Идея методологии

Методология объединяет современные подходы к обучению с подкреплением (RL) для улучшения рекомендательных систем, решая ключевые проблемы:

- Учет многоплатформенного поведения пользователей.
- Устойчивое обучение для работы с офлайн-данными.
- Эффективное использование как положительных, так и отрицательных сигналов обратной связи.
- Усиление адаптивности моделей благодаря использованию трансформеров.

Этапы реализации

1. Многоплатформенное моделирование поведения

Идея: Рекомендательные системы часто работают в экосистемах, где пользователь взаимодействует с несколькими платформами (например, соцсети, музыкальные сервисы, торговые площадки). Неучет этих взаимодействий ведет к фрагментарной картине поведения.

Реализация:

- Формализуем проблему как частный случай федеративного обучения, где каждая платформа выступает как агент в RL-среде.
- Применяем алгоритм, подобный **FedSlate**, чтобы совместить данные о пользователе из нескольких источников. Это позволяет моделям обучаться на общих предпочтениях пользователей, сохраняя при этом конфиденциальность данных【9†source】. (https://arxiv.org/abs/2409.14872)

Решаемые проблемы:

- Учет комплексных предпочтений пользователей.
- Улучшение обобщения моделей в условиях разнородных данных.

2. Устойчивое обучение с помощью Conservative Q-Learning (CQL)

Идея: При обучении RL-моделей на фиксированных наборах данных (offline-RL) часто возникает проблема переоценки действий, не представленных в обучающей выборке.

Реализация:

- Определяем значение действия Q(s,a)Q(s, a) как нижнюю границу ожидаемого вознаграждения, чтобы минимизировать переоценку:
  Q(s,a)=min(Qdataset(s,a),Rmax),Q(s, a) = \min(Q_{dataset}(s, a), R_{max}),
  где QdatasetQ_{dataset} — значение действия, рассчитанное на основе обучающей выборки, а RmaxR_{max} — максимальная наблюдаемая награда.

- Определяем значение действия $Q(s, a)$ как нижнюю границу ожидаемого вознаграждения, чтобы минимизировать переоценку:

$$Q(s, a) = \min(Q_{dataset}(s, a), R_{max}),$$

где $Q_{dataset}$ — значение действия, рассчитанное на основе обучающей выборки, а R_{max} — максимальная наблюдаемая награда.

- Добавляем регуляризацию:

$$L = \sum_{(s,a)\in D} [Q(s, a) - (R + \gamma \max_{a'} Q(s', a'))]^2 + \alpha \cdot \mathbb{E}_{a\sim\pi} Q(s, a),$$

- Добавляем регуляризацию:

  L=∑(s,a)∈D[Q(s,a)−(R+γmaxa′Q(s′,a′))]2+α⋅Ea∼πQ(s,a),L = \sum_{(s, a) \in D} [Q(s, a) - (R + \gamma \max_{a'} Q(s', a'))]^2 + \alpha \cdot

\mathbb{E}_{a \sim \pi} Q(s, a),
где α\alpha контролирует баланс между штрафами за избыточно высокие оценки и стандартным TD-обновлением【10†source】.
(https://arxiv.org/abs/2305.18820)

Решаемые проблемы:

- Устранение переоценки действий.
- Устойчивое обучение в условиях ограниченных данных.

3. Интеграция отрицательных сигналов обратной связи

Идея: Большинство RL-алгоритмов для рекомендательных систем игнорируют отсутствие взаимодействия пользователя с предложенным контентом, что приводит к смещению в сторону позитивных сигналов.

Реализация:

- Добавляем отрицательные действия (например, игнорированные рекомендации) в тренировочный процесс через метод SNQN. Это снижает вероятность выбора таких действий, усиливая сигналы:
- $$A(s,a) = Q(s,a) - \max_{a' \neq a} Q(s,a'),$$

  A(s,a)=Q(s,a)−maxa′≠aQ(s,a′),A(s, a) = Q(s, a) - \max_{a' \neq a} Q(s, a'), где A(s,a)A(s, a) — преимущество положительного действия перед отрицательными【10†source】.(https://arxiv.org/abs/2305.18820)

Решаемые проблемы:

- Устранение смещения модели.
- Улучшение качества рекомендаций.

4. Использование трансформеров

Идея: Трансформеры могут эффективно обрабатывать последовательности взаимодействий пользователя, позволяя захватывать долгосрочные зависимости.

Реализация:

- Интегрируем трансформер в RL-агент для кодирования состояний:

  $$h_t = \text{Transformer}(x_{t-1}, x_{t-2}, \ldots, x_0),$$

  ht=Transformer(xt−1,xt−2,…,x0),h_t = \text{Transformer}(x_{t-1}, x_{t-2}, \ldots, x_0), где hth_t — скрытое состояние пользователя, использующееся RL-алгоритмом для принятия решений.

Решаемые проблемы:

- Повышение адаптивности системы к изменениям в предпочтениях пользователей.
- Эффективная обработка временных зависимостей.

Проблемы, которые решает методология

1. Учет сложных взаимодействий пользователей.
2. Устранение смещения моделей.
3. Устойчивое обучение в условиях офлайн-данных.
4. Интеграция контекста взаимодействий.

Возможные слабые стороны

1. **Сложность реализации**:
   - Совмещение моделей RL с трансформерами требует значительных вычислительных ресурсов.
2. **Данные**:

    - Нужны большие объемы данных для обучения с учетом всех факторов (контекста, отрицательных сигналов, межплатформенных взаимодействий).
3. **Обобщение**:
    - Методы могут быть менее эффективны в условиях новых или неизвестных данных (cold start).

Поддержка предложений статьями

- **FedSlate** демонстрирует, как можно применять федеративное обучение в многоплатформенных экосистемах, решая проблемы разнородности данных【9†source】.(https://arxiv.org/abs/2409.14872)
- **CQL** доказал эффективность в условиях офлайн-RL, снижая переоценку действий【10†source】(https://arxiv.org/abs/2305.18820).
- **SNQN** и интеграция отрицательных сигналов усиливают надежность рекомендаций, устраняя положительное смещение【10†source】(https://arxiv.org/abs/2305.18820).

Core Idea of the Methodology

This methodology integrates contemporary approaches in reinforcement learning (RL) to enhance recommender systems. It focuses on:

1. Accounting for cross-platform user behavior.
2. Ensuring stable learning with offline data.
3. Incorporating both positive and negative feedback signals.
4. Leveraging transformer models for adaptive user modeling.

Step-by-Step Implementation

1. Modeling Cross-Platform Behavior

Concept: Users often interact with multiple platforms (e.g., social media, e-commerce, streaming services). Ignoring cross-platform behavior creates an incomplete understanding of user preferences.

Implementation:

- Treat each platform as a separate RL agent, modeled in a federated learning framework.
- Use algorithms like **FedSlate** to share knowledge across platforms while respecting data privacy. This allows the model to capture broader user preferences without direct data sharing【9†source】.

Problem Solved:

- Incorporates complex, multi-source user behaviors.
- Improves generalization across heterogeneous data.

---

2. Stable Learning with Conservative Q-Learning (CQL)

Concept: Offline RL models often overestimate rewards for actions not present in the training data.

Implementation:

- Define the action value Q(s,a)Q(s, a) conservatively to avoid overestimation:
  Q(s,a)=min(Qdataset(s,a),Rmax),Q(s, a) = \min(Q_{dataset}(s, a), R_{max}),
  where QdatasetQ_{dataset} is the observed value from the dataset, and RmaxR_{max} is the maximum observed reward.

- Add a regularization term to the loss function:
  L=∑(s,a)∈D[Q(s,a)−(R+γmaxa′Q(s′,a′))]2+α·Ea∼πQ(s,a),L = \sum_{(s, a) \in D} [Q(s, a) - (R + \gamma \max_{a'} Q(s', a'))]^2 + \alpha \cdot \mathbb{E}_{a \sim \pi} Q(s, a),
  where α\alpha adjusts the trade-off between penalizing overestimated values and traditional temporal difference (TD) updates【10†source】.

Problem Solved:

- Eliminates overestimation in Q-value predictions.
- Stabilizes learning when using limited offline data.

---

3. Incorporating Negative Feedback

Concept: Many RL-based recommender systems rely only on positive interactions (e.g., clicks, purchases) while ignoring implicit or explicit negative feedback (e.g., skips, dislikes).

Implementation:

- Include negative actions (e.g., ignored recommendations) in the training set.
- Use methods like **Stochastic Negative Q-Network (SNQN)** to balance the positive and negative feedback signals: $A(s,a)=Q(s,a)-\max_{a' \neq a} Q(s,a')$, where $A(s,a)$ captures the advantage of the selected action over others【10†source】.

Problem Solved:

- Reduces model bias towards positive actions.
- Enhances recommendation quality by factoring in user disinterest.

---

4. Leveraging Transformers

Concept: Transformers excel at capturing sequential dependencies, making them ideal for modeling user behavior over time.

Implementation:

- Use a transformer encoder to represent user states: $h_t = \text{Transformer}(x_{t-1}, x_{t-2}, \ldots, x_0)$, where h_t is the hidden state summarizing the user's behavior up to time t. This state is then input to the RL model to make decisions.

Problem Solved:

- Improves adaptability to evolving user preferences.
- Efficiently models long-term dependencies in user interactions.

Problems Addressed by the Methodology

1. Captures complex user interactions across multiple platforms.
2. Reduces bias by incorporating negative feedback.
3. Stabilizes learning processes in offline settings.
4. Adapts to user behavior changes through transformer-based modeling.

Potential Weaknesses

1. **Implementation Complexity**:
    - Combining RL with transformers and cross-platform data requires substantial computational resources.
2. **Data Requirements**:
    - Requires extensive and diverse datasets to train models effectively.
3. **Generalization**:
    - May struggle with cold-start problems or new user scenarios.

Supporting Literature

1. **FedSlate**: Explores federated learning to model cross-platform behavior while maintaining privacy【9†source】.
2. **CQL**: Demonstrates effective stabilization for offline RL by addressing overestimation biases【10†source】.

3. **SNQN**: Enhances recommender systems by integrating negative feedback signals to reduce overfitting to positive data【10†source】.

This methodology provides a robust foundation for designing next-generation recommender systems that are both stable and adaptive to user preferences.

Presentation Text: Reinforcement Learning for Recommender Systems (15 Minutes)

Introduction (2 minutes)

"Good morning, everyone! Today, we'll talk about **reinforcement learning (RL)** and how it can improve recommender systems. Our goal is to explore a new method that solves key challenges, such as using data from multiple platforms, handling offline datasets, and learning from both positive and negative feedback.

I'll explain the method step by step, why it works, and what problems it solves. Let's get started!"

Section 1: Why Use RL in Recommender Systems? (3 minutes)

"Recommender systems predict what users will like—movies, products, or songs. Traditional systems use fixed algorithms based on past interactions, but RL is more dynamic. It learns to adapt to user preferences by treating the recommendation process as a series of **actions** in a changing environment.

For example:

- A user interacts with a platform.
- The system makes a recommendation (an action).
- The user's reaction (like, click, or skip) is feedback, which RL uses to improve future decisions."

Challenges in RL for recommendations:

1. **Multiple platforms**: Users interact across websites and apps.

2. **Offline data**: Real-world systems can’t always test recommendations live.
3. **Bias**: Systems often focus only on positive actions, ignoring skipped content.

Our method addresses these issues."

Section 2: The Proposed Methodology (6 minutes)

Step 1: Cross-Platform Behavior

"First, we handle data from multiple platforms. For example, users may watch YouTube, shop on Amazon, and use Spotify. To capture their full preferences:

- We treat each platform as an **agent** in a shared RL system.
- Using **federated learning**, the platforms share what they learn without exposing private data.

This allows the system to combine insights from all platforms and recommend better."

Step 2: Stable Learning with Offline Data

"Next, we use **Conservative Q-Learning (CQL)** to solve problems with offline datasets. Offline RL struggles because it often overestimates the rewards for unseen actions.

CQL fixes this by:

- Penalizing overestimation in the model.
- Focusing on actions observed in real data.

This makes the system stable and prevents it from suggesting irrelevant recommendations."

Step 3: Learning from Negative Feedback

"Most systems only learn from positive actions, like clicks or purchases. But negative signals, like skipped videos or ignored products, are also important.

We use **Stochastic Negative Q-Networks (SNQN)** to include these signals:

- The model reduces the chances of recommending skipped items again.
- This improves the relevance of suggestions for each user."

Step 4: Using Transformers

"Finally, we use **transformer models** to process sequences of user actions. Transformers are excellent for handling time-based data, so they:

- Understand long-term user preferences.
- Adapt quickly when preferences change.

For example, if a user starts watching more sci-fi movies, the system will learn this pattern faster."

Section 3: Benefits and Challenges (3 minutes)

"This method solves several problems:

1. It captures user behavior across platforms.
2. It learns effectively from limited offline data.
3. It considers both positive and negative feedback.

4. It adapts quickly using transformers.

However, it has challenges:

1. It requires advanced computational resources.
2. It needs large datasets for accurate learning.
3. It might struggle with new users (cold-start problem)."

Conclusion (1 minute)

"To sum up:

- Reinforcement learning offers dynamic, personalized recommendations.
- Our method improves cross-platform learning, stability, and adaptability.

Thank you for your attention! I’d be happy to answer any questions."