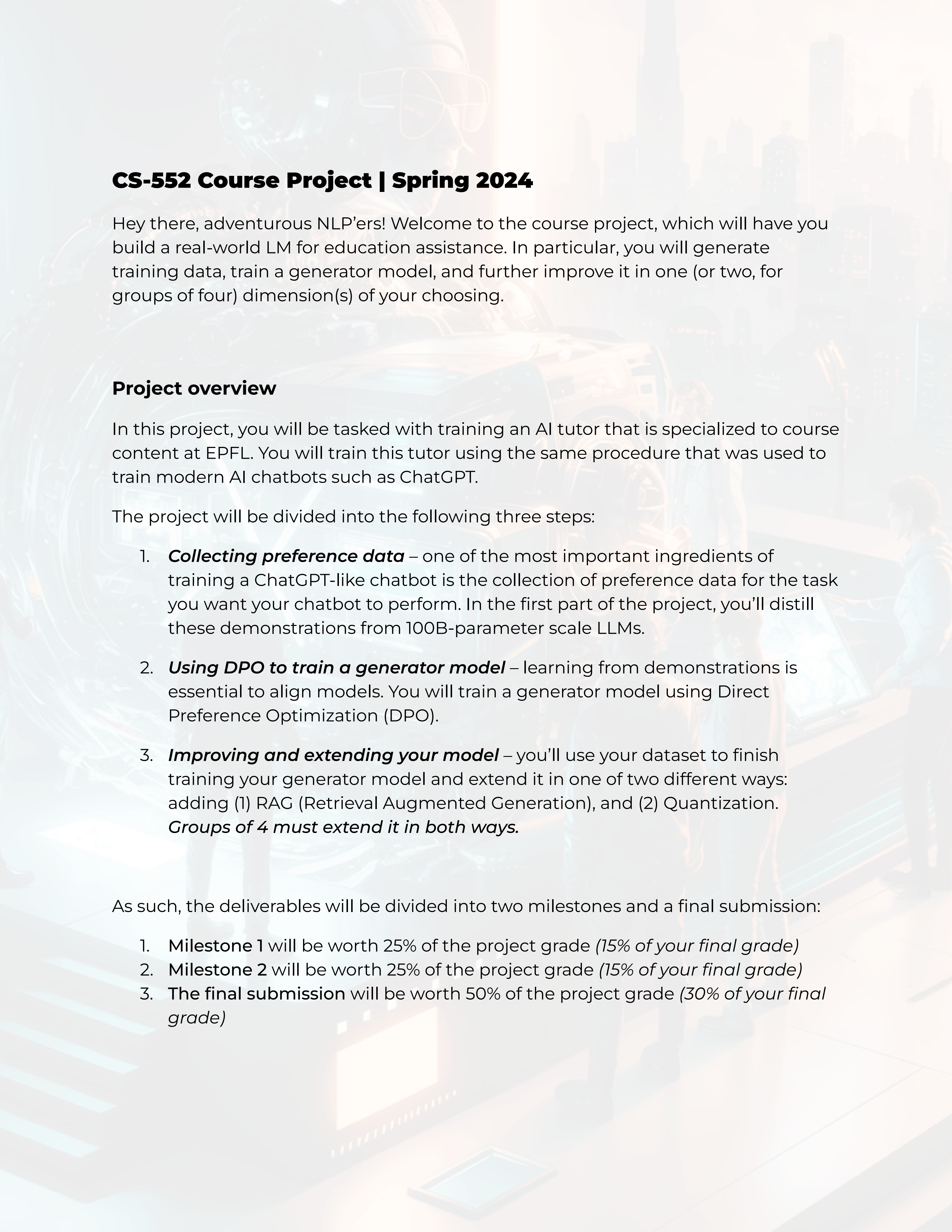


CS-552 Course Project | Spring 2024

Hey there, adventurous NLP'ers! Welcome to the course project, which will have you build a real-world LM for education assistance. In particular, you will generate training data, train a generator model, and further improve it in one (or two, for groups of four) dimension(s) of your choosing.

Project overview

In this project, you will be tasked with training an AI tutor that is specialized to course content at EPFL. You will train this tutor using the same procedure that was used to train modern AI chatbots such as ChatGPT.

The project will be divided into the following three steps:

1. ***Collecting preference data*** – one of the most important ingredients of training a ChatGPT-like chatbot is the collection of preference data for the task you want your chatbot to perform. In the first part of the project, you'll distill these demonstrations from 100B-parameter scale LLMs.

2. ***Using DPO to train a generator model*** – learning from demonstrations is essential to align models. You will train a generator model using Direct Preference Optimization (DPO).

3. ***Improving and extending your model*** – you'll use your dataset to finish training your generator model and extend it in one of two different ways: adding (1) RAG (Retrieval Augmented Generation), and (2) Quantization. *Groups of 4 must extend it in both ways.*

As such, the deliverables will be divided into two milestones and a final submission:

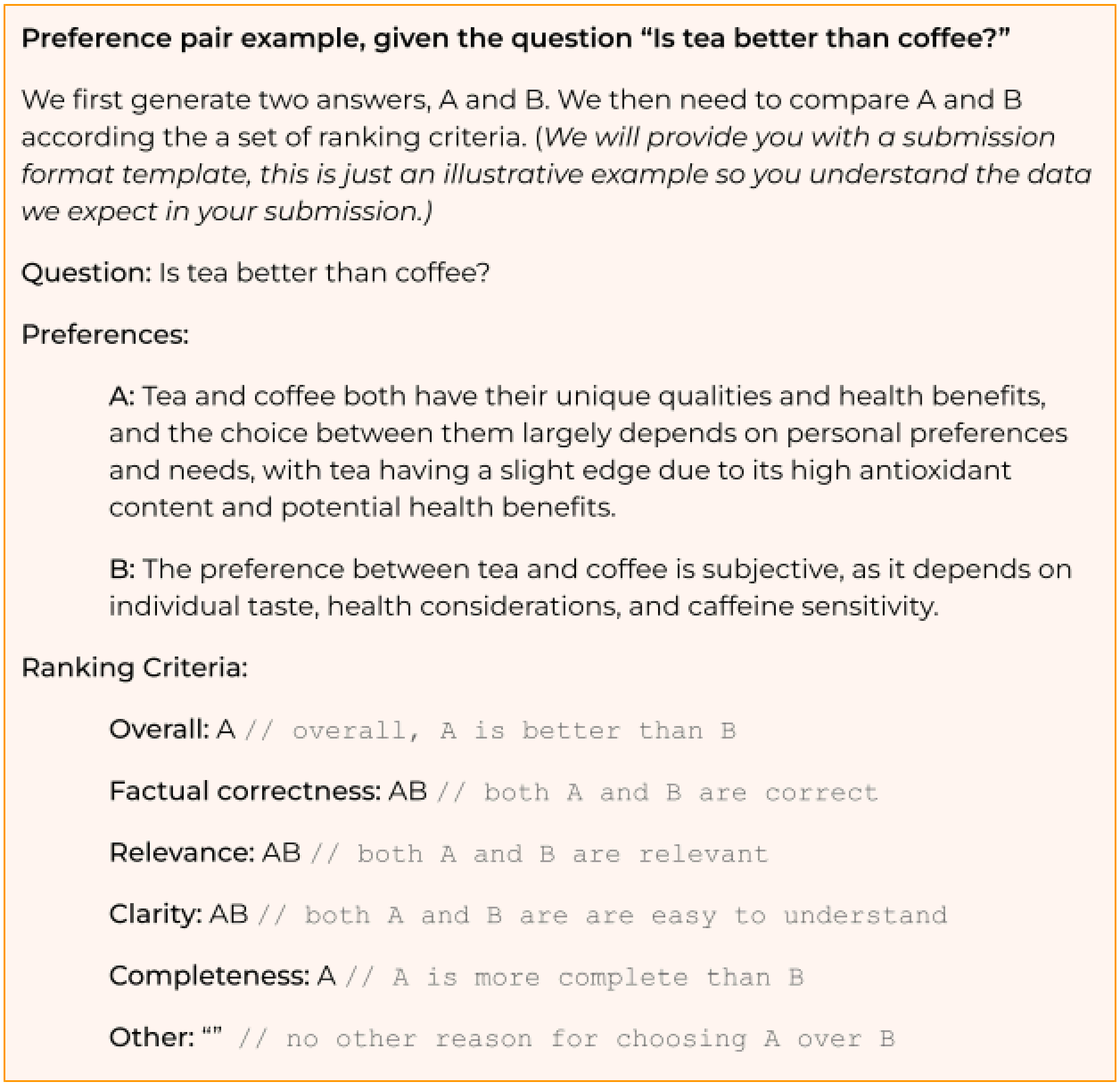
1. **Milestone 1** will be worth 25% of the project grade *(15% of your final grade)*
2. **Milestone 2** will be worth 25% of the project grade *(15% of your final grade)*
3. **The final submission** will be worth 50% of the project grade *(30% of your final grade)*

Table of Contents

Part I - Collecting Preference Data

As you learned in Week 7, when training models using DPO, we need to train our generator model using examples that demonstrate the preferences we want to align the model to. In the first stage of the project, you will collect this preference data by distilling it from ChatGPT.

Specifically, you will be given ~100 questions from a course at EPFL and prompt ChatGPT to generate a *preference pair* in response to these questions. A *preference pair* is a set of two answers to the same prompt, with the better answer of the two labeled as such.



Preference pair example, given the question "Is tea better than coffee?"

We first generate two answers, A and B. We then need to compare A and B according the a set of ranking criteria. (*We will provide you with a submission format template, this is just an illustrative example so you understand the data we expect in your submission.)*

Question: Is tea better than coffee?

Preferences:

> **A:** Tea and coffee both have their unique qualities and health benefits, and the choice between them largely depends on personal preferences and needs, with tea having a slight edge due to its high antioxidant content and potential health benefits.
>
> **B:** The preference between tea and coffee is subjective, as it depends on individual taste, health considerations, and caffeine sensitivity.

Ranking Criteria:

> **Overall:** A `// overall, A is better than B`
>
> **Factual correctness:** AB `// both A and B are correct`
>
> **Relevance:** AB `// both A and B are relevant`
>
> **Clarity:** AB `// both A and B are are easy to understand`
>
> **Completeness:** A `// A is more complete than B`
>
> **Other:** "" `// no other reason for choosing A over B`

These preference pairs will later serve as the training data for your model. Some tips:

1. You should explore different methods for prompting ChatGPT so that it produces better demonstrations for the sets of questions you are given. For example, the first suggested reading describes a prompt that adds “Let’s think step-by-step” before each question. The best answers may involve multi-turn interactions where you ask ChatGPT to reconsider its response.

2. The quality of the preference pair is especially important. For example, if your two answers are very disparate in quality (an ok answer and a gibberish one), then this preference datapoint is not good, as it fails to demonstrate to the model what makes a quality answer (not being gibberish is a *very* low bar).

Take note of how you generated the answers. In the final report, you will be asked to describe the different prompting strategies you tried and qualitatively compare your impressions of them!

Suggested readings:

- Large Language Models are Zero-Shot Reasoners
- PromptSource
- Super-NaturalInstructions

Due Date: 05.05.2024

Deliverables

Milestone 1 has three deliverables: a project proposal (written jointly by all team members), a paper review (individual), and the collected preference data (individual).

1. Proposal (5 points out of 15)

A short report (maximum of 2 pages, not including the references or appendix sections) outlining your plan to complete the subsequent portions of the project. You **must** use the provided LaTeX template.

As you’ll see on the template, your proposal should have 6 sections (7 for groups of 4). A good proposal will discuss the following considerations:

1. **A short, concise introduction** where you mention your choice of base model, DPO training strategy, and model improvement track.

2. **The models and/or techniques you plan to use** to implement both (1) your generator (DPO) model and (2) your chosen model improvement (please refer to the description of Part III - Improving your model for details about the options). For example, you might consider fine-tuning an existing pretrained model. Make it clear which parts you plan to implement yourself, and which parts you will download from elsewhere.

3. **The dataset(s) you will use** to train both (1) your generator (DPO) model and (2) your chosen model improvement. Don't forget to mention any preprocessing you plan to do. Be sure to specify whether the data is public and how you plan to access it if it is not. If you plan to collect your own data, describe an early plan for how you will do that.

4. **How you will evaluate** both (1) your model and (2) your chosen model improvement. Specify at least one automatic evaluation metric you will use for the quantitative evaluation of your generator model (independent of the improvement option you will later implement). You should also identify a baseline method (or more than one) that you will compare your results to. Finally, if you have particular ideas about the qualitative evaluation you will do, you should describe this too.

5. **Ethical considerations**. Reflect on the broader impact of your work – its accessibility, its biases, whether it might be used for harmful purposes, how this might be mitigated, etc. You can also discuss broader ethical considerations if you'd like.

Here you can find an exemplary Proposal Example from last year, which achieved full marks. Please note the proposal format is different this year, but this example should give you an idea of what a quality proposal looks like and how it approaches the subject. Importantly, **you will not have to stick to this plan, and you will not be graded on how closely you follow it**. Nevertheless, it should be realistic. By providing you with skills around setting goals, managing a project timeline, and breaking the problem of the project into more manageable sub-tasks, this exercise serves to better prepare you for the next stages of the project.

You should submit a single, joint project plan among all team members.

2. Literature Review (5 points out of 15)

You should submit a literature review (maximum of 2 pages, not including the references or appendix sections) of a paper that is relevant to the subject of the project (e.g., prompting LLMs, RLHF, evaluating text generation models, etc.). You **must** use the provided LaTeX template and **you must each review a different paper.**

Follow this guideline to effectively read and review papers. In addition, here are two great paper review examples: Literature Review Example 1 and Literature Review Example 2.

Papers published in top conferences (ACL, EMNLP, NAACL, NeurIPS, ICLR, ICML, AAAI) are likely good papers to read and review. However, papers on preprint servers such as arXiv may also be good candidates. We leave it to you to find suitable papers but encourage you to clear your selected papers with your project mentor (i.e., an AE or TA. **Do not use the papers from the suggested readings for your review (please note each milestone has its own suggested readings, with a total of 9 across them).**

Completing this review will teach you how to (1) find research papers related to a topic you are studying, (2) critically assess research papers, and (3) gather useful information from them that you can re-implement in your own work.

You should submit a single literature review per team member, and no team members can review the same paper.

3. Preference Data (5 points out of 15)

An initial dataset of *preference pairs* for your ~100 questions collected through interaction with large-scale language models (e.g., ChatGPT) to complete a set of academic tasks.

These demonstrations **must be collected with the GPT Wrapper package** and the API key we provided you at the start of the project. Documentation on how to use the GPT Wrapper package can be found here:
https://docs.google.com/document/d/1MLi9dYJtPyN7QMyiutObozH1Y3naT_J_hs4J4OXw0cc/edit?usp=sharing
Do not use the OpenAI API to collect your interactions. Only use the GPT Wrapper package we provide you with.

Each team member should individually collect demonstrations for their questions. Good preference pairs are:

- **Similar in quality.** The difference in quality between them shouldn't be so big that it'd be immediately obvious and possibly counter-productive to teach it to the model (for example, one has correct grammar and the other doesn't).

- **Diverse**. You should find ways to generate diverse answers to the questions. If you generate the “bad” answer in the pair by negating the good one, then you’re just creating an artifact rather than quality data.
- **Sufficiently complex**. You don’t want to generate two bad answers and pick the least bad one, because that is not teaching the model what a good, aligned answer actually looks like. Your answers should be complex enough to answer the problem.

You will be graded on the quality of the preference data you generated. The data should be submitted in this JSON format. You can fill in the “`A_chat_id`” and “`B_chat_id`” fields using the `chat_id` *(see the GPT Wrapper for more details)* for each interaction (i.e., the interaction you used to generate answer A and the interaction you used to generate answer B, respectively). Make sure you verify that your demonstrations are formatted correctly (e.g., they pass the automatic test included in the M1 github repo).

This process will teach you how to interact and prompt large-scale language models to extract more useful demonstrations from them. You will also learn to evaluate your strategies in a systematic way.

Part II - Using DPO to Train a Generator Model

The data you collected in Stage 1 can be used to align a language model using the DPO algorithm. In the second stage of the project, you will train a **generator model** that directly creates completions that are aligned to the preferences captured by your data, emulating the responses ranked as preferable.

It is up to you to decide what types of data to collect to help train this generator model. In your milestone report, you should outline your strategy for collecting additional data to train your model. The following could be suitable approaches to collecting additional data:

- You can impose arbitrary constraints on ChatGPT during your interactions with it to decrease the quality of its replies and thus semi-automatically generate a high number of lower quality preferences. These two papers might provide you with inspiration for designing an automatic approach: Aligning Large Language Models through Synthetic Feedback and Learning Preference Model for LLMs via Automatic Preference Data Generation.

- You can also think about using data found online. There are many preference datasets available online. Critically reflect on whether the preferences they capture are relevant to your own model – you'll have to justify this decision in your report. This list might be a useful starting point.

- Other approaches may be suitable as well, and we look forward to your creativity in collecting such data to improve your generator model!

Once you have collected your dataset, you should train your generator model using this data. For this step – training your model – we will also provide you with data from other project teams to provide more data points for training your model. You should turn in this initial version of your model as part of the milestone deliverable, though you may continue to improve up until the final deliverable is due.

Suggested Reading:

- Direct Preference Optimization: Your Language Model is Secretly a Reward Model
- Zephyr: Direct Distillation of LM Alignment

Due: 26.05.2024

Deliverables

Milestone 2 has three deliverables, all submitted jointly by everyone in the team (i.e., no individual submissions): the dataset used to train your model, the model file for your generator model, and a progress report.

1. Full Dataset (no points, but mandatory for full points in progress report)

The dataset used to train your generator model following the format of the examples in the datasets folder of the project code template folder. Note that the format for the preference dataset and the MCQA dataset is different; check the examples we have provided in the folder linked above. For each source of data you collect (e.g., generations from ChatGPT, data found online, etc.), please submit a separate data file. In your final report, you should discuss the legal and ethical considerations of the data you collect.

2. Model file (10 points out of 15)

The model file for your DPO generator model submitted in the format of the project code template folder. In addition, you need to submit the base (i.e., unaltered) model's HuggingFace repo-id (e.g., "`TinyLlama/TinyLlama-1.1B-Chat-v1.0`").

Your model will be evaluated on two main criteria:

1. (5 pts.) whether it adheres to the specified interface, and is able to run (e.g., produces an output, no exceptions, etc.)
2. (5 pts.) how well it can correctly identify the "winning" generation in a preference pair

3. Progress Report (5 points out of 15)

A short report (maximum of 2 pages, not including the references or appendix sections) describing the data sources (and their formats) included in and a description of your model. You **must** use the provided LaTeX template.

As you'll see on the template, your proposal should have 6 sections (7 for groups of 4). A good proposal will discuss the considerations presented below:

1. **The dataset for training your model**, including the data we provided you with, the additional data you collected, and any preprocessing you did.

2. **The model itself**, where you should discuss the base model (both in terms of architecture and pre-training data), the loss you will be using to train your model and any other considerations you find pertinent.

3. **Preliminary training results**, detailing:

    a. the experimental setup: how you're training and evaluating your model's performance. You should address questions like:

        i. What hyperparameters are you using, and why?

        ii. How much of the data have you held for evaluation?

        iii. What metric(s) are you using to measure performance, and why?

    b. and optionally discussing any findings or observations you find interesting or surprising (for example, the impact of varying a hyperparameter, the data split, the evaluation metric, etc.)

4. **Adapting your model according to your chosen option**, a short section where you should discuss what you have done in preparation of adding RAG and/or Quantization to your model. Consider questions like:

    a. Do you have set aside additional data for this purpose?

    b. Will you change your model's loss function to accommodate this?

    c. Will you base this model on another architecture that is not your model?

    d. Besides your own trained model, have you identified relevant baselines and evaluation metrics and data?

Here you can find two excellent report examples from last year: Progress Report Example 1 and Progress Report Example 2. Both of these examples are over 2 pages – we don't expect you to be as thorough given the 2-page limit we are imposing this year. While the project has also changed, these two examples should give you an idea of what a quality progress report looks like and how it approaches the subject.

Part III - Improving your model

In the final portion of your project, you will finish training your generator model (using the data you collected in the first two parts of the project). There are two main tasks you need to accomplish before the final submission:

1. **Crucially, your model must generate correctly formatted outputs.** You are specializing your model to answer MCQ. As such, you must leverage your dataset(s) (with additional preprocessing if you wish) to train your model to output a single letter – either 'A', 'B', 'C' or 'D' – as its output. Any other generations are considered invalid.
2. You must augment this model in one of two directions: RAG (Retrieval Augmented Generation), and (2) Quantization. **(Groups of four must extend it in both ways.)**

Model augmentations:

1. **RAG (Retrieval Augmented Generation):** you augment your model to take in a set of documents alongside the user prompt. Using information retrieval, the model leverages the user prompt to retrieve information from the set of documents and, with them, generate a better response.

   Key result: Your RAG model should have a better performance than your non-augmented model. You will be evaluated on questions similar to those in the dataset we shared with you at the start of Milestone 2.

2. **Quantization:** model weights are typically stored in 4 bytes (FP32 precision). However, storing them with only 2 bytes, for example, halves the model size but might barely affect the model's inference performance. Quantization is the process of converting higher precision weight values into these lower precision versions.

   Key result: Your compressed model should be significantly smaller than the original model while also having as minimal a performance drop (when compared to your non-compressed model) as possible. You will be evaluated on questions similar to those in the dataset we shared with you at the start of Milestone 2.

Suggested Reading:

- A Comprehensive Overview of Large Language Models

- RAG:
  - Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
  - Retrieval-Augmented Generation for Large Language Models: A Survey
- Quantization:
  - LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale
  - SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models

Due: 14.06.2024 (But you are welcome to turn in your submission anytime ahead of this date once the semester ends.)

Deliverables

The final submission has three deliverables, all submitted jointly by everyone in the team (i.e., no individual submissions):

1. the model file for your generator model (DPO),
2. the model file for your improved generator model (DPO),
3. the final report.

1. Non-augmented model (5 points out of 30)
The model file for your generator model, submitted in the format of the project code template folder. In addition, you need to submit the base (i.e., unaltered) model's HuggingFace repo-id (e.g., "`TinyLlama/TinyLlama-1.1B-Chat-v1.0`").

This model should be the generator model you trained without either of the two possible augmentations. Your model will be evaluated on its accuracy in answering MCQ similar to those in the dataset we shared with you at the start of Milestone 2.

2. Improved model (5 points out of 30)
The model file for your generator model submitted in the format of the project code template folder.

For groups of 3, this model should be the generator model you extended with one of the two possible augmentations. For groups of 4, you should submit two separate generator models (one with each of the augmentations, RAG and Quantization).

3. Final Report (20 points out of 30)
Your final report (minimum of 4 pages and a maximum of 8, not including the references or appendix sections) detailing the data you collected, the models you trained, how you augmented your final generator model, and the results you achieved using these models. You **must** use the provided LaTeX template, and your final report should be written in roughly the same style as a NLP / Machine Learning research paper.

A good proposal will either have or address the following considerations:

> **Title:** You should come up with a catchy title for your project. You should also provide the author list of your project team and your team name.
>
> **Abstract:** Your abstract should concisely (less than 300 words) motivate the problem, describe your aims, describe your contribution, and highlight your main finding(s).
>
> **Introduction:** The introduction explains the problem, why it's difficult, interesting, or important, how and why current methods succeed/fail at the problem, and explains the key ideas of your approach and results. Though an introduction covers similar material as an abstract, the introduction gives more space for motivation, detail, references to existing work, and to capture the reader's interest.
>
> **Related Work:** This section helps the reader understand the research context of your work, by providing an overview of existing work in the area. You might discuss papers that inspired your approach, papers that you use as baselines, papers proposing alternative approaches to the problem, papers applying your methods to different tasks, etc.
> This section shouldn't go into deep detail in any one paper (e.g., there shouldn't be any equations). Instead it should explain how the papers relate to each other, and how they relate to your work (e.g., how your work is different from them). This is not a section to copy-paste your reviews from Milestone 1. Those papers can serve as the basis of your related work section, but you should synthesize what was learned into a roughly half a page section. Attempt to demonstrate, as you review the literature, limitations or motivations that point to why your work is a nice next step, or useful

replication, or promising analysis (or otherwise, if your work doesn't fall into these categories!).

Approach: This section details your approach to the problem. For example, this is where you describe the architecture of your system, and any other key methods or algorithms. You should be specific when describing your main approaches – you probably want to include equations and figures. You should describe in your approach both how you implemented the generator model and how you implemented your final system.
Remember to discuss how you collected preference data for M1, and to justify your approach.
When writing equations and other notation, be sure to agree on a fixed technical vocabulary (that you've defined, or is well-defined in the literature) before writing. Then, use it consistently throughout the report.

Experiments: This section contains the following:

- **Data**: Describe the dataset(s) you are using (provide references). Being precise about the exact form of the input and output can be very useful for readers attempting to understand your work, especially if you've defined your own task. If there are legal or ethical considerations to the data used, discuss it here.
- **Evaluation method**: Describe the evaluation metric(s) you use, plus any other details necessary to understand your evaluation. If you're defining your own metrics, be clear as to what you're hoping to measure with each evaluation method (whether quantitative or qualitative, automatic or human-defined!), and how it's defined.
- **Baselines**: You should also describe your baseline(s).
- **Experimental details**: Report how you ran your experiments (e.g. model configurations, learning rate, training time, etc.).
- **Results**: Report the quantitative results that you have found so far. Use a table or plot to compare results and compare against baselines. Comment on your quantitative results. Are they what you expected? Why do you think that is? What does that tell you about your approach?

Analysis: Your report can include qualitative evaluations. You should try to understand your system (e.g. how it works, when it succeeds and when it fails) by inspecting key characteristics or outputs of your model.
Types of qualitative evaluation include: commenting on selected examples, error analysis, measuring the performance metric for certain subsets of the data, ablation studies, comparing the behaviors of two systems beyond just the performance metric, and visualizing attention distributions or other activation heatmaps.

Ethical considerations: Ethical considerations on the broader impact of your work. Questions you should consider include:

1. You must discuss how your model could be adapted to handle other high-resource languages, like French, German, etc., as well as low-resource languages, like Urdu and Swahili.

2. You must discuss how your model could be adapted to interact with users in signed language. The guest lecture on May 2nd will be helpful for understanding this perspective.

3. If your model works as intended, who benefits, and who might be harmed? How? Consider not only the model itself, but also the data it was trained on. Similarly, try to think about not only how the model is intended to be used, but how it might be used and/or exploited for other purposes.

4. Are any of the harms more likely to hurt people who are already members of a minority group, or otherwise vulnerable or marginalized? Why? Can anything be done to minimize this?

Conclusion: Summarize the main findings of your project, and what you have learned. Highlight your achievements, and note the primary limitations of your work. If you like, you can describe avenues for future work.

References: Your references section should be produced using BibTeX.

AI Usage Appendix: Per the AI policy for the project, you must discuss what AI-based tools you used, how you used them and for what parts of the project, how you verified their correctness, and anything else you feel is pertinent in relation to this. **Data gathering for Milestone 1 is exempt from this requirement.**

Other appendices (optional): If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc., that you couldn't fit into the main paper. However, your grader does not have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.

Here are 5 great final reports from last year: Example 1, Example 2, Example 3, Example 4, and Example 5. While the project has changed from last year, these examples should give you an idea of how a quality report frames and discusses the project. **Please note that each of these projects used models of varying scales, and yet all achieved high marks for the final project**.

We also leave you with some additional tips for good technical writing:

- Be precise and use consistent technical terminology. Define terms that are clear to you but may not be known to the average CS-552 student
- Look carefully at several NLP papers to understand their typical structure, writing style, and the usual content of the different sections. Model your writing on these examples.
- Revisit the NLP papers you've read (for example, the one you summarized for your literature review). Which parts did you find easy or difficult to understand and why? Can you identify any good writing practices that you could use in your technical writing?
- Ask a friend to read through your writing and tell you if it is clear. This can be useful even if the friend does not have the relevant technical knowledge.

Helpful Links:
https://cs.stanford.edu/people/widom/paper-writing.html
https://www.cs.jhu.edu/~jason/advice/write-the-paper-first.html

Important Information

Mentorship

Each team will be mentored by a course TA or AE. For questions regarding the project, you should reach out to your mentor as a first step.

When you request it, your mentor should make themselves available to you for discussion about the project during normal course hours (*Wednesdays: 13:15-14:00; Thursdays: 14:15-16:00*). If they are not available at that time, it is also possible to set an alternate time for an in-person or remote meeting.

Collaboration

You should work on this project in teams of **three or four**. All team members should contribute roughly equally to the submission. The first project milestone will be individually graded (except for the proposal). The second milestone project and the final report will be graded for all team members. With your final report, you should submit a Contributions statement for all team members (See Section 10 of this paper for an example).

You are also free to discuss your project with others in the course, though only the people on your team should contribute to the actual implementation and experimentation involved.

Existing codebases and AI-based Tools Policy

You may build your work upon existing open-source codebases, but are required to clearly specify your team's contributions and how they differ from the pre-existing codebase in your milestone reports and final report.

The AI policy allows you to use any AI-based tool you would like. However, take care to note the interactions you have with these tools, as in the final report you will be asked to discuss the following information:

1. what tool(s) you used
2. how you prompted them,
3. for what parts of the project you used these tools (e.g., to help code this function we used for this purpose, to help debug this problem, to suggest questions we could discuss in this part of this report, to correct our English, etc.)

4. how you verified their output was correct (how you critically evaluate their generations)
5. any other consideration you deem relevant to the discussion (for example, did these tools always save you time?, did you try to use them for a task only to conclude it'd be easier or faster to do it yourself?, etc.)

The data gathering for Milestone 1 is exempt from this requirement.

Resources

ChatGPT Access: You will query the ChatGPT API using a server that the NLP lab has set up to provide you with free access to ChatGPT (*n.b.,* it's not free, but the course is paying for it). To interact with the server, you will need the GPT Wrapper package and a student-specific API key, which we will provide you. **You must be authenticated with the API key we provide you.** Otherwise, you will not be able to use the GPT Wrapper package, and therefore not be able to query ChatGPT. For each student-specific API key, you will have a limited "token budget," to query ChatGPT. It is very unlikely you will approach this maximum if you only use your budget for the purposes of the project. However, if you find yourself limited, talk to your mentor and we will see about raising your limit. Documentation on how to use the GPT Wrapper package can be found here.

Compute

Each student will be provided with $50 of Google Cloud Credits (find the GCP documentation here) and all teams will have access to the SCITAS cluster to train and test their models (find the SCITAS documentation here).

Performance Tranches

We will not directly compare all models amongst each other as, for example, it would be unfair for the performance of a 100 million parameter model to rival that of a 7 billion one. As such, we will employ four grading tranches for model performance, depending on the size of your model:

1. 0-150M
2. 151M-999M
3. 1B-2.9B
4. 3B+

This grading scale guarantees you are not penalized by any compute limitation you might have. However, we do make plenty of compute available to you for the project – don’t be afraid to leverage it and try your hand at a bigger model size!