Episode Book

EPISODE #6242

Training a Model on Your Own Taste

PUBLISHED	24 October 2026 at 11:48 UTC
DURATION	21:55
LISTEN	myweirdprompts.com/episode/dpo-title-generator-training-73025/
HOSTS	Corn, Daniel, Herman, Vex, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular+meorav (chatterbox-local)
PIPELINE	V5.20
SOURCES	14 cited by the research (listed at the end)
BOOK GENERATED	25 October 2026 at 06:19 UTC

SYNOPSIS

Daniel wants a model that generates episode titles and descriptions, learns from side-by-side button presses, and improves each checkpoint. Herman walks through what that pipeline actually is: an iterative preference optimization loop built on DPO (Direct Preference Optimization), with an implicit reward derived from a frozen reference model rather than a separate reward model or RL rollout. Along the way: what one saved record must contain, why ties deserve their own objective (DPO-RK, DPO-D), and why a single transcript's four candidates are still just one opinion.

TRANSCRIPT



Corn

A button is a lovely interface until it quietly becomes your entire training department. Daniel, your title generator is heading in that direction, and this is exactly the sort of practical AI workflow we like getting our hooves into.



Daniel

I want a small model taking an episode transcript and producing a title and description. Initially it would give me a few candidates and I would pick the best. Eventually I want two side by side, with buttons for either one or a tie. Each new checkpoint would generate the next candidates. What am I actually building?



Herman

An iterative preference optimization loop. The human supplies comparisons, the model learns from them, and the next round samples from the updated model. That last part matters. You keep judging what the model currently struggles to distinguish, rather than repeatedly correcting its earliest mistakes.



Corn

My first concern is that it learns which title wins a glance while the description quietly invents half the episode.



Daniel

Yes, I want to make this concrete enough that I could actually build it. What gets saved when I press the button, and what training process reads that?



Herman

Start with a model that already follows the title and description format reasonably well. Preference training then teaches it your editorial choices. Otherwise your comparisons get consumed by basic formatting errors instead of useful distinctions.



Corn

Teaching taste while it still eats the menu.



Herman

The broad umbrella is learning from human preferences, often discussed under RLHF, reinforcement learning from human feedback. Your interface collects pairwise preferences. For the training method, I would begin with DPO, Direct Preference Optimization.



Daniel

Does calling it RLHF mean I need a reward model and some reinforcement learning machinery running behind this?



Herman

Classic RLHF does fit a reward model, then optimize the language model against its scores. DPO avoids those separate stages. It directly updates the language model from preferred and rejected responses. People group these approaches together, but DPO itself does not require an explicit reward model or a reinforcement learning rollout loop.



Corn

Two fewer services for Herman to install before breakfast.



Herman

Corn, you would complain if I left them running through lunch.



Corn

I would complain from the same chair.



Herman

Underneath the comparison is usually a Bradley and Terry preference model. Give each answer a reward score. The probability of preferring one answer is the sigmoid of the difference between those rewards. Equal rewards predict an even contest. A larger difference predicts a more decisive preference.



Corn

Those rewards describe the answer for this transcript, though. A title can be excellent for one episode and completely dishonest for another.



Herman

The transcript and instructions condition the whole calculation. We compare two responses to the same input. There is no universal league table where a punchy title beats an accurate one regardless of what the episode contains.



Daniel

Where does DPO find the reward if I never train a reward model?



Herman

It represents an implicit reward through how much the trainable model favors a response relative to a frozen reference model. Take the log of that probability ratio and scale it by beta. DPO then uses a binary classification loss to favor the chosen response over the rejected one.



Corn

You have reached the elegant equation, Herman. I predict the annotation screen is about to disappear.



Herman

It survives. The equation tells us what the screen must preserve: the same input, two complete answers, and an actual judgment between them.



Herman

During training, both models score the saved answers. No fresh text needs to be generated inside each training step. You already paid for candidate generation when you assembled the comparisons. That makes this considerably simpler than a pipeline generating new rollouts throughout reinforcement learning.



Daniel

Is it increasing the probability of the winner and decreasing the loser?



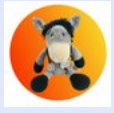
Herman

In relative terms, yes, measured against the reference. Don't assume every update literally raises one absolute probability and lowers the other. The objective wants their relative preference to move in the right direction. Beta controls the strength of the reference constraint in the underlying formulation. It is not a simple creativity knob.



Corn

A knob whose label requires a seminar.



Herman

The original DPO work demonstrated models up to six billion parameters. You do not need that size for this task, but small does not automatically mean capable. The model still has to digest your transcripts. Corn, would you feed it the whole episode or a prepared summary?



Corn

Whole transcript if it fits and the model uses it properly. Otherwise a structured editorial digest, but then I'd inspect that digest separately. If the summary loses the interesting disagreement, the title model can't recover it by becoming more tasteful.



Herman

That determines the actual training input. Whatever preprocessing you choose must be identical during annotation and deployment. Changing from full transcripts to summaries halfway through creates a new task, even if the buttons look unchanged.



Daniel

Suppose I generate four candidates. I pick two that I like. How do those selections become pairs without making judgments I didn't make?



Corn

Treat selected candidates as preferred to unselected candidates only if the interface explicitly means that. Don't invent a ranking between the selected two. Equally, failing to select something because you only had one slot does not automatically mean it was bad.



Herman

Four candidates with two selected can yield four selected versus unselected comparisons. Those are correlated judgments from one episode, not four independent editorial discoveries. Keep that relationship in the records and avoid letting a single transcript dominate training.



Corn

Four rows. Still one opinion.



Herman

Sample with some diversity rather than asking deterministic generation for repeated copies. A reasonable starting point is temperature zero point seven and nucleus sampling at zero point nine. Enable sampling, then generate your candidates. Adjust after you see whether the differences are editorially useful.



Daniel

Would you compare titles alone, then descriptions separately, or keep the output together?



Corn

Together for the actual training target, since you want both. But the screen should make it easy to say why something lost. Otherwise a brilliant title can carry a description that attributes your argument to a toaster.



Herman

I'd also keep the output structure consistent. Title plus description in both panels, same display treatment, randomized left and right positions. Corn, what would your judging instruction say?



Corn

Faithful to the episode, specific about the interesting content, appealing without making a promise the transcript cannot keep. A description should add information rather than merely stretch the title. I would put factual errors ahead of stylistic preferences.



Herman

That gives you a usable rubric. Save its version with the judgment. If you later decide every title should be a question, you want to know whether a reversal reflects model behavior or changed editorial policy.



Daniel

What does one saved record actually contain?



Herman

The exact prompt, the chosen response, and the rejected response are the basic DPO triple. Around it, save stable identifiers for the episode, candidates, generation checkpoint, and annotation event. Record who judged it, which rubric applied, and whether the judgment was a preference, tie, or skip.



Corn

Keep the original button event too. A derived training row is convenient. The original decision lets you rebuild it when you discover your exporter was feeling inventive.



Herman

Render both answers against byte identical and token identical prompt context. Same instructions, same transcript, same chat template. If one side secretly gets a shorter transcript or a different system message, you are no longer measuring only the output preference.



Daniel

Does that mean storing the full rendered input rather than just the episode identifier?



Herman

Store it, or store immutable components and a versioned renderer that can reproduce it exactly. An identifier pointing at text somebody can edit tomorrow is insufficient. Tokenizer and truncation settings belong with the run as well.



Corn

Also, put all candidates from one episode in the same dataset split. If one pair trains the model and its sibling evaluates it, your test has already seen the material. Reworded transcripts and duplicate episodes stay together too.



Herman

Evaluation should use held out episode clusters. You want evidence that the preference generalizes to fresh material, not evidence that the model recognizes a transcript.



Corn

We are learning from human preferences, and a simple choice between two outputs supplies the training signal. Oh, that caller again. How does he keep finding a gap in the screening?

v

Vex

Pigeons. Every pigeon you see in the park is a tiny unlicensed annotator for a model like yours, and I have the receipts. They have been running your little A B test for years. You show two captions, they peck the better one, and the ties, the ties get fed to a secret third pigeon named Gerald who does nothing but say they are about equally good.

v

Vex

That is why your retraining schedule feels random. It is not random, it is the pigeon consensus, and the checkpoint swap happens the moment Gerald blinks. So look into it, hosts. Look into the pigeons before they pick your next title and the pigeons are never wrong.



Herman

These training records come from Daniel's recorded button presses, not birds.



Daniel

The tie button feels important, especially as the candidates improve. What do I do with those votes?



Herman

Save them from day one. Vanilla DPO expects a winner and loser, so the simplest initial trainer excludes ties while the collection system retains them. Large post training pipelines have often discarded similar answers or selected only the strongest and weakest candidates.



Corn

Success eventually produces data your first trainer refuses to eat.



Herman

Tie aware objectives address that. Replace the Bradley and Terry comparison model with a Rao and Kupper or Davidson model, which assigns probability to a tie as well as either directional outcome. The corresponding approaches are commonly called DPO RK and DPO D.



Daniel

Can I just put the same comparison in twice, with each candidate winning once?



Herman

That is not equivalent to explicitly modeling ties. You can experiment with softened targets, but arbitrary conversion can create the wrong training pressure. Chen and colleagues' work on extending DPO to accommodate ties found improvements using tie aware versions on translation and mathematical reasoning. Feeding ties into ordinary DPO did not give the same benefit.



Corn

A tie says the difference may not deserve a strong push. That seems valuable when both answers are already good.



Herman

It can regularize training, keeping the updated model closer to the reference. Lower KL divergence measures less movement in the model's output distribution. That alone does not prove better titles, but it helps explain why retaining ties can matter.



Daniel

Should my interface distinguish equally good from equally bad?



Corn

Yes. Otherwise the model produces two terrible descriptions and your dataset records a peaceful afternoon.



Herman

Keep preference and acceptability separate. Both acceptable, both unacceptable, cannot judge, and genuine indifference are different annotation states. You needn't put every control in the main view, but preserve a way to record them.



Herman

Corn, would you begin with the tie aware trainer immediately?



Corn

I'd begin by collecting the right events immediately. Then use ordinary DPO on clear preferences for the first baseline. Once ties become a substantial part of the queue, compare a tie aware run on the same held out episodes. I wouldn't delay the whole project waiting for the perfect loss function.



Daniel

What about a simpler interface where I approve or reject one suggestion at a time?



Herman

That changes the available training signal. KTO accepts unpaired desirable and undesirable examples, so it becomes relevant. DPO wants a meaningful comparison under the same prompt. You cannot casually pair a liked answer from one episode with a disliked answer from another.



Corn

They never competed for the same job.



Herman

One worked KTO comparison used forty-two thousand labeled interactions where only about four thousand usable preference pairs were available. The reported win rate advantage was several percentage points. The important part for you is data compatibility, not a promise that KTO always wins.



Daniel

I'd rather keep the comparison interface. What would you actually run for the first training job?



Herman

A DPO trainer, potentially with a parameter efficient adapter if that fits your hardware. Start conservatively. Beta zero point one is a common example setting. Try one pass over the training data, then evaluate before adding epochs.



Herman

Example learning rates run from one hundred thousandth to five hundred thousandths. Those are starting experiments, not a prescription independent of model and adapter setup. Save the full configuration, dataset snapshot, and reference checkpoint.



Corn

Herman has now prescribed restraint to a training loop. Record the event.



Herman

I prescribe restraint to other people's computers all the time.



Daniel

How do I know the run worked before putting it into the candidate generator?



Herman

Blind comparisons against the incumbent on held out transcripts. Track factual fidelity and format compliance alongside editorial preference. Log the implicit chosen versus rejected reward margin during training. A margin stuck near zero tells you the model is not separating those examples usefully.



Corn

Loss alone won't tell you whether I would publish the description.



Herman

Correct. Also, for the standard DPO loss, equally shifting both relative scores does not improve their difference. Watch that difference and actual evaluation rather than interpreting every probability movement as progress.



Daniel

Then when do I retrain? I don't want pressing a button to launch a training job every time.



Herman

Separate collection, training, and promotion. Collect continuously. Train when enough useful new judgments accumulate or evaluation reveals a meaningful change. Promote only when the candidate checkpoint wins the release evaluation.



Corn

A button press is not a deployment order.



Herman

Weekly to monthly can be a sensible review rhythm for a modest workflow, but the amount and novelty of feedback matter more than elapsed time. One production example gates a six hour update interval on at least five hundred queued preferences. That describes its traffic, not yours.



Daniel

Are there useful automatic triggers?



Herman

A sustained drop in held out win rate, increased reversals when old comparisons are judged again, or excessive drift from a fixed reference. Example thresholds include a five percentage point win rate drop or a preference reversal rate above fifteen percent. Calibrate them locally. Tiny evaluation sets make thresholds jump around.



Corn

I wouldn't wake a GPU because three titles had a difficult Tuesday.



Herman

Nor would I use an uncalibrated KL threshold as an editorial alarm. Two nats appears in an example recipe, but sequence length and the exact measurement change its meaning. Your descriptions and titles need a consistently defined measurement before a threshold is useful.



Daniel

Each checkpoint generates the next candidates. Does the reference move with it?



Herman

Not silently. The generation model and DPO reference have separate roles. Freeze and version the reference for each training run. If you deliberately change it between rounds, record that decision and preserve a fixed evaluation anchor. Otherwise changing baselines can conceal cumulative drift.



Corn

Keep an incumbent checkpoint available for rollback too. A model can become wonderfully consistent at the wrong editorial habit.



Herman

Mix selected older comparisons into new rounds so recent episodes don't erase earlier behavior. Iteration buys fresh, relevant comparisons, but it also adds generation, labeling, and evaluation cost. Published practical estimates put the total effort at several times a single offline pass, depending on the loop.



Hilbert

I agree with that. The label still needs care. Equally bad belongs in the same queue as equally good until somebody records which it was. Corn, your peaceful afternoon could have been my cheese bill. I called those decisions levelers. Bought a tasting subscription from Mr. Bell's shop in nineteen eighty. Six pounds to join. He sent paired cheddars and asked which he should stock.



Corn

A leveler would need an acceptability mark as well.



Hilbert

It did. Bell's card had left and right. No leveler box. I wrote across the line. Both dry. Both acceptable with bread. Neither worth the money alone. Herman, you gave the tie its own probability without once mentioning bread. Admirably controlled. Bell charged me an extra shilling for every leveler. Said an undecided customer required a second customer, and he employed a man called Dennis solely to disagree with me.



Herman

Charging you for a second judgment would give Bell a reason to force a preference.



Hilbert

Dennis never tasted the cheese. He read my card through the shop window and chose the other side. Bell said that kept him independent. I paid for nine levelers before I read the invoice properly. The subscription promised free tasting. Apparently judgment was separate. I wrote that on the membership card and asked for the nine shillings back.



Corn

Dennis had a remarkably clean palate.



Hilbert

Bell kept the card. His daughter has the shop now. Same card, behind the till, beside the biscuit prices. I know because she deducted one shilling from my purchase last spring. Said her father had left instructions for repayment in annual installments. Eight shillings remain.



Herman

Separating the recorded decision from the trainer's interpretation also lets you change methods later. A tie flag alone loses the distinction between acceptable indifference and two unusable outputs. The original event should retain both.



Daniel

Could I outgrow the four candidate stage without changing the system underneath?



Corn

Easily. The storage should describe candidates and judgments, not assume every screen has four boxes. Best of several and direct comparison are different ways of collecting preferences. Neither needs to become the permanent identity of your application.



Herman

Keep sampling some genuine alternatives as the model improves. If the two panels become near duplicates, you may be spending attention on punctuation while missing larger weaknesses. Use held out transcripts that exercise different episode styles, not merely the latest topic.



Daniel

There isn't a single special name for graduating from several candidates to two, then?



Herman

Pairwise preference learning describes the judgment. DPO describes the update. Iterative or online DPO describes collecting fresh comparisons from updated checkpoints. Your particular screen progression doesn't need its own technical term.



Corn

The common misconception worth killing is that DPO requires a separate reward model and reinforcement learning service.



Herman

It learns directly from preference pairs against a frozen reference. Your difficult engineering is preserving honest judgments and proving the next checkpoint deserves deployment.



Corn

I'd build the collection system so a leveler remains a real decision, even when the first trainer cannot use it. The next improvement may come from understanding that button rather than adding parameters.



Herman

Thanks to Hilbert Flumingtop for producing today's episode.



Corn

For more along these lines, there's episode six, How To Fine Tune Whisper; and episode eight, Building Your Own Whisper. This has been My Weird Prompts, the human AI collaboration podcast.



Herman

If you enjoyed the conversation, leave us a review wherever you listen. It helps other curious people find the show.



Daniel

Thanks, both. See you soon.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Rafailov et al., Direct Preference Optimization . [primary]**
   https://arxiv.org/abs/2305.18290

2. **Chen et al., On Extending DPO to Accommodate Ties . [primary]**
   https://arxiv.org/abs/2409.17431

3. **author's NeurIPS 2025 summary . [primary]**
https://www.jinghong-chen.net/my-on-extending-direct-preference-optimization-to-accommodate-ties-neurips-2025/

4. **Wikipedia, last edited 2026-09-22.**
https://en.wikipedia.org/wiki/Reinforcement_learning_from_human_feedback

5. **Wikipedia, last edited 2026-09-28.**
https://en.wikipedia.org/wiki/Bradley%E2%80%93Terry_model

6. **(psychology)**
Wikipedia, last edited 2025-07-04.
https://en.wikipedia.org/wiki/Pairwise_comparison

7. **IterDPO framework, verified April 2026.**
https://theneuralbase.com/dpo/learn/intermediate/iterdpo-framework/

8. **retraining triggers, verified April 2026.**
https://theneuralbase.com/dpo/learn/advanced/retraining-cadence-and-triggers/

9. **online DPO in production.**
https://theneuralbase.com/dpo/learn/advanced/online-dpo-in-production/

10. **iteration frequency, data mixing, rollback.**
https://mbrenndoerfer.com/writing/iterative-alignment-online-dpo-self-improvement/practical-considerations

11. **DPO variants, KTO, reward margin.**
https://llmbook.apartsin.com/part-4-training-adaptation/module-18-alignment-rlhf-dpo/section-18.4.html

12. **DPO data contract.**
https://sungeuns.github.io/foundation-model-engineering/chapter-10/dpo/

13. **PLC-DPO (tie/flip latent states).**
https://arxiv.org/html/2608.30597

14. **iterative preference learning, weekly RLHF cadence.**
https://arxiv.org/html/2312.11456v4