AI in Society and Public Services

Session 4: GenAI for Production - File Generation & Workflows

Mário Antunes

January 30, 2026

Universidade de Aveiro

Part I: The “Text-First” Paradigm

Why LLMs “Speak” Text, Not Binary

- **The Fundamental Limit:** LLMs predict tokens (sub-words). They do not “draw” pixels or “compile” binaries natively.
- **Text vs. Binary:**
- **Text (ASCII/UTF-8):** Semantic, readable, token-efficient. (e.g., `.md`, `.csv`, `.json`).
- **Binary:** Opaque, position-dependent, token-heavy. (e.g., `.docx`, `.pdf`, `.xlsx`).

The Strategy: *Always ask the LLM to generate the* ***Source Code*** *of the document, then use a deterministic tool (compiler/converter) to build the final product.*

The Ideal File Stack for GenAI

- **Documentation:** Markdown (`.md`) The universal source for docs.
- **Data:** CSV (`.csv`) The universal source for spreadsheets.
- **Configuration:** YAML/JSON The universal source for settings.
- **Web/Code:** HTML/CSS/Python The universal source for applications.
- **Why?** These formats are **structured**, **human-readable**, and **machine-parseable**.

Part II: Deep Dive into Ideal Formats

Markdown (.md) - The King of LLM Formats

- **What is it?** A lightweight markup language that adds formatting elements to plain text.
- **Why LLMs love it:**
- **Low Token Cost:** `# Title` uses fewer tokens than XML's `<title>Title</title>`.
- **Hierarchical:** LLMs understand structure via Headers (#, ##, ###).
- **Use Cases:** Reports, Slides, Resumes, Documentation, Emails.
- **Example Prompt:** "Generate a Project Charter in Markdown. Use H1 for the Title, H2 for sections like 'Scope' and 'Risks', and bolding for key stakeholders."

CSV (Comma Separated Values) - Data Interoperability

- **Structure:** Text where columns are separated by commas and rows by newlines.
- **Why not ask for Excel (.xlsx)?** LLMs cannot generate the binary XML compression of an Excel file reliably. They *can* generate perfect text.
- **Workflow:**

1. LLM Generates CSV text.
2. User saves as `data.csv`.
3. User opens in Excel/Google Sheets (Auto-formatted).
   Example Prompt: *"Create a dummy dataset of 50 customers with columns: ID, Name, Email, Purchase_Value, and Date. Output strictly as CSV code block."*

Structured Configs (JSON / YAML / XML)

- **JSON:** Strict syntax. Great for web APIs. (e.g., `{ "name": "John" }`).
- **YAML:** Indentation-based. Great for human readability and configuration (e.g., Kubernetes, CI/CD).
- **XML:** Verbose. Used in legacy systems or specific industries (e.g., Invoice standards).
- **Tip:** If you need the LLM to fill a form, ask it to "Generate a JSON object" rather than a paragraph of text. It forces the model to categorize information.

Coding Formats (Python, HTML, CSS)

- **The Concept:** Don't ask for a "Web Page." Ask for "A single HTML file containing embedded CSS and JS."
- **Python:** The glue code. Instead of asking the LLM to "Calculate the average of this massive list," ask it to "Write a Python script to calculate the average."
- **Reason:** LLMs are bad at math (hallucination risk) but good at writing code *that* does math (logic).

Part III: Prompt Engineering for File Generation

The “Role & Constraints” Framework

- **Context:** “You are a Senior Data Engineer.”
- **Task:** “Convert the following raw text report into a structured JSON file.”
- **Constraint (Crucial):** “Output **ONLY** the code block. Do not write conversational filler like ‘Here is the file you asked for’. Start immediately with `{`.”
- **Why?** Allows for direct copy-pasting or programmatic extraction.

Handling Input Documents

- **Scenario:** You have a PDF report and need a CSV summary.
- **The “Context Window” Prompt:**

1. **Instruction:** “Review the text below. Extract all dates and financial figures.”
2. **Format:** “Present the output as a Markdown Table.”
3. **Data:** [Paste Text content here].
   Tip: *For massive files, ask the LLM to write a Python script to parse the file instead of pasting the text itself.*

Tips & Tricks for Clean Generation

- **Iterative Refinement:** “That CSV is good, but the date format is wrong. Rewrite it using ISO 8601 (YYYY-MM-DD).”
- **Separators:** Use clear delimiters (e.g., """ or –––) to separate your instructions from the data you want processed.
- **One-Shot Learning:** Provide *one* example of the output structure you want.
- *User:* “Format like this: Name | Role | Age”
- *LLM:* “Understood. Here is the rest...”

Part IV: The Power Tool - Pandoc

What is Pandoc?

- **Definition:** The “Universal Document Converter.” A command-line tool that converts files from one markup format into another.
- **The Workflow:**

1. **Ideation:** LLM generates content (Markdown).
2. **Conversion:** Pandoc converts Markdown DOCX / PDF / PPTX.
3. **Result:** Professional, styled documents without manual formatting.

Installation i

- **Windows:**
- Command Line: `winget install JohnMacFarlane.Pandoc`
- GUI Option: **PandocGUI** or **PanWriter** (Open source editors that run Pandoc in the background).

Installation ii

- **macOS:** `brew install pandoc` (via Homebrew).
- **Linux:** `sudo apt-get install pandoc`
- **Check installation:** Open terminal and type `pandoc --version`.

Basic Conversion (Markdown to Word)

- **Scenario:** You generated a report in Markdown, but your boss wants a Word Doc.
- **Command:** `pandoc report.md -o report.docx`
- **Magic:** It converts headers to Word Styles (Heading 1), lists to Word bullets, and tables to Word tables. No manual fixing required.

Advanced Word conversion (Using Reference Templates)

- **The Problem:** The default Pandoc Word style is plain.
- **The Solution:** Use a `--reference-doc`.

1. Create a Word doc (`template.docx`). Modify “Normal” and “Heading 1” styles with your corporate colors/fonts. Save it.
2. **Command:** `pandoc input.md --reference-doc=template.docx -o final_report.docx`
   Result: *A branded Word document generated instantly from plain text.*

Markdown to PDF

- **Method A (Direct):** Uses a LaTeX engine (requires installing MiKTeX or TeX Live).
  - `pandoc input.md -o output.pdf`
- **Method B (via Word):** Convert to DOCX first, then Save as PDF (Easier for non-technical users).
- **Method C (HTML to PDF):** Convert to HTML, then Print to PDF in Chrome.
  - `pandoc input.md -o output.html`

Markdown to Slides (PowerPoint & Web) i

- **PowerPoint (.pptx):**
- Each `## Heading` becomes a new slide title.
- **Command:** `pandoc slides.md -o presentation.pptx`

Markdown to Slides (PowerPoint & Web) ii

- **Web Slides (Reveal.js):**
- Creates interactive HTML slides.
- **Command:** `pandoc -t revealjs -s -o slides.html slides.md`

Pandoc for Windows (GUI Options)

- **Why use a GUI?** Command lines can be intimidating.
- **Tool: PanWriter**
- *Features:* Split screen (Markdown on left, Preview on right).
- *Export:* One-click export to PDF/DOCX using Pandoc under the hood.
- **Tool: PandocGUI**
- Simple drag-and-drop interface to select input file and desired output format.

Part V: Hands-On Workflows

Workflow 1 - The “Meeting Minutes” Pipeline

1. **Input:** Rough notes or transcript.
2. **GenAI Task:** “Clean these notes into a structured Markdown format with sections: Attendees, Decisions, Action Items.”
3. **Process:** User saves text as `minutes.md`.
4. **Pandoc:** `pandoc minutes.md --reference-doc=company_template.docx -o Minutes.docx`
5. **Output:** Corporate-ready Word document in 30 seconds.

Workflow 2 - The “Data Analysis” Pipeline

1. **Input:** Messy PDF bank statement.
2. **GenAI Task:** “Extract all transaction rows into a CSV format. Columns: Date, Description, Amount.”
3. **Process:** User saves text as `finance.csv`.
4. **Excel:** Open `finance.csv`.
5. **Output:** Analysis-ready spreadsheet.

Workflow 3 - The “Professor’s Slide Deck”

1. **GenAI Task:** “Generate a lecture outline on Photosynthesis using Markdown. Use H2 for Slide Titles and Bullet points for content.”
2. **Process:** Save as `lecture.md`.
3. **Pandoc:** `pandoc lecture.md -o lecture.pptx`
4. **PowerPoint:** Open file and click “Designer” pane to auto-style the slides.

Summary & Best Practices

- **Think in Text:** If you can write it in Notepad, GenAI can master it.
- **Separate Content from Style:** Let the LLM handle the *content* (Markdown). Let Pandoc handle the *style* (DOCX/PDF).
- **Validate:** Always check the generated code/CSV before converting.
- **Next Steps:** Install Pandoc today and try converting your first Markdown file.