

Upstage releases Solar Open 2, a 250 billion parameter Korean open model

Source copy kept by wheresthe.ai for the story "Upstage releases Solar Open 2, a 250 billion parameter Korean open model" (<https://wheresthe.ai/d/kr-upstage-solar-open-2-2026-07/>).

Source: <https://huggingface.co/upstage/Solar-Open2-250B>

Text saved 18 Oct 2026 (the page as published).

SHA-256 of the saved text: 1e90f2a985fe840d66dea62fcb5984ef433ecabcc136db51f50304a3195f822c

```
","eos_token": "<endoftext|>","pad_token": "<endoftext|>","unk_token": "<unk>"},"chat_template_jinja": "{#-
===== Template Parameters ===== #}\n{%- set add_generation_prompt = add_generation_prompt if
add_generation_prompt is defined else true %}\n{%- set provider_system_prompt = provider_system_prompt if
provider_system_prompt is defined else false %}\n{%- set reasoning_effort = reasoning_effort if reasoning_effort is defined
else \"high\" %}\n{%- set think_render_option = think_render_option if think_render_option is defined else
\"interleaved\" %}\n\n{#- ===== System Block State ===== #}\n{%- set sys_ns =
namespace(is_first_block=true) %}\n\n{#- ===== Find last user message index ===== #}\n{%- set
last_user_idx = namespace(value=-1) %}\n{%- for message in messages %}\n{%- if message.role == 'user' -
%}\n{%- set last_user_idx.value = loop.index0 %}\n{%- endif %}\n{%- endfor %}\n\n{#- ===== System
messages renderers ===== #}\n{%- macro render_system_message(user_system_messages) %}\n{%- if
provider_system_prompt %}\n{%- if not sys_ns.is_first_block %}{%- \"\n\n\" %}{%- endif %}\n{%- set
sys_ns.is_first_block = false %}\n{%- \"### Provider System Prompt\n\nYou are Solar Open2 250B, a large
language model trained by Upstage AI, a Korean startup. Your knowledge cutoff is 2026-02. The current date is \"
+ strftime_now(\"%Y-%m-%d\") + \".\" %}\n{%- endif %}\n{%- if user_system_messages %}\n{%- if not
sys_ns.is_first_block %}{%- \"\n\n\" %}{%- endif %}\n{%- set sys_ns.is_first_block = false %}\n{%- \"### System
Prompt\" %}\n{%- for system_message in user_system_messages %}\n{%- \"\n\n\" %}\n{%- system_message %}\n
{%- endfor %}\n{%- endif %}\n{%- endmacro %}\n\n{%- macro render_tool_instruction(tools, is_last_block) %}\n
{%- if not sys_ns.is_first_block %}{%- \"\n\n\" %}{%- endif %}\n{%- set sys_ns.is_first_block = false %}\n
{%- \"###
Tools\n- You may invoke one or more tools to assist with the user's query.\" %}\n{%- \"\n\n### Available Tools\n\"
%}\n{%- for tool in tools %}\n{%- \"<tool:start>\" %}\n{%- tool.function | tojson %}\n{%- \"<tool:end>\n\n\" %}\n{%-
endfor %}\n{%- \"\n\n### Tool Call Instruction\n\n\" %}\n{%- \"- If using a tool, any reasoning must strictly precede the
call. Do not append any text after the tool call.\n\n\" %}\n{%- \"- If no tool is required, answer directly from your
knowledge without ever mentioning the availability or absence of tools.\n\n\" %}\n{%- \"- Each tool call MUST use this
following format: <tool_call:start>{example-tool-name}<\n\n<tool_arg:start>{example-key-name-1}<tool_arg:value|
>{example-value-1}<tool_arg:end>\n\n<tool_arg:start>{example-key-name-2}<tool_arg:value|>{example-
value-2}<tool_arg:end>\n\n<tool_call:end>\n\n\" %}\n{%- endmacro %}\n\n{#- \n\nInput convention for
`response_format`\n- Per the OpenAI Chat Completions API, `response_format` is an object of one of:\n {\"type\":
\"json_schema\", \"json_schema\": {\"name\": ..., \"schema\": {...}, \"strict\": ...}}\n {\"type\": \"json_object\"}\n- Only
`response_format.json_schema.schema` (the inner JSON Schema) is injected\n into the prompt; the OpenAI
wrapper fields (`name`, `strict`, ...) are dropped.\n- For `json_object`, no schema exists, so we emit a generic
JSON-only instruction.\n- vLLM does NOT pass `response_format` to chat templates by default; it routes it\n to
guided decoding instead. This template/whale opts into prompt injection,\n so callers serving the model directly via
vLLM must opt in via\n `chat_template_kwargs` (or rely on whale's normalization).\n-#}\n{%- macro
render_json_response_format_instruction(response_format) %}\n{%- if not sys_ns.is_first_block %}{%- \"\n\n\" %}\n
{%- endif %}\n{%- set sys_ns.is_first_block = false %}\n{%- \"### Output Format Constraint\" %}\n{%- if
response_format.type == \"json_schema\" %}\n{%- \"\n\n- Your final response should follow the JSON schema:\n
\n```\n\n\" %}\n{%- response_format.json_schema.schema | tojson %}\n{%- \"\n\n```\n\n- Please ensure your answers
adhere to this format and do not contain any unnecessary text.\" %}\n{%- elif response_format.type ==
\"json_object\" %}\n{%- \"\n\n- Your final response must be a valid JSON object.\" %}\n{%- \"\n\n- Do not include any
text outside the JSON object.\" %}\n{%- endif %}\n{%- endmacro %}\n\n{#- \n\nInput convention for `tool_arguments`:
\n- Must be a mapping (dict), NOT a JSON-encoded string.\n- The OpenAI API spec defines
`tool_calls[].function.arguments` as a JSON string,\n but vLLM, HuggingFace transformers tool-use templates
(Qwen, Hermes, Llama, ...),\n and this template all expect the value to be parsed into a dict before rendering.\n-
vLLM normalizes this in `postprocess_messages()` (vllm/entrypoints/chat_utils.py).\n- Callers (whale, training
pipelines that call `tokenizer.apply_chat_template`\n directly, ...) are responsible for parsing the JSON string into a
```

```
def before_rendering(template):\n    #\n\n{% macro render_tool_arguments(tool_arguments) %}\n    {% for\nargs_name, args_value in tool_arguments.items %}\n        {{- \"<|tool_arg:start|>\" + args_name + \"<|tool_arg:value|\n>|\"}}\n    {% set args_value = args_value if args_value is string else (args_value | tojson | safe) %}\n    {{- args_value\n}}\n    {{- \"<|tool_arg:end|>\n\\n\" }}\n    {% endfor %}\n{% endmacro %}\n\n{# ===== Render system message\n===== #}\n{% set ns = namespace(system_messages=[]) -%}\n{% for message in messages -%}\n    {% if\nmessage.role == 'system' -%}\n        {% set ns.system_messages = ns.system_messages + [message.content] -%}\n    {% endif -%}\n{% endfor -%}\n\n{% if ns.system_messages or provider_system_prompt or tools or\nresponse_format -%}\n    {{- \"<|im:start|>system<|im:content|>|\" }}\n    {\nrender_system_message(ns.system_messages) }\n    {% if tools -%}\n        {\nrender_tool_instruction(tools, not\nresponse_format) }\n    {% endif %}\n    {% if response_format -%}\n        {\nrender_json_response_format_instruction(response_format) }\n    {% endif %}\n    {{- \"<|im:end|>\n\\n\" }}\n{% endif -\n%}\n\n{# ===== Render main messages ===== #}\n{% for message in messages -%}\n    {% if\nmessage.role == 'user' -%}\n        {{- \"<|im:start|>user<|im:content|>\" + message.content + \"<|im:end|>\n\\n\" }}\n    {% elif\nmessage.role == 'tool' -%}\n        {% if not message.tool_call_id -%}\n            {{- raise_exception(\"tool message is\nmissing required 'tool_call_id' field\") -}}\n        {% endif -%}\n        {% set prev_is_tool = loop.index0 > 0 and\nmessages[loop.index0 - 1].role == 'tool' -%}\n        {#- Only render at the start of a contiguous tool block; subsequent\ntool messages -#}\n        {#- are already emitted (reordered) within the first one. -#}\n        {% if not prev_is_tool -%}\n            {% set start_idx = loop.index0 -%}\n            {#- Collect contiguous tool messages starting at start_idx -#}\n            {% set tool_ns =\nnamespace(items=[], scanning=true) -%}\n            {% for j in range(start_idx, messages | length) -%}\n                {% if\n                tool_ns.scanning -%}\n                    {% if messages[j].role == 'tool' -%}\n                        {% set tool_ns.items = tool_ns.items + [messages[j]]\n                    -%}\n                    {% else -%}\n                        {% set tool_ns.scanning = false -%}\n                    {% endif -%}\n                {% endif -%}\n            {% endfor -%}\n            {{-\n            \"<|im:start|>tool<|im:content|>|\" }}\n            {% set prev_msg = messages[start_idx - 1] if start_idx > 0 else none -%}\n            {% if prev_msg and prev_msg.role == 'assistant' and prev_msg.tool_calls -%}\n                {#- Render tool responses in the\norder of the preceding assistant's tool_calls ids. -#}\n                {% set sep_ns = namespace(first=true, rendered_ids=[]) -\n            %}\n                {% for tool_call in prev_msg.tool_calls -%}\n                    {% for tool_msg in tool_ns.items -%}\n                        {% if\n                        tool_msg.tool_call_id == tool_call.id -%}\n                            {% if not sep_ns.first -%}\n                                {{- \"\\n\\n\" }}\n                            {% endif -%}\n                            {% set sep_ns.first\n= false -%}\n                            {% set sep_ns.rendered_ids = sep_ns.rendered_ids + [tool_msg.tool_call_id] -%}\n                            {{- \"<\ntool_response:start|>\" + tool_msg.content + \"<|tool_response:end|>|\" }}\n                        {% endif -%}\n                    {% endfor -%}\n                {% #- Append any orphan tool messages (no matching id) in their original order. -#\n            %}\n                {% for tool_msg\nin tool_ns.items -%}\n                    {% if tool_msg.tool_call_id not in sep_ns.rendered_ids -%}\n                        {% if not sep_ns.first -%}\n                            {{- \"\\n\\n\" }}\n                        {% endif -%}\n                        {% set sep_ns.first = false -%}\n                        {{- \"<|tool_response:start|>\" + tool_msg.content + \"<|\ntool_response:end|>|\" }}\n                    {% endif -%}\n                {% endfor -%}\n                {% else -%}\n                    {% for tool_msg in tool_ns.items -%}\n                        {% if not loop.first -%}\n                            {{- \"\\n\\n\" }}\n                        {% endif -%}\n                        {{- \"<|tool_response:start|>|\" + tool_msg.content + \"<|\ntool_response:end|>|\" }}\n                    {% endfor -%}\n                {% endif -%}\n            {{- \"\\n\\n<|im:end|>\n\\n\" }}\n        {% endif -%}\n        {% elif\n        message.role == 'assistant' -%}\n            {{- \"<|im:start|>assistant<|im:content|>|\" }}\n            {#- ===== Resolve reasoning\nfield (reasoning or reasoning_content) ===== #}\n            {% if message.reasoning is defined -%}\n                {% set\n                reasoning = message.reasoning -%}\n            {% elif message.reasoning_content is defined -%}\n                {% set reasoning =\n                message.reasoning_content -%}\n            {% else -%}\n                {% set reasoning = none -%}\n            {% endif -%}\n            {#- =====\nAssistant Thinking ===== #}\n            {% if think_render_option == \"preserved\" -%}\n                {% if reasoning -%}\n                    {{- \"<|\nthink:start|>\" + reasoning + \"<|think:end|>|\" }}\n                {% else -%}\n                    {{- \"<|think:start|><|think:end|>|\" }}\n                {% endif -\n            %}\n            {% elif think_render_option == \"interleaved\" -%}\n                {% if reasoning and loop.index0 > last_user_idx.value -\n            %}\n                    {{- \"<|think:start|>\" + reasoning + \"<|think:end|>|\" }}\n                {% else -%}\n                    {{- \"<|think:start|><|think:end|>|\" }}\n                {% endif -%\n            %}\n            {% endif -%}\n            {#- ===== Assistant Messages ===== #}\n            {% if message.content -%}\n                {{- message.content\n            }}\n            {% endif -%}\n            {#- ===== Assistant Tool calls ===== #}\n            {% if\n            message.tool_calls -%}\n                {% for tool_call in message.tool_calls -%}\n                    {% if not tool_call.id -%}\n                        {{- raise_exception(\"assistant tool_call is missing required 'id' field\") -}}\n                    {% endif -%}\n                    {% if not loop.first -%}\n                        {{- \"\\n\\n\" }}\n                    {% endif -%}\n                    {{- \"<|tool_call:start|>\" + tool_call.function.name + \"\\n\\n\" }}\n                    {% if\n                    tool_call.function.arguments is defined %}\n                        {{- render_tool_arguments(tool_call.function.arguments) }}\n                    {% endif\n                    %}\n                    {{- \"<|tool_call:end|>|\" }}\n                {% endfor -%}\n            {{- \"\\n\\n\" }}\n            {% endif -%}\n            {{- \"<|im:end|>\n\\n\" }}\n        {% endif -\n        %}\n    {% endfor -%}\n\n{% if add_generation_prompt -%}\n    {% if reasoning_effort in [\"medium\", \"high\",\n    \"xhigh\"] -%}\n        {{- \"<|im:start|>assistant<|im:content|><|think:start|>|\" }}\n    {% else -%}\n        {{- \"<|im:start|\n>assistant<|im:content|><|think:start|><|think:end|>|\" }}\n    {% endif -%}\n{% endif -\n%}\n\n{{- \"createdAt\":\"2026-07-22T02:40:42.000Z\",\"discussionsDisabled\":false,\"discussionsSorting\":\"recently-\ncreated\",\"downloads\":7096,\"downloadsAllTime\":48672,\"id\":\"upstage/Solar-Open2-250B\",\"isLikedByUser\":false,\"availableInferenceProviders\":\n[],\"showHuggingChatEntry\":false,\"inference\":\"\",\"lastModified\":\"2026-07-31T21:23:35.000Z\",\"likes\":757,\"pipeline_tag\":
```

```

generation", "library_name": "transformers", "librariesOther": [], "trackDownloads": true, "model-
index": null, "evalResults": [{"dataset": {"id": "TIGER-Lab/MMLU-
Pro", "isBenchmark": true, "task_id": "mmlu_pro", "value": 86.2, "source": {"url": "https://huggingface.co/upstage/Solar-
Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/mmlu-
pro.yaml", "verified": false, "pullRequest": 1, "rank": 12, "label": "Mmlu Pro"}, {"dataset": {"id": "SWE-bench/SWE-
bench_Verified", "isBenchmark": true, "task_id": "swe_bench_%_resolved", "value": 70.4, "source": {"url": "https://
huggingface.co/upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/
swe-bench_verified.yaml", "verified": false, "pullRequest": 1, "rank": 41, "label": "Swe Bench Resolved"}, {"dataset":
{"id": "Idavidrein/gpqa", "isBenchmark": true, "task_id": "diamond", "value": 86.3, "source": {"url": "https://huggingface.co/
upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/
gpqa.yaml", "verified": false, "pullRequest": 1, "rank": 36, "label": "Diamond"}, {"dataset": {"id": "mercor/apex-
agents", "isBenchmark": true, "task_id": "apex-agents", "value": 16.6, "source": {"url": "https://huggingface.co/upstage/
Solar-Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/apex-
agents.yaml", "verified": false, "pullRequest": 1, "rank": 7, "label": "Apex Agents"}, {"dataset": {"id": "MathArena/
aime_2026", "isBenchmark": true, "task_id": "MathArena/aime_2026", "value": 95.7, "source": {"url": "https://
huggingface.co/upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/
aime_2026.yaml", "verified": false, "pullRequest": 1, "rank": 10, "sizeRank":
{"rank": 3, "value": 50000000000, "id": "500B", "label": "< 500B"}, {"label": "MathArena Aime 2026"}, {"dataset":
{"id": "cais/hle", "isBenchmark": false, "task_id": "hle", "value": 28.8, "source": {"url": "https://huggingface.co/upstage/
Solar-Open2-250B", "name": "Model Card", "isExternal": false, "filename": ".eval_results/
hle.yaml", "verified": false, "pullRequest": 1, "rank": 40, "label": "Hle"}}, {"private": false, "repoType": "model", "gated": false, "tag
["transformers", "safetensors", "solar_open2", "text-
generation", "upstage", "solar", "moe", "llm", "vllm", "conversational", "en", "ko", "ja", "arxiv:2607.20062", "license:other", "eva
results", "endpoints_compatible", "region:us"], "tag_objs": [{"id": "text-generation", "label": "Text
Generation", "type": "pipeline_tag", "subType": "nlp"}, {"id": "transformers", "label": "Transformers", "type": "library"},
{"id": "safetensors", "label": "Safetensors", "type": "library"}, {"id": "en", "label": "English", "type": "language"},
{"id": "ko", "label": "Korean", "type": "language"}, {"id": "ja", "label": "Japanese", "type": "language"},
{"id": "solar_open2", "label": "solar_open2", "type": "other", "clickable": true},
{"id": "upstage", "label": "upstage", "type": "other", "clickable": true},
{"id": "solar", "label": "solar", "type": "other", "clickable": true}, {"id": "moe", "label": "Mixture of
Experts", "type": "other", "clickable": true}, {"id": "llm", "label": "llm", "type": "other", "clickable": true},
{"id": "vllm", "label": "vllm", "type": "other", "clickable": true},
{"id": "conversational", "label": "conversational", "type": "other", "clickable": true}, {"id": "eval-results", "label": "Eval
Results", "type": "other", "clickable": true}, {"id": "endpoints_compatible", "label": "Inference
Endpoints", "type": "other", "clickable": true}, {"id": "arxiv:2607.20062", "label": "arxiv:2607.20062", "type": "arxiv", "extra":
{"paperTitle": "Solar Open 2 Technical Report"}}, {"id": "license:other", "label": "other", "type": "license"},
{"type": "region", "label": "🇺🇸 Region: US", "id": "region:us"}], "transformersInfo":
{"auto_model": "AutoModelForCausalLM", "pipeline_tag": "text-generation", "widgetData": [{"text": "Hi, what can you
help me with?"}, {"text": "What is 84 * 3 / 2?"}, {"text": "Tell me an interesting fact about the universe!"},
{"text": "Explain quantum computing in simple terms."}], "safetensors": {"parameters":
{"BF16": 250287794944, "F32": 15360, "total": 250287810304, "sharded": true, "totalFileSize": 500585802154}, "hasBlock:
{"closed": 1, "open": 8, "total": 9}, "query": {}, "inferenceContextData": {"billableEntities": [], "entityName2Providers":
{}}, "hasQuantizations": true, "copyToBucketNamespaces": []}]>

```

```

Reasoning</strong>", "children": [], "isValid": true, "title": "Reasoning"}, {"id": "tool-calling", "label": "<strong>Tool
Calling</strong>", "children": [], "isValid": true, "title": "Tool Calling"}, {"id": "capabilities", "label": "<strong>Capabilities", {"id": "agentic-
use", "label": "<strong>Agentic Use</strong>", "children": [{"id": "claude-code", "label": "<strong>Claude Code</
strong>", "children": [], "isValid": true, "title": "Claude Code"}, {"id": "hermes-agent", "label": "<strong>Hermes Agent</
strong>", "children": [], "isValid": true, "title": "Hermes Agent"}, {"id": "best-
practices", "label": "Best Practices", "children": [], "isValid": true, "title": "Best Practices"},
{"id": "license", "label": "License", "children": [], "isValid": true, "title": "License"}, {"id": "citation", "label": "Citation", "children":
[], "isValid": true, "title": "Citation"}], "isValid": true, "title": "Solar Open 2"}]]>

```

Solar Open 2

Solar Open 2 is Upstage's 250B-A15B open-weight large language model, built for agentic use cases such as office productivity, document-intensive work, and coding. Its Hybrid-Attention Mixture-of-Experts (MoE) architecture with linear attention delivers highly efficient inference even in long-context settings.

Highlights

Agentic Specialist: Purpose-built for agentic workflows — tool calling, multi-step reasoning, and end-to-end task execution. Competitive with the strongest open-weight models on agent benchmarks.

Minimal Inference Cost: A 250B-parameter MoE that activates only 15B per token, built on a hybrid attention stack that interleaves three linear-attention layers with one softmax-attention layer — large-model capacity at small-model inference cost.

1M-Token Context: The linear-attention layers encode token order intrinsically in their recurrent state, so positional encoding is removed entirely (NoPE), lifting the RoPE extrapolation limit. Only 12 of the 48 layers keep a KV cache, holding long-context memory to roughly a quarter of an all-softmax model of the same shape.

Efficiently Trained at Low Cost: Initialized by selective weight transfer from Solar Open 1 (102B) — only the 2.3% of weights that survive the architectural change are carried over, and everything else is randomly initialized — which raises the starting point and accelerates early convergence at 250B scale.

Multilingual: English, Korean, and Japanese.

Model Overview

Field Value

Model Name Solar Open 2 (250B-A15B)

Architecture Hybrid-Attention Mixture-of-Experts (MoE)

Total Parameters 250B (250,287,794,944)

Active Parameters 15B (per token)

Layers 48

Hidden Size 4096

Attention Hybrid — Softmax + Linear Attention, pattern [Softmax, Linear×3] × 12

Position Encoding NoPE (no rotary positional encoding)

Number of Attention Heads (GQA) (Softmax) 64 query / 8 KV, (Linear) 64 query

Number of Experts 321 (320 routed + 1 shared)

Number of Activated Experts 8 routed (top-8) + 1 shared

Vocabulary 196,608

Context Length 1M

Pre-training Tokens ~12 Trillion

Supported Languages English, Korean, Japanese

Training Hardware NVIDIA B200 GPUs

Training GPU Time 2M GPU Hours

License Upstage Solar License (see LICENSE)

Hardware Requirements Minimum: H200 * 4ea / Recommended: H200 * 8ea

Performance

English Benchmarks

Benchmark Solar Open 2 250B-A15B Solar Open 100B 102B-A12B Command A+ 218B-A25B Mistral Medium 3.5 128B dense, high MiMo-V2.5 310B-A15B DeepSeek-V4-Flash 284B-A13B, max

Know. & Reasoning

MMLU-Pro 86.2 80.4 79.0 81.2 84.6 85.9
GPQA-Diamond 86.3 66.2 75.6 77.5 83.0 88.9
HLE (w/o tools) 28.8 11.5 11.4 12.8 24.3 32.3
LiveCodeBench (v6) 92.4 56.5 86.1 84.9 89.1 92.3
ArtifactsBench 55.9 43.4 42.8 49.8 59.3 61.0
HMMT2602 93.9 68.9 73.5 62.9 61.4 94.7
AIME2026 95.7 87.7 96.0 89.0 92.3 97.0

IF / Long

Multi-Challenge 61.0 40.5 45.8 49.8 39.0 62.0
IFBench 80.0 57.7 73.9 69.0 67.1 80.3
AA-LCR 62.3 36.0 46.0 61.0 62.7 63.7

Agent

SWE-Bench Verified 70.4 15.4 14.4 69.6 73.0 73.8
Terminal Bench Hard 28.3 2.3 25.0 33.3 41.7 34.1
APEX-Agents 16.6 2.4 1.6 6.1 13.4 13.2
MCP-Atlas 58.2 34.4 27.2 30.7 63.9 58.2
 τ^3 (banking) 19.6 7.4 5.8 5.8 8.7 22.3
GDPval-AA v2 (ELO) 1128 – 712 929 1145 1187

Korean Benchmarks

Benchmark Solar Open 2 250B-A15B Solar Open 100B 102B-A12B MiMo-V2.5 310B-A15B DeepSeek-V4-Flash 284B-A13B, max Claude Haiku 4.5 closed GPT-5.4 mini closed
KMMMLU-Pro 78.4 64.0 69.1 78.9 67.9 78.1
CLiCk 90.7 78.9 78.4 89.2 53.5 89.6
HAE-RAE v1.1 73.8 73.3 61.7 73.1 38.5 69.4
Ko-AIME'25† 97.7 80.0 88.0 98.0 81.7 90.7
HRM8K 92.2 87.6 90.7 93.4 90.6 91.3
KBank-MMLU† 80.8 65.5 71.0 79.5 68.9 79.0
KBL 75.5 65.5 69.8 72.8 69.9 75.3
KorMedMCQA 93.0 84.4 87.7 94.1 87.0 94.2
Ko-GDPval† 86.8 3.4 81.0 85.0 68.3 59.4

† in-house benchmarks.

Quickstart

The examples below assume 8 GPUs with at least 141 GB of memory each, such as NVIDIA H200 or B200 GPUs. Actual memory requirements depend on the context length and serving settings.

Transformers

Use the `Upstage Transformers` branch with native Solar Open 2 support for local experimentation. For production serving, we recommend `vLLM`.

Install the dependencies:

Install a CUDA-enabled PyTorch build for your platform before running this command. `fla-core` enables the optimized KDA kernels; without it, Transformers uses a substantially slower PyTorch fallback.

```
python -m pip install -U \ "git+https://github.com/upstageAI/transformers.git@v5.14.1-solar-open2" \ "fla-core[cuda]>=0.5.1" \ accelerate einops
```

Run the model:

```
import torch from transformers import AutoModelForCausalLM, AutoTokenizer
```

```
model_id = "upstage/Solar-Open2-250B"
```

```
tokenizer = AutoTokenizer.from_pretrained( model_id, trust_remote_code= False , ) model =  
AutoModelForCausalLM.from_pretrained( model_id, device_map= "auto" , dtype=torch.bfloat16,  
trust_remote_code= False , ) model.eval()
```

```
messages = [ { "role" : "user" , "content" : "What is Upstage?" } , ] prompt = tokenizer.apply_chat_template(  
messages, tokenize= False , add_generation_prompt= True , reasoning_effort= "high" , think_render_option=  
"preserved" , ) input_device = model.get_input_embeddings().weight.device model_inputs = tokenizer(prompt,  
return_tensors= "pt" ).to(input_device)
```

```
generated_ids = model.generate( **model_inputs, max_new_tokens= 32768 , do_sample= True , temperature=  
1.0 , top_p= 1.0 , )
```

```
new_token_ids = generated_ids[ 0 , model_inputs.input_ids.shape[ - 1 ] :].tolist() think_end_id =  
tokenizer.convert_tokens_to_ids( "<|think:end|>" )
```

```
if think_end_id in new_token_ids: # Split immediately after the final <|think:end|> token. answer_start = len  
(new_token_ids) - new_token_ids[::- 1 ].index(think_end_id) else : # No end marker usually means generation  
stopped while the model was reasoning. answer_start = len (new_token_ids)
```

```
reasoning = tokenizer.decode( new_token_ids[:answer_start], skip_special_tokens= True , ).strip() answer =  
tokenizer.decode( new_token_ids[answer_start:], skip_special_tokens= True , ).strip()
```

```
print ( "[reasoning]" , reasoning) print ( "[answer]" , answer)
```

If the answer is empty, generation likely reached `max_new_tokens` before the reasoning block ended. Increase `max_new_tokens` and try again.

Serving with vLLM (Recommended)

Option 1: Docker

The image below is based on vLLM v0.22.0 and CUDA 12.9.

```
docker run -- rm --gpus all --ipc=host \ -p 8000:8000 \ -v " ${HF_HOME:- $HOME /.cache/huggingface} :/  
root/.cache/huggingface" \ upstage/vllm-solar-open2 \ upstage/Solar-Open2-250B \ --served-model-name solar-  
open2-250b \ --tensor-parallel-size 8 \ --enable-expert-parallel \ --moe-backend triton \ --default-chat-template-  
kwargs '{"think_render_option":"preserved"}' \ --reasoning-parser solar_open2 \ --tool-call-parser solar_open2 \ --  
enable-auto-tool-choice \ --logits-processors  
vllm.v1.sample.logits_processor.solar_open2:SolarOpen2TemplateLogitsProcessor
```

Option 2: Install from source

Install the Upstage fork while reusing the matching vLLM v0.22.0 CUDA 12.9 wheel:

```
pip install -U uv
```

```
VLLM_PRECOMPILED_WHEEL_LOCATION= "https://github.com/vllm-project/vllm/releases/download/v0.22.0/  
vllm-0.22.0%2Bcu129-cp38-abi3-manylinux_2_28_x86_64.whl" \ VLLM_USE_PRECOMPILED=1 \ uv pip install --  
reinstall-package vllm --torch-backend=cu129 \ "git+https://github.com/UpstageAI/vllm.git@v0.22.0-solar-open2"
```

Start the server:

```
vllm serve upstage/Solar-Open2-250B \ --served-model-name solar-open2-250b \ --tensor-parallel-size 8 \ --
```

```
enable-expert-parallel \ --moe-backend triton \ --default-chat-template-kwarg '{"think_render_option":"preserved"}' \
--reasoning-parser solar_open2 \ --tool-call-parser solar_open2 \ --enable-auto-tool-choice \ --logits-processors
vllm.v1.sample.logits_processor.solar_open2:SolarOpen2TemplateLogitsProcessor
```

Send a chat completion request:

```
curl http://localhost:8000/v1/chat/completions \ -H "Content-Type: application/json" \ -d '{ "model": "solar-
open2-250b", "messages": [ { "role": "user", "content": "What is Upstage?" } ], "max_tokens": 131584,
"temperature": 1.0, "top_p": 1.0, "reasoning_effort": "high" }'
```

Quantized Versions

Official quantized models by NotaAI are available for deployment on smaller GPU configurations:

Solar-Open2-250B-Nota-INT4-GlobalPruned

Solar-Open2-250B-Nota-NVFP4

Solar-Open2-250B-Nota-INT4

Capabilities

Reasoning

Use `reasoning_effort="high"` for reasoning and `reasoning_effort="none"` for a direct response. The recommended vLLM configuration limits a reasoning block to 131,072 tokens.

Effort Behavior

`none` Direct response

`high` Reasoning, capped at 131,072 tokens

`max_tokens` limits the complete response, including reasoning and the final answer, so leave room beyond the reasoning cap.

from openai import OpenAI

```
client = OpenAI(api_key= "EMPTY", base_url= "http://localhost:8000/v1" )
```

```
response = client.chat.completions.create( model= "solar-open2-250b", messages=[ { "role": "user", "content":
"Prove that the square root of 2 is irrational.", }, ], reasoning_effort= "high", temperature= 1.0, top_p= 1.0,
max_tokens= 131584, )
```

```
# The reasoning trace is returned separately from the final answer. print (response.choices[ 0
].message.reasoning) print (response.choices[ 0 ].message.content)
```

Tool Calling

Tool calls follow the standard OpenAI function-calling interface. Start the server with `--tool-call-parser solar_open2` and `--enable-auto-tool-choice`.

from openai import OpenAI

```
client = OpenAI(api_key= "EMPTY", base_url= "http://localhost:8000/v1" )
```

```
tools = [ { "type": "function", "function": { "name": "get_weather", "description": "Get current weather for a
location", "parameters": { "type": "object", "properties": { "location": { "type": "string" }, }, "required": [ "location"
], }, }, }, ]
```

```
response = client.chat.completions.create( model= "solar-open2-250b", messages=[ { "role": "user", "content":
"What's the weather in Seoul?" }, ], tools=tools, )
```

```
print (response.choices[ 0 ].message.tool_calls)
```

Agentic Use

Both Anthropic's Claude Code and Nous Research's Hermes Agent can run on Solar Open 2 served locally with vLLM (see the vLLM deployment guide). A single vLLM server exposes both interfaces: Claude Code connects

through the Anthropic-compatible `/v1/messages` endpoint and Hermes Agent through the OpenAI-compatible `/v1` endpoint (model id `solar-open2-250b`), each needing only a few environment variables or one provider entry — no setup script required. Tools exposed over the Model Context Protocol (MCP) reach the model through the same tool-calling interface, and both agents support MCP natively.

Claude Code

vLLM exposes an Anthropic-compatible `/v1/messages` endpoint, so Claude Code connects directly — no proxy needed:

```
export ANTHROPIC_BASE_URL=http://localhost:8000 export ANTHROPIC_AUTH_TOKEN=dummy # any non-empty value
export ANTHROPIC_MODEL=solar-open2-250b export ANTHROPIC_SMALL_FAST_MODEL=solar-open2-250b claude
```

The model name must match the server's `--served-model-name` (`solar-open2-250b`).

Prerequisites: the Claude Code CLI installed and a running vLLM server.

Hermes Agent

Register the local vLLM server as a custom OpenAI-compatible provider in `~/.hermes/config.yaml`:

```
model: provider: custom default: solar-open2-250b base_url: http://localhost:8000/v1 api_key: dummy
```

Best Practices

Recommended client-side generation settings (the values a client / API caller should send)

Solar Open 2 is a reasoning-capable model. Use `reasoning_effort="high"` for complex or agentic tasks. The recommended vLLM configuration preserves the reasoning trace.

Parameter Recommended Notes

`reasoning_effort` `high` Recommended for complex reasoning and agentic tasks

`temperature` `1.0`

`top_p` `1.0`

`max_tokens` up to 256K Covers reasoning + output budget

Recommended settings by reasoning mode

Mode `temperature` `top_p` `max_tokens`

`reasoning_effort="none"` `1.0` `1.0` up to 128K

`reasoning_effort="high"` `1.0` `1.0` up to 256K

Set `max_tokens` high enough (up to 256K) — reasoning traces can be long and may otherwise truncate the answer.

The reasoning trace is preserved by default (`think_render_option=preserved`).

Multi-turn: Keep prior reasoning traces in the conversation history. The default `think_render_option=preserved` handles this automatically — do not strip reasoning from previous turns when constructing follow-up requests.

Parsing: the OpenAI-compatible server returns reasoning in a separate `message.reasoning` field with local transformers , split the raw output on the reasoning markers yourself.

License

Solar Open 2 is distributed under the Upstage Solar License .

Key requirements for Derivative AI Models (create / train / fine-tune / distill / improve using Solar Open 2):

Naming: prefix your model name with "Solar" (e.g., `Solar-MyModel-v1`).

Attribution: prominently display "Built with Solar" in related public-facing materials.

Notice: include a copy of the Upstage Solar License with your derivative model.

Citation

@misc{solaropen2-2026, title={Solar Open 2 Technical Report}, author={Upstage AI}, year={2026}, eprint={2607.20062}, archivePrefix={arXiv}, primaryClass={cs.CL}, url={https://arxiv.org/abs/2607.20062} }

Downloads last month 7,096

```
"eos_token": "<endoftext>", "pad_token": "<endoftext>", "unk_token": "<unk>", "chat_template_jinja": "{#-
===== Template Parameters ===== #}
\n{% set add_generation_prompt = add_generation_prompt if
add_generation_prompt is defined else true %}
\n{% set provider_system_prompt = provider_system_prompt if
provider_system_prompt is defined else false %}
\n{% set reasoning_effort = reasoning_effort if reasoning_effort is defined
else \"high\" %}
\n{% set think_render_option = think_render_option if think_render_option is defined else
\"interleaved\" %}
\n{%# ===== System Block State ===== #}
\n{% set sys_ns =
namespace(is_first_block=true) %}
\n{%# ===== Find last user message index ===== #}
\n{% set
last_user_idx = namespace(value=-1) %}
\n{% for message in messages %}
\n{% if message.role == 'user' -
%}
\n{% set last_user_idx.value = loop.index0 %}
\n{% endif %}
\n{% endfor %}
\n{%# ===== System
messages renderers ===== #}
\n{% macro render_system_message(user_system_messages) %}
\n{% if
provider_system_prompt %}
\n{% if not sys_ns.is_first_block %}
\n{{ \"\\n\\n\" }}
\n{% endif %}
\n{% set
sys_ns.is_first_block = false %}
\n{{ \"### Provider System Prompt\\n\\nYou are Solar Open2 250B, a large
language model trained by Upstage AI, a Korean startup. Your knowledge cutoff is 2026-02. The current date is \"
+ strftime_now(\"%Y-%m-%d\") + \".\" }}
\n{% endif %}
\n{% if user_system_messages %}
\n{% if not
sys_ns.is_first_block %}
\n{{ \"\\n\\n\" }}
\n{% endif %}
\n{% set sys_ns.is_first_block = false %}
\n{{ \"### System
Prompt\" }}
\n{% for system_message in user_system_messages %}
\n{{ \"\\n\\n\" }}
\n{% system_message %}
\n{%
endfor %}
\n{% endif %}
\n{% endmacro %}
\n{% macro render_tool_instruction(tools, is_last_block) %}
\n{%
if not sys_ns.is_first_block %}
\n{{ \"\\n\\n\" }}
\n{% endif %}
\n{% set sys_ns.is_first_block = false %}
\n{{ \"###
Tools\\n- You may invoke one or more tools to assist with the user's query.\" }}
\n{{ \"\\n\\n### Available Tools\\n\\n\"
}}
\n{% for tool in tools %}
\n{{ \"<tool:start>\" }}
\n{{ tool.function | tojson }}
\n{{ \"<tool:end>\\n\" }}
\n{%
endfor %}
\n{{ \"\\n### Tool Call Instruction\\n\\n\" }}
\n{{ \"- If using a tool, any reasoning must strictly precede the
call. Do not append any text after the tool call.\\n\" }}
\n{{ \"- If no tool is required, answer directly from your
knowledge without ever mentioning the availability or absence of tools.\\n\" }}
\n{{ \"- Each tool call MUST use this
following format: <tool_call:start>{example-tool-name}<tool_arg:start>{example-key-name-1}<tool_arg:value|
>{example-value-1}<tool_arg:end>\\n<tool_arg:start>{example-key-name-2}<tool_arg:value|>{example-
value-2}<tool_arg:end>\\n<tool_call:end>\\n\" }}
\n{% endmacro %}
\n{%#-\\nInput convention for
`response_format`:\\n- Per the OpenAI Chat Completions API, `response_format` is an object of one of:\\n {\"type\":
\\\"json_schema\\\", \"json_schema\": {\"name\": ..., \"schema\": {...}, \"strict\": ...}}\\n {\"type\": \"json_object\"}\\n- Only
`response_format.json_schema.schema` (the inner JSON Schema) is injected\\n into the prompt; the OpenAI
wrapper fields (`name`, `strict`, ...) are dropped.\\n- For `json_object`, no schema exists, so we emit a generic
JSON-only instruction.\\n- vLLM does NOT pass `response_format` to chat templates by default; it routes it\\n to
guided decoding instead. This template/whale opts into prompt injection,\\n so callers serving the model directly via
vLLM must opt in via\\n `chat_template_kwargs` (or rely on whale's normalization).\\n-#}\\n{% macro
render_json_response_format_instruction(response_format) %}
\n{% if not sys_ns.is_first_block %}
\n{{ \"\\n\\n\" }}
\n{% endif %}
\n{% set sys_ns.is_first_block = false %}
\n{{ \"### Output Format Constraint\" }}
\n{% if
response_format.type == \"json_schema\" %}
\n{{ \"\\n\\n- Your final response should follow the JSON schema:\\n
\\\"\\\"json\\\"\\\"\" }}
\n{{ response_format.json_schema.schema | tojson }}
\n{{ \"\\n\\\"\\\"\\n- Please ensure your answers
adhere to this format and do not contain any unnecessary text.\" }}
\n{% elif response_format.type ==
\"json_object\" %}
\n{{ \"\\n\\n\\n- Your final response must be a valid JSON object.\" }}
\n{{ \"\\n\\n- Do not include any
text outside the JSON object.\" }}
\n{% endif %}
\n{% endmacro %}
\n{%#-\\nInput convention for `tool_arguments`:
\\n- Must be a mapping (dict), NOT a JSON-encoded string.\\n- The OpenAI API spec defines
`tool_calls[].function.arguments` as a JSON string,\\n but vLLM, HuggingFace transformers tool-use templates
(Qwen, Hermes, Llama, ...),\\n and this template all expect the value to be parsed into a dict before rendering.\\n-
vLLM normalizes this in `postprocess_messages()` (vllm/entrypoints/chat_utils.py).\\n- Callers (whale, training
pipelines that call `tokenizer.apply_chat_template`\\n directly, ...) are responsible for parsing the JSON string into a
dict before\\n invoking the template.\\n-#}\\n{% macro render_tool_arguments(tool_arguments) %}
\n{% for
args_name, args_value in tool_arguments.items %}
\n{{ \"<tool_arg:start>\" + args_name + \"<tool_arg:value|
>\" }}
\n{% set args_value = args_value if args_value is string else (args_value | tojson | safe) %}
\n{{ args_value
}}
\n{{ \"<tool_arg:end>\\n\" }}
\n{% endfor %}
\n{% endmacro %}
\n{%# ===== Render system message
```

```

===== #\n%- set ns = namespace(system_messages=[]) -%\n%- for message in messages -%\n %- if
message.role == 'system' -%\n %- set ns.system_messages = ns.system_messages + [message.content] -%\n
%- endif -%\n%- endfor -%\n%- if ns.system_messages or provider_system_prompt or tools or
response_format -%\n {{- \<|im:start|>system<|im:content|>\n }}\n {{-
render_system_message(ns.system_messages) }}\n %- if tools -%\n {{- render_tool_instruction(tools, not
response_format) }}\n %- endif %}\n %- if response_format -%\n {{-
render_json_response_format_instruction(response_format) }}\n %- endif %}\n {{- \<|im:end|>\n\n }}\n%- endif -
%\n\n#- ===== Render main messages ===== #\n%- for message in messages -%\n %- if
message.role == 'user' -%\n {{- \<|im:start|>user<|im:content|>\n + message.content + \<|im:end|>\n\n }}\n %-
elif message.role == 'tool' -%\n %- if not message.tool_call_id -%\n {{- raise_exception("\tool message is
missing required 'tool_call_id' field") -}}\n %- endif -%\n %- set prev_is_tool = loop.index0 > 0 and
messages[loop.index0 - 1].role == 'tool' -%\n #- Only render at the start of a contiguous tool block; subsequent
tool messages -#\n #- are already emitted (reordered) within the first one. -#\n %- if not prev_is_tool -%\n %-
set start_idx = loop.index0 -%\n #- Collect contiguous tool messages starting at start_idx -#\n %- set tool_ns =
namespace(items=[], scanning=true) -%\n %- for j in range(start_idx, messages | length) -%\n %- if
tool_ns.scanning -%\n %- if messages[j].role == 'tool' -%\n %- set tool_ns.items = tool_ns.items + [messages[j]]
-%\n %- else -%\n %- set tool_ns.scanning = false -%\n %- endif -%\n %- endif -%\n %- endfor -%\n\n {{-
\<|im:start|>tool<|im:content|>\n }}\n %- set prev_msg = messages[start_idx - 1] if start_idx > 0 else none -%\n
%- if prev_msg and prev_msg.role == 'assistant' and prev_msg.tool_calls -%\n #- Render tool responses in the
order of the preceding assistant's tool_calls ids. -#\n %- set sep_ns = namespace(first=true, rendered_ids=[]) -
%\n %- for tool_call in prev_msg.tool_calls -%\n %- for tool_msg in tool_ns.items -%\n %- if
tool_msg.tool_call_id == tool_call.id -%\n %- if not sep_ns.first -%\n {{- \<\n\n }}\n %- endif -%\n %- set sep_ns.first
= false -%\n %- set sep_ns.rendered_ids = sep_ns.rendered_ids + [tool_msg.tool_call_id] -%\n {{- \<|
tool_response:start|>\n + tool_msg.content + \<|tool_response:end|>\n }}\n %- endif -%\n %- endfor -%\n %-
endfor -%\n #- Append any orphan tool messages (no matching id) in their original order. -#\n %- for tool_msg
in tool_ns.items -%\n %- if tool_msg.tool_call_id not in sep_ns.rendered_ids -%\n %- if not sep_ns.first -%\n {{-
\<\n\n }}\n %- endif -%\n %- set sep_ns.first = false -%\n {{- \<|tool_response:start|>\n + tool_msg.content + \<|
tool_response:end|>\n }}\n %- endif -%\n %- endfor -%\n %- else -%\n %- for tool_msg in tool_ns.items -%\n
%- if not loop.first -%\n {{- \<\n\n }}\n %- endif -%\n {{- \<|tool_response:start|>\n + tool_msg.content + \<|
tool_response:end|>\n }}\n %- endfor -%\n %- endif -%\n\n {{- \<\n\n<|im:end|>\n\n }}\n %- endif -%\n %- elif
message.role == 'assistant' -%\n {{- \<|im:start|>assistant<|im:content|>\n }}\n #- ===== Resolve reasoning
field (reasoning or reasoning_content) ===== #\n %- if message.reasoning is defined -%\n %- set
reasoning = message.reasoning -%\n %- elif message.reasoning_content is defined -%\n %- set reasoning =
message.reasoning_content -%\n %- else -%\n %- set reasoning = none -%\n %- endif -%\n\n #- =====
Assistant Thinking ===== #\n %- if think_render_option == "preserved" -%\n %- if reasoning -%\n {{- \<|
think:start|>\n + reasoning + \<|think:end|>\n }}\n %- else -%\n {{- \<|think:start|><|think:end|>\n }}\n %- endif -
%\n %- elif think_render_option == "interleaved" -%\n %- if reasoning and loop.index0 > last_user_idx.value -
%\n {{- \<|think:start|>\n + reasoning + \<|think:end|>\n }}\n %- else -%\n {{- \<|think:start|><|think:end|>\n }}\n
%- endif -%\n %- endif -%\n\n #- ===== Assistant Messages ===== #\n %- if message.content -%\n
{{- message.content }}\n %- endif -%\n\n #- ===== Assistant Tool calls ===== #\n %- if
message.tool_calls -%\n %- for tool_call in message.tool_calls -%\n %- if not tool_call.id -%\n {{-
raise_exception("\assistant tool_call is missing required 'id' field") -}}\n %- endif -%\n %- if not loop.first -%\n {{- \<
\n\n }}\n %- endif -%\n {{- \<|tool_call:start|>\n + tool_call.function.name + \<\n\n }}\n %- if
tool_call.function.arguments is defined %}\n {{- render_tool_arguments(tool_call.function.arguments) }}\n %- endif
%\n {{- \<|tool_call:end|>\n }}\n %- endfor -%\n {{- \<\n\n }}\n %- endif -%\n {{- \<|im:end|>\n\n }}\n %- endif -
%\n %- endfor -%\n\n %- if add_generation_prompt -%\n %- if reasoning_effort in ["medium", "high",
"xhigh"] -%\n {{- \<|im:start|>assistant<|im:content|><|think:start|>\n }}\n %- else -%\n {{- \<|im:start|
>assistant<|im:content|><|think:start|><|think:end|>\n }}\n %- endif -%\n %- endif -
%\n\n"},"createdAt":"2026-07-22T02:40:42.000Z","discussionsDisabled":false,"discussionsSorting":"recently-
created","downloads":7096,"downloadsAllTime":48672,"id":"upstage/Solar-
Open2-250B","isLikedByUser":false,"availableInferenceProviders":
[],"showHuggingChatEntry":false,"inference":"","lastModified":"2026-07-31T21:23:35.000Z","likes":757,"pipeline_tag":
"generation","library_name":"transformers","librariesOther":[],"trackDownloads":true,"model-
index":null,"evalResults":{"dataset":{"id":"TIGER-Lab/MMLU-
Pro"},"isBenchmark":true,"task_id":"mmlu_pro"},"value":86.2,"source":{"url":"https://huggingface.co/upstage/Solar-
Open2-250B","name":"Model Card","isExternal":false},"filename":"eval_results/mmlu-

```

```

pro.yaml", "verified": false, "pullRequest": 1, "rank": 12, "label": "Mmlu Pro"}, {"dataset": {"id": "SWE-bench/SWE-
bench_Verified", "isBenchmark": true, "task_id": "swe_bench_%_resolved"}, "value": 70.4, "source": {"url": "https://
huggingface.co/upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false}, "filename": ".eval_results/
swe-bench_verified.yaml", "verified": false, "pullRequest": 1, "rank": 41, "label": "Swe Bench Resolved"}, {"dataset":
{"id": "Idavidrein/gpqa", "isBenchmark": true, "task_id": "diamond"}, "value": 86.3, "source": {"url": "https://huggingface.co/
upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false}, "filename": ".eval_results/
gpqa.yaml", "verified": false, "pullRequest": 1, "rank": 36, "label": "Diamond"}, {"dataset": {"id": "mercor/apex-
agents", "isBenchmark": true, "task_id": "apex-agents"}, "value": 16.6, "source": {"url": "https://huggingface.co/upstage/
Solar-Open2-250B", "name": "Model Card", "isExternal": false}, "filename": ".eval_results/apex-
agents.yaml", "verified": false, "pullRequest": 1, "rank": 7, "label": "Apex Agents"}, {"dataset": {"id": "MathArena/
aime_2026", "isBenchmark": true, "task_id": "MathArena/aime_2026"}, "value": 95.7, "source": {"url": "https://
huggingface.co/upstage/Solar-Open2-250B", "name": "Model Card", "isExternal": false}, "filename": ".eval_results/
aime_2026.yaml", "verified": false, "pullRequest": 1, "rank": 10, "sizeRank":
{"rank": 3, "value": 500000000000, "id": "500B", "label": "< 500B"}, "label": "MathArena Aime 2026"}, {"dataset":
{"id": "cais/hle", "isBenchmark": false, "task_id": "hle"}, "value": 28.8, "source": {"url": "https://huggingface.co/upstage/
Solar-Open2-250B", "name": "Model Card", "isExternal": false}, "filename": ".eval_results/
hle.yaml", "verified": false, "pullRequest": 1, "rank": 40, "label": "Hle"}], "private": false, "repoType": "model", "gated": false, "tag
["transformers", "safetensors", "solar_open2", "text-
generation", "upstage", "solar", "moe", "llm", "vllm", "conversational", "en", "ko", "ja", "arxiv:2607.20062", "license:other", "eva
results", "endpoints_compatible", "region:us"], "tag_objs": [{"id": "text-generation", "label": "Text
Generation", "type": "pipeline_tag", "subType": "nlp"}, {"id": "transformers", "label": "Transformers", "type": "library"},
{"id": "safetensors", "label": "Safetensors", "type": "library"}, {"id": "en", "label": "English", "type": "language"},
{"id": "ko", "label": "Korean", "type": "language"}, {"id": "ja", "label": "Japanese", "type": "language"},
{"id": "solar_open2", "label": "solar_open2", "type": "other", "clickable": true},
{"id": "upstage", "label": "upstage", "type": "other", "clickable": true},
{"id": "solar", "label": "solar", "type": "other", "clickable": true}, {"id": "moe", "label": "Mixture of
Experts", "type": "other", "clickable": true}, {"id": "llm", "label": "llm", "type": "other", "clickable": true},
{"id": "vllm", "label": "vllm", "type": "other", "clickable": true},
{"id": "conversational", "label": "conversational", "type": "other", "clickable": true}, {"id": "eval-results", "label": "Eval
Results", "type": "other", "clickable": true}, {"id": "endpoints_compatible", "label": "Inference
Endpoints", "type": "other", "clickable": true}, {"id": "arxiv:2607.20062", "label": "arxiv:2607.20062", "type": "arxiv", "extra":
{"paperTitle": "Solar Open 2 Technical Report"}], {"id": "license:other", "label": "other", "type": "license"},
{"type": "region", "label": "🇺🇸 Region: US", "id": "region:us"}], "transformersInfo":
{"auto_model": "AutoModelForCausalLM", "pipeline_tag": "text-generation", "widgetData": [{"text": "Hi, what can you
help me with?"}, {"text": "What is 84 * 3 / 2?"}, {"text": "Tell me an interesting fact about the universe!"},
{"text": "Explain quantum computing in simple terms."}], "safetensors": {"parameters":
{"BF16": 250287794944, "F32": 15360, "total": 250287810304, "sharded": true, "totalFileSize": 500585802154}, "hasBlockc
{"billableEntities": [], "entityName2Providers": {}}, "canWrite": false}> Safetensors Model size 250B params

```

Tensor type BF16 · F32 ·

Chat template Files info

```

", "eos_token": "<|endoftext|>", "pad_token": "<|endoftext|>", "unk_token": "<unk>"}, "chat_template_jinja": "{#-
===== Template Parameters ===== #}\n{% set add_generation_prompt = add_generation_prompt if
add_generation_prompt is defined else true %}\n{% set provider_system_prompt = provider_system_prompt if
provider_system_prompt is defined else false %}\n{% set reasoning_effort = reasoning_effort if reasoning_effort is defined
else \"high\" %}\n{% set think_render_option = think_render_option if think_render_option is defined else
\"interleaved\" %}\n\n{#- ===== System Block State ===== #}\n{% set sys_ns =
namespace(is_first_block=true) %}\n\n{#- ===== Find last user message index ===== #}\n{% set
last_user_idx = namespace(value=-1) %}\n\n{% for message in messages %}\n\n{% if message.role == 'user' -
%}\n\n{% set last_user_idx.value = loop.index0 %}\n\n{% endif %}\n\n{% endfor %}\n\n{#- ===== System
messages renderers ===== #}\n\n{% macro render_system_message(user_system_messages) %}\n\n{% if
provider_system_prompt %}\n\n{% if not sys_ns.is_first_block %}{- \"\n\n\n\" %}{%- endif %}\n\n{% set
sys_ns.is_first_block = false %}\n\n{%- \"## Provider System Prompt\n\n\nYou are Solar Open2 250B, a large
language model trained by Upstage AI, a Korean startup. Your knowledge cutoff is 2026-02. The current date is \"
+ strftime_now(\"%Y-%m-%d\") + \".\" %}\n\n{% endif %}\n\n{% if user_system_messages %}\n\n{% if not
sys_ns.is_first_block %}{- \"\n\n\n\" %}{%- endif %}\n\n{% set sys_ns.is_first_block = false %}\n\n{%- \"## System

```

```
Prompt" }}\n {%- for system_message in user_system_messages %}\n {{- "\n\n" }}\n {{- system_message }}\n\n{%- endif %}\n\n{%- endmacro %}\n\n\n{%- macro render_tool_instruction(tools, is_last_block) %}\n\n{%- if not sys_ns.is_first_block %}\n{{- "\n\n" }}\n{%- endif %}\n{%- set sys_ns.is_first_block = false %}\n\n{{- "### Tools\n- You may invoke one or more tools to assist with the user's query." }}\n {{- "\n\n### Available Tools\n" }}\n\n{%- for tool in tools %}\n {{- "<tool:start>" }}\n {{- tool.function | tojson }}\n {{- "<tool:end>\n\n" }}\n\n{%- endfor %}\n\n{{- "\n\n### Tool Call Instruction\n" }}\n {{- "- If using a tool, any reasoning must strictly precede the call. Do not append any text after the tool call.\n" }}\n {{- "- If no tool is required, answer directly from your knowledge without ever mentioning the availability or absence of tools.\n" }}\n {{- "- Each tool call MUST use this following format: <tool_call:start>{example-tool-name}<tool_arg:start>{example-key-name-1}<tool_arg:value|>{example-value-1}<tool_arg:end>\n<tool_arg:start>{example-key-name-2}<tool_arg:value|>{example-value-2}<tool_arg:end>\n\n<tool_call:end>\n\n" }}\n\n{%- endmacro %}\n\n\n#- \nInput convention for `response_format`:\n- Per the OpenAI Chat Completions API, `response_format` is an object of one of:\n { "type": "json_schema", "json_schema": { "name": ..., "schema": {...}, "strict": ...} }\n { "type": "json_object" }\n- Only response_format.json_schema.schema` (the inner JSON Schema) is injected\ninto the prompt; the OpenAI wrapper fields (`name`, `strict`, ...) are dropped.\n- For `json_object`, no schema exists, so we emit a generic JSON-only instruction.\n- vLLM does NOT pass `response_format` to chat templates by default; it routes it\n to guided decoding instead. This template/whale opts into prompt injection,\nso callers serving the model directly via vLLM must opt in via\n`chat_template_kwargs` (or rely on whale's normalization).\n-#\n\n{%- macro render_json_response_format_instruction(response_format) %}\n\n{%- if not sys_ns.is_first_block %}\n{{- "\n\n" }}\n{%- endif %}\n\n{%- set sys_ns.is_first_block = false %}\n\n{{- "### Output Format Constraint" }}\n\n{%- if response_format.type == "json_schema" %}\n {{- "\n\n- Your final response should follow the JSON schema:\n\n```\n\n```\n\n" }}\n {{- response_format.json_schema.schema | tojson }}\n {{- "\n\n```\n\n- Please ensure your answers adhere to this format and do not contain any unnecessary text." }}\n\n{%- elif response_format.type == "json_object" %}\n {{- "\n\n- Your final response must be a valid JSON object." }}\n {{- "\n\n- Do not include any text outside the JSON object." }}\n\n{%- endif %}\n\n{%- endmacro %}\n\n\n#- \nInput convention for `tool_arguments`:\n- Must be a mapping (dict), NOT a JSON-encoded string.\n- The OpenAI API spec defines `tool_calls[].function.arguments` as a JSON string,\nbut vLLM, HuggingFace transformers tool-use templates (Qwen, Hermes, Llama, ...),\nand this template all expect the value to be parsed into a dict before rendering.\n- vLLM normalizes this in `_postprocess_messages()` (vllm/entrypoints/chat_utils.py).\n- Callers (whale, training pipelines that call `tokenizer.apply_chat_template`\n directly, ...) are responsible for parsing the JSON string into a dict\nbefore\ninvoking the template.\n-#\n\n{%- macro render_tool_arguments(tool_arguments) %}\n\n{%- for args_name, args_value in tool_arguments.items %}\n {{- "<tool_arg:start>" + args_name + "<tool_arg:value|>" }}\n {{- set args_value = args_value if args_value is string else (args_value | tojson | safe) }}\n {{- args_value }}\n {{- "<tool_arg:end>\n\n" }}\n\n{%- endfor %}\n\n{%- endmacro %}\n\n\n#- ===== Render system message ===== #\n\n{%- set ns = namespace(system_messages=[]) -%}\n\n{%- for message in messages -%}\n\n{%- if message.role == 'system' -%}\n\n{%- set ns.system_messages = ns.system_messages + [message.content] -%}\n\n{%- endif -%}\n\n{%- endfor -%}\n\n\n{%- if ns.system_messages or provider_system_prompt or tools or response_format -%}\n {{- "<lim:start>system<lim:content|>" }}\n {{- render_system_message(ns.system_messages) }}\n\n{%- if tools -%}\n {{- render_tool_instruction(tools, not response_format) }}\n\n{%- endif %}\n\n{%- if response_format -%}\n {{- render_json_response_format_instruction(response_format) }}\n\n{%- endif %}\n\n\n#- ===== Render main messages ===== #\n\n{%- for message in messages -%}\n\n{%- if message.role == 'user' -%}\n {{- "<lim:start>user<lim:content|>" + message.content + "<lim:end>\n\n" }}\n\n{%- elif message.role == 'tool' -%}\n\n{%- if not message.tool_call_id -%}\n {{- raise_exception("tool message is missing required 'tool_call_id' field") -}}\n\n{%- endif -%}\n\n{%- set prev_is_tool = loop.index0 > 0 and messages[loop.index0 - 1].role == 'tool' -%}\n\n#- Only render at the start of a contiguous tool block; subsequent tool messages\n-#- are already emitted (reordered) within the first one. -#\n\n{%- if not prev_is_tool -%}\n\nset start_idx = loop.index0\n\n#- Collect contiguous tool messages starting at start_idx -#\n\n{%- set tool_ns = namespace(items=[], scanning=true) -%}\n\n{%- for j in range(start_idx, messages | length) -%}\n\n{%- if tool_ns.scanning -%}\n\n{%- if messages[j].role == 'tool' -%}\n\nset tool_ns.items = tool_ns.items + [messages[j]]\n\n{%- else -%}\n\nset tool_ns.scanning = false\n\n{%- endif -%}\n\n{%- endif -%}\n\n{%- endfor -%}\n\n\n{{- "<lim:start>tool<lim:content|>" }}\n\n{%- set prev_msg = messages[start_idx - 1] if start_idx > 0 else none -%}\n\n{%- if prev_msg and prev_msg.role == 'assistant' and prev_msg.tool_calls -%}\n\n#- Render tool responses in the order of the preceding assistant's tool_calls ids. -#\n\n{%- set sep_ns = namespace(first=true, rendered_ids=[]) -%}\n\n{%- for tool_call in prev_msg.tool_calls -%}\n\n{%- for tool_msg in tool_ns.items -%}\n\nif\n\n    tool_msg.tool_call.id == tool_call.id\n\n{%- if not sep_ns.first -%}\n {{- "\n\n" }}\n{%- endif -%}\n\nset sep_ns.first
```

```

= false -%}\n {%- set sep_ns.rendered_ids = sep_ns.rendered_ids + [tool_msg.tool_call_id] -%}\n {{- \<|
tool_response:start|>\" + tool_msg.content + \"\<|tool_response:end|>\" }}\n {%- endif -%}\n {%- endfor -%}\n {%-
endfor -%}\n {%- Append any orphan tool messages (no matching id) in their original order. -#}\n {%- for tool_msg
in tool_ns.items -%}\n {%- if tool_msg.tool_call_id not in sep_ns.rendered_ids -%}\n {%- if not sep_ns.first -%}\n {{-
\"\\n\\n\" }}\n {%- endif -%}\n {%- set sep_ns.first = false -%}\n {{- \<|tool_response:start|>\" + tool_msg.content + \"\<|
tool_response:end|>\" }}\n {%- endif -%}\n {%- endfor -%}\n {%- else -%}\n {%- for tool_msg in tool_ns.items -%}\n
{%- if not loop.first -%}\n {{- \"\\n\\n\" }}\n {%- endif -%}\n {{- \<|tool_response:start|>\" + tool_msg.content + \"\<|
tool_response:end|>\" }}\n {%- endfor -%}\n {%- endif -%}\n {{- \"\\n\\n\<|im:end|>\\n\\n\" }}\n {%- endif -%}\n {%- elif
message.role == 'assistant' -%}\n {{- \<|im:start|>assistant\<|im:content|>\" }}\n {{- #- ===== Resolve reasoning
field (reasoning or reasoning_content) ===== #}\n {%- if message.reasoning is defined -%}\n {%- set
reasoning = message.reasoning -%}\n {%- elif message.reasoning_content is defined -%}\n {%- set reasoning =
message.reasoning_content -%}\n {%- else -%}\n {%- set reasoning = none -%}\n {%- endif -%}\n {{- #- =====
Assistant Thinking ===== #}\n {%- if think_render_option == 'preserved' -%}\n {%- if reasoning -%}\n {{- \<|
think:start|>\" + reasoning + \"\<|think:end|>\" }}\n {%- else -%}\n {{- \<|think:start|>\<|think:end|>\" }}\n {%- endif -
%}\n {%- elif think_render_option == 'interleaved' -%}\n {%- if reasoning and loop.index0 > last_user_idx.value -
%}\n {{- \<|think:start|>\" + reasoning + \"\<|think:end|>\" }}\n {%- else -%}\n {{- \<|think:start|>\<|think:end|>\" }}\n
{%- endif -%}\n {{- #- ===== Assistant Messages ===== #}\n {%- if message.content -%}\n
{{- message.content }}\n {%- endif -%}\n {{- #- ===== Assistant Tool calls ===== #}\n {%- if
message.tool_calls -%}\n {%- for tool_call in message.tool_calls -%}\n {%- if not tool_call.id -%}\n {{-
raise_exception(\"assistant tool_call is missing required 'id' field\") -%}\n {%- endif -%}\n {%- if not loop.first -%}\n {{-
\\n\\n\" }}\n {%- endif -%}\n {{- \<|tool_call:start|>\" + tool_call.function.name + \"\\n\\n\" }}\n {%- if
tool_call.function.arguments is defined %}\n {{- render_tool_arguments(tool_call.function.arguments) }}\n {%- endif
%}\n {{- \<|tool_call:end|>\" }}\n {%- endfor -%}\n {{- \"\\n\\n\" }}\n {%- endif -%}\n {{- \<|im:end|>\\n\\n\" }}\n {%- endif -
%}\n {{- endfor -%}\n {{- if add_generation_prompt -%}\n {%- if reasoning_effort in [\"medium\", \"high\",
\\xhigh\"] -%}\n {{- \<|im:start|>assistant\<|im:content|>\<|think:start|>\" }}\n {%- else -%}\n {{- \<|im:start|
>assistant\<|im:content|>\<|think:start|>\<|think:end|>\" }}\n {%- endif -%}\n {%- endif -
%}\n {{- \"createdAt\":\"2026-07-22T02:40:42.000Z\",\"discussionsDisabled\":false,\"discussionsSorting\":\"recently-
created\",\"downloads\":7096,\"downloadsAllTime\":48672,\"id\":\"upstage/Solar-
Open2-250B\",\"isLikedByUser\":false,\"availableInferenceProviders\":
[],\"showHuggingChatEntry\":false,\"inference\":\"\",\"lastModified\":\"2026-07-31T21:23:35.000Z\",\"likes\":757,\"pipeline_tag\":
generation\",\"library_name\":\"transformers\",\"librariesOther\":[],\"trackDownloads\":true,\"model-
index\":null,\"evalResults\":{\"dataset\":{\"id\":\"TIGER-Lab/MMLU-
Pro\",\"isBenchmark\":true,\"task_id\":\"mmlu_pro\",\"value\":86.2,\"source\":{\"url\":\"https://huggingface.co/upstage/Solar-
Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/mmlu-
pro.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":12,\"label\":\"Mmlu Pro\"},{\"dataset\":{\"id\":\"SWE-bench/SWE-
bench_Verified\",\"isBenchmark\":true,\"task_id\":\"swe_bench_%_resolved\"},\"value\":70.4,\"source\":{\"url\":\"https://
huggingface.co/upstage/Solar-Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/
swe_bench_verified.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":41,\"label\":\"Swe Bench Resolved\"},{\"dataset\":
{id\":\"Idavidrein/gpqa\",\"isBenchmark\":true,\"task_id\":\"diamond\"},\"value\":86.3,\"source\":{\"url\":\"https://huggingface.co/
upstage/Solar-Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/
gpqa.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":36,\"label\":\"Diamond\"},{\"dataset\":{\"id\":\"mercor/apex-
agents\",\"isBenchmark\":true,\"task_id\":\"apex-agents\"},\"value\":16.6,\"source\":{\"url\":\"https://huggingface.co/upstage/
Solar-Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/apex-
agents.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":7,\"label\":\"Apex Agents\"},{\"dataset\":{\"id\":\"MathArena/
aime_2026\",\"isBenchmark\":true,\"task_id\":\"MathArena/aime_2026\"},\"value\":95.7,\"source\":{\"url\":\"https://
huggingface.co/upstage/Solar-Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/
aime_2026.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":10,\"sizeRank\":
{\"rank\":3,\"value\":50000000000,\"id\":\"500B\",\"label\":\"< 500B\"},\"label\":\"MathArena Aime 2026\"},{\"dataset\":
{id\":\"cais/hle\",\"isBenchmark\":false,\"task_id\":\"hle\"},\"value\":28.8,\"source\":{\"url\":\"https://huggingface.co/upstage/
Solar-Open2-250B\",\"name\":\"Model Card\",\"isExternal\":false,\"filename\":\".eval_results/
hle.yaml\",\"verified\":false,\"pullRequest\":1,\"rank\":40,\"label\":\"Hle\"}],\"private\":false,\"repoType\":\"model\",\"gated\":false,\"tag
[\"transformers\",\"safetensors\",\"solar_open2\",\"text-
generation\",\"upstage\",\"solar\",\"moe\",\"llm\",\"vllm\",\"conversational\",\"en\",\"ko\",\"ja\",\"arxiv:2607.20062\",\"license:other\",\"eva
results\",\"endpoints_compatible\",\"region:us\"],\"tag_objs\":{\"id\":\"text-generation\",\"label\":\"Text
Generation\",\"type\":\"pipeline_tag\",\"subType\":\"nlp\"},{\"id\":\"transformers\",\"label\":\"Transformers\",\"type\":\"library\"},
{id\":\"safetensors\",\"label\":\"Safetensors\",\"type\":\"library\"},{\"id\":\"en\",\"label\":\"English\",\"type\":\"language\"},

```

```
{
  "id": "ko", "label": "Korean", "type": "language",
  "id": "ja", "label": "Japanese", "type": "language",
  "id": "solar_open2", "label": "solar_open2", "type": "other", "clickable": true,
  "id": "upstage", "label": "upstage", "type": "other", "clickable": true,
  "id": "solar", "label": "solar", "type": "other", "clickable": true,
  "id": "moe", "label": "Mixture of Experts", "type": "other", "clickable": true,
  "id": "llm", "label": "llm", "type": "other", "clickable": true,
  "id": "vllm", "label": "vllm", "type": "other", "clickable": true,
  "id": "conversational", "label": "conversational", "type": "other", "clickable": true,
  "id": "eval-results", "label": "Eval Results", "type": "other", "clickable": true,
  "id": "endpoints_compatible", "label": "Inference Endpoints", "type": "other", "clickable": true,
  "id": "arxiv:2607.20062", "label": "arxiv:2607.20062", "type": "arxiv", "extra": {
    "paperTitle": "Solar Open 2 Technical Report",
    "id": "license:other", "label": "other", "type": "license",
    "type": "region", "label": "🇺🇸 Region: US", "id": "region:us"
  }, "transformersInfo": {
    "auto_model": "AutoModelForCausalLM",
    "pipeline_tag": "text-generation",
    "widgetData": {
      "text": "Hi, what can you help me with?",
      "text": "What is 84 * 3 / 2?",
      "text": "Tell me an interesting fact about the universe!",
      "text": "Explain quantum computing in simple terms."
    }, "safetensors": {
      "parameters": {
        "BF16": 250287794944, "F32": 15360, "total": 250287810304, "sharded": true, "totalFileSize": 500585802154
      }, "hasBlock": {}, "inferenceContextData": {
        "billableEntities": [], "entityName2Providers": {}
      }, "linkToInferenceDiscussion": {
        "url": "/spaces/huggingface/InferenceSupport/discussions/11210", "numReactions": 13, "isLoggedIn": false, "canWrite": false, "examples": []
      }, "widgetType": "conversational", "isMaximized": false, "isMac": true
    }
  }
}> Inference Providers NEW
```

Text Generation

This model isn't deployed by any Inference Provider. 🙏 13 Ask for provider support

Model tree for upstage/Solar-Open2-250B

Finetunes

3 models

Quantizations

15 models

Space using upstage/Solar-Open2-250B 1

Paper for upstage/Solar-Open2-250B Paper • 2607.20062 • Published Jul 24 • 6

```
"eos_token": "<endoftext>", "pad_token": "<endoftext>", "unk_token": "<unk>", "chat_template_jinja": "{#-
===== Template Parameters ===== #}\n%- set add_generation_prompt = add_generation_prompt if
add_generation_prompt is defined else true %}\n%- set provider_system_prompt = provider_system_prompt if
provider_system_prompt is defined else false %}\n%- set reasoning_effort = reasoning_effort if reasoning_effort is defined
else \"high\" %}\n%- set think_render_option = think_render_option if think_render_option is defined else
\"interleaved\" %}\n{%- ===== System Block State ===== #}\n%- set sys_ns =
namespace(is_first_block=true) %}\n{%- ===== Find last user message index ===== #}\n%- set
last_user_idx = namespace(value=-1) %}\n%- for message in messages %}\n  {%- if message.role == 'user' -
%}\n  {%- set last_user_idx.value = loop.index0 %}\n  {%- endif %}\n%- endfor %}\n{%- ===== System
messages renderers ===== #}\n%- macro render_system_message(user_system_messages) %}\n  {%- if
provider_system_prompt %}\n  {%- if not sys_ns.is_first_block %}\n    {%- \"\n\n\n\" %}\n  {%- endif %}\n  {%- set
sys_ns.is_first_block = false %}\n  {%- \"## Provider System Prompt\n\nYou are Solar Open2 250B, a large
language model trained by Upstage AI, a Korean startup. Your knowledge cutoff is 2026-02. The current date is \"
+ strftime_now(\"%Y-%m-%d\") + \".\" %}\n  {%- endif %}\n  {%- if user_system_messages %}\n  {%- if not
sys_ns.is_first_block %}\n    {%- \"\n\n\n\" %}\n  {%- endif %}\n  {%- set sys_ns.is_first_block = false %}\n  {%- \"## System
Prompt\" %}\n  {%- for system_message in user_system_messages %}\n    {%- \"\n\n\n\" %}\n  {%- system_message %}\n
{%- endfor %}\n  {%- endif %}\n%- endmacro %}\n%- macro render_tool_instruction(tools, is_last_block) %}\n
{%- if not sys_ns.is_first_block %}\n  {%- \"\n\n\n\" %}\n  {%- endif %}\n  {%- set sys_ns.is_first_block = false %}\n  {%- \"##
Tools\n- You may invoke one or more tools to assist with the user's query.\" %}\n  {%- \"\n\n\n### Available Tools\n\"
%}\n  {%- for tool in tools %}\n    {%- \"<tool:start>\" %}\n    {%- tool.function | tojson %}\n    {%- \"<tool:end>\n\n\" %}\n  {%-
endfor %}\n  {%- \"\n\n\n### Tool Call Instruction\n\n\" %}\n  {%- \"- If using a tool, any reasoning must strictly precede the
call. Do not append any text after the tool call.\n\n\" %}\n  {%- \"- If no tool is required, answer directly from your
knowledge without ever mentioning the availability or absence of tools.\n\n\" %}\n  {%- \"- Each tool call MUST use this
```

```

Following format: <|tool_call:start|>{example-tool-name-1}<|tool_arg:value|>{example-value-1}<|tool_arg:end|>\n<|tool_arg:start|>{example-key-name-2}<|tool_arg:value|>{example-value-2}<|tool_arg:end|>\n<|tool_call:end|>\n"}\n{% endmacro %}\n\n#-Input convention for `response_format`: \n- Per the OpenAI Chat Completions API, `response_format` is an object of one of:\n {"type": "json_schema", "json_schema": {"name": "...", "schema": {...}, "strict": ...}}\n {"type": "json_object"}\n- Only `response_format.json_schema.schema` (the inner JSON Schema) is injected\n into the prompt; the OpenAI wrapper fields (`name`, `strict`, ...) are dropped.\n- For `json_object`, no schema exists, so we emit a generic JSON-only instruction.\n- vLLM does NOT pass `response_format` to chat templates by default; it routes it\n to guided decoding instead. This template/whale opts into prompt injection,\n so callers serving the model directly via vLLM must opt in via\n `chat_template_kwargs` (or rely on whale's normalization).\n-#\n{% macro render_json_response_format_instruction(response_format) %}\n{% if not sys_ns.is_first_block %}{- "\n\n"}\n{% endif %}\n{% set sys_ns.is_first_block = false %}\n{{- "## Output Format Constraint" }}\n{% if response_format.type == "json_schema" %}\n{{- "\n\n- Your final response should follow the JSON schema:\n\n```\n\n```\n\n"}\n{{- response_format.json_schema.schema | tojson }}\n{{- "\n\n```\n\n- Please ensure your answers adhere to this format and do not contain any unnecessary text." }}\n{% elif response_format.type == "json_object" %}\n{{- "\n\n- Your final response must be a valid JSON object." }}\n{{- "\n\n- Do not include any text outside the JSON object." }}\n{% endif %}\n{% endmacro %}\n\n#-Input convention for `tool_arguments`: \n- Must be a mapping (dict), NOT a JSON-encoded string.\n- The OpenAI API spec defines `tool_calls[].function.arguments` as a JSON string,\n but vLLM, HuggingFace transformers tool-use templates (Qwen, Hermes, Llama, ...),\n and this template all expect the value to be parsed into a dict before rendering.\n- vLLM normalizes this in `_postprocess_messages()` (vllm/entrypoints/chat_utils.py).\n- Callers (whale, training pipelines that call `tokenizer.apply_chat_template`\n directly, ...) are responsible for parsing the JSON string into a dict before\n invoking the template.\n-#\n{% macro render_tool_arguments(tool_arguments) %}\n{% for args_name, args_value in tool_arguments.items %}\n{{- "<|tool_arg:start|>"+ args_name + "<|tool_arg:value|>"+ args_value + "<|tool_arg:end|>\n\n"}}\n{% set args_value = args_value if args_value is string else (args_value | tojson | safe) %}\n{{- args_value }}\n{{- "<|tool_arg:end|>\n\n"}}\n{% endfor %}\n{% endmacro %}\n\n#- ===== Render system message =====\n{% set ns = namespace(system_messages=[]) -%}\n{% for message in messages -%}\n{% if message.role == 'system' -%}\n{% set ns.system_messages = ns.system_messages + [message.content] -%}\n{% endif -%}\n{% endfor -%}\n{% if ns.system_messages or provider_system_prompt or tools or response_format -%}\n{{- "<|im:start|>system<|im:content|>"+ "\n\n" }}\n{% if tools -%}\n{{- render_tool_instruction(tools, not response_format) }}\n{% endif %}\n{{- render_json_response_format_instruction(response_format) }}\n{% endif %}\n{{- "<|im:end|>\n\n"}}\n\n#- ===== Render main messages =====\n{% for message in messages -%}\n{% if message.role == 'user' -%}\n{{- "<|im:start|>user<|im:content|>"+ message.content + "<|im:end|>\n\n"}}\n{% elif message.role == 'tool' -%}\n{% if not message.tool_call_id -%}\n{{- raise_exception("tool message is missing required 'tool_call_id' field") -%}}\n{% endif -%}\n{% set prev_is_tool = loop.index0 > 0 and messages[loop.index0 - 1].role == 'tool' -%}\n{% #- Only render at the start of a contiguous tool block; subsequent tool messages -#}\n{% #- are already emitted (reordered) within the first one. -#}\n{% if not prev_is_tool -%}\n{% set start_idx = loop.index0 -%}\n{% #- Collect contiguous tool messages starting at start_idx -#}\n{% set tool_ns = namespace(items=[], scanning=true) -%}\n{% for j in range(start_idx, messages | length) -%}\n{% if messages[j].role == 'tool' -%}\n{% set tool_ns.items = tool_ns.items + [messages[j].content] -%}\n{% else -%}\n{% set tool_ns.scanning = false -%}\n{% endif -%}\n{% endfor -%}\n{{- "<|im:start|>tool<|im:content|>"+ "\n\n" }}\n{% set prev_msg = messages[start_idx - 1] if start_idx > 0 else none -%}\n{% if prev_msg and prev_msg.role == 'assistant' and prev_msg.tool_calls -%}\n{% #- Render tool responses in the order of the preceding assistant's tool_calls ids. -#}\n{% set sep_ns = namespace(first=true, rendered_ids=[]) -%}\n{% for tool_call in prev_msg.tool_calls -%}\n{% for tool_msg in tool_ns.items -%}\n{% if tool_msg.tool_call_id == tool_call.id -%}\n{% if not sep_ns.first -%}\n{{- "\n\n" }}\n{% endif -%}\n{% set sep_ns.first = false -%}\n{{- "<|tool_response:start|>"+ tool_msg.content + "<|tool_response:end|>\n\n" }}\n{% endif -%}\n{% endfor -%}\n{% #- Append any orphan tool messages (no matching id) in their original order. -#}\n{% for tool_msg in tool_ns.items -%}\n{% if tool_msg.tool_call_id not in sep_ns.rendered_ids -%}\n{% if not sep_ns.first -%}\n{{- "\n\n" }}\n{% endif -%}\n{% set sep_ns.first = false -%}\n{{- "<|tool_response:start|>"+ tool_msg.content + "<|tool_response:end|>\n\n" }}\n{% endif -%}\n{% endfor -%}\n{{- "\n\n<|im:end|>\n\n" }}\n{% endif -%}\n{% elif

```

```

message.role == 'assistant' -%}\n {{- \<|im:start|>assistant<|im:content|>\n }}\n\n {{- ===== Resolve reasoning
field (reasoning or reasoning_content) ===== #}\n {{- if message.reasoning is defined -%}\n {{- set
reasoning = message.reasoning -%}\n {{- elif message.reasoning_content is defined -%}\n {{- set reasoning =
message.reasoning_content -%}\n {{- else -%}\n {{- set reasoning = none -%}\n {{- endif -%}\n\n {{- =====
Assistant Thinking ===== #}\n {{- if think_render_option == \"preserved\" -%}\n {{- if reasoning -%}\n {{- \<|
think:start|>\n\" + reasoning + \<|think:end|>\n\" }}\n {{- else -%}\n {{- \<|think:start|><|think:end|>\n\" }}\n {{- endif -
%}\n {{- elif think_render_option == \"interleaved\" -%}\n {{- if reasoning and loop.index0 > last_user_idx.value -
%}\n {{- \<|think:start|>\n\" + reasoning + \<|think:end|>\n\" }}\n {{- else -%}\n {{- \<|think:start|><|think:end|>\n\" }}\n
{{- endif -%}\n {{- endif -%}\n\n {{- ===== Assistant Messages ===== #}\n {{- if message.content -%}\n
{{- message.content }}\n {{- endif -%}\n\n {{- ===== Assistant Tool calls ===== #}\n {{- if
message.tool_calls -%}\n {{- for tool_call in message.tool_calls -%}\n {{- if not tool_call.id -%}\n {{-
raise_exception(\"assistant tool_call is missing required 'id' field\") -%}\n {{- endif -%}\n {{- if not loop.first -%}\n {{- \<|
\\n\\n\" }}\n {{- endif -%}\n {{- \<|tool_call:start|>\n\" + tool_call.function.name + \<|\\n\\n\" }}\n {{- if
tool_call.function.arguments is defined %}\n {{- render_tool_arguments(tool_call.function.arguments) }}\n {{- endif
%}\n {{- \<|tool_call:end|>\n\" }}\n {{- endfor -%}\n {{- \<|\\n\\n\" }}\n {{- endif -%}\n {{- \<|im:end|>\\n\\n\" }}\n {{- endif -
%}\n {{- endfor -%}\n\n {{- if add_generation_prompt -%}\n {{- if reasoning_effort in [\"medium\", \"high\",
\\xhigh\"] -%}\n {{- \<|im:start|>assistant<|im:content|><|think:start|>\n\" }}\n {{- else -%}\n {{- \<|im:start|
>assistant<|im:content|><|think:start|><|think:end|>\n\" }}\n {{- endif -%}\n {{- endif -
%}\n\n\", \"createdAt\": \"2026-07-22T02:40:42.000Z\", \"discussionsDisabled\": false, \"discussionsSorting\": \"recently-
created\", \"downloads\": 7096, \"downloadsAllTime\": 48672, \"id\": \"upstage/Solar-
Open2-250B\", \"isLikedByUser\": false, \"availableInferenceProviders\":
[], \"showHuggingChatEntry\": false, \"inference\": \"\", \"lastModified\": \"2026-07-31T21:23:35.000Z\", \"likes\": 757, \"pipeline_tag\":
generation\", \"library_name\": \"transformers\", \"librariesOther\": [], \"trackDownloads\": true, \"model-
index\": null, \"evalResults\": [{\"dataset\": {\"id\": \"TIGER-Lab/MMLU-
Pro\", \"isBenchmark\": true, \"task_id\": \"mmlu_pro\", \"value\": 86.2, \"source\": {\"url\": \"https://huggingface.co/upstage/Solar-
Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/mmlu-
pro.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 12, \"label\": \"Mmlu Pro\"}, {\"dataset\": {\"id\": \"SWE-bench/SWE-
bench_Verified\", \"isBenchmark\": true, \"task_id\": \"swe_bench_%_resolved\", \"value\": 70.4, \"source\": {\"url\": \"https://
huggingface.co/upstage/Solar-Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/
swe-bench_verified.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 41, \"label\": \"Swe Bench Resolved\"}, {\"dataset\":
{\"id\": \"Idavidrein/gpqa\", \"isBenchmark\": true, \"task_id\": \"diamond\", \"value\": 86.3, \"source\": {\"url\": \"https://huggingface.co/
upstage/Solar-Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/
gpqa.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 36, \"label\": \"Diamond\"}, {\"dataset\": {\"id\": \"mercor/apex-
agents\", \"isBenchmark\": true, \"task_id\": \"apex-agents\", \"value\": 16.6, \"source\": {\"url\": \"https://huggingface.co/upstage/
Solar-Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/apex-
agents.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 7, \"label\": \"Apex Agents\"}, {\"dataset\": {\"id\": \"MathArena/
aime_2026\", \"isBenchmark\": true, \"task_id\": \"MathArena/aime_2026\", \"value\": 95.7, \"source\": {\"url\": \"https://
huggingface.co/upstage/Solar-Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/
aime_2026.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 10, \"sizeRank\":
{\"rank\": 3, \"value\": 500000000000, \"id\": \"500B\", \"label\": \"< 500B\", \"label\": \"MathArena Aime 2026\"}, {\"dataset\":
{\"id\": \"cais/hle\", \"isBenchmark\": false, \"task_id\": \"hle\", \"value\": 28.8, \"source\": {\"url\": \"https://huggingface.co/upstage/
Solar-Open2-250B\", \"name\": \"Model Card\", \"isExternal\": false, \"filename\": \".eval_results/
hle.yaml\", \"verified\": false, \"pullRequest\": 1, \"rank\": 40, \"label\": \"Hle\"}}, \"private\": false, \"repoType\": \"model\", \"gated\": false, \"tag
[\"transformers\", \"safetensors\", \"solar_open2\", \"text-
generation\", \"upstage\", \"solar\", \"moe\", \"llm\", \"vllm\", \"conversational\", \"en\", \"ko\", \"ja\", \"arxiv:2607.20062\", \"license:other\", \"eva
results\", \"endpoints_compatible\", \"region:us\"], \"tag_objs\": [{\"id\": \"text-generation\", \"label\": \"Text
Generation\", \"type\": \"pipeline_tag\", \"subType\": \"nlp\"}, {\"id\": \"transformers\", \"label\": \"Transformers\", \"type\": \"library\"},
{\"id\": \"safetensors\", \"label\": \"Safetensors\", \"type\": \"library\"}, {\"id\": \"en\", \"label\": \"English\", \"type\": \"language\"},
{\"id\": \"ko\", \"label\": \"Korean\", \"type\": \"language\"}, {\"id\": \"ja\", \"label\": \"Japanese\", \"type\": \"language\"},
{\"id\": \"solar_open2\", \"label\": \"solar_open2\", \"type\": \"other\", \"clickable\": true},
{\"id\": \"upstage\", \"label\": \"upstage\", \"type\": \"other\", \"clickable\": true},
{\"id\": \"solar\", \"label\": \"solar\", \"type\": \"other\", \"clickable\": true}, {\"id\": \"moe\", \"label\": \"Mixture of
Experts\", \"type\": \"other\", \"clickable\": true}, {\"id\": \"llm\", \"label\": \"llm\", \"type\": \"other\", \"clickable\": true},
{\"id\": \"vllm\", \"label\": \"vllm\", \"type\": \"other\", \"clickable\": true},
{\"id\": \"conversational\", \"label\": \"conversational\", \"type\": \"other\", \"clickable\": true}, {\"id\": \"eval-results\", \"label\": \"Eval
Results\", \"type\": \"other\", \"clickable\": true}, {\"id\": \"endpoints_compatible\", \"label\": \"Inference

```

Endpoints", "type": "other", "clickable": true}, {"id": "arxiv:2607.20062", "label": "arxiv:2607.20062", "type": "arxiv", "extra": {"paperTitle": "Solar Open 2 Technical Report"}}, {"id": "license:other", "label": "other", "type": "license"}, {"type": "region", "label": "🇺🇸 Region: US", "id": "region:us"}], "transformersInfo": {"auto_model": "AutoModelForCausalLM", "pipeline_tag": "text-generation"}, "widgetData": [{"text": "Hi, what can you help me with?"}, {"text": "What is $84 * 3 / 2$?"}, {"text": "Tell me an interesting fact about the universe!"}, {"text": "Explain quantum computing in simple terms."}], "safetensors": {"parameters": {"BF16": 250287794944, "F32": 15360}, "total": 250287810304, "sharded": true, "totalFileSize": 500585802154}, "hasBlock": true}, {"id": "tiger-lab", "label": "Evaluation results TIGER-Lab/MMLU-Pro · Mmlu Pro View evaluation results leaderboard 86.2", "type": "leaderboard"}], [{"id": "swe-bench", "label": "SWE-bench/SWE-bench_Verified · Swe Bench Resolved View evaluation results leaderboard 70.4", "type": "leaderboard"}], [{"id": "diamond", "label": "ldavidrein/gpqa · Diamond View evaluation results leaderboard 86.3", "type": "leaderboard"}], [{"id": "apex-agents", "label": "mercor/apex-agents · Apex Agents View evaluation results leaderboard 16.6", "type": "leaderboard"}], [{"id": "math-arena", "label": "MathArena/aime_2026 · MathArena Aime 2026 View evaluation results leaderboard 95.7", "type": "leaderboard"}], [{"id": "hle", "label": "cais/hle · Hle View evaluation results 28.8", "type": "leaderboard"}]