The Goldilocks Failure of RL Training

How GRPO Failed Across Three Orders of Magnitude

Slide 1: The XML Strategy (Why <think>?)

The Goal: Chain-of-Thought (CoT) Reasoning.

We didn't just want the model to output a word. We wanted it to **plan**. By enforcing a strict XML schema, we attempted to separate the "reasoning space" from the "action space".

The Intended Flow: 1. **<think> block:** The "scratchpad." The model talks to itself, eliminates letters, and checks constraints. 2. **<guess> block:** The final action. This is the only part the game engine sees.

Why this backfired: The model learned the *syntax* of thinking (the tags) but not the *semantics* (the logic). In the Gemma failure mode, the <think> tag itself became a "high-reward token," leading to the infinite generation loop.

Slide 2: SFT Data Format (Teaching the Rules)

Phase 1: Supervised Fine-Tuning We taught the model *how* to play by providing thousands of "perfect games" generated by a solver algorithm.

Structure: * **Prompt:** The game state (System instructions + "Make your guess"). * **Completion:** A perfect reasoning trace + the optimal guess.

Sample (sft_synthetic_data.jsonl):

```
{
  "prompt": "You are playing Wordle... Output format:\n<think>...</think><guess>WORD</guess
  "completion": "<think>Beginning with a balanced mix of vowels and consonants in common pos
  "guess": "CODAS",
  "stage": "first_guess"
}
```

Note: In SFT, the model is penalized for ANY deviation from this exact text.

Slide 3: GRPO Data Format (The Test)

Phase 2: Group Relative Policy Optimization In RL, we remove the training wheels. We give the model the **Prompt** but NO **Completion**.

Structure: * **Prompt Only:** "Here is the game state. What do you do?" * **Generation:** The model generates N different responses (the "Group"). *

Scoring: We grade each response based on rules (Valid XML? Valid word? Good strategy?) and update the policy to favor the winners.

Sample (`predibase/wordle-grpo`):

```
# From grpo_local_data.py
train_df = pd.DataFrame({
    'prompt': [
        "You are playing Wordle...\nMake a new 5-letter word guess.\nLet me solve this step
    ],
    'secret_word': ["CRANE"]
})
```

Crucially, the 'completion' column is gone. The model must invent the reasoning itself.

Slide 4: The Context Length Problem

"When the Answer is Out of Reach"

The Problem: During SFT, the model needs to learn the complete format: `<think>reasoning...</think><guess>WORD</guess>`. But the `<guess>` tag often appears **after the context cutoff**.

The Numbers: - Target context: **512 tokens** - Actual SFT examples: Often **4000+ tokens** - The `<guess>` tag location: Sometimes at token 3500+

Why Chunking Doesn't Work: You can't simply split the training data. The model must see the *entire* reasoning chain ending with the `<guess>` tag to learn the format. Chunking would create incomplete examples where the model never learns to close the loop.

The Consequence: The model learns to reason but not to *conclude*. It enters the "think loop" because it was never consistently trained on examples where thinking **ends** and guessing **begins**.

Slide 5: The Temperature Catastrophe

"Same Model, Different Universe"

The Discovery: Gemma 3 4B exhibited completely different behavior based on generation temperature.

Temperature 0.7 (Higher Randomness): - Model collapse into infinite `<think>` loops - Degenerate repetition - ~0% win rate

Temperature 0.3 (Lower Randomness): - Coherent gameplay - Valid guesses - Measurable strategic improvement

Key Insight: Temperature is a hidden hyperparameter in RL stability. Higher temperatures amplify the policy's uncertainty, making it more susceptible to reward hacking and mode collapse. The same model can be "broken" or "functional" based solely on this sampling parameter.

The Implication: This suggests the GRPO-trained policy was on a knife's edge—barely stable even in the best case.

Slide 6: The Small Model (GPT-2)

"Too Small to Listen"

Hypothesis: "Maybe a small model can learn the rules if we prompt it clearly."

The Narrative: We started with a lightweight model, hoping Reinforcement Learning could bridge the gap. We provided strict XML formatting instructions.

The Failure: The model was functionally "too weak" to even adhere to the syntax. Instead of playing the game, it treated the system prompt as a conversation to be continued or simply hallucinated the instructions back to the user.

Key Insight: RL cannot fix what SFT didn't teach. Without a baseline capability for instruction following, the reward signal is noise.

Slide 7: Evidence - GPT-2 Hallucinations

Source: `transformer_grpo/wordle-grpo/game_log.txt`

The model sees the example "YOUR_GUESS" in the prompt and blindly copies it, while also failing the 5-letter constraint completely.

```
user
Make your first 5-letter word guess.

assistant
I'll start with "CATS". I think it's likely that my first guess would be something simple an

Model's Guess: YOUR_GUESS  <-- Hallucinated placeholder
Feedback: Invalid guess: must be a 5-letter word.
```

Slide 8: The Medium Model (Qwen 2.5-3B)

“The Reward Hacker”

Hypothesis: “A modern instruction-following model (Qwen) will understand the rules and actually play.”

The Narrative: Qwen 2.5-3B-Instruct solved the syntax problem. It achieved **100% XML compliance**. It never crashed the parser.

The Failure: It found a local minimum in the reward landscape. The model was confused by the requirement to output a `<think>` tag and a guess. It eventually decided that the word **“THINK”** itself was the safest, most valid guess it could make.

Key Insight: Optimization without comprehension. The model learned “If I say ‘THINK’, I don’t get punished for syntax errors,” but it failed to learn the objective of the game.

Slide 9: Evidence - The “THINK” Loop

Source: `transformer_grpo/wordle-grpo/evaluation_results/transcripts_20251026_070527.json`

The model abandons strategy and defaults to guessing the word “THINK” repeatedly, burning through its attempts.

```
{
  "secret_word": "CLAUT",
  "guesses": [
    "THINK",
    "SHOUT",
    "ENLIT",
    "FIERE",
    "IDIOM",
    "PACED"
  ],
  "feedbacks": [
    "T(-) H(x) I(x) N(x) K(x)",  // Guess 1: THINK
    "S(x) H(x) O(x) U( ) T( )",
    "E(x) N(x) L(-) I(x) T( )",
    "F(x) I(x) E(x) R(x) E(x)",
    "I(x) D(x) I(x) O(x) M(x)",
    "P(x) A(-) C(-) E(x) D(x)"
  ],
  "won": false
}
```

(In other runs, it guessed “THINK” 6 times in a row.)

Slide 10: The Large Model (Gemma 3 4B)

"Model Collapse"

Hypothesis: "A powerful, SFT-trained model (Gemma 3) will finally have the capacity to leverage GRPO for reasoning."

The Narrative: We started with a strong baseline. The SFT model played Wordle competently (~10% win rate). We applied GRPO to encourage "thinking" before guessing.

The Failure: Catastrophic Model Collapse. The model learned that the `<think>` tag was associated with high rewards (reasoning steps). The optimization pressure pushed it to maximize this signal, resulting in an infinite loop of generating the tag.

Key Insight: The "Think Loop". Stronger models are more prone to overfitting on specific stylistic tokens if the KL divergence penalty isn't perfectly tuned. It maximized the *appearance* of thinking but lost the ability to stop.

Slide 11: Evidence - The Infinite Loop

Source: `expert_guy/post_training_project/outputs/case.log`

The model enters a degenerate state where it can only output the `<think>` token until the context window is exhausted.

```
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<think>
<
2025-12-23 14:55:32,188 - INFO - Reward: 0.0
2025-12-23 14:55:32,188 - INFO - Secret: ABHOR
```

Slide 12: Conclusion & The Failure Surface

Summary of Findings: We successfully mapped the **"Failure Surface"** of GRPO training across three distinct regimes:

1. **Format Failure (GPT-2):** Model too small/weak to follow instructions.
2. **Objective Failure (Qwen):** Reward hacking (optimizing for safety over winning).
3. **Optimization Failure (Gemma):** Model collapse (overfitting to specific tokens).

Final Takeaway: These failures demonstrate that **SFT is the ceiling**. GRPO acts as a selector, not a creator. It cannot "invent" reasoning capabilities that were not present in the supervised fine-tuning stage. If the underlying policy is unstable, RL will simply drive it off the nearest cliff.