AI Final Project Writeup

*Emma Crawford*

*12/18/2023*

# Overview

I used a custom-built environment with a Q-learning training loop and a DQN training loop. My environment simulates the process of a child learning language over the first three years of their life, with different phonemes and sentence structures introduced each year. The environment is designed with specific states and actions that reflect the phoneme selection and sentence formation process. The reward system is based on correct guesses, partial matches, and the responsiveness of the parent in the simulated environment. In addition, the parent provides “clues” by giving the child a response sentence if they are at least partially successful. The agent's actions involve selecting phonemes to form words and sentences, aiming to match a target structure depending on the current year of learning. I compared Q-learning and DQN to learn the optimal strategy. Throughout the training process, the child (i.e., agent) explores and exploits the environment by selecting phonemes that are appropriate for their age to try and elicit a response from the parent, ideally, a cookie which has the highest reward. This implementation demonstrates how reinforcement learning can be applied to a language learning simulation.

# Approach

## Environment

### Objective

Teach the child to say “cookie” to get a cookie, along with gaining the attention of the parent within temporal and linguistic development constraints.

### Setting & Mechanics

#### Characters:

There are two main characters - Parent (within the environment) and Child (AI agent)

#### Time Progression:

The simulation takes place over three years with each step represented by an hour. Each year introduces more complex language elements. Each hour simulates varying probabilities for success. For example, the child has a lower probability of success when the parent is sleeping, even if they provide a correct answer.

#### Phoneme Libraries:

* **Child Dictionary**: Contains simplified phoneme sets for each year, with increasing complexity.
* **Adult Dictionary**: A comprehensive set of phonemes used to form responses and guide the child.

#### Actions:

The parent selects phonemes to respond to the child's attempts at communication. The parent’s choices include consonants and vowels from the adult dictionary. The game's action space changes based on the child's current year, reflecting their language development stage. The child can choose to rest or produce a word, which can be informed by the parent’s response which is guaranteed to include the target word (e.g., “cookie”).

#### States:

* **Time Variables**: Hour of the day, day of the year, and the child's year level.
* **Parent and Child States**: Represent activities like sleeping, eating, responding, or giving/receiving a cookie based on the time and interactions.

### Rules

#### Interaction and Responses:

For each step, the child can either give no response or can attempt a guess at the target word with their linguistic constraints. For example, in year 1, the child can only combine a consonant and vowel and the types of consonants and vowels are limited when compared to year 3 when the child can use up to 2 consonants and 2 vowels in a guess. If the child guesses at least partially correctly, the parent will reward them with points and a clue.

#### Reward & Punishment System:

* **Cookie Count**: Earned when the child guesses the target correctly and the parent is available.
* **Parent Responses**: Points and clues are given when the child guesses at least partially correct.
* **Non**-**Response**: Points are deducted for opting to not take a guess.
* **Wrong Guess**: If the child’s guess doesn’t even partially match the target, points will be deducted.
* **Total Reward**: Calculated based on performance, including cookie counts, frequency of parental responses, and penalties for wrong guesses or non-response.
* **Reward Function**: 1\*cookie\_count + 0.15\*parent\_response - 0.05\*wrong\_guess - 0.075\*no\_action

#### End:

The simulation concludes after three years, with a final score based on the overall performance of the child.

## Models

### Q-Learning

I started with the model-free reinforcement learning model to get a baseline. The model generates valid actions based on the child's year, considering the probability of taking no action. The training loop runs for a set number of episodes, updating the Q-table based on rewards received and the agent's actions. I started testing with 10-25 episodes, but ultimately tested with up to 100 episodes. I chose this algorithm to start because Q-Learning can adapt its strategy to the changing complexities of the environment and because it is relatively simple, making it a good place to start and collect some data about how the agent performs. In addition, the use of epsilon, discount rate, and learning rate allowed me to experiment with difference features of the agent to find the ideal balance, particularly of exploration and exploitation.

### DQN

This network consists of two dense layers with 64 neurons each, with ReLU activation functions for non-linearity. To prevent overfitting, each dense layer is followed by a dropout layer, which randomly omits a fraction of the neurons during training, forcing the network to learn more robust features. The agent's has replay memory as a component that stores past experiences as the current state, the action taken (the child's guess), the reward received (points or feedback from the parent), the subsequent state, and a flag indicating whether the episode has ended. By learning from this replay memory, the agent can benefit from past experiences, which is particularly important in an environment where some learning opportunities are infrequent or certain actions are more critical than others.

## Methods

### Q-Learning

I walked through a series of 7 experiments to optimize the agent’s policy. I started by modeling a child's learning pattern by focusing on short-term rewards and exploring the action space more quickly with less detailed attention. I then exaggerated the changes from this experiment until I found diminishing returns. I tested the effect of slower learning, given a smaller decay rate, and increased episodes to get a more robust look at the agent’s activities. I then allowed for more time for exploration and reverted to a higher learning rate, which showed success in earlier experiments. I then experimented with removing enforced inaction to see if that improved performance. Overall, I focused on balancing exploration and exploitation, as well as simulating as many real-world unpredictable outcomes as possible given my processing constraints.

### DQN

*Troubleshooting & Improvements*

1. **Increased Model Complexity**: The complexity of the neural network has been increased by using more neurons in each layer. This change allows the model to capture more complex patterns and relationships in the data, which can be particularly beneficial in environments with a large state space or more nuanced distinctions between different states and actions.
2. **Regularization**: L2 regularization has been added to the layers of the neural network. This regularization technique penalizes large weights, encouraging the model to find simpler patterns that may generalize better, thus reducing the risk of overfitting to the training data.
3. **Dropout Layers**: Dropout layers have been introduced following the dense layers. During training, dropout randomly sets a fraction of input units to 0 at each update, which helps prevent overfitting by ensuring that the network does not rely on any single neuron and is able to find more robust features.
4. **Batch Size Adjustment**: The batch size has been adjusted to 64 from 100. A smaller batch size can lead to faster convergence and can also provide a regularizing effect, as each update is noisier. However, it's small enough to maintain a good balance between the speed of convergence and stability of the learning process.
5. **Learning Rate**: The learning rate remains unchanged at 0.001, which is typically a good starting point for many deep learning tasks. However, it's important to note that the learning rate is a crucial hyperparameter that often requires tuning. The Adam optimizer is used, which is an adaptive learning rate optimizer and can mitigate some of the sensitivities to the choice of the initial learning rate.
6. **Epsilon Decay**: The epsilon decay rate was not modified and remains at 0.995. This decay rate allows the agent to gradually shift from exploring the environment to exploiting its learned knowledge. The rate is conservative enough to ensure sufficient exploration before the agent starts to exploit its policy.

# Result

## Experiments & Iterative Improvements

### Q-Learning

#### For all experiments

epsilon = 1.0

max\_epsilon = 1.0

min\_epsilon = 0.01

#### Experiment 1:

learning\_rate = 0.5

discount\_rate = 0.99

decay\_rate = 0.001

no action probability = 0.10

*Troubleshooting & Improvements*: none – first attempt

A graph and diagram of a graph

Description automatically generated with medium confidence

#### Experiment 2:

learning\_rate = 0.5

discount\_rate = 0.75

decay\_rate = 0.01

no action probability = 0.10

*Troubleshooting & Improvements*: Decrease discount rate to more accurately represent a child’s typical learning pattern. Children are more focused on short-term rewards, rather than long-term rewards. Increase decay rate for efficiency of exploring the exploration-exploitation space.

A graph of a graph and a graph of a graph

Description automatically generated

#### Experiment 3:

learning\_rate = 0.50

discount\_rate = 0.25 # how much a child cares about LT reward

decay\_rate = 0.1

no action probability = 0.10

*Troubleshooting & Improvements*: Same improvements as the previous experiment, simply more exaggerated.

A graph of a graph

Description automatically generated with medium confidence

#### Experiment 4:

learning\_rate = 0.25

discount\_rate = 0.25 # how much a child cares about LT reward

decay\_rate = 0.01

no action probability = 0.10

*Troubleshooting & Improvements*: Increase the number of episodes to get more data because the previous experiment seemed on a positive trend for total reward, decreased learning rate to experiment with slower learning given a smaller decay rate.

A group of blue graphs

Description automatically generated

#### Experiment 5:

learning\_rate = 0.4

discount\_rate = 0.25 # how much a child cares about LT reward

decay\_rate = 0.05

no action probability = 0.10

*Troubleshooting & Improvements*: the previous experiment actually did worse, so I increased the learning rate again and increased the decay rate simply for testing efficiency.

A group of blue graphs

Description automatically generated

#### Experiment 6:

learning\_rate = 0.5

discount\_rate = 0.25 # how much a child cares about LT reward

decay\_rate = 0.001 # more time to explore

no action probability = 0.10

*Troubleshooting & Improvements*: I increased my number of experiments drastically so I could get a better idea of the long-term learning potential of the model. I decreased the decay rate because previous experiments suggested that the agent may need more time to explore the environment. I increased the learning rate back up to 0.5 because that was most successful of the previous experiments.

A group of blue graphs

Description automatically generated

#### Experiment 7:

learning\_rate = 0.5

discount\_rate = 0.25 # how much a child cares about LT reward

decay\_rate = 0.001 # more time to explore

no action probability = 0

*Troubleshooting & Improvements:* This was more of a curiosity run of how the agent would perform without the more real-world no action probability in place forcing the agent to occasionally not take an action, which simulates the child not knowing a resource is available, being distracted by something in the environment, or any number of factors that would prevent a child from being able to ask for something they want.

A group of blue squares

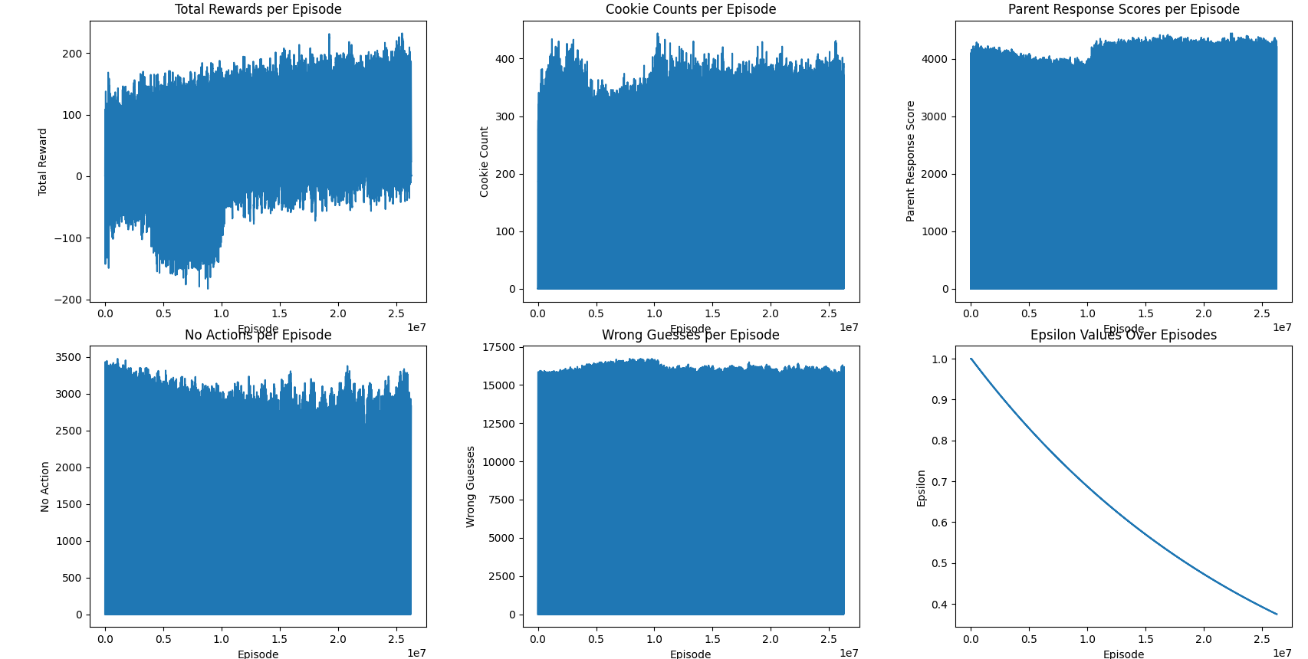
Description automatically generated

## Results Summary & Interpretation

### Q-Learning

The Q-Learning agent appears to be learning effectively, as evidenced by the general increase in total rewards and cookie counts. The stable number of wrong guesses and no actions suggest that the agent has found a strategy that works reasonably well within the environment, although there might be room for improvement in reducing errors. The decline in epsilon is consistent with the agent gaining confidence in its learned policy. The plots suggest that the agent could potentially benefit from further training or from tuning the hyperparameters to explore different strategies that might lead to fewer mistakes and a higher reward outcome. The following shows 3 stages of iterative improvements across the 7 experiments.

A group of blue graphs

Description automatically generatedA group of blue squares

Description automatically generated

# Conclusion

## Discussion

Reflection: what went well or not and why, general thoughts

## Suggestions for Improvement

* Use resources that enable more efficient and more robust processing of the data. I would have liked to do 10x the amount of episodes I was able to run in the time I had.
* For the DQN agent, I think it would have benefited from more time on hyperparameter optimization and additional layers. However, time constraints and processing power were an issue for me.

## Future Work

# Reference

Github: <https://github.com/ecraw24/EarlyLanguageSimulator>

Youtube: …

# Notes