. **Preprocesamiento:**
   Los datos de revisión se formatean adecuadamente para ser utilizados por la librería **`surprise`**.
3. **División de Datos:**
   Se dividen los datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo.
4. **Entrenamiento del Modelo:**
   Se utiliza el algoritmo SVD (Singular Value Decomposition) para entrenar el modelo con los datos de entrenamiento.
5. **Generación de Recomendaciones:**
   Se define una función **`recommend()`** que toma un ID de usuario y genera recomendaciones basadas en las predicciones del modelo entrenado.

Experimentación Justificada

La elección de la factorización matricial mediante el algoritmo SVD está justificada por su capacidad para manejar grandes volúmenes de datos y su eficiencia en la predicción de interacciones faltantes en sistemas de recomendación. Este enfoque es ampliamente reconocido por su precisión y capacidad para capturar relaciones latentes entre usuarios e items, lo que lo convierte en una elección óptima para sistemas de recomendación colaborativos.

Resultados Esperados

Los resultados esperados incluyen:

- Recomendaciones precisas y personalizadas para los usuarios, basadas en sus interacciones anteriores.
- Mejora en las métricas de evaluación como RMSE y MAE, indicando la precisión del modelo.
- Una lista de items recomendados que no han sido previamente vistos por el usuario, lo que aumenta la diversidad y la satisfacción del usuario con el sistema de recomendación.

Esta implementación y justificación proporcionan una base sólida para un sistema de recomendación eficiente y efectivo utilizando factorización matricial.

E.G:

Recomendaciones para el usuario 76561198091973414: ['Sword of Asumi', 'Sword of Asumi - Soundtrack', 'Sword of Asumi - Graphic Novel']

Codigo.

```
import pandas as pd
from surprise import Dataset, Reader, SVD, accuracy
from surprise.model_selection import train_test_split

# Cargar datos desde archivos JSON
with open('user_item_data.json', 'r') as f:
    user_item_data = json.load(f)

with open('review_data.json', 'r') as f:
    review_data = json.load(f)

with open('bundle_data.json', 'r') as f:
    bundle_data = json.load(f)

# Crear DataFrames
user_item_df = pd.DataFrame(user_item_data['items'])
review_df = pd.DataFrame([review_data])
bundle_df = pd.DataFrame(bundle_data['items'])

# Preprocesamiento
```

```
# Convertir los datos de revisión en un DataFrame adecuado para Surprise
review_df = review_df.rename(columns={'username': 'userID', 'product_id':
'itemID', 'text': 'rating'})
review_df['rating'] = 1  # Añadir una columna de calificaciones ficticias para
que el formato sea adecuado

reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(review_df[['userID', 'itemID', 'rating']], reader)

# Entrenar el modelo con todos los datos disponibles
trainset = data.build_full_trainset()
algo = SVD()
algo.fit(trainset)

# Función de recomendación
def recommend(userID, user_item_df, review_df, bundle_df, algo, n=10):
    # Obtener los items que el usuario ya ha visto
    user_items =
user_item_df[user_item_df['item_id'].isin(review_df['itemID'])]['item_id'].tolist
()
    user_item_names =
user_item_df[user_item_df['item_id'].isin(review_df['itemID'])]['item_name'].toli
st()

    # Generar predicciones para los items no vistos
    all_items = bundle_df['item_id'].unique()
    recommendations = []

    for itemID in all_items:
        if itemID not in user_items:
            pred = algo.predict(userID, itemID)
            recommendations.append((itemID, pred.est))

    # Ordenar las recomendaciones por calificación predicha
    recommendations.sort(key=lambda x: x[1], reverse=True)

    # Seleccionar los top n items
    top_recommendations = recommendations[:n]
    recommended_items = [item[0] for item in top_recommendations]

    # Obtener nombres de los items recomendados
    recommended_item_names =
bundle_df[bundle_df['item_id'].isin(recommended_items)]['item_name'].tolist()
```

```
    return recommended_item_names

# Ejemplo de uso
userID = '76561198091973414'  # ID de usuario del ejemplo
recommendations = recommend(userID, user_item_df, review_df, bundle_df, algo,
n=3)
print(f"Recomendaciones para el usuario {userID}: {recommendations}")
```