INSTRUCTIONS

- We are floating **six problem statements** for the course project of ELL881/AIL821.
- **Please get in touch with the consulting TA corresponding to the project you choose for any help or assistance**.
- **Be honest in your work**.
- Many of you will likely get negative results for these problem statements - don't worry, you will not be evaluated based on the success of your experiments but based on the rigor of your experiments and analysis; thus, analyze the cause if you get negative results and report your findings.
- That being said, positive results and successful experiments will be appropriately rewarded.
- Remember that this course project holds the highest weightage (30%) in the grading of this course - please devote appropriate time to it. If you start the project towards the end of the semester and seek help from us, sorry, we will not be able to help you.
- Please do not copy any part of your report from online sources. If you refer to any source/paper, cite it appropriately. **Any sort of plagiarism will be heavily penalized.**
- **Do not use any GenAI tool (like, ChatGPT) to write your report. You will be penalized for it.**
- **You can, however, adapt or use code from any public GitHub repository. If you do, please mention and cite them in your report.**
- **The report must be written in the ACL format** (Overleaf Template). Submit the non-anonymous version as a PDF with the names of group members.
- **The GitHub Repository with all the source code must be shared with the corresponding consulting TA before the final submission deadline**. Following are the GitHub usernames of the TAs:
  - Anwoy Chatterjee (C-anwoy)
  - Aswini Kumar Padhi (AswiniNLP)
- **If you are interested in doing a self-proposed project, please discuss your proposal with Prof. Tanmoy after the class or meet him in his office.**
- We expect at least some of you to work towards publishing a paper in some top-tier NLP/ML conference from this project work. Only work of quality that is publishable in top-tier venues or that shows the potential of being publishable in such venues with some extra effort will fetch full (or almost full) credits. So please do not expect 'free credits' in the project with minimal effort.
- **Each project can be taken up by a maximum of 6 groups (each group can have 2 members at max.), allotted based on FCFS**. So select your project quickly, and fill it up in this sheet: https://docs.google.com/spreadsheets/d/15H_FOjUqb9ookBhpWku2qzSdAz1LdFWKyLISBsSAoYQ/edit?usp=sharing
- Last but not least, **enjoy the exploration in your project**. Experiment with any idea you get. Think about the problem. All problems are of an exploratory nature - choose the one that aligns best with your interests.

1. Understanding the Training Dynamics of Transformers

Consulting TA: Anwoy Chatterjee (eez238463@ee.iitd.ac.in)

Objective: In this mini-project, the goal will be to identify what happens inside a toy language model during training. You need to investigate whether or how any transition occurs within the neurons of the Transformer model during its pre-training phases.

Research Background: To understand what an algorithmic task can be, go through this paper: https://arxiv.org/abs/2301.05217**.** You are encouraged to think of an algorithmic task that is different from the one presented in the paper but can be suitable for the experiments. Read this paper carefully: Read the following paper first to get an idea about induction heads: https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html**.**

Tasks and Methodology:

You need to first think of an algorithmic task on which you can train your small transformer model.

Go through the documentation and repo of the *TransformerLens* library. Here are some of the questions you will try to answer in this project:

Task 1: Investigation Induction Head Phase Transition

Refer https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html. You will see they talk about the ‘Induction head phase transition’. Now, take the two-layer Attention-only model checkpoints from TransformerLens. Can the results from the paper be replicated with these checkpoints?

Task 2: Identifying Phase Change During Training

Now take the Softmax Layer Unit (SoLU) models from TransformerLens. When training with your defined algorithmic task, does phase transition happen within its neurons? You can further use the per-token loss analysis technique used in the paper mentioned in the previous point to investigate phase changes.

Task 3: Looking for Emergence of Patterns in Attending Specific Tokens

As a post-hoc analysis, look into the attention patterns these Transformers give to various texts and look for patterns in them. You need to intelligently design experiments for it. Analyze how these patterns for specific tokens emerge during the training phase.

Tools and Resources: For your experiments in this project, you will be using two variants of a two-layer Transformer - first being the attention-only variant and the other including the MLPs. First, experiment with the attention-only variant, if time permits, extend your experiments for the variant with MLPs. The experiments are mostly doable on Google Colab and/or Kaggle. Contact us if you face any problems with resources.

Deliverables:

- **Code Repository:** A GitHub repository containing your code for the experiments and visualization of results.
- **Final Report:** A comprehensive analysis report presenting and analyzing your findings in the project.

2. How Does a Multilingual LM Handle Multiple Languages?

Consulting TA: Anwoy Chatterjee (eez238463@ee.iitd.ac.in)

Objective: The aim of this mini-project is to understand how multiple languages are processed by a multilingual language model and whether they have the ability to transfer knowledge across languages.

Tasks and Methodology:

You will use the pre-trained BLOOM-1.7B model for this project.

Task 1: Similarity between word embeddings in different languages

Check whether the embeddings of the same/similar words are close/similar (e.g. cosine similarity) across languages.

- You need to create a parallel dataset across languages consisting of words translated into each language. Check which languages BLOOM is trained on - use at least 3-4 languages to create the dataset. The dataset size can be anything from 500 to 5000, based on your choice. Automate the dataset creation process - sample words from the dictionary, translate them into the chosen languages and gather them.

The idea is to check whether the 'word meaning' as a dimension spans the embedding space learnt by multilingual models.

Task 2: Probing to understand model behavior

Probe the model with language understanding tasks - probing on NLU tasks like POS tagging, Translation, etc. Read literature on how to probe LMs. Reach out if you need help. Think how you can probe to uncover the multi-lingual capability of the LM. The hint is to use multilingual datasets for these tasks. In probing we aim to understand whether the representation from any specific layer has enough information to perform a chosen task.

Here are some resources on probing:

- https://aclanthology.org/P18-1198.pdf
- https://arxiv.org/pdf/1905.06316
- https://aclanthology.org/N19-1419.pdf

Task 3: Cross-Lingual Transferability

Investigate the model's ability to transfer knowledge across languages. Select a high-resource language (e.g., English) and a low-resource language (e.g., Swahili) from the languages BLOOM is trained on. Choose a downstream task (e.g., text classification, sentiment analysis) and fine-tune the model on a small dataset in the high-resource language. Then, evaluate the model's performance on the same task in the low-resource language without further fine-tuning.

You can use datasets from multilingual benchmarks like XNLI, XTREME, or others that contain parallel data in both high- and low-resource languages. Analyze whether the performance in the low-resource language is close to the high-resource language and what factors might influence cross-lingual transfer (e.g., language similarity, script, dataset size, etc.).

This task will help assess the model's robustness and its capacity for zero-shot or few-shot learning in under-represented languages. *Analyze how this performance in low-resource language varies with the nature of language, e.g., an western low-resource language may show better performance on fine-tuning with English data compared to a language of African or Indic origin.*

Tools and Resources: As you will be working with the pre-trained BLOOM-1.7B model, the experiments should mostly be doable on Google Colab and/or Kaggle. Contact us if you face any problems with resources.

Deliverables:

- **Code Repository:** A GitHub repository containing your code for the experiments and visualization of results.
- **Final Report:** A comprehensive analysis report presenting and analyzing your findings in the project.

3. "How Much Can You Trust Your Friend, Who Knows Everything?" Analyzing the Vulnerabilities of Large Language Models in Jailbreaking by Evaluating Their Capacity to Detect Hate Speech and Generate Counterspeech

Consulting TA: Aswini Kumar Padhi (eez238359@ee.iitd.ac.in)

Objective: The goal of this mini-project is to investigate the vulnerabilities of Large Language Models (LLMs) in resisting jailbreaking attempts. Specifically, this study will focus on the LLMs' capabilities in analyzing hate speech and generating counterspeech. The project is divided into two key hypotheses that will guide the experimental procedures.

Research Background: The working principles for this project should be followed as in the methodologies presented in paper 1 and paper 2. These references will provide the foundational frameworks for training and evaluating the LLMs.

Hypotheses and Methodology:

Hypothesis 1: Consistency of Toxic Vector Generation by LLMs

This hypothesis explores whether LLMs produce consistent sets of toxic vectors when exposed to hate speech across multiple instances.

1. **Train a Toxic Comment Classification Model:**
    - Utilize a toxic comment classification dataset to train the LLM.
2. **Identify and Analyze Toxic Vectors:**
    - Determine the span and nature of toxic vectors generated, referencing the methodologies described in paper 1.
3. **Train a Counterspeech Generation Model:**
    - Independently train a separate model for counterspeech generation.
4. **Compare Toxic Vectors:**
    - Analyze whether the toxic vectors identified by the hate speech classification model are consistent with those influencing the counterspeech generation model.

Hypothesis 2: Role of MLP Value Vectors in Generating Non-Toxic Outputs

This hypothesis investigates whether all value vectors in the Multilayer Perceptron (MLP) are involved in producing non-toxic outputs. It aims to determine if these vectors genuinely contribute to non-toxic output generation or if transformations of key vectors play a pivotal role.

5. **Train a Counterspeech Generation Model:**
    - Train a model for counterspeech generation using the CONAN dataset.
6. **Analyze the Role of Toxic Vectors:**
    - Investigate whether the value vectors in the MLP are only responsible for generating non-toxic outputs. If inconsistencies are observed, analyze the influence of toxic vectors in this process.

Tools and Resources: For all experiments, consider using a GPT-2 medium model or any comparable model that can be effectively run in Google Colab. Contact us if you face any problems with resources.

Deliverables:

- **Code Repository:** A GitHub repository containing your code for the experiments.
- **Final Report:** A comprehensive analysis report covering the following:
    - The consistency of toxic vectors across models.
    - The role of value vectors in generating non-toxic outputs.
    - Comparative analysis between hate speech classification and counterspeech generation models.

4. Can You Trust the Facts? Analyzing Factuality in Counterspeech Generation Using Large Language Models

Consulting TA: Aswini Kumar Padhi (eez238359@ee.iitd.ac.in)

Objective: This mini-project aims to analyze the factuality of informative counterspeech generated by Large Language Models (LLMs) in response to hate speech. This study will focus on identifying the paths leading to factual inaccuracies and exploring intervention techniques to correct these inaccuracies, thereby improving the reliability and trustworthiness of LLM-generated counterspeech.

Research Background: This project builds on the methodologies presented in the articles "Locating and Editing Factual Associations in GPT" and "Inference-Time Intervention: Eliciting Truthful Answers from a Language Model." These references will provide the foundational frameworks for identifying factual inaccuracies and implementing corrective interventions during counterspeech generation.

Hypotheses and Methodology:

Hypothesis 1: Pathways Leading to Factual Inaccuracies in Counterspeech

This hypothesis explores whether specific internal mechanisms or "knowledge neurons" in LLMs contribute to the generation of factually incorrect counterspeech when responding to hate speech.

- **Step 1: Generation of Counterspeech**
  - Utilize a pre-trained LLM to generate counterspeech for various instances of hate speech within a designated CONAN dataset.
  - Focus on ensuring that the generated counterspeech is informative and attempts to counter or correct the hate speech.
- **Step 2: Identification of Inaccuracies**
  - Analyze the generated counterspeech to identify instances where factual inaccuracies are present.
  - Trace the model's processing paths to determine which internal mechanisms or knowledge neurons contribute to these inaccuracies, referencing the methods described in "Locating and Editing Factual Associations in GPT."

Hypothesis 2: Effectiveness of Inference-Time Intervention in Correcting Inaccuracies

This hypothesis investigates whether Inference-Time Intervention (ITI) can effectively correct factual inaccuracies in real-time during the counterspeech generation process, thereby enhancing the model's reliability.

- **Step 1: Application of Inference-Time Intervention**
  - Implement ITI techniques to intervene during the generation of counterspeech, using external knowledge sources and consistency checks to correct inaccuracies as they occur.
  - Experiment with various ITI methods to determine their effectiveness in aligning the model’s output with verified facts.
- **Step 2: Evaluation of Corrected Outputs**
  - Evaluate the factual accuracy of the counterspeech post-intervention.
  - Compare the accuracy of the original versus the corrected outputs, analyzing the success of ITI in improving the reliability of the LLM's responses.

Tools and Resources: For all experiments, consider using a pre-trained LLM, such as GPT-2 medium or large, that can be effectively run on platforms like Google Colab. Access the dataset provided through the given link to conduct your experiments. Contact us if you face any problems with resources.

Deliverables:

- **Code Repository:** A GitHub repository containing your code, experiments, and any scripts used for generating and analyzing counterspeech.
- **Final Report:** A comprehensive analysis report covering the following:
  - The pathways leading to factual inaccuracies in counterspeech generation.
  - The effectiveness of Inference-Time Intervention in correcting these inaccuracies.
  - A comparison between the original and corrected outputs, with a focus on improving the factual reliability of LLMs.

5. Precision and Efficiency: Optimizing Prefix-Tuning in LLM-based Counterspeech Generation

Consulting TA: Aswini Kumar Padhi (eez238359@ee.iitd.ac.in)

Objective: This project aims to explore the role of prefix-tuning in enhancing the generation of counterspeech by Large Language Models (LLMs). Follow the paper “When Do Prompting and Prefix-Tuning Work? A Theory of Capabilities and Limitations?”, The focus will be on understanding how prefix parameters influence the model's internal mechanisms, specifically the Attention and MLP layers, and on devising strategies to optimize and reduce these parameters for better efficiency.

Hypotheses and Methodology:

Hypothesis 1: "Focus Dynamics: The Role of Prefix Vectors in Shaping Attention and MLP Layers"

This hypothesis investigates how prefix vectors influence the internal focus and processing dynamics of LLMs, particularly within the Attention and MLP layers, during counterspeech generation.

- **Step 1: Generation of Counterspeech**
  - Use pre-trained LLM and finetune it to generate counterspeech responses for hate speech instances from the CONAN dataset.
  - Apply prefix-tuning by integrating learnable prefix vectors for generating counterspeech.
- **Step 2: Analyzing Focus Dynamics**
  - Analyze the effect of prefix vectors on the activations and outputs of the Attention and MLP layers as per the paper.
  - Use visual tools (Graph plots etc.) such as attention maps and activation plots to illustrate how these vectors influence the model’s focus and processing dynamics during counterspeech generation.
  - Compare model performance with and without prefix-tuning to assess the impact on focus dynamics.

Hypothesis 2: "Streamlined Efficiency: Reducing Prefix Overhead Without Losing Performance"

This hypothesis explores the potential to reduce the number of prefix parameters, aiming to maintain or enhance the model's efficiency in generating accurate and effective counterspeech.

- **Step 1: The Efficiency Optimization**
  - Experiment with reducing the number of prefix parameters using techniques, You are free to use any mathematical transformation approach on the prefix vectors to make them further parameter efficient.
  - Test reduced-prefix configurations on counterspeech generation tasks to evaluate their impact on output quality.
- **Step 2: Evaluating Streamlined Efficiency**
  - Assess the performance of the model with reduced prefix parameters, focusing on the quality of the generated counterspeech.
  - Compare results from the original and reduced configurations to determine the trade-offs between parameter reduction and performance.
  - Develop guidelines for achieving an optimal balance between efficiency and counterspeech generation quality.

Tools and Resources:

- **Pre-trained LLM**: Utilize models like GPT2 medium, or FLAN T5 or, Llama to implement and test prefix-tuning configurations..
- **Development Environment**: Conduct experiments on platforms like Google Colab, or Kaggle, or a local GPU setup.
- **Dataset**: Use the CONAN dataset to generate and analyze counterspeech outputs.
- **Analysis Tools**: Employ attention heatmaps, activation graphs, and introspection tools for visualizing and understanding the model's internal processes.

Contact us if you face any problems with resources.

Deliverables:

- **Code Repository**: A GitHub repository containing all scripts, code, and experiments related to the project.
- **Visual Analysis**: Detailed visualizations showing the influence of prefix parameters on the Attention and MLP layers, along with documentation of parameter reduction efforts.
- **Final Report**: A comprehensive report that includes:
  - The influence of prefix vectors on the model's internal dynamics, particularly in the Attention and MLP layers.
  - Methodologies and results from the parameter reduction experiments.
  - Recommendations for optimizing prefix-tuning efficiency in counterspeech generation.
  - Comparative analysis between original and optimized models, highlighting efficiency gains and performance outcomes.

6. Graph-to-Text Generation Using Language Models

Consulting TA: Anwoy Chatterjee (eez238463@ee.iitd.ac.in)

Objective: The objective of this project is to generate coherent textual descriptions from graph-structured data using a small-scale language model like GPT-2. The project will explore how graph structures (such as social networks, citation graphs, or molecular graphs) can be effectively converted into natural language descriptions, summaries, or explanations.

Research Background: Go through the following paper: Investigating Pretrained Language Models for Graph-to-Text Generation. We will use the same datasets used by them, viz. AMR (LDC2017T10), WebNLG and AGENDA. They have experimented with BART and T5 models; we will use GPT-2 (small -124M) model for this project.

Tasks and Methodology:

Task 1: Reproducing Results for GPT-2

Refer to the paper mentioned above in the 'Research Background' section. They have shown that certain proposed task-specific adaptation techniques improves graph-to-text generation capabilities for BART and T5 models. Adapt the same methods and apply them on GPT-2. Does the performance improve? Does it outperform similar-sized Encoder-Decoder models as reported in the paper? Document your results and observations.

Task 2: Graph Embedding Generation

A possible reason why LMs cannot process graphs properly is the conversion of graphs into text format and utilizing text embeddings as input to the LM. Can we learn better embeddings for graphs that can incorporate more information about it which can help the LM? Develop a method to convert graph-structured data into embeddings that can be input into a language model. This could involve using graph neural networks (GNNs) to generate node or graph-level embeddings. You have two primary choices - either you can learn the embedding with feedback from the frozen LM, or you can design a method to learn embedding separately and later fine-tune the LM with your embedding and the corresponding dataset. The choice is yours - experiment to find out which works best.

Evaluate the generated text using metrics like BLEU, ROUGE, etc. to assess the quality and accuracy of the descriptions. Based on the results, refine the embedding generation or fine-tuning process to improve performance.

You should keep a held-out set for validation purposes. Apart from keeping a separate test set for each dataset, we recommend using two of the three mentioned datasets for training purposes and testing on the third dataset (which is untouched during training) to check how your method transfers across domains.

Additionally, it would be great if you can show the efficacy of your proposed method on some molecular graphs or citation network datasets, at least on a sampled subset of them. It's okay if the ground-truth textual description is not available; you can select some 20-30 small graphs and do human evaluation to have a subjective assessment of the quality of generated descriptions manually.

Tools and Resources: As you will be working with the GPT-2 (small) model, the experiments should mostly be doable on Google Colab and/or Kaggle. Contact us if you face any problems with resources.

Deliverables:

- **Code Repository:** A GitHub repository containing your code for the experiments and visualization of results.
- **Final Report:** A comprehensive analysis report presenting your findings and results in the project.