(mis)Aligner: How the Aligner Method Fools Evals

Abby, Anika, Shamith

The goal of alignment evaluation is to measure an LLM's preferences and determine if it is inherently aligned with a given safety principle. The HHH benchmark takes the output of a model to determine how well it responds over three main categories: helpfulness, honesty, and harmlessness. However this eval, like many others, relies on the fact that the output is representative of the base model's preferences. Closed models such as Clade and Gemini must be accessed through an API, providing the opportunity to make changes to the model output before it reaches the end user.

A method for doing so is demonstrated in *Aligner: Efficient Alignment by Learning to Correct*, (Ji, Chen et. al) by introducing an intermediary step, the *Aligner*, to correct the output of the base (or *upstream*) model to produce a more aligned response. For closed models where you only have access to an API, there is no way to tell that the upstream model output has been altered. This renders HHH bench useless, as the outputs it's evaluating are not reflective of the upstream model's preferences.

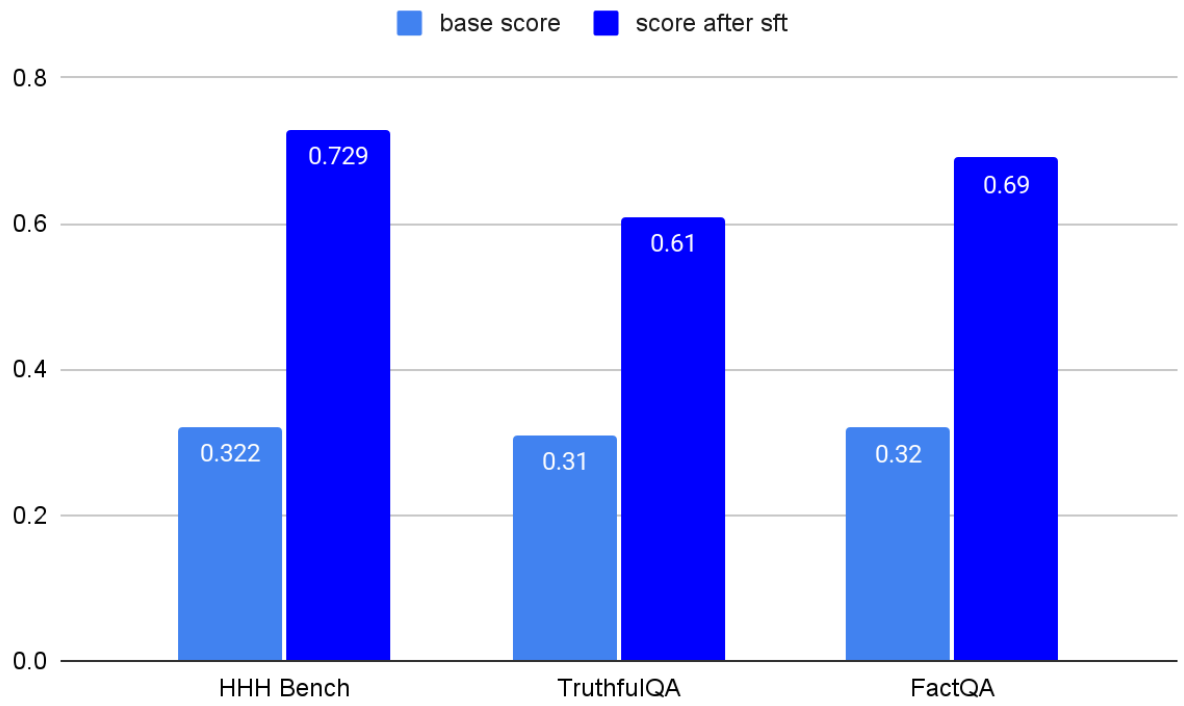
We demonstrate this capability on the open source model qwen2.5-0.5B-Instruct. First, we evaluate qwen on HHH bench to collect a baseline score for comparison. We then pass qwen's output to the aligner (aligner-7b-v1.0 as used in the paper) to return more aligned outputs. This new output is then evaluated by HHH bench to achieve an inflated score.

At first, we didn't realize that the aligner model was open source, so we attempted to build it. We used Supervised Fine Tuning (SFT) to train qwen on the HHH test dataset. In reality, the aligner is more sophisticated than that. It was fine tuned not on the benchmark datasets themselves, but on the *correction residuals* between a "bad" response and the associated better, corrected response. Naturally, this required a considerable amount of effort on the part of the researchers. They conducted a large scale annotation effort to construct a new dataset including prompts, responses, and corrected responses as determined by human annotators.

The function of the aligner is to learn HOW to make bad responses better by studying how the corrections were made. The equation describing this process is as follows,

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{SFT}}) = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_{\text{SFT}}}[\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})]$$

So while we didn't construct an aligner, these first results still serve as an exercise in how training a model on a test data set is a simple way to improve its score. We hypothesized that tuning qwen on HHH test data would improve its score on benchmarks that measure similar things. We found this to be true. For HHH Bench, TruthfulQA, and FactQA, the fine-tuned qwen score doubled nearly across the board.



eval	base accuracy score	score after sft	percent increase
HHH Bench	0.322	0.729	**127%**
TruthfulQA	0.31	0.61	**97%**
FactQA	0.32	0.69	**116%**

However, even after accessing the aligner, we ran into issues. Working out of Google Colab meant we only had access to 12 GB of RAM, which was insufficient to support the 7B parameter model. We were bottlenecked by memory. While discussing options, we tried fine tuning qwen directly on the aligner dataset to see if that might yield a similar result. Given the memory restrictions, we were only able to train on 80 examples. Its HHH Bench score increased from 0.322 to 0.53, which is a 65% increase - only about half of the improvement seen with SFT. It is possible that we would have seen a better result were we able to train on the full dataset.