vLLM

Luka Meladze
National University of Singapore
Project: **LLM Post-training with Reinforcement Learning**

Abstract

This report discusses the findings from an experiment conducted to understand the behavior of small, open-source Large Language Models (LLMs) using the vLLM inference engine. The experiment focused on evaluating two primary aspects: the effect of the temperature sampling parameter on model output and the difference in performance between pre-trained base models and their post-trained instruction-tuned counterparts. The following four models from the Qwen series were tested: Qwen/Qwen2.5-0.5B (Base), Qwen/Qwen2.5-0.5B-Instruct (Instruct), Qwen/Qwen3-0.6B-Base (Base), Qwen/Qwen3-0.6B (Instruct). A simple set of four prompts was used to test each model's capabilities in math, knowledge, factual mistakes, and hallucinations. Each prompt was run with four different temperature settings: 0, 0.6, 1.0, and 1.5. The prompts were: 1. "How many positive whole-number divisors does 196 have?"; 2."The capital of Singapore is"; 3."Who are you?"; 4. "What is the range of the output of tanh?"

1 Observations and Comparative Analysis of models

1.1 Temperature Parameter

The temperature parameter controls the randomness of the model's output. A lower temperature (e.g., 0) makes the model deterministic, causing it to pick the most likely next token. A higher temperature (e.g., 1.0 or 1.5) increases randomness, allowing the model to choose less likely tokens, which can lead to more creative, diverse, but also less coherent and potentially incorrect responses. The temperature setting is a critical dial for controlling the trade-off between factual accuracy and creativity. For tasks requiring precision and correctness (like math or code generation), a low temperature (0 to 0.4) is ideal. For creative writing or brainstorming, a moderate temperature (0.6 to 0.8) can be beneficial. Temperatures of 1.0 and above significantly increase the risk of hallucination and incoherence, especially in smaller base models.

- Temperature 0 (Deterministic): At this setting, all models produced their most factual and stable outputs.
  - **Qwen/Qwen2.5-0.5B & Qwen/Qwen3-0.6B-Base Models**
    At temperature=0, since the models are deterministic the base models produced highly repetitive output, getting stuck in a loop. For instance, with 0.5B parameters it get stuck into the loop: "What is your purpose? What is your purpose in life? What is your purpose in life? What is your purpose in life?..." While for the Qwen3-0.6B after a few tokens, it gets stuck into the loop of "What is your favorite type of music?". This occurs because the model repeatedly selects the most probable next token, which in this context, creates a recursive questioning pattern it cannot break out of.
- Temperature 0.6 (Slightly Creative): At this setting there are minor variations in wording and structure without significantly compromising the factual content. The Qwen/Qwen2.5-0.5B-Instruct model, for example, gave slightly different but still coherent introductions to its "Who are you?" response that changes every time the model is executed since the output is not deterministic anymore. However, for the base model (Qwen/Qwen2.5-0.5B), this temperature was enough to degrade its

mathematical accuracy, as it incorrectly factored 196. Base models for questions like "Who are you?" are trapped, but the slightly increased randomness allows it to jump to a different, yet still repetitive, high-probability phrases: *"How do you feel about yourself? Do you like yourself? ... You know yourself."*

- Temperature 1.0 (Creative/Unstable): At this level, responses became much more varied. At temperature=1.0, the model generates a creative but fabricated answers: "What matters most to you? ... Im a clown. Its actually a pretty interesting job when I interact with a bunch of kids. ... I have the dreaded cotton balls to soak wet wiping sports balls in." Here, the higher randomness allows the model to create a narrative. It's no longer stuck in a loop but has sacrificed factuality for creativity, inventing a personality and bizarre details, especially for base models that are **drawing from its vast but unstructured training data.**
- At temperature=1.5 (Chaotic), the output degrades into complete incoherence, mixing languages and random characters. This is a classic example of model collapse at very high temperatures where the token selection becomes too random to form meaningful text. *"Contoperfectoid-statement word guessing- rand -profindr .slf thread..."* This demonstrates that while high temperature can foster creativity, excessive levels destroy the model's structural and semantic integrity.

1.2 Base vs Instruct Models

Base models are trained to predict the next token in a sequence, so they are good at at text completion but do not behave like assistants. Instruct models are fine-tuned on instruction-response pairs, teaching them to follow commands and act as helpful, conversational agents.

Qwen2.5-0.5B & Qwen3-0.6B-Base vs. Qwen2.5-0.5B-Instruct & Qwen3-0.6B (temp < 1)

- Prompt: "The capital of Singapore is":
  - Base Model (Qwen2.5-0.5B & Qwen3-0.6B-Base) continues the prompt as if it were a multiple-choice question, due to its text-completion function. Response: "A. Singapore B. Kuala Lumpur C. Singapore D. Singapore Answer: A"
  - Instruct Model (Qwen2.5-0.5B-Instruct & Qwen3-0.6B) correctly interprets the prompt as a question and provides a direct answer: the capital of Singapore is Singapore City Hall."

  The fundamental difference in behavior is clear: one completes a pattern, the other follows an instruction.
- **Prompt: "Who are you?**:
  - Base Models fail to answer and enter a repetitive loop because they have no post-training and are simply next token predictors.
  - Instruct Model correctly provides answers like "I am Qwen, a large language model created by Alibaba Cloud."

1.3 LLM output correctness

LLM outputs are not guaranteed to be correct. The outputs received show a mix of accurate, subtly incorrect, and completely fabricated answers. Increasing the temperature generally decreases factual reliability. At temperature=0, instruct models correctly answer the math problems for both the divisors of 196 and the range of tanh. However, at temperature=0.6, the base model Qwen2.5-0.5B makes an error in factorization and says: ""We can write 196 as $196 = 2^2 * 19$, the number of divisors is $(2 + 1) * (1 + 1) = 6$ which is obviously incorrect.

The Qwen models in this test (0.5B and 0.6B parameters) are significantly smaller than leading proprietary models like ChatGPT GPT-4 model. GPT-4 model actually has trillions of parameters. This size difference is the primary reason for performance disparities. The higher number of parametrs (even comparing models with 0.5B and 0.6B parameters) show that larger models have slightly lower chance at the same temperature of making calculation

errors like the $196 = 2^2 * 19$ mistake. Much large models like ChatGPT GPT-4 provides much more nuanced and factually accurate answers.

```python
import torch
import gc
from vllm import LLM, SamplingParams
from transformers import AutoTokenizer

models = [
    "Qwen/Qwen2.5-0.5B",
    "Qwen/Qwen2.5-0.5B-Instruct",
    "Qwen/Qwen3-0.6B-Base",
    "Qwen/Qwen3-0.6B",
]
prompts = [
    "How many positive whole-number divisors does 196 have?",
    "The capital of Singapore is",
    "Who are you?",
    "What is the range of the output of tanh?",
]
temperatures = [0, 0.6, 1.0, 1.5]

model_selected = models[2] # The index for independent runs: 0, 1, 2, 3

llm = LLM(
    model=model_selected,
    trust_remote_code=True,
    dtype=torch.float16,
    gpu_memory_utilization=0.9
)

tokenizer = AutoTokenizer.from_pretrained(model_selected,
                                          trust_remote_code=True)

is_instruct_model = "instruct" in model_selected.lower() or "Qwen3-0.6B"
                                          == model_selected.split('/')[-1]

if is_instruct_model:
    formatted_prompts = []
    for prompt in prompts:
        messages = [{"role": "user", "content": prompt}]
        formatted_prompt = tokenizer.apply_chat_template(
            messages, tokenize=False, add_generation_prompt=True)
        formatted_prompts.append(formatted_prompt)
else:
    formatted_prompts = prompts

for temp in temperatures:
    print(f"\n>>> TEMPERATURE: {temp} <<<")
    sampling_params = SamplingParams(temperature=temp, max_tokens=4096)
    outputs = llm.generate(formatted_prompts, sampling_params)

    for i, output in enumerate(outputs):
        original_prompt = output.prompt
        generated_text = output.outputs[0].text
        print(f"\nPrompt: {prompts[i]}")
        print(f"Generated Text: {generated_text.strip()}")
del llm
del tokenizer
gc.collect()
torch.cuda.empty_cache()
```