Project Assignment: Fine-Tuning Models for Cross Document Event Coreference (CDEC) Resolution

Due Date: 2025-02-25

Objective

In this project, you will fine-tune a small language model (e.g., RoBERTa) and a large language model (e.g., LLaMA) to improve performance on a cross-document event coreference task. The goal is to enable the models to correctly predict whether two event mentions in a sentence pair refer to the same real-world event.

Event Example:

Sentence: "On January 20, 2021, Joe Biden was **inaugurated** as the President of the United States in Washington, D.C."

- **Event Trigger Word**: "inaugurated"
  (This is the word that signifies the occurrence of the event.)
- **Participants**:
  - "Joe Biden" (person being inaugurated)
  - "President of the United States" (role being assumed)
- **Time**:
  - "January 20, 2021" (date of the event)
- **Location**:
  - "Washington, D.C." (place where the event occurred)

Dataset

You will receive:

- **Train, Dev, and Test Splits:** Each line in a document representing a data sample (a sentence pair containing target event mentions).
- **Format:** Each line includes (separated by tabs):
  - Unique ID of event mention1 from sentence1(e.g. 01_04_35#2_3_3)
  - Unique ID of event mention2 from sentence2(e.g. 01_04_35#2_3_3)
  - Sentence 1
  - Start token index of event1 trigger word
  - End token index of event1 trigger word(inclusive)
  - Start token index of event1 participant phrase 1
  - End token index of event1 participant phrase 1(inclusive)
  - Start token index of event1 participant phrase 2
  - End token index of event1 participant phrase 2(inclusive)

- Start token index of event1 time
- End token index of event1 time (inclusive)
- Start token index of event1 location
- End token index of event1 location (inclusive)
- Sentence 2
- Start token index of event2 trigger word
- End token index of event2 trigger word (inclusive)
- Start token index of event2 participant phrase 1
- End token index of event2 participant phrase 1 (inclusive)
- Start token index of event2 participant phrase 2
- End token index of event2 participant phrase 2 (inclusive)
- Start token index of event2 time
- End token index of event2 time (inclusive)
- Start token index of event2 location
- End token index of event2 location (inclusive)
- `label`: A binary label indicating whether the events are coreferent (`1`) or not (`0`).

  NOTE: index -1 means this information is not provided in data. You can choose to leave it be -1 or you can extract the participants, time and location of the event mentions for extra credit. Event trigger word is provided in the data.

Tasks

1. **Preprocessing**:
   - Prepare the data for input into the models, including padding, truncation, and creating attention masks.
2. **Fine-Tuning**:
   - Fine-tune a small language model (e.g., RoBERTa).
   - Fine-tune a large language model (e.g., LLaMA).
   - Use the provided training set for fine-tuning and validate performance using the dev set.
   - Some recommendations for fine-tuning:
     - Start with **LoRA** or **QLoRA** using **PEFT** + **bitsandbytes**. These methods fine-tune only a small subset of parameters (or add lightweight adapters), drastically reducing memory usage and training time.
     - Use **TRL's SFTTrainer** to guide you through a more standard and well-supported pipeline. TRL has extensive tutorials that can help you troubleshoot common pitfalls.
3. **Evaluation**:
   - Evaluate both models on the test set using metrics such as precision, recall, F1-score, and accuracy.
   - Analyze and compare the performance of the two models.
4. **Analysis**:
   - Provide an analysis of your results. For example:

        - Which model performed better, and why do you think that?
        - Were there particular types of errors that were common for one model but not the other?
5. **Experimentation**:
    - Experiment with at least one hyperparameter (e.g., learning rate, batch size, or number of epochs).
    - Document how the changes impacted performance.

Deliverables

You are required to submit:

1. **Code**:
    - A well-documented script or notebook for preprocessing, fine-tuning, and evaluation.
2. **Report**:
    - A concise report (within 2 pages) including:
        - An introduction to the task and models used.
        - A detailed description of your methodology.
        - Results and analysis.
        - Challenges faced and solutions.
        - Conclusions and any potential future work.
3. **Presentation:**
    - Prepare a short presentation based on the report and present it to class
4. **Model Checkpoints**:
    - Save and submit the fine-tuned weights for both models.

Guidelines

- Use Python with PyTorch and the Hugging Face libraries. You are free to explore other libraries as well; if you are uncertain about your choice of library, consult the TA for guidance.
- Follow best practices for fine-tuning large language models (e.g., using gradient accumulation for memory efficiency).
- Document all steps clearly in your code and report.

Extra Credit

Extra credit will be awarded for implementing novel methods that demonstrably improve the pairwise binary classification performance. For example, as mentioned in the data section above, you can leverage techniques such as semantic role labeling to extract event arguments, providing additional contextual information that enhances model performance. Creativity and well-documented approaches are encouraged.