BOOK RECOMENDER SYSTEM

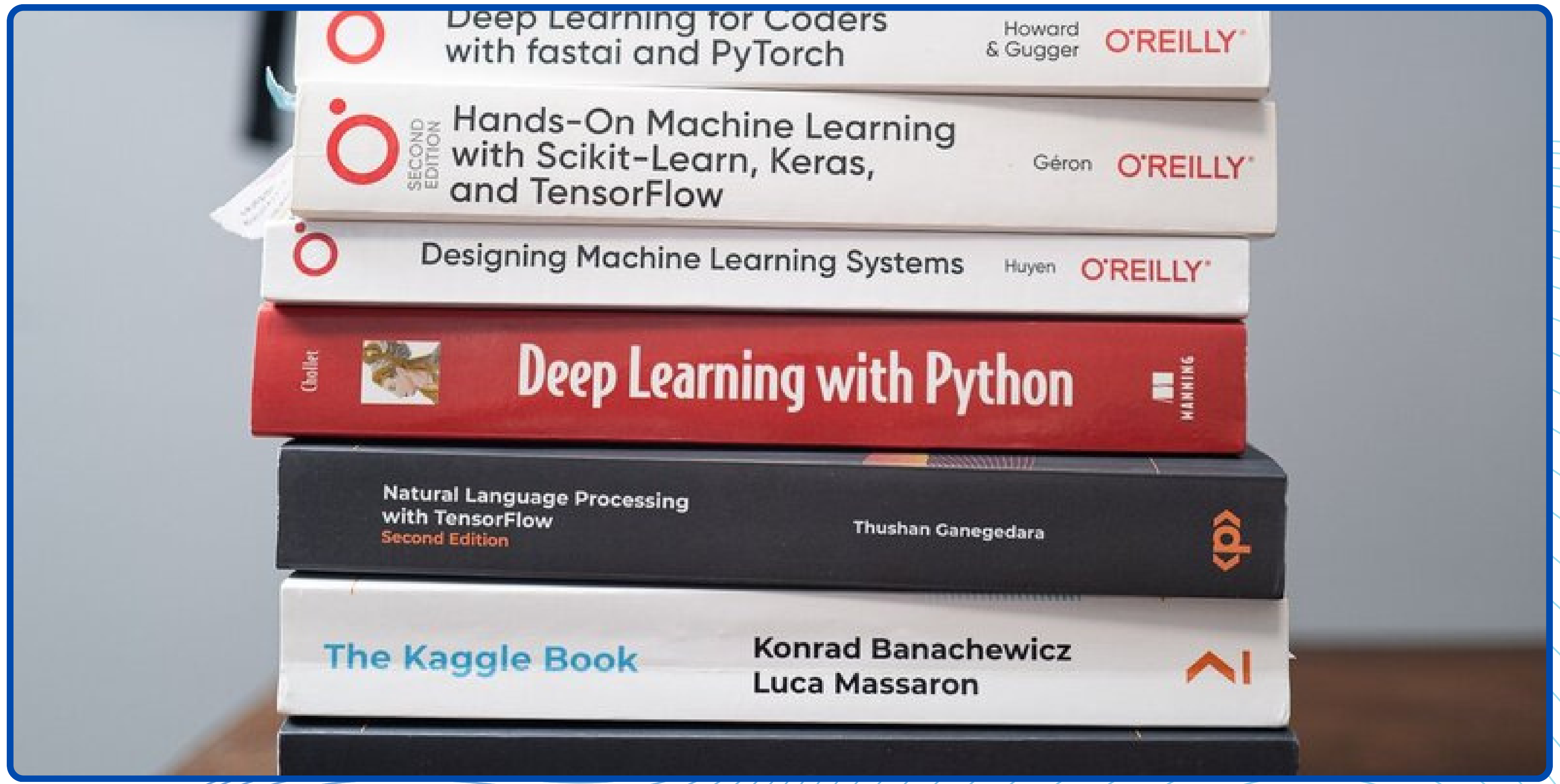
APPLYING KNOWLEDGE-BASED & REINFORCEMENT LEARNING

TABLE OF CONTENTS

DATASET OVERVIEW

Book Recommendation Dataset

- Dataset contains 3 files:
  - User data
  - Book data
  - Rating data
- Preprocessed to remove null values
- Merged into one dataframe for convenience



Deep Learning for Coders with fastai and PyTorch
Howard & Gugger
O'REILLY
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow
SECOND EDITION
Géron
O'REILLY
Designing Machine Learning Systems
Huyen
O'REILLY
Chollet
Deep Learning with Python
MANNING
Natural Language Processing with TensorFlow
Second Edition
Thushan Ganegedara
The Kaggle Book
Konrad Banachewicz
Luca Massaron

MODEL IMPLEMENTATIONS

COLLABORATIVE FILTERING

Overview:

- Two main types: item-based and user-based.
- Utilizes user interactions to generate recommendations.

Dataset Challenges

- Absence of diverse item features (like genre) complicates item-based filtering.
- Relying on highest average ratings as a primary measure for grouping and recommendation.

Approach in Our Use Case:

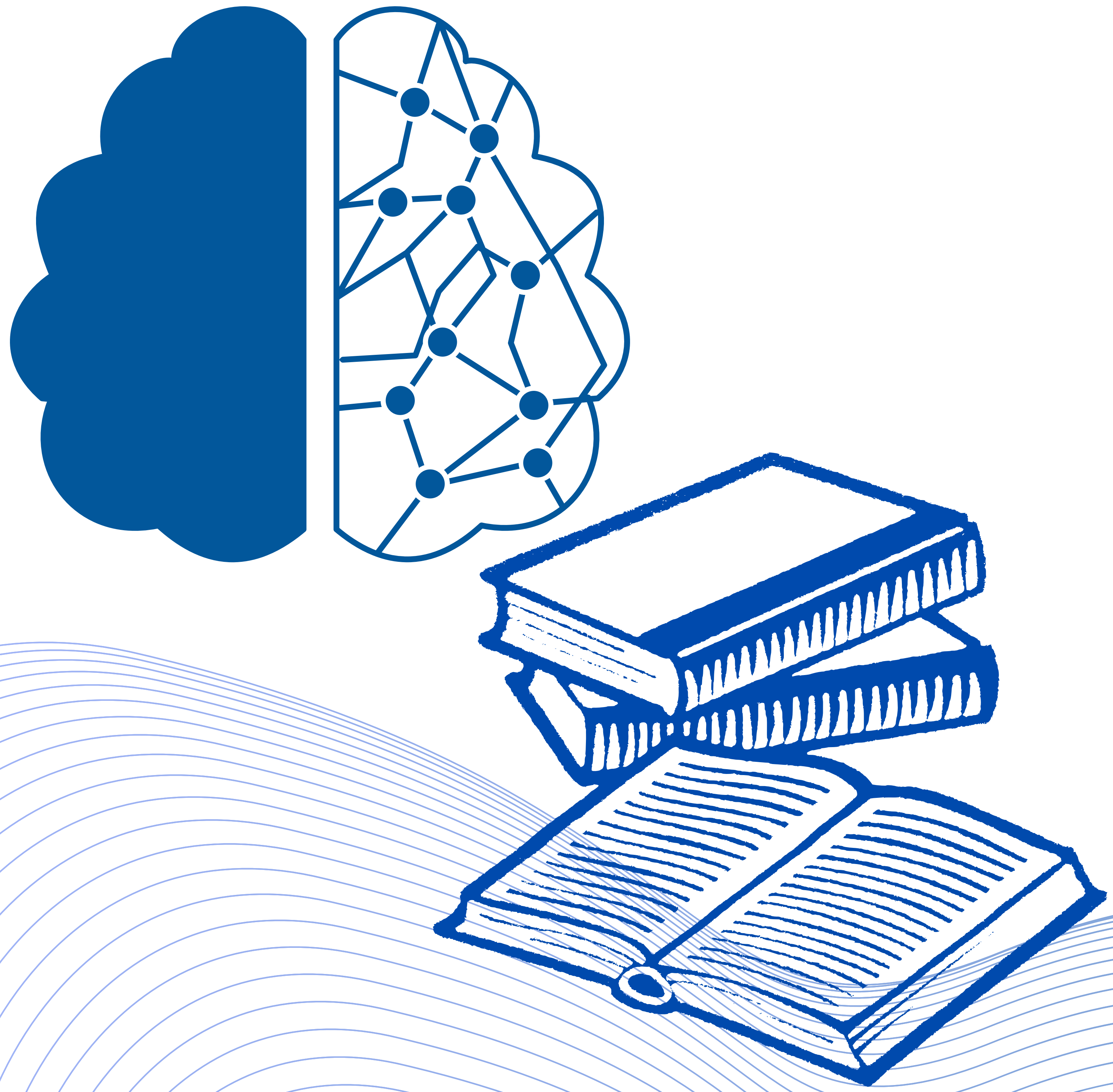
- Initial focus on item-based collaborative filtering due to available data structure.
- Proposes asking users about top-rated content to gather preference data for user-based filtering.

Developer Insights:

- Identified the need for iterative data collection to enhance user-based filtering.
- Noted limitations in item-based filtering due to the lack of varied item attributes for grouping.

KNOWLEDGE BASED LEARNING

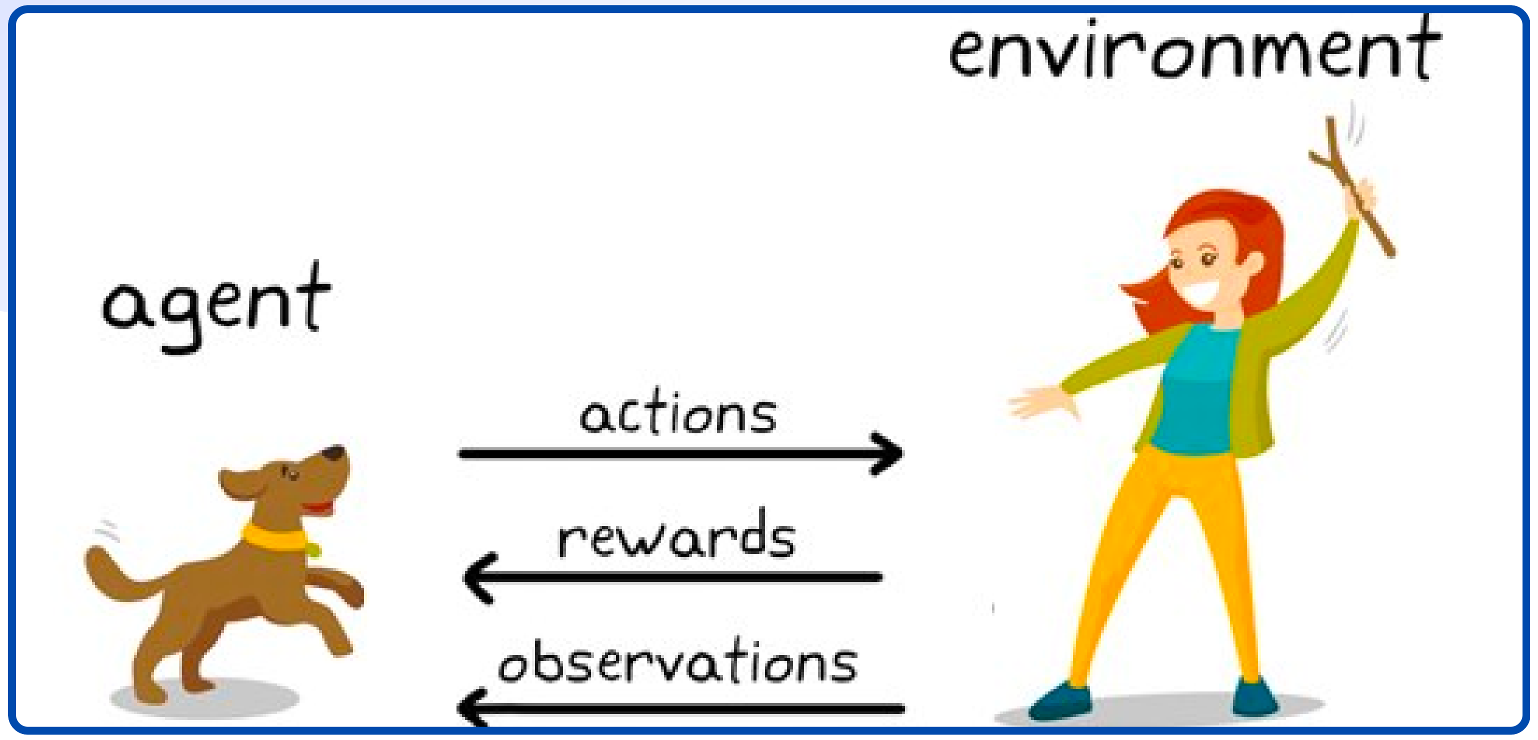
- Uses a coordinate system matrix of the users with their books/ratings of those books.
- Cosine similarity is calculated between these users using sklearn.metrics
- Top five users' books are recommended to the user.

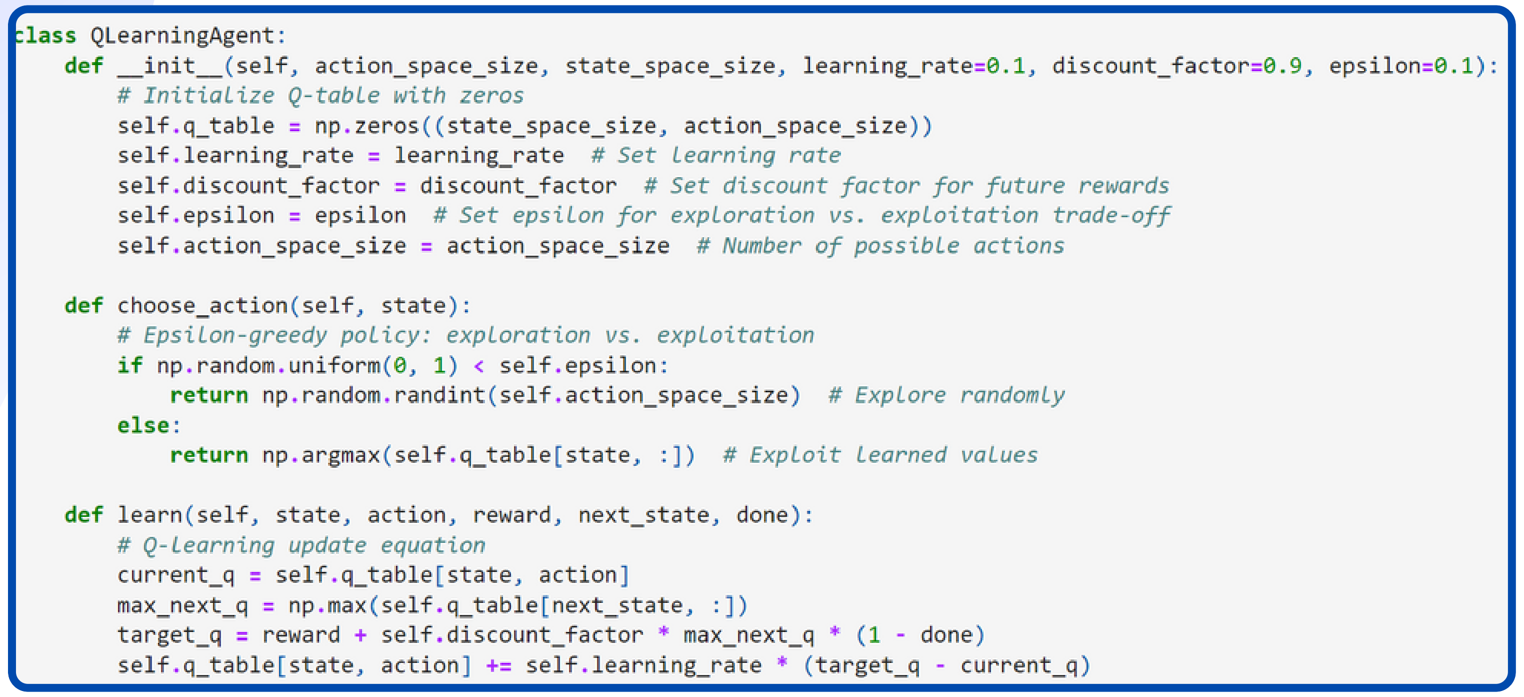


```
book_to_user_rating_matrix = scipy.sparse.coo_matrix((combined_df["Book-Rating"], (combined_df["User-ID"], combined_df["ISBN-Key"])))
```

REINFORCEMENT LEARNING

Reinforcement Learning, a core technique in AI, involves an agent learning from interaction with its environment. Through trial and error, the agent explores actions, receives feedback (rewards), and adjusts its strategy to maximize cumulative rewards





```python
class QLearningAgent:
    def __init__(self, action_space_size, state_space_size, learning_rate=0.1, discount_factor=0.9, epsilon=0.1):
        # Initialize Q-table with zeros
        self.q_table = np.zeros((state_space_size, action_space_size))
        self.learning_rate = learning_rate  # Set learning rate
        self.discount_factor = discount_factor  # Set discount factor for future rewards
        self.epsilon = epsilon  # Set epsilon for exploration vs. exploitation trade-off
        self.action_space_size = action_space_size  # Number of possible actions

    def choose_action(self, state):
        # Epsilon-greedy policy: exploration vs. exploitation
        if np.random.uniform(0, 1) < self.epsilon:
            return np.random.randint(self.action_space_size)  # Explore randomly
        else:
            return np.argmax(self.q_table[state, :])  # Exploit learned values

    def learn(self, state, action, reward, next_state, done):
        # Q-learning update equation
        current_q = self.q_table[state, action]
        max_next_q = np.max(self.q_table[next_state, :])
        target_q = reward + self.discount_factor * max_next_q * (1 - done)
        self.q_table[state, action] += self.learning_rate * (target_q - current_q)
```

Goal

- Recommend Personalized book suggestions through continuous user interaction
- System learning to improve recommendation accuracy over time.

Model Setup

- Utilizes Q-Learning Agent in a Custom Environment.
- Q-table based decision-making for book recommendations.

Process

- Agent learns from user interactions and ratings to make personalized book recommendations.
- Recommends books based on learned preferences and previous user feedback.

HYBRID APPROACH

- **Cold Start Problem**
  - Knowledge Based: Uses existing knowledge to make recommendations, even when user data is limited
  - Reinforcement Learning: Improves recommendations as it gathers more data
  - Collaborative filtering allows for more diversity
- **Handling Sparse Data**
  - Reinforcement takes knowledge based learning and learns from user feedback and interactions, adapting to sparse and delayed rewards thus improving recommendations over time.

OBSTACLES

Data Handling Challenges:

- Faced "ValueError: negative dimensions" due to excessive book listings for users.
- Non-numeric ISBNs obstructed efficient matrix operations.

Model Complexity

- Computational Resources and Parameter Tuning
- Sample/User Efficiency

Hybrid System Refinement

- Integrating Models
- Hybrid Collaborative Filtering

Model Enviroment

- Suboptimal Environment for user interaction
- Complex Environment

QUESTIONS?