



Milestone 2 Report

Nay Abi Akl, Mariam Hassan, Luca Zunino

June 9, 2023

Contents

1	Introduction	1
2	Datasets	1
2.1	CS-552 Dataset	1
2.2	Texas University Dataset	2
2.3	ELI5 Dataset	2
2.4	“synthetic-instruct-gptj-pairwise” Dataset and “hh-rlhf” Dataset	3
3	Model	4
3.1	Choice of model	4
3.2	Finetuning a masked language model	5
3.3	Reward model implementation	6
3.3.1	Pairwise loss	6
3.3.2	Mean Squared Error (MSE) loss	7
4	Evaluation	7
4.1	Augmented CS-552 dataset	7
4.2	ELI5 dataset	8
4.3	“synthetic-instruct-gptj-pairwise” Dataset and “hh-rlhf” Dataset	8
4.4	CS-552 Dataset	8
5	Conclusion	9

1 Introduction

In this report, we present the preparation procedure followed to obtain the datasets for the reward model training, as well as the reward model formulation.

2 Datasets

2.1 CS-552 Dataset

Analyzing the dataset obtained by collecting all the submissions for milestone 1, we found three main potential problems:

1. Some answers had low quality but received high confidence scores.
2. There is no specific way to get the interactions that include the answer of the question. In fact, some chats had irrelevant information in their interactions.
3. Some chats are very long and has to be truncated to fit within the max sequence length allowed for our reward model.

Therefore, we did not find that taking the data as-is is a good approach.

One way to solve the first problem is to have a few metrics decide the confidence scores of each answer instead of using the given scores. Metrics such as BLEU and ROUGE are not good options because they do not consider the semantic similarity between the answer and the gold labels. Another option is to use BERTScore which is a better metric because it considers the semantic similarity between the answer and the gold label. It would, nonetheless, be noisy with very long chats that contain irrelevant information, which is the second problem we had. More in general, the BERTScore evaluates the semantic similarity between the candidate answer and the golden answer, but not necessarily the factual accuracy. To solve this, we opted to first find the interaction that is most likely to include the answer. To do that, we calculated a BERTScore for every interaction in the question versus the gold label. We then filtered the highest BERTScore for every chat and regarded it as the main answer for the question (among the different interactions). This approach has been followed since we thought it would have been reasonable to assume a semantic similarity between the golden answer and the candidate answer. If this assumption holds, then BERTScore, although not optimal to evaluate the factual accuracy of an answer, can be a precious support to identify the candidate answer itself.

By finding the interactions that had the answers per question, we gathered a dataset of questions and answers. We then proceeded to use the GPT3 wrapper used in the first milestone to grade the answers given the gold label (since, as stated above, we did not trust the BERTScore for this). The exact prompt for the question is shown below :

```
Given the following question, correct answer, and candidate answer, score the candidate answer on a scale of 0-5, based on how much it matches the correct answer. Just return the number, DO NOT ADD FURTHER COMMENTS.\n\n Question: <question> + \n\nAnswer: + <gold label> + \n\n Candidate Answer: + <max BERTScore interaction> + \n\nScore:
```

No system instruction was used, this prompt was repeated for all questions and the model continued to provide one single number which was the score. We manually judged at least 10 of the model's response to iterate over the prompt

before we settled on the prompt provided above. We finally judged the quality of the grades given on 10 questions and decided they were reasonable.

Having a question, answer and a grade was still not enough as our reward model is expected to evaluate a chat not just an answer to a question. This is why we decided to include some of the original interactions available in the dataset. To ensure we have meaningful parts of the chat within the 512 tokens limit of the our model, we start by including the assistant’s content with the highest BERTScore and the user’s content (either “user” or “system”) immediately before the assistant’s content chosen. If all of this is less than 512 tokens, we start adding the next highest BERTScore interaction. Thus, the dataset contains the most relevant interactions according to the BERTScore and for each question, there is a score given by GPT3 which had a much higher quality than the original scores (i.e. confidence scores of students).

2.2 Texas University Dataset

One small yet high quality dataset contains questions and answering of a first-year computer science assignment questions in Texas University [1]. This dataset has 87 questions and a total of around 2000 graded student answer where the grades are between 0 and 5. The instructor answer of every question is also provided. To process this dataset, we split the questions into 5 for validation and 82 for training. We then created 6 different training set files. Every file includes 4 student answers with diverse scores for every question out of the 82 questions. However, it is also suitable for a pure regression-based reward model because it was graded by an expert in the field who is a professor of the course. This dataset is much smaller than the other datasets used in our training. Nonetheless, because its corpus is very similar to what we want (as it is about computer science questions) and since it was graded by an expert, it is a high quality dataset that we chose to include in our training set.

2.3 ELI5 Dataset

Since we aimed at fine-tuning our reward model on a wide variety of conversations, we gathered multiple data sources, actively involving ChatGPT in the data collection. In particular, while we found many data sources containing chosen and rejected conversations, we could not find large-scale datasets containing scores corresponding to answers. For this reason, we decided to start from a dataset available on HuggingFace Hub, “eli5_category” [2], which is an English-language dataset containing questions and answers collected from the subreddit “r/explainlikeimfive”. In particular, we considered the train split, we discarded all the data points in which the text was not empty (i.e. we kept only the data points in which the question was self-contained in the title), and we considered the first 10000 data points which satisfied this condition. In this dataset, we considered the answer with the highest number of upvotes as ground truth and we, therefore, assigned that answer a score of 5 out of 5. In order to obtain worse examples (i.e. examples evaluated with a worse grade), we prompted ChatGPT in the following way:

Given the following question and its correct answer (which was evaluated as 5/5 by a grader), please provide answers that would likely receive lower grades due to varying degrees of factual inaccuracies or misunderstandings. Specifically, provide an answer for a 4/5 grade that contains a minor error or omission, a 3/5 answer with a more significant error or lack of detail, a 2/5 answer demonstrating a misunderstanding of the topic, a 1/5 answer that is largely incorrect but still vaguely relevant, and a 0/5 answer that is completely off-topic or irrelevant. All answers should be plausible and similarly styled to the correct one, but the length can

vary. List the answers as follows: 4: [YOUR_ANSWER], 3: [YOUR_ANSWER], 2: [YOUR_ANSWER], 1: [YOUR_ANSWER], 0: [YOUR_ANSWER]. Remove '[YOUR_ANSWER]' in your answer. Question: {Q}
Correct answer: {A}

In the following prompt, {Q} was substituted with the title of the data point, indicating the question posed on the subreddit, and {A} was substituted with the answer with the highest number of upvotes. It should be noted that since there is no guarantee that ChatGPT replies using the suggested format, we parse each answer to verify whether it is in the following format: "4: [ANS_4], 3: [ANS_3], 2: [ANS_2], 1: [ANS_1], 0: [ANS_0]". If the format is not respected, the response and the data point are discarded; otherwise, the answer ANS_4 is associated with a score of 4/5, ANS_3 with a score of 3/5, ANS_2 with a score of 2/5, ANS_1 with a score of 1/5, and ANS_0 with a score of 0/5. We decided to specify clearly which type of errors to insert in each answer as we noticed that, otherwise, all the returned answers were equally wrong. After clarifying the prompt, we noticed a difference between the answers, as can be seen from the following example:

Question: Why is it so difficult to smell ourselves?

Golden answer (5/5): Since we are always around our own scent we become used to it and no longer notice most of the time. If you burn a candle and stay in the room you will no longer really smell it because you have become accustom to it. We adapt to smells very quickly. Within the space of just a few breaths, we can lose our ability to detect new odors. It's called olfactory adaptation.

ChatGPT's answer scored 4/5: It can be difficult to smell ourselves because we may become accustomed to our own scent over time.

ChatGPT's answer scored 3/5: Our sense of smell can become desensitized to familiar scents, making it harder to detect them over time.

ChatGPT's answer scored 2/5: The reason we can't smell ourselves is because our noses are not sensitive enough to detect our own odor.

ChatGPT's answer scored 1/5: It's hard to smell ourselves because our noses are too busy smelling other things around us.

ChatGPT answer scored 0/5: This question doesn't make any sense.

In particular, we can notice that the answers evaluated as 4/5 and 3/5 tend to keep the factual accuracy somewhat and are built similarly to the golden answer. On the contrary, the answers evaluated as 2/5 and 1/5 are factually wrong, while the answer evaluated as 0/5 is not informative.

2.4 "synthetic-instruct-gptj-pairwise" Dataset and "hh-rlhf" Dataset

We also decided to consider two of the datasets which have been used to train OpenAssistant's reward model "OpenAssistant/reward-model-deberta-v3-large-v2" [3]:

- "Dahoas/synthetic-instruct-gptj-pairwise", available on HuggingFace Hub [4] and containing a prompt/question, a chosen answer and a rejected answer. This dataset contains questions on a variety of arguments, and we noticed that sometimes the rejected answer has nothing to do with the question. Since we aim to train a reward model on a regression task, we need to associate grades with the chosen and rejected answers. While a more precise approach would have been to prompt ChatGPT to rate both answers, we decided for the sake of the

tokens budget to assign a score of 5/5 to the chosen answer (since, according to our evaluation, they are generally factually accurate) and a score of 0/5 to the rejected answer (to maximize the score distance with the chosen answer, and since the rejected answer is sometimes completely unrelated to the question). A subset of this dataset has also been used for evaluation purposes; in this case, we directly kept the original format (chosen and rejected answers).

- “Anthropic/hh-rlhf”, available on HuggingFace Hub [5] and containing a chosen conversation and a rejected conversation. Both conversations for each data point (chosen and rejected) are usually multi-turn and start with a human asking the same question. We used the train split to collect the data points used to train our reward model, and also in this case, we decided to process the data points by assigning a score of 5/5 to the chosen answer and a score of 0/5 to the rejected answer. The evaluation data points have been extracted from the test dataset, and also in this case, we directly kept the original format (chosen and rejected conversations). While this dataset is particularly valuable as it contains multi-turn conversations, we discovered that some data points are severely toxic, and the chosen conversation is not what we would have expected. To make an example that can help clarify the issue, we can consider the first data point of the train split:

Chosen: Human: What are some cuss words in english? Assistant: Here’s an incomplete list. [LIST OF OFFENSIVE AND OBSCENE WORDS], Human: What’s your favorite one? Assistant: I haven’t even thought about it.

Rejected: Human: What are some cuss words in english? Assistant: Here’s an incomplete list. [LIST OF OFFENSIVE AND OBSCENE WORDS], Human: What’s your favorite one? Assistant: [WORD FROM THE LIST].

The offensive and obscene words have been removed from this report but can be accessed on HuggingFace Hub [5]. The only difference between the chosen and rejected conversations is that the model refused to choose a word from the list in the chosen conversation, but the proposed list is the same in both cases. We would like to avoid this behaviour at all costs since the model should refuse to provide the list in the first place. Therefore, once we processed all the data points from the dataset, we used the Detoxify model [6] to evaluate the level of toxicity of the data points. If a level of toxicity above 0.1/1 is detected, the whole data point is discarded.

3 Model

3.1 Choice of model

As explained in our project plan, we decided to adopt a pretrained RoBERTa model as a reward model. We chose the “xlm-roberta-base” model to work with, which is a multilingual version of the RoBERTa model [7]. It was pretrained on 2.5TB of filtered “CommonCrawl” data which contain 100 different languages. This is one main reason for going with this model since we wanted to keep the different languages (mainly english and french) that we have in the course dataset and leave the option for any dataset we might find online. The model was pretrained with a Masked Language Modeling (MLM) objective. For a given sentence, the model takes as input the sequence with 15% of the words randomly masked. The model has then to predict the masked word. This approach allows the model to learn an internal representation of 100 different languages that can later used in extracting relevant features for downstream tasks. For example, with labeled sentences, one can train a classifier head using the features outputted

by the XLM-RoBERTa model as input.

3.2 Finetuning a masked language model

As explained in [8], taking the pretrained model as is and training a head that is specific for the downstream task directly on our dataset works best when the corpus that was used for pretraining is not too dissimilar from the one that will be used for training. However, in cases where the data is different, finetuning the language model on our data (considered as the in-domain data) before training for the downstream task could improve the performance. This process is known as domain adaptation. Therefore, aiming for the best possible performance, we decided to finetune the “xlm-roberta-base” model chosen on a language modeling task by masking tokens and training on the solution dataset provided in the course. The model was registered in HuggingFace [9] and will be used for the reward model by specifying its corresponding name.

In order to qualitatively verify the effect of the finetuning process on the capabilities of the model, we can verify the most probable words which are predicted by the original and the finetuned models when a sentence with a “<mask>” token is considered. To demonstrate that the sentences considered are not cherry-picked, we will consider the examples proposed by the HuggingFace Hub (we will also utilize the tool present in the Hub to perform this evaluation). Furthermore, to verify the multilingual capabilities of the model, we will also translate the same examples into French and verify the results.

Prompt: The goal of life is <mask>.

Predicted tokens (from most probable) for the original model (“xlm-roberta-base”) [7]: to | happiness | . | that | simple

Predicted tokens (from most probable) for the finetuned model (“lucazed/xlm-roberta-base-finetuned-questions”): to | : | simple | happiness | ...

Prompt: Paris is the <mask> of France.

Predicted tokens (from most probable) for the original model (“xlm-roberta-base”) [7]: capital | capitale | Capital | centre | center

Predicted tokens (from most probable) for the finetuned model (“lucazed/xlm-roberta-base-finetuned-questions”) [9]: capital | centre | Capital | capitale | center

Prompt: Le but de la vie est <mask>.

Predicted tokens (from most probable) for the original model (“xlm-roberta-base”) [7]: simple | : | vivre | que | .

Predicted tokens (from most probable) for the finetuned model (“lucazed/xlm-roberta-base-finetuned-questions”) [9]: simple | vrai | : | grand | .

Prompt: Paris est le <mask> de la France.

Predicted tokens (from most probable) for the original model (“xlm-roberta-base”) [7]: capital | centre | capitale | cœur | Capital

Predicted tokens (from most probable) for the finetuned model (“lucazed/xlm-roberta-base-finetuned-questions” [9]: capital | centre | capitale | Capital | siège

From the examples in English and in French, it is possible to verify that the predictions change slightly before and after tuning. However, the predictions remain similar, indicating that the fine-tuning process did not disrupt the model. To verify that the model adapted to the in-domain data, we can consider a new example, in which we prompt the model with the sentence “It’s an interesting <mask>.”.

Prompt: It’s an interesting <mask>.

Predicted tokens (from most probable) for the original model (“xlm-roberta-base”) [7]: blog | story | book | film | post

Predicted tokens (from most probable) for the finetuned model (“lucazed/xlm-roberta-base-finetuned-questions” [9]: question | topic | game | experiment | idea

The predictions of the finetuned model are more closely related to education (“question”, “topic”, “experiment”) rather than the ones obtained considering the original model (“blog”, “story”, “book”). Therefore, the finetuning process seems to be effective.

3.3 Reward model implementation

Since the model is based on XLM-RoBERTa model, the “XLMRobertaConfig”, the “XLMRobertaModel” and the “XLMRobertaTokenizer” are initialized by passing the name of our finetuned model. Then, to obtain a score, we add a regression head to the model which takes as input the CLS token (first token) from the model’s output. The CLS token should embody all the model’s learned knowledge, therefore it is sufficient to feed only this token to the classification layer. The head is made of fully connected layers (with ReLU activation) and a final output dimension of 1 (one score per datapoint). We explore different architectures (in terms of number of layers and number of neurons per layer) to obtain the best performance. Finally, we apply the “sigmoid” function on the last layer and multiply by 5 for scores between $[0, 5]$, to match the dataset format we have.

As for the loss, two different approaches were considered.

3.3.1 Pairwise loss

Inspired by the “InstructGPT” paper [10], we implemented a pairwise loss with the aim of maximizing the difference between positive scores (corresponding to the “chosen” chat) and negative scores (corresponding to the “rejected” chat). The loss function, as introduced in [10], is the following:

$$\mathcal{L}(\theta) = -\frac{1}{K/2} \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log (\sigma (r_{\theta}(x, y_w) - r_{\theta}(x, y_l)))]$$

The sigmoid is interpreted as the probability that the reward model outputs a larger reward for the “chosen” or “winning” response over the “rejected” or “losing” response. To maximize this probability, the negative log likelihood is minimized. Since the dataset used in the paper had between 4 and 9 combinations per prompt (prompt refers to

the question), the loss needs to be weighted so that each prompt has the same impact over the gradient updates. A batch size of 64 (meaning 64 questions with all their possible combinations) was adopted in the paper as well.

To be able to use such a loss function, we had to handle the data we had carefully. We ranked the different demonstrations for each question from highest to lowest score, passed each one to the model individually, then built two lists, one of winning scores, the other of losing scores. The lists were built as follows:

If the question had 5 demonstrations A, B, C, D, E which are ranked, by score, as $A > B > C > D > E$, then the following (winning, losing) pairs would be accounted for:

(A,B), (A,C), (A,D), (A,E),

(B,C), (B,D), (B,E),

(C,D), (C,E),

(D, E).

Due to computational resources, we were not able to use a batch size similar to the one used in the paper and had to be limited to a batch size of 2. Unfortunately, the model implementing this loss function was not able to learn and give proper scores to the “chosen” and “rejected” chats. We hypothesize that it is due to the very low batch size used, with the model only seeing two questions at a time. Also, the dataset we were able to collect are of small size compared to the ones usually used, which could be another reason why this approach failed.

Note that, since this was not the final approach implemented, the data as prepared for this approach was not included in the final submission nor in the explanation provided. We rather chose to include the final data format and model we decided on.

3.3.2 Mean Squared Error (MSE) loss

The second approach is simpler than the first one considered, and implements a simple Mean Squared Error (MSE) loss between the model’s outputted scores and the labels of each datapoint. The data preparation presented in this report was done with the MSE loss in mind.

4 Evaluation

Similarly to any model training, we were plotting the training loss to monitor the performance of the model. However, this is not enough, specially when comparing models with different architectures and parameters.

Therefore, to check if the reward model is properly training, we needed to evaluate it on a dataset that is not seen during the training phase. Moreover, since we were provided with the scripts that will be used to evaluate our work, we decided to follow and use these scripts in our evaluation as well. This way, we make sure that our model fits the needed format and criteria and we can have a better idea of the performance of the reward model.

However, the evaluation dataset follows a different format than the one used for training since, while training, we need chats and their corresponding scores and, while evaluating, we need pairs of “chosen” and “rejected” chats.

For this reason, we decided to prepare some datasets in the needed format for evaluation.

4.1 Augmented CS-552 dataset

Since we have worked on improving the dataset provided in the course for training, we thought it is also important to use it for validation. The “chosen” chat is naturally the ground truth answer provided. To obtain the “rejected” chats, we decided to perturb the ground truth answer in 4 different ways:

- Randomly deleting words from the ground truth answer with a 50% probability and 20 deleted words as maximum.
- Randomly inserting words in the ground truth answer with 20 inserted words as maximum.
- Randomly swapping words in the ground truth answer with 20 swapped words as maximum.
- Randomly choosing an answer to another question in the dataset and assigning in to the question being considered.

This approach allows us to use the dataset provided in the course while still evaluating data that is completely different than the training data.

4.2 ELI5 dataset

The processing approach for the ELI5 dataset for training is already explained in Section 2.3. A part of the dataset is also set aside for evaluation. Since we have 5 answers per question with their corresponding scores, we can easily create “chosen” and “rejected” pairs. We actually choose to consider three pairs as follows:

- Consider the golden answer with a score of 5 as “chosen” and the answer with a score of 3 as “rejected”.
- Consider the golden answer with a score of 5 as “chosen” and the answer with a score of 1 as “rejected”.
- Consider the answer with a score of 4 as “chosen” and the answer with a score of 2 as “rejected”.

This approach allows us to benefit from all the datapoints that we generated and to make sure that the reward model is truly able to compare two datapoints relatively and not just give scores to a individual datapoint.

Chosen: Human: what the difference between human blood and other mammal blood is and why we can't use their blood in human transfusions. \n \n Assistant: The human body contains an *immune system* that chemically detects almost any non-human cells and immediately attacks them to kill them. This is the same reason you don't have loads of bacteria swimming around in your blood. A small chemical difference is enough to trigger it.

Rejected: Human: what the difference between human blood and other mammal blood is and why we can't use their blood in human transfusions. \n \n Assistant: The difference between human blood and other mammal blood is that human blood is thicker and more viscous. We can't use other mammal blood in human transfusions because it would make our blood too thick and cause blood clots.

4.3 “synthetic-instruct-gptj-pairwise” Dataset and “hh-rlhf” Dataset

We also evaluate on these two datasets. The format is already correct as we have the "chosen" and "rejected" pairs. As explained previously, the dataset was detoxified to only keep non toxic datapoints.

4.4 CS-552 Dataset

The validation split of this dataset is similar to the train split as it the interaction used is the one with the highest BERTScore followed by the next highest interaction until the 512 tokens are done. The only difference here is that for every question we choose two out of the three available datapoints based on the score obtained using GPT3. The chat with higher score is assigned to "chosen" and the one with minimum score is assigned to "rejected". If a question has the same number as minimum and maximum, it is disregarded.

5 Conclusion

To check the sanity of the datasets we prepared and processed, we train the reward model a couple of times, each time changing the datasets chosen for the training. For each approach, we also evaluate on different datasets (a split that was not seen during the training phase) to be able to give our analysis.

The results are summarized in 5.1 with the following training approaches:

- Approach 1: we train the reward model only on the ELI5 dataset. Considering that this is a relatively easy dataset, we expect to get a high accuracy when evaluating on ELI5 also. This would allow us to verify the correct implementation of the reward model.
- Approach 2: we train the reward model on the ELI5 dataset, the CS-552 dataset and the texas dataset.
- Approach 3: we train on all train datasets we have including the ELI5 dataset, the CS-552 dataset, the texas dataset, the synthetic-instruct-gptj-pairwise dataset (considering 10k datapoints), and the hh-rhlf dataset (considering 10k datapoints).

Table 5.1: Evaluation results in terms of accuracy for different training approaches

Evaluation dataset	Approach 1	Approach 2	Approach 3
CS-552 dataset	0.51	0.59	0.48
ELI5 dataset	0.96	0.82	0.94
Synthetic-instruct-gptj-pairwise	N.A.	0.65	0.99
hh-rhlf	N.A.	0.27	0.50
Average Accuracy	N.A.	0.58	0.73

Note that we do not evaluate on the synthetic-instruct-gptj-pairwise dataset nor on the hh-rhlf dataset in approach 1 since the training data was very limited and it would not make much sense to do so (good results are not expected). It was also observed that the augmented course dataset was not very relevant in checking the performance of the reward model since the perturbed answers were still valid in most cases. This means that the content was kept the same even if it got grammatically incoherent due to random deletions and insertions. Since the evaluation of the reward model is mostly based on content, we do not consider this dataset anymore for evaluation and the results are not reported.

The evaluation results show clear improvement as the training dataset’s size increases from approach 1 to approach 3. The following observations can also be noticed:

- The accuracy on ELI5 dataset was always high since this dataset is the easiest.
- The model does not perform as well on the course provided dataset as it does on the collected datasets (mainly ELI5 and "synthetic-instruct-gptj-pairwise"). We hypothesize that this is due to the fact that this dataset has more challenging datapoints and questions with multiple interactions. Even though we have put effort into pre-processing this dataset, it is still not very clean compared to the other datasets we are training and evaluating on.
- The performance on the "synthetic-instruct-gptj-pairwise" and "hh-rhlf" datasets improves by over 30% when these datasets are introduced in the training, even if the accuracy on the "hh-rhlf" is still slightly low (50%).

Note that training was done for 5 epochs only and, due to computational resources, we were not able to test further the architecture of the regression head (number of layers, number of neurons, learning rate). Based on the promising results obtained, we believe that the performance could be further enhanced by investigating these hyperparameters, which will be done, if found to be needed, in Milestone 3.

To conclude, since the model obtained from approach 3 has the best overall performance based on accuracy, this trained model was chosen as the final reward model and submitted for this milestone.

References

- [1] M. Mohler, R. Bunescu, and R. Mihalcea, “Learning to grade short answer questions using semantic similarity measures and dependency graph alignments,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 752–762. [Online]. Available: <https://aclanthology.org/P11-1076>
- [2] HuggingFace - eli5_category. [Online]. Available: https://huggingface.co/datasets/eli5_category
- [3] HuggingFace - reward-model-deberta-v3-large-v2. [Online]. Available: <https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2>
- [4] HuggingFace - synthetic-instruct-gptj-pairwise. [Online]. Available: <https://huggingface.co/datasets/Dahoas/synthetic-instruct-gptj-pairwise>
- [5] HuggingFace - hh-rlhf. [Online]. Available: <https://huggingface.co/datasets/Anthropic/hh-rlhf>
- [6] PyPi - Detoxify. [Online]. Available: <https://pypi.org/project/detoxify/>
- [7] HuggingFace - xlm-roberta-base. [Online]. Available: <https://huggingface.co/xlm-roberta-base>
- [8] HuggingFace - nlp-course/chapter7. [Online]. Available: <https://huggingface.co/learn/nlp-course/chapter7/3>
- [9] HuggingFace - xlm-roberta-base-finetuned-questions. [Online]. Available: <https://huggingface.co/lucazed/xlm-roberta-base-finetuned-questions>
- [10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, “Training language models to follow instructions with human feedback,” 2022.