Recommender System & Deep Reinforcement

Zeham Management Technologies BootCamp
by SDAIA

October 2024

SDAIA
الهيئة السعودية للبيانات
والذكاء الاصطناعي
Saudi Data & AI Authority

Agenda

- Introduction to Recommender Systems
- Collaborative Filtering
- Content-based Filtering
- Introduction to Deep Reinforcement Learning
- Practical Use Cases of APIs

Introduction to Recommender Systems

Introduction to Recommender Systems

Recommender systems are algorithms that suggest items (movies, products, services, etc.) to users based on their preferences or past behavior.

Importance:

Used in personalized services like Netflix, Amazon, Spotify, and YouTube to enhance user experience.

Increases user engagement, retention, and ultimately drives revenue.

Introduction to Recommender Systems

Types of Recommendation Approaches:

1. Collaborative Filtering: Uses user behavior (ratings, interactions) to make recommendations.
2. Content-Based Filtering: Recommends items based on the features or content of the items (e.g., genres of movies, keywords in articles).
3. Hybrid Systems: Combines both collaborative and content-based filtering to improve recommendations.

Collaborative Filtering

Collaborative Filtering

Makes recommendations based on the idea that users who have similar preferences in the past will continue to have similar preferences in the future.

Two Main Types:

User-based filtering: Finds users who are similar to the target user and recommends items those similar users liked.

Item-based filtering: Recommends items that are similar to the ones the user has previously liked or interacted with.

Collaborative Filtering

Steps to Build a Collaborative Filtering System:

1. Load a dataset:

Example: MovieLens dataset, a popular dataset containing user ratings of movies.

Use libraries like surprise, scikit-learn, or pandas to load and preprocess the data.

2. Implement Collaborative Filtering:

User or item-based collaborative filtering can be implemented using algorithms like KNN (K-Nearest Neighbors) or matrix factorization techniques.

3. Evaluate the Model:

Use metrics like Root Mean Square Error (RMSE) to evaluate the accuracy of recommendations.

Split the dataset into training and testing sets using train-test splits to ensure the model is tested on unseen data.

Collaborative Filtering

Collaborative Filtering Code Example:

python Copy code

```python
from surprise import Dataset, KNNBasic
from surprise.model_selection import train_test_split
from surprise import accuracy

# Load data
data = Dataset.load_builtin('ml-100k')
trainset, testset = train_test_split(data, test_size=0.2)

# Implement KNN-based collaborative filtering
algo = KNNBasic()
algo.fit(trainset)

# Test and evaluate
predictions = algo.test(testset)
accuracy.rmse(predictions)
```

Content-based Filtering

Content-based Filtering

Content-based filtering recommends items by analyzing the attributes or features of the items.

In movies, for example, the system looks at features like genre, director, actors, etc.

Deep Learning: Can be used to automatically learn features of items from data, allowing for more accurate recommendations, especially when using text (e.g., movie descriptions or product reviews) as inputs.

Content-based Filtering

Steps to Build a Content-Based Recommender System:

1. Preprocess the Data:

Text data can be processed using techniques like TF-IDF (Term Frequency-Inverse Document Frequency)or more advanced word embeddings like Word2Vec or BERT.

2. Build the Model:

Build a neural network to represent item features and predict which items a user might like based on these features.

Use deep learning frameworks like TensorFlow or PyTorch to implement the model.

Content-based Filtering

Content-Based Recommender System Code Example:

python   Copy code

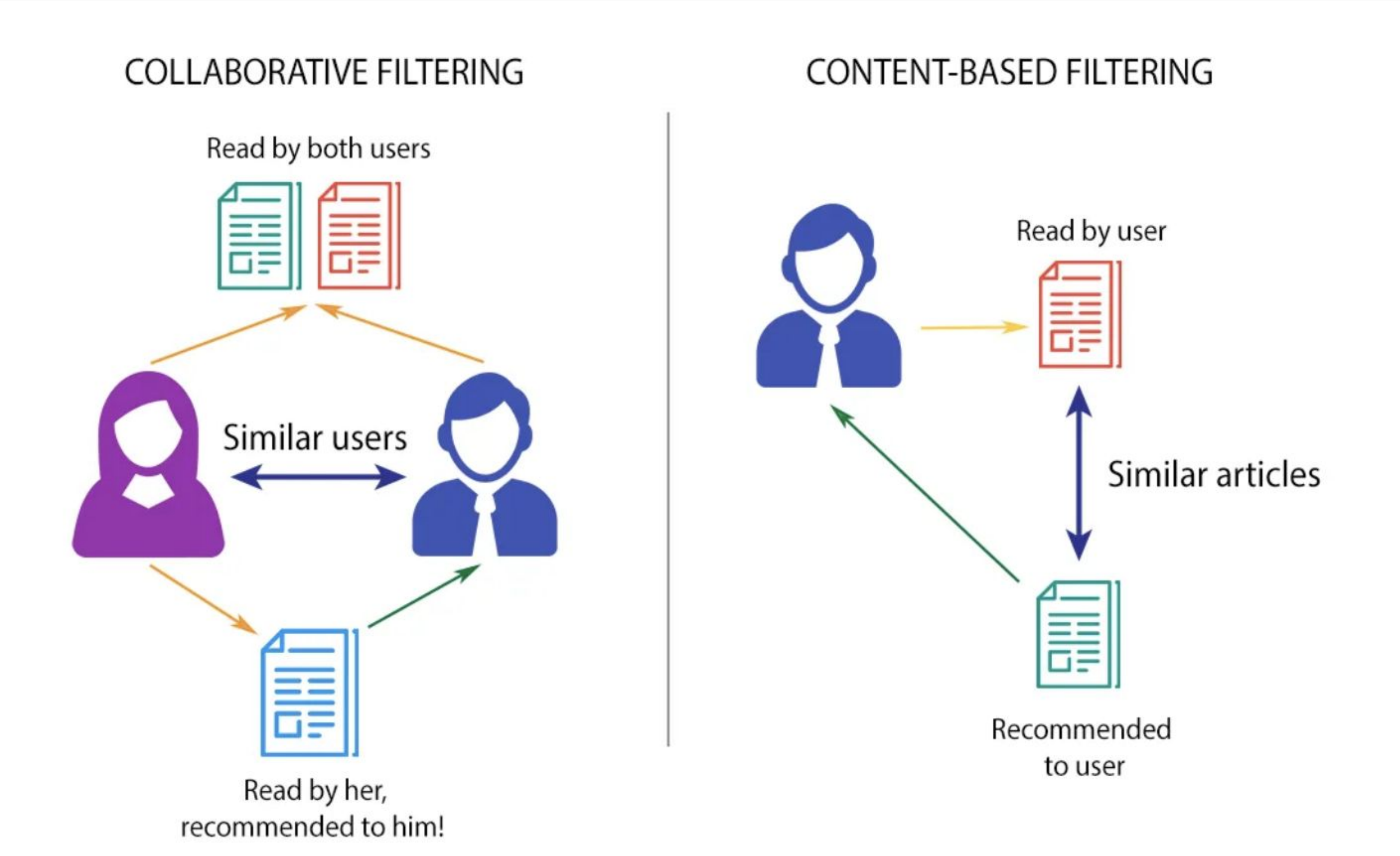
```python
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# Example movie descriptions
movie_descriptions = ["A love story set in Paris", "Action-packed superhero movi

# Convert descriptions into TF-IDF features
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(movie_descriptions)

# Simple neural network model using TensorFlow or PyTorch
# You can then use the learned item features to predict recommendations
```

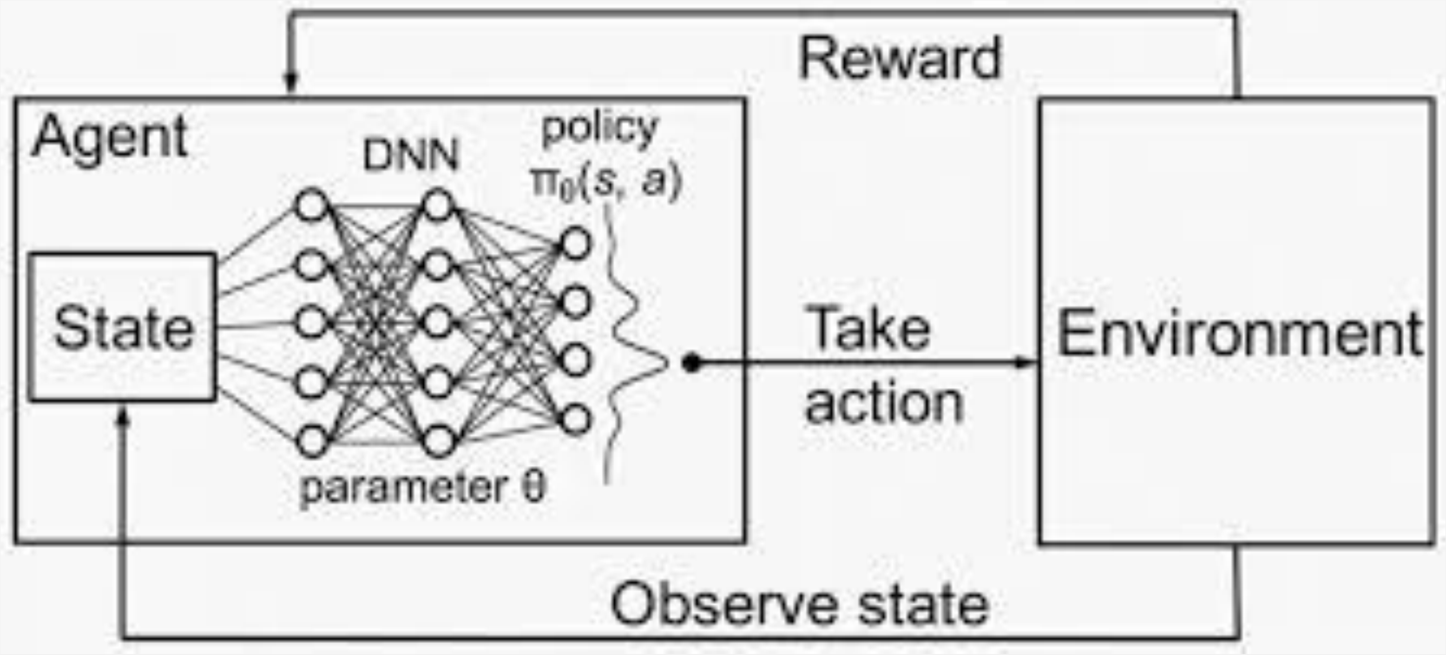
Collaborative vs Content-based Filtering



Introduction to Deep Reinforcement Learning

Introduction to Deep Reinforcement Learning

Deep Q-learning (DQN): Uses deep neural networks to approximate Q-values and improve decision-making.



Introduction DRL

Hands-on Steps:

1. Use OpenAI's Gym Environment: OpenAI Gym provides a set of environments to train RL agents, such as CartPole, where the goal is to balance a pole on a moving cart.

2. Build a Deep Q-Network (DQN): The DQN uses a neural network to estimate Q-values for each action the agent can take in a given state.

3. Train the Agent: Train the agent using rewards and penalties over time to optimize its performance in the environment.

Introduction DRL

Deep Reinforcement Learning Code Example:

python   Copy code

```python
import gym

# Load CartPole environment
env = gym.make('CartPole-v1')
state = env.reset()

# Take random actions for demonstration
for _ in range(1000):
    action = env.action_space.sample()  # Take a random action
    state, reward, done, info = env.step(action)  # Perform action
    if done:
        state = env.reset()

env.close()
```

Integration of Recommender Systems with Reinforcement Learning

- Advanced Concept:

Use reinforcement learning to optimize long-term user engagement by recommending sequences of items (e.g., articles, products, or videos) to maximize cumulative rewards such as clicks or time spent on the platform.

- Scenario:

In an e-commerce or media platform, RL can learn to recommend items in a sequence that maximizes user retention or purchase likelihood.

Example: Instead of recommending individual items in isolation, the system learns to recommend a series of products that lead to a purchase decision.

Conclusion

- **Recommender systems** are crucial for improving user experience and engagement in platforms like Netflix, Amazon, and Spotify. Both collaborative filtering and content-based filtering can be combined to create more powerful hybrid recommendation systems.

- **Deep Reinforcement Learning** is a powerful approach to solving dynamic and complex problems, making it essential for cutting-edge AI applications in personalized recommendations, gaming, and autonomous systems.

Thank you!

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority