**EV Charging Q&A Pipeline: Step-by-Step Guide**

**1. Data Scraping**

The project begins with scraping data from multiple sources related to electric vehicle (EV) charging stations. Two types of sources are used:

* **Web pages**: HTML content from informative sites (Wikipedia, PlugShare, etc.) is parsed and extracted using BeautifulSoup.
* **PDFs**: Research papers and reports are downloaded and processed using pdfplumber.

This step ensures we have a wide range of structured and unstructured text data.

**2. Text Cleaning (Before Chunking)**

Before chunking the data, each scraped document is cleaned to improve downstream results. Cleaning involves:

* Removing **URLs** using regex
* Stripping **HTML tags** from web content
* Eliminating **citations** like [1], [2-3]
* Removing **figure/table references** (e.g., "Figure 1:", "Table 2:")
* Normalizing **whitespace**

This ensures that chunking and language model analysis aren’t affected by noise.

**3. Chunking Methodology**

To convert long documents into manageable pieces, a **hybrid chunking approach** was used:

* **Primary split**: By section headers using regex and structure-based parsing
* **Secondary split** (if too long): Further divided using NLTK's sentence tokenizer if a chunk exceeds the max\_words threshold

Each split maintains the original section heading for context continuity. This method:

* Avoids overwhelming the LLM API
* Increases question quality by narrowing the context
* Ensures semantic integrity

**4. Chunk Size Evaluation**

To determine the best max\_words value, three chunk sizes were tested: **300**, **400**, and **500** words.

Using 10 random samples, each chunk was evaluated using a language model on:

* **Coherence (1-5)**
* **Incomplete Sentence Check (Yes/No)**
* **Token Count**
* **Semantic Overlap (1-5)** with the previous chunk

**Result:**

* 500-word chunks provided the best balance of coherence and completeness, while minimizing API calls.

**5. Evaluation Metrics**

An OpenRouter-compatible LLM was used to evaluate the chunks using these metrics:

* **Coherence**: Logical flow of text
* **Incomplete Sentences**: Whether chunk starts or ends mid-sentence
* **Token Count**: Approximate size
* **Overlap**: Semantic similarity with the previous chunk (1-5 scale)

**6. Q&A Generation**

Once evaluated, each chunk is passed to the Groq API using the **Gemma 2 9B model** to generate **3 Q&A pairs per chunk**.

* API prompt instructs the model to return structured JSON: a list of question-answer pairs
* Rate limits and retries are handled automatically
* Output is stored in a CSV format

**7. Duplicate Removal**

To ensure quality, duplicate and near-duplicate questions are removed using:

* **Fuzzy string matching** with rapidfuzz
* Similarity threshold set via config (90%)

**Alternative considered**: Embedding-based similarity (vector comparison)

* Pros: More accurate
* Cons: Slower and computationally heavier

**Chosen**: Fuzzy matching for speed and simplicity.

**8. Fine-Tuning the Model**

The fine-tuning process was performed on the llama-3.2-3b model using LoRA adapters. The following configuration was used:

model = FastLanguageModel.get\_peft\_model(

model,

r = 16,

target\_modules = ["q\_proj", "k\_proj", "v\_proj", "o\_proj",

"gate\_proj", "up\_proj", "down\_proj"],

lora\_alpha = 16,

lora\_dropout = 0,

bias = "none",

use\_gradient\_checkpointing = "unsloth",

random\_state = 3407,

use\_rslora = False,

loftq\_config = None,

)

**Trainer configuration:**

trainer = SFTTrainer(

model=model,

tokenizer=tokenizer,

train\_dataset=formatted\_dataset['train'],

eval\_dataset=formatted\_dataset['validation'],

dataset\_text\_field="text",

max\_seq\_length=max\_seq\_length,

packing=False,

args=SFTConfig(

per\_device\_train\_batch\_size=2,

gradient\_accumulation\_steps=4,

num\_train\_epochs=10,

warmup\_steps=10,

learning\_rate=1e-5,

logging\_steps=1,

eval\_strategy="steps",

eval\_steps=10,

save\_steps=20,

optim="adamw\_8bit",

weight\_decay=0.01,

lr\_scheduler\_type="linear",

seed=3407,

output\_dir="outputs",

report\_to="wandb" if wandb.run else None,

),

)

Experiments were conducted with multiple learning rates: 1e-5, 2e-5, 5e-6, and 1e-6, and each tested over 5 to 10 epochs. The best-performing configuration (lowest training and evaluation loss) was:

* **Learning rate**: 1e-5
* **Epochs**: 10

All experiments were tracked on **Weights & Biases**.

The final model was saved in **4-bit quantization** and deployed to **Hugging Face Spaces** using CPU on the free tier. Due to CPU limitations, **inference latency** ranged from **200 to 350 seconds** per generation. Future improvements will include GPU-based deployment.

**9. Future Plans**

* **Data Expansion**: Collect more high-quality EV content
* **Further Fine-Tuning**: Incorporate new Q&A pairs
* **RAG Option**: If retraining isn’t feasible, integrate new data using Retrieval-Augmented Generation with a vector database