(generative-models)=

Generative Models

vLLM provides first-class support for generative models, which covers most of LLMs.

In vLLM, generative models implement the {class}`~vllm.model_executor.models.VllmModelForTextGeneration` interface.
Based on the final hidden states of the input, these models output log probabilities of the tokens to generate,
which are then passed through {class}`~vllm.model_executor.layers.Sampler` to obtain the final text.

For generative models, the only supported `--task` option is `"generate"`.
Usually, this is automatically inferred so you don't have to specify it.

Offline Inference

The {class}`~vllm.LLM` class provides various methods for offline inference.
See [Engine Arguments](#engine-args) for a list of options when initializing the model.

`LLM.generate`

The {class}`~vllm.LLM.generate` method is available to all generative models in vLLM.
It is similar to [its counterpart in HF Transformers](https://huggingface.co/docs/transformers/main/en/main_classes/text_generation#transformers.GenerationMixin.generate),
except that tokenization and detokenization are also performed automatically.

```python
llm = LLM(model="facebook/opt-125m")
outputs = llm.generate("Hello, my name is")

for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

You can optionally control the language generation by passing {class}`~vllm.SamplingParams`.
For example, you can use greedy sampling by setting `temperature=0`:

```python
llm = LLM(model="facebook/opt-125m")
params = SamplingParams(temperature=0)
outputs = llm.generate("Hello, my name is", params)

for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

A code example can be found here: <gh-file:examples/offline_inference/basic.py>

`LLM.beam_search`

The {class}`~vllm.LLM.beam_search` method implements [beam search](https://huggingface.co/docs/transformers/en/generation_strategies#beam-search-decoding) on top of {class}`~vllm.LLM.generate`.
For example, to search using 5 beams and output at most 50 tokens:

```python
llm = LLM(model="facebook/opt-125m")
params = BeamSearchParams(beam_width=5, max_tokens=50)
outputs = llm.generate("Hello, my name is", params)

for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

`LLM.chat`

The {class}`~vllm.LLM.chat` method implements chat functionality on top of {class}`~vllm.LLM.generate`.
In particular, it accepts input similar to [OpenAI Chat Completions API](https://platform.openai.com/docs/api-reference/chat)
and automatically applies the model's [chat template](https://huggingface.co/docs/transformers/en/chat_templating) to format the prompt.

:::{important}

In general, only instruction-tuned models have a chat template.

Base models may perform poorly as they are not trained to respond to the chat conversation.

:::

```python
llm = LLM(model="meta-llama/Meta-Llama-3-8B-Instruct")
conversation = [
    {
        "role": "system",
        "content": "You are a helpful assistant"
    },
    {
        "role": "user",
        "content": "Hello"
    },
    {
        "role": "assistant",
        "content": "Hello! How can I assist you today?"
    },
    {
        "role": "user",
        "content": "Write an essay about the importance of higher education.",
    },
]
outputs = llm.chat(conversation)
```

```
for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

A code example can be found here: <gh-file:examples/offline_inference/chat.py>

If the model doesn't have a chat template or you want to specify another one,
you can explicitly pass a chat template:

```python
from vllm.entrypoints.chat_utils import load_chat_template

# You can find a list of existing chat templates under `examples/`
custom_template = load_chat_template(chat_template="<path_to_template>")
print("Loaded chat template:", custom_template)

outputs = llm.chat(conversation, chat_template=custom_template)
```

Online Serving

Our [OpenAI-Compatible Server](#openai-compatible-server) provides endpoints that correspond to the offline APIs:

- [Completions API](#completions-api) is similar to `LLM.generate` but only accepts text.

- [Chat API](#chat-api) is similar to `LLM.chat`, accepting both text and [multi-modal inputs](#multimodal-inputs) for models with a chat template.