OpenAI-Compatible Server

Contents

vLLM provides an HTTP server that implements OpenAI's Completions API, Chat API, and more!

You can start the server via the `vllm serve` command, or through Docker:

```
vllm serve NousResearch/Meta-Llama-3-8B-Instruct --dtype auto --api-key token-abc123
```

To call the server, you can use the official OpenAI Python client, or any other HTTP client.

```
from openai import OpenAI
client = OpenAI(
    base_url="http://localhost:8000/v1",
    api_key="token-abc123",
)

completion = client.chat.completions.create(
  model="NousResearch/Meta-Llama-3-8B-Instruct",
  messages=[
    {"role": "user", "content": "Hello!"}
  ]
)

print(completion.choices[0].message)
```

Supported APIs

We currently support the following OpenAI APIs:

- Completions API (`/v1/completions`)
  - Only applicable to text generation models (`--task generate`).
  - *Note: `suffix` parameter is not supported.*
- Chat Completions API (`/v1/chat/completions`)
  - Only applicable to text generation models (`--task generate`) with a chat template.
  - *Note: `parallel_tool_calls` and `user` parameters are ignored.*
- Embeddings API (`/v1/embeddings`)
  - Only applicable to embedding models (`--task embed`).

In addition, we have the following custom APIs:

- Tokenizer API (`/tokenize`, `/detokenize`)
  - Applicable to any model with a tokenizer.
- Pooling API (`/pooling`)
  - Applicable to all pooling models.
- Score API (`/score`)
  - Only applicable to cross-encoder models (`--task score`).
- Re-rank API (`/rerank`, `/v1/rerank`, `/v2/rerank`)
  - Implements Jina AI's v1 re-rank API
  - Also compatible with Cohere's v1 & v2 re-rank APIs
  - Jina and Cohere's APIs are very similar; Jina's includes extra information in the rerank endpoint's response.
  - Only applicable to cross-encoder models (`--task score`).

Chat Template

In order for the language model to support chat protocol, vLLM requires the model to include a chat template in its tokenizer configuration. The chat template is a Jinja2 te
how are roles, messages, and other chat-specific tokens are encoded in the

v0.7.2

An example chat template for `NousResearch/Meta-Llama-3-8B-Instruct` can be found here

Some models do not provide a chat template even though they are instruction/chat fine-tuned. For those model, you can manually specify their chat template in the `--chat-template` parameter with the file path to the chat template, or the template in string form. Without a chat template, the server will not be able to process chat and all chat requests will error.

```
vllm serve <model> --chat-template ./path-to-chat-template.jinja
```

vLLM community provides a set of chat templates for popular models. You can find them under the examples directory.

With the inclusion of multi-modal chat APIs, the OpenAI spec now accepts chat messages in a new format which specifies both a `type` and a `text` field. An example is provided below:

```
completion = client.chat.completions.create(
  model="NousResearch/Meta-Llama-3-8B-Instruct",
  messages=[
    {"role": "user", "content": [{"type": "text", "text": "Classify this sentiment: vLl
  ]
)
```

Most chat templates for LLMs expect the `content` field to be a string, but there are some newer models like `meta-llama/Llama-Guard-3-1B` that expect the content to be formatted according to the OpenAI schema in the request. vLLM provides best-effort support to detect this automatically, which is logged as a string like *"Detected the chat template content format to be..."*, and internally converts incoming requests to match the detected format, which can be one of:

- `"string"`: A string.
  - Example: `"Hello world"`
- `"openai"`: A list of dictionaries, similar to OpenAI schema.
  - Example: `[{"type": "text", "text": "Hello world!"}]`

If the result is not what you expect, you can set the `--chat-template-content-format` CLI argument to override which format to use.

v0.7.2

Extra Parameters

vLLM supports a set of parameters that are not part of the OpenAI API. In order to use them, you can pass them as extra parameters in the OpenAI client. Or directly merge them into the JSON payload if you are using HTTP call directly.

```
completion = client.chat.completions.create(
  model="NousResearch/Meta-Llama-3-8B-Instruct",
  messages=[
    {"role": "user", "content": "Classify this sentiment: vLLM is wonderful!"}
  ],
  extra_body={
    "guided_choice": ["positive", "negative"]
  }
)
```

Extra HTTP Headers

Only `X-Request-Id` HTTP request header is supported for now. It can be enabled with `--enable-request-id-headers`.

> Note that enablement of the headers can impact performance significantly at high QPS rates. We recommend implementing HTTP headers at the router level (e.g. via Istio), rather than within the vLLM layer for this reason. See this PR for more details.

```
completion = client.chat.completions.create(
  model="NousResearch/Meta-Llama-3-8B-Instruct",
  messages=[
    {"role": "user", "content": "Classify this sentiment: vLLM is wonderful!"}
  ],
  extra_headers={
    "x-request-id": "sentiment-classification-00001",
  }
)
print(completion._request_id)

completion = client.completions.create(
  model="NousResearch/Meta-Llama-3-8B-Instruct",
  prompt="A robot may not injure a human being",
  extra_headers={
    "x-request-id": "completion-test",
  }
)
print(completion._request_id)
```

CLI Reference

`vllm serve`

The `vllm serve` command is used to launch the OpenAI-compatible server.

```
usage: vllm serve [-h] [--host HOST] [--port PORT]
                  [--uvicorn-log-level {debug,info,warning,error,critical,trace}]
                  [--allow-credentials] [--allowed-origins ALLOWED_ORIGINS]
                  [--allowed-methods ALLOWED_METHODS]
                  [--allowed-headers ALLOWED_HEADERS] [--api-key API_KEY]
                  [--lora-modules LORA_MODULES [LORA_MODULES ...]]
                  [--prompt-adapters PROMPT_ADAPTERS [PROMPT_ADAPTERS ...]]
                  [--chat-template CHAT_TEMPLATE]
                  [--chat-template-content-format {auto,string,openai}]
                  [--response-role RESPONSE_ROLE] [--ssl-keyfile SSL_KEYFILE]
                  [--ssl-certfile SSL_CERTFILE] [--ssl-ca-certs SSL_CA_CERTS]
                  [--ssl-cert-reqs SSL_CERT_REQS] [--root-path ROOT_PATH]
                  [--middleware MIDDLEWARE] [--return-tokens-as-token-ids]
                  [--disable-frontend-multiprocessing]
                  [--enable-request-id-headers] [--enable-auto-tool-choice]
                  [--enable-reasoning] [--reasoning-parser {deepseek_r1}]
                  [--tool-call-parser {granite-20b-fc,granite,hermes,internlm,jamba,ll
                  [--tool-parser-plugin TOOL_PARSER_PLUGIN] [--model MODEL]
                  [--task {auto,generate,embedding,embed,classify,score,reward}]
                  [--tokenizer TOKENIZER] [--skip-tokenizer-init]
                  [--revision REVISION] [--code-revision CODE_REVISION]
                  [--tokenizer-revision TOKENIZER_REVISION]
                  [--tokenizer-mode {auto,slow,mistral}] [--trust-remote-code]
                  [--allowed-local-media-path ALLOWED_LOCAL_MEDIA_PATH]
                  [--download-dir DOWNLOAD_DIR]
                  [--load-format {auto,pt,safetensors,npcache,dummy,tensorizer,sharded
                  [--config-format {auto,hf,mistral}]
                  [--dtype {auto,half,float16,bfloat16,float,float32}]
                  [--kv-cache-dtype {auto,fp8,fp8_e5m2,fp8_e4m3}]
                  [--max-model-len MAX_MODEL_LEN]
                  [--guided-decoding-backend {outlines,lm-format-enforcer,xgrammar}]
                  [--logits-processor-pattern LOGITS_PROCESSOR_PATTERN]
                  [--distributed-executor-backend {ray,mp,uni,external_launcher}]
                  [--pipeline-parallel-size PIPELINE_PARALLEL_SIZE]
                  [--tensor-parallel-size TENSOR_PARALLEL_SIZE]
                  [--max-parallel-loading-workers MAX_PARALLEL_LOADING_WORKERS]
                  [--ray-workers-use-nsight] [--block-size {8,16,32,64,128}]
                  [--enable-prefix-caching | --no-enable-prefix-caching]
                  [--disable-sliding-window] [--use-v2-block-manager]
                  [--num-lookahead-slots NUM_LOOKAHEAD_SLOTS] [--seed SEED]
                  [--swap-space SWAP_SPACE] [--cpu-offload-gb CPU_OFFLOAD_GB]
                  [--gpu-memory-utilization GPU_MEMORY_UTILIZATION]
                  [--num-gpu-blocks-override NUM_GPU_BLOCKS_OVERRIDE]
                  [--max-num-batched-tokens MAX_NUM_BATCHED_TOKENS]
                  [--max-num-seqs MAX_NUM_SEQS] [--max-logprobs MAX_LOGPROBS]
                  [--disable-log-stats]
                  [--quantization {aqlm,awq,deepspeedfp,tpu_int8,fp8,fbgemm_fp8,modelo
                  [--rope-scaling ROPE_SCALING] [--rope-theta ROPE_THETA]
                  [--hf-overrides HF_OVERRIDES] [--enforce-eager]
                  [--max-seq-len-to-capture MAX_SEQ_LEN_TO_CAPTURE]
                  [--disable-custom-all-reduce]
                  [--tokenizer-pool-size TOKENIZER_POOL_SIZE]
                  [--tokenizer-pool-type TOKENIZER_POOL_TYPE]
                  [--tokenizer-pool-extra-config TOKENIZER_POOL_EXTRA_CONFIG]
                  [--limit-mm-per-prompt LIMIT_MM_PER_PROMPT]
```

v0.7.2

```
                 [--mm-processor-kwargs MM_PROCESSOR_KWARGS]
                 [--disable-mm-preprocessor-cache] [--enable-lora]
                 [--enable-lora-bias] [--max-loras MAX_LORAS]
                 [--max-lora-rank MAX_LORA_RANK]
                 [--lora-extra-vocab-size LORA_EXTRA_VOCAB_SIZE]
                 [--lora-dtype {auto,float16,bfloat16}]
                 [--long-lora-scaling-factors LONG_LORA_SCALING_FACTORS]
                 [--max-cpu-loras MAX_CPU_LORAS] [--fully-sharded-loras]
                 [--enable-prompt-adapter]
                 [--max-prompt-adapters MAX_PROMPT_ADAPTERS]
                 [--max-prompt-adapter-token MAX_PROMPT_ADAPTER_TOKEN]
                 [--device {auto,cuda,neuron,cpu,openvino,tpu,xpu,hpu}]
                 [--num-scheduler-steps NUM_SCHEDULER_STEPS]
                 [--multi-step-stream-outputs [MULTI_STEP_STREAM_OUTPUTS]]
                 [--scheduler-delay-factor SCHEDULER_DELAY_FACTOR]
                 [--enable-chunked-prefill [ENABLE_CHUNKED_PREFILL]]
                 [--speculative-model SPECULATIVE_MODEL]
                 [--speculative-model-quantization {aqlm,awq,deepspeedfp,tpu_int8,fp8
                 [--num-speculative-tokens NUM_SPECULATIVE_TOKENS]
                 [--speculative-disable-mqa-scorer]
                 [--speculative-draft-tensor-parallel-size SPECULATIVE_DRAFT_TENSOR_PA
                 [--speculative-max-model-len SPECULATIVE_MAX_MODEL_LEN]
                 [--speculative-disable-by-batch-size SPECULATIVE_DISABLE_BY_BATCH_SI
                 [--ngram-prompt-lookup-max NGRAM_PROMPT_LOOKUP_MAX]
                 [--ngram-prompt-lookup-min NGRAM_PROMPT_LOOKUP_MIN]
                 [--spec-decoding-acceptance-method {rejection_sampler,typical_accepta
                 [--typical-acceptance-sampler-posterior-threshold TYPICAL_ACCEPTANCE_
                 [--typical-acceptance-sampler-posterior-alpha TYPICAL_ACCEPTANCE_SAMP
                 [--disable-logprobs-during-spec-decoding [DISABLE_LOGPROBS_DURING_SPE
                 [--model-loader-extra-config MODEL_LOADER_EXTRA_CONFIG]
                 [--ignore-patterns IGNORE_PATTERNS]
                 [--preemption-mode PREEMPTION_MODE]
                 [--served-model-name SERVED_MODEL_NAME [SERVED_MODEL_NAME ...]]
                 [--qlora-adapter-name-or-path QLORA_ADAPTER_NAME_OR_PATH]
                 [--otlp-traces-endpoint OTLP_TRACES_ENDPOINT]
                 [--collect-detailed-traces COLLECT_DETAILED_TRACES]
                 [--disable-async-output-proc]
                 [--scheduling-policy {fcfs,priority}]
                 [--override-neuron-config OVERRIDE_NEURON_CONFIG]
                 [--override-pooler-config OVERRIDE_POOLER_CONFIG]
                 [--compilation-config COMPILATION_CONFIG]
                 [--kv-transfer-config KV_TRANSFER_CONFIG]
                 [--worker-cls WORKER_CLS]
                 [--generation-config GENERATION_CONFIG]
                 [--override-generation-config OVERRIDE_GENERATION_CONFIG]
                 [--enable-sleep-mode] [--calculate-kv-scales]
                 [--disable-log-requests] [--max-log-len MAX_LOG_LEN]
                 [--disable-fastapi-docs] [--enable-prompt-tokens-details]
```

v0.7.2

Named Arguments

`--host`

Host name.

`--port`

Port number.

Default: 8000

`--uvicorn-log-level`

Possible choices: debug, info, warning, error, critical, trace

Log level for uvicorn.

Default: “info”

`--allow-credentials`

Allow credentials.

Default: False

`--allowed-origins`

Allowed origins.

Default: [‘*’]

`--allowed-methods`

Allowed methods.

Default: [‘*’]

`--allowed-headers`

Allowed headers.

Default: [‘*’]

`--api-key`

If provided, the server will require this key to be presented in the header.

`--lora-modules`

LoRA module configurations in either ‘name=path’ formator JSON forr v0.7.2
format): `'name=path'` Example (new format): `{"name": "name", "path": "lora_path", "base_model_name": "id"}`

`--prompt-adapters`

Prompt adapter configurations in the format name=path. Multiple adapters can be specified.

`--chat-template`

The file path to the chat template, or the template in single-line form for the specified model.

`--chat-template-content-format`

Possible choices: auto, string, openai

The format to render message content within a chat template.

- "string" will render the content as a string. Example: `"Hello World"`
- "openai" will render the content as a list of dictionaries, similar to OpenAI schema. Example: `[{"type": "text", "text": "Hello world!"}]`

Default: "auto"

`--response-role`

The role name to return if `request.add_generation_prompt=true`.

Default: assistant

`--ssl-keyfile`

The file path to the SSL key file.

`--ssl-certfile`

The file path to the SSL cert file.

`--ssl-ca-certs`

The CA certificates file.

`--ssl-cert-reqs`

Whether client certificate is required (see stdlib ssl module's).

Default: 0

`--root-path`

FastAPI root_path when app is behind a path based routing proxy.

`--middleware`

Additional ASGI middleware to apply to the app. We accept multiple –middleware arguments. The value should be an import path. If a function is provided, vLLM will add it to the server using `@app.middleware('http')`. If a class is provided, vLLM will add it to the server using `app.add_middleware()`.

Default: []

`--return-tokens-as-token-ids`

When `--max-logprobs` is specified, represents single tokens as strings of the form 'token_id:{token_id}' so that tokens that are not JSON-encodable can be identified.

Default: False

`--disable-frontend-multiprocessing`

If specified, will run the OpenAI frontend server in the same process as the model serving engine.

Default: False

`--enable-request-id-headers`

If specified, API server will add X-Request-Id header to responses. Caution: this hurts performance at high QPS.

Default: False

`--enable-auto-tool-choice`

Enable auto tool choice for supported models. Use `--tool-call-parser` to specify which parser to use.

Default: False

`--enable-reasoning`

Whether to enable reasoning_content for the model. If enabled, the model will be able to generate reasoning content.

Default: False

`--reasoning-parser`

Select the reasoning parser depending on the model that you're using. This is used to parse the reasoning content into OpenAI API format. Required for `--enable-reasoning`.

`--tool-call-parser`

Select the tool call parser depending on the model that you're using. This is used to parse the model-generated tool call into OpenAI API format. Required for `--enable-auto-tool-choice`.

`--tool-parser-plugin`

Special the tool parser plugin write to parse the model-generated tool into OpenAI API format, the name register in this plugin can be used in `--tool-call-parser`.

Default: “”

`--model`

Name or path of the huggingface model to use.

Default: “facebook/opt-125m”

`--task`

Possible choices: auto, generate, embedding, embed, classify, score, reward

The task to use the model for. Each vLLM instance only supports one task, even if the same model can be used for multiple tasks. When the model only supports one task, `"auto"` can be used to select it; otherwise, you must specify explicitly which task to use.

Default: “auto”

`--tokenizer`

Name or path of the huggingface tokenizer to use. If unspecified, model name or path will be used.

`--skip-tokenizer-init`

Skip initialization of tokenizer and detokenizer.

Default: False

`--revision`

The specific model version to use. It can be a branch name, a tag name, or a commit id. If unspecified, will use the default version.

`--code-revision`

The specific revision to use for the model code on Hugging Face Hub. It can be a branch name, a tag name, or a commit id. If unspecified, will use the default version.

`--tokenizer-revision`

Revision of the huggingface tokenizer to use. It can be a branch name, a tag name, or a commit id. If unspecified, will use the default version.

`--tokenizer-mode`

Possible choices: auto, slow, mistral

The tokenizer mode.

- "auto" will use the fast tokenizer if available.
- "slow" will always use the slow tokenizer.
- "mistral" will always use the *mistral_common* tokenizer.

Default: "auto"

`--trust-remote-code`

Trust remote code from huggingface.

Default: False

`--allowed-local-media-path`

Allowing API requests to read local images or videos from directories specified by the server file system. This is a security risk. Should only be enabled in trusted environments.

`--download-dir`

Directory to download and load the weights, default to the default cache dir of huggingface.

`--load-format`

Possible choices: auto, pt, safetensors, npcache, dummy, tensorizer, sharded_state, gguf, bitsandbytes, mistral, runai_streamer

The format of the model weights to load.

- "auto" will try to load the weights in the safetensors format and fa v0.7.2 bin format if safetensors format is not available.
- "pt" will load the weights in the pytorch bin format.

- "safetensors" will load the weights in the safetensors format.
- "npcache" will load the weights in pytorch format and store a numpy cache to speed up the loading.
- "dummy" will initialize the weights with random values, which is mainly for profiling.
- "tensorizer" will load the weights using tensorizer from CoreWeave. See the Tensorize vLLM Model script in the Examples section for more information.
- "runai_streamer" will load the Safetensors weights using Run:aiModel Streamer
- "bitsandbytes" will load the weights using bitsandbytes quantization.

Default: "auto"

`--config-format`

Possible choices: auto, hf, mistral

The format of the model config to load.

- "auto" will try to load the config in hf format if available else it will try to load in mistral format

Default: "ConfigFormat.AUTO"

`--dtype`

Possible choices: auto, half, float16, bfloat16, float, float32

Data type for model weights and activations.

- "auto" will use FP16 precision for FP32 and FP16 models, and BF16 precision for BF16 models.
- "half" for FP16. Recommended for AWQ quantization.
- "float16" is the same as "half".
- "bfloat16" for a balance between precision and range.
- "float" is shorthand for FP32 precision.
- "float32" for FP32 precision.

Default: "auto"

`--kv-cache-dtype`

Possible choices: auto, fp8, fp8_e5m2, fp8_e4m3

Data type for kv cache storage. If “auto”, will use model data type. CUDA 11.8+ supports fp8 (=fp8_e4m3) and fp8_e5m2. ROCm (AMD GPU) supports fp8 (=fp8_e4m3)

Default: “auto”

`--max-model-len`

Model context length. If unspecified, will be automatically derived from the model config.

`--guided-decoding-backend`

Possible choices: outlines, lm-format-enforcer, xgrammar

Which engine will be used for guided decoding (JSON schema / regex etc) by default. Currently support outlines-dev/outlines, mlc-ai/xgrammar, and noamgat/lm-format-enforcer. Can be overridden per request via guided_decoding_backend parameter.

Default: “xgrammar”

`--logits-processor-pattern`

Optional regex pattern specifying valid logits processor qualified names that can be passed with the *logits_processors* extra completion argument. Defaults to None, which allows no processors.

`--distributed-executor-backend`

Possible choices: ray, mp, uni, external_launcher

Backend to use for distributed model workers, either “ray” or “mp” (multiprocessing). If the product of pipeline_parallel_size and tensor_parallel_size is less than or equal to the number of GPUs available, “mp” will be used to keep processing on a single host. Otherwise, this will default to “ray” if Ray is installed and fail otherwise. Note that tpu only supports Ray for distributed inference.

`--pipeline-parallel-size, -pp`

Number of pipeline stages.

Default: 1

`--tensor-parallel-size, -tp`

Number of tensor parallel replicas.

Default: 1

`--max-parallel-loading-workers`

Load model sequentially in multiple batches, to avoid RAM OOM when using tensor parallel and large models.

`--ray-workers-use-nsight`

If specified, use nsight to profile Ray workers.

Default: False

`--block-size`

Possible choices: 8, 16, 32, 64, 128

Token block size for contiguous chunks of tokens. This is ignored on neuron devices and set to `--max-model-len`. On CUDA devices, only block sizes up to 32 are supported. On HPU devices, block size defaults to 128.

`--enable-prefix-caching, --no-enable-prefix-caching`

Enables automatic prefix caching. Use `--no-enable-prefix-caching` to disable explicitly.

`--disable-sliding-window`

Disables sliding window, capping to sliding window size.

Default: False

`--use-v2-block-manager`

[DEPRECATED] block manager v1 has been removed and SelfAttnBlockSpaceManager (i.e. block manager v2) is now the default. Setting this flag to True or False has no effect on vLLM behavior.

Default: True

`--num-lookahead-slots`

Experimental scheduling config necessary for speculative decoding. This will be replaced by speculative config in the future; it is present to enable correctness tests until then.

Default: 0

`--seed`

Random seed for operations.

Default: 0

`--swap-space`

CPU swap space size (GiB) per GPU.

Default: 4

`--cpu-offload-gb`

The space in GiB to offload to CPU, per GPU. Default is 0, which means no offloading. Intuitively, this argument can be seen as a virtual way to increase the GPU memory size. For example, if you have one 24 GB GPU and set this to 10, virtually you can think of it as a 34 GB GPU. Then you can load a 13B model with BF16 weight, which requires at least 26GB GPU memory. Note that this requires fast CPU-GPU interconnect, as part of the model is loaded from CPU memory to GPU memory on the fly in each model forward pass.

Default: 0

`--gpu-memory-utilization`

The fraction of GPU memory to be used for the model executor, which can range from 0 to 1. For example, a value of 0.5 would imply 50% GPU memory utilization. If unspecified, will use the default value of 0.9. This is a per-instance limit, and only applies to the current vLLM instance.It does not matter if you have another vLLM instance running on the same GPU. For example, if you have two vLLM instances running on the same GPU, you can set the GPU memory utilization to 0.5 for each instance.

Default: 0.9

`--num-gpu-blocks-override`

If specified, ignore GPU profiling result and use this number of GPU blocks. Used for testing preemption.

`--max-num-batched-tokens`

Maximum number of batched tokens per iteration.

`--max-num-seqs`

Maximum number of sequences per iteration.

`--max-logprobs`

Max number of log probs to return logprobs is specified in SamplingParams.

Default: 20

`--disable-log-stats`

Disable logging statistics.

Default: False

`--quantization, -q`

Possible choices: aqlm, awq, deepspeedfp, tpu_int8, fp8, fbgemm_fp8, modelopt, marlin, gguf, gptq_marlin_24, gptq_marlin, awq_marlin, gptq, compressed-tensors, bitsandbytes, qqq, hqq, experts_int8, neuron_quant, ipex, quark, moe_wna16, None

Method used to quantize the weights. If None, we first check the *quantization_config* attribute in the model config file. If that is None, we assume the model weights are not quantized and use *dtype* to determine the data type of the weights.

`--rope-scaling`

RoPE scaling configuration in JSON format. For example, `{"rope_type":"dynamic","factor":2.0}`

`--rope-theta`

RoPE theta. Use with *rope_scaling*. In some cases, changing the RoPE theta improves the performance of the scaled model.

`--hf-overrides`

Extra arguments for the HuggingFace config. This should be a JSON string that will be parsed into a dictionary.

`--enforce-eager`

Always use eager-mode PyTorch. If False, will use eager mode and CUDA graph in hybrid for maximal performance and flexibility.

Default: False

`--max-seq-len-to-capture`

Maximum sequence length covered by CUDA graphs. When a sequence has context length larger than this, we fall back to eager mode. Additionally for encoder-
sequence length of the encoder input is larger than this, we fall back t

v0.7.2

Default: 8192

`--disable-custom-all-reduce`

See ParallelConfig.

Default: False

`--tokenizer-pool-size`

Size of tokenizer pool to use for asynchronous tokenization. If 0, will use synchronous tokenization.

Default: 0

`--tokenizer-pool-type`

Type of tokenizer pool to use for asynchronous tokenization. Ignored if tokenizer_pool_size is 0.

Default: “ray”

`--tokenizer-pool-extra-config`

Extra config for tokenizer pool. This should be a JSON string that will be parsed into a dictionary. Ignored if tokenizer_pool_size is 0.

`--limit-mm-per-prompt`

For each multimodal plugin, limit how many input instances to allow for each prompt. Expects a comma-separated list of items, e.g.: *image=16,video=2* allows a maximum of 16 images and 2 videos per prompt. Defaults to 1 for each modality.

`--mm-processor-kwargs`

Overrides for the multimodal input mapping/processing, e.g., image processor. For example: `{"num_crops": 4}`.

`--disable-mm-preprocessor-cache`

If true, then disables caching of the multi-modal preprocessor/mapper. (not recommended)

Default: False

`--enable-lora`

If True, enable handling of LoRA adapters.

Default: False

`--enable-lora-bias`

If True, enable bias for LoRA adapters.

Default: False

`--max-loras`

Max number of LoRAs in a single batch.

Default: 1

`--max-lora-rank`

Max LoRA rank.

Default: 16

`--lora-extra-vocab-size`

Maximum size of extra vocabulary that can be present in a LoRA adapter (added to the base model vocabulary).

Default: 256

`--lora-dtype`

Possible choices: auto, float16, bfloat16

Data type for LoRA. If auto, will default to base model dtype.

Default: "auto"

`--long-lora-scaling-factors`

Specify multiple scaling factors (which can be different from base model scaling factor - see eg. Long LoRA) to allow for multiple LoRA adapters trained with those scaling factors to be used at the same time. If not specified, only adapters trained with the base model scaling factor are allowed.

`--max-cpu-loras`

Maximum number of LoRAs to store in CPU memory. Must be >= than max_loras. Defaults to max_loras.

`--fully-sharded-loras`

By default, only half of the LoRA computation is sharded with tensor parallelism. Enabling this will use the fully sharded layers. At high sequence length, max rank or tensor parallel size, this is likely faster.

Default: False

`--enable-prompt-adapter`

If True, enable handling of PromptAdapters.

Default: False

`--max-prompt-adapters`

Max number of PromptAdapters in a batch.

Default: 1

`--max-prompt-adapter-token`

Max number of PromptAdapters tokens

Default: 0

`--device`

Possible choices: auto, cuda, neuron, cpu, openvino, tpu, xpu, hpu

Device type for vLLM execution.

Default: “auto”

`--num-scheduler-steps`

Maximum number of forward steps per scheduler call.

Default: 1

`--multi-step-stream-outputs`

If False, then multi-step will stream outputs at the end of all steps

Default: True

`--scheduler-delay-factor`

Apply a delay (of delay factor multiplied by previous prompt latency) before scheduling next prompt.

Default: 0.0

`--enable-chunked-prefill`

If set, the prefill requests can be chunked based on the max_num_batched_tokens.

`--speculative-model`

The name of the draft model to be used in speculative decoding.

`--speculative-model-quantization`

Possible choices: aqlm, awq, deepspeedfp, tpu_int8, fp8, fbgemm_fp8, modelopt, marlin, gguf, gptq_marlin_24, gptq_marlin, awq_marlin, gptq, compressed-tensors, bitsandbytes, qqq, hqq, experts_int8, neuron_quant, ipex, quark, moe_wna16, None

Method used to quantize the weights of speculative model. If None, we first check the *quantization_config* attribute in the model config file. If that is None, we assume the model weights are not quantized and use *dtype* to determine the data type of the weights.

`--num-speculative-tokens`

The number of speculative tokens to sample from the draft model in speculative decoding.

`--speculative-disable-mqa-scorer`

If set to True, the MQA scorer will be disabled in speculative and fall back to batch expansion

Default: False

`--speculative-draft-tensor-parallel-size, -spec-draft-tp`

Number of tensor parallel replicas for the draft model in speculative decoding.

`--speculative-max-model-len`

The maximum sequence length supported by the draft model. Sequences over this length will skip speculation.

`--speculative-disable-by-batch-size`

Disable speculative decoding for new incoming requests if the number of enqueue requests is larger than this value.

`--ngram-prompt-lookup-max`

Max size of window for ngram prompt lookup in speculative decoding.

`--ngram-prompt-lookup-min`

Min size of window for ngram prompt lookup in speculative decoding.

`--spec-decoding-acceptance-method`

Possible choices: rejection_sampler, typical_acceptance_sampler

Specify the acceptance method to use during draft token verification in speculative decoding. Two types of acceptance routines are supported: 1) RejectionSampler which does not allow changing the acceptance rate of draft tokens, 2) TypicalAcceptanceSampler which is configurable, allowing for a higher acceptance rate at the cost of lower quality, and vice versa.

Default: “rejection_sampler”

`--typical-acceptance-sampler-posterior-threshold`

Set the lower bound threshold for the posterior probability of a token to be accepted. This threshold is used by the TypicalAcceptanceSampler to make sampling decisions during speculative decoding. Defaults to 0.09

`--typical-acceptance-sampler-posterior-alpha`

A scaling factor for the entropy-based threshold for token acceptance in the TypicalAcceptanceSampler. Typically defaults to sqrt of –typical-acceptance-sampler-posterior-threshold i.e. 0.3

`--disable-logprobs-during-spec-decoding`

If set to True, token log probabilities are not returned during speculative decoding. If set to False, log probabilities are returned according to the settings in SamplingParams. If not specified, it defaults to True. Disabling log probabilities during speculative decoding reduces latency by skipping logprob calculation in proposal sampling, target sampling, and after accepted tokens are determined.

`--model-loader-extra-config`

Extra config for model loader. This will be passed to the model loader corresponding to the chosen load_format. This should be a JSON string that will be parsed into a dictionary.

`--ignore-patterns`

The pattern(s) to ignore when loading the model.Default to *original/*** v0.7.2
loading of llama’s checkpoints.

Default: []

`--preemption-mode`

If ‘recompute’, the engine performs preemption by recomputing; If ‘swap’, the engine performs preemption by block swapping.

`--served-model-name`

The model name(s) used in the API. If multiple names are provided, the server will respond to any of the provided names. The model name in the model field of a response will be the first name in this list. If not specified, the model name will be the same as the `--model` argument. Noted that this name(s) will also be used in *model_name* tag content of prometheus metrics, if multiple names provided, metrics tag will take the first one.

`--qlora-adapter-name-or-path`

Name or path of the QLoRA adapter.

`--otlp-traces-endpoint`

Target URL to which OpenTelemetry traces will be sent.

`--collect-detailed-traces`

Valid choices are model,worker,all. It makes sense to set this only if `--otlp-traces-endpoint` is set. If set, it will collect detailed traces for the specified modules. This involves use of possibly costly and or blocking operations and hence might have a performance impact.

`--disable-async-output-proc`

Disable async output processing. This may result in lower performance.

Default: False

`--scheduling-policy`

Possible choices: fcfs, priority

The scheduling policy to use. “fcfs” (first come first served, i.e. requests are handled in order of arrival; default) or “priority” (requests are handled based on given priority (lower value means earlier handling) and time of arrival deciding any ties).

Default: “fcfs”

`--override-neuron-config`

Override or set neuron device configuration. e.g. `{"cast_logits_dtype": "bloat16"}`.

`--override-pooler-config`

Override or set the pooling method for pooling models. e.g. `{"pooling_type": "mean", "normalize": false}`.

`--compilation-config, -O`

torch.compile configuration for the model.When it is a number (0, 1, 2, 3), it will be interpreted as the optimization level. NOTE: level 0 is the default level without any optimization. level 1 and 2 are for internal testing only. level 3 is the recommended level for production. To specify the full compilation config, use a JSON string. Following the convention of traditional compilers, using -O without space is also supported. -O3 is equivalent to -O 3.

`--kv-transfer-config`

The configurations for distributed KV cache transfer. Should be a JSON string.

`--worker-cls`

The worker class to use for distributed execution.

Default: “auto”

`--generation-config`

The folder path to the generation config. Defaults to None, no generation config is loaded, vLLM defaults will be used. If set to ‘auto’, the generation config will be loaded from model path. If set to a folder path, the generation config will be loaded from the specified folder path. If *max_new_tokens* is specified in generation config, then it sets a server-wide limit on the number of output tokens for all requests.

`--override-generation-config`

Overrides or sets generation config in JSON format. e.g. `{"temperature": 0.5}`. If used with –generation-config=auto, the override parameters will be merged with the default config from the model. If generation-config is None, only the override parameters are used.

`--enable-sleep-mode`

Enable sleep mode for the engine. (only cuda platform is supported)

v0.7.2

Default: False

`--calculate-kv-scales`

This enables dynamic calculation of k_scale and v_scale when kv-cache-dtype is fp8. If calculate-kv-scales is false, the scales will be loaded from the model checkpoint if available. Otherwise, the scales will default to 1.0.

Default: False

`--disable-log-requests`

Disable logging requests.

Default: False

`--max-log-len`

Max number of prompt characters or prompt ID numbers being printed in log.

Default: Unlimited

`--disable-fastapi-docs`

Disable FastAPI's OpenAPI schema, Swagger UI, and ReDoc endpoint.

Default: False

`--enable-prompt-tokens-details`

If set to True, enable prompt_tokens_details in usage.

Default: False

Configuration file

You can load CLI arguments via a YAML config file. The argument names must be the long form of those outlined above.

For example:

```yaml
# config.yaml

host: "127.0.0.1"
port: 6379
uvicorn-log-level: "info"
```

To use the above config file:

```
vllm serve SOME_MODEL --config config.yaml
```

Note

In case an argument is supplied simultaneously using command line and the config file, the value from the command line will take precedence. The order of priorities is `command line > config file values > defaults`.

API Reference

Completions API

Our Completions API is compatible with OpenAI's Completions API; you can use the official OpenAI Python client to interact with it.

Code example: examples/online_serving/openai_completion_client.py

Extra parameters

The following sampling parameters are supported.

```
    use_beam_search: bool = False
    top_k: Optional[int] = None
    min_p: Optional[float] = None
    repetition_penalty: Optional[float] = None
    length_penalty: float = 1.0
    stop_token_ids: Optional[List[int]] = Field(default_factory=list)
    include_stop_str_in_output: bool = False
    ignore_eos: bool = False
    min_tokens: int = 0
    skip_special_tokens: bool = True
    spaces_between_special_tokens: bool = True
    truncate_prompt_tokens: Optional[Annotated[int, Field(ge=1)]] = None
    allowed_token_ids: Optional[List[int]] = None
    prompt_logprobs: Optional[int] = None
```

The following extra parameters are supported:

```
    add_special_tokens: bool = Field(
        default=True,
        description=(
            "If true (the default), special tokens (e.g. BOS) will be added to "
            "the prompt."),
    )
    response_format: Optional[ResponseFormat] = Field(
        default=None,
        description=
        ("Similar to chat completion, this parameter specifies the format of "
         "output. Only {'type': 'json_object'}, {'type': 'json_schema'} or "
         "{'type': 'text' } is supported."),
    )
    guided_json: Optional[Union[str, dict, BaseModel]] = Field(
        default=None,
        description="If specified, the output will follow the JSON schema.",
    )
    guided_regex: Optional[str] = Field(
        default=None,
        description=(
            "If specified, the output will follow the regex pattern."),
    )
    guided_choice: Optional[List[str]] = Field(
        default=None,
        description=(
            "If specified, the output will be exactly one of the choices."),
    )
    guided_grammar: Optional[str] = Field(
        default=None,
        description=(
            "If specified, the output will follow the context free grammar."),
    )
    guided_decoding_backend: Optional[str] = Field(
        default=None,
        description=(
            "If specified, will override the default guided decoding backend "
            "of the server for this specific request. If set, must be one of "
            "'outlines' / 'lm-format-enforcer'"))
    guided_whitespace_pattern: Optional[str] = Field(
        default=None,
        description=(
            "If specified, will override the default whitespace pattern "
            "for guided json decoding."))
    priority: int = Field(
        default=0,
        description=(
            "The priority of the request (lower means earlier handling; "
            "default: 0). Any priority other than 0 will raise an error "
            "if the served model does not use priority scheduling."))
    logits_processors: Optional[LogitsProcessors] = Field(
        default=None,
        description=(
            "A list of either qualified names of logits processors, or
            "constructor objects, to apply when sampling. A constructor is "
            "a JSON object with a required 'qualname' field specifying the "
```

v0.7.2

```
            "qualified name of the processor class/factory, and optional "
            "'args' and 'kwargs' fields containing positional and keyword "
            "arguments. For example: {'qualname': "
            "'my_module.MyLogitsProcessor', 'args': [1, 2], 'kwargs': "
            "{'param': 'value'}}."))
```

Chat API

Our Chat API is compatible with OpenAI's Chat Completions API; you can use the official OpenAI Python client to interact with it.

We support both Vision- and Audio-related parameters; see our Multimodal Inputs guide for more information.

- *Note:* `image_url.detail` *parameter is not supported.*

Code example: examples/online_serving/openai_chat_completion_client.py

Extra parameters

The following sampling parameters are supported.

```
    best_of: Optional[int] = None
    use_beam_search: bool = False
    top_k: Optional[int] = None
    min_p: Optional[float] = None
    repetition_penalty: Optional[float] = None
    length_penalty: float = 1.0
    stop_token_ids: Optional[List[int]] = Field(default_factory=list)
    include_stop_str_in_output: bool = False
    ignore_eos: bool = False
    min_tokens: int = 0
    skip_special_tokens: bool = True
    spaces_between_special_tokens: bool = True
    truncate_prompt_tokens: Optional[Annotated[int, Field(ge=1)]] = None
    prompt_logprobs: Optional[int] = None
```

The following extra parameters are supported:

```
    echo: bool = Field(
        default=False,
        description=(
            "If true, the new message will be prepended with the last message "
            "if they belong to the same role."),
    )
    add_generation_prompt: bool = Field(
        default=True,
        description=
        ("If true, the generation prompt will be added to the chat template. "
         "This is a parameter used by chat template in tokenizer config of the "
         "model."),
    )
    continue_final_message: bool = Field(
        default=False,
        description=
        ("If this is set, the chat will be formatted so that the final "
         "message in the chat is open-ended, without any EOS tokens. The "
         "model will continue this message rather than starting a new one. "
         "This allows you to \"prefill\" part of the model's response for it. "
         "Cannot be used at the same time as `add_generation_prompt`."),
    )
    add_special_tokens: bool = Field(
        default=False,
        description=(
            "If true, special tokens (e.g. BOS) will be added to the prompt "
            "on top of what is added by the chat template. "
            "For most models, the chat template takes care of adding the "
            "special tokens so this should be set to false (as is the "
            "default)."),
    )
    documents: Optional[List[Dict[str, str]]] = Field(
        default=None,
        description=
        ("A list of dicts representing documents that will be accessible to "
         "the model if it is performing RAG (retrieval-augmented generation)."
         " If the template does not support RAG, this argument will have no "
         "effect. We recommend that each document should be a dict containing "
         "\"title\" and \"text\" keys."),
    )
    chat_template: Optional[str] = Field(
        default=None,
        description=(
            "A Jinja template to use for this conversion. "
            "As of transformers v4.44, default chat template is no longer "
            "allowed, so you must provide a chat template if the tokenizer "
            "does not define one."),
    )
    chat_template_kwargs: Optional[Dict[str, Any]] = Field(
        default=None,
        description=("Additional kwargs to pass to the template re
                     "Will be accessible by the chat template."),
    )
    guided_json: Optional[Union[str, dict, BaseModel]] = Field(
        default=None,
```

v0.7.2

```
        description=("If specified, the output will follow the JSON schema."),
    )
    guided_regex: Optional[str] = Field(
        default=None,
        description=(
            "If specified, the output will follow the regex pattern."),
    )
    guided_choice: Optional[List[str]] = Field(
        default=None,
        description=(
            "If specified, the output will be exactly one of the choices."),
    )
    guided_grammar: Optional[str] = Field(
        default=None,
        description=(
            "If specified, the output will follow the context free grammar."),
    )
    guided_decoding_backend: Optional[str] = Field(
        default=None,
        description=(
            "If specified, will override the default guided decoding backend "
            "of the server for this specific request. If set, must be either "
            "'outlines' / 'lm-format-enforcer'"))
    guided_whitespace_pattern: Optional[str] = Field(
        default=None,
        description=(
            "If specified, will override the default whitespace pattern "
            "for guided json decoding."))
    priority: int = Field(
        default=0,
        description=(
            "The priority of the request (lower means earlier handling; "
            "default: 0). Any priority other than 0 will raise an error "
            "if the served model does not use priority scheduling."))
    request_id: str = Field(
        default_factory=lambda: f"{random_uuid()}",
        description=(
            "The request_id related to this request. If the caller does "
            "not set it, a random_uuid will be generated. This id is used "
            "through out the inference process and return in response."))
    logits_processors: Optional[LogitsProcessors] = Field(
        default=None,
        description=(
            "A list of either qualified names of logits processors, or "
            "constructor objects, to apply when sampling. A constructor is "
            "a JSON object with a required 'qualname' field specifying the "
            "qualified name of the processor class/factory, and optional "
            "'args' and 'kwargs' fields containing positional and keyword "
            "arguments. For example: {'qualname': "
            "'my_module.MyLogitsProcessor', 'args': [1, 2], 'kwargs': "
            "{'param': 'value'}}."))
```

v0.7.2

Embeddings API

Our Embeddings API is compatible with OpenAI's Embeddings API; you can use the official OpenAI Python client to interact with it.

If the model has a chat template, you can replace `inputs` with a list of `messages` (same schema as Chat API) which will be treated as a single prompt to the model.

> **Tip**
>
> This enables multi-modal inputs to be passed to embedding models, see this page for details.

Code example: examples/online_serving/openai_embedding_client.py

Extra parameters

The following pooling parameters are supported.

```
additional_data: Optional[Any] = None
```

The following extra parameters are supported by default:

```
add_special_tokens: bool = Field(
    default=True,
    description=(
        "If true (the default), special tokens (e.g. BOS) will be added to "
        "the prompt."),
)
priority: int = Field(
    default=0,
    description=(
        "The priority of the request (lower means earlier handling; "
        "default: 0). Any priority other than 0 will raise an error "
        "if the served model does not use priority scheduling."))
```

For chat-like input (i.e. if `messages` is passed), these extra parameters are s

v0.7.2

```
    add_special_tokens: bool = Field(
        default=False,
        description=(
            "If true, special tokens (e.g. BOS) will be added to the prompt "
            "on top of what is added by the chat template. "
            "For most models, the chat template takes care of adding the "
            "special tokens so this should be set to false (as is the "
            "default)."),
    )
    chat_template: Optional[str] = Field(
        default=None,
        description=(
            "A Jinja template to use for this conversion. "
            "As of transformers v4.44, default chat template is no longer "
            "allowed, so you must provide a chat template if the tokenizer "
            "does not define one."),
    )
    chat_template_kwargs: Optional[Dict[str, Any]] = Field(
        default=None,
        description=("Additional kwargs to pass to the template renderer. "
                     "Will be accessible by the chat template."),
    )
    priority: int = Field(
        default=0,
        description=(
            "The priority of the request (lower means earlier handling; "
            "default: 0). Any priority other than 0 will raise an error "
            "if the served model does not use priority scheduling."))
```

Tokenizer API

Our Tokenizer API is a simple wrapper over HuggingFace-style tokenizers. It consists of two endpoints:

- `/tokenize` corresponds to calling `tokenizer.encode()`.
- `/detokenize` corresponds to calling `tokenizer.decode()`.

Pooling API

Our Pooling API encodes input prompts using a pooling model and returns the corresponding hidden states.

The input format is the same as Embeddings API, but the output data can
nested list, not just a 1-D list of floats.

v0.7.2

Code example: examples/online_serving/openai_pooling_client.py

Score API

Our Score API applies a cross-encoder model to predict scores for sentence pairs. Usually, the score for a sentence pair refers to the similarity between two sentences, on a scale of 0 to 1.

You can find the documentation for these kind of models at sbert.net.

Code example: examples/online_serving/openai_cross_encoder_score.py

Single inference

You can pass a string to both `text_1` and `text_2`, forming a single sentence pair.

Request:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/score' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "model": "BAAI/bge-reranker-v2-m3",
  "encoding_format": "float",
  "text_1": "What is the capital of France?",
  "text_2": "The capital of France is Paris."
}'
```

Response:

```
{
  "id": "score-request-id",
  "object": "list",
  "created": 693447,
  "model": "BAAI/bge-reranker-v2-m3",
  "data": [
    {
      "index": 0,
      "object": "score",
      "score": 1
    }
  ],
  "usage": {}
}
```

Batch inference

You can pass a string to `text_1` and a list to `text_2`, forming multiple sentence pairs where each pair is built from `text_1` and a string in `text_2`. The total number of pairs is `len(text_2)`.

Request:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/score' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "model": "BAAI/bge-reranker-v2-m3",
  "text_1": "What is the capital of France?",
  "text_2": [
    "The capital of Brazil is Brasilia.",
    "The capital of France is Paris."
  ]
}'
```

Response:

```
{
  "id": "score-request-id",
  "object": "list",
  "created": 693570,
  "model": "BAAI/bge-reranker-v2-m3",
  "data": [
    {
      "index": 0,
      "object": "score",
      "score": 0.001094818115234375
    },
    {
      "index": 1,
      "object": "score",
      "score": 1
    }
  ],
  "usage": {}
}
```

You can pass a list to both `text_1` and `text_2`, forming multiple sentence pairs where each pair is built from a string in `text_1` and the corresponding string in `text_2` (similar to `zip()`). The total number of pairs is `len(text_2)`.

Request:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/score' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "model": "BAAI/bge-reranker-v2-m3",
  "encoding_format": "float",
  "text_1": [
    "What is the capital of Brazil?",
    "What is the capital of France?"
  ],
  "text_2": [
    "The capital of Brazil is Brasilia.",
    "The capital of France is Paris."
  ]
}'
```

Response:

```
{
  "id": "score-request-id",
  "object": "list",
  "created": 693447,
  "model": "BAAI/bge-reranker-v2-m3",
  "data": [
    {
      "index": 0,
      "object": "score",
      "score": 1
    },
    {
      "index": 1,
      "object": "score",
      "score": 1
    }
  ],
  "usage": {}
}
```

Extra parameters

The following pooling parameters are supported.

```
    additional_data: Optional[Any] = None
```

The following extra parameters are supported:

```
    priority: int = Field(
        default=0,
        description=(
            "The priority of the request (lower means earlier handling; "
            "default: 0). Any priority other than 0 will raise an error "
            "if the served model does not use priority scheduling."))
```

Re-rank API

Our Re-rank API applies a cross-encoder model to predict relevant scores between a single query, and each of a list of documents. Usually, the score for a sentence pa
similarity between two sentences, on a scale of 0 to 1.

v0.7.2

You can find the documentation for these kind of models at sbert.net.

The rerank endpoints support popular re-rank models such as `BAAI/bge-reranker-base` and other models supporting the `score` task. Additionally, `/rerank`, `/v1/rerank`, and `/v2/rerank` endpoints are compatible with both Jina AI's re-rank API interface and Cohere's re-rank API interface to ensure compatibility with popular open-source tools.

Code example: examples/online_serving/jinaai_rerank_client.py

Example Request

Note that the `top_n` request parameter is optional and will default to the length of the `documents` field. Result documents will be sorted by relevance, and the `index` property can be used to determine original order.

Request:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/v1/rerank' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "model": "BAAI/bge-reranker-base",
  "query": "What is the capital of France?",
  "documents": [
    "The capital of Brazil is Brasilia.",
    "The capital of France is Paris.",
    "Horses and cows are both animals"
  ]
}'
```

Response:

```
{
  "id": "rerank-fae51b2b664d4ed38f5969b612edff77",
  "model": "BAAI/bge-reranker-base",
  "usage": {
    "total_tokens": 56
  },
  "results": [
    {
      "index": 1,
      "document": {
        "text": "The capital of France is Paris."
      },
      "relevance_score": 0.99853515625
    },
    {
      "index": 0,
      "document": {
        "text": "The capital of Brazil is Brasilia."
      },
      "relevance_score": 0.0005860328674316406
    }
  ]
}
```

Extra parameters

The following pooling parameters are supported.

```
    additional_data: Optional[Any] = None
```

The following extra parameters are supported:

```
    priority: int = Field(
        default=0,
        description=(
            "The priority of the request (lower means earlier handling; "
            "default: 0). Any priority other than 0 will raise an error "
            "if the served model does not use priority scheduling."))
```