**Reinforcement Learning in Recommendation System**

**PROBLEM STATEMENT**

Traditional recommendation task can be treated as sequential decision making problem. Recommender (i.e. agent) interacts with users (i.e. environment) to sequentially suggest set of items. The goal is to maximize clients' satisfaction (i.e. reward). More specifically:

* State is a vector computed using the user embedding and the embedding’s of N latest positive interactions. In the code (replay buffer) state is represented by (user, memory)
* Action is a vector. To get ranking score we took dot product of the action and the item embedding (similar to word2vec and other embedding models).
* Reward is taken from user-item matrix (1 if rating > 3, 0 otherwise)

Reinforcement Learning can help recommendation at least in 2 ways.

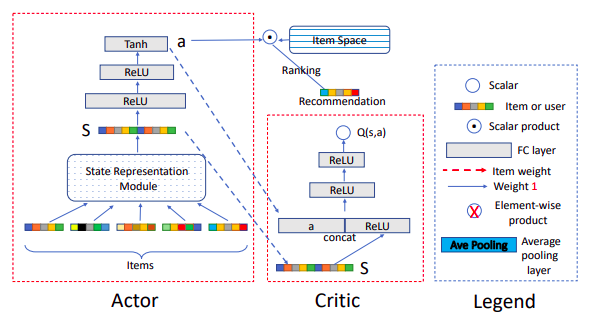
1. User’s preference on previous items will affect his choice on the next items. User tends to give a higher rating if he has consecutively received more satisfied items (and vice versa). So, it would be more reasonable to model the recommendation as a sequential decision making process.
2. It is important to use long-term planning in recommendations. For example, after reading the weather forecast, the user is not willing to read similar news. On the other hand, after watching funny videos or reading memes the user can constantly do the same.

**Environment**

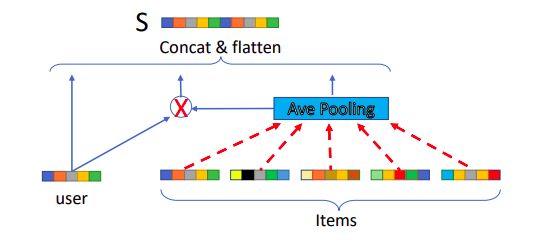
* **Observation space**. As mentioned before, to get state we need N latest positive items (memory) and embedding of user. State\_Repr\_Module transform it to the vector of dimensionality embedding\_dim \* 3.
* **Action space**. For every user we sample nonrelated items (the same count as related). All available\_items which wasn't viewed before form action space.

Given a state we get action embedding, compute dot product between this embedding and embeddings of all items in action space, take 1 top ranked item, compute reward, update viewed\_items and memory, and store transition in buffer.

**OVERALL MODEL**



**STATE REPRESENTATION**



For evaluation we take 1 positive and 99 sampled negatives items per batch, select 10 items with best scores and calculate hit\_rate@10 and nDCG@10. During training we choose user 6039 and track hit and dcg only for him (for evaluation speed). Final scores was computed on the whole test data.