Assignment 4: LLMs and Machine Learning

B551, Elements of Artificial Intelligence, 2024

Optional Assignment. Due Date: 11:59pm, Friday, December 13.
Note: The usual late penalty for this assignment is waived, for submission through 11:59pm 12/15 without penalty. No later submissions can be accepted.

This assignment will give you experience prompting Large Language Models for well defined tasks, in the context of a case study of LLM performance on an adversarial problem you have already addressed using a traditional AI model, for comparison. It also provides experience implementing and applying the K-Nearest Neighbor (K-NN) algorithm for machine learning.

This assignment is **optional** and provides an opportunity to increase your homework average. If you do this assignment your lowest homework assignment percentage will be replaced by the percentage on this, if that increases your average.

Guidelines for this assignment

Coding requirements. For fairness and efficiency, we use an automatic program to grade your submissions. This means you must write your code carefully so that our program can run your code and understand its output properly. In particular:

1. You must code this assignment in Python 3, not Python 2.
2. Make sure to use the program file name we specify.
3. Use the skeleton code we provide, and follow the instructions in the skeleton code (e.g., to not change the parameters of some functions). Your code must obey the input and output specifications given below.

Groups. You'll work in a group of 1-3 people for this assignment according to your preferences. You should only submit one copy of the assignment for your team, through IU GitHub. All the people on the team will receive the same grade, except in unusual circumstances.

Coding style and documentation. We will not explicitly grade coding style, but it's important that you write your code in a way that we can easily understand it. Please use descriptive variable and function names, and use comments when needed to help us understand code that is not obvious. For each of these problems, you will face some design decisions along the way. Your primary goal is to write clear code that finds the correct solution in a reasonable amount of time.

Report. Please put a report describing your assignment in the `Readme.md` file in your Github repository. For each problem, please include: (1) a description of how you formulated each problem; (2) a brief description of how your program works; (3) and discussion of any problems you faced, any assumptions, simplifications, and/or design decisions you made. These comments are especially important if your code does not work perfectly, since it is a chance to document the energy and thought you put into your solution. Additionally, in your report please describe (1) how you divided the work among team members, (2) the contribution of each team member.

Academic Integrity. We take academic integrity very seriously. To maintain fairness to all students in the class and integrity of our grading system, we will prosecute any academic integrity violations that we discover. Before beginning this assignment, make sure you are familiar with the Academic Integrity policy of the course, as stated in the syllabus, and ask us about any doubts or questions you may have. To briefly summarize, you may discuss the assignment with other people at a high level, e.g. discussing general strategies to solve the problem, talking about Python syntax and features, etc. You may also consult printed and/or online references, including books, tutorials, etc., but you must cite these materials (e.g. in source code comments). We expect that you'll write your own code and not copy anything from anyone else, including online resources. However, if you do copy something (e.g., a small bit of code that you think is particularly clever), you have to make it explicitly clear which parts were copied and which parts were your own. You can do this by putting a very detailed comment in your code, marking the line above which the copying began, and the line below which the copying ended, and a reference to the source. Any code that is not marked in this way must be your own, which you personally designed and wrote. You may not share written answers or code with any other students, nor may you possess code written by another student, either in whole or in part, regardless of format.

1 Part 0: Getting started

For this project, you can find your starter code by logging into IU Github at https://github.iu.edu/cs-b551-fall2024 and navigating to the repository a4-release. The final team names will be published in the following format: userid1-a4, userid1-userid2-a4, or userid1-userid2-userid3-a4, where the other user ID(s) correspond to your teammate(s). You should submit your code in a repository with the same name as your team name. If you don't already know your teammates, you can write them at userid@iu.edu.

To get started, clone the github repository:

```
git clone git@github.iu.edu:cs-b551-fall2024/a4-release.git
```

If that doesn't work, instead try:

```
git clone https://github.iu.edu/cs-b551-fall2024/a4-release.git
```

(If neither command works, you probably need to set up an IU GitHub ssh key.)

Part 1: Is an LLM smart enough to play "tic-tac-what?"

After proposing the game to Adi and having a fun time equalizing the playing field, Cam began to wonder if an LLM could take over and beat Adi for him. We will try to help Cam answer the question or at the very least provide Cam with some interesting insights.

The rules of the game remain the same:

1. The game is played on a grid board
2. Unlike regular Tic-Tac-Toe, both players use X's (there are no O's)
3. Players take turns placing a single X in any empty cell and moves cannot be skipped or passed
4. Players must place exactly one X on their turn and a cell that already contains an X cannot be used again
5. The player who completes a line of specified length of X's loses the game (the line can be vertical, horizontal or diagonal)

In this part of the assignment, we have provided you with a python notebook that has the starter code. The starter code includes code blocks for installation and LLM setup. It also includes a simple single move example for the LLama3 model, where we use a structured prompt (a prompt with well defined instructions), pass it to the model to generate structured output (for example - json, yaml, xml etc.), and we set up a parser to parse the structured output to retrieve relevant values from the LLM output. You will have to run the notebook on Google Colab (https://colab.google/) as it allows you to utilize a GPU for free. In order to run the provided starter code, upload the notebook to Google Colab and follow instructions provided in the python notebook.

You will need to build a prompt for the model such that the model follows the instructions specified and plays the game. We have provided you some test cases with a 3x3 board where a single move will specify a win or a loss for a sequence length of 3 (in the NOTE.txt file in the Part1 folder). You must answer the following questions -

1. Which test cases does your implementation pass (i.e chooses the winning move) and fail? Does it also pass on a 4x4 board with sequence length of 4 (build a similar test case if required).
2. How does changing the prompt impact the performance of the model and what changes help the model choose better moves? (Example - when you ask the model to think step by step, etc.)
3. On a 3x3 board, can your model win against a random player in a game starting with an empty board? or, how many moves does it play before it begins to fail.

Provide a python notebook with outputs. You can provide multiple python noteboooks if you choose to run different experiments in different notebooks while mentioning the experiment details at the begining of the notebook. Answer each question in an objective manner under a specified heading in the report, and provide tables if needed to show the results from your experiments.

Part 2: K-Nearest Neighbors Algorithm

You are provided with three different synthetic datasets (dataset_1, dataset_2, dataset_3), each containing training and test sets. Your task is to implement the K-Nearest Neighbors (KNN) algorithm from scratch and evaluate the accuracy of your model on the test set. Your Implementation should -

1. Use Euclidean distance as the distance metric.
2. Handle ties in case there is a tie between multiple classes
3. Not use existing ML libraries. (you can use libraries that come installed with python)

In addition, you must conduct the following ablation studies on your model (use the table format provided in the README.md file to show your results from the study)-

1. By calculating the accuracy of your model on test set for k=3, k=6, k=9 for all datasets provided in the assignment.
2. By replacing the distance metric with other distance metrics of your choice.

Last, you should try to interpret and explain the trends in the results. You must answer the following questions (feel free to use plots to answer the questions) -

1. How and why does the accuracy change across datasets?
2. What insights can you draw about how and why model accuracy changes across different values of k?
3. How does the choice of distance metric impact the performance of the model?

We've provided you with some skeleton code to get you started. In this part of the assignment, you should build a function that implements K-NN algorithm. Your program should take as input dataset_path (path to the datset file provided), k (number of nearest Neighbors considered in the analysis) and distance_metric (distance metric for K-NN algorithm).

```
python3 main.py --dataset_path 'dataset_1.json' --k 3 --distance_metric "euclidian"
```

Your report should use tables to show results and specify the relevant question being answered.

What to turn in

Be sure to include a report, as described at the start of this document. Create a private repository using the team name as described in Part 0, and keep the same directory structure as in a4-release (i.e., separating the programs into Part1 and Part2). Make sure that you stick to this naming scheme and directory structure so that our autograder can locate your submission correctly. And make sure it is private (not internal or public) so others will not see your submission. Turn in the two programs on GitHub (remember to add, commit, push) — we'll grade whatever version you've put there as of 11:59:59PM on the due date. To make sure that the latest version of your work has been accepted by GitHub, you can log into the github.iu.edu website and browse the code online.