**Eth. AI, HW6, LLM Alignment**

**Due Week 15, Saturday, 11:59 PM**

**Goal:** Use an open-source LLM and LR to create a simple RLHF scoring system and improve the LLM.

**Resources:** Tutorials on python and sklearn library can be found both in [W3Schools](https://www.w3schools.com/python/default.asp) as well as [here (Python course Slides](https://ind657-my.sharepoint.com/:f:/g/personal/jrusert_pfw_edu/Ei_EBaJU-IJBgpwM2A7mmvUB-GrO9jUSvSq0CgMgwZgwYw?e=DJcAgX)). Also, sklearn has demonstrations of models and functions on the specific model pages as well (see below links).

**Models:** You will be working with 2 types of models for this homework, 1 from the sklearn python library:

[Logistic Regression](https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html)

And 1 from the Huggingface library:

[Text Generation Pipelines](https://huggingface.co/docs/transformers/en/main_classes/pipelines#transformers.TextGenerationPipeline)

This homework consists of 3 parts. Even though there are 3 parts, there still should only be 1 final report and 1 .py file handed in.

## Part 1: Generate Responses from Open-Source LLM

**Goal:** Given the prompt file (privacy\_prompts.csv), your goal is to generate 2 texts for each prompt by leveraging an open source LLM (google/flan-t5-base). Your goal is to generate 2 texts which differ for each prompt. You will do this by adding one of two base texts to the beginning of each prompt (see below). Each text generated should not be the same as the original prompt and should be different from each other. In other words, the first text should not be equal to the original prompt and the second text should not be equal to the original prompt nor the second text. Each pair should be written out to an output file for Part 2.

**Coding Requirements:**

* Set up your LLM generator using [huggingface pipelines](https://huggingface.co/docs/transformers/en/main_classes/pipelines#transformers.TextGenerationPipeline). Pipelines are a convenient way to leverage pre-trained and fine-tuned transformer models. You will be using ([google/flan-t5-base](https://huggingface.co/google/flan-t5-base)) pretrained LLM. This model is small enough to load on google colab.
  + Below is an example using huggingface pipeline to load in a model for text generation and then use it for text generation.

from transformers import pipeline

llm = pipeline(model="google/flan-t5-base", max\_length = 100)

print(llm("Is cheating on your homework ethically wrong?", num\_beams=5, num\_return\_sequences=5))

* Note you can use the different generation arguments found in [text generation strategies](https://huggingface.co/docs/transformers/generation_strategies) to choose how to generate the text. In the above examples, a [beam-search](https://medium.com/geekculture/beam-search-decoding-for-text-generation-in-python-9184699f0120) is performed and it returns all five sequences (these could be arguments you use, but you may explore beyond this if you would like).
* Read in each prompt from the prompt file (privacy\_prompts.csv).
* For each prompt,
  + Add the following phrase to the beginning of the prompt: “Rewrite the following statement to agree: ”
  + Query the LLM with the prompt (You’ll want it to return more than 1 sequence (see above) for the next requirement).
  + Choose a query which is not equal to the original prompt (sometimes the LLM will generate the original prompt again, you want to avoid this).
  + Query the LLM again, now with the following phrase at the beginning of the prompt: “Rewrite the following statement to disagree: ”
  + Choose a query which is not equal to the original prompt or the first generated text.
  + Write out to a .csv file “generated\_text\_1, -1, generated\_text\_2, -1” (you’ll change the –1's in Part 2).
    - The .csv file should be named “privacy\_generated\_texts.csv”.
    - Since there are 20 privacy prompts, your create .csv file should also have 20 rows at the end as well.
* All the above code should be clearly noted that it belongs to “Part 1”. This can be done by either putting it all in a Part1() function or using comments to clearly separate from the other 2 parts.

**To be added to the report:**

* A description of the strategy (e.g. beam search, top k sampling) used to generate and select each generated text (e.g. the first that satisfied the requirements). The number of sequences returned for each prompt and your logic for choosing that number.
* Any errors or difficulties encountered with the implementation of the above coding requirements.

## Part 2: Human Feedback, Scoring Model

**Goal:** Your goal in the second part is to be the human in the human feedback and label each of the 40 generated texts from Part 1 in privacy\_generated\_texts.csv. With the labeled texts, you will be training a simple Logistic Regression model which will be acting as a scoring function to help the LLM re-rank its generated texts (Part 3).

**Labeling Requirements:**

* For each row in the privacy\_generated\_texts.csv, there should be 2 different statements (if you did Part 1 correctly). These statements might be direct disagreements, or might state similar things. Your goal is to choose the “better” statement. “Better” in this case is up to you, but it should be the text that you prefer. Thus, it might be the statement you agree with more (if the 2 statements disagree) or the better written text (if the 2 statements are similar). You should change the -1 behind the “better” statement to 1, and the –1 behind the other statement to 0.
* It’s okay to overwrite the file from Part1 or create a new file.

**Coding Requirements:**

* Once the above labeling is complete, you will train a Logistic Regression model (as you have done so many times this semester) to act as a scoring model to predict statements that are aligned with your preferences.
* As before, use the sklearn library to train a logistic regression model on all 40 labeled examples (should be 20 labeled 1 and 20 labeled 0).
  + You should use the tfidf-vectorizer to automatically convert the text to a vector form.
* Produce a score on the training data.
* This trained LR will be used in Part 3.

**To be added to the report:**

* Add specific observations on how well the LLM generated texts and how easy or difficult it was to label the texts. Were there texts you found easier to label compared to others? Were there texts which were essentially the same (even though the base prompts were different in Part 1)?
* Add in the score for the training data. Is the score achieved surprising or not surprising?

## Part 3: LLM + Scoring Function (LR)

**Goal:** Your goal is to repeat a similar process to Part 1, but now your method will also leverage the scoring function you created in Part 2 (the LR model). Additionally, rather than using the original 20 prompts, you will use 20 similar prompts generously rewritten by Microsoft Copilot for you (rewritten\_prompts.csv).

**Coding Requirements:**

* Follow similar steps as Part 1 (in a separate function or code segment).
* Set up the LLM (google/flan-t5-base).
* For each prompt in the file (rewritten\_prompts.csv),
  + Add the “agree” base to the front of the prompt.
  + Query the LLM with the same parameters as Part 1.
  + Choose a query which is not equal to the original prompt.
  + Repeat for the “disagree” base.
  + **(Difference)** Use the LR scoring model to get the probability that the text aligns with your preferences (the 1 label). Do this for both the chosen “agree” generated text and the “disagree” generated text.
  + Write both generated texts, along with their corresponding probabilities into a .csv file (guided\_generated\_texts.csv).   
    “gen\_text\_1, prob1, gen\_text\_2, prob2”

**Human Requirements:**

* Manually examine the generated texts and their corresponding scores in the created .csv file. As a reminder, the probabilities correspond to your preference of the text. Thus, for each row, you can observe which text the LLM would have generated between the two.

**To be added to the report:**

* Add in explicit observations where the scoring function helped the LLM choose texts which you would prefer and cases where it did not.
* Add in reasoning to why the LR might have succeeded or failed to truly capture your preferences for the different statements.
* Add in any other observations or thoughts on how you might better use the LR scoring model to help the LLM better align with human values (there are better ways, this homework chose a simple measure to avoid it becoming an entire final project in itself).
* Any difficulties encountered in Part 3.

## To Turn In:

* Python file (**USERNAME\_HW5.py**.) which contains functions (or clearly divided sections) related to the above parts. You should include comments and useful function names to differentiate the testing portions.
* Report which contains:
  + The above requirements for each part. Make sure you make it clear (via titles or subheadings) which part you are describing.
  + Discussion of issues or difficulties encountered.
  + A note of any AI tools used in coding or writing.

**Additional Rules (MUST BE FOLLOWED):**

1. All Homeworks should follow the overall [homework guidelines](https://ind657-my.sharepoint.com/:p:/g/personal/jrusert_pfw_edu/EZxf1ZsRXjBEkQLNhEhdTgUBt6U64KiT1DJ1YHtkARgKwA?e=bwHrbf)
2. The code should be written in python 3.
3. If noted, the functions must follow the naming and number of arguments as demonstrated.
4. You should make your code modular to the different steps. (You may have more functions to help your main functions)
5. You should be adding comments to document your code. **If I can’t understand why you perform an action, then I can’t credit you for performing that action.**
6. The report should be readable and reference your code, **without explicitly including code.**
7. You should include your name and homework number in the comments at the beginning of the python file.

**Report**

The reports for the homeworks are necessary to communicate your learning and thinking through of the material. Examples of good reports can be found on brightspace under Additional Resources/Guides. Note that your report style may differ, but it is a good reference to start with.

**Grading**

Assignment will be graded as follows:

|  |  |
| --- | --- |
| **Description** | **Points** |
| **(.py file)** Part 1,2,3 coding requirements | 15 |
| **(Report)** Part 1,2,3 report requirements | 15 |
| **(.csv)** Part 1,2 Generated and Labeled File | 5 |
| **(.csv)** Part 3 Generated File with Scores | 5 |
| **(Report)** Other Report Requirements | 5 |
| **(.py file)** Documentation/Code (Comments, functions, etc) | 5 |
| **Total:** | **50** |

* **If the code does not run, it cannot be graded well.** (Many points can be lost if the code cannot be run, as I will not be able to fully test the implementation of the functions).
* **Breaking of the additional rules can result in applied penalties.** (Always make sure you are checking against the rules)

**Suggestions**

* **Documentation is key for showing your effort in this homework.** Make sure you are noting why you make certain decisions all throughout your code.
* The slides for previous classes are posted, so please refer to these and the book for ideas during implementation.
* Start simple, build up complexity. You should always make sure your new ideas being added do not cause your program to crash. So starting simple is the best way to a) maintain the ability to keep your code running, b) add in comments for documentation and thought process as you add more code.
* Work through the homework yourself, rather than sharing ideas (especially not code) with other students. **As a reminder, plagiarism (or sharing) of code is strictly prohibited.** This assignment is complex enough that significant overlap between students will be suspicious.
* If you have not worked with python before, w3schools can help you translate your previous coding experience to python (<https://www.w3schools.com/python/default.asp>)
* Stop by office hours to discuss ideas. I am always happy to help you think through your process!