RoboFriend Project

General description and details on algorithm implementation

Matan Weksler 302955372, Ido Glanz 302568936

1. General Description:

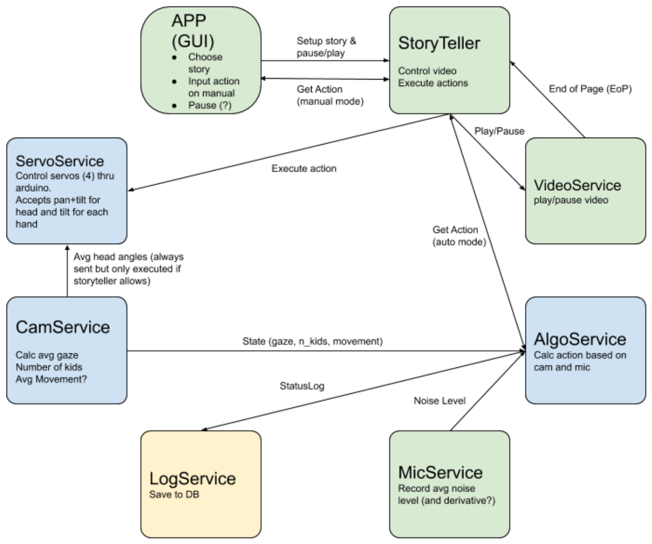
Design and implementation of a story-telling teddy-bear robot, with an RL-based algorithm for interaction control. The robot monitors (locally) the state of the children using image processing (ML gaze analysis and excitation metrics) and sound analysis, while reading through well-known children's books learning to interact with the children as to maximise their engagement and hopefully their cognitive capabilities.

2. Architecture:

The code architecture is based on our RabbitMQ ROS-like implementation of process control, allowing us to simultaneously control the physical robots servos, camera, screen etc. and run the different algorithms.

On each page, the robot can decide on its next action (continue reading, ask a question, move head etc.) leveraging an Actor-Critic RL algorithm optimising the interactions.

Servo control is done using a Python-controlled Arduino and an external camera and screen are used for capturing and displaying video.



3. Algorithm:

We implemented a policy-gradient RL algorithm with an actor-critic scheme together with an added LTL constraints module aimed to allow a way to elaborate constraints in a way the algorithm could interpret and capture in it's learning process. The general flow can be seen in the figure below.

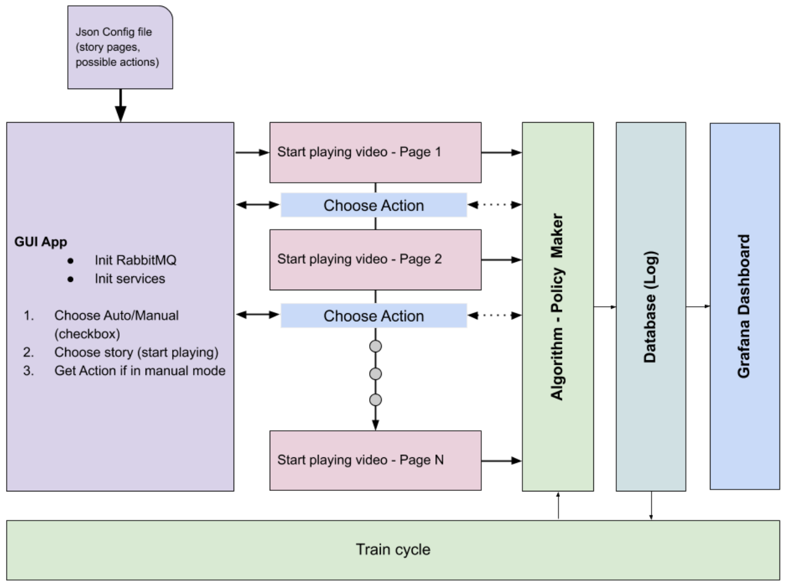
Generally speaking, the algorithm is fed after each page a state vector constructed from the accumulated metrics and calculates the next action as to maximize/optimize the engagement of the children w.r.t a rewarding mechanism accounting for both childrens state and the confinement to a given LTL (Linear Temporal Logic) formula pre-defined by the user.

The LTL statement is pre-configured per the users best understanding of the task and a general sense of creating “fairness” and “rationality” of the problem and to enable the algorithm to better capture those understandings without the need to learn them entirely from scratch. An example of such a constraint could be “Eventually(play next page)” or in ltl syntax: “F(play_next_page)” to avoid a long loop of repetitive actions such as asking a question or moving hands/head causing to eventually lose the children's attention. Should be noted that these constraints are not strictly limiting the robots actions (i.e. blocking it from asking 3 questions in a row) but are integrated as part of his reward mechanism so it would learn these but in a more explicit manner then just through observing the general state of the children.

The LTL is evaluated using a DFA (Deterministic Finite Automata) which is fed a state-trace with a time-sequence vector of previous actions which is also a part of the state vector fed to the policy function (thus the state space is extended per the LTL statements needs and changing the LTL would/could consequent in the need to re-train the algorithm), if the state-sequence confines to the LTL statement the robot is rewarded accordingly and penalised otherwise.

As the nature of this work/algorithm doesn’t allow the easy collection of data and each episode has to be run with actual groups of children, and while fearing the lack of pre training before actually letting it engage with children, we added an imitation learning option (a manual mode) allowing for a human supervisor to manually choose the next action, thus replace the algorithm inference.

That said, while playing, the algorithm is still running in the background and is learning the humans’ choices in a supervised manner, i.e. viewing the supervisors choice as a ground truth/label. This way, the first few trials of the robot could be manually run while the algorithm is learning in the background and potentially would have some prior knowledge/intuition when moving back to auto (algorithm) mode.


Json Config file (story pages, possible actions)
GUI App
Init RabbitMQ
Init services
1. Choose Auto/Manual (checkbox)
2. Choose story (start playing)
3. Get Action if in manual mode
Start playing video - Page 1
Choose Action
Start playing video - Page 2
Choose Action
Start playing video - Page N
Algorithm - Policy Maker
Database (Log)
Grafana Dashboard
Train cycle