**Solution 1**

Now going back to cleaning the extracted data, the plan was take this extracted data and feed it to an offline ai model to clean, what are the general steps to start this phase. The task that i will do is checking if the details in the title and selftext are comprehensive for the final llm and check if the comment contains good solutions for the car issues mentioned in the selftext and title so that it can help car owners who are going to download the app in emergencies and help mechanics to solve the problem/error s in the cars.  
  
  
**✅ Phase Overview: Clean, Enrich, and Prepare for LLM Processing**

You’ll be transforming raw Reddit data into **structured and relevant problem-solution datasets** using an **offline LLM**.

**🔧 PHASE 1: Initial Cleaning & Preprocessing**

**Goal**: Normalize, filter noise, and prepare structured input for the LLM.

**Step 1.1 — Normalize Your Data**

* Fix encoding issues (utf-8, emoji cleanups)
* Lowercase where applicable
* Remove HTML entities, Reddit markdown, broken links

**Step 1.2 — Filter Irrelevant Data**

* Drop empty titles, selftexts, or comments
* Drop [deleted], [removed], bot comments
* Filter based on subreddit quality (you probably already filtered subreddits in extraction)

**Step 1.3 — Add Basic Heuristics**

* Add flags like:
  + has\_car\_model: check if title/selftext mention car models/brands
  + has\_error\_code: regex match for codes like P0300, P0420, etc.
  + has\_solution\_comment: if any comment has keywords like “fixed”, “solved”, “replaced”, etc.

**🧠 PHASE 2: Dataset Structuring for LLM**

**Goal**: Turn each Reddit post into an LLM-friendly sample.

**Step 2.1 — Build an Input Schema**

Structure each sample as:

json

CopyEdit

{

"post\_id": "abc123",

"title": "My 2012 Civic makes a ticking noise",

"selftext": "It starts when I accelerate. I recently changed the oil.",

"comments": [

"Check your lifters, could be worn.",

"Happened to me, turned out to be a loose exhaust manifold.",

...

]

}

You’ll later feed these into an LLM prompt.

**Step 2.2 — Filter Down to “Repair-Intent” Posts**

You may want to discard meme, joke, or news-style posts:

* Use basic keyword filtering: "help", "issue", "problem", "noise", "smoke", "won’t start", etc.
* Optionally classify with a small fine-tuned classifier (simple BERT model or rules-based)

**🤖 PHASE 3: Run Offline LLM to Extract Problem-Solution Pairs**

**Goal**: Summarize the car issue and extract the most relevant solution from comments.

**Step 3.1 — Create Prompt Template**

Example LLM input:

vbnet

CopyEdit

You are a car diagnostic assistant.

Given the Reddit post title, description, and a list of comments, extract the car issue and the most likely solution from the comments.

Title: My 2012 Civic makes a ticking noise

Description: It starts when I accelerate. I recently changed the oil.

Comments:

1. Check your lifters, could be worn.

2. Happened to me, turned out to be a loose exhaust manifold.

3. Take it to a mechanic just in case.

Output:

Issue: Ticking noise during acceleration in a 2012 Honda Civic.

Likely Solution: Check lifters and exhaust manifold for wear or looseness.

You can structure the output as JSON, or text that you later parse.

**Step 3.2 — Batch Process Locally**

* Run your LLM on batches of posts
* Save enriched outputs to a structured format (e.g., JSONL or CSV with issue and solution columns)

If running an offline LLM like Mistral, LLaMA, or DeepSeek, use something like transformers or vLLM to handle inference.

**📦 PHASE 4: Post-Processing & QA**

**Goal**: Final filtering, quality check, and export.

**Step 4.1 — Post-filter Results**

* Drop LLM outputs that say "not enough info"
* Flag uncertain answers ("maybe", "could be")

**Step 4.2 — Manual Review (Optional but Recommended)**

* Spot-check a sample (e.g., 100 outputs)
* Score accuracy of issue and solution pairs
* Feedback loop for improving prompt quality

**📱 PHASE 5: Prepare for Integration into App**

* Store final results in a structured database or Firestore/SQLite for fast mobile access
* Optionally, group solutions by:
  + Car model
  + Problem type (engine, brakes, electronics, etc.)
  + Frequency/popularity of solutions

**🛠️ Tools & Technologies**

| **Task** | **Recommended Tool** |
| --- | --- |
| Preprocessing | pandas, re, nltk, spacy |
| Offline LLM | transformers, vLLM, llama-cpp-python, DeepSeek, etc. |
| QA & Evaluation | Manual + scoring script |
| Export Format | JSON, Parquet, SQLite, Firebase |

 An example of code that prepares the Reddit data for LLM?

 Prompt optimization help?

 Help choosing or configuring an offline LLM for this?

**Solution 2**

Now going back to cleaning the extracted data, the plan was take this extracted data and feed it to an offline ai model to clean, what are the general steps to start this phase. The task that i will do is checking if the details in the title and selftext are comprehensive for the final llm and check if the comment contains good solutions for the car issues mentioned in the selftext and title so that it can help car owners who are going to download the app in emergencies and help mechanics to solve the problem/error s in the cars. Consider that the data is huge and manual work is partially impossible, so the initial cleaning, data structuring for for llm,Run Offline LLM to Extract Problem-Solution Pairs (needs to be autonomous, Post-Processing & QA,Prepare for Integration into App. Also given the fact that this is the next phase pipeline, and we will still need to integrate it somehow to github actions(which might cause problems cause we are working on an offline model, if a model from hugging face will work please tell me), and i will need to modify the previous pipeline to save the artifact to a subdirectory in my repo, then the cleaning code will take the data from that subdirectory and clean it and then save it to another directory. Think outloud for an initial plan to solve these issues.

Great, you're entering an exciting and technically rich phase. Here's a thought-out initial plan to tackle the **Offline LLM-Based Reddit Data Cleaning Pipeline**, aligned with your requirements.

**🧠 Problem Summary**

You have a large dataset of Reddit posts (title + selftext + comments) extracted into CSV files. You want to:

1. Automatically extract **problem-solution pairs**.
2. Use an **offline LLM** (or Hugging Face model if feasible).
3. Integrate the pipeline into **GitHub Actions**, or at least automate it in a modular/reproducible way.
4. Save artifacts to specific repo subdirectories.

**🪜 Step-by-Step Initial Plan**

**✅ 1. Revise Existing Pipeline to Support Artifacts**

* **Save raw extracted CSV files to**: data/raw/
* Modify the last step of the Reddit pipeline to:

yaml

CopyEdit

- name: Upload raw data artifacts

if: always()

uses: actions/upload-artifact@v4

with:

name: raw-reddit-data

path: data/raw/

**🧼 2. Design Cleaning Pipeline (New Phase)**

Create a separate Python script or module:

bash

CopyEdit

reddit\_cleaner/

│

├── clean\_pipeline.py # Entry point to run cleaner logic

├── processor.py # Core logic to structure inputs for LLM

├── prompts/ # Templates to feed LLM

├── postprocessor.py # Sanitize, QA, dedupe

├── config.py # Paths, thresholds, rules

└── data/

├── raw/ # input from extractor

└── cleaned/ # structured outputs

**🔌 3. Choose Model (Offline or Hugging Face)**

* **Option A: Offline**
  + Use [LLama.cpp](https://github.com/ggerganov/llama.cpp) or LM Studio
  + Models: Mistral, DeepSeek-Coder, Phi-3, TinyLlama (if small GPU/CPU)
  + Load via Python wrapper (llama-cpp-python)
* **Option B: Hugging Face (Easier to test with initially)**
  + Model: mistralai/Mistral-7B-Instruct-v0.2 or meta-llama/Meta-Llama-3-8B-Instruct
  + Use with transformers and pipeline(..., model=...)

**🧾 4. Define the Prompt & Task**

Input template for each Reddit entry:

vbnet

CopyEdit

Input:

Title: My car won’t start in the morning

Selftext: It only happens when it's cold, and sometimes the dashboard flickers.

Top Comments:

1. Try changing your battery.

2. I had the same issue — turned out to be the alternator.

Task: Extract the problem and possible solutions. Format:

Problem: ...

Solution(s): ...

Use prompt templating to build prompts at scale (consider jinja2 or even string formatting).

**🤖 5. Run in Batches (autonomously)**

* Avoid feeding entire dataset at once.
* Run in batches:

python

CopyEdit

for chunk in pd.read\_csv('data/raw/extracted.csv', chunksize=50):

results = model.generate(problem\_solution\_prompt(chunk))

save(results)

**🧪 6. Post-Processing and QA**

Clean, deduplicate, and validate:

* Remove hallucinated solutions
* Flag "no solution found"
* Add confidence scores if applicable
* Save structured output to: data/cleaned/structured\_problems.csv

**🔁 7. Offline-Friendly Execution**

Even if GitHub Actions can't run the offline model:

* Automate the trigger via GitHub Action **but execute locally**
* Use a marker file (trigger\_cleaning.txt)
  + If exists, the local script will run the cleaning phase
* Example local script:

bash

CopyEdit

if [ -f trigger\_cleaning.txt ]; then

python clean\_pipeline.py

rm trigger\_cleaning.txt

fi

Or better:

* Use Makefile or Prefect flows to make the pipeline modular

**🔧 8. Structure the Entire Process**

You will end up with this hierarchy:

bash

CopyEdit

reddit\_data\_pipeline/

│

├── extractor/ # Current working pipeline

│ └── data/raw/

│

├── cleaner/ # This new phase

│ ├── data/cleaned/

│ ├── clean\_pipeline.py

│ └── ...

│

├── .github/workflows/

│ ├── reddit\_pipeline.yml

│ └── clean\_trigger.yml # Optional: run cleaning stage or notify dev

**🌐 9. Considerations for Scale**

* Avoid full fine-tuning for now.
* Use LoRA adapters if you later want to personalize the LLM.
* Validate with a small portion first — then batch process.

**✅ Summary of Action Items:**

| **Task** | **Status** |
| --- | --- |
| Save extractor output to data/raw/ | 🟡 Start |
| Build LLM cleaning pipeline | 🔜 Design now |
| Choose model (offline vs. HuggingFace) | 🔜 Decide |
| Write prompt + batch logic | 🔜 Next step |
| Store structured outputs | 🔜 Next step |
| Integrate as sub-workflow | ⏳ Consider later |

Want help writing the clean\_pipeline.py scaffolding or the LLM wrapper?

**would this code push the new data to the main directory: - name: Upload raw data artifacts if: always() uses: actions/upload-artifact@v4 with: name: raw-reddit-data path: data/raw/**

**No, that code does not push the data to the main repo directory. Instead, it uploads the data/raw/ directory as a GitHub Actions artifact, which you can download later from the GitHub Actions UI, but it won't appear in the repository itself.**

**🔍 What that line actually does**

**This part:**

**yaml**

**CopyEdit**

**- name: Upload raw data artifacts**

**uses: actions/upload-artifact@v4**

**with:**

**name: raw-reddit-data**

**path: data/raw/**

**➡ Uploads the contents of data/raw/ as a downloadable artifact in GitHub Actions.  
It does not commit or push to your GitHub repository.**

**✅ If you want to push to your GitHub repo, you need to:**

1. **Add, commit, and push the generated files.**
2. **Use a personal access token (PAT) for authentication.**

**🔧 GitHub Action to Commit and Push Files**

**Here’s how to modify your workflow to push files back to the repo:**

**yaml**

**CopyEdit**

**- name: Commit and push raw data**

**run: |**

**git config --global user.name "github-actions[bot]"**

**git config --global user.email "github-actions[bot]@users.noreply.github.com"**

**git add data/raw/**

**git commit -m "Add new raw Reddit data [$(date +'%Y-%m-%d')]"**

**git push**

**env:**

**GIT\_AUTHOR\_NAME: github-actions**

**GIT\_AUTHOR\_EMAIL: github-actions@github.com**

**GIT\_COMMITTER\_NAME: github-actions**

**GIT\_COMMITTER\_EMAIL: github-actions@github.com**

**But to push successfully, you'll also need to:**

* **Set up a Personal Access Token (PAT) with repo scope.**
* **Save it as a secret, e.g., GH\_PAT.**

**Then modify the push command to:**

**bash**

**CopyEdit**

**git push https://x-access-token:${{ secrets.GH\_PAT }}@github.com/your-username/your-repo.git**

**🛑 TL;DR**

**Your original upload-artifact step is just temporary storage for inspection/logs, not a repo update.**

**If you want cleaned data (or any output) to live in the repo, you must explicitly commit + push using Git commands in your workflow, authenticated with a PAT. Let me know if you want a working example tailored to your repo.**

**You said:**

**I need the data to be commited and pushed automatically in the data/raw subdirectory, then later on we will extract the data from there and feed it to the offline ai model then the new cleaned data will be saved in data/cleaned subdirectory**

**ChatGPT said:**

**Perfect — you’re setting up a two-stage data pipeline within your GitHub repository:**

1. **Stage 1: Reddit scraper runs daily via GitHub Actions, saves CSV data to data/raw/, and commits/pushes it to the repository.**
2. **Stage 2: An offline AI model reads from data/raw/, cleans the content, and saves structured outputs (e.g., problem-solution pairs) into data/cleaned/, which also gets committed/pushed.**

**✅ Step-by-Step: Automatically Push data/raw/ to Repo**

**Here’s what you need to add at the end of your existing GitHub Action (after your scraping step):**

**✅ Full git commit & push step (add after your data is saved)**

**yaml**

**CopyEdit**

**- name: Commit and Push New Raw Data**

**run: |**

**git config --global user.name "github-actions[bot]"**

**git config --global user.email "github-actions[bot]@users.noreply.github.com"**

**# Pull latest changes to avoid push errors**

**git pull origin ${{ github.ref\_name }}**

**# Stage new data**

**git add data/raw/**

**# Check if there's anything to commit**

**if git diff --cached --quiet; then**

**echo "No changes in data/raw/ to commit."**

**else**

**git commit -m "🔄 Add new raw Reddit data [$(date +'%Y-%m-%d %H:%M:%S')]"**

**git push origin ${{ github.ref\_name }}**

**fi**

**env:**

**GIT\_AUTHOR\_NAME: github-actions**

**GIT\_AUTHOR\_EMAIL: github-actions@github.com**

**GIT\_COMMITTER\_NAME: github-actions**

**GIT\_COMMITTER\_EMAIL: github-actions@github.com**

**🔐 Authentication: Use a Personal Access Token (PAT)**

**To allow GitHub Actions to push code:**

1. **Go to GitHub → Settings → Developer Settings → Personal Access Tokens → Tokens (classic).**
2. **Generate a new token with:**
   * **repo scope ✅**
   * **workflow scope ✅ (optional, but good for full automation)**
3. **Save it in your repo under:**
   * **Settings → Secrets and variables → Actions → New repository secret**
   * **Call it: GH\_PAT**

**🔁 Replace the git push line with token-based push:**

**Replace:**

**bash**

**CopyEdit**

**git push origin ${{ github.ref\_name }}**

**With:**

**bash**

**CopyEdit**

**git push https://x-access-token:${{ secrets.GH\_PAT }}@github.com/${{ github.repository }}.git HEAD:${{ github.ref\_name }}**

**This securely pushes your changes using the bot credentials.**

**✅ Output Structure Summary:**

| **Step** | **Directory** | **Action** |
| --- | --- | --- |
| **🔁 Scraper** | **data/raw/** | **Save new Reddit data daily** |
| **💾 Commit** | **data/raw/** | **Auto-push using GitHub Actions** |
| **🧹 Cleaner (offline model)** | **data/raw/ → data/cleaned/** | **Run AI cleaner locally/offline** |
| **💾 Commit** | **data/cleaned/** | **Optionally commit/push cleaned data** |

**Now lets clean up the file struture, where to place the new cleaner, my initial plan was inside python scripts we should make a new directory called data\_cleaner then in that directory it should contain all the needed python code to clean the data \**

**Your initial plan is solid and aligns well with Python project best practices. Here's how to clean up and organize your file structure to separate concerns clearly and make the pipeline modular.**

**✅ Cleaned Project Structure Plan**

**bash**

**CopyEdit**

**Car\_Clinic\_Project/**

**│**

**├── .github/**

**│ └── workflows/**

**│ └── main.yaml # GitHub Actions workflow**

**│**

**├── data/**

**│ ├── raw/ # Raw scraped Reddit data**

**│ └── cleaned/ # Cleaned problem-solution pairs**

**│**

**├── docs/ # Optional: architecture, setup docs**

**│**

**├── python\_scripts/**

**│ ├── reddit\_data\_extractor/ # Existing Reddit scraping pipeline**

**│ │ ├── \_\_init\_\_.py**

**│ │ ├── config.py**

**│ │ ├── extractor.py**

**│ │ ├── extractor\_Base.py**

**│ │ ├── flow.py**

**│ │ ├── reddit\_client.py**

**│ │ ├── scraper.py**

**│ │ ├── utils.py**

**│ │ └── writer.py**

**│ │**

**│ └── data\_cleaner/ # ✅ New cleaning pipeline**

**│ ├── \_\_init\_\_.py**

**│ ├── cleaner.py # Main cleaning logic**

**│ ├── llm\_runner.py # Wrapper for offline LLM**

**│ ├── preprocessor.py # Pre-cleaning (text filters, sanity checks)**

**│ ├── postprocessor.py # Post-LLM cleanup (validate pairs, reformat)**

**│ ├── utils.py # Shared helpers (logging, I/O, etc.)**

**│ └── flow.py # Orchestrate the full cleaning pipeline**

**│**

**├── run\_pipeline.py # Entrypoint for dev/testing both stages**

**├── requirements.txt**

**└── .gitattributes**

**📌 Breakdown of New data\_cleaner/ Directory**

| **File** | **Responsibility** |
| --- | --- |
| **cleaner.py** | **Core logic that reads raw CSV, structures prompts, sends to LLM** |
| **llm\_runner.py** | **Handles inference with the offline LLM or HF model** |
| **preprocessor.py** | **(Optional) Title/Selftext trimming, punctuation cleanup, encoding fixes** |
| **postprocessor.py** | **Clean up LLM results, validate JSONs, drop low-quality answers** |
| **utils.py** | **I/O operations, date/time, file naming, logging** |
| **flow.py** | **Coordinates all the above (can be used from GitHub Actions later)** |

**🔄 Suggested GitHub Workflow Later**

**Once the offline model is ready, you'd:**

1. **Pull latest data from data/raw/**
2. **Run python python\_scripts/data\_cleaner/flow.py**
3. **Save results to data/cleaned/**
4. **Commit + push cleaned data**