Training an AI Assistant for STEM Education

Francesco Salvi, Giacomo Orsi, Roberto Ceraolo
Department of Computer Science, EPFL, Switzerland
{first.last@epfl.ch}

Abstract

The paper presents our project focused on training an AI assistant for STEM education. We highlight the rise of large language models (LLMs) and their potential in solving complex tasks. We note that while generalist models have gained popularity, there is untapped potential in developing specialized models that provide expert assistance in specific domains. We propose a model that specializes in STEM education and can tutor EPFL students by explaining complex material in simple terms.

Our methodology involves collecting a dataset of questions from EPFL exams and Stack Exchange. The questions are used to train a reward model and a generative language model. The reward model is trained to rank answer generations based on criteria such as clarity, correctness, completeness, and rigour. The generative model is fine-tuned using supervised learning to generate clear and detailed answers.

Our results show that the fine-tuned model outperforms the base model in terms of text similarity metrics, indicating improved quality of generated answers. However, the fine-tuned model shows higher deviation from the reward function, suggesting a misalignment with the desired principles defined in the Constitution.

Overall, the paper presents our promising approach to training an AI assistant for STEM education. The focus on specialized models and the use of a Constitution-based AI provide interesting insights into developing expert-level guidance in specific domains. The limitations and avenues for future work discussed by us contribute to the understanding and potential improvement of the proposed approach.

1 Introduction

Large Language Models (LLMs) have seen a tremendous rise over the last few years, showing astonishing capabilities for solving complex tasks such as translation, reasoning, or answering elaborate questions. Online chatbots like ChatGPT (OpenAI, 2022) have significantly popularized and democratized their use, making it easy for anyone to interact with AI and recognize its potential. However, despite the rapid proliferation of models such as OpenAssistant (Köpf et al., 2023), Claude (Bai et al., 2022), Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), Koala (Geng et al., 2023), Dolly (Conover et al., 2023), Falcon (von Werra et al., 2023), and many others, most research efforts so far have only been focused on developing generalist models, that can serve well users with a wide range of needs. While this is an important stepping stone towards Artificial General Intelligence, we believe that there is a wide unexplored potential for models that specialize in specific domains, providing expert assistance to a smaller set of users but with higher reliability. In this project, we focus on the domain of STEM education, developing an AI chatbot that can tutor EPFL students by mastering their course content and explaining complex material in simple and clear terms. To do that, we use questions from EPFL exams to collect a high-quality dataset of demonstrations and preferences, ranking answers produced by ChatGPT (OpenAI, 2022) using ChatGPT itself, instructing it to favor clear and rigorous answers with Constitutional AI (Bai et al., 2022). We then enrich this data with answers from *Stack Exchange* and use it to fine-tune a Reward Model, starting from Microsoft's DeBERTaV3 (He et al., 2023), and a supervised policy, starting from OpenAI's Distilled-GPT-2 (Sanh et al., 2019). We evaluate our fine-tuned model on text-similarity metrics (BLEU, Rouge) and on the learned reward function. Our results show an improvement over text-similarity metrics against the base model.

The paper is organized as follows: we first review existing literature and related work about creation and fine-tuning of large language models. Secondly, we describe our approach to data collection, training of the reward model, and super-

vised fine-tuning of the model. Finally, we show some results we obtained from the evaluation of our Assistant.

2 Related work

Collecting human data Modern Language Models aligned for human dialogue largely follow some form of the 3-step recipe theorized for InstructGPT (Ouyang et al., 2022), i.e. (1) train a supervised policy with demonstration data, (2) train a reward model with comparison data, and (3) optimize the supervised policy with reinforcement learning using the reward model. This process heavily relies on collecting high-quality data for demonstrations and comparisons, a task that is usually carried out by human annotators through crowdworking. Such data collection, for example, is carried out in Dolly (Conover et al., 2023) and LaMDA (Cohen et al., 2022) for the supervised step, in Sparrow (Glaese et al., 2022) for the reinforcement learning step, and in OpenAssistant (Köpf et al., 2023) and ChatGPT (OpenAI, 2022) for both. More recently, however, alternative approaches have been proposed to take humans out of the loop, lowering the cost of data collection and enabling it to happen at a larger scale. Anthropic's Claude (Bai et al., 2022) does so by prompting a model to produce and rank generations following a *Constitution*, i.e. a collection of rules and principles that their Assistant should follow, and through an iterative critique-revision process. Our work directly builds on this approach, developing a Constitution specifically targeted to STEM education.

Imitating proprietary LLMs At the same time, our project fits into the line of research that aims to cheaply improve a weak open-sourced Language Model by finetuning it on outputs from a stronger model, such as a proprietary system like ChatGPT. Alpaca (Taori et al., 2023) and Self-Instruct (Wang et al., 2023) do so by fine-tuning LLaMA (Touvron et al., 2023) with 52k instruction-following demonstrations generated by ChatGPT. Vicuna (Chiang et al., 2023) and Koala (Geng et al., 2023) use user conversations with ChatGPT shared on the web.

3 Approach

3.1 Data Collection

The data collection step is a key part of our methodology and constitutes the most original piece of our project. We collect a novel dataset based on questions from EPFL exams, which we thus call *EPFL dataset*, and we complement it with a sample of questions from *Stack Exchange*. For each question in both, we produce a ranking of answers, that we then use either as supervised demonstrations or pairs of preferences for the reward model. The present section only discusses the collection of such rankings, while the process of partitioning the datasets and casting them in the correct form for the subsequent steps of our pipeline is detailed in Section 4.1.

EPFL dataset The core of our training data is made of questions from EPFL exams, which have been collected across a variety of STEM subfields across campus. For the course project of *CS 552: Modern Natural Language Processing*, hundreds of students have worked with different samples of such questions with the task of prompting ChatGPT to come up with the best possible answers for their sample. The results of this effort form a dataset of 10835 interactions distributed over 4500 questions, which is where we start to gather our training data. We replicate the method of *Constitutional AI* (Bai et al., 2022), writing a constitution targeted to STEM education that encapsulates the key principles and rules that we want our Assistant to follow. First, we remove all the questions without a ground-truth solution or with only one answer, which is not enough to produce a ranking. Then, we process the answers to *standardize* them across questions, combining them with a common prompt. The key challenge in this process, in fact, is that the strategies followed by students are vastly different, with a plethora of possible prompts, styles of collecting answers, and numbers of iterations. To create fair comparisons, we instead want generations for the same question to share a unique prompt, and only differ in how they answer it. To simplify this process, we assume that the last output with $role = Assistant$ from each interaction with ChatGPT contains an answer to the exam's original question, regardless of the student's prompt. We empirically verify that this is true in a majority of cases, because even when the prompting strategy involves multiple rounds of iterative reasoning, usually the last user prompt asks to summarize everything in a unique answer. We observe, however, one recurrent pattern where this is not true, corresponding to students asking the model to generate a

confidence score as their last instruction. Therefore, we filter out all the interactions where the last user instruction contains the word "confidence". Then, we artificially produce new demonstrations from those outputs by assuming that they all answer a basic prompt that just repeats the exam's question and, if present, the multiple-choice options. We emphasize that this is done because our goal is not to train an instruction-following Assistant, but rather one that can be helpful in the domain of STEM education. In other words, we are not primarily interested in making sure that our Assistant follows precisely the complicated instructions contained in the prompts crafted by students, but rather we want it to perform well in closed-book question-answering, producing accurate, complete, and clear answers for exam-like questions that an end user might ask, doing so without needing such articulate prompting strategies. Finally, we query ChatGPT to rank the extracted answers based on our Constitution and the ground-truth solutions, with the following prompt:

> Consider the following question from a scientific exam:
> QUESTION: {{QUESTION}}
>
> These are the available options (more than one can be correct!):
> - {{OPTION 1}}
> - {{OPTION 2}}
> - ...
>
> Now consider the following answers, produced by AI Assistants:
> - {{DEMONSTRATION 1}}
> - {{DEMONSTRATION 2}}
> - ...
>
> Finally, here is the correct answer, as written in the exam solutions:
> SOLUTION: {{SOLUTION}}
>
> Your task is now to produce a ranking of the AI answers, based on the following parameters:
> - Correctness (!!!). The answer should generally correspond to the exam solution.
> - Clarity. The answer should be clear and easy to understand by students.
> - Completeness. The answer should cover all the relevant aspects of the question.
> - Rigour. The answer should be rigorous and precise, following the scientific standards and logical reasoning.
> - Fluency. The answer should be fluent and well-written.
> - Coherence. The answer should be self-consistent and coherent, without referring to ambiguous external sources.
> - Harmlessness. The answer should not contain any harmful or unethical content.
>
> Notice that correctness is the most important parameter, and it should be prioritized over the others. Please only output a sequence of {{N_DEMONSTRATIONS}} numbers, corresponding to the indices of the AI answers, from the best to the worst. {{EXAMPLE}}
> RANKING:

Where the green text is only included for multiple-choice questions, and {{EXAMPLE}} corresponds to an example output given the number of demonstrations. For example, with 3 demonstrations the example is "For example, the sequence '312' means that the best answer is the third one, followed by the first one, and finally by the second one." Notice that our constitution insists on the characteristics that we want our Assistant to have, and makes completely transparent the criteria for producing the rankings. From an initial set of 4500 questions (10835 demonstrations), the filtering steps described previously leave us with 3111 (8711 demonstrations). Out of them, we obtain valid rankings from ChatGPT for 2836 questions (7752 demonstrations), corresponding to a total of 7568 pairs.

Stack Exchange We augment our data with the *Stack Exchange Preferences* (Lambert et al., 2023) dataset, which contains questions and answers from the Stack Exchange network. Since the dataset is very large, and we are only interested in domains that are related to STEM education, we filter only for a small number of substacks corresponding to the main STEM areas, that is: *Computer Science*, *Computer Science Theory*, *Cognitive Science*, *Biology*, *Bioinformatics*, *Engineering*, *Electronics*, *Economics*, *Earth Science*, *Math*, *Mechanics*, *Quantitative Finance*, *Quantum Computing*, *Physics*,

Robotics, *Statistics*, and finally from *StackOverflow*. We subsample each of those substacks to get approximately the same number of questions for each, resulting in a total of 69959 questions. Each answer in the filtered data comes with a score that reflects its number of updates, plus a bonus if the answer has been "accepted" by the user who posted the question. We, therefore, use those scores to produce a ranking of the answers for each question. We argue that the augmentation with this dataset is particularly relevant not only to increment the size of our data but also to provide the model with negative examples, i.e. bad generations, since the quality of demonstrations in the *EPFL dataset* is generally high.

3.2 Reward model

We cast the problem of training a reward model as a modified regression, adopting the training objective defined in Ouyang et al. (2022). That is, given a model r_θ parametrized by θ which takes in text generations y to output a scalar score, we define the loss function as:

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

Where (y_w, y_l) is a pair of winning and losing generations, D is our collected dataset of preferences, and σ indicates a sigmoid function.

Architecturally, we fine-tune a pre-trained language model and add a single linear layer on top of it, replacing the language modeling mask to predict a scalar score. As starting model, we decided to adopt `Microsoft/deberta-v3-base` (He et al., 2023), an 86M-parameter version of DeBERTa trained by Microsoft on a variety of text corpora.

3.3 Supervised fine-tuning

The main goal of supervised fine-tuning in our setting, is to teach the pretrained language model to give clear and detailed answers to the questions, to be as helpful as possible to students. In other words, we want its answers to follow closely the principles of our *Constitution*. We start from a pre-trained Distilled-GPT-2 (Sanh et al., 2019) model. It is an English-language model pre-trained with supervision, and was created using knowledge distillation and is a lighter and faster version of GPT-2 (Radford et al., 2019).

We fine-tune it on the task of generating answers to STEM questions. As loss function, we use Cross Entropy, and we train the model for 10 number of epochs epochs with a batch size of 4. We use the Adam optimizer (Kingma and Ba, 2017) with a starting learning rate of 1e-5, and we use a scheduler that keeps the learning rate constant, after an initial warmup period in which the rate increases linearly between 0 and the initial value. We use the HuggingFace `transformers` library (Wolf et al., 2020) for the implementation of the model and the training procedure. We are able to train in a supervised fashion because for both data sources, EPFL and StackExchange, we have a "Ground truth" answer.

The input given to the model during training is the exchange between the human and the assistant. The human question is marked The model learns autoregressively to predict a token given the previous ones. We apply masking to the label by "hiding" the prompt. This way, we make the model only learn to generate the answers, that come after the keyword "Assistant:".

4 Experiments

4.1 Data split

We partition both our collected datasets into 4 splits, which were used for the different parts of our project:

- The training dataset for the Reward model - 30% of the full datasets: It contains preferences in the form of (winning, losing) pairs.
- The training dataset for Supervised Fine-Tuning - 30%: It contains demonstrations in the form of (prompt, best_generation)
- The training dataset for Reinforcement Learning from Human Feedback - 30%: It only contains prompts.
- The test dataset - 10% of the full datasets: It was used to evaluate both our reward model and our chat Assistant, at each stage of its training.

The sampling was always done at the level of questions and not of demonstrations. For the EPFL dataset, we sample the first training dataset and the Test set from the set of questions for which our Constitutional prompt fits the ChatGPT's context size (4096 tokens), and for which ChatGPT managed to produce a valid ranking. For the other two sets, we sample from all the remaining questions, since we don't need any rankings.

4.2 Evaluation method

For the evaluation, we used several metrics to assess the performance of our models. These metrics include ROUGE, BLEU, and the output of the reward model. ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002) are commonly used evaluation metrics for text generation tasks, measuring the similarity between the generated text and the reference text. So we compute the average values on the test datasets, separately for the EPFL and the StackExchange cases. Secondly, in a previous phase of the project we trained our Reward model on rankings following the principles of our *Constitution*, so we decided to create a metric for evaluation based on such model. We compute the average reward score given by our fine-tuned reward model both on the Assistant's generations and on the Ground truth answers. Then, we check the difference this two aggregate values. We claim that such a difference is indicative of how closely our Assistant model match the criteria the reward model was trained on. In order to understand whether our fine-tuning process improved the model, we use the base model as a baseline to compare the metrics. Ideally, we would also want to compare to alternative models fine-tuned for the same task but most previous work entails the presence of a *context* or the models are very large, which makes the comparison unfair.

In table 1 we compare the results between:

- Our model, Distil-GPT-2 fine-tuned as described in Section 3.3
- Pretrained Distil-GPT-2 model, before any fine-tuning

4.3 Experimental details

In this section we report some relevant technical details of the models that we have trained, in order to ensure repeatability of results.

Reward model

In order to train the reward model we mix data from the training sets of the *EPFL dataset* and the *Stack Exchange dataset*, interleaving them to increase variety and stability. We then evaluate the model's performance separately on the respective test sets. We use an AdamW (Loshchilov and Hutter, 2019) optimizer with standard parameters and a linear warmup period equal to 20% of our training data, with a batch size of 4. To prevent overfitting and stabilize the training, we apply gradient clipping with a max norm of 1 and we add a L2-regularization penalty to the loss defined in Equation 1. In our first experiment we have chosen a learning rate of 1×10^{-5} and the regularization parameter $\beta = 0.001$. During training, we noticed a strong tendency of the model to overfit the training data, with the validation curve quickly reaching a minimum after the first epoch. This seems to be consistent with the literature, with the InstructGPT (Ouyang et al., 2022) paper reporting similar overfitting behavior after only one epoch. In a second experiment we have reduced the learning rate to 1×10^{-7} and increased the regularization parameter to $\beta = 0.01$. The training loss decreases more slowly and the validation loss keeps decreasing, but never reaches a value as low as the minimum reached in the first experiment. Therefore, as our final model we decided to adopt the one trained in the first experiment after the first epoch, hence applying early stopping to recover the best checkpoint on the validation set.

Supervised fine-tuning

For supervised fine-tuning, we started with a pretrained Distil-GPT-2 model and fine-tuned it on the task of generating answers to STEM questions. The loss function used was Cross-Entropy, and the model was trained over 10 epochs with a batch size of 4. The best model was selected based on the test loss, which reached its minimum at the fifth epoch. Figure 1 shows a breakdown of the loss curves for the fine-tune learning. We have used a scheduler for the learning rate which keeps it constant, after a warmup period in which the learning rate increases linearly between 0 and a value set at 1e-05. The optimizer we used is Adam (Loshchilov and Hutter, 2019).

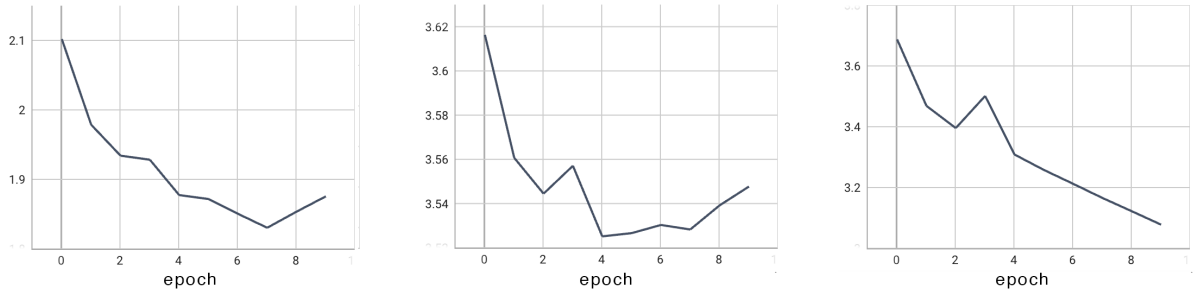
4.4 Results

The results of our experiments are presented in Table 1, which compares the performance of the DistilGPT-2 base model with the fine-tuned model on both the EPFL dataset and the Stack Exchange dataset. We evaluated the models using several metrics, including BLEU and Rouge scores for text similarity, as well as the reward deviation metric, which measures the difference between the reward function on the gold label and the generated sequence.

From the results, we can observe that the fine-tuned model generally outperforms the base model in terms of text similarity metrics (BLEU and Rouge), indicating that the fine-tuning process im-

	BLEU	Rouge-1	Rouge-2	Rouge-l	Reward deviation
DistilGPT-2 base, EPFL	0.0024	0.0115	0.0023	0.0077	0.4169
DistilGPT-2 finetuned, EPFL	0.0029	0.0150	0.0034	0.0099	1.0271
DistilGPT-2 base, StackExchange	0.0004	0.0135	0.0016	0.0077	0.5164
DistilGPT-2 finetuned, StackExchange	0.0006	0.0169	0.0022	0.0100	0.7674

Table 1: Comparison of the results obtained with the DistilGPT-2 model before and after fine-tuning, on the EPFL and StackExchange datasets.



(a) Test loss on EPFL (b) Test loss on Stack (c) Train loss

Figure 1: The above graphs show the trend of the loss during fine-tuning of Distil-GPT2. It can be seen that at around epoch 7 the model starts overfitting the training data, since the training loss is steeply decreasing but the test one is increasing. We choose to retain the checkpoints at epoch 4, since at that point the model achieved the minimum on the StackExchange dataset, and was close to the minimum for the EPFL dataset.

proves the quality of the generated answers. However, interestingly, we also see an increase in the reward deviation metric for the fine-tuned model. This suggests that although the fine-tuned model performs better in terms of text similarity, it does not necessarily converge to answers that follow the Constitution learned by the reward function.

5 Conclusion

In conclusion, we have presented an approach to developing an AI chatbot specialized in providing expert assistance in STEM education. Our methodology involved collecting a high-quality dataset of questions from EPFL exams and Stack Exchange, and using these questions to fine-tune a reward model and a generative large language model. We trained the reward model to rank answer generations based on clarity, correctness, completeness, rigour, fluency, coherence, and harmlessness.

Our approach shows promise in improving the performance of AI chatbots in STEM education by providing higher reliability and expert-level guidance. The fine-tuned model demonstrated better performance in terms of text similarity metrics, indicating improved quality of the generated answers. However, after fine-tuning the model showed worse performance over the reward function we trained, suggesting that the fine-tuned model did not converge to the principles defined in our constitution.

As the reward function was not used in the fine-tuning process, but it was only used for evalution, future work could explore the integration of Reinforcement Learning from Human Feedback (RLHF) techniques to optimize the model over the reward function. This would enhance the system's adaptability and alignment with the desired principles of the constitution.

Additionally, incorporating user feedback and conducting more extensive evaluations with human users would be valuable for further improving the system. By actively involving end-users in the training process, we can ensure that the AI chatbot meets their specific needs and preferences.

In summary, our work lays a foundation for developing AI assistants that provide expert-level guidance in STEM education. While there are areas for improvement, our approach demonstrates the potential of specialized AI models in enhancing the learning experience and supporting students in their educational journey.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christo-

pher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung ching Chang, Claire Cui, Cosmo Du, Daniel De Freitas Adiwardana, Dehao Chen, Dmitry (Dima) Lepikhin, Ed H. Chi, Erin Hoffman-John, Heng-Tze Cheng, Hongrae Lee, Igor Krivokon, James Qin, Jamie Hall, Joe Fenton, Johnny Soraker, Kathy Meier-Hellstern, Kristen Olson, Lora Mois Aroyo, Maarten Paul Bosma, Marc Joseph Pickett, Marcelo Amorim Menegali, Marian Croak, Mark Díaz, Matthew Lamm, Maxim Krikun, Meredith Ringel Morris, Noam Shazeer, Quoc V. Le, Rachel Bernstein, Ravi Rajakumar, Ray Kurzweil, Romal Thoppilan, Steven Zheng, Taylor Bos, Toju Duke, Tulsee Doshi, Vincent Y. Zhao, Vinodkumar Prabhakaran, Will Rusch, YaGuang Li, Yanping Huang, Yanqi Zhou, Yuanzhong Xu, and Zhifeng Chen. 2022. Lamda: Language models for dialog applications.

Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. Blog post.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. 2022. Improving alignment of dialogue agents via targeted human judgements.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment.

Nathan Lambert, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush. 2023. Huggingface h4 stack exchange preference dataset.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

OpenAI. 2022. Introducing chatgpt.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS $EMC^2 Workshop$*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca:

An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Leandro von Werra, Younes Belkada, Sourab Mangrulkar, Lewis Tunstall, Olivier Dehaene, Pedro Cuenca, Philipp Schmid, and Omar Sanseviero. 2023. The falcon has landed in the hugging face ecosystem. Blog post.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.