(faq)=

# Frequently Asked Questions

> Q: How can I serve multiple models on a single port using the OpenAI API?

A: Assuming that you're referring to using OpenAI compatible server to serve multiple models at once, that is not currently supported, you can run multiple instances of the server (each serving a different model) at the same time, and have another layer to route the incoming request to the correct server accordingly.

---

> Q: Which model to use for offline inference embedding?

A: You can try [e5-mistral-7b-instruct](https://huggingface.co/intfloat/e5-mistral-7b-instruct) and [BAAI/bge-base-en-v1.5](https://huggingface.co/BAAI/bge-base-en-v1.5);
more are listed [here](#supported-models).

By extracting hidden states, vLLM can automatically convert text generation models like [Llama-3-8B](https://huggingface.co/meta-llama/Meta-Llama-3-8B),
[Mistral-7B-Instruct-v0.3](https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3) into embedding models,
but they are expected be inferior to models that are specifically trained on embedding tasks.

---

> Q: Can the output of a prompt vary across runs in vLLM?

A: Yes, it can. vLLM does not guarantee stable log probabilities (logprobs) for the output tokens. Variations in logprobs may occur due to numerical instability in Torch operations or non-deterministic behavior in batched Torch operations when batching changes. For more details, see the [Numerical Accuracy section](https://pytorch.org/docs/stable/notes/numerical_accuracy.html#batched-computations-or-slice-computations).

In vLLM, the same requests might be batched differently due to factors such as other concurrent requests, changes in batch size, or batch expansion in speculative decoding. These batching variations, combined with numerical instability of Torch operations, can lead to slightly different logit/logprob values at each step. Such differences can accumulate, potentially resulting in different tokens being sampled. Once a different token is sampled, further divergence is likely.

## Mitigation Strategies

- For improved stability and reduced variance, use `float32`. Note that this will require more memory.
- If using `bfloat16`, switching to `float16` can also help.
- Using request seeds can aid in achieving more stable generation for temperature > 0, but discrepancies due to precision differences may still occur.