UNIVERSITY OF LONDON

BSc Computer Science

CM2015 Programming with Data

Midterm coursework assignment

This coursework is worth 50 marks.

Chatbot Project

Now that you have had a chance to explore some techniques and tools in Python, it is time to start integrating them into your own chatbot project. This is a chance for you to build a practical application using your knowledge of Python and Data Programming.

Expectations

- Develop a functional and interactive chatbot without errors.
- Demonstrate a strong use of core Python concepts, including:
    - Data structures (dictionaries, lists, tuples) to managing intents, patterns, and responses.
    - Using Conditional logic and loops appropriately to drive chatbot interaction.
    - Implementing functional and modular programs breaking logic into clear, reusable functions with well-defined inputs and outputs.
    - Organising code into multiple files (code + data) or modules to enhance maintainability and readability.
- Adopt software engineering best practices by keeping code modular and reusable using functions, classes (optional), and configuration files (e.g., JSON).
- Write test case**s** to verify and highlight the chatbot's functionality and robustness.
- Include clear and consistent documentation using comments.
- Utilise Data Processing Techniques to clean and tokenise input data (e.g., user queries or pattern sets).
- Uses libraries like NLTK, re (regex) for pattern matching, and tokenisation.
- Implements basic NLP features, such as stop word removal, stemming/lemmatisation, or part-of-speech tagging.

- (Optional) Incorporates sentiment analysis, keyword extraction, or named entity recognition for more intelligent responses.

Submission requirements

For the midterm coursework, you will submit the following documents on the submission page:

- A shareable link to the Jupyter notebook environment that has
  - A single Jupyter notebook file with chatbot demo (ipynb file)
  - Supplementary dataset (intents.json file). The dataset (intents.json) should not be more than 10MB in total size.
  - A PDF report
- Project ZIP file
  - ZIP file with all the files in the Jupyter notebook environment (ipynb, intents.json, PDF).
- An exported HTML file
  - Export the Jupyter notebook code into a HTML file and submit it.
- PDF report
  - A copy of the PDF report.

Marking rubric

The marking rubric includes a description of expectations and deliverables. Sections and corresponding marks given below.

Sub part	Criteria	Marks awarded	Mark breakdown
1	Title/Domain of chatbot	1	Provide the title of chatbot in the Jupyter notebook
2	Main loop	3	Chatbot uses a main loop that takes in user input and terminates only when the user types “exit” or “quit”.
3	Data Structures	3	The data structures used to manage intents, patterns, and responses are compact and involve the use of Python dictionaries.
4	Code organisation	5	The different chatbot components are specified as functions that are called from the main loop or other associated functions. This includes: • Function to load intents from JSON files • Function to load and search using regex patterns • Function to generate responses. Correct Interaction between main loop and functions (argument passing and return calls).
5	Pattern recognition	5	The chatbot must use pattern recognition incorporating regular expression-based matching. Students are expected to:

			• Write flexible pattern matching statements to recognise multiple instances of the same intent type. • Utilise basic regex constructs (e.g., \d, \w, ., *, +, ?) for robust pattern handling. • Build regex patterns that account for variations in user input (e.g., case insensitivity, optional words). • (Optionally) Implement advanced regex features like grouping, or lookaheads for more nuanced understanding.
6	Response generation	5	The chatbot must demonstrate diversity in response generation. The basic functionality involves retrieving one response for one input. Implement these additional techniques for more diverse responses: • Choose responses randomly from a list of options. • Applying dynamic strings substitutions with a memory (e.g., store user's name or colour or some personal information in an object and substitute it in a response). • Combination of techniques mentioned above.
7	File usage	5	To promote modularity, reusability, and scalability, all chatbot data including intents, patterns, and responses must be stored and loaded from external JSON files. Key expectations include: • Each intent should contain a list of regex-based patterns stored in the JSON file. • Responses should be written as template strings in the JSON file. These templates can include placeholders (e.g., {name}, {color}) that are dynamically filled at runtime using string substitution based on user input or stored memory. • The chatbot should implement a function that loads the JSON file at runtime**,** extracts patterns and responses, and builds internal structures such as pattern2intent and intent2response dictionaries.
8	Preprocessing	5	Expectations regarding text preprocessing techniques include: • Splitting user input into individual words or tokens to analyse structure and meaning more easily. • Eliminating characters like commas, or periods that are not essential for intent detection. • Reducing words to their root form to match patterns more broadly and accurately.

9	Other Advanced features	3	Any other advanced feature that you have added and features that you have added beyond lecture material.
10	Process reflection	5	Discuss the week-by-week iterative development of your chatbot. What was the feedback you received? How did you work on the feedback to improve chatbot? What new features did you add?
11	Report	5	Report should cover the main aspects of chatbot such as: - Chatbot application (e.g., use-case, domain of operation). - Describe 3 different test cases that clearly illustrate chatbot behaviour. - Any other steps to organise data/code that you implemented. - Describe advanced techniques that you used.
12	Code	5	Code should: - Be reproducible in the current notebook format including making relevant data sources and libraries accessible and explicit. - Use proper conventions e.g. relative path vs absolute. - Be explained or described where libraries are used in relation to their utility/ability to solve a particular problem in an efficient manner. - Notebooks should be structured with a logical set of processes/procedures including clear, logical headings. - Not overly verbose e.g. including comments to describe print statements.

[END OF COURSEWORK ASSIGNMENT]