Overview of Delivered Product

An Important Notice Regarding Delivered Product

In order to access the ChatGPT API, the personal OpenAI API key of a developer who worked on this project has been used. This key will be revoked at the end of June 2023, so in order to have continued access to ChatGPT functionality, this key will need to be replaced. The file which makes use of this key is NA-Boxjelly/src/cowriter_letter_learning/letter_learning_interaction/include/nao_settings.py.

Delivered Features and Functionality

The Port

The team has effectively transitioned the codebase from Python 2.7 to Python 3.8.10 and migrated the ROS framework from the Indigo version to Noetic.

All *.py files relating to CoWriter functionality were meticulously updated to ensure compatibility with Python 3.8.10. Now, the CoWriter program can leverage the latest language features and optimizations that Python 3.8.10 offers, ensuring a better performance and overall experience.

Furthermore, relevant files, such as CMakeLists.txt, package.xml, launch files, among others, have been adapted to function seamlessly with ROS Noetic. This change has significantly improved the CoWriter program's capabilities, compatibility with newer hardware, and access to the latest ROS packages and tools, providing an enhanced user experience.

The team also took the opportunity during the porting process to improve the quality of the existing codebase. Bad practices used in the old code, such as the use of global variables, were identified and remedied, and detailed documentation was created to facilitate future code understanding and maintenance.

Teaching Nao to Write

The CoWriter Letter Learning program's unique educational approach involves teaching the Nao robot to write, thereby actively engaging children in the learning process. We have implemented this feature in two ways, both designed to maximize the interactive and collaborative learning experience.

The first component is a demonstration-based learning mechanism. In this approach, children can demonstrate how to write letters to the robot using a tablet or touch screen device. The robot is capable of observing and interpreting these demonstrations, adjusting its own writing in response to the child's input. This interactive feature makes the learning process more engaging and fun, encouraging children to actively participate in teaching the robot and, in turn, improving their own handwriting skills.

The second component of teaching the robot to write involves sending specific words to the robot for it to write. This can be done through the user interface, where the child or the educator can input the desired word in text form. Upon receiving the word, the robot will then proceed to write it out. While the robot does not currently learn from this component of the interaction, it serves as an effective tool for demonstrating correct letter formation and providing a visual reference for the child. The child can then compare their own writing to the robot's demonstration, promoting learning through observation and comparison.

Integration of ChatGPT

A significant feature added to the program is the integration of OpenAI's ChatGPT API. As the new dialogue manager for the Nao robot, ChatGPT allows for highly engaging and fluid conversations, greatly enhancing the user experience and learning effectiveness. To capture the child's verbal interaction and input it into the ChatGPT, we have used the Whisper speech-to-text technology. Developed by OpenAI, Whisper accurately transcribes the child's spoken words into written text, providing a reliable input mechanism for the ChatGPT. This transcription process plays a vital role in creating a seamless interaction flow, allowing for real-time conversation between the child and the robot. This interaction is facilitated by enabling microphone input from the device. This feature allows the child to naturally communicate with the robot, as if conversing with a human companion, thereby enhancing the user's engagement and immersion in the learning process.

The way the interaction works with ChatGPT can be broken down into the following step:

- The microphone on the device records speech from the child or educator
- This audio is transcribed to text using OpenAI's Whisper model
- The transcribed audio is sent to ChatGPT as a prompt, via the OpenAI API
- The returned output from ChatGPT is used in Nao's text-to-speech, so that Nao will verbalise the response from ChatGPT

New and Improved User Interface

As part of our efforts to enhance the user experience of the CoWriter program, we have designed and implemented a new User Interface (UI) using PyQt, a popular Python binding for the Qt libraries often used for creating GUI applications.

The new UI incorporates a number of interactive features that greatly improve the engagement and learning experience for children:

- **Real-time Letter Drawing**: The child can draw letters directly on the UI, which are then sent to the robot. The UI has been designed to be highly responsive, showing the child's writing in real-time as though written with a pen. This feature brings an immediate and tactile dimension to the learning process, making it more engaging and effective.
- **Letter Emulation Display**: To further enhance the learning experience, the UI also displays the robot's attempt at emulating the child's writing. This visual feedback gives the child the opportunity to observe, correct, and learn from their writing in a highly interactive manner.
- **Direct Word Sending**: The UI provides the functionality for the child or the educator to send words directly to Nao. Upon receiving these words, Nao will write them out, serving as a powerful tool for demonstrating correct letter and word formation.
- **Talk Button for Conversation**: A notable feature is the ability for the child to press a button to talk to the robot. The spoken input is then transcribed and processed through ChatGPT, which generates a response. This response is then spoken by Nao, facilitating a natural and engaging conversation with the child. This interaction brings a new level of dynamism and personalisation to the learning experience, further enhancing the program's effectiveness as an educational tool.

The introduction of this new PyQt-based UI represents a significant step in the evolution of the CoWriter program, creating a more interactive and immersive learning environment for children.

Extensive Code Documentation and Testing

The original code had very little documentation, and no unit tests. This made the port quite challenging, as coming to grips with the functionality and purpose of individual components was a time consuming task. As this project will no doubt be extended in future, we sought to make the lives of future developers much easier by extensively documenting the code. We have endeavoured to provide docstrings for every significant class, method, and function, so that future developers may quickly come to grips with their purpose and how to make use of them.

In addition to the docstrings, we have developed a comprehensive suite of unit tests. The purpose of these tests is two-fold:

1. they ensure that the delivered functionality is robust and performs as expected;
2. the unit tests themselves can serve as additional documentation for classes and functions, demonstrated their expected behaviour.

Extensive Installation and Set Up Guide

We have provide extensive instructions on how to install and set up the required software, and launch the program. Both written and video instructions have been provided. Additionally, there is an automated set up bash script which can be ran to set up the packages and dependencies automatically, once Ubuntu have been installed.

Relating Functionality to Product Backlog

The functionality outlined above corresponds to the delivery of the following user stories (please consult the product backlog for more detail on the user stories):

- US01
- US02
- US03
- US04
- US05
- US06
- US10
- US17
- US18
- US19
- US20
- US21
- US22

Future Directions for CoWriter Development

Here we note some possible future directions for development or extension of the current software. These directions are mostly centered around lower priority user stories which we didn't have time to deliver.

A Bulleted List of Possible Directions

- Have Nao robot able to suggest words for children to practice; this could be based on their interests, and/or their current writing abilities
- Allow Nao robot to adopt multiple personas, targeted at different age groups
- Implement a method which allows Nao robot to distinguish educator speech from child speech
- Implement a more graceful termination of program (currently it shuts down without any problems with the Nao robot, but the ROS terminal will display some error messages)
- Implement ability for Nao robot to learn from full word demonstrations, rather than just individual letters

Relating Future Directions to Product Backlog

In some cases, the directions noted above relate to lower priority user stories which we did not have time to implement. The relevant user stories are as follows (please consult the product backlog for more detail on the user stories):

- US11
- US12
- US13
- US14
- US15
- US16