For my chatbot, I chose the topic of military conflicts. This was with the intention of having a chat bot that could answer high level questions about a variety of modern conflicts, such as the second world war or the Vietnam war. The user is also able to request the chatbot to identify military boats from civilian boats.

This was somewhat an ambitious topic, mainly due to the wide variety of subtopics included. One thing that resulted in what was essentially a “Pick n Mix” of subtopics instead of a coherent set was the difficulty of finding appropriate data sets. For example, the original plan for image classification was to identify different modern weapons, however the only suitable military based image dataset I could find was related to the navy instead.

The rule-based component of the chat bot largely went smoothly and according to plan. This means that the chat bot used the AIML language within an XML file to help choose an appropriate response to a given input. This was implemented through the use of the patterns mentioned in the original planning document with no issues.

The similarity component of the chat bot also went smoothly. To compensate for the very limited uses of a rule-based component, excluding the possibility of hard coding in hundreds or thousands of rules, a similarity based component was implemented. This component used a series of question/answer pairs, a bag of words model and TF-IDF values to calculate the cosine similarity of a given input. This expanded the capability of the chat bot by a reasonable amount, but eventually still succumbed to the same issue of being excessive and tedious to program all possibilities of user input.

The image classification component was relatively simply to implement. The main challenge faced during implementation was finding a suitable method to actually feed the user’s image into the chat bot client itself. This was eventually solved through the use of requesting a file name from the user. The chatbot would then seek an image with the entered name and use the CV2 library to convert the image to a vector. This vector could then be fed through a trained Keras neural network model with pretrained weights in order to classify the type of ship. Another issue faced with this task was to find a suitable dataset that actually had a sufficient amount of images for both training and testing. This meant that the original idea of classifying military weapons had to be scrapped in favour of identifying if a vessel was a military vessel etc.

The Toy world reasoning system was used to further enhance the discussion capabilities of the chat bot. This means that through the use of a toy world reasoning system, the chat bot could now process user input in such a way that it could answer further questions based on previous statements entered. An example of this would be the user telling the chatbot that X battle took place in Y country. The chatbot would then be able to answer such questions as “What battles took place in a particular country” or if a specific battle took place in a specific country etc. Despite the advanced capabilities of this approach, it still falls victim to the same issues as previous NPL methods – The inability to cover every possible input for such a large and open topic.

A sequence to sequence neural network was attempted. The idea behind using this approach was to provide a fallback option for any time the bot was unable to determine an input using other NPL methods. Instead of using this method, implementation of a transformer network was attempted. This never materialised, however. This was due to minute differences between the system used to train the model and the machines the model would run on. As a result, the trained weights of the model were not compatible with the actual model unless ran on the original system (Google colabs). Had the weights proven to be compatible with the model, it would have meant that the chatbot would be able to use a transformer network trained across a large dataset of conversations to further determine a response if no other method could succeed. This would have massively extended the capabilities of the chatbot, and potentially solved the key issue of covering every possible form of input if successfully implemented.

For the reinforcement learning, there was no suitable task that could be implemented within the chosen chatbot topic/theme. As a result, the chatbot was trained to play a simple game. This game was provided through the OpenAI Gym library and was trained using Deep Q Learning. This meant that once the model created was loaded into the chatbot, when the user requested to play a game with the chatbot, the game would load, and the computer would play a solved version of the game.

One last thing to address with the project is the diagram included with the original design document. This diagram was limited in scope and would have been much larger once all chatbot components were integrated. As it currently stands, the diagram essentially only shows the process flow for the tasks of rule based and similarity-based input handling. Despite the larger size of the diagram, the complexity of the diagram would largely remain the same. This is because only two additional main features were added – A trained game and image classification. The other components implemented were designed to further enhance the capabilities already shown within the diagram, and thus would have just been a few extra boxes as an extension.