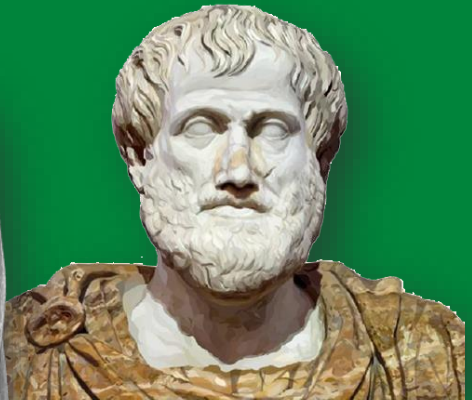
A Collective Understanding of Happiness

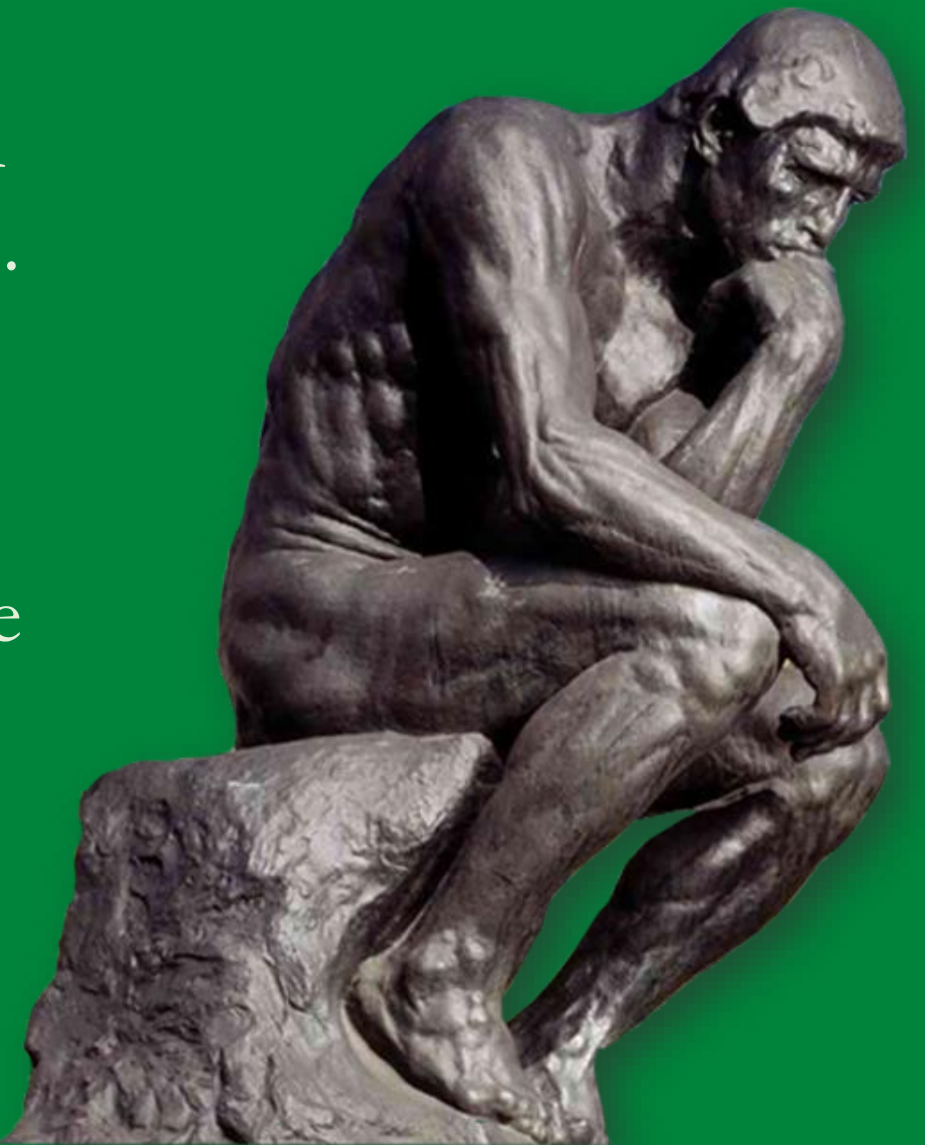
"Brother Gallio, all want to be happy, but when it comes to see clearly what makes life happy, they are shadowed by obscurity" - Seneca, *De Vita Beata*



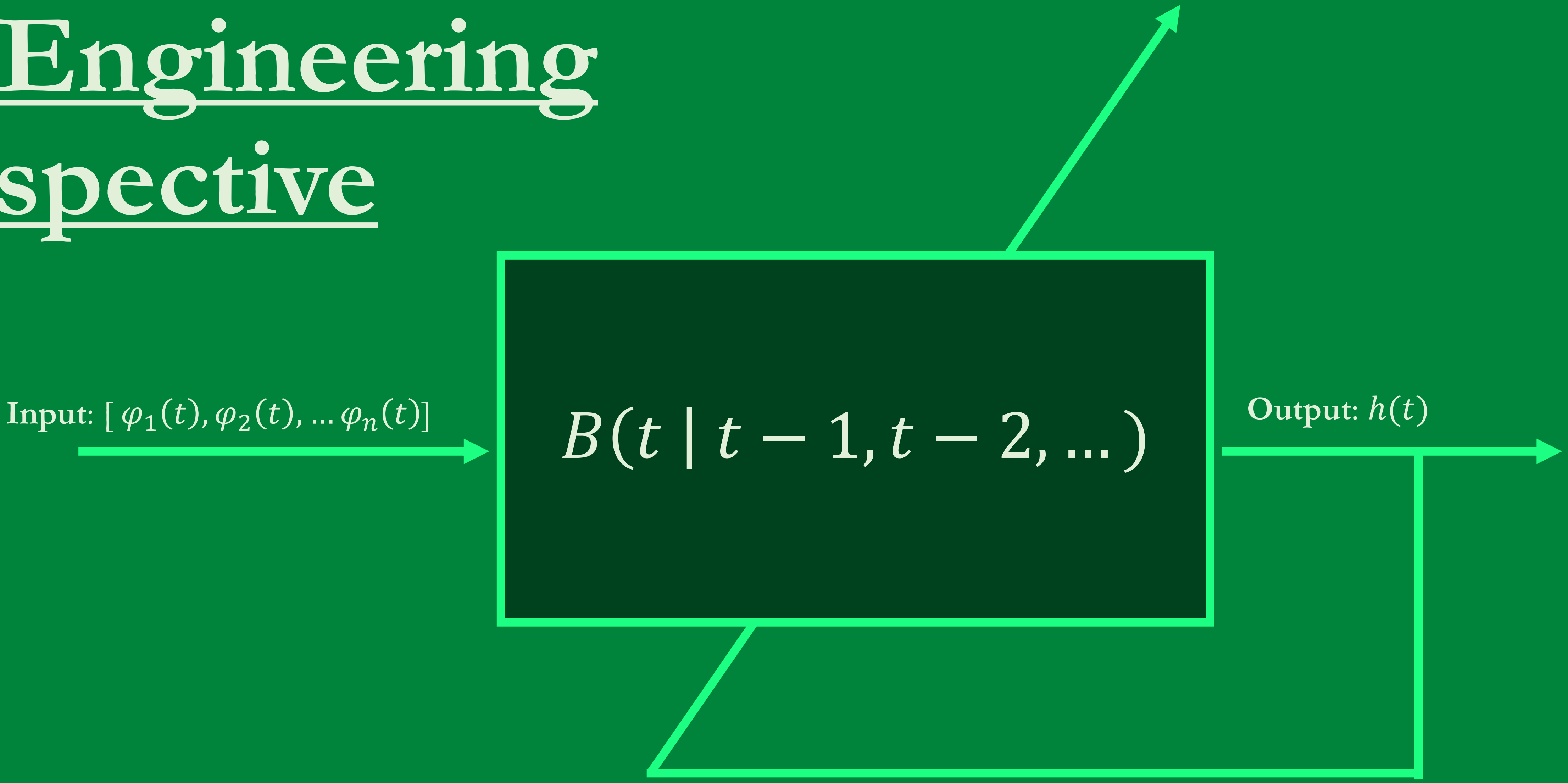
Presentation by Nima Karshenas

Motivations and Goals

- Employ engineering techniques to broaden our collective understanding of 'happiness’.
- Develop a tangible tool based on this understanding for anyone to use.
- Focus on implementing an actionable guide for users
- Empower individuals to embark on their own journey of sustained happiness.

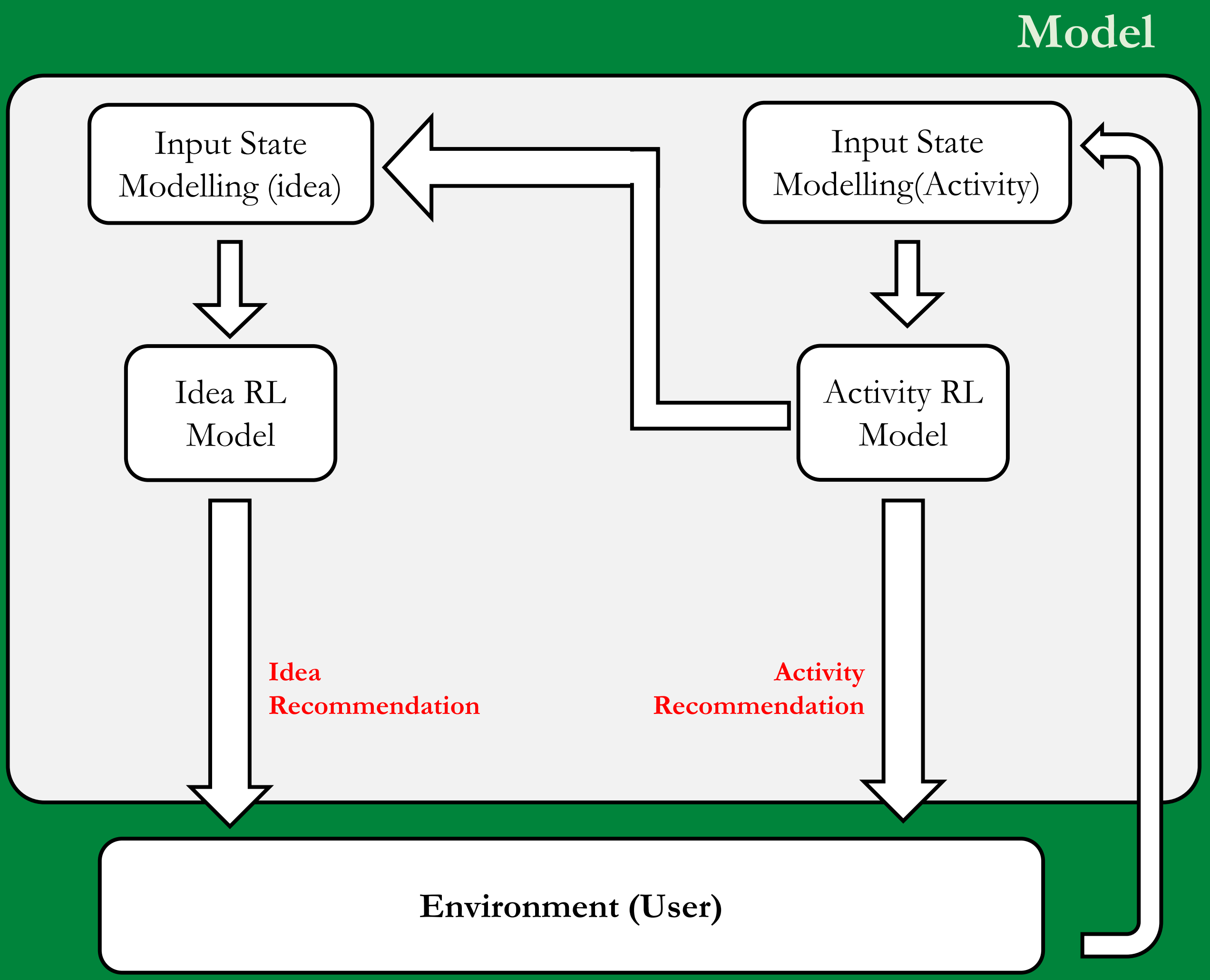


The Happiness System: An Engineering Perspective

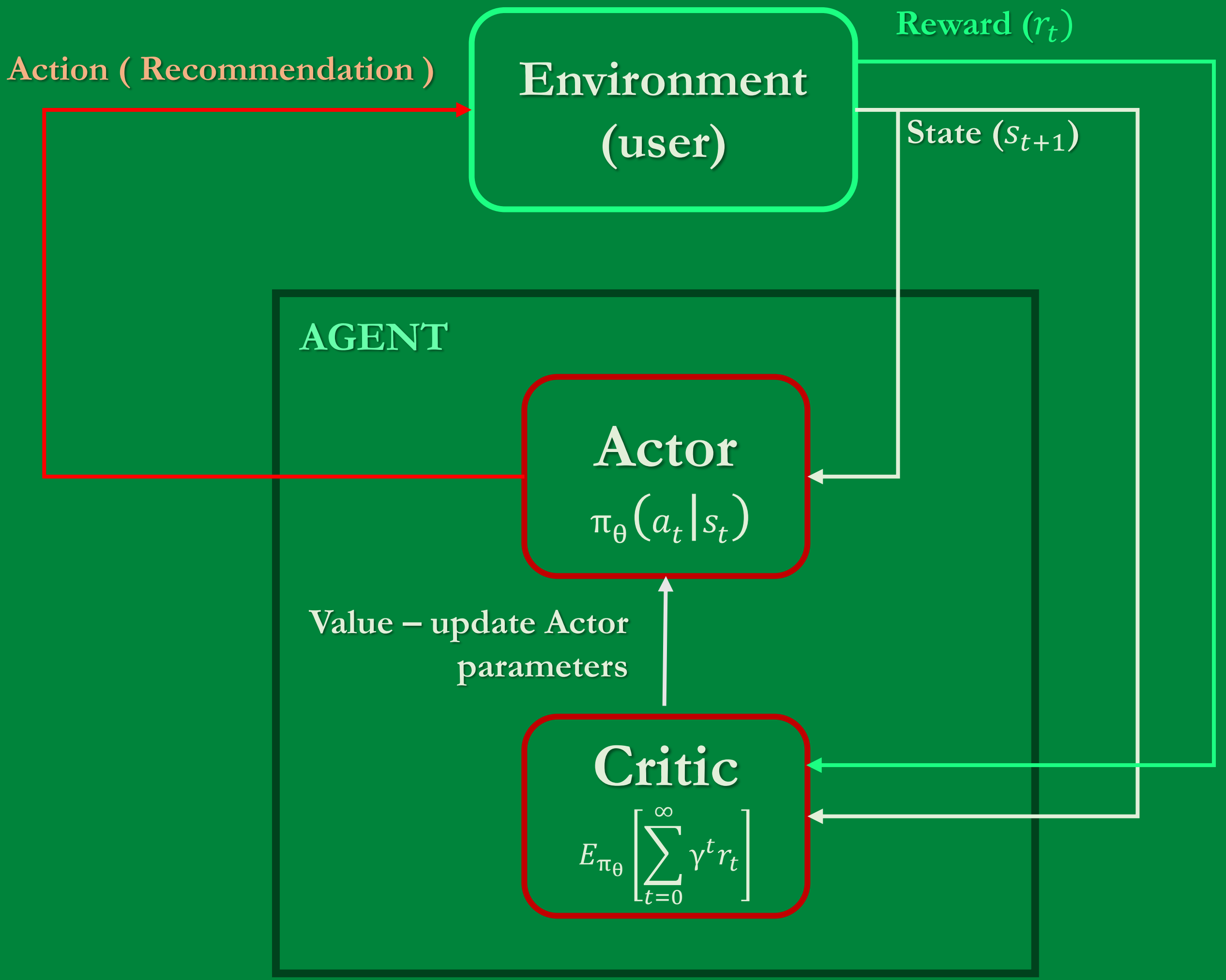

Input: $[\varphi_1(t), \varphi_2(t), \ldots \varphi_n(t)]$
$B(t \mid t-1, t-2, \ldots)$
Output: $h(t)$

System Design

- There is no available pre-existing data to draw from.
- Dealing with a semi-supervised environment.
- Recommendations are the 'guide' for the user.
- As the model interacts with users, its understanding of the environment grows.
- The model must allow for our own understanding to grow with the model.
- Recommend both an idea and an activity, allows for the investigation of the interplay between the two domains.



Actor-Critic and Reinforcement Learning



Action (Recommendation)

Environment (user)

Reward (r_t)

State (s_{t+1})

AGENT

Actor

$\pi_\theta(a_t|s_t)$

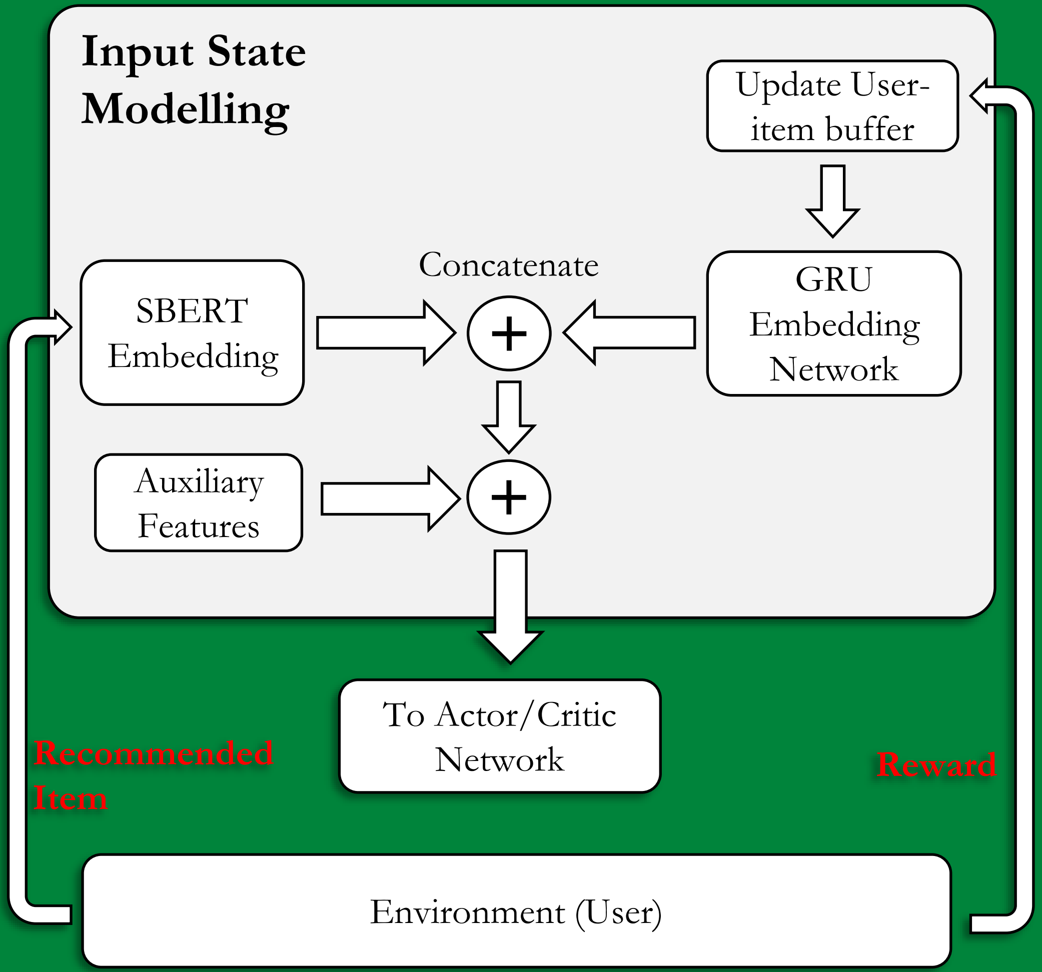
Value – update Actor parameters

Critic

$E_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$

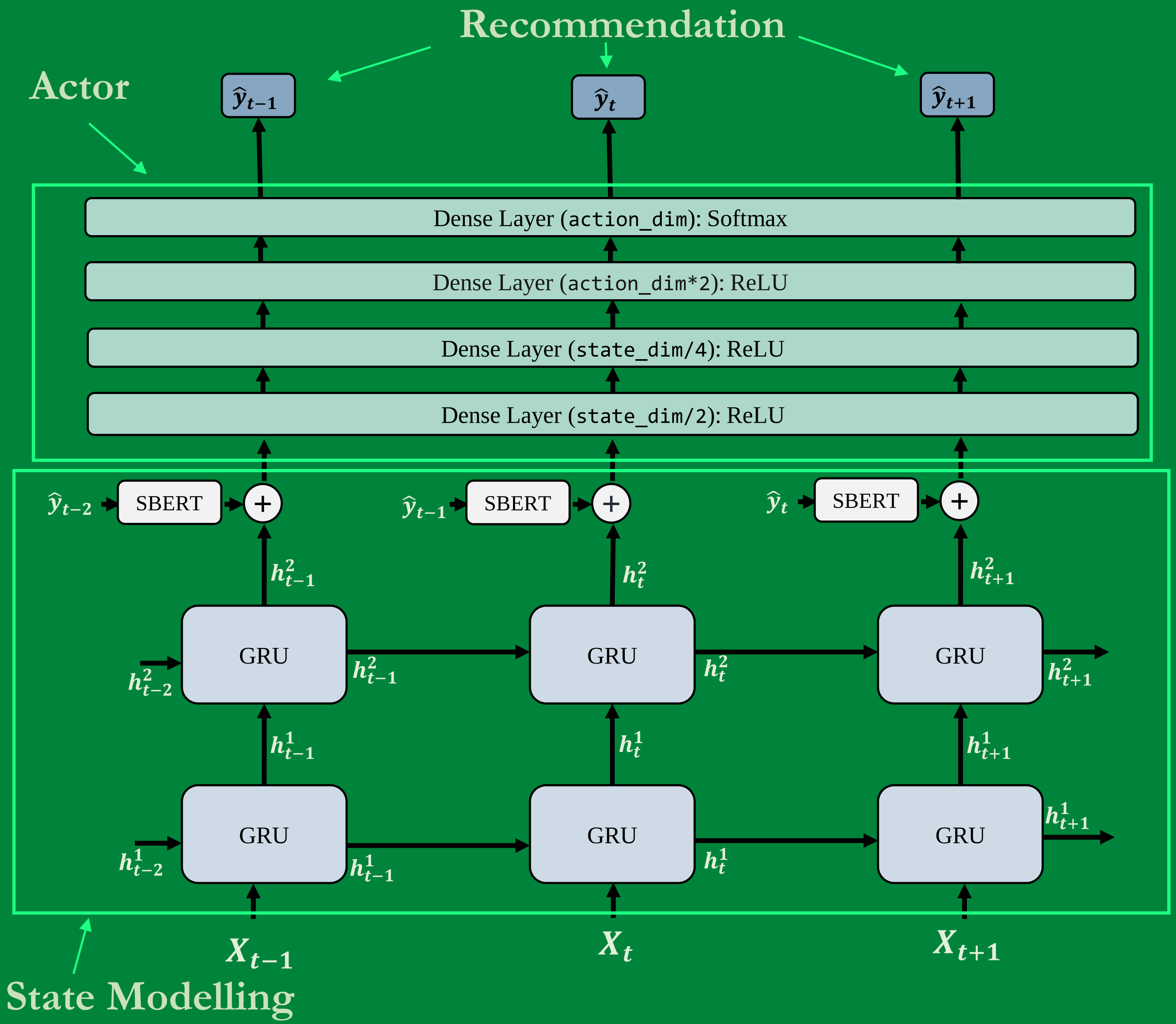
- Semi-supervised, Markov Assumption, consists of an agent interacting with an environment.
- Policy: The policy $\pi_\theta(a_t|s_t)$ determines the probability of taking action a_t when in a given state s_t, to arrive at a new state s_{t+1} .
- Reward: Instantaneous feedback from the environment captured by reward signal r (via Critic), used to improve the policy π.
- Value: Critic estimates the expected discounted future reward, $E_{\pi_\theta}[\sum_{t=0}^{\infty}\gamma^t r_t]$, and uses this to update both Critic and Actor parameters.
- Actor-Critic allows for more stable, and sample efficient learning than other popular counterparts (Q-Learning, REINFORCE etc.)
- Allows for transparency we are looking for.

State Modelling



- Content Modelling: Previous recommended item by the Actor is embedded by SBERT.
- Allows for relativised embeddings that is robust to a growing item space.
- User modelling: Trainable GRU embedding network.
- Taste-profile built from past interactions.
- Deals with the non-Markov property that was hypothesised.
- Include Auxiliary features such as personality type, age, location, etc.

Actor-Critic to Deep Advantage Actor-Critic



- Input State passed through Actor Densely-connected Neural Network.
- Outputs set of probabilities for each item/recommendation.
- Critic model is identical, but the final layer consists of a single neuron.
- Advantage guides the actor and is the improvement in value by taking action a_t when in state s_t:

$$\hat{A}(s_t, a_t) = r_t + \widehat{V_{\theta_V}}(s_{t+1}) - \widehat{V_{\theta_V}}(s_t)$$

Network Update Procedure

Actor Loss:

$$\mathcal{L}(\theta) \approx \sum_{t=1}^{T} \log \pi_\theta (a_t|s_t) \cdot \hat{A}(s_t, a_t) \quad \rightarrow \quad \nabla_\theta \mathcal{L}(\theta) \approx \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta (a_t|s_t) \cdot \hat{A}(s_t, a_t)$$

Critic Loss:

$$\mathcal{L}(\theta_V) = \sum_{t=1}^{T} \left(r_t + \gamma V_{\theta_V}(s_{t+1}) - V_{\theta_V}(s_t)\right)^2$$

- Losses update their respective network parameters via backpropagation.
- If Advantage of action is large, then the Actor's parameters are updated so that the log of the probability of taking that action is larger.
- Critic aims to minimise the squared value error estimation.
- It is biased since it is using model estimate to calculate the future discounted rewards, but allows for less variance, and iterative updates.

Reward Formulation

- Inspired by Lyubomirsky et al.'s widely accepted, and used, Subjective Happiness Scale (SHS), user presented with items:

For Activity Recommendation:

- On the whole, this activity contributed positively to my overall happiness.
- I found that this activity matched my interests well.

For idea Recommendation:

- On the whole, this idea contributed positively towards my overall happiness.
- I found that this idea complimented my current outlook and thoughts.

Evaluative Measure (not used to train parameters):

- On the whole, I am satisfied with my life, and I consider myself generally happy.

Activity and Idea data collection

- Agreed that ideas were to consist of ‘bitesize’ segments or summaries of philosophical text to keep user-interaction and data-collection frequent.
- First attempted to use state-of-the-art extractive summary networks for philosophy chapter summaries.
- Huge variance in chapter lengths and styles meant that summaries were extremely varied in quality.
- Data-mined the top 1000 philosophy quotes on GoodReads.
- Manually filtered quotes for them to be actionable or related to outlook.
- No definitive or formal list of activities existed online, so dataset was built by brainstorming with peers from a variety of backgrounds and ages.
- 186 Quotes and 84 Activities, was deemed sufficient for proof-of-concept purposes.

Experiment: Data Collection

- The infancy of this research topic meant no available dataset existed for such a problem.
- Couldn't settle for a fully simulated testing environment, it is difficult to simulate data for an environment you are attempting to investigate and know little about.
- Semi-simulated data, 5 subjects, each presented with each activity, and asked to give response between -5 and 5 (disagree to agree) for each item in each season, according to following questions:
    - **On the whole, I believe this activity can contribute positively to my overall happiness.**
    - **This activity matches my interests well.**
- Seasonal answers asserts some additional temporal information in the data.
- Experiments for the Idea network were omitted. Subjects had a strong conceptual idea of what each activity entailed, and how happy it may make them. Ideas were novel to them, and subjects couldn't give an evaluation of how they would manifest in their experience of happiness.

Experiment: Environment Design

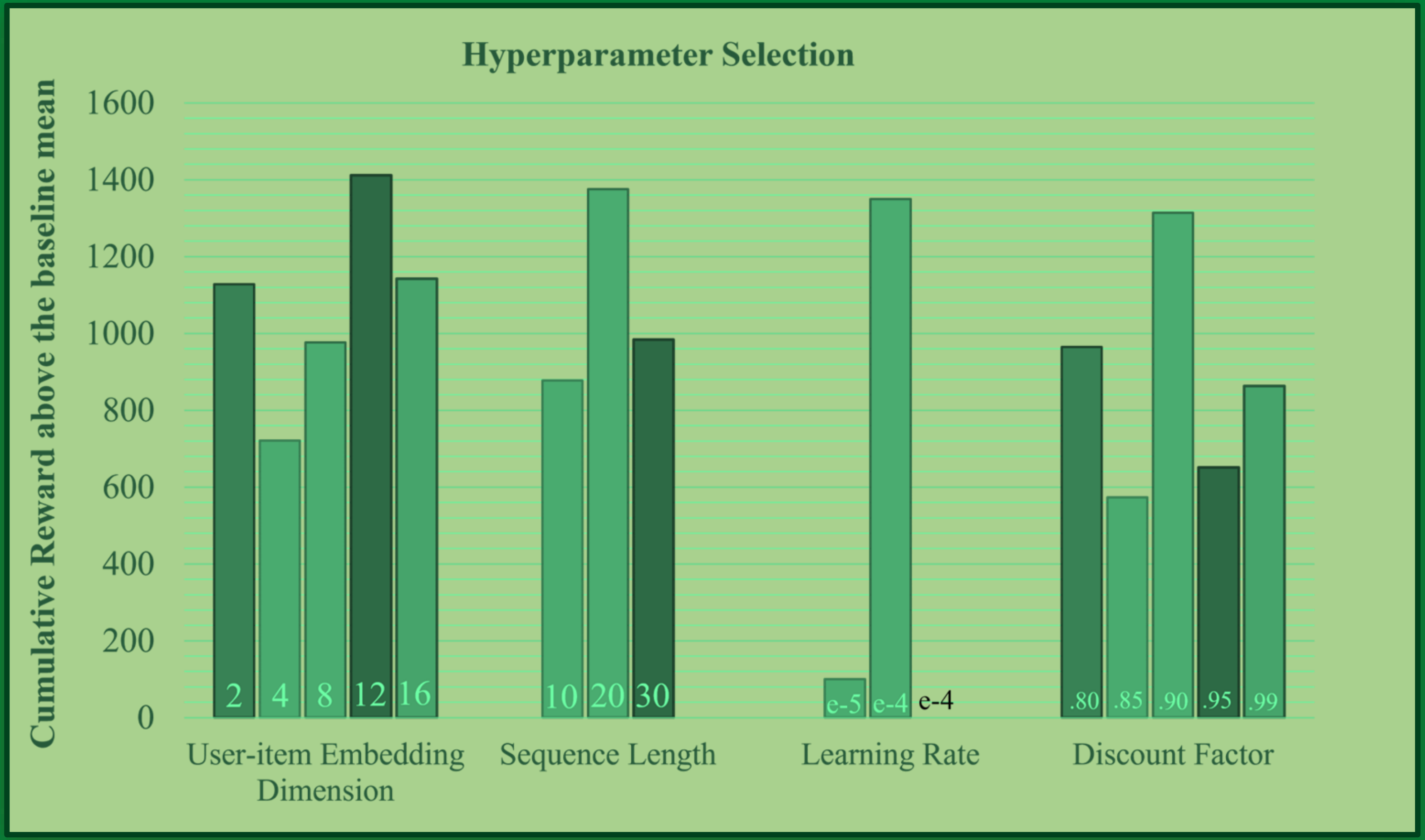
- Need to add additional temporal complexity, the model would trivially collapse onto a repeated recommendation for best activity for that season.
- Additional penalty for repeated recommendations in a fortnightly window:

$$r_t = \begin{cases} r_{season,j,i} \cdot 0.5^{C(t)}, & \text{if } \; r_{season,j,i} > 0 \\ r_{season,j,i} \cdot 2^{C(t)}, & \text{if } \; r_{season,j,i} < 0 \end{cases} \quad \longleftarrow \quad C(t) = \sum_{k=1}^{14} [a_{t-k} = a_t]$$

Where $r_{season,j,i}$ is the reward given by subject j for activity i for the given $season$.

Hyperparameter Selection: Finding Balance

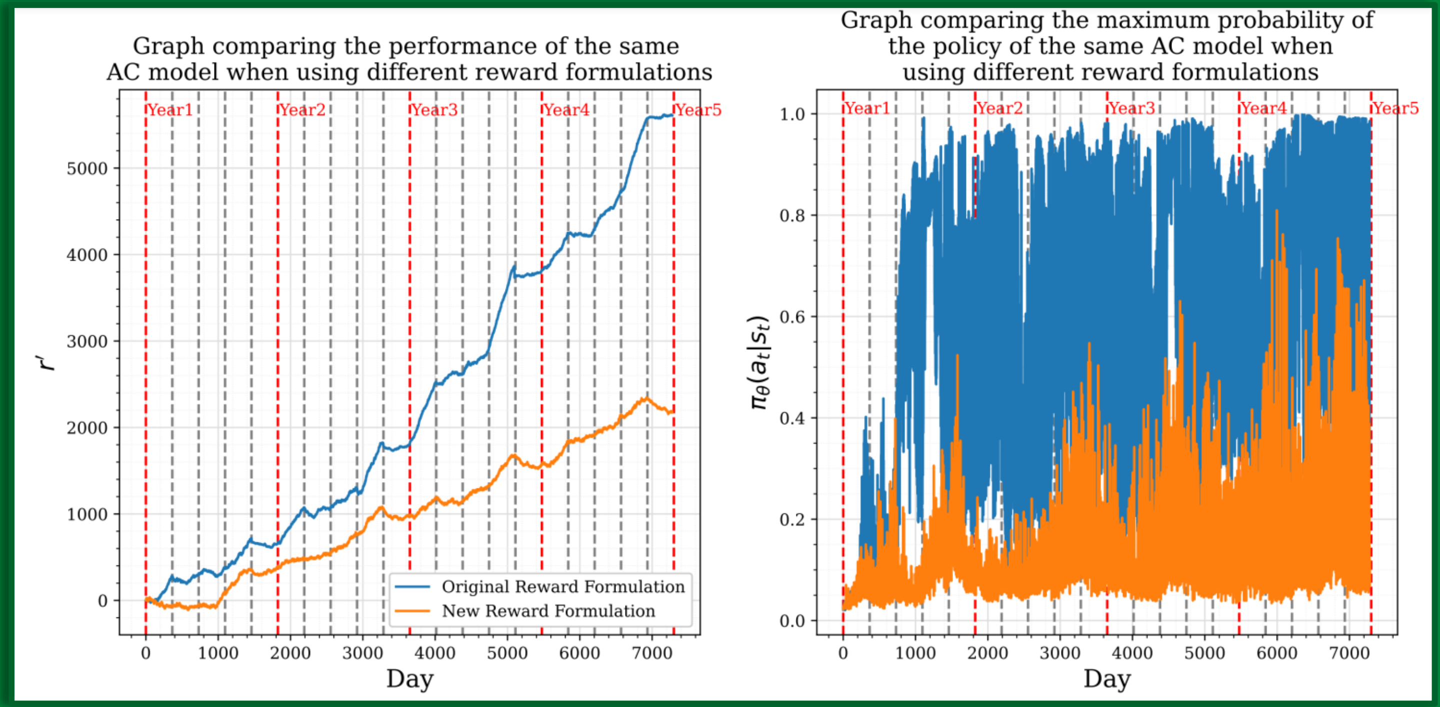
- **Embedding Dimension**: Balance between carrying sufficient information forward and complexity.
- **Sequence Length**: Need to capture the fortnightly buffer without adding complexity that slows learning.
- **Learning Rate**: Needs to be sufficient to adapt to changing environment at boundary of season, without causing instability and policy collapse.
- **Discount Factor**: Need to put enough weighting on rewards within fortnightly window without too much for rewards outside.



Importance of Reward Formulation

- Noticed model was increasingly making repeated recommendations.
- Penalised the model further by making reward negative for repeated recommendations.

$$r_t = \begin{cases} -5 + r_{season,j,i} \cdot 0.5^{C(t)}, & \text{if } r_{season,j,i} > 0, C(t) > 0 \\ r_{season,j}, & \text{if } r_{season,j,i} > 0, C(t) = 0 \\ r_{season,j,i} \cdot 2^{C(t)}, & \text{if } r_{season,j,i} < 0 \end{cases}$$



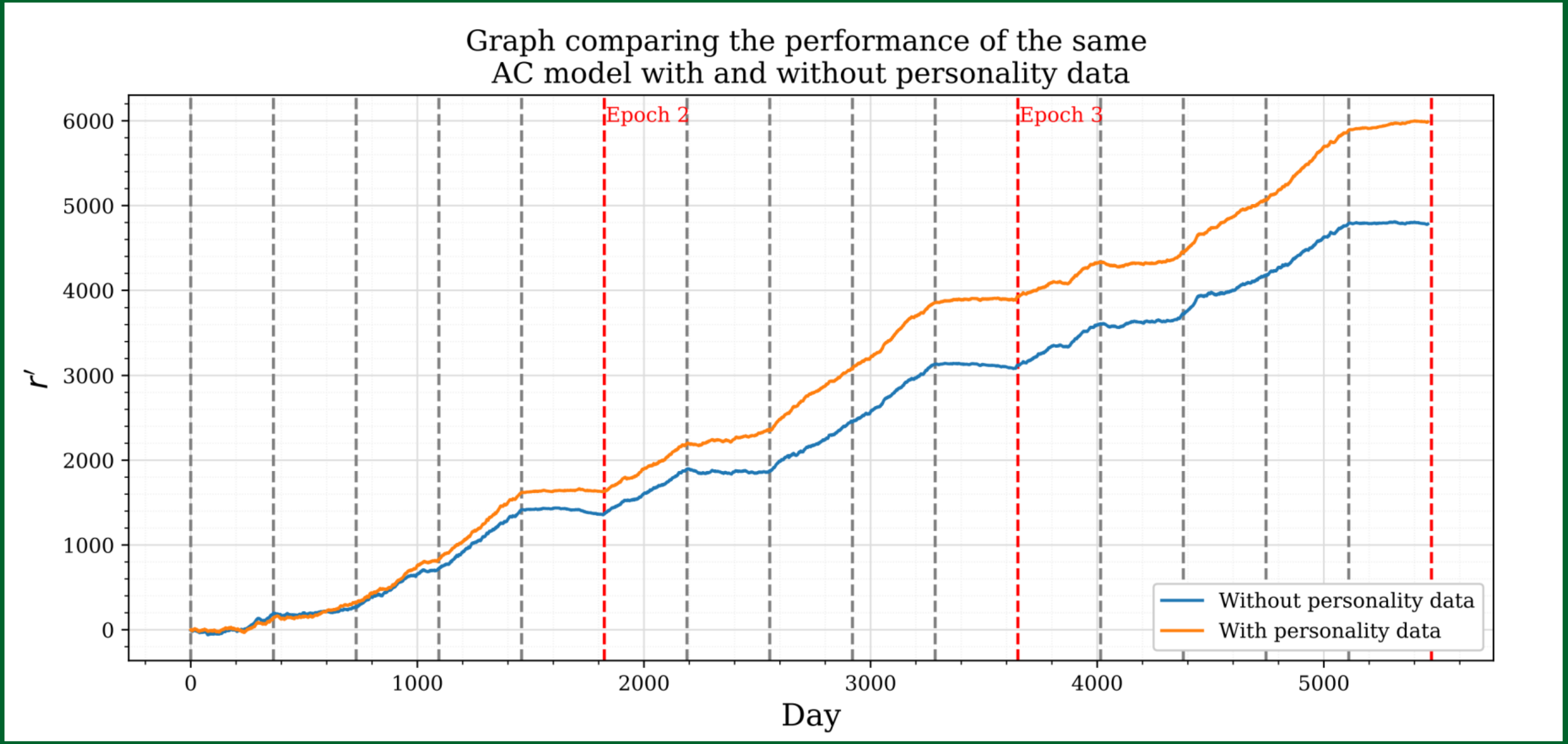
Graph comparing the performance of the same AC model when using different reward formulations

Graph comparing the maximum probability of the policy of the same AC model when using different reward formulations

- Slower to learn in more complex environment.
- Highlights the importance of reward formulation or nature of recommendations.

Reward Formulation	Number of repeated recommendations	Number of rewards equal > 3
Original	3217	790
New	1743	2292

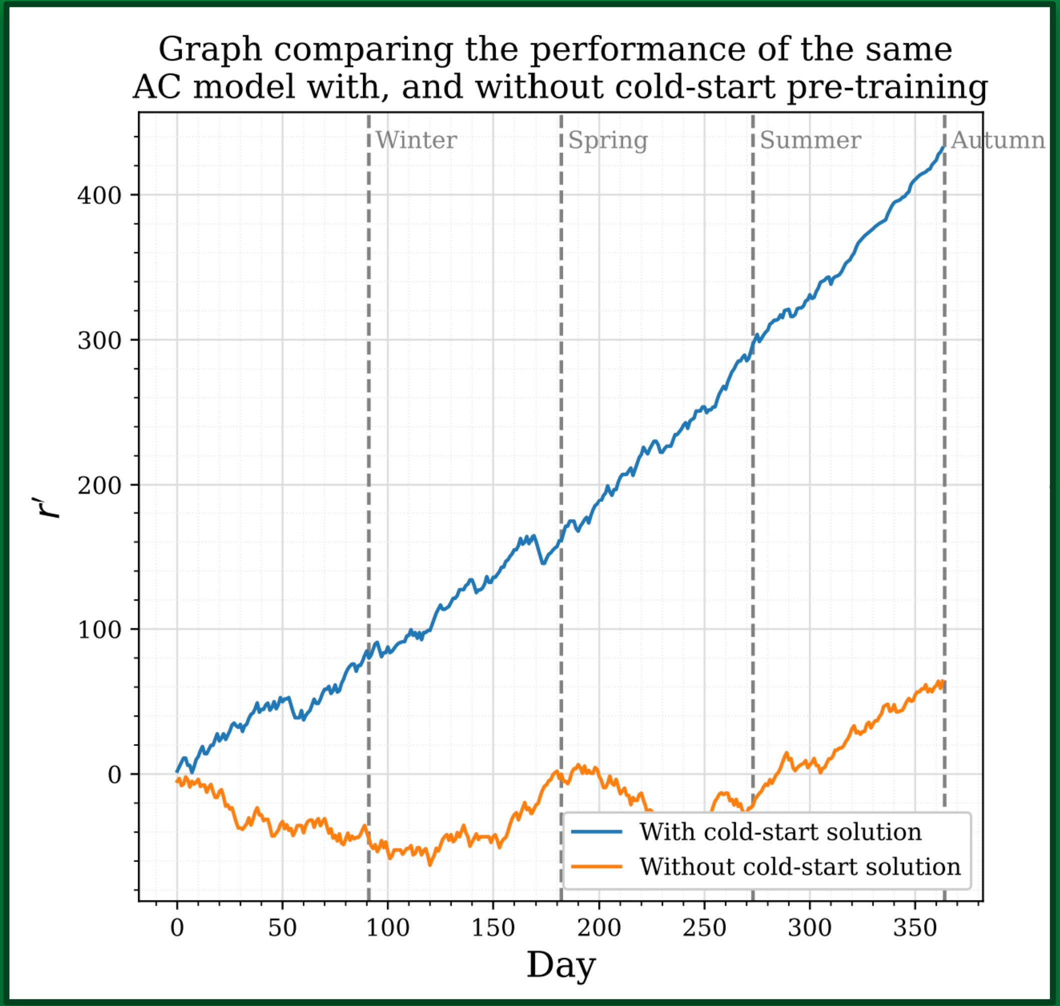
Prediction Power of Personality



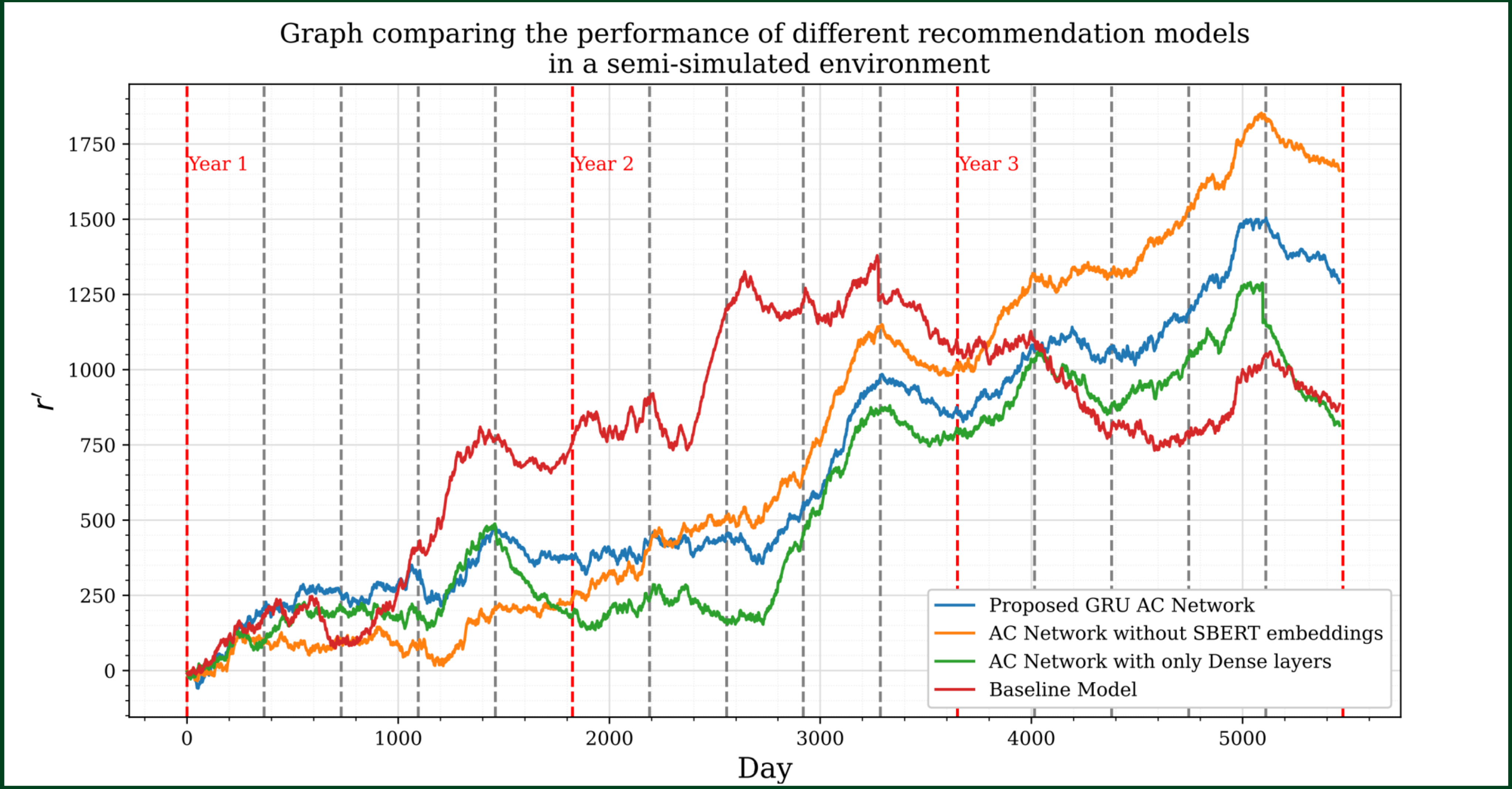
- As the model interacts more and more with the environment, personality is only increasing in its prediction power.
- Encourages the investigation and further use of user-based features.

The Cold-Start Problem

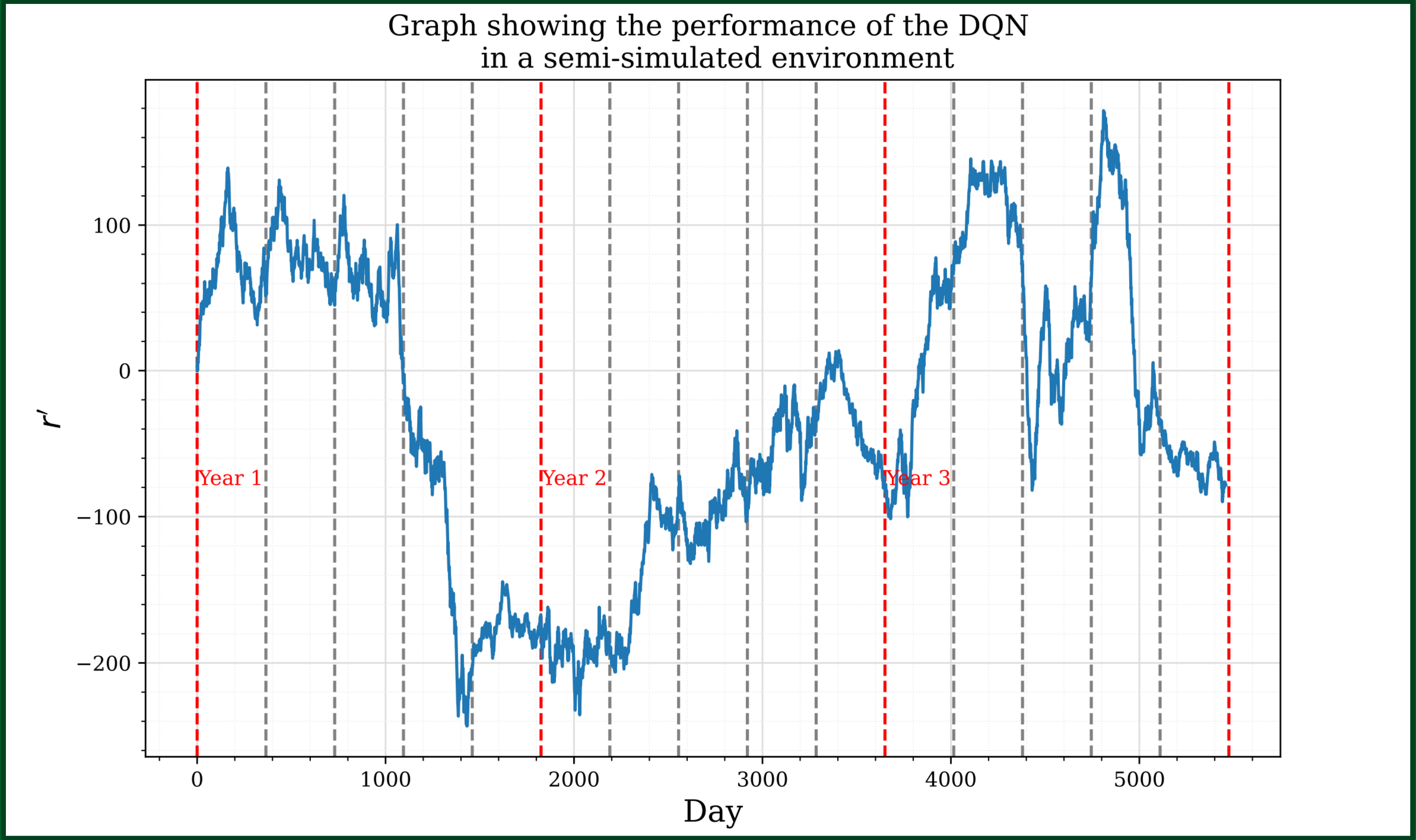
- Inherent problem in Recommender Systems, very difficult to give good recommendations when interactions have been few.
- Proposed solution: Ask user to list some of their favourite activities per season.
- Use cosine-similarity between list and rest of activities to get a set of rewards.
- Pre-train on the 'harsher' environment shown two slides ago.


Graph comparing the performance of the same AC model with, and without cold-start pre-training

Evaluation


Graph comparing the performance of different recommendation models
in a semi-simulated environment
Year 1
Year 2
Year 3
1750
1500
1250
1000
750
500
250
0
r'
0
1000
2000
3000
4000
5000
Day
Proposed GRU AC Network
AC Network without SBERT embeddings
AC Network with only Dense layers
Baseline Model

Poor Q-Learning performance



Future Work

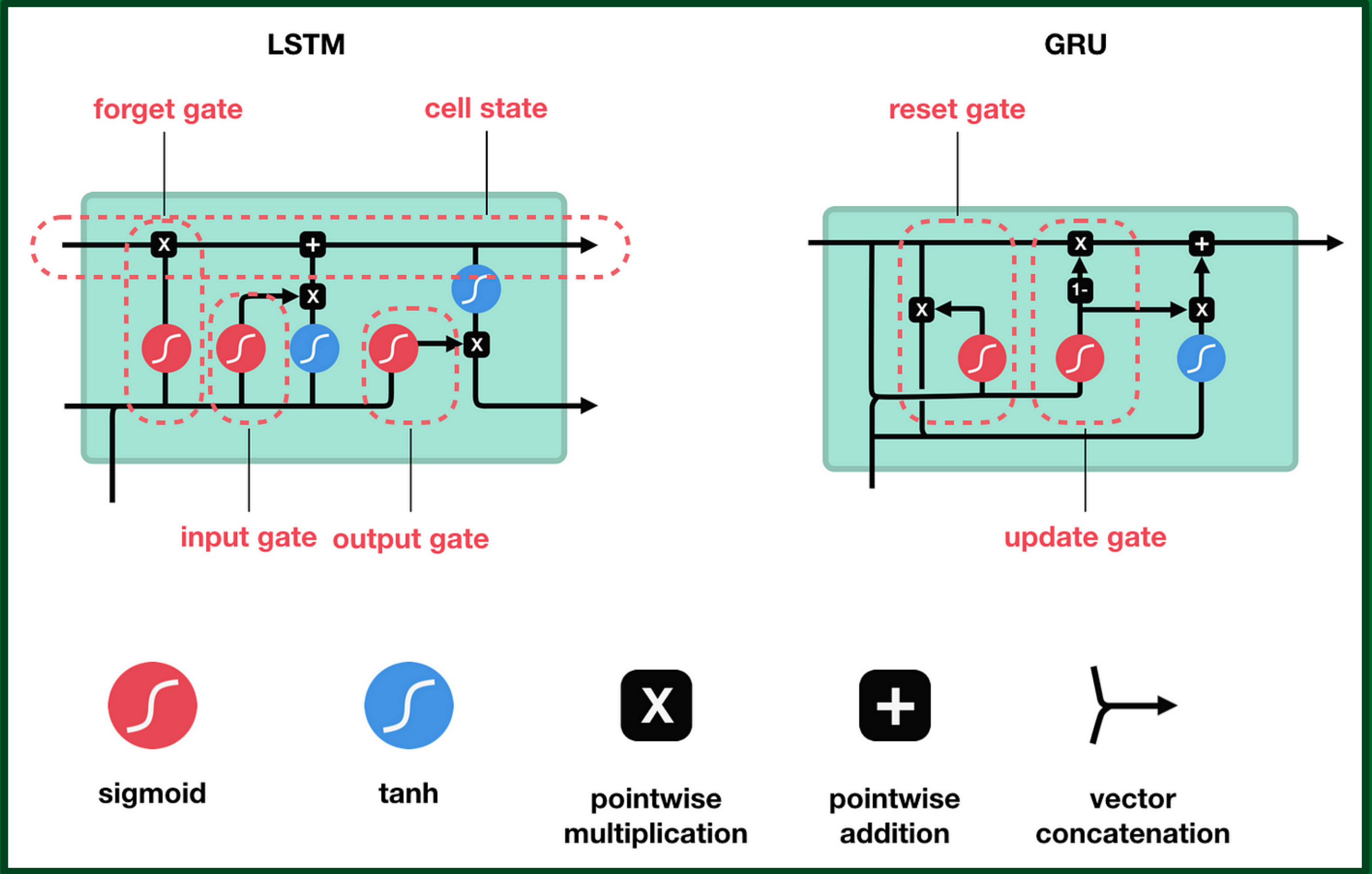
- Data collection from the intended environment, under full test conditions.
- More formal curation of ideas and activities
- Psychological research is needed to determine appropriate and optimal questions for user feedback.
- The initialization of the model for new users can be improved to enhance user experience and retention.
- Exploring variations to the model.
- Research into additional input features related to happiness, can enrich the model's representation and understanding of individual happiness.

Conclusions

- Consulted existing psychology and philosophy research on happiness and integrated it into an engineering framework.
- Developed a novel approach using deep learning models to maximize long-term experienced happiness.
- Designed a deep actor-critic network to capture non-linearities and time-variance of happiness, operating in a semi-supervised, model-free environment.
- Enabled inference about experienced happiness based on input features such as age, personality type, and location, utilizing model parameters and outputs.
- Laid the groundwork for future research in the field of engineering and happiness, although further work is needed for reliable conclusions.

Gated Recurrent Units (GRUs)

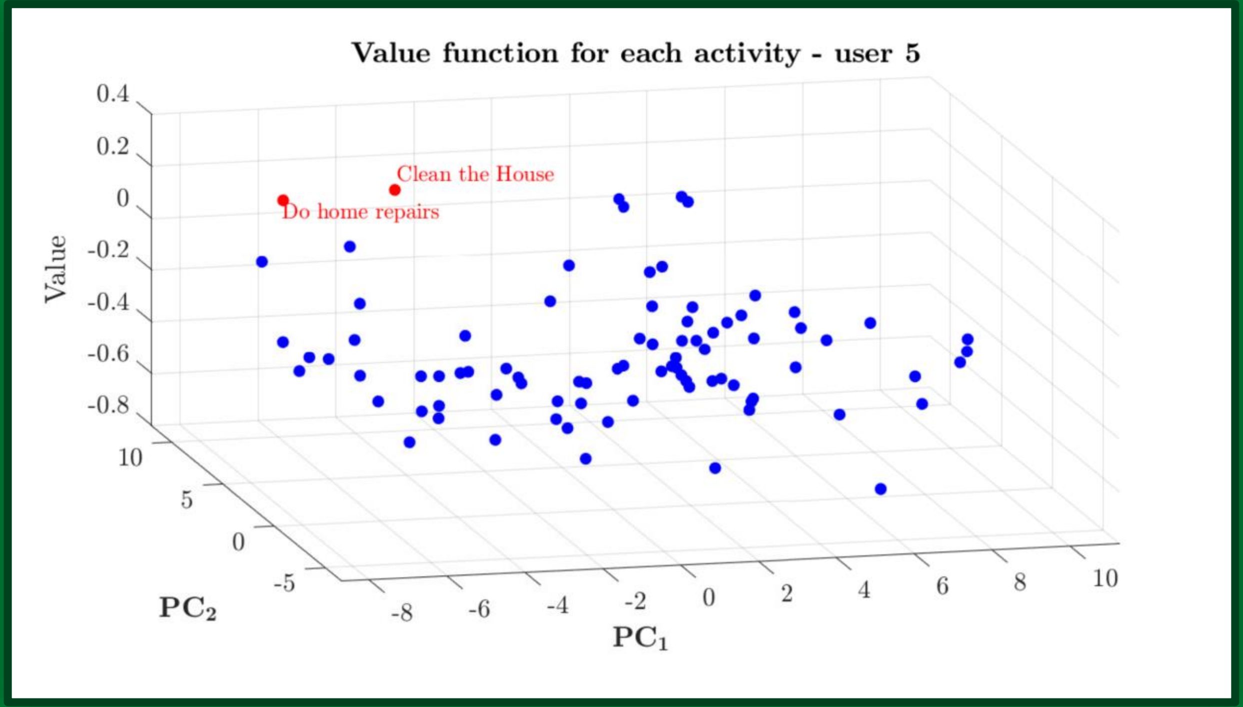
- The update gate helps the model to determine how much of the past information is carried forward.
- The reset gate is like the forget gate, it decides how much information is forgotten.



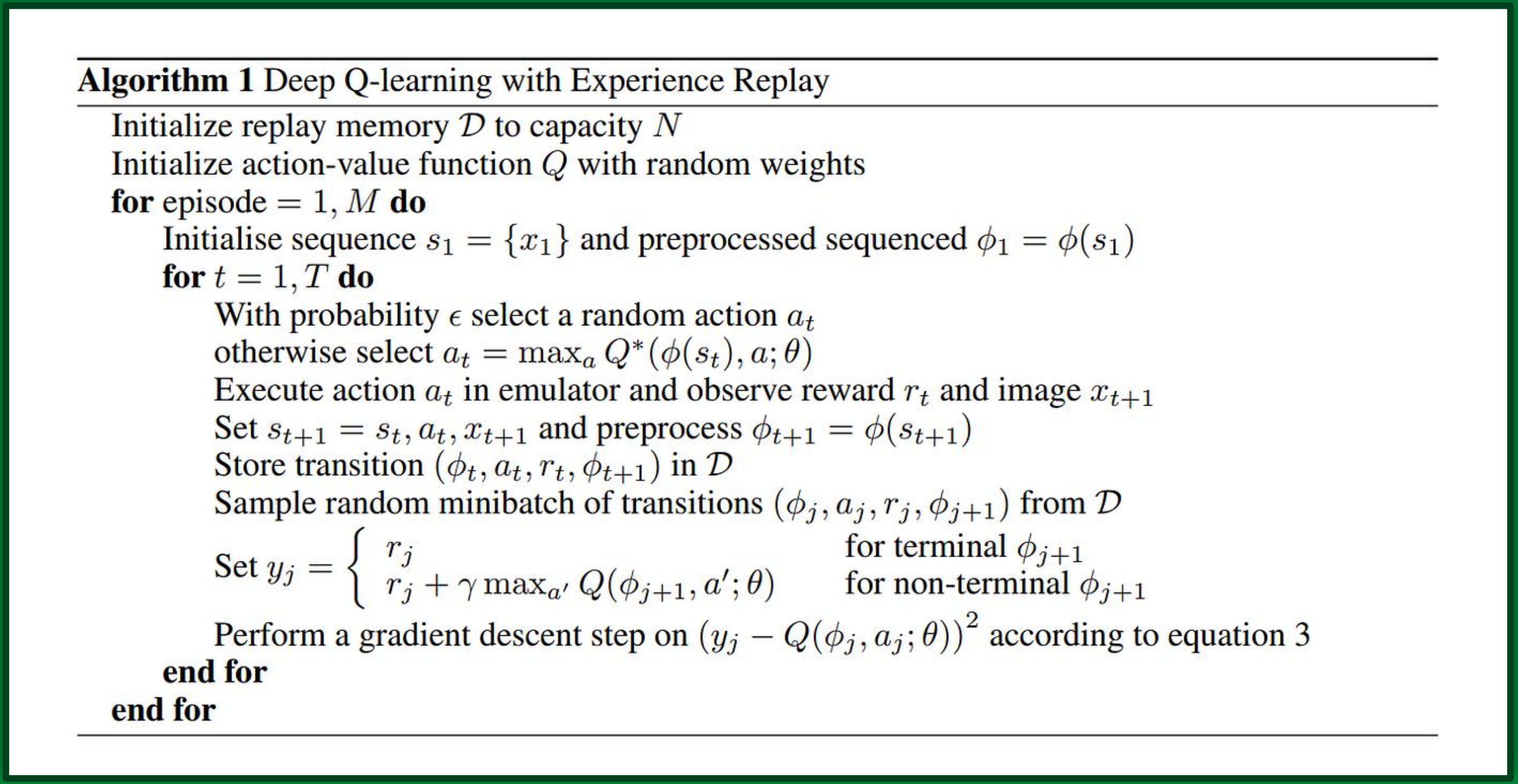
Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Inspecting the Value Function

- The average score given by user 5 over the four seasons for the ten activities with the highest value, in ascending order, left-to-right were: {0.5, 1, 3.25, 0.75, 2, 0, 2, 2.25, -2, -1}.
- Model has not converged yet.
- Value is more representative of what is more valuable to the model in terms of learning.



Deep Q-Learning

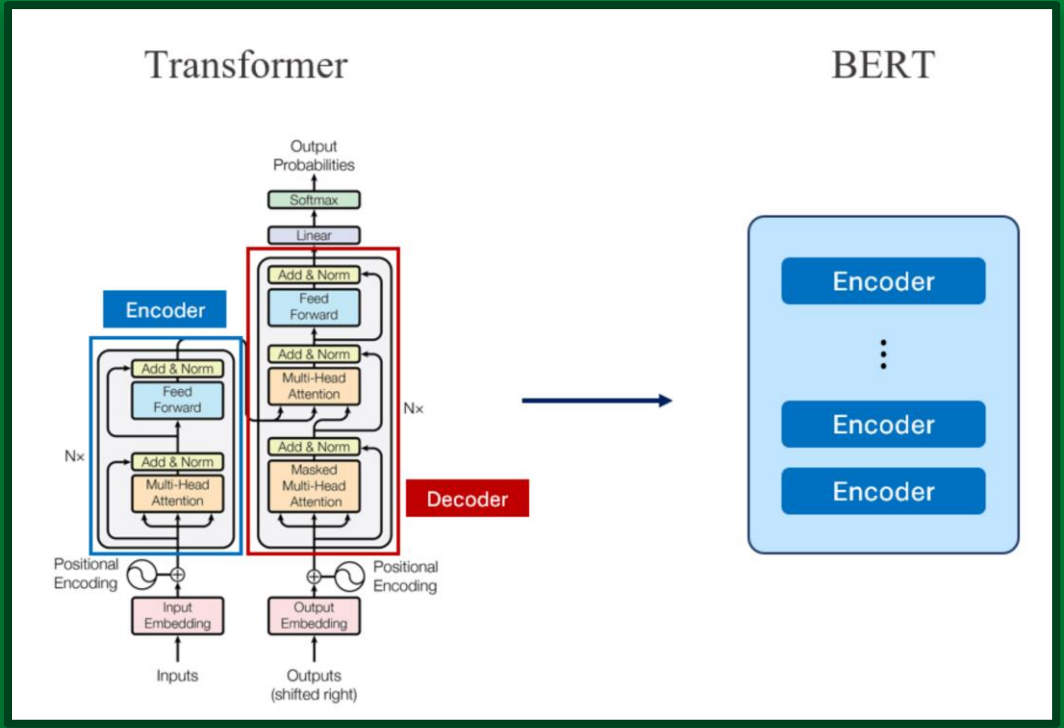


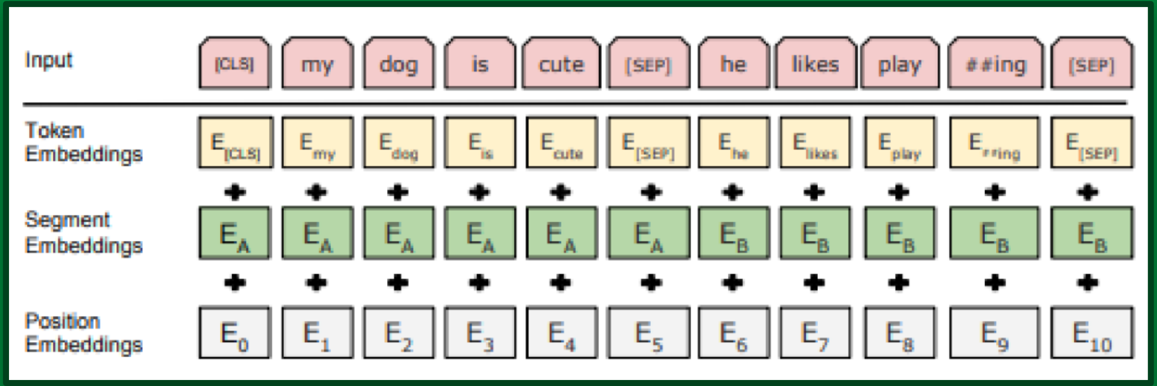
Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights
for episode $= 1, M$ do
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for
```

Source: https://andrew-gordienko.medium.com/reinforcement-learning-dqn-w-pytorch-7c6faad3d1e

BERT and SBERT





Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

- Trained on Mask Language Modelling (MLM) and Next Sentence Prediction (NSP).
- Byte pair-encoding: Most common pair of consecutive bytes of data is iteratively replaced with a byte that does not occur within that data