

LLM Booster SDK Developer Edition

Institutional evaluation, distribution, and technical guide

License scope: Internal institutional evaluation, testing, and development only. Production/commercial use and distribution to third parties are not licensed by this edition. Features are not technically disabled. See LICENSE-DEVELOPER.md.

1. Purpose and scope

This guide helps institutions evaluate the Developer Edition in a controlled technical environment. The Python SDK prepares agent context, performs keyword-based agent selection and text cleanup, and offers experimental transport coding. Python 3.10+ is required; the SDK has no runtime dependencies.

This edition is not a production or commercial enterprise license and provides no SLA, enterprise support, security certification, update commitment, or compliance approval. Users must comply with the access conditions in the license.

2. Download and install the ZIP

The private distribution ZIP contains the Python wheel, license, README, example context, and both Turkish and English SDK and institution guides. Download it from the distribution location. Python 3.10 or later is required. Run these PowerShell commands from the folder where you saved the archive:

```
py -0p
py -3.10 --version
Expand-Archive .\llm_booster_developer_edition-0.1.0.zip .\llm-booster-dev
Set-Location .\llm-booster-dev
py -3.10 -m venv .venv
$wheel = Get-ChildItem -Filter 'llm_booster-*.whl' | Select-Object -First 1
if (-not $wheel) { throw "Developer Edition wheel not found." }
.\.venv\Scripts\python.exe -m pip install $wheel.FullName
.\.venv\Scripts\python.exe -m pip show llm-booster
.\.venv\Scripts\llm-booster.exe --help
```

The virtual environment isolates the installation from system Python. The bundled sample input is examples/context.json. For editable development from a source checkout, use `py -3.10 -m venv .venv` followed by `.\.venv\Scripts\python.exe -m pip install -e .` from the repository root.

2.1. Run the packaged CLI examples

These commands prepare context and print JSON. They do not contact an LLM service:

```
$cli = ".\venv\Scripts\llm-booster.exe"
& $cli run --mode baseline --request invoice --context .\examples\context.json
& $cli run --mode cleanup --request invoice --context .\examples\context.json
& $cli run --mode semantic --top-k 2 --request invoice --context .\examples\context.json
& $cli run --mode linear_network_coding --request invoice --context .\examples\context.json
```

3. Technical behavior and data flow

Component	SDK behavior	Institutional review
Booster.run	Processes request and agent context locally; this call does not invoke an LLM.	Manage input under institutional data policies.
semantic mode	Ranks agents using keyword overlap.	Not model-based semantic understanding or a quality guarantee.
call_local_ollama	When explicitly called, sends HTTP to Ollama, default localhost:11434.	Review model, network, logging, and data retention.
Coding helpers	GF(2) can recover a single packet loss; GF(256) is a separate experiment.	No encryption, authentication, or corruption detection.

Data responsibility: Business, personal, confidential, and customer data remain subject to institutional classification and usage policies. No independent commitment is made regarding telemetry, retention, encryption, or regulatory compliance. Obtain security and legal approval before evaluating with sensitive data.

4. Local models using Ollama

The default CLI and `Booster.run` prepare context only; they do not run a model. Install and start Ollama for Windows (or run `ollama serve` in a separate PowerShell window if its service is not running). Initial model pulls require internet access and disk space; inference uses Ollama's local HTTP API.

Qwen 2.5 3B and NVIDIA Nemotron Mini 4B

```
ollama --version
ollama pull qwen2.5:3b
ollama run qwen2.5:3b
# Type /bye to exit the Ollama prompt.
ollama pull nemotron-mini:4b
ollama run nemotron-mini:4b
```

After confirming a model with `ollama run`, explicitly call `llm_booster.call_local_ollama`. Change the model name to any installed Ollama tag:

```
@'
from llm_booster import call_local_ollama

model = "qwen2.5:3b" # Alternatives: nemotron-mini:4b or kumru:2b.
prompt = "Answer in Turkish: What should I check on invoice 426 TL?"
result = call_local_ollama(prompt, model=model)
if not result["ok"]:
    raise RuntimeError(result["error"])
print(result["text"])
'@ | Set-Content -Encoding utf8 .\local_model_demo.py
.\.venv\Scripts\python.exe .\local_model_demo.py
```

Add Kumru 2B GGUF to Ollama

This example downloads the Q4_K_M GGUF file from `icecubetr/Kumru-2B-GGUF` on Hugging Face and creates an Ollama model using a Modelfile. The download utility is not an SDK runtime dependency:

```
.\.venv\Scripts\python.exe -m pip install huggingface_hub
New-Item -ItemType Directory -Force .\models\kumru | Out-Null
.\.venv\Scripts\hf.exe download `
    icecubetr/Kumru-2B-GGUF kumru-2b-Q4_K_M.gguf `
    --local-dir .\models\kumru

@'
FROM ./models/kumru/kumru-2b-Q4_K_M.gguf
'@ | Set-Content -Encoding ascii .\Modelfile
ollama create kumru:2b -f .\Modelfile
ollama run kumru:2b

# In local_model_demo.py, set:
model = "kumru:2b"
```

Model choice and license: Tags and model cards can change; check each model repository's license, terms, and file provenance. Qwen2.5 3B has different licensing from several other Qwen2.5 sizes; the Kumru GGUF repository identifies the base model under Apache 2.0. Nemotron Mini 4B is optimized for English; evaluate Turkish quality independently. Model weights may be free to download, but hardware can incur cost. The SDK Developer Edition license does not authorize production/commercial use.

The SDK's Ollama default is `http://localhost:11434`. On failure, `call_local_ollama` returns `ok: false` and an error field. `SemanticRouter().route_local_model(..., model="qwen2.5:3b")` invokes the local model separately for each selected agent.

5. License boundaries and safeguards

Permitted under this edition	Not licensed under this edition
Evaluation, testing, and development in an internal sandbox.	Production workloads or commercial services for customers/end users.
Controlled access for employees/contractors bound by the license.	Transfer to customers, unrelated affiliates, or other third parties.
Controlled internal copies required for licensed evaluation.	Resale, sublicensing, OEM/embedded use, or public package hosting.
Internal technical review of code and performance.	Representing the SDK as certified, SLA-backed, or production-ready.

Security and procurement checklist

- Have legal/procurement review the license and verify licensor identity and acceptance requirements.
- Record the wheel SHA-256 and validate the package source.
- Start in an isolated development environment with synthetic, non-sensitive data.
- Review network/data flows separately before using `call_local_ollama` or any external-model application code.
- Check logs, error reports, and test outputs for confidential or personal data.
- Do not move the Developer Edition into production; obtain a separate commercial/enterprise license and technical review.

6. Validation

```

.\.venv\Scripts\python.exe -m pip show llm-booster
# Source repository only; the wheel ZIP does not include tests.
.\.venv\Scripts\python.exe -m unittest discover -s tests -v

```

Institutions receiving only the wheel should verify the CLI help and run a CLI example against an approved local context file.

7. Production or broader distribution needs

The Developer Edition does not grant production, commercial service, enterprise-wide/affiliate distribution, SLA, support, OEM, or embedding rights. Those require a separate agreement and license. This guide makes no price, delivery-time, or support-level commitment.

Legal note: LICENSE-DEVELOPER.md is a draft requiring legal review before distribution. Complete the licensor, governing law, jurisdiction, and notice details before external delivery. This guide is not a security audit or legal advice.

Document version: Developer Edition 0.1.0 • Source: LLM-BOOSTER-SDK