Week 7 Report

Mu Junrong

Abstract

This report focuses on the introduction to the inference of Large Language Models (LLMs) from the Qwen series using the vLLM inference engine (Zheng et al. (2023)). Furthermore, four Qwen models are used on a set of factual and open-ended prompts, two base and two instruction-tuned. The output by each model using difference sampling temperatures are evaluated, based on their correctness and helpfulness.

1 Introduction to Large Language Models (LLMs) inference

Large Language Models (LLMs) inference refers to the process of using a trained large language model (LLM) (e.g., GPT, LLaMA, or PaLM) to generate outputs (i.e., text or answers) based on input prompts. The training process of an LLM involves adjusting its weights using large datasets, while inference runs the model on new data without changing its weights.

Large language models (LLMs) inference is usually used for real-world applications, including Chatbots and auto-complete in code editors. During inference, the prompt is tokenised, and tokens are passed through the transformer layers of the model. The transformer model then produces logits (i.e., probabilities for next token), and a token is selected (e.g. via greedy, sampling, beam search). The steps are repeated until the output is complete.

To experiment on LLMs inference, vLLM is used to run Qwen Large Language Models.

1.1 vLLM

vLLM (Zheng et al. (2023)) is an open-source high-performance inference server and runtime optimised specifically for large language models (LLMs). It is designed to serve LLMs efficiently, providing faster and more scalable text generation compared to standard implementations (i.e., it is the engine to run models on the laptop) (Dao et al. (2022)). It is built on top of PyTorch and NVIDIA CUDA for GPU acceleration.

vLLM introduces a new attention mechanism called PagedAttention (Zheng et al. (2023)), which significantly improves GPU memory efficiency. In normal transformer inference, every new token must attend to all past tokens, creating Key-Value cache. Each input also needs a separate chunk of memory for the Key-Value cache, causing fragmentation and wasted memory. For PagedAttention, prompt is received when the user sends a prompt, and PagedAttention assigns pages instead of allocating a long contiguous buffer for KV cache. It allocates small fixed-size pages and adds them to a page table per request. Attention uses the page table and during attention computation, query tokens look up past key/value tokens across the pages, and thus there is no need for continuous memory. When a request ends, its pages are returned to the pool and reused. Thus, it reduces memory needed and uses all GPU resources efficiently.

1.2 Qwen

Qwen (Tongyi Qianwen in Chinese, meaning "Thousand Questions of Tongyi") is a family of Large Language Models (LLMs) developed by Alibaba DAMO Academy. It shows strong performance on multilingual tasks (especially Chinese), and performs well on commonsense, math, and reading comprehension.

1.3 Experiment Setup

To examine different models' performance and the effect of different sampling temperatures, the experiment is set up as follows. NVIDIA GPU and Python 3.9 are used. For each model, four sampling temperatures (i.e., 0, 0.6, 1, 1.5) are used, and each model is trailed for three times. Four prompts are used to observe the LLMs' performance:

- "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?"

2 Analysis of different sampling temperatures

Sampling temperature (Holtzman et al. (2020)) is a hyperparameter used during inference, which determines the randomness and creativity of the Large Language Model (LLM) when generating text. It is part of the sampling strategy for selecting the next token during generation.

During inference, the model computes a logit (unnormalized score) for each possible next token. These logits are then transformed into probabilities using the softmax function

$$P_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Where z_i is the logit for token i and P_i is the probability of selecting token i.

Temperature modifies the logits before softmax application by

$$P_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

Where T is the temperature parameter

For example, the prompt is given by "The cat sat on the". The model predicts the next word by producing logits (unnormalized scores), where token "mat" has logit 3, token "couch" has logit 2.5 and token elephant has logit 1. When the sampling temperature is 0, the model always picks the highest logit which is "mat". When the sampling temperature is 1, after applying softmax, $P(couch) = 0.2$, thus "couch" may be chosen. The higher the sampling temperature, the more random the output.

2.1 Meaning of temperatures

As explained, the higher the T, the more uniform probabilities (more random outputs), which may be better for storytelling. On the other hand, lower T sharpen the distribution, which produces more deterministic outputs.

Temperature Value	Description
0	No randomness. The model always picks the most probable next token, which ensures the output to be deterministic and factual.
0.6	Low randomness. There is slight diversity but the output is still relatively reliable.
1.0	Balanced randomness. Outputs are diverse and more natural.
1.5+	High randomness. More surprising or creative results, but higher risk of being nonsensical or wrong.

Table 1: Effects of Different Temperature Settings

2.2 Analysis of different sampling temperatures on the output

For example, for the model Qwen 2.5, when the sampling temperature is set to 0, the output to the prompt "How many positive whole-number divisors does 196 have?" is always the same.

3 Analysis of different models

In the experiment, four models are used: Qwen2.5-0.5B (Qwen Team (2024a)), Qwen2.5-0.5B-Instruct (Qwen Team (2024b)), Qwen3-0.6B-Base (Qwen Team (2025b)) and Qwen3-0.6B (Qwen Team (2025a)). They are from Qwen 2.5 and Qwen 3, and two of them are base models while another two are instruct models. Overall, Qwen3-0.6B shows better performance since the base is newer, and it is instructed.

3.1 Qwen 2.5 v.s. Qwen 3

The model Qwen 2.5 has more parameters For example, when the prompt is a creative (instead of factual) question such as "Who are you?", Qwen 2.5 produces the output "What is your purpose in life?" for many times repetitively, which is wrong and not meaningful. On the other hand, Qwen 3 produces more relevant result.

	Qwen2.5-0.5B	Qwen3-0.6B
Training data and model size	The model is trained on web corpus, books and code (English, Chinese). It is trained on 0.5 billion parameters	The model is trained on Larger and more diverse corpus with cleaner filtering. It is trained on 0.6 billion parameters
Model type	Decoder-only transformer (GPT-style)	Decoder-only transformer (GPT-style)
Performance	Performance on basic math and reasoning is acceptable, with high correctness on factual outputs. The JSON formatting is poor, and it shows poor performance on creative questions.	Performance on factual questions are more robust and detailed. The JSON formatting is good with structured output generated.

Table 2: Qwen 2.5 model vs Qwen 3 model

3.2 Base model v.s. Instruct model

In the experiment, Qwen2.5-0.5B and Qwen3-0.6B-Base are base models, and Qwen2.5-0.5B-Instruct and Qwen3-0.6B are instruct models. Base models are pretrained LLMs trained only on next-token prediction, while instruct models are finetuned using Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al. (2022)).

	Base model	Instruct model
Training process	The model is trained on massive datasets via self-supervised learning (e.g., web text, books and code) to predict the next token.	The model is a finetuned version (using supervised instruction-following data and RLHF) of the base model, optimized to follow instructions in a human-aligned way.
Performance	It predicts the next token in a sequence, but it is not optimised for conversational tasks. It may produce unsafe and offensive outputs.	It responds to prompts such as "Write a summary" or "Give a step-by-step answer" with aligned, structured output.

Table 3: Base models vs Instruct models

References

Tri Dao, Daniel Y. Fu, Stefano Ermon, et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022. URL https://arxiv.org/abs/2205.14135.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1904.09751.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL https://arxiv.org/abs/2203.02155.

Alibaba DAMO NLP Qwen Team. Qwen2.5-0.5b. https://huggingface.co/Qwen/Qwen2.5-0.5B, 2024a. Accessed: 2025-06-27.

Alibaba DAMO NLP Qwen Team. Qwen2.5-0.5b-instruct. https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct, 2024b. Accessed: 2025-06-27.

Alibaba DAMO NLP Qwen Team. Qwen3-0.6b (instruct). https://huggingface.co/Qwen/Qwen3-0.6B, 2025a. Accessed: 2025-06-27.

Alibaba DAMO NLP Qwen Team. Qwen3-0.6b-base. https://huggingface.co/Qwen/Qwen3-0.6B-Base, 2025b. Accessed: 2025-06-27.

Zihao Zheng, Wei-Lin Chiang, Shouqian Li, Kevin Lin, et al. vllm: Easy, fast, and cheap llm inference with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.