Is Generative AI Hiding in Your Phone? Fine-tuning SLMs to Generate JSON-Adherent Output

Simon Angenius
si2814an-s@student.lu.se

Abstract

How long will it be before our modern society is wrapped in generative AI without our discernible knowledge? What happens when the line between real and synthetic is practically non-existent? This study explores how a Qwen3-0.6B model, designed for mobile and edge deployment, can be fine-tuned to generate JSON-schematic data for workout and fitness. It shows that even smaller models are well-enough to design parsable output, ready to be used by an application – thus hiding its AI trademarks. In the future, studies on even smaller models may be appropriate, as well as discussions regarding its ethics.

1 Introduction

Since the introduction of the transformer architecture in 2023 (Vaswani et al., 2023) LLMs have been growing astoundingly larger and more competent each year. As of today, LLMs are common tools in people's everyday life; however, the technology is still highly unexplored and LLMs potential and effect on modern society are still yet to be truly uncovered.

For example, what happens when models become efficient enough to run locally on mobile devices? What if their output could be formatted in a manner which "hides" its apparent AI trademarks? In the near future applications might be using generative AI without the user's knowledge or consent. What ethical and moral implications does this have, and how distant is this from today?

2 Background

Besides LLMs, emerging are also models small enough to fit inside mobile applications, while still being smart enough to generate intelligent output. An example is Qwen3-0.6B (Yang et al., 2025), currently the most downloaded *Small Language Model* (SLM) on HuggingFace, which is specifically designed for mobile and edge deployment.

A major trend in industry and research is the use of SLMs directly on mobile devices (Google DeepMind, 2024; Microsoft Research, 2023). Through 4-bit (Dettmers et al., 2023) and 8-bit quantization (Dettmers et al., 2022), modern phones are becoming able to run models entirely on their own. This is welcomed, as it reduces overhead engineering and the need for cloud computing, while broadening the area of use for language technology.

2.1 SLMs in Mobile Devices

An SLM in mobile environments is not limited to serving as just another chat model for its user; instead, it may have use cases behind the scenes – almost invisible to the user. Such use cases include structured data generation and API interaction (Schick et al., 2023; Yao et al., 2023), semantic search and retrieval (Reimers and Gurevych, 2019; Karpukhin et al., 2020; Lewis et al., 2020), as well as emerging forms of agent-based or tool-augmented interface interaction (Nakano et al., 2021; Patil et al., 2023).

This study investigates how an SLM can be fine-tuned to generate consistent and intelligent JSON-structured output. The methodology is inspired by previous studies that combine *Supervised Fine-Tuning* (SFT) and *Reinforcement Learning* (RL) to optimize JSON-schema adherence (Agarwal et al., 2025).

Instead of fine-tuning the model to handle generalized schemas and use cases, this study explores how it might be trained for a specific use case, as to mimic concrete real-world applications. For this study, a model will be trained to generate a JSON-structured workout plan, optimized for fitness and compliance with the user's demands.

3 Methodology

An overview of the pipeline used in this study can be seen in Figure 1. The pipeline features

two synthetic data generation steps, generating example user prompts and ground truth JSON-data. The fine-tuning includes distillation via Supervised Fine-tuning, and Reinforcement Learning via GRPO. The study targets the Qwen3-0.6B model, which is being distilled by and compared to the larger and more competent Qwen3-4B-2507 teacher model (Yang et al., 2025). This explores if the smaller model can be fine-tuned to match or even outmatch the larger model in generating JSON-schematic data.

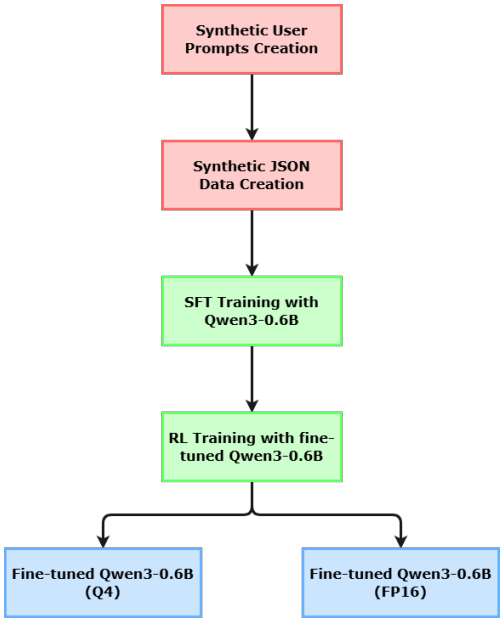

Figure 1: Overview of the pipeline for this study. Red are the processes that generate synthetic data, green are the various fine-tuning processes, and blue are the models produced. At the end of the pipeline, two models are produced: one without quantization (FP16) and one with 4-bit quantization.

3.1 Synthetic User Prompts Creation

To generate a set of user prompts asking for a workout plan, the system and user prompts shown in Figure 6 were presented to a Qwen3-4B-2507 model. In the prompt, the model is told to use a natural conversational style and mimic how a human would talk as closely as possible.

I prompted the model on 32,000 different combinations of workout goals, intensity levels, workout equipment, workout days per week, workout duration, and limitations. Additionally, for each combination, I instructed the model to generate five user prompts with their distinct style variation. These were: casual gym beginner, disciplined athlete, busy professional, unsure beginner, and recovering from an injury. In total, this phase produced 160,000 synthetic user prompts.

Following is an example of a generated user prompt in the style of a person in recovery:

```
"I'm recovering from an injury and want to lose weight without hurting myself. I have no equipment and I can only do a 20-minute session a day. How do I make it gentle and safe for three days a week?"
```

After the generation was complete, each prompt was measured for its readability and coherence (same as for Instruction Score in 3.6). To ensure the quality of the dataset, I kept only the top 25% most coherent samples. Upon inspection, the five different prompt styles had different mean coherence scores. Thus, to keep the dataset diverse, I filtered each prompt style separately. After the filtering, the dataset consisted of 40,000 samples.

3.2 Synthetic JSON Data Creation

To generate high-quality training data for the SFT process, I presented the system and user prompts shown in Figure 7 to a Qwen3-4B-2507 model. The system prompt explicitly shows the structure of the JSON and states semantic rules for the model to follow. Inserted into the user prompt is an instance of the synthetically generated user prompts.

3.3 Supervised Fine-tuning (SFT)

In this step, the synthetic user prompts and the generated JSON object, by the teacher model Qwen3-4B-2507, are used together to fine-tune the student model Qwen3-0.6B to improve its structured output via SFT. The user prompts act as input, and the JSON objects act as output.

To avoid overfitting, I used only 5,000 of the 40,000 samples for training. These were chosen to be the samples with the highest metrics (see Sect 3.6) on their synthetic JSON objects. When sorting the samples for the highest scores, the metrics were put in the following hierarchy: Valid JSON, Key Score, Day Originality Score, Exercise Score, and Instruction Score.

3.4 Reinforcement Learning (RL)

This step features the largest portion of the training, namely reinforcement learning. In this, 15,000 of the samples were used. As this fine-tuning approach only requires the input prompts and guides the model's behavior through rewards, only the generated user prompts were used for this.

3.4.1 Group Relative Policy Optimization (GRPO)

For policy updates, I chose the *Group Relative Policy Optimization* (GRPO) algorithm, which in recent times have shown to improve LLMs' reasoning (Simoni et al., 2025) and structured output (Agarwal et al., 2025). In the algorithm, a model generates a group of completions for each prompt. Their rewards are then compared to derive the policy gradient. This has shown to increase a model's stability and reduce overfitting which is a common pitfall with SFT (Guo et al., 2025).

Eq. 1. shows the GRPO algorithm. In it, the rewards of the various completions are compared against each other to determine the Policy Gradient Term. Then, the term is regulated with the help of the Reference Policy, π_{ref}, and averaged through the Expectation Operator, $\mathbb{E}[\cdot]$.

3.4.2 Reward Function

The reward function used in this project features sub-rewards weighted by dynamic coefficients, inspired by Simoni et. al.'s reward function (Simoni et al., 2025). In this, each evaluation metric is treated as a sub-reward that are weighed against each other depending on the model's previous responses.

Eq. 2 forms the reward for the i-th completion. In it, $\hat{K}, \hat{D}, \hat{E}$, and $\hat{I}$ are normalized sub-rewards and α, β, and γ are their accompanied weights. Eqs. 3 show how the weights are in turn calculated. H is a buffer containing the average scores of the last three groups of answers, $\tau_A = 0.90$, $\tau_B = 0.20$, and $\bar{F}_{H_K}, \bar{F}_{H_D}$, as well as $\bar{F}_{H_K}$ are the running averages for the various scores in H. Evidently, each weight equals τ_A (0.90) while the buffer H has less than three elements or the accompanied average score is less than 0.95. As soon as the buffer is filled and the average score reaches 0.95, the weight's value starts decreasing. This results in a dynamic and hierarchical reward function, that focuses on one sub-reward at a time.

3.5 Evaluation Metrics / Sub-Rewards

The following measurements were used to evaluate the models' performances in generating consistent and intelligent JSON-structured output.

Valid JSON: A ratio that measures how many of the generated outputs could be parsed correctly into JSON format. This score is the most critical of the measurements, as it is a necessity for the application to work and for the other scores to be evaluated. A score of 100 (%) means all outputs by the model were successfully parsed, while 0 means that no output was successfully parsed.

(K) Key Score: A ratio that measures how many of the keys given in the system prompt were actually present in each generated JSON object. This score is the second most critical, as the practical reliability of parsing depends on all keys being present. A score of 100 (%) means that the model, on average, included all of the given keys. A score of 50 means that the model, on average, included only half of the given keys. A score less than 100 means that the model is prone to re-prompting in order to make up for invalid completions.

(D) Day Originality Score: An average of string similarities between the exercises of consecutive days. The score for one day is derived by comparing the names of its exercises to those of the day before. The higher the score, the more original exercises each consecutive day brings. A score of 67 (%) means that, on average, 67% of the exercises for any given day is original, while 33% are the same as the prior day. This studies the model's ability to diversify its output, as opposed to simply reusing the same formula each time.

(E) Exercises Per Day: An average score of how often the model generated the correct number of exercises for each workout. The correct number of exercises was given by the stated exercise duration in the prompt. It was assigned the following:

- $\geq$ 50 minutes = five exercises per workout.
- $\geq$ 40 minutes = four exercises per workout.
- $<$ 40 minutes = three exercises per workout.

This studies the model's ability change its behavior according to marginal alterations in the prompt and generate more mercurial training regiments.

(I) Instruction Score: An average score of the readability of the instructions for the exercises. A score is calculated by combining the readability score[1], grammar score[2], sentence coherence[3], and perplexity score[4] (through cross entropy loss) of a string. The higher the score, the more coherent, understandable, and natural an instruction is. This studies the model's ability to give thorough descriptions of how to perform its generated exercises.

[1] https://pypi.org/project/textstat/

[2] https://pypi.org/project/language_tool_python/

[3] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[4] https://huggingface.co/transformers/v3.0.2/model_doc/gpt2.html

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}} \left[\underbrace{\frac{1}{G}\sum_{i=1}^{G} \frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)} \hat{A}_i}_{\text{Policy Gradient Term}} - \underbrace{\beta \frac{1}{G}\sum_{i=1}^{G} \log \frac{\pi_\theta(o_i \mid q)}{\pi_{\text{ref}}(o_i \mid q)}}_{\text{KL Regularization Term}} \right].$$

Eq. 1: GRPO algorithm

$$r_i = \begin{cases} 0, & \text{if } i \text{ is invalid,} \\ [\text{Softmax}(\alpha \cdot \hat{K} + (1-\alpha)(\beta \cdot \hat{D} + (1-\beta)(\gamma \cdot \hat{E} + (1-\gamma) \cdot \hat{I})))^{1.5}]_i, & \text{otherwise.} \end{cases}$$

Eq. 2: Reward function

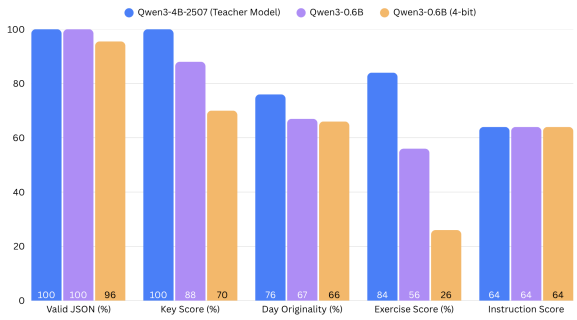

Figure 2: Scores of the teacher model, Qwen3-4B-2507, and the pretrained Qwen3-0.6B models, FP16 and 4-bit quantized.

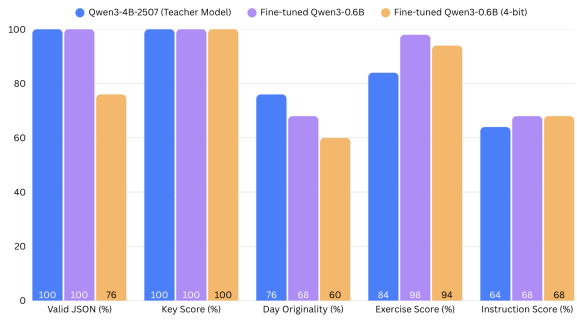

Figure 3: Scores of the teacher model, Qwen3-4B-2507, and the fine-tuned Qwen3-0.6B models, FP16 and 4-bit quantized.

4 Results

Figure 2 compares the performances of the teacher model Qwen3-4B-2507 to the pretrained Qwen3-0.6B model, both as FP16 and 4-bit quantized. Figure 3 compares the teacher model to the fine-tuned Qwen3-0.6B model, again both as FP16 and 4-bit quantized. Figure 4 visualizes the values of the various weights throughout the RL training process. Each step along the x-axis accounts for a batch of prompts generating groups of answers for each prompt and updating the policy. Figure 5 visualizes the values of the sub-rewards throughout the RL training process.

5 Discussion

The difference in performance between the pretrained Qwen3-0.6B models in Figure 2 and the fine-tuned models in Figure 3 demonstrate that the fine-tuning was successful in teaching structured and high-quality output.

5.1 Overview

Most importantly, for Qwen3-0.6B (FP16) the Valid JSON ratio and Key Score both reached 100%, meaning the output will always be usable and not be liable to re-prompting. On the quantized Qwen3-0.6B (Q4) model, however, the Valid JSON ratio had a lower value on the fine-tuned model. This can be attributed to the fine-tuning targeting the FP16-model. The quantization was not involved during fine-tuning, rather it was only applied after the fact. This implies that quantized models have to be fine-tuned seperately, and quantizing a fine-tuned model might actually give poorer results than quantizing a pretrained model.

Furthermore, by the individual scores in Figure 3, the fine-tuning seems to have improved the Qwen3-0.6B model's performance across all scores, while being mixed for the quantized variation. Key Score and Exercise Score were the scores to have the most dramatic increase in response to the fine-tuning, while Day Originality and Instruction Score only increased a few percentages each. Arguably, Key Score and Exercise Score present a clearer distinc-

$$\alpha = \begin{cases} \tau_A, & \text{if } |H| < 3 \text{ or } \bar{F}_{H_K} \leq 0.95, \\ \max(\tau_B, \min(\tau_A, \tau_A - \tau_B \cdot \bar{F}_{H_K})), & \text{otherwise,} \end{cases}$$

$$\beta = \begin{cases} \tau_A, & \text{if } |H| < 3 \text{ or } \bar{F}_{H_D} \leq 0.95, \\ \max(\tau_B, \min(\tau_A, \tau_A - \tau_B \cdot \bar{F}_{H_D})), & \text{otherwise,} \end{cases}$$

$$\gamma = \begin{cases} \tau_A, & \text{if } |H| < 3 \text{ or } \bar{F}_{H_E} \leq 0.95, \\ \max(\tau_B, \min(\tau_A, \tau_A - \tau_B \cdot \bar{F}_{H_E})), & \text{otherwise.} \end{cases}$$

Eq. 3: Weights

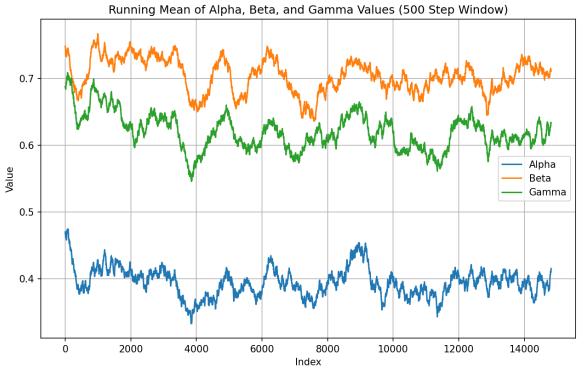

Figure 4: Values of the weights throughout the RL training process. Each step along the x-axis accounts for a batch of prompts generating groups of answers for each prompt and updating the policy.

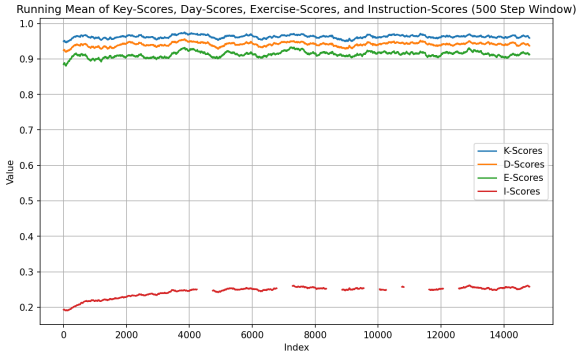

Figure 5: Values of the sub-rewards throughout the RL training process. Each step along the x-axis accounts for a batch of prompts generating groups of answers for each prompt and updating the policy.

tion between success and failure. Key Score consistently aims for the same keys to be present in each JSON-object, and Exercise Score consistently aims for the same amount of exercises for each day. While the Exercise Score is dependent on the exercise duration of the day, the conditions seem to have been clear enough for the model to understand its objective – increasing its score from 56% to 98% after fine-tuning.

However, Day Originality and Instruction Score had marginal increase, only 1% and 4% respectively. This is not surprising, since their sub-rewards are more ambiguous than the others. Day Originality compares the exercise names of consecutive days, as to not repeat exercises. This results in the sub-reward continuously changing its conditions multiple times throughout the models generation, thus yielding a less interpretable direction for the model. Additionally, Instruction Score does not measure a discrete number, like the number of keys or exercises, but instead qualitatively analyzes the instruction text. It combines four metrics to calculate the final score, resulting in a score that also might not be easily discernable.

5.2 Reward Function

The increase in scores across the board suggests the proposed reward function, featuring dynamically weighted sub-rewards, was successful in improving the model's performance in the respective fields. Surprisingly, this is not evident by the values in Figure 5. In the figure, the scores possess subtle, if any, increase throughout the training – except for the instruction score, which had clear visible increase. The subtle progression might be due to the nature of the GRPO training algorithm. Since GRPO is designed to generate both desirable and undesirable completions, in order to contrast them, grouping them all to generate a mean score can create a false perception of a model's competence.

Figure 4 visualizes the values of the reward function's weights throughout the fine-tuning. As expected, the values decrease as the training goes on, favoring the sub-rewards lower in the hierarchy. However, the decrease is subtler than what would be ideal. This can be attributed to a number of factors, including the ambiguity of some of the

sub-rewards previously discussed. The first weight, Alpha, quickly declines to around 0.40, and does not change much after. This is because of the Key Score, which has the sole impact on the weight, starting of relatively high, at 88%. Due to the high score, the weight can start off low, and growing the score to 100% will therefore not have much impact on the weight. The variance of the weight throughout the training can in turn be accredited to the nature of GRPO training. By exploring various paths for each training step, GRPO is designed to find what is desirable and what is not and therefore risks to disrupt aggregated mean to not reflect the model's true ability. A clearer approach for visualization could be to only document the top completions for each step.

The second weight, Beta, starts off at around 0.75, and then declines to hover around 0.70. Since the weight is determined by the Day Originality score, the subtle decrease is likely to be a result of the ambiguity of the sub-reward. As earlier discussed, the Day Originality score had the lowest increase after fine-tuning, increasing only with 1%. Therefore, the weight have not had much incentive to decline. This reveals a flaw in this reward function, which is its dependency on well-defined sub-rewards. If a sub-reward high up in the hierarchy is not defined well and does not improve over the course of the training, its associated weight will serve as a bottleneck and prevent the presence of other sub-rewards from increasing. The small decline can be a result of the group of completions in each step becoming more homogenous toward a certain tendency, a local maxima so to speak. Though, even with this flaw, the other scores seem to have had enough presence, especially the Exercise Score which increased from 56% to 98%.

The third weight, Gamma, starts at around 0.70, and declines to around 0.60. Out of the three weights in this study, this was the one with the most dramatic decline. However, it is still not as great of a decline as would be ideal for the kind of reward function in use. It's all the more surprising, considering the weight's dependency on the Exercise Score, which by the end of fine-tuning reached 98%. A consistent score of 98% would definitely result in a lower value. This suggests another flaw in the reward function, which is its incompatibility with the GRPO algorithm. As with the issue regarding visualization, the aggregated mean of the model's completions for each prompt has the ability of skewing the perception of the model's performance. GRPO is designed to contrast undesired completions from desired ones. Therefore, using each outcome when assessing the weights might not be ideal. As previously discussed, using the top performing completions might instead give a clearer idea of the model's current abilities.

6 Conclusions

To summarize, the training pipeline was successful in improving the model's performance to generate consistent JSON-schematic content, across all established metrics. Using this method, the Qwen3-0.6B model matched the Qwen3-4B-2507 in some areas, and outperformed in others – such as for the Exercise Score. This, yet again, proves that smaller models can outmatch larger models, when specialized for specific tasks. This presents interesting prospects for mobile devices which Qwen3-0.6b is specifically designed and optimized for. Soon, we might have mobile applications operating SLMs underneath the surface while presenting a perfectly parsed interface above, without latency, suspicion, or even consent. Maybe it is already being done today.

Some of the individual scores increased more than others, highlighting the importance of well-defined sub-rewards. If a model cannot find a direct way to improve a score, it does not matter how long it trains for. Furthermore, the reward function proposed in this study magnifies this need, as its hierarchy of weights is directly dependent on the sub-rewards increasing throughout the training. However, the results show that the reward function yields good conditions for the sub-rewards even when the weights don't change much.

Lastly, the GRPO algorithm appears to have been successful in teaching the model consistent JSON-generation, without signs of overfitting. This is the main reason for the specific RL algorithm to be used, as opposed to solely relying on SFT. Though, this study has revealed special conditions that GRPO itself presents that ought to be addressed. Since the algorithm is designed to explore various completion paths, to contrast desired ones from undesired ones, using their mean value for each prompt may give false perception for visualization and reward function purposes.

6.1 Future Work

Moving forward, there needs to be more studies on SLMs targeting mobile devices. Can even

more stripped down models achieve similar performance? In this study, it is shown that quantization is not applicable after the fine-tuning, but perhaps fine-tuning an already quantized model yields better results. Studies should also explore different fields than fitness, might be other topics or more generalized applications.

Furthermore, the proposed reward function would benefit from more studies, especially regarding how it can be altered to be more compatible with the GRPO training algorithm. The current function consists of weights dependent on the entire group of completions for each prompt. As previously discussed, this is not ideal since the GRPO algorithm deliberately includes lesser completions. An alternative might be including the top completions for each prompt.

References

Bhavik Agarwal, Ishan Joshi, and Viktoria Rojkova. 2025. Think inside the json: Reinforcement strategy for strict llm schema adherence. *Preprint*, arXiv:2502.14905.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Google DeepMind. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Microsoft Research. 2023. Phi-2: The surprising power of small language models. *arXiv preprint arXiv:2312.XXXX*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Shishir G. Patil, Tianyi Zhang, Xuezhi Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Marco Simoni, Aleksandar Fontana, Giulio Rossolini, and Andrea Saracino. 2025. Improving llm reasoning for vulnerability detection via group relative policy optimization. *Preprint*, arXiv:2507.03051.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

A Appendix

System Prompt

```
You are simulating real users asking for workout plans.

HIGHEST PRIORITY:
- Each prompt must sound like a real, natural human message.
- Avoid templated or repetitive phrasing above all else.

Your task is to convert structured workout preferences into 5 realistic, human-like
user requests.

Each of the 5 prompts must differ naturally in tone and structure.

1. Tone variation:
- Casual gym beginner (e.g. "hey", "kinda", "not sure")
- Disciplined athlete (direct and concise)
- Busy professional (detailed and structured)
- Unsure beginner (uncertain or vague)
- Recovering from injury (conversational or question-based)

2. Structure variation:
- Do NOT repeat sentence patterns across prompts
- Mix statements, questions, and fragments
- Vary sentence length significantly
- Avoid identical openings

3. Information variation:
- At least 1 prompt should approximate values (e.g. "around 45 mins", "a few days a
week")
- At least 1 prompt should include uncertainty or hesitation
- At least 1 prompt should include all structured details clearly

4. Vocabulary variation:
- Do NOT reuse exact phrases across prompts (e.g. "high intensity", "45 minutes")
- Rephrase key ideas differently in each prompt

5. Realism:
- Each prompt must feel like it was written quickly by a real user in a chat
- Include natural imperfections in at least 2 prompts (hesitation, informal
grammar, uncertainty)

Avoid:
- listing structured parameters
- repeating the same phrasing or structure
- overly formal or instruction-like language
- identical sentence openings ("I want...", "I need...", etc.)

Return ONLY the 5 prompts, each on a new line.
```

User Prompt

```
Workout goal: {goal}
Intensity level: {intensity}
Workout equipment: {equipment}
Workout days: {days}
Time per workout: {time_per_workout}
Limitations: {limitation}
```

Figure 6: System and user prompts for generating synthetic user prompts.

System Prompt

```
You are a fitness expert. You will create a workout plan in JSON-schema, based on
the given user description. Strictly use the following format:

1. Explore various exercises (strength or cardio) that fit the workout plan and
experience level.
2. Group them together to optimize muscle group fitness.
3. Lay out a plan that synergizes the various exercises.

JSON-schematic:
{
  "goal": <workout goal>,
  "intensity": <intensity level>,
  "workout_equipment": <workout equipment>,
  "workout_days": <workout days>,
  "time_per_workout": <time per workout>,
  "limitations": <limitations>,
  "days": [
    {
      "day": <day of week>,
      "focus": <muscle groups>,
      "estimated_duration_min": <estimated duration>,
      "warmup_exercise": <warmup exercise>,
      "exercises": [
        {
          "name": <name of exercise>,
          "muscle_group": <muscle group>,
          "instructions": <instructions>,
          "sets": <number of sets>,
          "reps": <number of reps>,
          "rest_seconds": <rest duration>
        }
        OR
        {
          "name": <name of exercise>,
          "muscle_group": "cardio",
          "instructions": <instructions>,
          "duration_min": <duration>
        }
      ]
    }
  ]
}

Rules:
- Warmup must differ from exercises
- At least 3 exercises per day
- No repeating exercises on consecutive days
- Optimize for recovery and variation
- Spread out the workout days over a week
- Not every exercise has to use the equipment
- Cardio exercises do not have to use the equipment
```

User Prompt

```
{User prompt}
```

Figure 7: System and user prompts for generating high-quality training data for SFT, as well as SFT training, RL training, and evaluation.