

**Université Sultan Moulay Slimane**  
École Nationale des Sciences Appliquées de Khouribga  
Département : Mathématiques & Informatique  
Filière : IID2  
Module : Text Mining

---

**Mira Chatbot**  
**Moroccan Darija Chatbot Project Report**  
Projet Darija Chatbot 2025

---



FIGURE 1 – Mira

**Réalisé par :**

M

**Encadré par :**

Pr. Hafidi

Année Universitaire 2025-2026

# Table des matières

|                                                                         |           |
|-------------------------------------------------------------------------|-----------|
| <b>Executive Summary</b>                                                | <b>4</b>  |
| <b>1 Project Objective</b>                                              | <b>6</b>  |
| 1.1 Final Objective . . . . .                                           | 6         |
| 1.2 Why This Is Difficult . . . . .                                     | 6         |
| <b>2 Phase 1 : Data Ingestion and Formatting</b>                        | <b>7</b>  |
| 2.1 Raw Source . . . . .                                                | 7         |
| 2.2 CSV Consolidation . . . . .                                         | 7         |
| 2.3 Why JSON/JSONL Was Preferred Over CSV . . . . .                     | 8         |
| <b>3 Phase 2 : Strict Quality Control and the Purge</b>                 | <b>10</b> |
| 3.1 Cleaning Philosophy . . . . .                                       | 10        |
| 3.2 Cleaning Targets . . . . .                                          | 10        |
| 3.3 Cleaning Logic . . . . .                                            | 11        |
| 3.4 Result . . . . .                                                    | 11        |
| <b>4 Phase 3 : Curriculum Learning Architecture – The Bucket System</b> | <b>12</b> |
| 4.1 Purpose of Bucketing . . . . .                                      | 12        |
| 4.2 Script Buckets . . . . .                                            | 12        |
| 4.3 Content Buckets . . . . .                                           | 13        |
| 4.4 Regex-Based Script Detection . . . . .                              | 13        |
| 4.5 Why Bucketing Was Useful . . . . .                                  | 14        |
| <b>5 Phase 4 : Rule-Based Data Augmentation</b>                         | <b>15</b> |
| 5.1 Objective . . . . .                                                 | 15        |
| 5.2 Metadata Tracking . . . . .                                         | 15        |
| 5.3 Augmentation Rules . . . . .                                        | 16        |
| 5.3.1 Anchor Rows . . . . .                                             | 16        |
| 5.3.2 Intent Multiplication . . . . .                                   | 16        |
| 5.3.3 Linguistic Bridge . . . . .                                       | 16        |
| <b>6 Phase 5 : New Topic Generation and Validation</b>                  | <b>17</b> |

|           |                                                              |           |
|-----------|--------------------------------------------------------------|-----------|
| <b>7</b>  | <b>Phase 6 : Labeling, Manual Review, and Final Cleaning</b> | <b>18</b> |
| 7.1       | Labeling . . . . .                                           | 18        |
| 7.2       | Manual Cleaning . . . . .                                    | 18        |
| <b>8</b>  | <b>Phase 7 : Multi-Turn Conversation Integration</b>         | <b>19</b> |
| 8.1       | Why Multi-Turn Was Needed . . . . .                          | 19        |
| 8.2       | Source . . . . .                                             | 19        |
| 8.3       | Audit and Reconstruction . . . . .                           | 20        |
| <b>9</b>  | <b>Phase 8 : Final SFT Dataset Assembly</b>                  | <b>21</b> |
| 9.1       | Claude’s Final Assembly . . . . .                            | 21        |
| 9.2       | JSONL Repair . . . . .                                       | 22        |
| <b>10</b> | <b>Phase 9 : Mistral Fine-Tuning Attempt</b>                 | <b>23</b> |
| 10.1      | Model and Setup . . . . .                                    | 23        |
| 10.2      | Environment Attempt on Kaggle . . . . .                      | 23        |
| 10.3      | Teacher’s Jupyter Machine . . . . .                          | 23        |
| 10.4      | Mistral Result . . . . .                                     | 24        |
| <b>11</b> | <b>Phase 10 : Gemma Fine-Tuning Attempts</b>                 | <b>25</b> |
| 11.1      | Why Gemma Was Chosen . . . . .                               | 25        |
| 11.2      | Gemma SFT Attempt on Full Dataset . . . . .                  | 25        |
| 11.3      | Gemma Result . . . . .                                       | 26        |
| 11.4      | Gemma with Multi-Turn Dataset . . . . .                      | 26        |
| 11.5      | Arabizi-Only Experiment . . . . .                            | 26        |
| 11.6      | Final Diagnosis of SFT Attempts . . . . .                    | 27        |
| <b>12</b> | <b>Phase 11 : Raw Darija Corpus Construction</b>             | <b>28</b> |
| 12.1      | Why Raw Corpus Was Needed . . . . .                          | 28        |
| 12.2      | Input File . . . . .                                         | 28        |
| 12.3      | Raw Corpus Outputs . . . . .                                 | 28        |
| 12.4      | Raw Corpus Statistics . . . . .                              | 29        |
| 12.5      | Clean CPT File . . . . .                                     | 29        |
| <b>13</b> | <b>Phase 12 : Stage A – Darija Continued Pretraining</b>     | <b>31</b> |
| 13.1      | Goal . . . . .                                               | 31        |
| 13.2      | Correct Model Order . . . . .                                | 31        |
| 13.3      | Recommended Stage A Settings . . . . .                       | 31        |
| 13.4      | Dataset Importation Cell . . . . .                           | 32        |
| 13.5      | CPT Collator Fix . . . . .                                   | 33        |

|                                                    |           |
|----------------------------------------------------|-----------|
| <b>14 Phase 13 : Stage B – SFT After CPT</b>       | <b>35</b> |
| <b>15 Phase 14 : RAG for Hallucination Control</b> | <b>36</b> |
| 15.1 Why RAG Is Needed . . . . .                   | 36        |
| 15.2 Proposed RAG Architecture . . . . .           | 36        |
| 15.3 RAG Prompt . . . . .                          | 37        |
| <b>16 Phase 15 : DPO / Preference Tuning</b>       | <b>38</b> |
| <b>17 Phase 16 : Auxiliary Resources</b>           | <b>39</b> |
| 17.1 Darija Phrasebook . . . . .                   | 39        |
| <b>18 Evaluation Plan</b>                          | <b>40</b> |
| 18.1 Evaluation Categories . . . . .               | 40        |
| 18.2 Metrics . . . . .                             | 40        |
| <b>19 Current File Inventory</b>                   | <b>41</b> |
| <b>20 Final Conclusion</b>                         | <b>42</b> |

# Executive Summary

The objective of this project is to build a Moroccan Darija chatbot able to communicate naturally in Arabic-script Darija, Arabizi, and mixed Darija–French–English. The goal is not to specialize the model in one domain such as economics or politics. The real goal is language adaptation : the base LLM already has general reasoning, world knowledge, English/French ability, and basic instruction-following behavior ; what it lacks is a robust Moroccan Darija communication layer.

The project evolved through several stages. First, raw CSV files containing Moroccan YouTube transcription-derived material were transformed into question–answer data. These files were merged, cleaned, converted into JSON/JSONL, bucketed by script and intent, augmented, and expanded. Later, multi-turn conversation data from the Hugging Face dataset `MBZUAI-Paris/Darija-SFT-Mixture` was integrated to teach conversational behavior. Several fine-tuning attempts were then performed with Mistral and Gemma. These attempts proved that the models could learn the dataset format and Q/A behavior, but they did not reliably acquire deep Darija fluency. The main diagnosis was that supervised fine-tuning alone teaches the model to imitate answer style ; it does not necessarily teach the language distribution itself.

The current solution is therefore a staged pipeline :

- Stage A – Darija continued pretraining (CPT)** : teach next-token Darija texture and language patterns using raw Darija text.
- Stage B – Supervised fine-tuning (SFT)** : teach assistant behavior using chat-formatted Darija conversations.
- Stage C – Retrieval-Augmented Generation (RAG)** : reduce factual hallucination by grounding answers in retrieved documents.
- Stage D – DPO / preference tuning** : prefer native Darija answers over MSA-like, robotic, translated, or hallucinated answers.
- Stage E – Evaluation** : measure Darija naturalness, Arabizi quality, hallucination rate, multi-turn consistency, and retention of English/French/reasoning abilities.

The current best baseline is the Gemma SFT-only model. The next experimental step is to start again from a fresh `google/gemma-3-4b-it`, apply Darija continued pretraining using `darija_cpt_clean.jsonl`, then run SFT again on the chat dataset. This will test whether language-adaptive pretraining improves Darija fluency before assistant-style fine-

tuning.

# Chapitre 1

## Project Objective

### 1.1 Final Objective

The final objective is :

To adapt a general LLM so that it can communicate naturally in Moroccan Darija, in Arabic script, Arabizi, and mixed Darija–French–English, while preserving the base model’s general reasoning and multilingual abilities.

This means the project is not a domain-specific Moroccan economics chatbot. It is a Moroccan language-support chatbot. The model should be able to answer a wide range of user questions in Darija, not only questions that resemble the training dataset.

### 1.2 Why This Is Difficult

A normal SFT dataset teaches the model a mapping :

`user question → assistant answer`

If most training rows are long factual answers, the model learns long factual answer behavior. It does not automatically acquire the deep next-token distribution of Darija. A language is learned mainly through broad exposure to natural text. For English and French, pretrained models see massive corpora before instruction tuning. For Darija, such broad exposure is much weaker in most open LLMs.

Therefore, the project moved from a pure SFT plan to a layered plan :

**Darija language exposure → assistant behavior → grounded factual  
answering → preference alignment**

# Chapitre 2

## Phase 1 : Data Ingestion and Formatting

### 2.1 Raw Source

The first raw sources were CSV files created from transcriptions of Moroccan YouTube videos. These CSV files were generated from Moroccan YouTubers' content and then transformed into question-answer pairs using Gemini. Each CSV file contained structured Q/A material extracted from one or more initial transcript sources.

### 2.2 CSV Consolidation

The first engineering task was to consolidate fragmented CSV files into one master CSV file. This step provided a single place where all initial Q/A pairs could be inspected, cleaned, and transformed.

Listing 2.1 – Simplified CSV consolidation logic

```
import pandas as pd
from pathlib import Path

csv_files = list(Path("csv_sources").glob("*.csv"))
frames = []

for path in csv_files:
    df = pd.read_csv(path)
    df["source_file"] = path.name
    frames.append(df)

master = pd.concat(frames, ignore_index=True)
master.to_csv("master_dataset.csv", index=False)
```

## 2.3 Why JSON/JSONL Was Preferred Over CSV

CSV was useful for initial spreadsheet-like consolidation, but JSON/JSONL is better for LLM training and dataset engineering for several reasons :

1. **Hierarchical structure** : chat training requires nested structures such as `messages`, with roles like `system`, `user`, and `assistant`. CSV is flat and cannot naturally represent multi-turn conversations.
2. **Metadata support** : JSON allows each row to contain fields such as `id`, `parent_id`, `script`, `topic`, `augmentation_type`, and `quality_status`. CSV can store these fields, but becomes fragile when rows contain long text, newlines, commas, or nested data.
3. **Training compatibility** : Hugging Face, TRL, PEFT, and modern SFT pipelines commonly consume JSON or JSONL datasets. JSONL is especially convenient because each line is independently parsed.
4. **Safety with multiline answers** : Darija answers may contain punctuation, commas, code snippets, or multi-line explanations. CSV can break easily if escaping is inconsistent.
5. **Conversation representation** : multi-turn examples need an array of messages. This is natural in JSON and awkward in CSV.

The first transformed structure was :

Listing 2.2 – Initial Q/A JSON format

```
{
  "question": "...",
  "answer": "..."
}
```

Later, for chat fine-tuning, the structure became :

Listing 2.3 – Chat-format JSONL structure

```
{
  "messages": [
    {"role": "system", "content": "..."},
    {"role": "user", "content": "..."},
    {"role": "assistant", "content": "..."}
  ],
  "meta": {
    "topic": "...",
```

```
    "script": "..."  
  }  
}
```

# Chapitre 3

## Phase 2 : Strict Quality Control and the Purge

### 3.1 Cleaning Philosophy

The guiding rule was :

Quality is more important than quantity. One noisy epoch can harm the model more than no training at all.

The initial raw merged dataset had over 6,800 rows. After strict filtering, the baseline was reduced to approximately 6,100 high-confidence rows. The purpose was not merely to delete data, but to create a trusted seed dataset from which later augmentation could be performed safely.

### 3.2 Cleaning Targets

The cleaning engine targeted :

- exact duplicates ;
- semantic or near-duplicate questions when they were not intentional augmentation ;
- incomplete sentences ;
- trailing punctuation artifacts ;
- broken characters and encoding noise ;
- web-scraping artifacts ;
- unnatural mixed-script “Frankenstein words” ;
- empty or tiny answers ;
- rows that sounded like machine translation or formal MSA instead of Darija.

### 3.3 Cleaning Logic

Listing 3.1 – Simplified row-cleaning logic

```
def is_bad_row(question, answer):
    if not question.strip() or not answer.strip():
        return True
    if len(answer.strip()) < 20:
        return True
    if has_encoding_noise(question) or has_encoding_noise(answer):
        return True
    if has_frankenstein_words(question) or has_frankenstein_words(answer):
        return True
    if looks_truncated(answer):
        return True
    return False
```

### 3.4 Result

The strict purge produced a clean baseline of roughly 6,100 rows. This baseline later became the anchor for augmentation. The important principle was that these original rows were frozen as ground truth.

# Chapitre 4

## Phase 3 : Curriculum Learning Architecture – The Bucket System

### 4.1 Purpose of Bucketing

Bucketing was introduced to organize the dataset across script and intent. The goal was to prevent the model from receiving an uncontrolled mix of scripts and task types. The model needed examples of :

- Arabic-script Darija ;
- Arabizi / Latin-script Darija ;
- mixed code-switching ;
- factual/objective answers ;
- subjective/creative answers.

### 4.2 Script Buckets

The script buckets were detected using regex-based character scanning.

| Bucket | Name                 | Purpose                                                                                                           |
|--------|----------------------|-------------------------------------------------------------------------------------------------------------------|
| A      | Pure Arabic script   | Rows written only with Arabic Unicode characters. These teach Arabic-script Darija structure.                     |
| B      | Pure Arabizi / Latin | Rows written with Latin characters and Moroccan numerals such as 3, 7, 9. These teach texting/youth Darija style. |

|   |                        |                                                                                                                |
|---|------------------------|----------------------------------------------------------------------------------------------------------------|
| C | Mixed / code-switching | Rows containing both Arabic and Latin scripts. These teach natural Moroccan bilingual/code-switching behavior. |
|---|------------------------|----------------------------------------------------------------------------------------------------------------|

---

## 4.3 Content Buckets

The content axis classified the intent of the row. In some project notes, the second content bucket was referred to differently ; this report standardizes the content axis as D/E.

| Bucket | Name                  | Purpose                                                                             |
|--------|-----------------------|-------------------------------------------------------------------------------------|
| D      | Factual / Objective   | Hard data, history, procedures, economics, definitions, and grounded explanations.  |
| E      | Subjective / Creative | Advice, storytelling, opinions, brainstorming, empathy, and human-like interaction. |

---

The final bucket combinations include :

A+D, A+E, B+D, B+E, C+D, C+E

## 4.4 Regex-Based Script Detection

Listing 4.1 – Simplified script bucket detection

```
import re

ARABIC_RE = re.compile(r"[\u0600-\u06FF]")
LATIN_RE = re.compile(r"[A-Za-z0-9]")

def detect_script_bucket(text):
    has_ar = bool(ARABIC_RE.search(text))
    has_la = bool(LATIN_RE.search(text))

    if has_ar and has_la:
        return "C" # mixed
```

```
if has_ar:
    return "A"  # Arabic script
if has_la:
    return "B"  # Arabizi / Latin
return "U"     # uncertain
```

## 4.5 Why Bucketing Was Useful

Bucketing gave the dataset a curriculum-like organization. It made it possible to inspect whether Arabic-script, Arabizi, and mixed rows were represented. It also helped separate objective factual examples from subjective conversational examples. This was important because the final model should not only answer factual questions; it should also speak naturally, handle casual interaction, and maintain Moroccan tone.

# Chapitre 5

## Phase 4 : Rule-Based Data Augmentation

### 5.1 Objective

After the baseline of approximately 6,100 rows, the dataset needed to scale. Blind generation was rejected because it could introduce hallucinated facts, unnatural style, and translation artifacts. Instead, augmentation was controlled and traceable.

### 5.2 Metadata Tracking

Each row was assigned metadata before augmentation :

Listing 5.1 – Example augmentation metadata

```
{
  "id": "row_000001",
  "parent_id": null,
  "augmentation_type": "original",
  "stage_1_decision": "approved",
  "question": "...",
  "answer": "..."
}
```

## 5.3 Augmentation Rules

### 5.3.1 Anchor Rows

The 6,100 baseline rows were frozen as original anchors. They represented the trusted data source.

### 5.3.2 Intent Multiplication

Questions were reframed while answers were frozen. This taught the model that different user phrasings can map to the same factual answer.

`same answer + different question wording`

This is why duplicate answers were not automatically considered errors. They were intentional paraphrase anchors, as long as they were capped and meaningful.

### 5.3.3 Linguistic Bridge

Script mirroring converted Arabic-script rows to Arabizi or vice versa without changing the meaning. This helped link equivalent semantics across scripts.

`Arabic-script Darija ↔ Arabizi`

# Chapitre 6

## Phase 5 : New Topic Generation and Validation

After the rule-based augmentation stage, approximately 5,000 new rows were generated to cover missing topics and improve diversity. The generation process used Gemma for row generation and Atlas for validation. The purpose was not to generate random data, but to fill gaps while maintaining the Darija language target.

After this stage, the dataset reached more than 20,000 clean rows. These rows were later merged, cleaned, and converted into the larger final dataset used for SFT experiments.

# Chapitre 7

## Phase 6 : Labeling, Manual Review, and Final Cleaning

### 7.1 Labeling

The dataset was labeled using script and topic/intent metadata. The labeling outputs included files such as :

- `dataset_darija_labeled.json`;
- `dataset_darija_labeled_same_structure.json`;
- `dataset_darija_labeling_report.json`;
- `arabizi_bucket_labeled.json`;
- `arabizi_bucket_labeled_report.json`.

A labeling report on one dataset version contained 26,630 rows and showed a script-pair distribution of approximately 13,086 Arabic-script rows, 12,257 Latin-script rows, 946 mixed pairs, and 341 mixed-script rows. It also reported issue categories such as duplicate-answer anchors, wrapper prefixes, duplicate questions, short answers, decimal spacing artifacts, and glued words.

### 7.2 Manual Cleaning

The final cleaning stage included manual inspection and correction. This was necessary because scripts can detect obvious problems, but only a Moroccan speaker can reliably judge whether a row sounds native, translated, robotic, or MSA-heavy.

# Chapitre 8

## Phase 7 : Multi-Turn Conversation Integration

### 8.1 Why Multi-Turn Was Needed

Single-turn Q/A teaches the model to answer isolated questions. It does not teach how to maintain context across turns. For a chatbot, this is a major limitation.

Therefore, multi-turn data was added to teach :

- follow-up handling ;
- context retention ;
- user correction ;
- clarification behavior ;
- staying in Darija over several turns.

### 8.2 Source

The multi-turn source was the Hugging Face dataset :

`MBZUAI-Paris/Darija-SFT-Mixture`

Two parquet files were used in the project pipeline :

- `conversations_20k1.parquet` ;
- `conversations_20k2.parquet`.

## 8.3 Audit and Reconstruction

The two parquet files reused conversation IDs independently. A naive merge would have mixed unrelated conversations. The reconstruction fixed this by prefixing IDs per file and splitting threads whenever the turn number reset.

The combined audit found :

- 5,326 Q/A turns ;
- 2,313 conversations ;
- no empty questions or answers ;
- no turn-order problems ;
- small normalized overlap across files.

Some raw conversations were identified as translation-style, English-answer suspects, synthetic instruction contexts, or code/error-log contexts. The final assembly kept useful examples and removed the most problematic ones.

# Chapitre 9

## Phase 8 : Final SFT Dataset Assembly

### 9.1 Claude’s Final Assembly

Claude produced the final SFT training package. The relevant files included :

- `train_final.jsonl`;
- `val_final.jsonl`;
- `train_single.jsonl`;
- `train_multi.jsonl`;
- `multiturn_removed.json`;
- `BUILD_REPORT.txt`;
- `DATASET_CARD.md`.

The final assembly report contained :

| Item                                   | Count  |
|----------------------------------------|--------|
| Single-turn train                      | 36,999 |
| Single-turn validation                 | 1,138  |
| Raw multi-turn conversations recovered | 2,313  |
| Multi-turn conversations kept          | 2,310  |
| Multi-turn conversations removed       | 3      |
| Multi-turn train                       | 2,241  |
| Multi-turn validation                  | 69     |
| Multi-turn upsampling                  | ×2     |
| Final train records                    | 41,481 |
| Final validation records               | 1,207  |

The dataset card described the final format as a Mistral-Instruct messages schema with a system message, alternating user/assistant turns, and a `meta` object for audit only.

## 9.2 JSONL Repair

The first `train_final.jsonl` contained invalid JSON escaping in some code/math examples. Specifically, some raw backslashes were not escaped correctly. A repair step produced :

- `train_final_repaired.jsonl`;
- `val_final_repaired.jsonl`;
- `jsonl_escape_repair_report.json`;
- `jsonl_bad_lines_summary.txt`.

The repaired files had :

- 41,481 train records;
- 1,207 validation records;
- zero JSON parse errors.

# Chapitre 10

## Phase 9 : Mistral Fine-Tuning Attempt

### 10.1 Model and Setup

The first serious model experiment used Mistral :

```
mistralai/Mistral-7B-Instruct-v0.3
```

The target was to teach the model Moroccan Darija communication using QLoRA/LoRA. The training used chat-format JSONL and assistant-response masking.

### 10.2 Environment Attempt on Kaggle

A Kaggle notebook was created first. The data issue was fixed, but Kaggle then failed during model loading because of a bitsandbytes/CUDA/Triton mismatch :

- No module named triton.ops;
- bitsandbytes compiled without GPU support;
- missing CUDA-linked bitsandbytes library.

This made Kaggle unreliable for the Mistral run.

### 10.3 Teacher's Jupyter Machine

The project then moved to a teacher-provided JupyterLab machine. The machine had :

- NVIDIA RTX 4090 GPU;
- 24 GB VRAM;
- AMD Ryzen 9 7900X CPU;

- approximately 61 GB RAM;
- PyTorch 2.11.0 + CUDA 13.0;
- bitsandbytes 0.49.2 with CUDA callable successfully.

This environment was suitable for 4-bit QLoRA.

## 10.4 Mistral Result

The Mistral training technically worked. However, the output quality was not satisfactory. The model learned Q/A and conversation behavior but failed to acquire robust Darija style and pattern. Some outputs were incoherent or unnatural, especially in Arabizi. The diagnosis was :

Mistral learned the answer distribution and dataset behavior, but did not learn Darija deeply as a language.

This became the first major failed attempt, but it was useful because it proved the training infrastructure worked and revealed the missing language-adaptation layer.

# Chapitre 11

## Phase 10 : Gemma Fine-Tuning Attempts

### 11.1 Why Gemma Was Chosen

After Mistral struggled, Gemma was tested because it appeared to have stronger multilingual behavior and possibly more useful prior exposure to Arabic-like and mixed-script patterns. The selected model was :

`google/gemma-3-4b-it`

The 4B model was chosen because it was realistic to fine-tune on the RTX 4090 with QLoRA.

### 11.2 Gemma SFT Attempt on Full Dataset

The first Gemma SFT attempt used a dataset with approximately 40,025 conversations. The notebook split it into :

- 38,826 training records ;
- 1,199 validation records.

The run completed successfully :

- model loaded in 4-bit ;
- LoRA trainable parameters : approximately 11.9M ;
- full training completed ;
- training time : approximately 117 minutes ;
- best evaluation loss : approximately 2.5078 ;

- adapter saved ;
- merged model saved ;
- zip created at approximately 6.82 GB.

The final packaging cell failed with `IN_COLAB` not defined, but this occurred after model saving and did not affect training.

## 11.3 Gemma Result

Gemma performed better than the Mistral attempt, but still learned the Q/A behavior more than the underlying Darija texture. Some outputs were understandable, but the model still produced MSA-flavored, incoherent, or hallucinated Darija in some prompts.

## 11.4 Gemma with Multi-Turn Dataset

A second Gemma training attempt used the richer dataset with multi-turn examples. The goal was to teach conversational behavior more explicitly. The result improved conversation behavior but did not fundamentally solve the Darija-pattern problem. The model still tended to imitate dataset style and hallucinated under some prompts.

## 11.5 Arabizi-Only Experiment

A later experiment used only the Arabizi portion of the dataset. The motivation was that the model had difficulty capturing Arabizi patterns. The experiment used a smaller Latin-script subset. In the notebook snapshot, the run used :

- 8,720 training records ;
- 269 validation records ;
- Latin/Arabizi script only ;
- LoRA rank 32 and alpha 64 ;
- 3 epochs ;
- learning rate 0.0002 ;
- 65,576,960 trainable parameters.

This run improved Arabizi slightly, but not enough. The model still had hallucination and unnatural generation issues.

## 11.6 Final Diagnosis of SFT Attempts

The key conclusion was :

SFT alone teaches the model how to answer like the dataset. It does not necessarily teach the model the deep statistical texture of Darija.

This is why the project moved to continued pretraining before SFT.

# Chapitre 12

## Phase 11 : Raw Darija Corpus Construction

### 12.1 Why Raw Corpus Was Needed

The failure of SFT-only training showed that the model needed raw Darija language exposure. In normal LLM training, language ability comes from large-scale next-token pretraining. Therefore, the next step was to create a raw Darija corpus from the existing chat data.

### 12.2 Input File

The raw corpus was built from :

`dataset_reformed.jsonl`

This file contained 40,025 records. Its inspected script distribution was approximately :

| Script          | Records |
|-----------------|---------|
| Arabic          | 27,359  |
| Latin / Arabizi | 11,826  |
| Mixed           | 840     |

The topic distribution included `quotidien`, `autre`, `economie`, `education`, `technologie`, `social`, `politique`, `religion`, `sport`, and others.

### 12.3 Raw Corpus Outputs

The generated raw-corpus package contained :

- darija\_raw\_corpus\_strict.txt;
- darija\_raw\_corpus\_strict\_dedup.txt;
- darija\_raw\_corpus\_arabic.txt;
- darija\_raw\_corpus\_latin.txt;
- darija\_raw\_corpus\_mixed.txt;
- darija\_conversation\_blocks.jsonl;
- darija\_raw\_corpus\_tagged\_utterances.jsonl;
- darija\_cpt\_train\_chunks.jsonl;
- darija\_cpt\_val\_chunks.jsonl;
- darija\_raw\_corpus\_report.json.

## 12.4 Raw Corpus Statistics

The raw corpus report showed :

| Metric                                     | Value  |
|--------------------------------------------|--------|
| Source records                             | 40,025 |
| Full non-system utterances                 | 85,832 |
| Strict Darija utterances                   | 83,984 |
| Strict deduplicated utterances             | 75,421 |
| Strict conversation blocks                 | 39,567 |
| Translation-style records detected         | 427    |
| Wrapper prefixes stripped in strict corpus | 3,036  |
| Train utterances                           | 81,449 |
| Validation utterances                      | 2,535  |
| Original train chunks                      | 5,675  |
| Original validation chunks                 | 171    |
| Approximate whitespace tokens              | 3.1M   |

For raw continued pretraining, wrappers were stripped even though the teacher required them in the SFT dataset. The reason is that raw CPT should teach natural Darija distribution, not synthetic prompt scaffolding.

## 12.5 Clean CPT File

A cleaned CPT file was later created :

`darija_cpt_clean.jsonl`

It had :

- 5,220 chunks;
- zero JSONL parse errors;
- one required key : `text`;
- zero empty text rows;
- zero duplicate normalized chunks;
- median chunk length around 3,100 characters.

This is the current recommended file for Stage A continued pretraining.

# Chapitre 13

## Phase 12 : Stage A – Darija Continued Pretraining

### 13.1 Goal

Stage A is designed to teach the model Darija language texture through causal language modeling :

`Darija text → predict next token`

This differs from SFT, which teaches question-answering behavior.

### 13.2 Correct Model Order

The correct order is :

`fresh Gemma 3 4B → CPT → SFT → DPO/RAG`

The previous SFT model should not be used as the starting point for Stage A. It remains a v1 baseline. Stage A should begin from the fresh base model.

### 13.3 Recommended Stage A Settings

| Parameter  | Recommended Value    |
|------------|----------------------|
| Base model | google/gemma-3-4b-it |
| Method     | QLoRA                |

|                     |                                          |
|---------------------|------------------------------------------|
| Objective           | causal language modeling                 |
| Dataset             | darija_cpt_clean.jsonl                   |
| Learning rate       | $1 \times 10^{-5}$ or $2 \times 10^{-5}$ |
| Epochs              | 1                                        |
| Max sequence length | 2048                                     |
| Batch size          | small, with gradient accumulation        |
| LoRA rank           | 16                                       |
| Output              | CPT LoRA adapter                         |

## 13.4 Dataset Importation Cell

Listing 13.1 – Clean CPT dataset importation cell

```
import os, json, glob, random

SEED = 42
VAL_RATIO = 0.03
EXPECTED_NAME = "darija_cpt_clean.jsonl"
random.seed(SEED)

def find_dataset_file(filename=EXPECTED_NAME):
    candidates = []
    candidates += glob.glob(f"/kaggle/input/**/{filename}", recursive=True)
    candidates += glob.glob(f"/kaggle/working/**/{filename}", recursive=True)
    candidates += glob.glob(f"./**/{filename}", recursive=True)
    candidates += [filename, f"data/{filename}", f"/workspace/{filename}", f"/workspace/data/{filename}"]

    seen = set()
    clean = []
    for p in candidates:
        if p not in seen and os.path.exists(p):
            seen.add(p)
            clean.append(p)

    if not clean:
        raise FileNotFoundError(f"Could not find {filename}")
    return clean[0]

DATASET_PATH = find_dataset_file()
```

```
rows = []
with open(DATASET_PATH, "r", encoding="utf-8") as f:
    for line_no, line in enumerate(f, start=1):
        line = line.strip()
        if not line:
            continue
        obj = json.loads(line)
        assert "text" in obj
        text = str(obj["text"]).strip()
        assert text
        rows.append({"text": text})

random.shuffle(rows)
val_size = max(1, int(len(rows) * VAL_RATIO))
val_rows = rows[:val_size]
train_rows = rows[val_size:]

print("Train rows:", len(train_rows))
print("Val rows:", len(val_rows))
```

## 13.5 CPT Collator Fix

The first CPT notebook created labels before padding, causing variable-length label errors. The correct solution is to create labels after padding.

Listing 13.2 – CPT collator for causal language modeling

```
class CausalLMCollator:
    def __init__(self, tokenizer, pad_to_multiple_of=8):
        self.tokenizer = tokenizer
        self.pad_to_multiple_of = pad_to_multiple_of

    def __call__(self, features):
        features = [
            {
                "input_ids": f["input_ids"],
                "attention_mask": f["attention_mask"],
            }
            for f in features
        ]
```

```
batch = self.tokenizer.pad(  
    features,  
    padding=True,  
    pad_to_multiple_of=self.pad_to_multiple_of,  
    return_tensors="pt",  
)  
  
labels = batch["input_ids"].clone()  
labels[batch["attention_mask"] == 0] = -100  
batch["labels"] = labels  
return batch
```

# Chapitre 14

## Phase 13 : Stage B – SFT After CPT

After Stage A, the CPT-adapted model should be used as the starting point for assistant fine-tuning. Stage B uses the chat dataset :

`dataset_reformed.jsonl`

The SFT objective teaches the assistant to respond in Darija, follow the user script, maintain chat behavior, and support multi-turn conversations.

Recommended settings :

- start from fresh Gemma + CPT adapter ;
- use assistant-only loss ;
- ignore `meta` ;
- keep epochs low, usually 1 ;
- learning rate around  $3 \times 10^{-5}$  to  $5 \times 10^{-5}$  ;
- evaluate against the old Gemma SFT-only baseline.

# Chapitre 15

## Phase 14 : RAG for Hallucination Control

### 15.1 Why RAG Is Needed

Fine-tuning should not be used to memorize current laws, prices, procedures, or factual details. For factual questions, the model should retrieve information from documents.

RAG should be used for :

- Moroccan law and procedures ;
- administrative documents ;
- school or university rules ;
- technical documentation ;
- project-specific knowledge ;
- current or source-dependent information.

RAG should not be used for simple small talk, greetings, casual emotional support, or general conversation.

### 15.2 Proposed RAG Architecture

`user query → router → retrieval if needed → Gemma Darija answer`

Recommended components :

- embedding model : BAAI/bge-m3 ;
- vector database : FAISS for the first demo ;
- generator : final CPT+SFT Gemma model ;
- prompt rule : use context for facts, answer in natural Darija, and refuse to invent if context is insufficient.

### 15.3 RAG Prompt

Listing 15.1 – Darija RAG system prompt

# Chapitre 16

## Phase 15 : DPO / Preference Tuning

DPO should be used after CPT+SFT, not before. The purpose is to teach the model that native Darija answers are preferred over MSA-like, robotic, translated, or hallucinated answers.

Example :

Listing 16.1 – Preference pair for Darija style

```
{
  "prompt": "          ",
  "chosen": "          .",
  "rejected": "          ."
}
```

For hallucination control :

Listing 16.2 – Preference pair for anti-hallucination behavior

```
{
  "prompt": "          ",
  "chosen": "          .",
  "rejected": "          ..."
}
```

# Chapitre 17

## Phase 16 : Auxiliary Resources

### 17.1 Darija Phrasebook

A Darija phrasebook was added as a possible auxiliary resource. It includes pronunciation conventions, greetings, first encounters, food, souk, time, numbers, money, health, agriculture, and architecture vocabulary.

It should not be trained directly as English-to-Darija translation data. Instead, it should be used for :

- RAG dictionary entries ;
- evaluation prompts ;
- phrase normalization ;
- situational dialogue generation.

# Chapitre 18

## Evaluation Plan

### 18.1 Evaluation Categories

The final model should be evaluated using a fixed prompt set covering :

- Arabic-script Darija ;
- Arabizi ;
- mixed French/Darija ;
- casual small talk ;
- multi-turn follow-up ;
- technical questions ;
- factual questions with RAG ;
- English/French forgetting tests ;
- uncertainty and refusal behavior.

### 18.2 Metrics

Validation loss alone is not enough. The important metrics are :

- Darija naturalness score from Moroccan speakers ;
- MSA leakage rate ;
- Arabizi coherence ;
- factual hallucination rate ;
- multi-turn consistency ;
- refusal correctness when sources are missing ;
- retention of English/French and reasoning ability.

# Chapitre 19

## Current File Inventory

| File                                  | Role in Project                                                                                                   |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| dataset_reformed.jsonl                | Main chat/SFT dataset with 40,025 records. Contains <code>messages</code> and <code>meta</code> .                 |
| train_final_repaired.jsonl            | Repaired Mistral-style final train split with 41,481 records.                                                     |
| val_final_repaired.jsonl              | Repaired validation split with 1,207 records.                                                                     |
| darija_raw_corpus_package.zip         | Raw corpus package extracted from SFT data for language-adaptive pretraining.                                     |
| darija_cpt_clean.jsonl                | Current clean Stage A CPT file with 5,220 text chunks.                                                            |
| darija-phrasebook.pdf                 | Auxiliary phrasebook for vocabulary, RAG dictionary, and evaluation prompts.                                      |
| labeli.zip                            | Project archive containing earlier cleaning reports, datasets, notebooks, final train/val files, and audit files. |
| Untitled.ipynb                        | One of the model training notebooks from the local Jupyter experiments.                                           |
| Untitled (1).ipynb                    | Gemma/local notebook variant or setup notebook.                                                                   |
| Untitled (2).ipynb                    | Gemma full SFT notebook snapshot.                                                                                 |
| Untitled (3).ipynb                    | Arabizi-focused Gemma notebook snapshot.                                                                          |
| gemma3_darija_stage1_cpt_kaggle.ipynb | Stage 1 CPT Kaggle notebook.                                                                                      |

# Chapitre 20

## Final Conclusion

This project evolved from a simple dataset-building task into a full low-resource language-adaptation pipeline. The first assumption was that a large Q/A SFT dataset would be enough to teach Darija. The experiments disproved that assumption. Mistral and Gemma both learned assistant behavior and dataset style, but neither fully acquired robust native Darija fluency from SFT alone.

The current diagnosis is clear :

The project does not only need more supervised fine-tuning. It needs a Darija language foundation before assistant fine-tuning.

The current solution is therefore :

1. run Darija continued pretraining on `darija_cpt_clean.jsonl` ;
2. run SFT again using `dataset_reformed.jsonl` ;
3. add RAG for factual grounding ;
4. add DPO preference tuning for native Darija style and anti-hallucination behavior ;
5. evaluate with fixed prompts and Moroccan human judgment.

This path directly addresses the core problems discovered during the project :

- CPT addresses weak Darija texture ;
- SFT addresses assistant behavior ;
- RAG addresses factual hallucination ;
- DPO addresses MSA leakage, robotic tone, and fake confidence ;
- evaluation prevents guessing whether the model improved.

The next milestone is to complete Stage A and compare :

Gemma SFT-only baseline    vs.    Gemma CPT + SFT

If the CPT+SFT model improves Darija flow and reduces hallucinated phrasing, the diagnosis will be confirmed and the project can proceed to RAG and DPO as the final improvement layers.