**iCyber  
Artificially intelligent simple chat bot**

**Description:**

iCyber is an artificial intelligence (AI) chat bot that uses neural networks and Natural Language Processing (NLP) to analyse text inputted from the user and response accordingly. The bot is written in Python and uses the PyTorch open-source machine learning library developed by Facebook's AI Research lab (FAIR). The program utilises the NLP basics such as tokenisation, stemming, and bag of words techniques.

**Objective:**

The aim of the chat bot is to provide users with the latest information on cyber security. This can range from the user asking what the newest threats are to be aware of, top news of the day, or the user asking about specific elements of computer security (malware, viruses, phishing). By using AI instead of the user simply Googling questions the intent is that they will receive more relevant information with a personal touch. Since the bot is operating within a closed domain all responses are more tailored for the subject rather than the generalist results search engines will return. The intent – as with most chat bots, is to also provide the user with the experience of asking a human being but without all the baggage of social interaction.

**Software tools used:**

* Visual Studio Code
* Python 3.9.9
  + PyTorch
  + NLTK (Natural Language Toolkit)
  + NumPy
* Command prompt

**How to run the application**

1. Open command prompt or terminal
2. Navigate to the ‘hand-in’ folder which contains all the Python files  
   Example: cd *“C:\Users\chris\Dropbox\Documents\Master of Software Development\SWEN 504\Week 23 (10-17 January)\Hand-in”*
3. Run ‘train.py’  
   Example: *py train.py*
4. Wait for the program to train itself on the dataset set. Once finished ‘*Training complete. File saved to data.pth’* will display.
5. Run ‘chat.py’  
   Example: *py chat.py*
6. The chatbot will display a welcome message with instructions. You can now start talking to the bot. Type ‘quit’ to exit the program.

**Text

Description automatically generatedHow to test the application**

Figure 1 – tokenisation  
Creating a print statement that will print out the tokenised version of the original sentence. It will have correctly worked when each word or symbol has been tokenised as pictured.

Graphical user interface, text

Description automatically generated

Figure 2 – stemming  
Putting variations of a word into an object and printing it to see they can be correct processed. In this case, organis is the stemmed version of organise. But I realised this would have been developed using American English, so I needed to retest.

*Text

Description automatically generated*

Figure 3 – stemming  
Retesting with American version of organise (organize). Returns expected result of organ.

*Text

Description automatically generated*

Figure 4 – all words tokenised  
Printing all the words in the all\_words array to see if they have been tokenised properly.

*Text

Description automatically generated*

Figure 5 – training the AI on the data  
Running the train.py program to build the neural network on the dataset. Using GPU acceleration.

*Text

Description automatically generated*

Figure 6 – conversing with the chatbot   
Using known phrases and sentences to interact with the bot. Using the Wikipedia search feature to test results.