**Project Documentation: Arabizi Synthetic Dataset Generator**

**🔍 Overview**

The **Arabizi Synthetic Dataset Generator** transforms English dialogue data into Lebanese Arabizi using a pipeline that combines:

* GPT-based translation
* Regex-based phonetic transliteration
* Orthographic variation generation

The goal is to produce high-quality synthetic data for training NLP models that understand Arabizi.

**✅ Features**

* Converts English prompts to Lebanese Arabizi
* Cleans and post-processes Arabizi with regex
* Generates 2–3 orthographic variants per sentence
* Saves structured dataset in JSON format
* Modular and extendable design

**🏗 Architecture**

English Text

↓

[GPT-based Arabizi Translation]

↓

[Regex-based Phonetic Correction]

↓

[Orthographic Variant Generation]

↓

**Structured Output (JSON)**

**🚀 Usage**

1. Place your input data in data/input\_dataset.json in this format:

[

{

"prompt": "How are you?",

"response": "I'm fine, thanks."

}

]

1. Run the pipeline:

python main.py

1. Output will be saved to data/output\_dataset.json:

{

"prompt\_en": "How are you?",

"response\_en": "I'm fine, thanks.",

"prompt\_arabizi": "kifak?",

"response\_arabizi": "mnih, merci",

"prompt\_variants": ["keefak?", "kifak?", "kefak?"],

"response\_variants": ["mnee7, masi", "mnih, mercee", "mnih, merci"]

}

**⚙️ Configuration**

You can customize rules and patterns:

* config/regex\_rules.json: regex mappings for correcting GPT output
* config/variant\_rules.json: mapping of common Arabizi patterns to alternatives

**📁 Folder Structure**

arabizi-synthetic-dataset-generator/

│

├── main.py

├── data/

│ ├── input\_dataset.json

│ └── output\_dataset.json

│

├── src/

│ ├── utils.py

│ ├── gpt\_wrapper.py

│ ├── transliteration.py

│ └── orthographic\_variants.py

│

├── config/

│ ├── regex\_rules.json

│ └── variant\_rules.json

│

├── .env

├── requirements.txt

└── README.md

**🧠 Core Modules**

**main.py**

* Entry point for running the full pipeline

**transliteration.py**

* Applies regex-based corrections (e.g., masi → merci, shukr → shoukran)

**orthographic\_variants.py**

* Generates alternative spellings using character rules

**utils.py**

* Load/save JSON files, logging, formatting helpers

**🧪 Examples**

**Input:**

{ "prompt": "Good morning", "response": "How are you today?" }

**Output:**

{

"prompt\_en": "Good morning",

"response\_en": "How are you today?",

"prompt\_arabizi": "sabah el kheir",

"response\_arabizi": "kifak lyom?",

"prompt\_variants": ["saba7 el kheir", "saba el khir"],

"response\_variants": ["keefak lyom?", "kefak alyom?"]

}

**📜 Scripts Overview (scripts/ Folder)**

The scripts/ folder contains the **core transformation pipeline**, broken down into modular Python files. Each script performs a specific stage of the pipeline and works closely with files in the data/ directory and helpers from the utils/ folder.

Below is a breakdown of each script and its role:

**🔹 preprocess.py**

* **Purpose:** Loads the raw English dialogue dataset from data/raw/test.csv and applies basic **text cleaning** and **pair extraction**.
* **Input:** data/raw/test.csv
* **Output:** Processed dialogue pairs saved to data/translated/preprocessed.csv
* **Key Outputs:** Structured format — prompt, response

**🔹 translate\_gpt.py**

* **Purpose:** Translates the cleaned English dialogues into **Lebanese Arabizi** using the OpenAI GPT model.
* **Input:** data/translated/preprocessed.csv
* **Output:** Translated file saved to data/translated/translated.csv
* **Uses Helper:** utils/gpt\_api.py – handles API calling and prompt generation

**🔹 postprocess\_regex.py**

* **Purpose:** Applies **regex-based phonetic corrections** to the GPT-translated Arabizi output to ensure consistency and accuracy.
* **Input:** data/translated/translated.csv
* **Output:** Corrected Arabizi file saved to data/correct/corrected.csv
* **Uses Helper:** utils/regex\_rules.py – loads and applies regex transformation rules

**🔹 generate\_variants.py**

* **Purpose:** Generates **2–3 orthographic spelling variants** for each Arabizi pair to simulate real-world diversity in spelling.
* **Input:** data/correct/corrected.csv
* **Output:** Final dataset with variants stored in JSON format at data/final/final.json
* **Uses Helper:** utils/variant\_generator.py – applies variant rules to generate spelling diversity

**🔹 main.py**

* **Purpose:** Acts as the **master controller** that runs the entire pipeline in order:
  1. Preprocess English
  2. Translate to Arabizi
  3. Apply regex correction
  4. Generate variants
  5. Save final dataset
* **Benefit:** Provides a one-click way to automate the complete synthetic dataset generation pipeline.

**📂 Data Folder Summary**

| **Folder** | **Description** |
| --- | --- |
| data/raw/ | Contains the original input English CSV/JSON dataset |
| data/translated/ | Stores preprocessed and GPT-translated Arabizi files |
| data/correct/ | Contains regex-corrected Arabizi outputs |
| data/final/ | Final JSON/JSONL dataset with variants included |