# Modern NLP M2 Report

Team UniGPT

June 2023

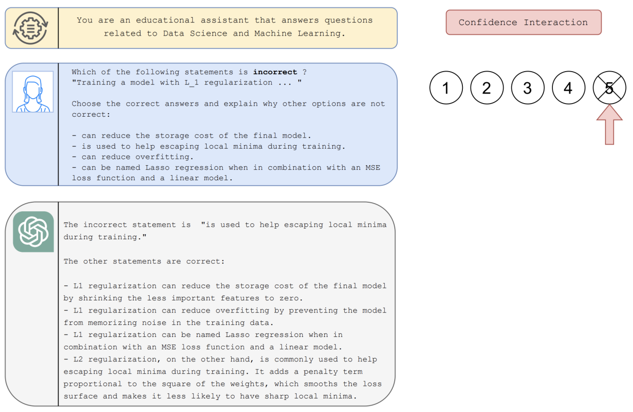
## 1 Introduction

One year ago, the state of Natural Language Processing was substantially different from what we see nowadays. ”Old” models like GPT3, Megatron-Turing, and others, despite generating good text, weren’t good enough to make people use them. And now with the release of ChatGPT a Language Model that integrated Reinforcement Learning with Human Feedback(RLHF) in the fine-tuning process stated in [2], was the start of a new era on what could be called ”the state of GPT”. This new novel method of RLHF involves three general steps: data collection, the creation of a Reward Model to evaluate interactions, and using the RM to train a Language Model to Generate high-quality responses.

The project is divided into three Milestones, following the Instruct GPT ”pipeline”. In Milestone 1, each group was required to interact with ChatGPT 100 times by posing a specific prompt technique question related to an EPFL Courses. This resulted in the collection of a database of 10,385 interactions. After, in Milestone 2, the task is to separate all the interactions collected to train a reward model that will be used in Milestone 3. During milestone 2, we faced some challenges processing the data, such as imbalanced data and very long interactions. To overcome these issues, we proposed three data augmentation techniques for the interactions. Additionally, we had some limitations with computational resources due to normal LLMs being trained on several GPUs. So when we select the Pretrained model we had these in mind, and finally, we used DistilBERT.

## 2 Data preprocessing

The initial dataset, collected in Milestone 1, consisted of 10,835 interactions. Each interaction had the question ID and confidence score from 1 to 5 (from very bad to very good), the format can be seen in Fig. 1. During the analysis of the dataset, first, we cleaned the interactions with invalid scores such as -1, -4, 6, and 45 and kept scores 0-5. After that, we converted the data into the required format by merging ”system”, ”user”, and ”assistant” entries into one ”chat”. Also, to follow the accepted/rejected method as in [2], we binarized the labels such that chats with confidence scores 0-3 have label 0 (rejected-bad), and those with confidence scores 4-5 have label 1 (accepted-good). The obtained dataset of 10830 samples was used as the initial dataset and part of it served as a test set for all further experiments.



You are an educational assistant that answers questions related to Data Science and Machine Learning.

```
Which of the following statements is incorrect ?
"Training a model with L_1 regularization ... "

Choose the correct answers and explain why other options are not
correct:

- can reduce the storage cost of the final model.
- is used to help escaping local minima during training.
- can reduce overfitting.
- can be named Lasso regression when in combination with an MSE
loss function and a linear model.
```

```
The incorrect statement is  "is used to help escaping local minima
during training."

The other statements are correct:

- L1 regularization can reduce the storage cost of the final model
by shrinking the less important features to zero.
- L1 regularization can reduce overfitting by preventing the model
from memorizing noise in the training data.
- L1 regularization can be named Lasso regression when in
combination with an MSE loss function and a linear model.
- L2 regularization, on the other hand, is commonly used to help
escaping local minima during training. It adds a penalty term
proportional to the square of the weights, which smooths the loss
surface and makes it less likely to have sharp local minima.
```

Figure 1: Example Interaction

The main challenges of using this initial dataset for training are long sequences and label imbalance. Long sequences may cause a loss of information after tokenization since most of the models have a maximum of 512 tokens. Indeed, 44.8% of the initial interactions had a length bigger than that. Also, the data imbalance ended in that our initial models learn a constant value for any input corresponding to the overrepresented class. That in the initial case was the label 1 To address these issues, we proposed these three data augmentation techniques to avoid both problems (Fig. 2):

1. **Splitting the interactions by "System" instructions.** We noticed that some interactions are rather long and they have not only several "Human" - "Assistant" entries but also more than one "System" entry, i.e. instruction. We divided these interactions into smaller ones based on the system instructions, such that each new interaction has only one instruction (but still can have more than one "Human" - "Assistant" entry).

2. **Data augmentation with synonyms replacement.** To tackle data imbalance in label 1, we augmented samples with label 0 by replacing random words with synonyms. We experimented with different rate words: 15, 50, and 200 words to replace.

3. **Data augmentation by paraphrasing with ChatGPT.** Alternatively, we augmented samples with label 0 by asking ChatGPT to paraphrase the "Assistant" part of those interactions. This approach is more promising because it seems to be a more logical "rewrite" of a sentence, than just synonym replacement.

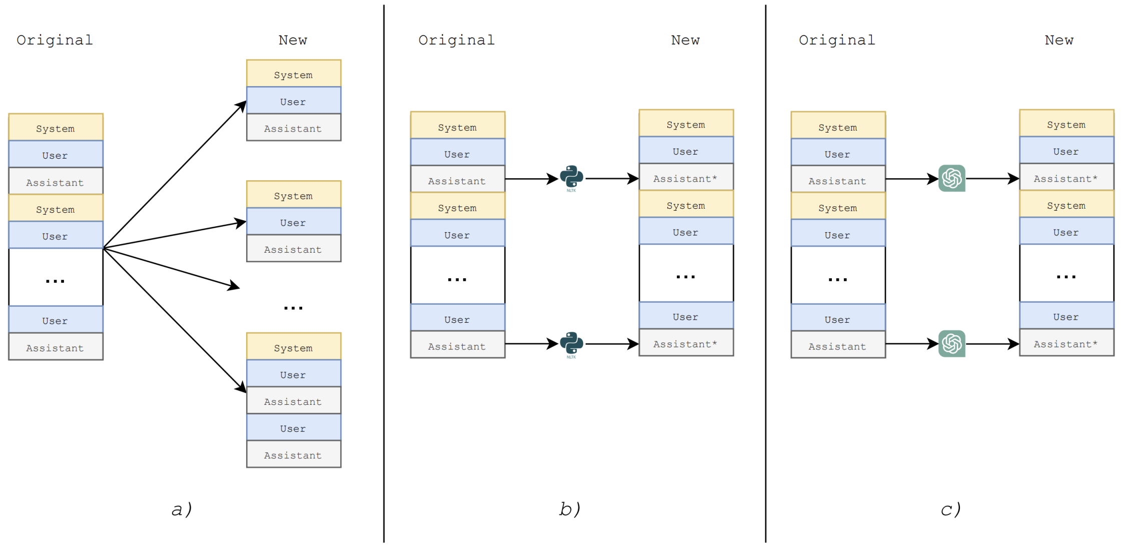


Figure 2: a) Splitting interactions by System Instructions, b) Augmentation with synonyms replacementes, c) Augmentation by paraphrasing with ChatGPT

## 3 Experimental setup

### 3.1 Model architecture

We experimented with BioBERT [1] model preptrained on a large set of biomedical articles and DistilBERT [3] model which is a smaller version of BERT with 40% fewer parameters. Both models were used in a "for sequence classification" format with a linear layer on top of the LM head.

### 3.2 Training setting

We stick with the default optimization parameters provided by HuggingFace Trainer function and rather paid more attention to the complex dataset. We used AdamW optimizer and Cross Entropy loss. We experimented with different numbers of training epochs from 3 to 8, however, we soon found that most of our models started to overfit if we train them for too long so the final model was trained for 3 epochs.

### 3.3 Evaluation Procedure

To evaluate the model performance, we used F1 score as the most common metric for classification tasks. Moreover, it is more robust to data imbalance than accuracy metric since it considers both precision and recall. We also monitored the validation loss to find out when the model starts to overfit the training set.

## 4 Results

Our early experiments were with BioBert and DistilBERT on the initial multilabel confidence, but we noticed that they performed poorly, achieving no more than 0.2 F1 score with the problem of constant values. After, the Label binarization helped to substantially improve the results. With this setting, our first experiment on DistilBERT resulted in a F1 score of 0.6583. For the final round of experiments, we tried three preprocessing techniques alone and by mixing some of them. With this, we thought that the solution to the long sequences and imbalanced data could give a better understanding of the interactions to the model.

Table 1: Data Augmentation techniques and DistilBERT perfomance for sequence classification model

| **Data** | **Validation loss** | **F1 score** |
|---|---|---|
| Initial | 0.5832 | 0.6583 |
| Splitted | 0.5557 | 0.5968 |
| Augmented with synonyms | 0.4554 | 0.7969 |
| Augmented with synonyms and splitted | 0.5220 | 0.7573 |
| Augmented with Paraphrased | 0.5767 | 0.6437 |
| Augmented Paraphrased and splitted | **0.4025** | **0.8232** |

In table 1 we can see the validation loss and F1 score for DistilBERT classification model trained on datasets after different preprocessing attempts.

With this, the best performance was achieved by combining the data augmented by paraphrasing with ChatGPT and split interactions (F1 score of 0.8232). The scheme of this augmentation technique is shown in Fig. 3. This dataset was chosen as our **final reward dataset**, it has a size of 16528 samples, and DistilBERT for sequence classification trained on this data was chosen as our **final reward model**. Augmentation by synonyms' replacement also gives a good result on the initial test set (F1 score of 0.7969). In addition to the final dataset, all other datasets we used in our experiments can be found in a `datasets.zip` file in the repository.
We can see that indeed tackling data imbalance and splitting the interactions into shorter sequences help to get better results. We believe that more steps in this direction of data preprocessing can further improve the model.

## 5 Discussion

As can be seen in Table 1, the models do not achieve a low enough validation loss, indicating that more work is needed to create a good reward model. We think that the main issue is a rather challenging dataset with long sequences and multiple Human/Assistant interactions. Also, the labeling was done separately by many students without any common technique. Thus there always will be some noise due to subjectivity which makes it difficult to use such data for training. Other reasons for not good enough reward model are limited time and computational resources.

However, we have some ideas about further data preprocessing/Augmentation to improve the performance. One has an approach to have a dataset similar to "accepted/rejected examples" with the same prompt from the user. For interactions with label 1, you can give ChatGPT the solution and ask it to provide a confusing and wrong solution. This way, we can have the "accepted/rejected" pair. And for label 0, we can do the same by giving ChatGPT the solution to the question and asking it to explain it. Finally, we could have a dataset with all the initial interactions in "accepted/rejected" pairs.

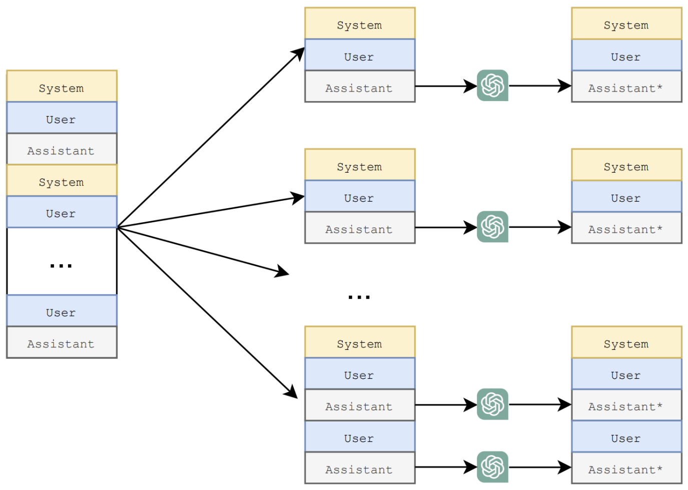


Figure 3: Splitting interactions and Paraphrasing with ChatGPT

## 6 Miscellaneous

Our repository contains two models in the `models` directory. The model in `models/reward_model` is the final model which contains the implementation of `get_reward` function and it should be used with `evaluate.py`. The other model, which is present in `models/classification_model` is the model that we trained. It is used in the file `model.py`. In summary, the model in `models/reward_model` is a wrapper around the model in `models/classification_model`.

## References

[1] Lee Jinhyuk et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (Sept. 2019). Ed. by Jonathan Wren, pp. 1234–1240. DOI: `10.1093/bioinformatics/btz682`.

[2] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[3] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: `1910.01108 [cs.CL]`.

## A Appendix: Regression

In addition to the binary classification described above, we also tried some regression experiments. The general problem we faced with regression was that the model started predicting the same value for all the inputs. We briefly describe our experiments below:

- **Regression experiments with Distilbert model as the base model.**

  We tried a few experiments keeping the Distilbert model as the base model. We added multiple linear layers on top of the base model to finally predict a scalar. We then trained this model on the `interactions_v1.json` dataset provided in the project reference folder. We preprocessed the dataset such that for every question $q$, if it has responses $r_1, r_2$, we join the responses to yield a combined sentence. This combined sentence is assigned a label equal to the difference between the confidence scores of $r_1$ and $r_2$. So finally we get a dataset of the form $(combine(r_1, r_2), c_1 - c_2)$ where $c_1$ and $c_2$ are the confidence scores of $r_1$ and $r_2$. This we way obtained a total of 18552 datapoints.

  We obtained the following results

| Epoch | Train loss | Eval loss | Train accuracy | Eval accuracy |
|---|---|---|---|---|
| 1 | 2.203 | 2.201 | 31.31% | 30.96% |
| 2 | 2.177 | 2.201 | 31.61% | 30.96% |
| 3 | 2.177 | 2.201 | 31.61% | 30.96% |

Table 2: Results for regression experiment using Distilbert

  The loss reported is the MSE loss. The accuracy is calculated as follows: for the dataset, if the model predicts the relative ordering between the two responses $r_1$ and $r_2$, then we count it as a success. We can see the model converges almost after the first epoch itself.

- **Regression experiments with OpenAssistant/reward-model-deberta-v3-base model as the base model.**

  We also tried a few experiments by using the `OpenAssistant/reward-model-deberta-v3-base` model as the base model for producing a scalar output. The difference between this experiment and the experiment described above is that here we do not preprocess the dataset as we did above. Instead we use it as it is: $(r_i, c_i)$ the response confidence pairs. However the confidence values were scaled down to make them between 0 and 1. We obtained the following results

| Epoch | Train loss | Eval Loss |
|---|---|---|
| 1 | 0.080 | 0.066 |
| 2 | 0.066 | 0.066 |

Table 3: Results for regression experiment using OpenAssistant/reward-model-deberta-v3-base as the base model

  As before, the loss being reported is MSE loss.

## B Appendix: Classification as NLI

We also tried another classification experiment, where we modelled this problem similar to NLI. For every question $q$, if it has responses $r_1, r_2, ...$ for every pair of response $(r_i, r_j)$, we add a label $l_{ij}$ such that $l_{ij}$ equals 0, 1 or 2 depending upon $r_i$ has a lower, equal or higher confidence score than $r_j$. The we used the encoding code similar to Assignment 2 to join the responses $r_i$ and $r_j$. We trained this using Distilbert as the base model, and obtained the following results.

| Epoch | Train loss | Validation loss | F1 |
|---|---|---|---|
| 1 | 1.098900 | 1.109295 | 0.204897 |
| 2 | 1.076400 | 1.092894 | 0.386848 |
| 3 | 1.043000 | 1.061519 | 0.327846 |

Table 4: Results of experiment with modelling the problem as NLI

We did not pursue this idea further because the results that we obtained were highly variable. We can see that the F1 decreases significantly from 2nd to 3rd epoch. Such variance was observed across different runs of this experiment, so we did not pursue it further.

The notebook for this experiment can be found here.