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

Zlata

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.

We'll explain our method step by step, why it would work, and what problems it solves. Let's get started!"

"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."

Slide 2: Core Idea

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 by core ideas represented on the slide that will be reviewed further.

Nastya

Slide 3: Modeling 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.

Slide 4: Stable Learning with Offline Data (CQL)

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.

Slide 5: Incorporating 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 can 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."

Diana

Slide 6: 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.

Slide 7: Addressing Key Challenges

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.

Slide 8: Moving Forward

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).

To sum up:

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

Thank you for your attention!