

IDEO-Lab Professional Guide NLLB-200 Local Translation Engine Complete Installation, Administration and Integration Manual

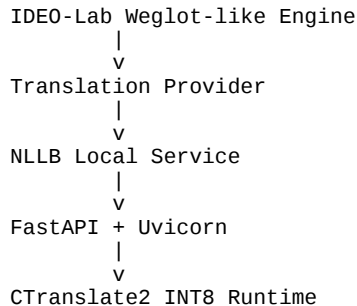
Table of Contents

1. Introduction
2. NLLB Architecture
3. Why NLLB Instead of Google or DeepL
4. Hardware Sizing
5. AWS Deployment Strategy
6. Ubuntu Preparation
7. Python Environment
8. Hugging Face Setup
9. Downloading NLLB Models
10. CTranslate2 Validation
11. Tokenizer Validation
12. First Translation Test
13. Building the FastAPI Service
14. REST API Reference
15. Systemd Service Installation
16. Monitoring and Logs
17. Troubleshooting Guide
18. Security Hardening
19. Nginx Reverse Proxy
20. Weglot-like Integration
21. Django Integration Examples
22. Translation Cache Strategy
23. Production Recommendations
24. Performance Benchmarks
25. Supported Languages
26. Appendix

Introduction

NLLB (No Language Left Behind) is Meta's multilingual translation system.
This guide documents a complete production-grade installation validated on an AWS Ubuntu server.

Target architecture:



Why NLLB

Advantages:

- No per-character billing.
- Local execution.
- More than 200 languages.
- Suitable for large translation workloads.
- Ideal for SaaS translation platforms.

Limitations:

- Requires significant RAM.
- Initial setup is more complex than SaaS APIs.
- Quality can vary depending on language pairs.

Hardware Sizing

Development:

4 CPU / 16 GB RAM

Recommended Production:

8 CPU / 32 GB RAM

Large Production:

16 CPU / 64 GB RAM

Validated IDEO-Lab configuration:

8 vCPU

30 GB RAM

NLLB-200 Distilled 1.3B CT2 INT8

Ubuntu Preparation

```
sudo apt update
sudo apt upgrade -y
```

```
sudo apt install -y python3 python3-venv python3-pip curl wget git
```

Create Environment

```
mkdir ~/nllb_engine
cd ~/nllb_engine
```

```
python3 -m venv venv
source venv/bin/activate
```

Install Dependencies

```
pip install --upgrade pip
pip install ctranslate2 transformers sentencepiece huggingface_hub fastapi uvicorn pydantic
```

Download Model

```
hf auth login

hf download OpenNMT/nllb-200-distilled-1.3B-ct2-int8 --local-dir ./models/nllb-1.3b-int8
```

Validate Model

```
python

import ctranslate2

translator = ctranslate2.Translator(
    "./models/nllb-1.3b-int8",
    device="cpu",
    compute_type="int8"
)

print("MODEL LOADED")
```

Validate Tokenizer

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(
    "./models/nllb-1.3b-int8",
    local_files_only=True
)

print(tokenizer("Hello world"))
```

First Translation Test

French:
Bonjour, ceci est un test de traduction pour Ideo-Lab.

Expected:
Hello, this is a translation test for Ideo-Lab.

FastAPI Service

The service exposes:

```
GET /health
POST /translate
```

Port:
127.0.0.1:8810

API Example

POST /translate

```
{
  "text": "Bonjour le monde",
  "source_lang": "fra_Latn",
  "target_lang": "eng_Latn"
}
```

Response:

```
{
  "text": "Hello world"
}
```

Systemd Installation

```
File:
/etc/systemd/system/ideo-nllb.service

[Unit]
Description=IDEO-Lab NLLB Translation Service

[Service]
WorkingDirectory=/home/gon2000fr/nllb_engine
ExecStart=/home/gon2000fr/nllb_engine/venv/bin/uvicorn app:app --host 127.0.0.1 --port 8810
Restart=always
```

Service Commands

```
sudo systemctl daemon-reload
sudo systemctl enable ideo-nllb
sudo systemctl start ideo-nllb
sudo systemctl status ideo-nllb
```

Monitoring

Health Check:

```
curl http://127.0.0.1:8810/health
```

Logs:

```
sudo journalctl -u ideo-nllb -f
```

Port:

```
ss -lntp | grep 8810
```

Troubleshooting

Address already in use:

```
sudo fuser -k 8810/tcp
```

Restart service:

```
sudo systemctl restart ideo-nllb
```

Security Hardening

Recommendations:

- Bind only on localhost.
- Protect through Nginx.
- Enable rate limiting.
- Restrict firewall access.
- Use HTTPS externally.

Nginx Reverse Proxy

```
location /translate/ {
    proxy_pass http://127.0.0.1:8810;
    proxy_set_header Host $host;
}
```

Weglot-like Integration

Replace Google/DeepL provider.

Current:

```
translate(text)
```

New:

```
POST http://127.0.0.1:8810/translate
```

Django Integration Example

```
import requests

response = requests.post(
    "http://127.0.0.1:8810/translate",
    json={
        "text": text,
        "source_lang": "fra_Latn",
        "target_lang": "eng_Latn"
    }
)
```

Translation Cache Strategy

Strong recommendation:

Store:
SHA256(source_text)

Avoid retranslating identical segments.

Benefits:

- Faster responses
- Lower CPU usage
- Higher scalability

Production Recommendations

1. Use caching.
2. Batch translations.
3. Monitor memory.
4. Use systemd.
5. Backup configuration.
6. Keep models local.
7. Maintain language glossary.

Performance Notes

Observed configuration:

8 CPUs
30 GB RAM

Translation latency:
Short sentence: <1 second
Medium paragraph: 1-3 seconds

Results depend on workload.

Language Codes

eng_Latn English
fra_Latn French
spa_Latn Spanish
deu_Latn German
ita_Latn Italian
nld_Latn Dutch
por_Latn Portuguese

Appendix A - Full Installation Checklist

- [] Ubuntu updated
- [] Python installed
- [] Virtual environment created
- [] Dependencies installed
- [] Model downloaded
- [] Translator validated
- [] Tokenizer validated
- [] Translation test successful
- [] FastAPI service created
- [] Systemd service enabled
- [] Health check validated

[] weglot-like integration completed