CS-552 Course Project | Spring 2025

Hey there, adventurous NLP'ers! Welcome to the CS-552 course project, which will have you train a real-world LLM for education assistance. In particular, you will collect training data, train a generative reasoning model, and improve it in two dimensions.

Project overview

In this project, you will be tasked with training a pre-trained LLM (Qwen3-0.6B-Base) to become an AI tutor that is specialized in course content at EPFL. You will train this tutor using the same procedure that was used to train modern AI chatbots such as ChatGPT, Claude, and Gemini.

The project will be divided into the following three phases:

1. ***Collecting preference data*** – one of the most important ingredients of training a ChatGPT-like assistant 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 state-of-the-art LLMs.
2. ***Training a generative reasoning model*** – your chatbot must demonstrate complex reasoning capabilities to be useful in practice. You will train a generative reasoning model using Direct Preference Optimization (DPO) and Supervised Fine-Tuning (SFT). Reasoning models must be efficient and have up-to-date information, so you will also implement Quantization and Retrieval Augmented Generation (RAG) training pipelines.
3. ***Improving your models*** – you'll use your dataset(s) to finish training your generative models, improving their performance as much as you can.

You will be expected to complete the following components and deliverables:

1. **Proposal:** worth 1% of your final grade
2. **Milestone 1:** worth 10% of your final grade
3. **Milestone 2:** worth 5% of your final grade
4. **Final submission:** worth 54% of your final grade

Table of Contents

Team Formation

Please make sure you register your team here:
https://forms.gle/8ePXaXRH3wuR9sgW8

Even if you don't yet have 4 members, **you must still register by 15.04.2025**, and we will then match you with other students to form a complete 4-person team.

Part 0 - Proposal

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

Release Date: 16.04.2025

Due Date: 04.05.2025

As you'll see on the template, your proposal should have 5 sections. A good proposal will discuss the following considerations:

1. **A short, concise introduction** where you mention your choice of training data and evaluation process.
2. **The dataset(s) you will use** to train and evaluate your (1) reward model, (2) MCQA model, (3) quantized model, and (4) RAG model. 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.
3. **Training details:** Describe the training strategy you will follow to fine-tune your different models. Provide details about your choices revolving around the losses, mix of data, and other relevant training details.
4. **How you will evaluate** your different models. Specify at least one automatic evaluation metric you will use for the quantitative evaluation of your models. You should also identify a baseline method (or more than one) to which you will compare your results. If you have particular ideas about the qualitative evaluation you will do, you should describe this too.
5. **Student roles:** Finally, describe how you will split the work among team members. Each member will be responsible for exactly one model among the (1) preference model, (2) MCQA model, (3) quantized model, and (4) RAG model. You must explicitly identify who will be principally responsible for which model.

Here you can find an exemplary Proposal Example from 2023, which achieved full marks. Please note that 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 can find the proposal template on Overleaf here, or you can download it as a zip file here.

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

Part I - Collecting Preference Data

As you learned in class, 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 (with their answers) 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`

Since it might be difficult to come up with good preference pairs if you don't know the correct answer for a given question, we will additionally provide you with the gold answers for each question. However, you should **NOT** simply assign these answers as the preferred answers, but rather use them to guide your prompting and preference generation strategy. 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 progress 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

Release Date: 28.04.2025

Due Date: 11.05.2025

Deliverables

Milestone 1 has two deliverables: literature review (group work), and the collected preference data (individual).

1. Literature Review (5 points out of 10)

You should submit a literature review (maximum of 1 page, not including the references or appendix sections) of papers that are relevant to the subject of the

project (e.g., prompting LLMs, RLHF, evaluating text generation models, etc.). This is generally known as the "Related Work" section of an academic paper and the goal of this section is to position your work in the context of existing research and discuss its similarities and differences to related works.

A good literature review synthesizes themes across papers rather than simply summarizing them one by one and groups related work by topic, method or perspective. We recommend you check out the related works section of papers published in top conferences (ACL, EMNLP, NAACL, NeurIPS, ICLR, ICML, AAAI) for examples of good literature review. You can also follow this guideline to effectively read and review papers in general.

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

Submission and Evaluation

You should submit **a single literature review for your group**, and you must use the provided template in the repository. We will evaluate your literature review on the following five dimensions (1 point for each):

- **Completeness:** Review cites at least 5 different works and from at least 2 different domains (e.g. QA, RAG, Quantization etc.)
- **Relevance:** Citations are relevant to the subject of the course.
- **Synthesis:** Review correctly synthesizes the themes from the mentioned works
- **Analysis:** Review identifies important gaps and/or connections between related works.
- **Comparison:** Review makes an appropriate comparison on how your work is different or similar to the other mentioned works.

2. Preference Data (5 points out of 10)

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:
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.

How should you generate the preference pairs?

At the most naïve level, you can imagine prompting ChatGPT asking for two answers of different quality, or getting a zero-shot answer and trying another sufficiently different prompting strategy for the other answer in the pair. We expect you to go a bit beyond this in order to ensure you are generating quality preference pairs, however (answers should be comparable in quality, diverse, and sufficiently complex)!

For MCQs you should consider generating more than just the option. You can ask the model to generate the rationale behind its choice and base your ranking on that. Answers containing (correct) rationales are much richer preference data to align the model to, compared to just a single letter/MCQ option.

How should you label a preference pair?

The labeling should be done manually. After generating the preference pairs (ensuring both answers are reasonable, in-domain, not introducing artifacts across the dataset), we expect you to manually fill in the ranking criteria for each of them. To facilitate the annotation process, we have provided you with a simple python notebook that can read a json file with answers filled in and show a visual interface to set your preferences. Note that this is just an optional tool to help you.

You will be penalized if you have pairs for which the overall ranking isn't either A or B. That is the essence of preference data: you have two answers, one of which you're trying to gear the model towards and another that you are, in contrast, discouraging it from outputting. There isn't a preference in a pair if the overall preference is None

or AB (so make sure all of your submitted pairs have either A or B for their overall ranking).

Things to keep in mind

- You should **NOT** use the provided correct answer as is for a preferred answer, but rather use it to understand what the correct answer is and guide your response generation and ranking process.
- In general, you **can** modify the model's answer. In case you wish to do so, you need to add two new fields into the json dict specifying they have been changed as follows:

  ```
  ...
  "A" : "...",
  "modified_A": true,
  "original_A": "...", # so we can check the generation
  "B" : "...",
  "modified_B": true,
  "original_B": "...", # so we can check the generation
  ...
  ```

  We will check modified answers to ensure the original answers were generated by the model.

- You should **NOT** modify the questions. The only exception is if you do so within the chat (e.g. hint the model toward the right answer/interpretation). Remember we will consolidate everyone's data at the end of M1, so it'd be hard to then merge various versions of the same original question. However, after that, you are free to generate more preference data in any way you wish, including modified questions.

Submission and Evaluation

You will be graded on the quality of the preference data you generated based on the criteria mentioned above. 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). We conceptualized you using different chats for obtaining each preference, but you can do both in the same chat. In this case you will use the same id for both

“`A_chat_id`” and “`B_chat_id`”. However, please note this might mean you're passing longer contexts to the model (thus consuming more of your token budget for the GPT Wrapper, and worst case scenario going over its context window, but this is unlikely). If you find asking everything in a single message works best, you're free to do so. 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 - Train a Generative Reasoning Model

In this part of the project, you will be designing and executing the post-training stage of Large Language Models (LLMs). The post-training stage is responsible for converting a pre-trained LLM into a **generative reasoning model**, capable of answering advanced reasoning problems in various high-level subjects. In the current scope, you will be focusing on advanced STEM subjects.

All teams are required to start from the same pre-trained LLM:Qwen3-0.6B-Base and perform the necessary post-training stage on top of this pre-trained model. You should research and explore potential methods for effective post-training, such as instruction-tuning or Direct Preference Optimization (DPO).

The data you collected in Stage 1 can be used to align a language model in your post-training stage. For example, in the case of DPO, you can optimize your **model** to generate completions that are aligned with the preferences captured by your data, emulating the responses ranked as preferable.

You can collect more data to enhance your model's reasoning ability. However, you will have to decide what types or domains of data to collect to help train this reasoning model. In your progress report, you should outline your strategy for collecting additional data to train your model. The following could be suitable approaches to collecting high-quality post-training data:

- You can distill a high-quality reasoning dataset from commercial reasoning models such as ChatGPT O1/O3 or DeepSeek R1. You can instruct the commercial models to generate synthetic reasoning trajectories or different types of thinking, such as refinement, backtracking, etc. This blog post can be a great resource for looking at methods used in current industry development.
- You can think about using data available online as “study materials” to improve the model’s reasoning performance. For example, textbooks, research papers, and encyclopedias are all sources you can explore. There is also a vast collection of datasets proposed by prior works for post-training.
- 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 human preference data found online. There are many preference datasets available on the Internet. Critically reflect on

whether the preferences they capture are relevant to your 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!

IMPORTANT! You must not train on any benchmark's test sets! For example, the following 10 datasets are prohibited from being included as part of your training data:

1. MMLU
2. MMLU-Pro
3. GPQA
4. INCLUDE
5. NLP4Education
6. AI2_ARC
7. TheoremQA
8. MuSR
9. GSM8K
10. AIME

We will conduct randomized checks to see whether your models have been contaminated with the potential benchmark test set data!

Once you have collected your dataset, you should train your reasoning model using this data. For this step – training your model – we will also provide you with data from other project teams to potentially provide more data points for the training. 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.

From the post-training stage, each team needs to provide four models, where each team member is responsible for one particular model:

1. **DPO model**: A DPO (Direct Preference Optimization) trained model aims to enhance alignment by directly learning from human or model-generated preferences. This approach optimizes models to reliably select preferred outputs in a pair of responses, ensuring the generated responses align closely with human values, intent, and judgment. DPO trains a policy directly by maximizing the likelihood of preferred outputs relative to less preferred ones, effectively learning an implicit reward function.

   Key Result: We will evaluate its performance based on the data selection of likelihood-based preference. In particular, we evaluate how well your model can recognize a human preferred response from a human rejected response.

2. **MCQA model**: Multiple-choice question answering is an efficient and systematic way to evaluate the reasoning performance of a generative reasoning model. You need to optimize a pre-trained large language model's ability to generate the correct option for a multiple-choice question in advanced STEM subjects.

   Key Result: We will evaluate the model's performance on a private benchmark that consists of multiple-choice questions on advanced STEM subjects.

3. **Quantized model**: A large language model's weights are typically stored in 4 bytes (FP32 precision). However, storing them with only 2 bytes, for example, halves the model size while retaining the model's performance at a higher precision. Thus, quantization is the process of converting higher precision weight values into lower precision versions without significant performance degradation.

   Key Result: We will evaluate the quantized model's performance using the same benchmark as the MCQA model. However, the model must be quantized (i.e., weights are stored with a lower precision than the original model's precision) in order to receive any credits.

4. **RAG (Retrieval Augmented Generation) model**: You augment your model by taking in a set of relevant documents alongside the user prompt. Using information retrieval, an embedding model leverages the user prompt to retrieve information from the set of documents that are most relevant or useful for responding to the user prompt. Your model should be able to use the retrieved information to provide more accurate responses.

   Key Result: We will evaluate the RAG model's performance using the same benchmark as the MCQA model. However, the model must be formulated as an RAG model in order to receive any credits. Specifically, you have to submit all the required modules for the pre-defined RAG pipeline in the official evaluation suite, including the **(1) set of documents you collected to retrivel from, (2) embedding model (for embedding the query and documents), (3) reasoning model that answers questions using the retrived documents and the query together as the input**.

Note that for Milestone 2, we will only evaluate if the four models you submitted can be loaded and evaluated successfully with the evaluation framework. The particular performance-based evaluation will be used for Milestone 3.

Suggested Reading:

- **DPO & Preference Data**

    - Direct Preference Optimization: Your Language Model is Secretly a Reward Model
    - Zephyr: Direct Distillation of LM Alignment
- **Post-training & Reasoning Model**
    - Tulu 3: Pushing Frontiers in Open Language Model Post-Training
    - DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models
    - Open-R1: a fully open reproduction of DeepSeek-R1
- **RAG**:
    - ChatQA 2: Bridging the Gap to Proprietary LLMs in Long Context and RAG Capabilities
    - 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
- A Comprehensive Overview of Large Language Models

Release Date: 12.05.2025

Due: 25.05.2025

Useful Links

- Evaluation suite
- Compute guideline
- MCQA demo dataset
- DPO demo dataset
- RAG documents demo
- RAG embedding models hub
- Evaluation Implementation Slides

Deliverables

Milestone 2 has 1 joint deliverable and 8 individual deliverables. The individual deliverables include the **dataset** used to train your responsible model and the **model checkpoints** for your responsible model for each of the following branches: **(1) DPO model, (2) MCQA model, (3) Quantized MCQA model, and (4) RAG MCQA model**. Additionally, for the RAG model, the responsible individual needs to submit two deliverables: **(4) RAG documents and (5) Embedding model**. The one joint deliverable is a **progress report**.

This year, we will be using the Huggingface Hub as the submission portal for your dataset and model checkpoints. Please read through the API documents and get familiar with how datasets and models are uploaded and shared. Make sure you have a Huggingface Hub account created for the model for which you are responsible.

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

You need to submit the formatted and complete dataset used to train your reasoning model. You have to submit your dataset via the Huggingface Hub as a dataset entity. You must name the repo_id for your submission properly with the following format:

1. DPO Model's Train Data: "`<HF_USERNAME_team_member_DPO>/MNLP_M2_dpo_dataset`"
2. MCQA Model's Train Data: "`<HF_USERNAME_team_member_MCQA>/MNLP_M2_mcqa_dataset`"
3. Quantized Mode's Train Data: "`<HF_USERNAME_team_member_QUANTIZED>/MNLP_M2_quantized_dataset`"
4. RAG Model's Train Data: "`<HF_USERNAME_team_member_RAG>/MNLP_M2_rag_dataset`"

The `<HF_USERNAME_team_member_X>` will be the username of your Huggingface Hub account. Here is an example of what a dataset should look like: TULU-V2. Note that the "dataset" column labels the source of the data in that dataset (i.e., whether a data point comes from flan_v2 or sharegpt). In your dataset, you will also need to include this column to indicate the source of each data point you collected (e.g., generations from ChatGPT, data found online, data from a particular dataset, etc).

2. Model file (4 points out of 5)

You have to submit your model checkpoints via the Huggingface Hub as a model entity. You must name your repo_id properly with the following format:

1. DPO Model: "`<HF_USERNAME_team_member_DPO>/MNLP_M2_dpo_model`"

2. MCQA Model: "`<HF_USERNAME_team_member_MCQA>/MNLP_M2_mcqa_model`"
3. Quantized Model: "`<HF_USERNAME_team_member_QUANTIZED>/MNLP_M2_quantized_model`"
4. RAG Model: "`<HF_USERNAME_team_member_RAG>/MNLP_M2_rag_model`"

For the team member responsible for the RAG model, submit the additional two deliverables:

5. RAG Documents: "`<HF_USERNAME_team_member_RAG>/<RAG_DOCUMENT_REPO_NAME>`"
6. RAG Embedding Model: "`<HF_USERNAME_team_member_RAG>/MNLP_M2_document_encoder`"

The `<HF_USERNAME_team_member_X>` will be the username of your Huggingface Hub account. All model checkpoints need to be compatible with HuggingFace transformers' `AutoModelForCausalLM.from_pretrained` and `AutoTokenizer.from_pretrained` loading functions. You can look at an example of Qwen-3-0.6B's model checkpoint in HuggingFace format here.

For quantization and RAG, make sure you define any config variables required to run your model so that they can be automatically loaded from your HuggingFace repo. In particular, in the file `config.json`, you need to have the fields `quantization_config` and `rag_config` accordingly.

Your model will be evaluated on the following main criteria:

(4 pt.) Whether the four models adhere to the evaluation interface and are able to run a complete evaluation that yields a metric score at the end. 1 pt for each of the models specified above.

Note: Do not update your M2 GitHub repo and all the M2 Huggingface Hub repos after the deadline for M2. Otherwise, your submission status as on time will be affected.

3. Progress Report (1 point out of 5)

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

As you'll see on the template, your progress report should have 4 sections. A good progress report 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. Optionally discuss 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 for adding RAG and/or Quantization to your model. Consider questions like:
    a. Have you 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, 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 models

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

1. Finish training your models, optimizing their performance as well as you can, and submit them, as well as the corresponding training code.
2. Submit the training data used for each model.
3. Write and submit the final report.

Note that for this final submission, the models will be evaluated based on their performance.

Release Date: 25.05.2025

Due: 08.06.2025 (But you are welcome to turn in your submission anytime ahead of this date once the semester ends.)

Deliverables

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

1. The model file for your DPO model,
2. The model file of your MCQA model,
3. The model file of your Quantized model,
4. The model file of your RAG model,
5. The model file of your RAG embedding model,
6. The documents for your RAG model,
7. The data used for training each of these four models,
8. The source code for training all four of these models,
9. Your final report.

Responsibility

- **Everyone** is equally responsible for the final report.
- Each **model** (items 1–4) will have **one main team member responsible for it.**
    - That person will be **graded more heavily** based on that model.
    - Breakdown of grading:
        - **Lead person** for a model: **15 points** (out of 54 total).
        - **Other team members** for that model: **3 points** each.

- This breakdown encourages team collaboration while tying the majority of your grade (for models) to the model you individually train.

1. Model file (24 out of 54)

You have to submit your model checkpoints via the Huggingface Hub as a model entity. You must name your repo_id properly with the following format:

1. DPO Model: "`<HF_USERNAME_team_member_DPO>/MNLP_M3_dpo_model`"
2. MCQA Model: "`<HF_USERNAME_team_member_MCQA>/MNLP_M3_mcqa_model`"
3. Quantized Model: "`<HF_USERNAME_team_member_QUANTIZED>/MNLP_M3_quantized_model`"
4. RAG Model: "`<HF_USERNAME_team_member_RAG>/MNLP_M3_rag_model`"

For the team member responsible for the RAG model, submit the additional two deliverables:

1. RAG Documents: "`<HF_USERNAME_team_member_RAG>/<RAG_DOCUMENT_REPO_NAME>`"
2. RAG Embedding Model: "`<HF_USERNAME_team_member_RAG>/MNLP_M2_document_encoder`"

As with Milestone 2:

- The `<HF_USERNAME_team_member_X>` will be the username of your Huggingface Hub account.
- All model checkpoints need to be compatible with HuggingFace transformers' `AutoModelForCausalLM.from_pretrained` and `AutoTokenizer.from_pretrained` loading functions. You can look at an example of LLaMA-3.2-1B's model checkpoint in HuggingFace format here.
- For quantization and RAG, make sure you define any config variables required to run your model so that they can be automatically loaded from your HuggingFace repo. In particular, in the file `config.json`, you need to have the fields `quantization_config` and `rag_config` accordingly.

The DPO model will be evaluated on a held-out test set of the preference data in the same domains as the data you annotated for milestone 1.

For the MCQA, RAG, and Quantized models, we use a held-out (non-public) test set of MCQ from the same domains you annotated. The questions have a single correct answer and **have exactly 4 options**.

2. Data file (Not graded but mandatory for reproducibility)

As with Milestone 2, you need to submit the formatted and complete dataset used to train your reasoning model. You have to submit your dataset via the Huggingface Hub as a dataset entity. You must name the repo_id for your submission properly with the following format:

1. DPO Model's Train Data: "`<HF_USERNAME_team_member_DPO>/MNLP_M3_dpo_dataset`"
2. MCQA Model's Train Data: "`<HF_USERNAME_team_member_MCQA>/MNLP_M3_mcqa_dataset`"
3. Quantized Mode's Train Data: "`<HF_USERNAME_team_member_QUANTIZED>/MNLP_M3_quantized_dataset`"
4. RAG Model's Train Data: "`<HF_USERNAME_team_member_RAG>/MNLP_M3_rag_dataset`"

The `<HF_USERNAME_team_member_X>` will be the username of your Huggingface Hub account.

The data format should be the same as that of milestone 2.

In your final report, you should discuss the legal and ethical considerations of the data you collect.

3. Final Report (30 points out of 54)

Your final report (minimum of 4 pages and a maximum of 6, not including the references or appendix sections) detailing 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 an 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.

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 how you implemented the generator model as well as the MCQA, RAG, and Quantization models..

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 them 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. 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 also how it might be used and/or exploited for other purposes.

3. 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 previous years: Example 1, Example 2, Example 3, Example 4, and Example 5. While the project has changed from previous years, these examples should give you an idea of how a quality report frames and discusses the 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 ones 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, non-lecture 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 **four**. All team members should contribute roughly equally to the submission. For the proposal, a grade will be given for the whole team. The first project milestone will be individually graded (except for the literature review). The second milestone project and the final report will be graded for all team members. For the final submission, each team member is primarily responsible for one model, which will count for a higher contribution to their project grade compared to the other ones. 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 and Evaluation Pipelines: You will be provided with compute to train your models as well as a library for running the models' evaluation. Details will come at a later date.