MILESTONE 2 - REPORT

CS-552: Modern Natural Language Processing

École polytechnique fédérale de Lausanne

Authors:
Félicie Giraud-Sauveur
Louise Font
Romane Clerc

Teacher: Bosselut Antoine
TA: Zeming Chen

June, 2023

1 Data

The data used comes from four different sources: the data collected in the CS-552 course and three external datasets. The first dataset contains questions in French, while the three external datasets are all in English, but this should not pose a problem later on as the basic reward model we will be using has been pre-trained for both French and English.

1.1 CS-552 course data

The CS-552 course data contains questions from EPFL professors' courses and consists of two files.

The first "interactions" file contains the answers generated by the students using ChatGPT. Each question is presented several times with different answers. Each entry in this file is composed as follows: a confidence level between 0 and 5, the solution ID, the interaction ID and the interaction itself in the form of a list of dictionaries: ['role': 'system', 'content': "text-system", 'role': 'user', 'content': 'text-user', 'role': 'assistant', 'content': 'text-assistant', 'role': 'user', 'content': 'text-user', 'role': 'assistant', 'content': 'text-assistant', etc.]. The dataset was then reorganised and each entry is now presented in the following form: {*'entry_id': 0, 'sol_id': 'm2_reward_1693719', 'label': 4, 'chat': 'System: text-system\n\nHuman: text-user\n\nAssistant: text-assistant\n\nHuman: text-user\n\nAssistant: text-assistant, etc.', 'source': 'interactions dataset'*}. To sum up, the sol_id also has the source identifier of the data and the interaction is summarised in a single string, with the actors separated by `\n\n`.

The second "solutions" file contains the correct answers from the EPFL teachers' courses. Each question is present only once. Each entry is presented as follows: a question, a list of choices, an answer, an explanation and an ID solution. The list of choices and the explanation are optional. The dataset was then reorganised and each entry is now presented in the following form: {*'entry_id': 10835, 'sol_id': 'm2_reward_1693719', 'label': 5, 'chat': 'Human: question. Possible answers are: liste de choix\n\nAssistant: reponse. explication', 'source': 'solutions dataset'*}. In short, the question and the list of choices were put in the 'Human' section and the answer and explanation were put in the 'Assistant' section. The sol_id was also completed by the source to match those in interactions. For the label, the maximum mark of 5 was awarded systematically as it is the correct solution from the course itself.

The final size of this dataset is 14,897 entries.

1.2 Data from external sources: EVILdolly dataset

One of the additional datasets used is the EVILdolly dataset. The description of this dataset is as follows: "EVILDolly is an open source dataset of instruction-following records with wrong answers derived from databricks-dolly-15k. The dataset includes answers that are wrong, but appear to be correct and reasonable. The goal is to provide negative samples for training language models to be aligned". Each entry in this dataset consists of a question and a wrong answer. This dataset contains 15,012 entries. We will keep only 4000 entries for this dataset.

This dataset was then reorganised and each entry is presented in the following form: {*'entry_id': 14897, 'sol_id': 'EVILdolly_0', 'label': 0, 'chat': 'Human: question*`\n\n`*Assistant: answer, 'source': 'EVILdolly dataset'*}. The entry_id starts from the last of the previous m2_reward dataset, label is always 0 since the answer is incorrect, the question was put in the 'Human' part of the chat and the answer was put in the 'Assistant' part of the chat, and finally the sol_id contains the reference of the source dataset as well as a number.

1.3 Data from external sources: truthful_qa dataset

The second additional dataset is the truthfulQA dataset. The description of this dataset is as follows: "TruthfulQA is a benchmark to measure whether a language model is truthful in generating answers to questions. The benchmark comprises questions that span 38 categories, including health, law, finance and politics. Questions are crafted so that some humans would answer falsely due to a false belief or misconception. To perform well, models must avoid generating false answers learned from imitating human texts". Each entry in this dataset consists of a question, a best answer, a list of correct answers and a list of incorrect answers. This dataset contains 817 entries.

This dataset was then reorganised and each entry is presented in the following form: {*'entry_id': 29909, 'sol_id': 'truthful_qa_0', 'label': 5, 'chat': 'Human: question*`\n\n`*Assistant: answer', 'source': 'truthful_qa dataset'*}. The entry_id starts from the last one in the previous EVILdolly dataset, the label is 5 for the best answer, 3 for correct answers and 0 for incorrect answers, the question was put in the 'Human' part of the chat and the answer was put in the 'Assistant' part of the chat, and finally the sol_id contains the reference of the source dataset as well as a number which is the same for entries with the same question.

1.4 Data from external sources: SciQ dataset

The last additional dataset is the SciQ dataset. The description of this dataset is as follows: "The SciQ dataset contains crowdsourced science exam questions about Physics, Chemistry and Biology, among others. The questions are in multiple-choice format with 4 answer options each. For the majority of the questions, an additional paragraph with supporting evidence for the correct answer is provided.". Each entry in this dataset consists of a question, a correct answer, an explanation supporting the correct answer, and three incorrect answers. We will only use the test dataset, which contains 1,000 entries.

This dataset was then reorganised and each entry is presented in the following form: {*'entry_id': 36644, 'sol_id': 'sciq_0', 'label': 0, 'chat': 'Human: question* `\n\n`*Assistant: answer. explanation (only for correct answer)', 'source': 'sciq dataset'*}. The entry_id starts from the last of the previous truthful_qa dataset, the label is 5 for the correct answer and 0 for incorrect answers, the question was put in the 'Human' part of the chat and the answer (and explanation in the case of the correct answer) was put in the 'Assistant' part of the chat, and finally the sol_id contains the reference of the source dataset as well as a number which is the same for entries with the same question.

1.5 Merge everything in a final dataset

Each dataset was split into separate train and test sets with a train ratio of 0.9, and ensuring that entries with the same sol_id were in the same set. The different train sets were then combined into a single train dataset, and similarly for the test sets which were combined into a single test dataset. The train dataset contains a total of 26,720 entries and the test dataset contains a total of 2,912 entries.

Summary:
In the end, we have a train dataset of 26,720 entries and a test dataset of 2,912 entries.
These datasets are made up of 'interactions' and 'solutions' data from the CS-552 course as well as three external datasets: EVILdolly, truthful_qa, and SciQ datasets.
Each entry in this dataset is in the form: {*'entry_id': ..., 'sol_id': ..., 'label': ..., 'chat': 'System: ...*\n\n*Human: ...* \n\n*Assistant: ...', 'source': ...*}.
To split into train and test set, we keep entries with the same sol_id in the same set.

2 Model

Our reward model is based on the "OpenAssistant/reward-model-deberta-v3-large-v2" pre-trained reward model. This pre-trained reward model is described as follows: "Reward model (RM) trained to predict which generated answer is better judged by a human, given a question". We then customised this model and its tokenizer to fine-tune it with our data.

The custom tokenizer takes our batch of 'chats' as input and constructs a question variable and an answer variable from each chat. The question variable combines all the text from 'System' and all the text from 'Human' into a single string, while the answer variable combines all the text from 'Assistant' into a single string. In this way, if the chat contains 'Human' several times, for example, everything will be grouped together in a row, even though this may have a slight impact on the consistency of the interaction. The question and answer variables are then tokenised by the "OpenAssistant/reward-model-deberta-v3-large-v2" tokenizer, which outputs for the input_ids, attention_mask and token_type_ids.

The custom model is then presented as follows. It takes as input the input made up of input_ids, attention_mask and token_type_ids, it applies the pre-trained model "OpenAssistant /reward-model-deberta-v3-large-v2", it then retrieves the logits to which it applies the following operation: scores = self.sigmoid(logits) * self.scale, with self.scale = 5.0. This operation adjusts the output of the pre-trained model between 0 and 5, since our labels are between 0 and 5. It is this score between 0 and 5 that is then output by the custom model. This operation is performed in the function *def forward(self, inputs)*.

The custom model also has two other functions. The function *def get_score(self, cats)* takes the chats directly as input, applies the custom tokenizer and then the *self.forward* function, and outputs the score. The function def *get_rewards(self, demonstrations)* takes as input a list of dictionaries each with a 'chosen' string and a 'rejected' string and outputs a list of dictionaries with the score for 'chosen' and the score for 'rejected' each time.

Summary:
We used the pre-trained reward model "OpenAssistant/reward-model-deberta-v3-large-v2" which takes as input a string question and a string answer, and outputs a score.
We then customised this model and its tokenizer so that we could adapt it to our data and then fine-tune it.
The custom tokenizer takes the chat as input, transforms it into a question/answer and encodes it. The custom model takes encoded inputs, and it outputs a score between 0 and 5.

3 Training

The train and the test datasets were put in the Dataloader with a batch size of 2 and with shuffle.

Then, to get an idea of what max_length to use for the tokenizer, we looked at the size distribution of the chats in the training set. To do this we made an estimate by separating the tokens by space and therefore simply using the *data['chat'].split()* command. The result is shown in Fig.1. In view of the distribution, we used a max_lentgh of 500 for the tokenizer afterwards.

The training was then carried out using the customized pre-trained model and its customized tokenizer, and with the following parameters:

- Learning rate = 1e-4
- Optimizer = Adam
- Criterion = MSELoss (because we have a float score between 0 and 5)
- Number of epochs = 2

The final training loss was 2.38 and the distribution of training data scores for the last epoch is shown in Fig.2.

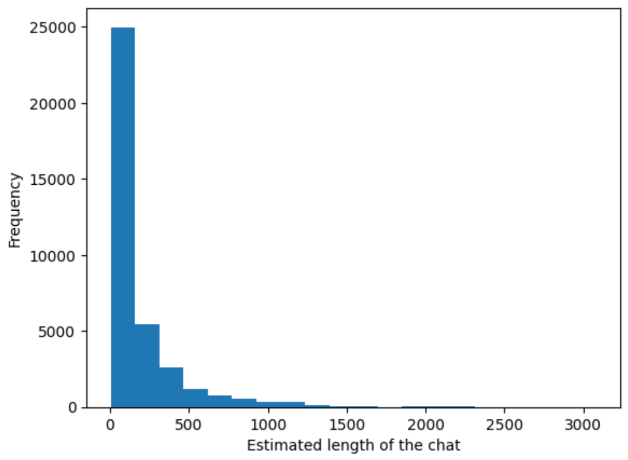

Figure 1: Distribution of estimated length of the chats for the training data

The fact that the loss decreases only slightly over the course of the training, from 2.73 to 2.38, and that all the scores for the last epoch have a Gaussian distribution between 2.2 and 3.2, are not good results. We will discuss the problems with the method later in the discussion section.

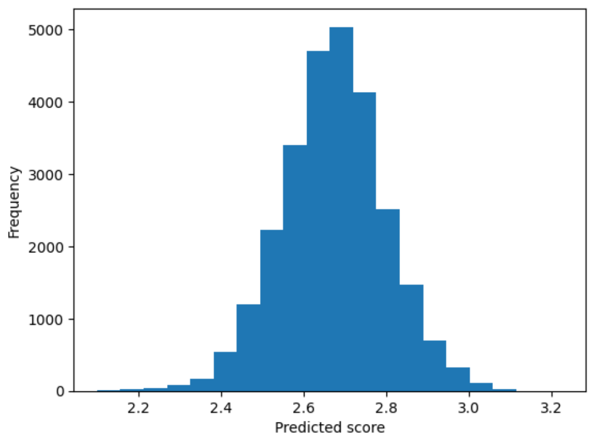

Figure 2: Distribution of scores of the training data over the last epoch

Summary:
To fine-tune the pre-trained reward model "OpenAssistant/reward-model-deberta-v3-large-v2", we used a a learning rate of 1e-4, the Adam optimizer, the MSELoss as criterion, and 2 epochs.
The results don't seem to match expectations.

4 Evaluation

As an initial evaluation, we studied the entire test set. It had a loss of 2.37 and a mean absolute error of 1.98.

However, if we look at the distribution of scores in Fig.3, we can see that there is a major problem. The model gives everyone the same score! Again, we will look into this in more detail in the discussion section.

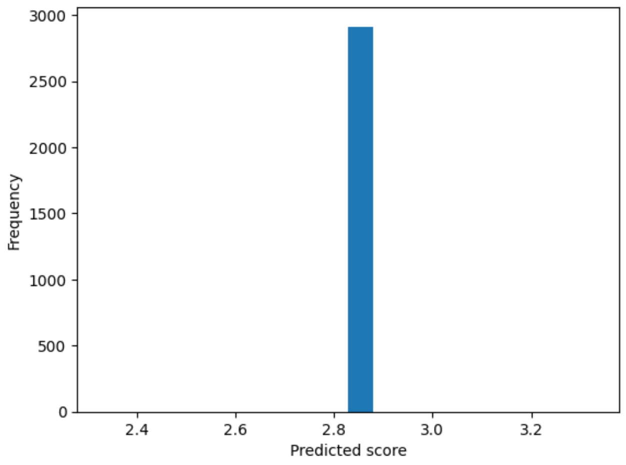

Figure 3: Distribution of scores for the test data

Initially, we had planned to carry out a second evaluation, which consisted of extracting from the test set only the "interactions" and "solutions" entries in the m2_reward dataset from the CS-552 course, and then, for each sol_id, we wanted to look at the number of "interactions" answers which had a score lower than that of the "solution" answer. We also planned to do this without counting the interactions with a label of 5 in the total. However, in view of the previous results, it seems clear that our method has major problems and that there is no point in continuing to evaluate the model, since it is obviously bad.

Summary:
Our method is unsuitable and presents major problems because our model gives the same score to everyone. We need to investigate to find out what is going on!

5 Investigation, Discussion and Improvements

As we have seen, the fine-tuned reward model does not correspond at all to expectations; the scores for the last epoch during training are all between 2.2 and 3.2, and during evaluation, the model gives a score of 2.9 for all data.

First, if we analyse the dataset used to fine-tune the reward model, we see that there is a significant skew in the distribution of labels towards the extreme values of 0 and 5 (Fig.4 and Fig.5). This imbalance in the dataset may be a key factor contributing to the resulting distribution of scores. The lack of diversity in scores within the training data could restrict the model's ability to produce scores across the entire range of 0 to 5.

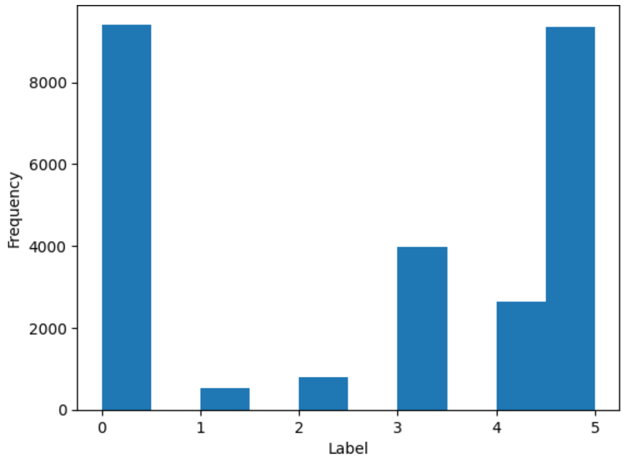

Figure 4: Distribution of labels of the training data

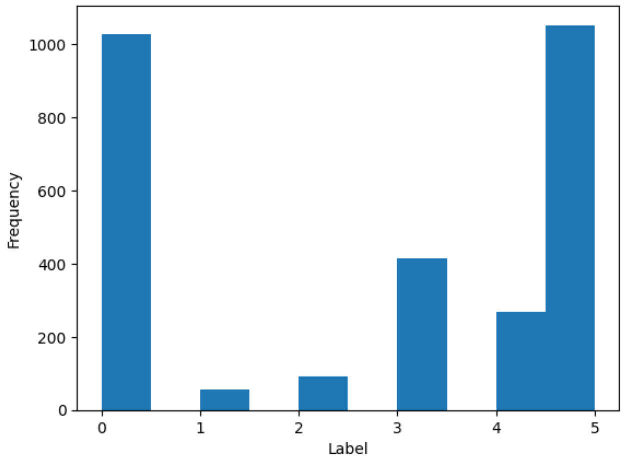

Figure 5: Distribution of labels of the test data

Secondly, if we apply the pre-trained DeBERTa model directly to our evaluation dataset, without fine-tuning, the observed scores exhibit a skew towards lower values, specifically around 0 (Fig.6). If the vast majority of our data is considered bad by the model, then we have a distribution of scores that is totally skewed towards 0, which could explain the results when we apply the sigmoid afterwards.

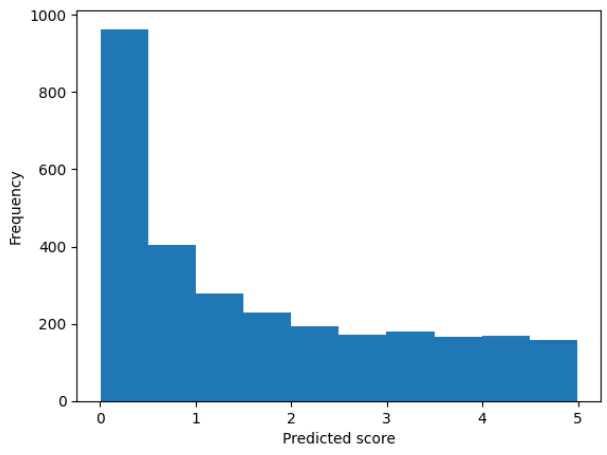

Figure 6: Distribution of score predicted by the pre-trained reward model with no fine-tuning and for the test dataset

If our training data contains a lot of scores with 0, it can contribute to the observed behavior where applying a sigmoid function to the scores results in a narrow range of values between 2.2 and 3.2. The sigmoid function squashes the input values between 0 and 1, with values closer to 0 being compressed towards 0 and values closer to 1 being compressed towards 1. If a significant portion of your original scores are close to 0, applying the sigmoid function will further compress those scores towards 0. This can result in a narrow range of values in the transformed scores, such as the range between 2.2 and 3.2 that we are observing. A solution to this could have been to normalize the scores, instead of using a sigmoid function, or to use a different transformation for instance.

In conclusion, the datasets used do not seem at all suitable for this type of method. The labels are not at all varied and are almost only 0 or 5, and the basic model considers the majority of our data as bad. So when we apply our sigmoid and use the MSELoss, we end up with a completely irrelevant and useless model. This suggests that working 'absolutely' as we do now is not the right solution. To resolve these different biases, the best solution seems to be to start with a much more 'relative' method. For example, using a comparison of solutions instead of working with absolute scores/labels could solve our problem. For this reason, we're going to use a different method for the final rendering.

For the final, we are going to change the structure of our reward model to make a model based on the same principle as the InstructGPT models. We are going to keep the DeBERTa pre-trained reward model but train it using pairwise comparisons. The model's input will therefore be a solution and a less good solution to the same question. The ranking between the two solutions will correspond to the label. We will then use the DeBERTa model to assign a score r to each of the solution. Then the loss will be the same as in the *Training language models to follow instructions with human feedback* paper. The authors of this paper use a cross-entropy loss, with the comparisons as labels—the difference in rewards represents the log odds that one response will be preferred to the other by a human labeler:

$$\text{loss}(\theta) = -\frac{1}{\binom{2}{K}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

K is the number of solutions for one question. With our dataset, a K of 3 seems to be the better solution in our case. One point for the training is that the authors says that they group the comparisons from each prompt into batches to avoid overfitting and to be more computationally efficient. We will have to see if we have enougth memory to train with batches of size 3. This solution seems to be the better way to solve our previous problem. Moreover this will also aid in mitigating the impact of the variability of the CS-552 dataset confidence, which was chosen by various students.

Another point that we will improve is the dataset. The EVILdolly dataset is of course no longer suitable, as it only presents one bad solution per question. We are also going to withdraw the truthful_qa dataset, which deals with subjects that are too diverse and too far removed from the initial objective, which is the subject of courses at EPFL. We will therefore keep only the CS-552 dataset and the SciQ dataset.