High-Performance Multilingual PDF Glossary Generator

Engine: Python 3.8+ (ReportLab, Pandas, Deep-Translator) **Architecture:** Linear Pipeline (Asset Acquisition → ETL Transformation → Vector Rendering) **Version:** 1.0.0

---

1. Executive Summary

The **Multilingual PDF Glossary Generator** is a specialized engineering solution designed to bridge the gap between raw terminological data and professional, print-ready typesetting across 30 global languages.

Standard text processors often fail when mixing complex scripts (e.g., Arabic, Chinese, and Hindi) on the same page, resulting in "Tofu" artifacts (□) or disconnected letters. This system bypasses the host operating system's font rendering, implementing a **Self-Contained Typography Engine** to guarantee binary reproducibility across Windows, Linux, and macOS.

1.1 Core Architecture: The "Safe-Render" Pipeline

The system operates on three strict architectural principles to ensure typographic fidelity.

Stage	Principle
Stage 0	**Asset Isolation**
Stage 1	**Metadata Injection**
Stage 2	**Algorithmic Shaping**

---

2. Technical Requirements

2.1 System Prerequisites

- **Operating System:** Windows 10/11, macOS Catalina+, or Linux (Ubuntu 20.04+).
- **Python Runtime:** Python 3.8 or higher (Strict requirement for `python-bidi` stability).
- **Storage:** ≈600MB free space (400MB reserved for Font Assets).

2.2 Dependency Stack

The system relies on a precise stack of libraries for ETL and rendering. Install strictly via pip:

```
pip install pandas openpyxl reportlab arabic-reshaper python-bidi deep-translato
```

Package	Target Version	Critical Role
`pandas`	`1.3.5+`	High-performance DataFrame IO.
`openpyxl`	`3.0.10+`	Excel styling & metadata injection.
`reportlab`	`3.6.12+`	PDF generation & Canvas drawing.
`arabic-reshaper`	`3.0.0+`	Contextual analysis (Ligatures).
`python-bidi`	`0.4.2+`	Unicode Bidirectional Algorithm.
`deep-translator`	`1.9.1+`	Google Translate API Wrapper.

---

3. Font Asset Architecture

This section mandates the physical layout required for the "Smart Font Engine". **Missing files may cause runtime errors.**

Ensure your project folder matches this exact structure. The system scans the `/fonts/` directory recursively.

```
/your_project/
  get_fonts.py              # [Stage 0] Asset Downloader
  script_translate.py       # [Stage 1] ETL & Translator
  script_glossary.py        # [Stage 2A] Vertical PDF Renderer
  script_table.py           # [Stage 2B] Panoramic PDF Renderer
  english.xlsx              # Source Data (User Provided)
  glossary.xlsx             # Generated Data (Intermediate)
  fonts/                    # MUST contain the specific files below

      NotoSansLiving-Regular.ttf    # Latin/Cyrillic (Fallback)
      NotoSans-Bold.ttf             # Headers/Titles
      NotoSansCJK.ttc               # CJK Super-Collection
      NotoSansArabic-Regular.ttf    # Arabic/Persian
      NotoNastaliqUrdu-Regular.ttf  # Urdu (Cascading Style)
      NotoSansDevanagari-Regular.ttf # Hindi/Marathi
      NotoSansBengali-Regular.ttf   # Bengali
      NotoSansGujarati-Regular.ttf  # Gujarati
      NotoSansTamil-Regular.ttf     # Tamil
      NotoSansTelugu-Regular.ttf    # Telugu
      NotoSansGurmukhi-Regular.ttf  # Punjabi
      NotoSansThai-Regular.ttf      # Thai
      NotoSansJavanese-Regular.ttf  # Javanese
```

To guarantee zero "Tofu" (   ) and perfect rendering in both Excel and PDF, you must download the specific font files listed below.

A. THE "CORE" FONTS (Latin, Cyrillic, Greek & Headers)

- Filenames: ”NotoSansLiving-Regular.ttf” AND ”NotoSans-Bold.ttf”
- Source: https://github.com/notofonts/notofonts.github.io/tree/main/megamerge
- Why:
  1. ”Regular” (Living): This specific ”Mega-Merge” version covers ~80% of languages (English, French, Russian, etc.) in a single file. It fixes rendering issues for Turkish (İ, ş) and Vietnamese (stacked diacritics) that standard Arial often breaks.
  2. ”Bold”: MANDATORY for the PDF generator. Without 'NotoSans-Bold.ttf', section headers (e.g., ”CATEGORY”) will fail to render, causing the script to crash or print blank headers.

B. THE ”CJK” SUPER-FONT (Chinese, Japanese, Korean)

- Filename: ”NotoSerifCJK.ttc” (or NotoSansCJK.ttc)
- Source: https://github.com/notofonts/noto-cjk
- Why:
  1. Scale: CJK languages require >65,000 glyphs. Standard fonts do not contain them all.
  2. Efficiency: The ”.ttc” (TrueType Collection) format bundles Simplified Chinese (SC), Traditional Chinese (TC), Japanese (JP), and Korean (KR) into one efficient file.
  3. Compatibility: This script is tuned to detect the ”TTC” collection. Using individual ”.otf” files may result in Excel failing to recognize the font family.

C. MIDDLE EASTERN (Right-to-Left Scripts)

- Filenames:
  1. ”NotoSansArabic-Regular.ttf” (Essential for Arabic, Persian, & Standard Urdu)
  2. ”NotoNastaliqUrdu-Regular.ttf” (Recommended for Urdu aesthetics)
- Source: https://www.google.com/get/noto/
- Why:
  1. Shaping: Arabic letters change shape based on position (Start/Middle/End). Standard fonts often break these ”ligatures,” leaving letters disconnected (e.g.,   instead of .(

  .2 Style: Urdu users prefer ”Nastaliq” (cascading style). If present, the script uses it; otherwise, it safely falls back to the standard Naskh style.

D. SOUTH ASIAN (Indic Scripts / Abugidas)

- Filenames:
  - ”NotoSansDevanagari-Regular.ttf” (Hindi, Marathi)
  - ”NotoSansBengali-Regular.ttf” (Bengali)
  - ”NotoSansGujarati-Regular.ttf” (Gujarati)
  - ”NotoSansTamil-Regular.ttf” (Tamil)
  - ”NotoSansTelugu-Regular.ttf” (Telugu)

    – ”NotoSansGurmukhi-Regular.ttf” (Western Punjabi)
- Source: https://github.com/notofonts/noto-fonts (Download the ”Phase 3” zip)
- Why:
    1. Complex Layout: These scripts use engines where vowels ”float” above, below, or wrap around consonants.
    2. Rendering: Without these specific fonts, vowels will detach from their consonants and render as dotted circles ( ) or meaningless boxes.

E. SOUTHEAST ASIAN

- Filenames: ”NotoSansThai-Regular.ttf”, ”NotoSansJavanese-Regular.ttf”
- Why: Thai tone marks must stack vertically at precise heights. Javanese is a rare historical script often completely missing from standard Windows/Mac systems.

---

4. Script Logic & Capabilities

4.1 Stage 0: Asset Acquisition (`get_fonts.py`)

- **Role:** Infrastructure Initialization.
- **Discovery Algorithm:** Implements ”Brute Force Discovery”. It iterates through a priority list of 5 potential repository structures (Mega-Merge, Main Hinted, Main Unhinted, Static Mirror) until a valid HTTP 200 stream is established.
- **Idempotency:** Checks for existing files before downloading to support CI/CD pipelines.
- **Timeout Handling:** Uses a specialized 120-second timeout for the 100MB+ CJK collection file.

4.2 Stage 1: ETL & Translation (`script_translate.py`)

- **Role:** Data Transformation.
- **Polite Throttling:** Implements `CHUNK_SIZE = 50` and `REQUEST_DELAY = 1.5s` to strictly adhere to API rate limits and prevent IP bans (HTTP 429).
- **Font Injection:** Iterates through Excel columns and applies `cell.font = Font(name="Noto Sans Arabic")` properties based on the detected language, ensuring the Excel file itself looks correct.

4.3 Stage 2: Vector Rendering Engines

The system offers two distinct rendering engines depending on the desired output format.

Stage 2A uses a vertical list format on a fixed A4 portrait page, displaying words with their descriptions in a dictionary style. It has a static page size of 21cm by 29.7cm and is suitable for flashcards or study guides.

Stage 2B presents information as a wide, panoramic table in landscape mode. The table layout is a grid with only words, and its width adjusts automatically depending on the number of columns.

---

5. Supported Language Matrix

The system currently supports typographic rendering for 30 distinct language identifiers.

ID	Language	Script Family	Primary Font Resource
1	**English**	Latin	`NotoSansLiving`
2	**Mandarin Chinese**	Hanzi (Simplified)	`NotoSansCJK (SC)`
3	**Hindi**	Devanagari	`NotoSansDevanagari`
4	**Spanish**	Latin	`NotoSansLiving`
5	**Portuguese**	Latin	`NotoSansLiving`
6	**Standard Arabic**	Arabic (Naskh)	`NotoSansArabic`
7	**Bengali**	Bengali	`NotoSansBengali`
8	**French**	Latin	`NotoSansLiving`
9	**Russian**	Cyrillic	`NotoSansLiving`
10	**Urdu**	Arabic (Nastaliq)	`NotoNastaliqUrdu`
11	**Indonesian**	Latin	`NotoSansLiving`
12	**German**	Latin	`NotoSansLiving`
13	**Japanese**	Kanji/Kana	`NotoSansCJK (JP)`
14	**Marathi**	Devanagari	`NotoSansDevanagari`
15	**Telugu**	Telugu	`NotoSansTelugu`
16	**Turkish**	Latin	`NotoSansLiving`
17	**Tamil**	Tamil	`NotoSansTamil`
18	**Yue Chinese**	Hanzi (Traditional)	`NotoSansCJK (TC)`
19	**Wu Chinese**	Hanzi (Simplified)	`NotoSansCJK (SC)`
20	**Korean**	Hangul	`NotoSansCJK (KR)`
21	**Vietnamese**	Latin (Stacked)	`NotoSansLiving`
22	**Hausa**	Latin (Pan-Nigerian)	`NotoSansLiving`
23	**Iranian Persian**	Arabic (Naskh)	`NotoSansArabic`
24	**Egyptian Arabic**	Arabic (Naskh)	`NotoSansArabic`
25	**Swahili**	Latin	`NotoSansLiving`
26	**Javanese**	Javanese	`NotoSansJavanese`
27	**Italian**	Latin	`NotoSansLiving`
28	**Western Punjabi**	Gurmukhi	`NotoSansGurmukhi`
29	**Gujarati**	Gujarati	`NotoSansGujarati`
30	**Thai**	Thai	`NotoSansThai`

---

6. Execution Guide

Follow this sequence to generate your documents.

Step 1: Initialize Assets

Run the downloader to verify and populate the `/fonts` directory.

```
python get_fonts.py
```

Step 2: Input Configuration

Create a file named `english.xlsx` in the root directory. It must contain the following columns:

- `Category` (Optional, for grouping)
- `English_word`
- `English_descr`

Step 3: ETL & Translation

Run the translator to generate the intermediate `glossary.xlsx`.

```
python script_translate.py
# Output: glossary.xlsx
```

Step 4: Final Rendering

Choose your desired output format.

Option A: Vertical Glossary Generates a document suitable for reading descriptions.

```
python script_glossary.py
# Output: glossary.pdf
```

Option B: Panoramic Table Generates a wide comparison table.

```
python script_table.py
# Output: table.pdf
```