Modern NLP Course Project
Milestone 2: Reward Model

Antoine Bonnet
antoine.bonnet@epfl.ch

Silvia Romanato
silvia.romanato@epfl.ch

Alexander Sternfeld
alexander.sternfeld@epfl.ch

School of Computer Science
Ecole Polytechnique Fédérale de Lausanne

1 Introduction

This report details how we trained a reward model to rate the quality of answers given a question. This model will later be used to train a generative model with reinforcement learning from human feedback (RLHF). In Section 2, we discuss the two sources of data used to train our reward model: the EPFL *NLP4Education* (2.1) and StackOverflow (2.2) datasets and their respective potential issues. In Section 3, we go over how we used a pre-trained sequence classification model from Huggingface to model synthetic rewards. Finally, we analyze our results in Section 4.

2 Data

The goal of our reward model was to generate scalar rewards for tuples of the form *(prompt, response)*. We collected data from two sources; the EPFL dataset (2.1) and StackOverflow dataset (2.2). We pre-processed both sources into the same form *Human: Question. Assistant: Answer.*

2.1 EPFL dataset

First dataset. For this project, we were provided with data on 4450 unique questions from courses at the École Polytechnique Fédérale Lausanne (EPFL) for the NLP4Education project. Students were each assigned a subset of 100 of these questions and were asked to manually collect interactions with ChatGPT. The aim of the students was to find the best prompting strategy to answer these questions. For each interaction, the students assigned a confidence score indicating how certain they were that the answer is correct. Table 1 shows the distribution of the confidence scores and the number of collected answers per question.

Table 1: Statistics for the manually collected interactions with chat-GPT

Number of distinct chats per question		Confidence score	
Value	Percentage of the data	Value	Percentage of the data
1	21.4%	0	1.2%
2	19.2%	1	5.4%
3	54.2%	2	8.1%
4	5.1%	3	16.5%
5	0.02%	4	26.8%
6	0.09%	5	41.9%

Submitted to *CS-552: Modern Natural Language Processing* at EPFL in June 2023.

Pre-processing. We removed the prompting instructions that the students prompted to ChatGPT since otherwise the model would learn to identify the presence of system instructions as student interactions rather than solutions, which could bias the label produced by the model. We were also provided with the official solutions to the questions. When provided, the explanations for the solutions were concatenated with the answer body.

Challenges. We faced three challenges when processing this data:

1. Each student introduces bias when ranking interactions, since they might not know the true answer to each question and they might have a different confidence scale.
2. We are not assured that the official solutions are the best answers to the questions when compared to the user interactions. In fact, ChatGPT might have provided a better answer than the official solution.
3. Another problem is that some user interactions are few-shot, meaning that multiple prompts were used to generate the answer (see Figure 1). This is never the case for solutions, which are all one-shot. The model might learn to recognize chats with multiple *Human:* and *Assistant:* prompts as ChatGPT interactions and so introduce bias in its output, since incorrect chats are all ChatGPT interactions.

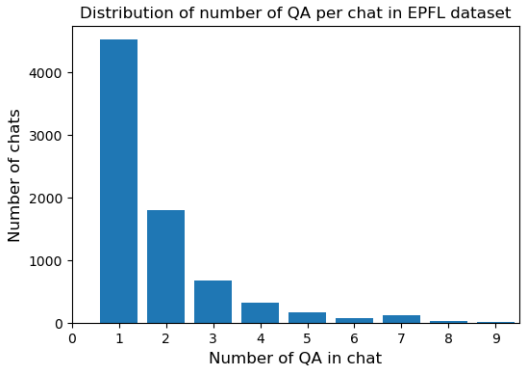

Figure 1: Distribution of the number of user interactions per question in the EPFL dataset.

Assumptions. To address these challenges, we make three simplifying assumptions;

1. We assume that the official solution is a correct answer.
2. We assume that ChatGPT interactions with confidence 5 are also correct answers.
3. Removing all few-shot interactions would remove about 40% of the dataset. We therefore chose to keep them, and add all interactions with confidence 5 to the correct answers mix. This means that both correct and incorrect answers contain few-shot interactions, reducing possible biases in the model training.

Labelling. We then combine user interactions and official solutions in the same dataset. We label interactions with confidence 1, 2 or 3 as incorrect answers, and the official solutions as well as interactions with confidence 5 as correct answers. We keep only questions for which we have at least one correct answer and one incorrect answer. The resulting EPFL dataset contains 2284 distinct questions and 6955 unique answers.

2.2 StackOverflow dataset

Second dataset. Our main objective was to fine-tune a model to recognize correct from incorrect answers from the EPFL dataset. To augment the training data and improve generalization, we decided to add a second data source to the training mix. As a second source, we used the open-source StackOverflow dataset, which is a publicly available question-and-answer platform. The dataset contains over 27 million answers on a wide variety of topics, of which we extract 3 specific topics: Computer Science, Computer Science theory and Data Science. We choose these topics as the aim of

our final generative model will be to answer questions from EPFL courses, which are mostly related to engineering. The dataset contains several (question, answer) pairs. Each question has a title, body view count and ID (with topic identifier attached to it). Each answer has a body, number of upvotes and label (1 if the answer was accepted by the original poster, 0 otherwise).

Pre-processing. We perform several steps of pre-processing for this data. First, we retain only the posts that have an accepted answer. We only retrain questions for which we have one correct answer and at least one (and up to 30) non-accepted answers per question. The distribution of the number of answers per question is displayed in Figure 2. One can see that most questions in our final data set have a relatively low number of answers. We concatenate question and answers into a single string *Human: [Question] Assistant: [Answer]*. Each question-answer pair is assigned a label 1 ('correct') if the answer is accepted by the poster, and 0 ('incorrect') otherwise. This labelling choice relies on the assumptions that (1) the accepted answer is of the highest quality compared to all other answers and that (2) all non-accepted answers are of lower quality than the accepted answer. After preprocessing, we obtain a total of 8086 questions and 22441 corresponding answers.

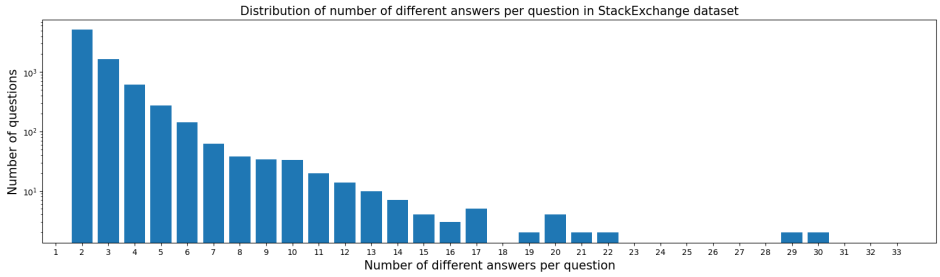

Figure 2: Distribution of the number of answers per question in the StackOverflow dataset.

2.3 Legal and Ethical considerations

When considering the legal and ethical implications of the datasets, three main factors need careful consideration. First and foremost, privacy is protected as both the students and the users of Stack Overflow have given informed consent, and their personal data will not be disclosed or utilized. However, one should note that one can not neglect the possibility that students have shared private information through there interactions with chat-GPT. Likewise, it may be that users of Stack Overflow have shared personal details in a post or answer. As the data is not filtered, this private data may not be protected well when the data is used in downstream applications.

Second, it is important to recognize that student biases may exist based on their prior knowledge, potentially affecting the confidence scores they assign. This introduces the possibility of bias in the reward model's performance. If the collected data is not of high quality, the reward model may learn incorrect features from the answers, amplifying any existing biases. Similarly, the accepted answers on Stack Overflow may reflect biases of the population that uses the platform. Therefore it may not translate well to the general population.

Last, we are using raw data, hence it has not been filtered for offensive, incorrect or toxic content. For the NLP4Education dataset, there may be examples where an answer has been assigned a confidence level of 5, while it is perceived as offensive by others. On Stack Overflow, there are more preventive measures to avoid offensive posts and answers, such as the possibility for another user to mark a post as "offensive". However, there remains a possibility of the presence of offensive content in the data. In downstream applications, there is a risk that these offensive answers persist.

3 Reward model

3.1 First Attempt: Contrastive Regression Model

Regression model. Our initial approach was to train a regression model to produce synthetic rewards directly for each Q&A pair. Following recommendations from the InstructGPT paper [1], we constructed a regression model with the pre-trained RoBERTa base model [2] and a regression head (fully-connected layer(s) with sigmoid activation) added on top of the [CLS] token embedding.

Contrastive learning. We trained this model to predict the score of answers given a question. Pairs of correct and incorrect answers *[(question - 'correct' answer), (question - 'incorrect' answer)]* pairs were passed through the network to train the model to compare the quality of answers. The model was fine-tuned on the objective of minimizing the negative log of the sigmoid of the difference between the reward for the correct answer y_{correct} and the reward for the incorrect answer $y_{\text{incorrect}}$:

$$\mathcal{L}(r_\theta) = -\mathbb{E}_{(x, y_{\text{correct}}, y_{\text{incorrect}}, i) \sim \mathcal{D}} \big[\log(\sigma(r_\theta(x, y_{\text{correct}}) - r_\theta(x, y_{\text{incorrect}}))) \big]$$

However, this approach was unfruitful; comparing rewards for correct and incorrect Q&A samples showed accuracy of 47% on the StackOverflow test set. We therefore switched gears and trained a classifier model instead.

3.2 Second Attempt: Classification Model

Classification model. In our second approach, we frame the task as a binary classification problem by training a classifier model to differentiate 'correct' from 'incorrect' chats. To do so, we use the Huggingface pre-trained sequence classification model (RobertaForSequenceClassification) with the RoBERTa base model (*roberta-base*) [2]. This model consists of the Transformer RoBERTa base model with its corresponding pre-trained tokenizer and a classification head added on top of the [CLS] token embedding.

Base model. The Transformer RoBERTa base model was trained on approximately 160GB of English data, using a masked-language modeling (MLM) objective which means that the model is trained to predict randomly masked tokens in a sequence [2]. It is composed of a stack of 12 identical layers with 12 attention heads each.

Classification head. The classification head takes the 768-dimensional embedding, applies a first linear layer with sigmoid and 10% dropout into a hidden layer of equal dimension 768. A second linear layer is added with output dimension 2 (one for each label (0 for 'incorrect', 1 for 'correct'). Each class therefore receives an associated output logit. The resulting label is generated as the maximum output logit. This classification head enables us to predict whether a given question-answer pair contains a correct answer or not. A scalar reward can also be extracted from the output logits by applying a sigmoid to map them to the $[0, 1]$ range.

Tokenizer. We tokenized the data using the RoBERTa pre-trained tokenizer. We also truncated the tokenized question-answer pairs to 512 tokens (maximum length of a sequence that can be processed by RoBERTa). We found that too many question-answer samples exceeded this limit, so instead of removing those from the dataset we decided to truncate all text samples to the first 512 tokens. We also padded the text to 512 tokens if it was shorter than 512 tokens.

Loss function. Considering the class imbalances inherent to the datasets, where we typically have a single correct answer among 1 to 30 incorrect answers, we address this issue by training the model on the weighted cross entropy loss between the output logits and the 0-1 label for each sample. This is a modified version of binary cross-entropy loss that mitigates the impact of class imbalances. It achieves this by weighing the loss for each sample based on the class frequency within the batch. This weighting mechanism ensures that the model is trained effectively, even in this scenario with imbalanced class distributions.

3.3 Training

Training procedure. As we wanted to optimize our model to perform best on the EPFL dataset, we first started training the model with the StackOverflow dataset as a form of pre-training for 20 epochs. We then trained our model for another 20 epochs on the EPFL dataset as a final fine-tuning. After every epoch, the model was evaluated on several metrics over a reserved validation set for the current dataset trained on, as shown in Figure 3. The best model was loaded after every dataset training from the epoch at which its weighted cross entropy loss on the validation set was minimized.

Training hyperparameters. We used the Adam with weight decay (AdamW) optimizer, with batches of 16 chats with weight decay 10%. Each dataset was trained on with a different linear scheduler; StackOverflow started at learning rate 10^{-5}, while EPFL started at the lower learning rate 10^{-6}.

Mixed precision training and gradient accumulation over 4 batches were used to speed up training to 5 hours on a single GPU.

Splitting the data. For the training of the reward model, we will split the training data into a training set and a validation set. The validation set can then be used to evaluate the performance of the reward model. We split the dataset into training, validation and test sets using a 70/10/20 split with shuffling. Answers to the same questions were not necessarily batched together; however we ensured that all answers to the same questions were contained in the same set to avoid any data leakage and to ensure that no test data had been trained on before evaluation.

4 Results

Evaluation metrics. We were performing a binary classification task on two imbalanced datasets; in each one, we had more incorrect answers than correct ones. In this imbalanced context, the usual accuracy metric loses its meaning. We therefore used other metrics commonly used in such instances, which we detail briefly.

- *Precision*: fraction of correct samples among those classified as correct. High precision indicates that when the model labels an instance as correct, it is likely to be correct.
- *Recall* (also called true positive rate/sensitivity): fraction of correct instances that are correctly classified as correct. A high recall indicates that the model is effective at identifying correct instances.
- *Specificity* (also called true negative rate): Specificity is the percentage of correct samples that are correctly classified as incorrect. A high specificity indicates that when the model predicts an instance as incorrect, it is likely to be incorrect.
- *F1-score*: Harmonic mean of precision and recall. Serves as a balance between the two measures and is a good indicator of the overall the model's performance. It is based on the predicted labels.
- *ROC-AUC*: Area under the Receiver Operating Characteristic curve. This performance metric evaluates the model's ability to differentiate correct from incorrect chats across different classification thresholds. It is based on the models' predicted scores (i.e. logits), and not the predicted labels.

Validation metrics. We logged all validation metrics on the current dataset during training after every epoch, as shown below. During the first 20 epochs of training on the StackOverflow dataset, most metrics increased slightly except Recall, which varied unstably. From 20 epochs onwards, the EPFL dataset saw a high increase in validation metrics in the first epochs, then increased marginally but stably until the end of training. If we were to rerun training, we would use smaller learning rates to see if the metrics would improve more stably.

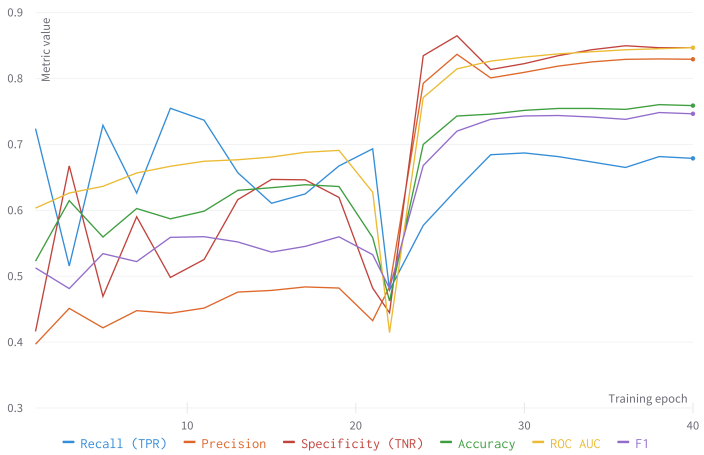

Figure 3: Classifier model validation metrics on current dataset during training

We evaluated the model on the reserved test sets of both datasets after each training round, as shown in the table below.

Table 2: Evaluation Metrics After Each Round of Training

Metric	Round 1		Round 2	
	StackOverflow	EPFL	StackOverflow	EPFL
Accuracy	55.8%	46.2%	57.4%	74.0%
Specificity	48.2%	44.4%	67.6%	84.2%
Recall	69.3%	47.9%	39.7%	64.5%
Precision	43.2%	48.4%	41.1%	81.7%
F1-score	53.2%	48.1%	40.3%	72.1%
ROC AUC	0.627	0.414	0.557	0.818

Performance analysis. From the above table, we see that all metrics had significant improvements on the EPFL test set after training on the EPFL dataset (as expected). On the EPFL dataset (for which we were most interested in having good performance), we reached a F1-score of 72.1% and a ROC AUC of 0.818, which are considered strong performance. We reach lower final performance on the StackOverflow dataset, with a F1-score of 40.3% and a ROC AUC of 0.557.

This may be explained by the fact that StackOverflow answers are hard to classify; differentiating accepted from non-accepted answers is complex, even for human annotators, because the original poster can only accept a single answer as correct, while multiple answers may well be considered as correct as well. One way to mitigate this bias would be to use the user upvotes of each answers to train the model to identify answers with many upvotes as correct.

Another reason for the low performance on the StackOverflow dataset may also be that training on the EPFL dataset hurts performance on the StackOverflow dataset. As shown in Table 3, the F1-score goes from 53.2% to 40.3% and the ROC AUC goes from 0.627 to 0.557 after training on the EPFL dataset. This means that the model generalizes poorly, and is best-suited to the last dataset it was trained on because it has short-term memory. A way to mitigate these undesired effects would be to use smaller learning rates, especially for the second dataset, and to increase the capacity of the classification head by increasing the number of layers to improve its memory capacity and generalization ability.

References

[1] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL].

[2] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. July 2019. DOI: 10.48550/arXiv.1907.11692. URL: http://arxiv.org/abs/1907.11692 (visited on May 12, 2023).