AI DOCUMENT TRANSLATOR (ENGINEERING EDITION) - GUI

Version: 1.0 (Final Production / Batch Support / Anti-Leak / Deep Context)

---

1. SYSTEM ARCHITECTURE & PHILOSOPHY

This application represents a paradigm shift from standard machine translation. It is architected as a **"Document Reconstruction Engine"** rather than a linear text processor.

[THE "DIGITAL TWIN" METHODOLOGY]

Standard translators (Google/DeepL) often strip formatting to process raw text, destroying the document's layout. This engine employs a **"Digital Twin"** approach:

- **a) ATOMIC DISSECTION:** The Python-docx library is leveraged to surgically parse the .docx file into its atomic XML constituents (Paragraphs > Runs > Tables > Cells).
- **b) DNA EXTRACTION:** Before translation, the engine extracts the "Visual DNA" of every text segment. This includes Font Family, Size, RGB Color, Highlight (Background), Bold/Italic/Underline flags, and complex XML Numbering properties.
- **c) COMPILER INJECTION:** Text is encapsulated with semantic tags (e.g., "{b}Text{/b}") and injected into a local Large Language Model (LLM) via Ollama. The LLM is treated not as a chatbot, but as a deterministic "Text Compiler" with temperature=0.0.
- **d) HALLUCINATION SANITIZATION:** The output passes through a rigorous "Firewall" to strip AI meta-commentary, ensuring only pure translation data remains.
- **e) SURGICAL RECONSTRUCTION:** The document is rebuilt from zero. The translated text is fused with the original "Visual DNA" and injected back into the XML skeleton, preserving headers, footers, and complex table layouts.

---

2. CORE CAPABILITIES

[A] BATCH PROCESSING ORCHESTRATOR

- **CONCURRENT QUEUE MANAGEMENT:** Capable of ingesting unlimited file paths via the GUI. The engine isolates file operations in a worker thread to prevent GUI freezing ("Not Responding").
- **PREDICTIVE WORKLOAD CALCULATION:** Performs a "Global Pre-Scan" algorithm that traverses every section (Body, Header, Footer, Tables) of every file to count total processable blocks. This ensures the progress bar represents real-time atomic progress, not just "File 1 of 5".
- **FAULT TOLERANCE:** If a single file is corrupt (XML errors), the engine logs the failure and automatically proceeds to the next file in the queue without crashing the batch.

[B] ANTI-LEAK FIREWALL (MULTI-LAYERED SANITIZATION)

- **THE PROBLEM:** LLMs are prone to "leaking" internal monologue (e.g., "Here is the translation," "I cannot translate names," or internal JSON markers).
- **THE SOLUTION:** A heuristic "Regex Firewall" that scrubs the output in five passes:

1. **CONVERSATIONAL FILLER REMOVAL:** Strips introductory phrases ("Sure,", "Note:", "Likely means").
2. **TAG REPAIR:** Detects and fixes broken formatting tags often mangled by AI tokenizers (e.g., repairing "{/ b}" to "{/b}" or "{tab}" to valid XML tab stops).
3. **MARKDOWN STRIPPING:** Removes unrequested Markdown artifacts (**bold**, ## Header) that LLMs habitually add.
4. **ENTITY DECODING:** Converts HTML entities (, <) back to Word-compatible unicode.
5. **INVISIBLE CLEANUP:** Purges Unicode Category 'C' (Control characters) and JSON artifacts (leftover braces or "1.1.1" key prefixes) that corrupt .docx XML.

[C] REFLEXION & SELF-CORRECTION (ACTOR-CRITIC LOOP)

- **STABILITY ENGINE:** The system implements an "Actor-Critic" architecture.
- **PASS 1 (THE ACTOR):** The primary model attempts the translation.
- **PASS 2 (THE CRITIC):** An internal logic check analyzes the output for "Chattiness" (length ratio > 1.8x original) or "Leakage" (detection of forbidden phrases).
- **PANIC MODE:** If the Critic rejects the output, the engine engages "Panic Mode." It discards the complex prompt and retries using a simpli-

fied, high-restriction prompt to force a raw literal translation, ensuring no data is lost even in difficult segments.

[D] XML STRUCTURAL & DATA PRESERVATION

- **NUMBERING INTEGRITY (`numPr`):** Engineering specifications rely on strict hierarchy (1.0, 1.1, 1.1.2). The engine deep-copies the XML `numPr` (Numbering Properties) from the source and re-attaches it to the translated paragraph. This keeps lists "live" and clickable in Word.
- **DATA PROTECTION:** The engine utilizes Regex pattern matching to identify "Pure Numeric Blocks" (e.g., "500", "10/2023", "Ø 15mm"). These blocks are bypassed entirely to prevent the AI from hallucinating numbers (e.g., changing "100" to "one hundred" or "100.0").

[E] CONSULTANT MODE (DUAL-LLM TERMINOLOGY AUDIT)

- **THE "TWO-PASS" SYSTEM:** Optionally engages a Secondary Expert Model (e.g., trained on ISO Standards).
- **WORKFLOW:**

1. The Primary Model (e.g., Gemma 2) drafts the translation.
2. The Consultant Model reviews the draft specifically for "Technical Accuracy" (Units, Material Names, ISO Codes).
3. If the Consultant detects an error (e.g., "Concrete Screed" vs "Cement Paste"), it patches the translation via a strict JSON protocol.

- **EFFICIENCY:** Includes a length-heuristic filter to only trigger the Consultant on complex technical sentences, skipping simple headers to save inference time.

[F] COMPILER-STRICT PROMPTING ARCHITECTURE

- **SYSTEM PROMPT ENGINEERING:** The LLM is not prompted as a "Helpful Assistant." It is prompted as a "Data Processing Engine" with strict Opcodes.
- **FORMAT ENFORCEMENT:** Enforces a JSON-only output structure (`{"t": "translated_string"}`). This containerization prevents the translation from bleeding into surrounding text and allows the program to programmatically validate success (checking for valid JSON parsing).
- **TAG INVARIANCE:** Instructions explicitly forbid the translation or removal of injected tags ({b}, {i}, {tab}), treating them as "Memory Pointers" that must be returned intact.

---

3. INSTALLATION & CONFIGURATION GUIDE

STEP A: INSTALLING OLLAMA (THE AI RUNTIME)

Ollama is the engine that allows you to run powerful AI models locally without internet.

1. **Download:**

- Windows: Visit [https://ollama.com/download/windows%5D(https://ollama.com/download/windows) and run the .exe installer.
- macOS: Visit [https://ollama.com/download/mac%5D(https://ollama.com/download/mac) and download the disk image.
- Linux: Run `curl -fsSL https://ollama.com/install.sh | sh` in your terminal.

2. **Verify:** Open terminal/cmd and run `ollama --version`.

STEP B: CHOOSING & INSTALLING MODELS

The GUI detects installed models. You must "pull" (download) them via terminal first.

OPTION 1: THE SPECIALIST: TranslateGemma (Recommended)

- **Description:** A fine-tuned version of Google's Gemma 2 (27B), specifically optimized for high-fidelity document translation. It excels at adhering to "Compiler-Strict" instructions, ensuring formatting tags (like {b}, {tab}) are preserved while translating complex engineering syntax without dropping data.
- **Pros:** Best balance of speed, formatting adherence, and technical accuracy.
- **Hardware:** 16GB+ RAM (Required for 27b version).
- **Link:** [https://ollama.com/library/translategemma%5D(https://ollama.com/library/translategemma)
- **> Command:** `ollama pull translategemma:27b`

OPTION 2: THE POWERHOUSE: GPT-OSS 20B

- **Description:** A massive open-source model designed to rival commercial APIs like DeepL. Its large parameter count allows it to grasp subtle nuances in "Technical Prose" and ambiguous sentence structures often found in legal/contractual specs.
- **Pros:** Superior semantic understanding. Less likely to misinterpret context.
- **Cons:** Slower generation. Requires a powerful computer (16GB+ RAM or 12GB+ VRAM GPU).
- **Link:** [https://ollama.com/library/gpt-oss%5D(https://ollama.com/library/gpt-oss)

- > **Command:** `ollama pull gpt-oss:20b`

OPTION 3: THE POLYGLOT: Qwen-MT

- **Description:** Alibaba's model specialized specifically for translation tasks. Trained on massive multilingual corpora, it is the gold standard for Asian languages (Chinese/Japanese/Korean) and handles technical manuals with high precision.
- **Pros:** Exceptional for CJK languages. Very stable output structure.
- **Link:** [https://ollama.com/library/qwen-mt%5D(https://ollama.com/library/qwen-mt)
- > **Command:** `ollama pull qwen-mt`

OPTION 4: THE SPEEDSTER: Qwen 2.5 (7B)

- **Description:** A highly efficient general-purpose model. While not exclusively a translator, its instruction-following capabilities are top-tier for its size. Perfect for drafting or when running on legacy hardware (older laptops).
- **Pros:** Extremely fast. Low resource usage.
- **Link:** [https://ollama.com/library/qwen2.5%5D(https://ollama.com/library/qwen2.5)
- > **Command:** `ollama pull qwen2.5`

--- SECONDARY CONSULTANT MODELS (OPTIONAL) --- These models are used to double-check terminology but not for the main translation.

- **[CONSULTANT A] ENGINEERING MANUALS**
- Description: Specialized in the tone and vocabulary of technical manuals.
- Link: [https://ollama.com/ALIENTELLIGENCE/engineeringtechnicalmanuals%5D(https://ollama.com/ALIENTELLIGENCE/engineeringtechnicalmanuals)
- > **Command:** `ollama pull ALIENTELLIGENCE/engineeringtechnicalmanuals`
- **[CONSULTANT B] STRUCTURAL LLAMA**
- Description: Deep knowledge of structural engineering terms (Concrete, Beams, Loads).
- Link: [https://ollama.com/joreilly86/structural_llama%5D(https://ollama.com/joreilly86/structural_llama)
- > **Command:** `ollama pull joreilly86/structural_llama`

STEP C: PYTHON ENVIRONMENT & COMPILING TO EXECUTABLE

You can run as a script or compile to a standalone .exe to avoid installing Python on other machines.

OPTION 1: RUNNING AS A SCRIPT (DEV MODE)

1. Install Python 3.8+: [https://python.org%5D(https://python.org)
2. Create Virtual Env (Recommended):

- Windows: `python -m venv venv` then `venv\Scripts\activate`
- Mac/Linux: `python3 -m venv venv` then `source venv/bin/activate`

3. Install Libs: `pip install ollama python-docx langdetect pyinstaller`

OPTION 2: COMPILING TO OPTIMIZED EXECUTABLE (DEPLOYMENT MODE) We use PyInstaller with specific exclusions to remove bloat (numpy, pandas, etc.) reducing size by ~60MB.

1. Activate your virtual environment (see above).
2. Run the compilation command for your OS:

[WINDOWS BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean ^
  --name "translator_gui" ^
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas ^
  --exclude-module scipy --exclude-module IPython --exclude-module pytest ^
  translator_gui.py
```

[macOS BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean \
  --name "translator_gui" \
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas \
  --exclude-module scipy --exclude-module IPython --exclude-module pytest \
  translator_gui.py
```

[LINUX BUILD]

```
pyinstaller --noconfirm --onefile --windowed --clean \
  --name "ai_translator_eng_v5" \
  --exclude-module matplotlib --exclude-module numpy --exclude-module pandas \
  --exclude-module scipy \
  translator_gui.py
```

3. **Result:** The standalone app will be in the `dist/` folder.

---

4. HOW TO RUN

1. Run `python translator_gui.py` (or open the compiled Exe).
2. Click **"Browse"** and select one or multiple .docx files.
3. Choose your **Source/Target** languages via the Dropdowns (e.g., English UK).
4. Select your **Primary Model**.
5. Click **"Start Translation"**.
