# ISYS30221 Artificial Intelligence 2021-22

## Coursework Documentation Template

## 1- About this submission

|  |  |
| --- | --- |
| Student Name | Kypros Tsolakis |
| Student ID | N0950765 |
| Chatbot Topic | Video Games |
| Tasks implemented in this submission (a,b,c,or d) | The tasks that I have implemented are tasks A (and extra functionality), B and C. |
| Files inventory (excluding this file) | All the files that I used are in the zip file.  mybot-basic.py (Part A)  mybot-basic.xml (Part B)  game\_topic\_qa.csv (Part A)  kb.csv (Part B)  game\_classifier.h5 (Part C)  runPhotos (Testing photos) (Part C)  Main.py (Main program) (Part C)  modelTrain (Train the model) (Part C) |
| Demo video URL | https://web.microsoftstream.com/video/e7206e6b-b245-4546-abd4-027e6fd64a98 |
| Checklist | I will submit this file separately (without compression) into DropBox  All other files are zipped and will be submitted into DropBox  The demo video is recorded as instructed, and the sharing link is inserted above  I have made sure that the demo video is shared according to the instructions, so that I allowed everybody in the university to view it.  All the sections below are populated accordingly. |

## 2- Design notes (shrink/grow as needed, add images where applicable)

|  |  |
| --- | --- |
| General explanations of the system and its goals | As all we know everybody occasionally uses the internet to look up specialised information on a particular subject. We have used a programme that accomplishes just that in this coursework. Our chatbot is about video games and features several League of Legends champions.  Part A: The code generates a basic chatbot system that can accept text or voice input from the user and respond based on knowledge found in an AIML XML file, a CSV file, or Wikipedia. In order to discover the best match, the system computes cosine similarity scores between the user's input and the stored data using natural language processing (NLP) techniques.  The system's key objectives are to respond to user questions and enquiries in a conversational manner about a certain subject (presumably related to games, based on the CSV file name used in the code). The system uses NLP approaches to comprehend user input and match it with pertinent data in order to improve the user experience by giving accurate and helpful responses.  Part B: The chatbot does inference and reasoning on a knowledge base (KB) of logical expressions, which is initialised from a CSV file, using the Natural Language Toolkit (NLTK) module. The knowledge base (KB) has information and guidelines that the chatbot can utilise to analyse user input and deliver suitable responses. Based on the user's input, the chatbot employs Resolution Prover from the NLTK inference module to verify consistency and infer new facts.  The chatbot uses Artificial Intelligence Markup Language (AIML) in addition to the knowledge base to offer a more conversational interface. A markup language called AIML is used to build chatbots and virtual assistants. It includes a number of pre-made templates and patterns for typical conversational situations.  Also adds two more parts for dealing with logical queries. When the user enters a statement with the format "I know that \* is \*," where \* is a wildcard, the first is activated. The chatbot will separate the statement's object and subject, turn it into a logical expression, and then determine whether it conflicts with any already-existing knowledge in the KB. The chatbot will include the new information in the KB if it does not conflict with any already published facts.  When a statement with the format "verify that \* is \*" is entered by the user, the second logical query component is triggered. The chatbot will separate the statement's object and subject, turn it into a logical expression, and then determine whether it can be inferred from the information already in the KB. The chatbot will answer with "Correct" if it can be deduced. The chatbot will reply with "Incorrect" if it conflicts with other known facts. The chatbot will say "Sorry, I don't know" if it is unable to identify whether the assertion is true or untrue.  Part C: Convolutional neural networks (CNNs) are used in this machine learning system to categorise photos of two different League of Legends characters. The algorithm can accurately recognise new photographs of Lee Sin, Garen, and Ahri because it was trained on a dataset of these characters' images. When given an image file location, the system loads and preprocesses the image, predicts the character in the image using the trained CNN model, and then returns the character's name. Also, the system has the capacity to process fresh images constantly until the user closes the application and convert PNG images to JPG format. The system includes code to shuffle the training data and divide it into training and testing sets in addition to the capability mentioned above. Also, it contains code for assembling the CNN model using the Adam optimizer and SparseCategoricalCrossentropy loss function, as well as code for normalising the picture input. When a classification is required, the system loads the trained model back into memory from a file it has previously saved. In general, this system is an excellent illustration of how to use CNNs and machine learning to perform picture categorization tasks. |
| The system requirements, i.e., the list of what the system should do/have from a user’s perspective | Part A:   1. Enable the user to select between text and voice input. 2. Respond to user questions regarding games with information that is truthful and pertinent. 3. Identifying and answering a request to leave the chatbot system. 4. When voice recognition fails or no data is present in the AIML, CSV, or Wikipedia sources, handle the error gently. 5. Input and output interfaces should be user-friendly and intuitive, with clear instructions and feedback messages for the user. 6. Maintain the confidentiality and security of user data, including recordings of voice input, and refrain from storing or transmitting such data without the user's permission.   Part B:   1. Conversational response to user input: The chatbot should be able to have a natural conversation with users. 2. Greeting the user and replying to typical queries or requests are just two fundamental conversation scenarios that the chatbot should be able to handle. 3. Handle logical queries: Based on user input, the chatbot should be able to handle logical questions and deduce new information. On the basis of the information already contained in the knowledge base, it ought to be able to determine if a particular assertion is true or incorrect. 4. Feedback: The chatbot should be able to let the user know whether a particular statement is true or incorrect or whether a request has been acknowledged. 5. Handle unexpected input: The chatbot should be able to respond to unexpected user input by requesting the user to rephrase their input or sending a message indicating that it did not understand what they had said.   Part C:   1. Prepare photos for training and classification by loading them. 2. The training data should be mixed up and divided into training and testing sets. 3. Image data should be normalised. 4. Convolutional neural network (CNN) models can be trained to divide images into many categories. 5. When a classification is required, save the trained model to a file and load it back into memory. 6. fresh photos can be classified by entering their file locations and seeing the suggested class name. 7. Ask the user for new image file paths repeatedly until they decide to quit the programme.   The system should also meet the following specifications:   1. Use of the OpenCV and TensorFlow libraries for machine learning and image processing tasks. 2. a collection of photos with their corresponding classes labelled for the purpose of training the CNN model. 3. Adequate computing power and memory to handle huge image datasets and train the CNN model. |
| The employed AI techniques, and the explanation of program codes and the supplied files. | Part A: AI techniques:   1. Natural Language Processing (NLP): The application preprocesses user input and AIML patterns by carrying out operations such tokenization, lemmatization, and text data cleaning using the Natural Language Toolkit (NLTK) module. 2. Information Retrieval: The software converts the preprocessed text data into a numerical representation that can be used to determine the cosine similarity between the user input and data from other sources using the TF-IDF vectorizer from the scikit-learn module. 3. Cosine Similarity: The application employs cosine similarity to assess how closely user input and text data from several sources, including an AIML file, a CSV file containing question-and-answer pairs, and Wikipedia articles, compare to each other. 4. Voice Recognition: To transform user speech input to text, the software makes use of the SpeechRecognition library. 5. Text-to-Speech: The application converts text responses into speech output using the Pyttsx3 library.   The programme is divided up into numerous sections, each of which handles a certain task:   1. Opening an AIML XML file: The programme opens an AIML file that contains predefined chatbot replies and patterns. 2. The software defines an NLTK lemmatizer to preprocess text input and a scikit-learn TF-IDF vectorizer to convert the preprocessed data into a numerical representation. 3. Recognizer and Text-to-Speech Engine Creation: To recognise user speech input and turn text responses into speech output, the software creates instances of the SpeechRecognition and Pyttsx3 libraries, respectively. 4. User Input: The programme asks the user to select either voice input or text input mode before waiting for their response. 5. Processing User Input: After tokenizing, lemmatizing, and cleaning the text data, the programme preprocesses user input before calculating the cosine similarity between it and patterns in the AIML file. 6. In response to user input, the programme obtains the relevant response and converts it to speech output if a match is discovered in the AIML file. If no match is discovered, the computer looks for a matching question and retrieves the related response in a CSV file that contains question-answer pairings. If still no match is discovered, the computer looks through Wikipedia pages for a relevant topic and gets the opening paragraph. A default message is returned by the software if no match is discovered in any source.   Part B: AI techniques:   1. Natural Language Processing (NLP): The code analyses user input to produce responses using NLP techniques. To be more precise, it defines patterns for recognising user input and producing suitable responses using the AIML (Artificial Intelligence Markup Language) library. 2. Semantic Reasoning: Based on user input and already-existing facts in the knowledge base, the code use semantic reasoning to infer new facts. It accomplishes this by generating logical expressions from user input and determining whether or not they can be supported by the data in the knowledge base. A new fact is added to the knowledge base if it can be deduced. 3. Using the information already present in the knowledge base, the code employs automated reasoning to determine whether a particular statement is true or incorrect. It accomplishes this by utilising a resolution theorem prover to try to prove the assertion or its negation.   The programme is broken up into numerous sections, each of which handles a certain task:   1. The knowledge base's initial set of facts are contained in the CSV file kb.csv. The Expression.fromstring() method from the NLTK package is used to transform each row's contents, which correspond to individual facts, into logical expressions. 2. mybot-logic.xml: This AIML file specifies processing patterns for logical inquiries. It has preset responses for phrases like "I know that \* is \*" and "verify that \* is \*," among others. The necessary object and subject are extracted from the input by the templates using placeholders, which are then used to produce logical expressions and carry out reasoning. 3. The primary software file, main.py, sets up the chatbot and manages user input. It parses user input, produces replies, and engages in logical reasoning using the NLTK and AIML libraries. The while loop in the code is responsible for reading user input from the command line and sending it to the chatbot for processing. 4. Throughout the code, there are comments that describe what each block of code is doing.   Part C: A Convolutional Neural Network (CNN) model is being implemented in the code to divide images into two groups. The application uses TensorFlow and Keras libraries to build and train the model. The Lee Sin, Garen, and Ahri characters from video games are represented by images in the training dataset. The programme loads the photographs, shapes them to a certain size, normalises them, and divides them into training and testing sets. Then, using the training set of data, the CNN model is constructed, defined, and trained.  Following training, the model is saved to a file, and then later loaded to categorise fresh photos. The model is used to classify an image into one of the two categories once the user is queried for the path to an image. The application uses the os library to manage file and directory operations and the cv2 library to load and preprocess the photos.  In order to make sure that any photos that are not in JPEG format are converted to JPEG format and made compatible with the software, the code additionally includes some image conversion processes. Overall, the programme classifies photos using deep learning methods and libraries, showcasing the effectiveness of AI for image identification applications. If we have more photos the accuracy is going to be better. |

3- Conversation log (insert text, screenshots and/or images as required)

|  |
| --- |
| Part 1:          Part B:    Part B with inconsistencies in the KB    Part C: |

(no word count is necessary)