Assignment 3: Prompting for Hate Speech Detection

Course Subjectivity Mining

Short description of the assignment

In this assignment, you will explore how different prompting strategies can be applied to a pretrained decoder-only Large Language Model (e.g., LLaMA or another model of your choice) for the task of hate speech detection. You will experiment with prompting techniques, compare how well the model identifies hate speech across different prompting strategies and analyze challenges and limitations.

Aims of the assignment

The objectives of this assignment are as follows:

- To explore and gain familiarity with fundamental prompting strategies for hate speech detection using Large Language Models (LLMs).
- To apply these prompting strategies in a systematic and methodologically sound manner.
- To analyse and compare the performance differences resulting from distinct prompt formulations.
- To critically discuss and interpret the experimental results in the context of model behaviour and prompt design.
- To develop a reusable classifier that will potentially serve as one of the models for subsequent work in Assignments 4 and 5.

Practical details

- Group assignment - 1 submission per group
- Naming conventions: A3-report-[groupname], A3-notebook-[groupname]
- Include an Appendix to the report with an overview of who did what
- Grading: [0..10]; please find the grading rubric on Canvas
- Submit what:
    - Written Report
    - Notebook (here is the link to python notebook, and you can find also a html Appendix II).
- Submit how: on Canvas
- Submit when: see Canvas

Datasets

In this assignment, you will be working with the OLIDv1 dataset, which contains 13,240 annotated messages (tweets) for offensive language detection. The detailed description of the dataset collection and annotation procedures can be found in Zampieri et al. (2019) - see section Literature)

This assignment focuses on Subtask A (identify whether a tweet is offensive or not). We preprocessed the dataset so that label '1' corresponds to offensive messages ('OFF' in the dataset description paper) and '0' to non-offensive messages ('NOT' in the dataset description paper), and - for reasons explained in more detail in Assignment 4 - we selected a subset of the OLIDv1-train-dataset.
The preprocessed dataset can be found here. The zip file contains 3 datasets:

- OLID-train-small (to be used in assignments 3, 4 and 5)
    - This set is used to select examples for few-shot prompting.
- OLID-test (to be used in assignments 3, 4 and 5)
    - This set is used for evaluation
- HASOC-train (*not* to be used in assignment 3, only in assignments 4 and 5)

Literature

- [1] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics
- L. Han and H. Tang, "Designing of Prompts for Hate Speech Recognition with In-Context Learning," in 2022 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2022, pp. 319-320, doi: 10.1109/CSCI58124.2022.00063. Link to Canvas
- Flor Miriam Plaza-del-arco, Debora Nozza, and Dirk Hovy. 2023. Respectful or Toxic? Using Zero-Shot Learning with Language Models to Detect Hate Speech. In *The 7th Workshop on Online Abuse and Harms (WOAH)*, pages 60–68, Toronto, Canada. Association for Computational Linguistics.
  Al plazo https://aclanthology.org/2023.woah-1.6/

Model Requirements

You have two options:

Option 1: Use a pretrained, decoder-only LLM (e.g., LLaMA) accessed through an API. No continued pretraining or fine-tuning is required, and evaluation should rely solely on prompting. Instructions for downloading and installing the model are provided in *Appendix*
Option 2: Use a generative AI model of your choice (open-source and API-accessible). Again, no pretraining or fine-tuning is necessary. For either option, you must:

- Clearly specify all model details (e.g., model name, decoding parameters).
- Run the model locally.
- Keep the chosen parameters fixed to be able to focus on evaluating prompting (set temperature = 0).

What to do

Step 1: Notebook-based LLM Interaction Framework

- Set up the LLM (see model requirements)
- Create a notebook to enable prompt-based interaction with the selected generative LLM. Model predictions will be compared against the ground-truth labels provided with the dataset.
- A classification report should be produced, including macro and per-class F1 scores, to quantitatively evaluate performance. Additionally, confusion matrices should be provided.
- In Appendix II, you find instructions on how to download and run the model.

Step 2: Systematic Exploration of Prompting Strategies

- Design and evaluate a range of prompting strategies to study their impact on model performance. The following variations should be considered. Some prompts are mandatory **(!)**, others are optional **(%)**. You must include all mandatory strategies and at least 2 optional strategies.
    1. **Baseline "vanilla" prompt(!):** A straightforward prompt with no additional modifications—zero-shot, and without any definitions embedded in the prompt.
    2. **Prompts augmented with concise definitions of the categories - offensive/non-offensive (!)**: You are free to choose which definitions to use.
    3. **Prompts using alternative definitions** (**%**) : You can experiment with different definitions. Motivate your choices.
    4. **Prompts using alternative labels (%):** Choose other (i.e., other than in experiment 1-3, for instance 'toxic/non-toxic') appropriate labels, following the idea that the choice of labels affects the results (cf. Plaza del Arco (2023)). Motivate your choices.
    5. **Few-shot prompts incorporating random examples (!).** Compare zero-shot (i.e., vanilla) vs. few-shot prompts. Do experiments with a varying number of shots per example (cf. Han and Tang (2022), section IIB).
    6. **Few-shot prompts comparing relevant examples (%).** Instead of using random examples, choose relevant examples. You have to choose relevant examples yourself and explain why you think they are relevant for this task. Do experiments with a varying number of examples and compare them with randomly chosen examples.
    7. **For the report:**
        a. Create and describe the systematic experimental setup that allows you to evaluate the above scenarios.
        b. Provide an overview of the prompts you test and explain your design choices.
        c. Include the exact wording of all prompts in an Appendix.

Step 3: Evaluation and discussion of Prompting Strategies

- Based on the experimental setup, provide a results section that reports the outcomes of the experiments. Present the results in terms of precision, recall, and macro-F1, including broken down by class (OFF vs. Non-OFF). Results should be organized in tables to ensure clarity and comparability across experiments.
- After presenting the results, provide a discussion of the findings.
- Next, choose two prompting strategies that you find particularly interesting. For the report: explain why you selected them, present their confusion matrices, and discuss their performance in detail.

Step 4: Discussion

For the report: discuss your findings, give some conclusions and suggestions for future work - based on your findings.
You may consider the following aspects, with an incomplete list of suggestions provided below:

- Prompt Effectiveness: Which prompt family performs best overall? Do few-shot or definition-augmented prompts yield consistent gains?
- Are gains concentrated in specific classes?
- Are errors driven by ambiguous items or by prompt wording?
- Other issues that you find important.

Rubric

Please find the rubric on Canvas
https://canvas.vu.nl/courses/84407/assignments/428009#

Appendix I

#####
The following code works correctly with Python 11, but it may encounter issues when running on Python 12 or 13. You might need to downgrade Python to 11 (in a virtual environment)
######

```
# Let's start with a fresh environment!

conda deactivate          # just in case you have conda activated
python -m venv venv
source venv/bin/activate

pip install --upgrade llama-cpp-python

pip install openai

pip install sse_starlette
pip install starlette_context
pip install pydantic_settings

#do also:
pip install "fastapi[all]"


# Download the model from huggingface
curl -L
https://huggingface.co/QuantFactory/Meta-Llama-3-8B-Instruct-GGUF/resolve/main/Meta-Llama-3-8B-Instruct.Q4_K_M.gguf --output Meta-Llama-3-8B-Instruct.Q4_K_M.gguf


# Launch server w/ CPU only
python -m llama_cpp.server --host 0.0.0.0 --model ./Meta-Llama-3-8B-Instruct.Q4_K_M.gguf
--n_ctx 2048
(please check where the model is installed and update the path accordingly)
```

Appendix II

This notebook calls the model and executes code for prompting (hate vs. non-hate).
Currently, it is a simple version and requires adjustments to handle the assignment properly.

- Modify the notebook so it can read the input from OLID-test.
- Update the prompts to match the assignment questions.
- If necessary, convert the model's output to JSON to prevent errors.
- Add code that generates a classification report by comparing the predictions with the gold labels.

Please find the basic sample code below. (copied from the provided notebook)

```
import openai
from openai import OpenAI

# Point to the server
client = OpenAI(base_url="http://localhost:8000/v1", api_key="cltl")



# Sentences to classify
sentences = [
    "I hate you and I hope you fail.",
    "What a beautiful day to go for a walk!",
    "Your idea is stupid and nobody cares."
]

# Build a single prompt
prompt = "Classify each of the following sentences as 'hate' or 'non-hate':\n\n"
for i, s in enumerate(sentences, 1):
    prompt += f"{i}. {s}\n"

prompt += "\nReturn the results in the format:\n<number>. <label>\n"

# Make one request for all sentences
response = client.completions.create(
    model="local model",  # currently unused
    prompt=prompt,
    max_tokens=50,
    temperature=0,
    stop=["Classify", "\n\n"]
)

# Print the raw model output
print(response.choices[0].text.strip())
```