Automation of
AI Translations for Authors

Christine & Alan Tipper

15 July 2026

Email cgt@christinetipper.com
Web www.christinetipper.com

A Short Presentation for writers

Based on practical experience

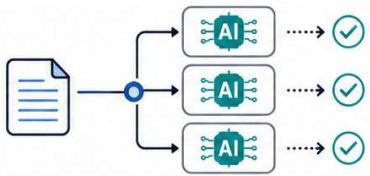
Why Automate?

Key benefits for AI-assisted book translation workflows

1 Remove Manual Cut / Paste

- Avoid repetitive copying between tools
- Reduce formatting mistakes and missed text
- Save time on every chapter

2 Parallel Execution of AI Requests



- Process multiple chapters simultaneously
- Increase throughput dramatically
- Shorten turnaround time for large books

3 Direct Write of Results to Disk

- Save outputs automatically to the correct folders
- Keep clear audit trails and file versions
- Eliminate manual file handling

Overall Impact

- Faster workflow
- Fewer human errors
- Better scalability
- More consistent outputs

We have concentrated on option #1

Automating Book Translation

Two Powerful Approaches – Which One Is Right for You?

OPTION 1: CONCURRENT API PIPELINE

Programmatic workflow using the Claude API

Chapters Input → API (Concurrent API Calls) → Automated Processing → Outputs, Logs & Reports

KEY STRENGTHS

- ✓ Highest throughput and scalability
- ✓ Best cost control (token tracking, caching, model routing)
- ✓ Fully automated, repeatable, and reliable
- ✓ Precise rate-limit, retry, and error handling
- ✓ Ideal for high volume and multi-book workflows

CONSIDERATIONS

- Requires coding and technical setup
- More time to build and maintain
- Less flexible during early workflow design
- Editorial judgment still needs human oversight

Best for production scale

Maximum speed, automation, and cost efficiency once built.

VS

OPTION 2: MULTIPLE AGENTS (CLAUDE CODE)

Agent team with specialized roles

Translation Agent · Bilingual QA Agent · Line Editor Agent · Terminology Agent · Continuity Agent

KEY STRENGTHS

- ✓ Easy to use – natural language instructions, no coding
- ✓ Role separation enables stronger quality and editorial review
- ✓ Flexible and iterative during workflow development
- ✓ Better for complex judgment and consistency decisions
- ✓ Faster to start and adapt

CONSIDERATIONS

- Less predictable throughput
- Higher token usage and weaker cost control
- More manual supervision required
- Harder to scale across many books/languages

Best for workflow design and supervision

Ideal for developing the process, handling tricky chapters, and ensuring high-quality results.

AT A GLANCE

FACTOR	CONCURRENT API PIPELINE	MULTIPLE AGENTS (CLAUDE CODE)
Cost Control	★★★★★	★★★☆☆
Throughput	★★★★★	★★★★☆
Ease of Use	★★☆☆☆	★★★★★
Repeatability	★★★★★	★★★☆☆
Editorial Flexibility	★★★☆☆	★★★★★
Setup Effort	★★☆☆☆	★★★★☆

TYPICAL WORKFLOW (BOTH APPROACHES)

1. Style Guide & Glossary → 2. Translation → 3. Bilingual Accuracy Check → 4. Line Editing → 5. Whole-Novel Consistency Audit → Final Translated Novel

BOTH APPROACHES DELIVER

- ✓ Scalable chapter-level processing
- ✓ Bilingual accuracy and quality control
- ✓ Consistent terminology and style
- ✓ Audit trails and change logs
- ✓ Publication-ready translations

– API – Applications Programming Interface

Workflow: CONCURRENT API PIPELINE

- AI Translations for Authors
  - Style guide
  - Baseline translation
  - Bilingual review
  - Line edit
  - Final human proofread

- No change to the basic workflow BUT FOR EACH PASS
  - Auto-generate the AI prompts
  - Submit prompt(s) to AI using the Claude API instead of web chat
  - Use concurrent API jobs
  - Separate the AI response into notes and target text
  - Merge the responses back into a single target document plus notes
- Create a batch job to sequence these tasks automatically

- Utilise the Python language to manipulate the files and submit to AI.

Evolution of the Concurrent API Process

- Manual web interface → Easy to do but time consuming (weeks)
- Python prompt generation → fairly easy and worth doing even for web chat use (days)
- API automation → Complicated but worthwhile to get the big gains (hours)
- Fully automated pipeline → the "full nine yards" (~ 20 minutes for 80,000 word novel)

- Keeping to the original process but automating in small steps allows manual intervention at any stage and all intermediate results are preserved.
- API limits only tokens/minute not total tokens so no hard cut off and you don't need expensive subscriptions.

Performance Improvements on 6 Books (English → Spanish, English → German)

- Book 1: 10 days using the original workflow (English → Spanish)
- Book 2: ~1 day, ~£9 API charge. First try at using the API (English → Spanish)
- Book 3: ~3 hours Using concurrent API chapter translation (concurrency:6 chapters)
- Books 4–5: ~1 hour using the fully automated pipeline (English → Spanish)
- Bottleneck: API rate limits
    - 90,000 tokens/minute when we did this – limits the concurrency
    - It's now increased to 1M/minute
- Extending the concurrency with the new API rate limits
- A new 60,000 word novel → 25 minutes for passes 2,3,4 into Spanish
    - Concurrency: 13 chapters
- Translate Book 1 above into German → 19 minutes for passes 2,3,4 into German
    - Concurrency: 39 chapters (the whole book in one go !)

Practical Lessons

- Keep the detailed logs for late-stage review
- Retain intermediate files (can go back and edit if needed)
- Human proofreading still required
- Tune concurrency carefully (we found 6 chapters at a time worked well when rate limit was 90k tokens/minute)
- With 1 Million tokens/min it is possible to do a whole book in one concurrency run.
- Beware Claude not behaving as expected
  - On the German translation Claude did not always preserve the markdown '#' in the chapter titles (4 cases out of 39)
  - This can mess up the python splitting tools – need to check Claude outputs carefully

Want to try it?

- Our automation scripts used for a 5 book series are publicly available on github (subject to opensource license)
- No warranty etc and you will need to customize them for your use
- You will need to install Python and a few other (free) utilities
- You will need to set up a Claude platform account for the pay per use part
- You will need to add your own style guide, prompt templates and markdown manuscript
- It probably needs some more robust error handling!
- https://github.com/Alantipper/Translation-Automation/archive/refs/heads/main.zip

The Claude Code Pipeline

- Need to install Claude Code and link it to your API Key
- Works with plain English Prompts i.e no code!
- Can handle concurrency and writing results back to disk
- Slower than Concurrent API
  - 25 minutes to do 3 passes of 3 chapter novella to Spanish
  - The Concurrent API process did this in 9 minutes

Example Claude Code Dialogue

concurrently translate each chapter of @manuscript/Novella.md using the style guide @01Style/styleguide.txt and put the translated chapters in folder 02Translate.

⎿ Read manuscript/Novella.md (530 lines)
⎿ Read 01Style/styleguide.txt (244 lines)

Read 1 file, listed 3 directories (ctrl+o to expand)

● I've confirmed the manuscript has 3 chapters (lines 1–127, 128–300, 301–530) and reviewed both the style guide and the project's documented translation pipeline. There's no existing scripts/templates setup in this project, so I'll translate each chapter directly and write the results into 02Translate/.

Launching three parallel agents now, one per chapter, each following the style guide precisely….

The Claude Code Pipeline (2)

- The prompt opposite created 6 translation agents working concurrently across 46 chapters
- Took ~ 1hour to complete
- API charge ~ $10
- Estimate 3-4 hours and $40 for full 3 pass automation
- Slower than Concurrent API
  - The Concurrent API process did this in 1 hour and cost $14

Example Claude Code Prompt

Translate all chapters of @"manuscript/murder at the match.md" chapter by chapter in concurrent groups of 6 chapters using the style guide in @01Stle/styleguide.txt and translation template in @scripts/Translation_template.txt. Send the translated text of each chapter to individual files in folder 02translate along with the translator notes in separate notes files.

Confirms the ability of Claude Code to create multi-agent translations with no user code required

Claude Code vs Coworker

- Coworker has a better user interface so is easier to use
- Coworker only supports subscription so pay as you go API is not available
- Coworker has less control over local files
- BUT
- It is probably the easiest non-technical route to concurrent translation

Conclusion

- Original *AI Translations for Authors* method took us ~ 2 weeks to complete a novel.
- Concurrent processing dramatically reduces this time
- Claude code can do a complete 3 pass translation in around 4 hours with no need for user code
- Claude Coworker should give similar results with an easier to use interface
- Fully optimised Concurrent API can do this in 20-30 minutes at much lower cost but with added user complexity

Demonstration of the Concurrent API method

- Translation of a 5000 word English Novella to European Spanish
- Small enough to see the method and execute quickly