[![Hugging Face's logo](/front/assets/huggingface_logo-noborder.svg) Hugging Face](/)

* [Models](/models)
* [Datasets](/datasets)
* [Spaces](/spaces)
* [Posts](/posts)
* [Docs](/docs)
* [Enterprise](/enterprise)
* [Pricing](/pricing)
* * * * *

* [Log In](/login)
* [Sign Up](/join)

#

[](/NousResearch)

[NousResearch](/NousResearch)

/

[Hermes-2-Pro-Llama-3-8B](/NousResearch/Hermes-2-Pro-Llama-3-8B)

like 416

Follow

NousResearch 1.2k

[Text Generation](/models?pipeline_tag=text-generation)[Transformers](/models?library=transformers)[Safetensors](/models?library=safetensors)

teknium/OpenHermes-2.5

[English](/models?language=en)[llama](/models?other=llama)[Llama-3](/models?other=Llama-3)[instruct](/models?other=instruct)[finetune](/models?other=finetune)[chatml](/models?other=chatml)[DPO](/models?other=DPO)[RLHF](/models?other=RLHF)[gpt4](/models?other=gpt4)[synthetic data](/models?other=synthetic+data)[distillation](/models?other=distillation)[function calling](/models?other=function+calling)[json mode](/models?other=json+mode)[axolotl](/models?other=axolotl)[conversational](/models?other=conversational)[text-generation-inference](/models?other=text-generation-inference)[Inference Endpoints](/models?other=endpoints_compatible)

License: llama3

[Model card](/NousResearch/Hermes-2-Pro-Llama-3-8B)[Files Files and versions](/NousResearch/Hermes-2-Pro-Llama-3-8B/tree/main)[Community 27](/NousResearch/Hermes-2-Pro-Llama-3-8B/discussions)

Train

Deploy

Use this model

[interstellarninja](/interstellarninja) commited on Sep 14, 2024

Commit

f798274

·

verified ·

1 Parent(s): [e52178d](/NousResearch/Hermes-2-Pro-Llama-3-8B/commit/e52178d17276cb3738f158b2ec6d6a8a0140bf7d)

fixing typo with extra '}; with tool use template

[Browse files](/NousResearch/Hermes-2-Pro-Llama-3-8B/tree/f798274b30e7d2d4797c369edcc0cc7473b6e6f2)

Files changed (1) hide show

1. tokenizer_config.json +1 -1

tokenizer_config.json CHANGED Viewed

| @@ -2057,7 +2057,7 @@
---|---
2057 | },
2058 | {
2059 | "name": "tool_use",
2060 | - "template": "{%- macro json_to_python_type(json_spec) %}\n{%- set basic_type_map = {\n \"string\": \"str\",\n \"number\": \"float\",\n \"integer\": \"int\",\n \"boolean\": \"bool\"\n} %}\n\n{%- if basic_type_map[json_spec.type] is defined %}\n {{- basic_type_map[json_spec.type] }}\n{%- elif json_spec.type == \"array\" %}\n {{- \"list[\" \+ json_to_python_type(json_spec|items) + \"]\"}}\n{%- elif json_spec.type == \"object\" %}\n {%- if json_spec.additionalProperties is defined %}\n {{- \"dict[str, \" \+ json_to_python_type(json_spec.additionalProperties) + ']'}}\n {%- else %}\n {{- \"dict\" }}\n {%- endif %}\n{%- elif json_spec.type is iterable %}\n {{- \"Union[\" }}\n {%- for t in json_spec.type %}\n {{- json_to_python_type({\"type\": t}) }}\n {%- if not loop.last %}\n {{- \",\" }} \n {%- endif %}\n {%- endfor %}\n {{- \"]\" }}\n{%- else %}\n {{- \"Any\" }}\n{%- endif %}\n{%- endmacro %}\n\n\n{{- bos_token }}\n{{- '<|im_start|>system\n' }}\n{{- \"You are a function calling AI model. You are provided with function signatures within <tools></tools> XML tags. You may call one or more functions to assist

with the user query. Don't make assumptions about what values to plug into functions. Here are the available tools: <tools> \" }}\n{%- for tool in tools %}\n {%- if tool.function is defined %}\n {%- set tool = tool.function %}\n {%- endif %}\n {{- '{\"type\": \"function\", \"function\": ' }}\n {{- '{\"name\": \"' \+ tool.name + '\", ' }}\n {{- '\"description\": \"' \+ tool.name + '(' }}\n {%- for param_name, param_fields in tool.parameters.properties|items %}\n {{- param_name + \": \" \+ json_to_python_type(param_fields) }}\n {%- if not loop.last %}\n {{- \", \" }}\n {%- endif %}\n {%- endfor %}\n {{- \")\" }}\n {%- if tool.return is defined %}\n {{- \" -> \" \+ json_to_python_type(tool.return) }}\n {%- endif %}\n {{- \" \- \" \+ tool.description + \"\n\n\" }}\n {%- for param_name, param_fields in tool.parameters.properties|items %}\n {%- if loop.first %}\n {{- \" Args:\n\" }}\n {%- endif %}\n {{- \" \" \+ param_name + \"(\" \+ json_to_python_type(param_fields) + \"): \" \+ param_fields.description|trim }}\n {%- endfor %}\n {%- if tool.return is defined and tool.return.description is defined %}\n {{- \"\n Returns:\n \" \+ tool.return.description }}\n {%- endif %}\n {{- '\"' }}\n {{- ', \"parameters\": ' }}\n {%- if tool.parameters.properties | length == 0 %}\n {{- \"{}\" }}\n {%- else %}\n {{- tool.parameters|tojson }}\n {%- endif %}\n {{- \"}\" }}\n {%- if not loop.last %}\n {{- \"\n\" }}\n {%- endif %}\n{%- endfor %}\n{{- \" </tools>\" }}\n{{- 'Use the following pydantic model json schema for each tool call you will make: {\"properties\": {\"name\": {\"title\": \"Name\", \"type\": \"string\"}, \"arguments\": {\"title\": \"Arguments\", \"type\": \"object\"}}, \"required\": [\"name\", \"arguments\"], \"title\": \"FunctionCall\", \"type\": \"object\"}}\n' }}\n{{- \"For each function call return a json object with function name and arguments within <tool_call></tool_call> XML tags as follows:\n\" }}\n{{- \"<tool_call>\n\" }}\n{{- '{\"name\": <function-name>, \"arguments\": <args-dict>}\n' }}\n{{- '</tool_call><|im_end|>\n' }}\n{%- for message in messages %}\n {%- if message.role == \"user\" or message.role == \"system\" or (message.role == \"assistant\" and message.tool_calls is not defined) %}\n {{- '<|im_start|>' \+ message.role + '\n' \+ message.content + '<|im_end|>' \+ '\n' }}\n {%- elif message.role == \"assistant\" %}\n {{- '<|im_start|>' \+ message.role }}\n {%- for tool_call in message.tool_calls %}\n {{- '\n<tool_call>\n' }} {%- if tool_call.function is defined %}\n {%- set tool_call = tool_call.function %}\n {%- endif %}\n {{- '{' }}\n {{- '\"name\": \"' }}\n {{- tool_call.name }}\n {{- '\"' }}\n {{- ', '}}\n {%- if tool_call.arguments is defined %}\n {{- '\"arguments\": ' }}\n {%- if tool_call.arguments is string %}\n

{{- tool_call.arguments }}\n {%- else %}\n {{- tool_call.arguments|tojson }}\n {%- endif %}\n {%- endif %}\n {{- '}' }}\n {{- '\n</tool_call>' }}\n {%- endfor %}\n {{- '<|im_end|>\n' }}\n {%- elif message.role == \"tool\" %}\n {%- if loop.previtem and loop.previtem.role != \"tool\" %}\n {{- '<|im_start|>tool\n' }}\n {%- endif %}\n {{- '<tool_response>\n' }}\n {{- message.content }}\n {%- if not loop.last %}\n {{- '\n</tool_response>\n' }}\n {%- else %}\n {{- '\n</tool_response>' }}\n {%- endif %}\n {%- if not loop.last and loop.nextitem.role != \"tool\" %}\n {{- '<|im_end|>' }}\n {%- elif loop.last %}\n {{- '<|im_end|>' }}\n {%- endif %}\n {%- endif %}\n{%- endfor %}\n{%- if add_generation_prompt %}\n {{- '<|im_start|>assistant\n' }}\n{%- endif %}\n"~~}~~

2061 | }

2062 |],

2063 | "clean_up_tokenization_spaces": true,

|

---|---

2057 | },

2058 | {

2059 | "name": "tool_use",

2060 | + "template": "{%- macro json_to_python_type(json_spec) %}\n{%- set basic_type_map = {\n \"string\": \"str\",\n \"number\": \"float\",\n \"integer\": \"int\",\n \"boolean\": \"bool\"\n} %}\n\n{%- if basic_type_map[json_spec.type] is defined %}\n {{- basic_type_map[json_spec.type] }}\n{%- elif json_spec.type == \"array\" %}\n {{- \"list[\" \+ json_to_python_type(json_spec|items) + \"]\"}}\n{%- elif json_spec.type == \"object\" %}\n {%- if json_spec.additionalProperties is defined %}\n {{- \"dict[str, \" \+ json_to_python_type(json_spec.additionalProperties) + ']'}}\n {%- else %}\n {{- \"dict\" }}\n {%- endif %}\n{%- elif json_spec.type is iterable %}\n {{- \"Union[\" }}\n {%- for t in json_spec.type %}\n {{- json_to_python_type({\"type\": t}) }}\n {%- if not loop.last %}\n {{- \",\" }} \n {%- endif %}\n {%- endfor %}\n {{- \"]\" }}\n{%- else %}\n {{- \"Any\" }}\n{%- endif %}\n{%- endmacro %}\n\n\n{{- bos_token }}\n{{- '<|im_start|>system\n' }}\n{{- \"You are a function calling AI model. You are provided with

function signatures within <tools></tools> XML tags. You may call one or more functions to assist with the user query. Don't make assumptions about what values to plug into functions. Here are the available tools: <tools> \" }}\n{%- for tool in tools %}\n {%- if tool.function is defined %}\n {%- set tool = tool.function %}\n {%- endif %}\n {{- '{\"type\": \"function\", \"function\": ' }}\n {{- '{\"name\": \"' \+ tool.name + '\", ' }}\n {{- '\"description\": \"' \+ tool.name + '(' }}\n {%- for param_name, param_fields in tool.parameters.properties|items %}\n {{- param_name + \": \" \+ json_to_python_type(param_fields) }}\n {%- if not loop.last %}\n {{- \", \" }}\n {%- endif %}\n {%- endfor %}\n {{- \")\" }}\n {%- if tool.return is defined %}\n {{- \" -> \" \+ json_to_python_type(tool.return) }}\n {%- endif %}\n {{- \" \- \" \+ tool.description + \"\n\n\" }}\n {%- for param_name, param_fields in tool.parameters.properties|items %}\n {%- if loop.first %}\n {{- \" Args:\n\" }}\n {%- endif %}\n {{- \" \" \+ param_name + \"(\" \+ json_to_python_type(param_fields) + \"): \" \+ param_fields.description|trim }}\n {%- endfor %}\n {%- if tool.return is defined and tool.return.description is defined %}\n {{- \"\n Returns:\n \" \+ tool.return.description }}\n {%- endif %}\n {{- '\"' }}\n {{- ', \"parameters\": ' }}\n {%- if tool.parameters.properties | length == 0 %}\n {{- \"{}\" }}\n {%- else %}\n {{- tool.parameters|tojson }}\n {%- endif %}\n {{- \"}\" }}\n {%- if not loop.last %}\n {{- \"\n\" }}\n {%- endif %}\n{%- endfor %}\n{{- \" </tools>\" }}\n{{- 'Use the following pydantic model json schema for each tool call you will make: {\"properties\": {\"name\": {\"title\": \"Name\", \"type\": \"string\"}, \"arguments\": {\"title\": \"Arguments\", \"type\": \"object\"}}, \"required\": [\"name\", \"arguments\"], \"title\": \"FunctionCall\", \"type\": \"object\"}}\n' }}\n{{- \"For each function call return a json object with function name and arguments within <tool_call></tool_call> XML tags as follows:\n\" }}\n{{- \"<tool_call>\n\" }}\n{{- '{\"name\": <function-name>, \"arguments\": <args-dict>}\n' }}\n{{- '</tool_call><|im_end|>\n' }}\n{%- for message in messages %}\n {%- if message.role == \"user\" or message.role == \"system\" or (message.role == \"assistant\" and message.tool_calls is not defined) %}\n {{- '<|im_start|>' \+ message.role + '\n' \+ message.content + '<|im_end|>' \+ '\n' }}\n {%- elif message.role == \"assistant\" %}\n {{- '<|im_start|>' \+ message.role }}\n {%- for tool_call in message.tool_calls %}\n {{- '\n<tool_call>\n' }} {%- if tool_call.function is defined %}\n {%- set tool_call = tool_call.function %}\n {%- endif %}\n {{- '{' }}\n {{- '\"name\": \"' }}\n {{- tool_call.name }}\n {{- '\"' }}\n {{- ', '}}\n {%- if

tool_call.arguments is defined %}\n {{- '\"arguments\": ' }}\n {%- if tool_call.arguments is string %}\n {{- tool_call.arguments }}\n {%- else %}\n {{- tool_call.arguments|tojson }}\n {%- endif %}\n {%- endif %}\n {{- '}' }}\n {{- '\n</tool_call>' }}\n {%- endfor %}\n {{- '<|im_end|>\n' }}\n {%- elif message.role == \"tool\" %}\n {%- if loop.previtem and loop.previtem.role != \"tool\" %}\n {{- '<|im_start|>tool\n' }}\n {%- endif %}\n {{- '<tool_response>\n' }}\n {{- message.content }}\n {%- if not loop.last %}\n {{- '\n</tool_response>\n' }}\n {%- else %}\n {{- '\n</tool_response>' }}\n {%- endif %}\n {%- if not loop.last and loop.nextitem.role != \"tool\" %}\n {{- '<|im_end|>' }}\n {%- elif loop.last %}\n {{- '<|im_end|>' }}\n {%- endif %}\n {%- endif %}\n{%- endfor %}\n{%- if add_generation_prompt %}\n {{- '<|im_start|>assistant\n' }}\n{%- endif %}\n"
2061 | }
2062 |],
2063 | "clean_up_tokenization_spaces": true,