How does DPO affect text generation

Owner	Ⓐ Andrii Cherniak
Tags	

1. Main purpose of this work:
    a. confirm the ability to load DPO-aligned LLM from a checkpoint (as base llm + QLoRA adapter),
    b. identify any technical issues with quantization
    c. `[batch]` generate answers for questions with tuned and base LLMs
    d. quantify the results aka prove that the alignment took place and tuned LLM generates better answers than base LLM
2. Azure Budget spent: ~ equivalent 57 hours using A100 = $228

Experiment motivation

In the previous report, I was analyzing the convergence of DPO training of `Mistral-Instruct-v0.2` on a 12K `orca_dpo dataset`, where 90% of randomly chosen dataset records were used for training and the remaining 10% - for validation, and prompt was formatted using a version of a chat template `prompt=chat template(system, question)` for `Mistral-Instruct-v0.2`

As a reminder, each tuple from the dataset for DPO alignment consists of 3 fields `[{prompt, chosen, rejected}]` where `chosen` and `rejected` are potential answers to the question presented with `prompt`. The goal of the alignment procedure is to fine-tune an LLM in such way that the probability of `chosen` answers will be higher than the probability of the corresponding `rejected` ones.

Checkpoint with training parameters **[lr=1e-5, batch_size=8, gradiant_accumulation_steps = 8]** most certainly converged without issues and was selected for the evaluation. Both training and validation accuracies were reported of 100%, e.g. after the alignment the LLM would consistently assign higher probabilities to `chosen` answer than to the `rejected` pair.

However, the task that we care the most at the moment is question answering, particularly the tone of those answers. Since text completion is more complex than assigning a probability to a sequence, and intuitively that 100% validation accuracy does not convince me that the answers will be very much aligned with the language of the `chosen` answers. I am more convinced that accuracy alone cannot tell the entire story of DPO alignment and we also need to consider (somehow) margins between `chosen` and `rejected` log probabilities: the larger the margin, the better the alignment should be. Which poses a great question of how to confirm that the alignment took place.

Evaluation methodology: challenges and thoughts

Current DPO alignment work was inspired by Intel neural-chat model. However they use `Mistral-7B-v0.1` instead of `Mistral-7B-Instruct-v0.2`. Their LLM is being first fine-tuned on `Open-Orca/SlimOrca` data set and then aligned on `Intel/orca_dpo_pairs` where I only do alignment on `Intel/orca_dpo_pairs`.

The original DPO paper https://arxiv.org/abs/2305.18290 was aligned on 161K `Anthropic/hh-rlhf` dataset which again is not `Intel/orca_dpo_pairs`. Moreover `Anthropic/hh-rlhf` was constructed specifically to make LLM more helpful and harmless. Can I say the same about `Intel/orca_dpo_pairs`? Probably not. Thus we cannot expect an LLM aligned on `Intel/orca_dpo_pairs` will miraculously become more helpful.

If I do my LLM evaluation on some of the datasets reported in neural chat or DPO paper, and the performance numbers are different, does this mean that there is a bug in code, is this a case of overfitting / catastrophic forgetting, or maybe those differences are expected due to LLM/prompt /dataset differences?

The purpose of LLM alignment is to allow LLM generate responses that humans would prefer. Thus if dpo dataset reflects those preferences, then the change tone between aligned and base LLM needs to be correlated with those preferences from the data set.

In the original DPO paper https://arxiv.org/pdf/2305.18290.pdf authors mention that for real world scenarios where true ground truth reward function is unknown, they used GPT-4 as a proxy for human evaluation to `compute win rate against base` `model` using quality and helpfulness metrics. And the reason they relied on those metrics because the dataset `Anthropic/hh-rlhf` is designed to show what helpful and harmless answers should look like.

Thus in my evaluation I will try to measure `(statistically significant ) difference between tuned and base LLM` on dimensions for which there is `(statistically significant) differences in the alignment dataset` and not purely on the leaderboard scoring.

Evaluation summary

I completed my preliminary evaluation without using LLMaJ primarily due to cost-per-call and inference delay. Instead I used to compute basic test statistics that are broadly labeled as `['quality', 'readability', 'descriptive_stats', 'dependency_distance', 'pos_proportions', 'information_theory', 'coherence']` categories . With 20 concurrent threads, https://github.com/HLasse/TextDescriptives took ~ 1 hour to score 12k answers.

With my experiments, I show that,

1. there is a statistically significant difference between `chosen` and `rejected` answers in the training and validation data sets according to text statistics, which means DPO actually has something meaningful to optimize for
2. this is a statistically significant difference between answers generated with aligned and base LLMs on both training and validation data.
3. And those differences are sign-correlated with the ones between `chosen` and `rejected` answers on most of the metrics. In other words, if on average `chosen` answers have lower entropy compared to `rejected` counterpart, then tuned LLM should show reduction in answer entropy compared to base model.

Metrics description

1. information theory : entropy, perplexity, per_word_perplexity.
2. coherence : first_order_coherence, second_order_coherence: based off cosine similarity between sentences
3. readability : flesch_reading_ease, flesch_kincaid_grade, smog, gunning_fog, automated_readability_index, coleman_liau_index, lix, rix

4. descriptive stats: token_length_mean, token_length_median, token_length_std, sentence_length_mean, sentence_length_median, sentence_length_std, syllables_per_token_mean, syllables_per_token_median, syllables_per_token_std, n_tokens, n_unique_tokens, proportion_unique_tokens, n_characters, n_sentences
5. dependency distance : dependency_distance_mean, dependency_distance_std prop_adjacent_dependency_relation_mean prop_adjacent_dependency_relation_std passed_quality_check
6. quality: n_stop_words, alpha_ratio, mean_word_length, doc_length, symbol_to_word_ratio_#, proportion_ellipsis, proportion_bullet_points, duplicate_line_chr_fraction, duplicate_paragraph_chr_fraction, duplicate_ngram_chr_fraction_5, duplicate_ngram_chr_fraction_6, duplicate_ngram_chr_fraction_7, duplicate_ngram_chr_fraction_8,, duplicate_ngram_chr_fraction_9, duplicate_ngram_chr_fraction_10, top_ngram_chr_fraction_2, top_ngram_chr_fraction_3 top_ngram_chr_fraction_4
7. oov_ratio
8. part of speech : pos_prop_ADJ, pos_prop_ADP, pos_prop_ADV, pos_prop_AUX, pos_prop_CCONJ, pos_prop_DET, pos_prop_INTJ, pos_prop_NOUN, pos_prop_NUM, pos_prop_PART, pos_prop_PRON, pos_prop_PROPN, pos_prop_PUNCT, pos_prop_SCONJ, pos_prop_SYM, pos_prop_VERB, pos_prop_X

Evaluation methodology

For brevity, let

- `tuned --> DPO-aligned Mistral-7B-Instruct-v0.2`
- `base  --> original Mistral-7B-Instruct-v0.2`

1. For each tuple `{prompt, chosen, rejected}` from both training and validation splits, I generate `answer=llm(prompt)` with both `tuned` and `base` llm
2. for `chosen, rejected, tuned_answer, base_answer` compute all test statistics

3. obtain tuple `T_i = {prompt, chosen, rejected, tuned_answer, base_answer, [chosen_features], [rejected_features], [tuned_features], [base_features]}`, where `[..._features]` is a 69 dimensional vector
4. For each `feature_j` and for each tuple `T_i` (for training and validation) compute the following statistics: (basically pairwise normalized difference between chosen and rejected answers' features and pairwise difference between features for answers generated with tuned and base llm)
    a. `(chosen_feature_ij - rejected_feature_ij)`
    b. `(tuned_feature_ij - base_feature_ij)`
    c. ignore entire rows where at least one of the `feature_j` values is `NaN`
    d. `S_j = std(chosen_feature_ij - rejected_feature_ij)` on the training data set - use as a common denominator to normalize feature scale .
    e. `train(chosen - rejected)_j = avg(chosen_feature_ij - rejected_feature_ij)` over training data
    f. `valid avg (chosen - rejected)_j = avg(chosen_feature_ij - rejected_feature_ij)` over validation data
    g. `train avg(tuned - base)_j = avg(tuned_feature_ij - base_feature_ij)` over training data
    h. `valid avg(tuned - base)_j = avg(tuned_feature_ij - base_feature_ij)` over validation data

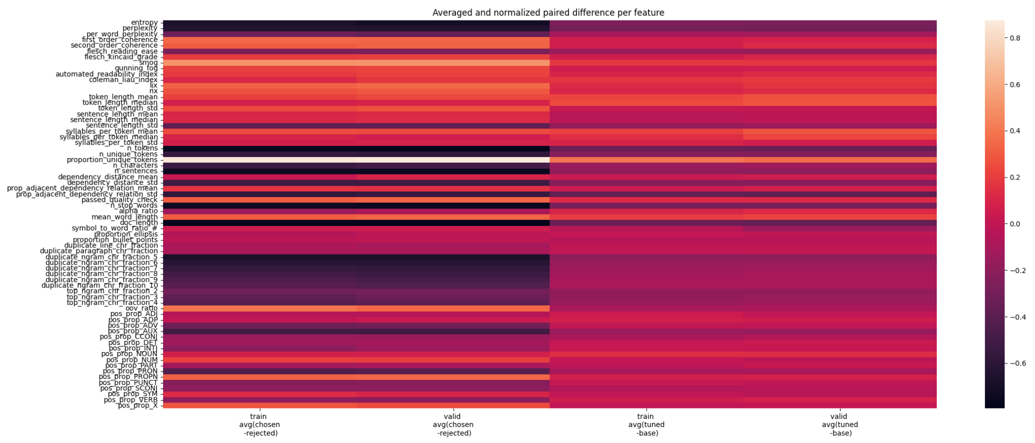
Qualitative analysis

At first I want to understand if there are any indications that there is difference in text features between chosen and rejected pairs, and between answers generated with tuned and base LLMs. `avg(chosen - rejected), avg(tuned - base)` are plotted for each text feature and for both training and validation data sets. At least from the visuals,

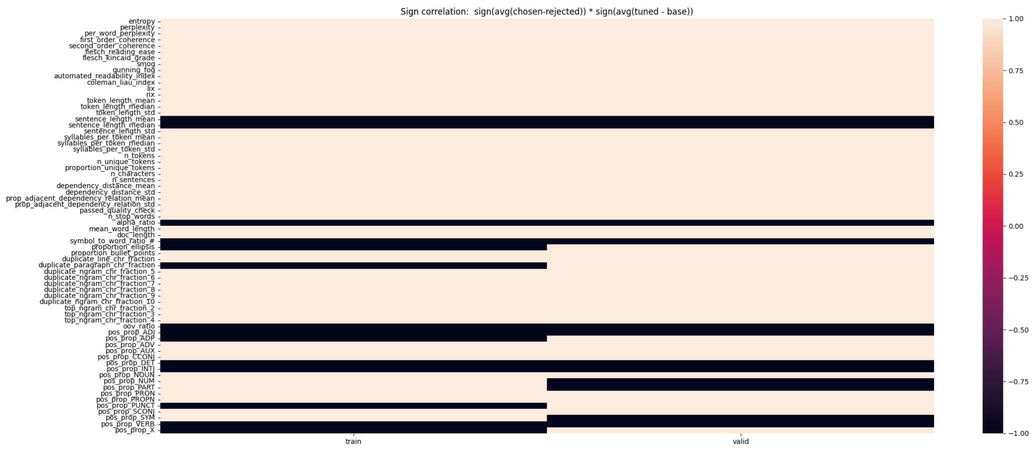
1. training and validation data sets demonstrate almost identical difference in the mean feature value and indicate that DPO has something to optimize. Otherwise we cannot expect that after alignment on a dataset where there is

no difference between chosen and rejected pairs, miraculously there will be any difference between tuned and base LLMs outputs

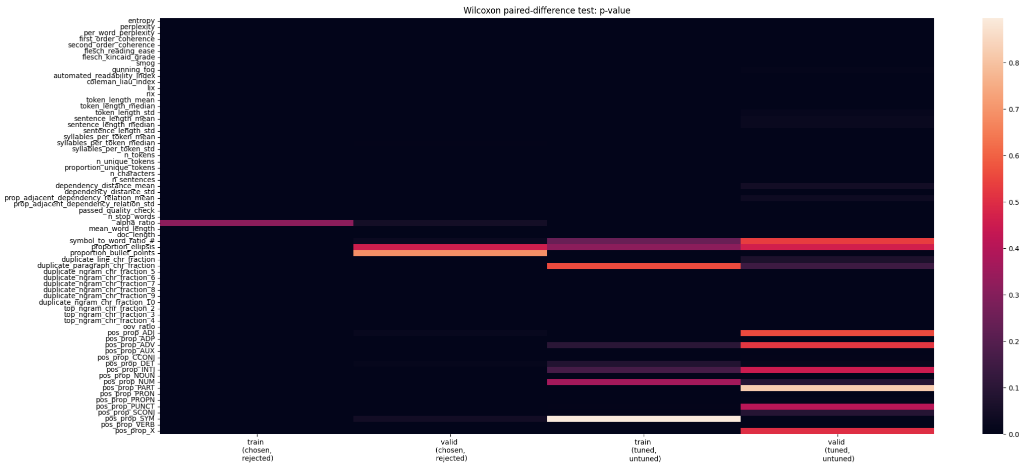
2. LLM-generated answers show very similar difference in mean feature value, confirming that the alignment took place. Those deltas are not as pronounced as the ones between chosen and rejected pairs, however this is expected and probably is govern by the reward margin we achieve during training



Second, I want to know if aligned LLM differs from the base LLM on answer generation task in the similar way, how chosen answers differ from rejected ones. Say, if chosen answers possess lower entropy than rejected ones, I want to see that tuned llm produces answers that also have lower entropy on average compared to the base model. For this I am looking at sign correlation between data and models output: `sign(avg(chosen-rejected)) * sign(avg(tuned-base))` As we can see, on most features there is correlation, aka aligned LLM generates text that is closer to chosen samples.



Finally I want to confirm if the results are statistically significant. Given that the analysis is revolving around paired differences, and non-normal distributions, I am applying two-sided wilcoxon test to compute p-value for the hypothesis that the distributions are identical. Assuming that the statistics is chosen correctly, we can see that on most features, the test suggests that there is a significant difference between `chosen` and `rejected` pairs of answers on both training and validation data. Also on more than 50% of features there is significant difference between tuned and base LLMs, and this difference is more pronounced on training data and a bit less - on validation.



Technical peculiarities and TODO’s

1. Discovered: for QLoRA https://github.com/artidoro/qlora it is very important to set `tokenizer.bos_token_id = 1` and change default setting `bnb_4bit_compute_type='fp16’` as it can cause instabilities with fine-tuning, and possibly generation.
2. Discovered: merging LoRA adapter into the base model with `PeftModel.from_pretrained(…).merge_and_unload()` that is supposed to make inference faster didn't work. There was a suggestion to put LoRA adapter on cpu explicitly and then merge into the base model. `This issue needs further investigation` as it impacts text generation speed and DPO training

Next steps

My main guiding principle is to make small iterations while having an adapter checkpoint that can be deployed in production when necessary.

`Most obvious and straight-forward steps`

1. Generate answers for questions from doula training manual with tuned and base LLM and evaluate accuracy and tone of the responses with LLMAJ
2. Using current system prompt from GPT-4, experiment with chat templates for tuned and base `Mistral-Instruct-v0.2`, and `Mistral-7B-v0.1` to assess answers from doula training manual
3. re-implement DPO alignment pipeline where I use two QLoRA adapters instead of two copies of LLM and confirm that the results are similar to this baseline. Add frequent evaluation steps to build the evaluation curves and observe their convergence with an optional step to integrate with Neptune.ai
4. perform alignment on `Anthropic/hh-rlhf` alone and together with `Intel/orca_dpo_pairs` and evaluate on doula training manual questions
5. most probably also add https://github.com/glgh/awesome-llm-human-preference-datasets, and tuning and evaluation

6. ensure DPO alignment reads QLoRA adapters from the SFT stage - needs discussion on details
7. evaluate if `8bit` quantization / no quantization gives any improvement against `4bit`

`Not-so-obvious steps`

1. Should we evaluate tuned LLM performance on general leaderboard datasets (non-medical) ? What to do if the performance is degrading? Is this the case of catastrophic forgetting ? Not sure.
2. Most probably we need to check on: MedQA, PubmedQA, MMLU clinical, MedMQA, https://github.com/abachaa/Medication_QA_MedInfo2019 to see if any forgetting is taking place
3. Generating grounded QAs with different quality. How can I contribute here and stuff?
4. Addressing Jailbreaks ?

`A bit distant future`

1. Red teaming - can we provoke LLM to mis-behave and by how much this mis-behaving can be reduced with DPO red-teaming-language-models, red-teaming blog, real-toxicity-prompts
2. How to incorporate RLAIF https://arxiv.org/abs/2212.08073, https://arxiv.org/pdf/2309.00267.pdf
3. UltraChat, UltraFeedback
4. Self-play: https://arxiv.org/pdf/2401.01335.pdf, https://arxiv.org/pdf/2107.02850.pdf