**Now back to working on the LLM cleaner: i was exploring faster options, and the other option was together ai, I want to know all the needed aspects about it to integrate it into the project.**  
  
To integrate **Together AI** into your LLM-based data cleaner project, especially as a potential replacement or supplement to your current offline llama-cpp-python setup, you need to understand the following key aspects:

**✅ 1. What is Together AI?**

Together AI is a platform that provides **hosted inference APIs** for open-source LLMs, including fast, scalable access to models like:

* **Mixtral**, **LLaMA 3**, **Gemma**, **Code LLaMA**, **Yi**, **OpenChat**, etc.
* You can use their models via simple HTTP API requests.

**Pros for Your Use Case:**

* Hosted and **fully managed** (no need to run local inference)
* **Fast**, production-ready latency
* **Supports instruct-style prompts** for cleaning, tagging, extracting info
* Optionally **fine-tunable** (advanced)
* Free tier with generous limits
* Python SDK available

**🔐 2. Authentication**

You'll need an **API key**:

* Go to https://platform.together.xyz
* Create an account
* Generate an API key from your dashboard

You will send this key in the Authorization header of your HTTP requests:

http

CopyEdit

Authorization: Bearer <your-api-key>

**⚙️ 3. Endpoint & Request Format**

The basic endpoint:

https

CopyEdit

POST https://api.together.xyz/inference

Or for chat-style models (recommended for your use case):

https

CopyEdit

POST https://api.together.xyz/v1/chat/completions

**Example Request:**

python

CopyEdit

import requests

url = "https://api.together.xyz/v1/chat/completions"

headers = {

"Authorization": f"Bearer {API\_KEY}",

"Content-Type": "application/json"

}

data = {

"model": "mistralai/Mixtral-8x7B-Instruct-v0.1",

"messages": [

{"role": "system", "content": "You are a helpful assistant that extracts a structured problem and solution from user comments."},

{"role": "user", "content": "My car makes a weird noise when I turn left, and I think it's the brakes. I replaced them and now it's fine."}

],

"temperature": 0.2,

"max\_tokens": 300

}

response = requests.post(url, headers=headers, json=data)

print(response.json())

**🧠 4. Model Choices**

You can choose from a wide variety depending on needs:

* For **general extraction/cleaning**: mistralai/Mixtral-8x7B-Instruct-v0.1, meta-llama/Llama-3-8b-chat-hf
* For **code or logic-heavy tasks**: deepseek-ai/deepseek-coder-6.7b-instruct, codellama/CodeLlama-13b-Instruct-hf

They publish a full list of models here

**📊 5. Rate Limits & Pricing**

As of now (Aug 2025):

* **Free tier** available (30M tokens/month)
* **Paid plans** scale up from that
* Tokens are billed as input + output
* Example: 1,000 characters = ~250 tokens

Pricing page

**🔄 6. Replacing llama-cpp-python in Your Pipeline**

You already have this logic:

python

CopyEdit

for uncleaned\_file in find\_files\_to\_process():

df = pd.read\_csv(uncleaned\_file)

for row in df.itertuples():

prompt = generate\_prompt(row)

output = call\_llm(prompt) # currently llama-cpp-python

# parse and update DataFrame

You can refactor call\_llm() to use Together API instead:

python

CopyEdit

def call\_llm\_together(user\_input: str) -> str:

url = "https://api.together.xyz/v1/chat/completions"

headers = {

"Authorization": f"Bearer {API\_KEY}",

"Content-Type": "application/json"

}

data = {

"model": "mistralai/Mixtral-8x7B-Instruct-v0.1",

"messages": [

{"role": "system", "content": SYSTEM\_PROMPT},

{"role": "user", "content": user\_input}

],

"temperature": 0.1,

"max\_tokens": 500

}

response = requests.post(url, headers=headers, json=data)

return response.json()["choices"][0]["message"]["content"]

**🧩 7. When to Use It?**

You might want to:

* Use Together AI **only for new data** (keep old behavior as fallback)
* Use it when llama-cpp-python is offline or slow
* Toggle it via environment variable or CLI flag

**📁 8. Directory-Level Integration Plan**

Your current structure:

css

CopyEdit

data/

raw/YYYY-MM-DD/

cleaned/YYYY-MM-DD/

models/

TheDataCleaningModel/

python\_scripts/

reddit\_data\_cleaner/

llm\_cleaner.py

You might extend llm\_cleaner.py to have:

python

CopyEdit

if USE\_TOGETHER\_AI:

call\_llm = call\_llm\_together

else:

call\_llm = call\_llm\_local

**📌 9. Caveats**

* **Requires internet access**
* Watch for **token limits and latency** on big files
* Not as private as local inference
* Slight chance of **format drift** — need consistent prompt templates

If you want, I can:

* Write a drop-in call\_llm\_together() for your script
* Help compare latency between local vs hosted
* Help build retry logic + backoff

**Applying Together ai to the previous roadmap for the previous roadmap and its effects:**

Adding **Together AI** as an **alternative to llama-cpp-python** in your pipeline would **not disrupt the structure**, but it would influence specific phases — particularly **Phase 3.3 (LLM Cleaning Execution)** — and introduce new **environment, network, and error-handling needs**.

Here's how **each phase** is affected:

**🔍 Phase-by-Phase Impact of Switching to Together AI**

**✅ Phase 0: Environment & Tooling Setup**

**What to change/add**:

* ❌ No need to install or store GGUF models locally.
* ✅ Add **Together AI API key** as an environment variable (TOGETHER\_API\_KEY).
* ✅ (Optional) Install requests if not already in use.
* ✅ (Optional) Add .env management or use config.json.

**New .env example:**

env

CopyEdit

USE\_TOGETHER\_AI=true

TOGETHER\_API\_KEY=sk-xxxxxxxxx

**🔁 Phase 1: Raw Data Ingestion on GitHub**

❌ **No impact** — still GitHub-driven, model-agnostic.

**🕐 Phase 2: Wake-up + Notification System**

❌ No change — it just signals that cleaning is needed.

**🧩 Phase 3.1: Local Boot Scheduler**

**Minor addition**:

* ✅ Add a check for presence of TOGETHER\_API\_KEY if USE\_TOGETHER\_AI=true.
* Optional: Log a warning if Together API key is missing.

**🧠 Phase 3.2: Data Selection & Deduplication**

❌ No change — this phase determines which files need cleaning, not how they're cleaned.

**🧼 Phase 3.3: LLM Cleaning Execution ✅ Primary Impact Zone**

**🔄 Replace:**

python

CopyEdit

result = run\_llm(prompt)

**With:**

python

CopyEdit

if use\_together\_ai:

result = call\_llm\_together(prompt)

else:

result = run\_llm\_local(prompt)

**✅ Add new logic:**

* Retry if Together API request times out (e.g., retry 3 times with exponential backoff)
* Log HTTP errors (429, 500, etc.)
* Detect & gracefully handle malformed output

**✅ New Error Handling Scenarios:**

| **Scenario** | **Strategy** |
| --- | --- |
| 🚫 API quota exceeded | Log error, skip or switch to fallback |
| ⌛ Timeout / no response | Retry 3x, log failure |
| 💣 Bad JSON / malformed reply | Log and skip row |
| 📡 No internet | Log, exit gracefully, optionally fallback to offline model |

**✅ Phase 3.4: Atomic File Writing & Validation**

❌ No change — input/output flow stays the same.

**💾 Phase 4: Git Commit + Push**

❌ No change — same post-processing regardless of LLM engine used.

**📆 Phase 5: Automation Setup**

✅ Recommend updating the script runner to accept a --use\_together\_ai CLI flag or environment toggle:

bash

CopyEdit

python llm\_cleaner.py --use\_together\_ai

**📣 Phase 6: Post-Cleaning Notification**

✅ Optional: Include model used in notification:

✅ "Cleaning for 2025-08-02 complete using Together AI model."

**📋 Phase 7: Error & Health Logging System**

✅ Expand logs/error\_log.txt to capture:

* API errors, timeouts, and Together-related messages
* Fallback usage logs (e.g., "Falling back to local model due to API failure")

**🔁 Updated Flow Summary (Modified Phases in Bold)**

| **Phase** | **Description** | **Trigger** | **Error Handling Summary** | **Model-Specific Impact** |
| --- | --- | --- | --- | --- |
| 0 | Setup | One-time | Add API Key, config switch | ✅ New variable/config |
| 1 | Data Ingestion | GitHub Action | No model impact | ❌ |
| 2 | Notification | GitHub Action | No model impact | ❌ |
| 3.1 | Device Wakeup | System Boot | Check API key | ⚠️ Warn if missing |
| 3.2 | File Selection | Auto-scan | No model impact | ❌ |
| **3.3** | **LLM Cleaning** | **Local/remote** | **Catch HTTP, API, JSON errors** | ✅ Major refactor |
| 3.4 | Output Write | Local | No change | ❌ |
| 4 | Git Commit & Push | Local | No change | ❌ |
| 5 | Automation Setup | One-time | Add toggle | ✅ Optional CLI flag |
| 6 | Notify | Optional | Show model used | ✅ Enhance message |
| 7 | Logging | Throughout | Log API-specific issues | ✅ Expand error log |

**🧰 Action Items for Integration**

* Add a .env or config.py to load API key and toggle
* Refactor run\_llm() into:
  + run\_llm\_local(prompt)
  + run\_llm\_together(prompt)
* Build retry wrapper for Together requests
* Expand error logging and fallback logic
* Update CLI or scheduler to toggle remote mode
* Optionally: add usage stats/logging (requests per day, failures)

If you want, I can generate:

* ✅ A complete run\_llm\_together(prompt) function
* ✅ A config switch system (config.py or .env)
* ✅ CLI parser that supports --use\_together\_ai
* ✅ Logging enhancements

Would you like that packaged now?