CS7CS4/CSU44061 – Machine Learning
Academic Year 2025-2026
Trinity College Dublin

Final assignment 2025-26

- You must do this assignment entirely yourself - you must not discuss or collaborate on the assignment with other students in any way, you must write answers in your own words and write code entirely yourself. If you use any online or other external content in your report you should take care to cite the source. It is mandatory to complete the declaration that the work is entirely your own and you have not collaborated with anyone - the declaration form is available on Blackboard. All submissions will be checked for plagiarism.
- Include the source of code written for the assignment as an appendix in your submitted pdf report. Also include a separate zip file containing the executable code and any data files needed. Programs should be running code written in Python, and should load data etc. when run so that we can unzip your submission and just directly run it to check that it works. Keep code brief and clean with meaningful variable names etc.
- Important: For each problem, your primary aim is to articulate that you understand what you're doing - not just running a program and quoting numbers it outputs. Generally most of the credit is given for the explanation/analysis as opposed to the code/numerical answer.
- If you use machine learning models not covered in the course then you must take care to show that you understand them and are not just running code in a "black box" fashion (so explain how predictions are generated from an input, what the cost function is, what the model parameters and hyperparameters are and how they affect the predictions etc).
- Reports should typically be about 5 pages, with 8 pages the upper limit (excluding appendix with code).
- The **submission** on Blackboard should include: **(A)** the report in PDF format, **(B)** the final model (only one) for each part of the assignment (please find the instructions for saving and loading the stored model below), **(C)** an appendix with a brief selection of prompt-output pairs for each of the two models, demonstrating their strengths and weaknesses, **(D)** the final Python code. Note that the PDF must be uploaded as a separate file (not as part of a zip file).
- Instructions to save and load the models:
  - **Save:**
    torch.save(model1.state_dict(), "model_weights_part1.pth") # Please use these filenames
    torch.save(model2.state_dict(), "model_weights_part2.pth")
    # note that model1 and model2 are instances of the GPTLanguageModel class
  - **Load:**
    model = GPTLanguageModel()
    model.load_state_dict(torch.load("model_weights_part1.pth"))
    model.eval()

General summary

In this assignment, you will repurpose the GPT implementation covered in class to solve symbolic tasks like arithmetic and Boolean logic, exploring architectural and algorithmic adaptations that best suit those specific applications. This assignment will involve:

- Generating curated datasets for symbolic reasoning tasks;
- Training a basic GPT implementation on those datasets;
- Identifying appropriate evaluation metrics;
- Experimenting with architectural modifications (e.g., tokeniser, embedding size, cost function) and analyse their impact.

The report should include clear, legible figures with appropriate figure numbers and concise captions. Please note that the primary basis for your grade will be your critical thinking, methodological rigor, and approach to addressing challenges. It is preferable to develop a simple, well-functioning model rather than attempting an overly complex one that performs poorly.

Part 1 — Math GPT (46 marks)
Build a transformer model capable of solving simple arithmetic expressions. We suggest starting with a very simple set of operations (e.g., single-digit arithmetic, sum), evaluate if the model is producing correct outputs, and then extend to more complex operations if possible.
Example inputs:

- *3+2=5*
- *47+38=85*
- *(3+2)*4=20*
- *12-5=7*
- *20/4=5*

Tasks:

1.1. Build appropriate dataset(s) for training and testing your model (8 marks)
1.2. Define appropriate evaluation metrics (8 marks)
1.3. Explore adaptations of the model architecture to make the model optimal for this specific application. You are expected to motivate your choices, including your reasoning and numerical comparisons between different choices. Architectural changes may involve, for example: vocabulary and tokeniser, embedding size, number of heads, number of layers, block_size, cost function (15 marks)
1.4. Explore a range of arithmetic operations and discuss what operations are learnt correctly, and which ones are not, providing both quantitative evidence (e.g., error rate) and representative example of prompts with the corresponding outputs (15 marks)

Part 2 — Boolean Logic GPT (46 marks)
Build a transformer model that evaluates Boolean expressions.
Example inputs:

- True AND False = False
- NOT True = False
- (True OR False) AND True = True
- True XOR True = False

Tasks:

2.1. Build appropriate dataset(s) for training and testing your model (8 marks)
2.2. Define appropriate evaluation metrics (8 marks)
2.3. Explore adaptations of the model architecture to make the model optimal for this specific application. You are expected to motivate your choices, including your reasoning and numerical comparisons between different choices. Architectural changes may involve, for example: vocabulary and tokeniser, embedding size, number of heads, number of layers, block_size, cost function (15 marks)
2.4. Explore a range of arithmetic operations and discuss what operations are learnt correctly, and which ones are not, providing both quantitative evidence (e.g., error rate) and representative example of prompts with the corresponding outputs (15 marks)

Part 3 — Discussion (8 marks)
Task 3.1. Write a critical analysis comparing the Math GPT and Boolean GPT parts. Identify which architectural elements (e.g., number of layers, cost function, activation functions, regularisation techniques, embedding size) were optimal for both tasks and which required adaptation. Discuss the similarities and differences in these design choices, and what these imply about the nature of the tasks. Reflect on how these choices impacted the model evaluation (keeping in mind that different evaluation metrics may be employed), generalisability, or computational cost. Discuss how these models fail, what solutions mitigated those issues, what issues remain unsolved, and what actions could be taken to that end. Include any other relevant insights drawn from this comparison.