INK: for Narrative Knowledge [Development Track]

David Liang
dlian4@illinois.edu
University of Illinois Urbana-Champaign
Urbana, Illinois, USA

Abstract

In the modern world, web novels bring a distinct set of challenges to LLMs and semantic analysis. Anything from requiring aggregation from scattered sources, multilingual translation with narrative fidelity, and even the need for deep semantic understanding beyond basic keyword search all pose significant difficulties in working with web novels. INK (INK for Narrative Knowledge) is an infrastructure/project designed to meet those challenges head-on. The current implementation, **InkScribe**, is a Python-based CLI framework built to streamline and automate the end-to-end pipeline for working with web novels. However, it is only one component of the infrastructure. Due to over-zealous ambition, my work is scattered across many prototypes created during the empirical testing of many algorithms and methodologies to solve various problem that occurred over the development of INK.

InkScribe handles several core tasks, providing a modular, extensible system for fetching webnovels from a range of sources, translating chapters using LLMs with an emphasis on maintaining consistent character and terminology usage (context), and supporting sophisticated semantic annotation and retrieval. This includes a plot-based search workflow.

In other prototypes, INK implements, enhances, and combines several cutting-edge NLP techniques such as: zero-shot independent tagging via GLiNER [8], advanced plot structure analysis and dependent tagging drawing on GraphRAG [2], HyDE [4], and FLARE [5], and an original approach to dynamic glossary creation inspired by PageRank to ensure translation consistency throughout a novel's run. A local web interface brings the system together, making it possible to read, review, and interact with annotated content in-browser.

This paper presents the motivation and the architectural foundations of INK. Unfortunately, due to time-constraints (as well as monetary-constraints), I was unable to go into as much detail as I would have liked in the evaluation of my ideas, actual screenshots of workflows (very expensive and time-consuming), and complete source code publication (only some components are implemented into InkScribe). Due to this, we focus on the usage patterns, system components, and practical instructions of Ink at a higher level, only going into detail for core pipelines. Because of the project's scale and constraints of a single-developer over a semester, full quantitiative benchmarking is still being developed and current results are a direction for future work.

1 Introduction

The proliferation of web novels has created a vast repository of narrative content, often contained within specific platforms and languages. For readers, translators, authors, and editors interested in exploring works beyond their native language, searching for specific plot elements, or ensuring translation consistency across millions of words, existing tools can be incomplete or inadequate. Manually fetching, translating, organizing, and deeply analyzing such content is time-consuming, error-prone, and often fails to capture the nuanced semantic fabric of these narratives. INK aims to address these significant "pain points." Its current primary software implementation is InkScribe, an integrated, open-source software toolkit designed for individuals to manage and interact with their web novel reading and processing experience. The primary motivation behind INK is to revolutionize how users engage with long-form, multilingual serialized web novels by automating laborious tasks and leveraging advanced NLP techniques for deeper comprehension, consistent translation, and sophisticated discovery. The novelty of the INK project, as realized through InkScribe, lies in its comprehensive, integrated approach to the entire lifecycle of web novel interaction, and particularly in its application and adaptation of cutting-edge research in areas like zero-shot tagging using GLiNER [8], graph-based analysis inspired by GraphRAG [2], and hypothetical document generation with HyDE [4] for enhanced semantic understanding and search. This is for the tagging component of INK.

InkScribe, as the current implementation of the INK vision, is a Python application designed to:

- Fetch novel metadata and chapter content from supported web novel sites, navigating various security measures.
- Translate fetched content into a target language using LLMs, with a strong emphasis on contextual history and dynamic glossary generation for narrative consistency.
- Build a local semantic search index, enabling users to search for novels or specific plot points using natural language queries enhanced by HyDE [4].
- Provide a simple local web interface for reading downloaded and translated novels.

The primary users are individuals who wish to create a personal, translated, and deeply searchable archive of web novels. INK, through InkScribe, aims to empower these users by automating laborious tasks and leveraging modern NLP techniques for better comprehension and discovery. It also serves as a valuable tool for language learners, hobbyist translators, and potentially researchers studying narrative structures. However, the complete implementation should be a software as a service due to the active costs of using LLMs intensively. Each of the components that are part of this project should be implemented as some sort of microservice.

In this report, I describe the InkScribe system as a realization of the INK project's goals. Section 2 details its architecture, major functions, and key implementation aspects, including how principles from recent research papers are applied. Section 3 serves as a user guide for InkScribe. Section 4 discusses its evaluation methodology and presents findings. Finally, Section 6 summarizes the project, its limitations, and potential future directions for INK.

2 System Description

INK's current software artifact, InkScribe, is created as a command-line first tool with an optional web UI for content consumption. It integrates web scraping, LLM-powered translation and text generation, advanced semantic tagging, and local vector database search. The system is designed to be iterative and scalable to handle the large volume and continuous generation of web novel content, aligning with the broader vision of INK.

2.1 Intended Users

InkScribe, as part of the INK project, is primarily designed for:

- **Readers of Web Novels:** Individuals who follow novels on supported source websites and wish to maintain an offline, translated, and deeply searchable archive.
- **Language Learners:** Users who might use translated novel content as reading material to aid in language acquisition, benefiting from consistent terminology.
- **Hobbyist Translators/Archivists:** Individuals interested in experimenting with LLM translation for long-form narratives and building a personal, semantically rich library.
- **Authors/Editors (Potentially):** Could leverage semantic analysis features for plot consistency checks or thematic exploration.
- **Researchers (indirectly):** By facilitating the collection and structured processing of novel data, InkScribe can be a preliminary tool for researchers studying narrative structures or cross-lingual literary analysis.

The tool is geared towards personal use, managing content locally on the user's machine, with the INK vision encompassing future collaborative enhancements.

2.2 Major Functions

InkScribe provides its functionalities through a Typer-based CLI and a FastAPI-based web application.

2.2.1 Command-Line Interface (CLI). The `inkscribe` CLI is the primary way to interact with the system:

- **Novel Fetching (`inkscribe fetch`):**
  - `fetch metadata <URL>`: Scrapes and saves novel metadata.
  - `fetch all <URL>`: Scrapes metadata and all chapter content.
- **Content Translation (`inkscribe translate`):**
  - `translate metadata <novel_name>`: Translates title and synopsis.
  - `translate chapters <novel_name>`: Translates chapters with contextual history and glossary support.
- **Semantic Analysis & Search (`inkscribe search` and related tagging commands):**
  - Commands for generating independent, dependent, and meta tags (details in Implementation).
  - `search build-index`: Creates/updates a local semantic search index (LanceDB [3]).
  - `search plot "<query>"`: Semantic plot search using HyDE [4].
- **Web UI Server (`inkscribe serve`):**
  - `serve start`: Launches the local FastAPI web server.

2.2.2 Web User Interface (Web UI). Served by the `inkscribe serve start` command, the web UI provides basic reading functionalities for downloaded and translated novels.

2.3 System Architecture

InkScribe is a Python application composed of several key modules, leveraging a rich set of external libraries.

The main components of InkScribe include: Core Application & CLI, Web Scrapers, NLP and Translation, Semantic Search (and Tagging), Web UI Server, Data Storage, Configuration, Shared Models, and Logging. The novelty of the INK project, as implemented in InkScribe, lies in how these components, particularly NLP, Translation, and Search/Tagging, integrate advanced techniques.

2.4 Implementation Details

This section details the core technical approaches within InkScribe and how principles from recent research are applied to achieve the INK project's goals.

2.4.1 Web Scraping with Robustness Considerations. Acquiring novel content is challenging due to various security measures employed by websites. InkScribe's scraping components (`src/inkscribe/sources`) are designed to navigate these:

- **Bypassing Common Protections:** Scrapers attempt to handle techniques like disabled copy-paste, text rendered as images (though direct text extraction is preferred over OCR), scrambled content requiring client-side JavaScript for reassembly, and devtools detection.
- **Handling API Obfuscation:** For sites using API encryption or obfuscation, the scrapers may need to simulate legitimate application behavior or reverse-engineer API call patterns. This is a continuous engineering effort due to evolving site security.
- **Technology:** Uses `BeautifulSoup4` and `curl-cffi`, supporting asynchronous operations for efficiency.

Due to the sensitive nature of these techniques and the potential for misuse, the specific source code for bypassing certain advanced security measures is not detailed further here, nor is it detailed in the public repository, but focuses on standard parsing and HTTP interaction.

2.4.2 Advanced Semantic Tagging. A core feature of INK is its multi-faceted semantic tagging system, designed to capture different layers of narrative meaning. This system categorizes tags into Independent, Dependent, and Meta tags. Unfortunately, this specific component is not yet integrated into InkScribe and instead exists as a separate prototype. It requires several days to calculate for even a single web novel.

- **Independent Tagging with GLiNER:** For tags identifiable from local context (e.g., "fighting scene", "character dialogue"), INK leverages GLiNER [8], a generalist Named Entity Recognition model. GLiNER's strength lies in its zero-shot capability: it can identify arbitrary entity types based on natural language descriptions without requiring fine-tuning for each new tag. This is achieved by encoding both the input text and tag descriptions into a latent space and matching

them, offering significant flexibility over traditional supervised models that need retraining for new tags or languages.

- **Dependent Tagging using GraphRAG, HyDE, and FLARE Principles:** Dependent tags capture complex narrative arcs or relationships that span large portions of text (e.g., "protagonist weak to strong", "non-linear storytelling"). Identifying these requires deeper contextual understanding. INK's approach is inspired by:
  - **GraphRAG [2]:** We adapt the core idea of building a narrative knowledge graph from the text, identifying key entities (characters, locations, plot devices) and their evolving relationships. This structured representation allows querying for patterns indicative of dependent tags (e.g., tracking a character's attributes over the graph to infer a "weak to strong" arc).
  - **HyDE [4]:** For complex tag definitions that might not have direct textual matches, HyDE's principle of generating a "hypothetical document" (an ideal textual example of the tag) is used. This hypothetical example is then used to find semantically similar narrative segments or patterns in the indexed content.
  - **FLARE [5]:** FLARE's concept of forward-looking active retrieval is adapted for analysis. If the system is uncertain about a dependent tag, it can be prompted to dynamically retrieve more context from other parts of the novel or the narrative knowledge graph to make a more informed decision.
- **Meta Tagging with an Agentic Framework:** Meta tags rely on external information (e.g., "adapted to movie", "original source language"). INK employs an "Agentic Tagger" component. This is an orchestrated system of AI agents equipped with tools to query external sources (e.g., web search APIs, databases like IMDb or Wikipedia). These agents search for specific information, analyze results, and assign relevant meta-tags. This process is designed to be run periodically to capture updates.

2.4.3 Semantic Plot Search Enhanced by HyDE. The deep contextual understanding developed for dependent tagging also underpins InkScribe's semantic plot search functionality (`search/core.py`). Instead of simple keyword matching, users can query their novel collection using natural language descriptions of plot events or themes.

- **Query Enhancement with HyDE [4]:** User queries are processed using Hypothetical Document Embeddings. An LLM generates one or more hypothetical document snippets that perfectly exemplify the user's plot query.
- **Vector Search:** These hypothetical documents are embedded using Google GenAI models (e.g., `models/text-embedding-004`) [7], and their embeddings are averaged to form a robust query vector. This vector is then used to search the LanceDB [3] index for the most semantically similar text chunks from the novels.
- **Result Explanation:** A generative model (e.g., `models/gemini-2.5-flash-latest`) [7] provides natural language explanations for why each retrieved chunk is relevant to the original query and an overall summary.

2.4.4 Contextual Translation with Dynamic Glossary. InkScribe's translation module (`nlp/translate.py`) prioritizes narrative consistency.

- **LLM Core with Rich Context:** Translations are performed by LLMs (via LiteLLM [1] and Instructor [6]), provided with the entire previously translated chapter to maintain immediate narrative flow.
- **Dynamic Glossary Generation:** A key innovation is the automatic generation and use of a dynamic glossary for each novel.
  - **Entity/Term Identification:** A customized PageRank-inspired algorithm identifies important entities, character names, unique items, and recurring phrases that require consistent translation. This algorithm sections words and creates a graph of co-occurrences, weighted by usage frequency within the novel and inversely by general background frequency. Long, repeated, and relatively unique terms receive higher importance scores.
  - **Glossary Application:** This glossary is provided to the LLM during translation to ensure consistent rendering of these key terms, maintaining character voice, and matching the narrative's tone and setting.
- **Structured Output:** Instructor [6] ensures that the translated output maintains the original chapter structure (e.g., paragraph breaks, dialogue).

2.4.5 Dataset Creator. To support supervised learning approaches for tagging or other analytical tasks, the Dataset Creator component leverages community-provided genres and tags from platforms like NovelUpdates. It scrapes this metadata and, in conjunction with the web scraper, can associate actual novel content with these labels to build training datasets.

3 How to Use InkScribe

InkScribe is the command-line tool that serves as the current primary interface to the INK project's functionalities. This section outlines installation, configuration, and usage of its main commands. Screenshots and more detailed visual guides of CLI operations and the web UI can be found in the final project presentation.

3.1 Installation

(1) Ensure Python 3.11 or newer is installed.
(2) Clone the InkScribe repository:
`git clone` https://github.com/davliang/inkscribe-public
(3) Navigate to the project directory: `cd inkscribe-public` (or your chosen directory name)
(4) Install dependencies (preferably in a virtual environment):
`pip install .` (or `pip install -e .` for development)
(5) (Optional) Create a `requirements.txt` using `pip freeze > requirements.txt` after installation for easier environment replication.

3.2 Configuration

InkScribe requires API keys for LLM services (Google GenAI [7] for search, and whichever provider you choose for translation via LiteLLM [1], e.g., OpenAI, Google Gemini, Anthropic).

(1) Create a `.env` file in the root directory of the InkScribe project (where `pyproject.toml` is located).
(2) Add your API keys and any other desired settings to the `.env` file. For example:

```
# .env file
GOOGLE_API_KEY="your_google_api_key_here"
OPENAI_API_KEY="your_openai_api_key_here"
# ANTHROPIC_API_KEY="your_anthropic_api_key_here"

# Optional: Specify a default translation model for
    LiteLLM
# DEFAULT_TRANSLATION_MODEL="gpt-4o-mini"
# DEFAULT_TRANSLATION_MODEL="ollama/llama3" # If
    Ollama is running

# Optional: Specify output directory
# OUTPUT_DIR="./my_novels"
```

InkScribe will load these settings. Refer to `src/inkscribe/config.py` for all configurable options.

3.3 CLI Commands

All commands are run via `inkscribe [COMMAND] [SUBCOMMAND] [ARGUMENTS] [OPTIONS]`. Use `inkscribe --help` or `inkscribe [COMMAND] --help` for detailed help.

3.3.1 Fetching Novels.

(1) **Fetch Metadata Only:** `inkscribe fetch metadata <novel_url>`
Example:

```
inkscribe fetch metadata
    "https://www.69shuba.com/book/12345.htm"
```

This creates `./output/novel_title_slug/novel.json`.
(2) **Fetch Full Novel (Metadata and Chapters):** `inkscribe fetch all <novel_url> [OPTIONS]`
Example:

```
inkscribe fetch all
    "https://www.69shuba.com/book/12345.htm" \
  --start-chapter 10 --end-chapter 20
```

This creates `novel.json` and chapter files in `./output/novel_title_slug/chapters/`.

3.3.2 Translating Content. Ensure your LLM provider API keys are set in `.env`.

(1) **Translate Novel Metadata:** `inkscribe translate metadata <novel_directory_name> [OPTIONS]`
Example:

```
inkscribe translate metadata "novel_title_slug" \
  --model "gemini/gemini-1.5-flash-latest"
```

This reads `novel.json` and creates `novel_translated.json`.
(2) **Translate Chapters:** `inkscribe translate chapters <novel_directory_name> [OPTIONS]`
Example:

```
inkscribe translate chapters "novel_title_slug" \
  --model "gpt-4o-mini" --history 3 --start-chapter 5
```

This translates chapter files and creates `_translated.json` files.

3.3.3 Semantic Analysis & Search. Ensure `GOOGLE_API_KEY` is set in `.env` for Google GenAI [7] functionalities. (Tagging commands would be detailed here if fully integrated into CLI).

(1) **Build Search Index:** `inkscribe search build-index [OPTIONS]`
Example:

```
inkscribe search build-index --force
```

This processes novels in `./output/` and populates the LanceDB [3] index in `./inkscribe_lancedb/`. This may take time depending on the corpus size.
(2) **Search by Plot:** `inkscribe search plot "<your_plot_query>" [OPTIONS]`
Example:

```
inkscribe search plot \
  "a story about a reincarnated hero who seeks
     revenge" \
  --top-k 3
```

Example (search within a specific novel):

```
inkscribe search plot "scenes with betrayal" \
  --novel "novel_title_slug"
```

This outputs ranked relevant chunks with AI-generated explanations and an overall summary.

3.3.4 Serving Web UI.

(1) **Start the Local Web Server:** `inkscribe serve start [OPTIONS]`
Example:

```
inkscribe serve start --port 8080
```

By default, it runs on http://127.0.0.1:8021. Open this URL in your browser.
(2) **Browse Novels:** The web UI will show a list of novels from your output directory. You can click on a novel to see its details and chapter list. Click on a chapter to read its content (translated if available).

4 Evaluation

The evaluation of InkScribe, as the primary software deliverable for the INK project, focused on demonstrating the functional correctness of its core components, its usefulness in addressing the identified "pain points" for users, and the overall usability of the tool. This was primarily achieved through operational testing and qualitative assessment of outputs. Illustrative examples of CLI outputs and UI interactions, including screenshots, can be found in the final project presentation; providing new, comprehensive screenshots for this report was challenging due to the significant time and computational cost associated with running the full processing pipelines for large novels. Given the ambitious scope of the INK project and the single-semester development (by just me) for its InkScribe implementation, some advanced features are in earlier

stages of integration, and evaluation reflects this. I hope that my ambition for this project and already expended effor is recognized, even if not all the envisioned components have reached full maturity within the timeframe.

4.1 Evaluation Methodology

The evaluation prioritized showcasing that InkScribe performs its intended tasks effectively and provides tangible benefits towards the INK vision. The methodology involved:

- **Functional Testing of Core CLI Commands:** Systematically executing fetching, basic translation, index building, plot search, and web UI serving commands.
- **Output Verification:** Inspecting generated files (`novel.json`, `novel_translated.json`, chapter files, search index presence).
- **Qualitative Assessment of NLP Features:**
  - **Translation:** Coherence of metadata and chapter translations, consistency of terms when using history/glossary (based on sample checks).
  - **Semantic Search:** Relevance and clarity of HyDE-enhanced [4] plot search results and AI-generated explanations.
- **Usefulness Assessment:** Empirical reflection on how InkScribe streamlines the workflow for users interested in web novels, aligning with INK's goals.

4.1.1 Note on Quantitative Evaluation of Translation Consistency. While a key goal of INK is to improve translation consistency, providing explicit quantitative evaluation results for this aspect proved challenging within the project's timeframe. Developing a robust metric to measure the "consistency" of translation based on techniques like BM25 keyword matching and semantic similarity between the source and target languages across entire novels requires significant further research, dataset preparation, and implementation. Initial explorations were undertaken, but a full run of such an evaluation across a substantial corpus and the finalization of a universally accepted consistency metric were beyond the scope of a single semester for a solo developer. The qualitative assessments mentioned above provide preliminary insights into the potential of the contextual history and dynamic glossary features within InkScribe.

4.2 Demonstration of Functionality and Correctness

4.2.1 Web Scraping. Successfully fetched metadata and chapters from test novels on supported sites. The challenges of navigating security measures highlight the utility of the dedicated scraping modules in InkScribe.

4.2.2 Translation. Metadata and chapter translations were generated. The contextual history and dynamic glossary (even in its current implementation stage) showed promise in improving consistency for key terms in sample translated segments compared to context-less translation.

4.2.3 Semantic Tagging and Search.

- **Independent Tagging:** Preliminary tests with GLiNER [8] on sample texts showed its capability to identify described tags without specific fine-tuning for those tags.
- **Dependent and Meta Tagging:** The frameworks inspired by GraphRAG [2], HyDE [4], FLARE [5], and the Agentic Tagger are conceptually designed within the INK project. While full, automated end-to-end tagging for these complex types across entire novels is a significant undertaking and part of ongoing development for InkScribe, the individual techniques (e.g., knowledge graph entity extraction, HyDE [4] for query formulation) showed promise in experimental scripts.
- **Plot Search:** The `search plot` command in InkScribe, enhanced by HyDE [4], returned relevant passages for natural language queries, with AI-generated explanations aiding comprehension.

4.2.4 Web User Interface (`serve start`). The local web UI correctly listed and displayed downloaded novels and chapters (translated where available).

4.3 Usefulness and Addressing User Needs

InkScribe, even in its current development stage, directly addresses key "pain points" identified by the INK project:

- **Automation:** Reduces manual effort in fetching and initial translation.
- **Centralized Archive:** Enables creation of a local, organized library.
- **Enhanced Discovery:** Semantic search and the vision for advanced tagging offer powerful ways to explore content beyond simple keyword searches.
- **Improved Accessibility:** Translation makes more content accessible.

4.4 Performance and Resource Observations (Qualitative)

- **Scraping & Basic CLI Operations:** Generally performant for individual novels.
- **LLM-Heavy Operations:** Translation, advanced tagging (especially agentic meta-tagging or full GraphRAG-style [2] processing), and HyDE-enhanced [4] search are computationally intensive and can incur significant API costs and time, particularly for long novels or large batches. The generation of a full narrative knowledge graph or a comprehensive dynamic glossary for a multi-million-word novel is a substantial batch process. These highlight the need for future optimizations and cost-management features within the INK framework.

5 Challenges and Future Work

5.1 Challenges Encountered

- **Web Scraping Robustness & Evolving Security:** Creating and maintaining scrapers for diverse websites with dynamic content and anti-bot measures is an ongoing challenge.
- **LLM Prompt Engineering & Structured Output:** Designing effective prompts for consistent, structured output

(Pydantic models via Instructor [6]) for complex tasks like full chapter translation or multi-faceted entity extraction required significant iteration.
- **Context Management for LLMs:** Balancing sufficient context for consistency (e.g., in translation) against token limits and API costs.
- **Orchestrating Complex AI Pipelines:** Implementing and integrating multi-step processes involving different AI models and techniques (e.g., the agentic tagger, GraphRAG-inspired [2] analysis) is complex.
- **Scalability and Cost of Advanced Features:** Applying deep semantic analysis or extensive LLM calls to entire multi-million-word novels is resource-intensive. Spending money is easy. Earning money is not.
- **Scope Management for Solo Developer:** The breadth of features envisioned for INK (scraping, translation, multiple advanced tagging types, semantic search, UI) is substantial for a solo developer to implement fully in InkScribe within a single semester, leading to varying levels of integration for different components. This reflects that I vastly overestimated what I could modestly achieve versus my ambition for it.

5.2 Limitations

- **Source Coverage:** InkScribe currently supports a limited number of novel sources.
- **LLM Dependency and Cost:** Quality and cost are tied to external LLM APIs.
- **Translation Imperfections:** LLM translations are not flawless.
- **Error Handling:** Scrapers and other components could have more sophisticated error recovery.
- **Web UI Functionality:** Currently basic, primarily for reading.
- **Code Cohesion and Integration Level:** Due to the ambitious scope of INK and solo development of InkScribe, some advanced features (especially complex tagging and full graph analysis) exist more as experimental scripts and are not yet fully integrated into the main CLI workflow or UI. The GitHub repository represents the most stable and conglomerated core components of InkScribe.
- **Quantitative Evaluation Gaps:** Comprehensive quantitative evaluation, particularly for nuanced aspects like translation consistency across long narratives, remains an area for future work due to the aforementioned challenges.
- **Incompleteness:** InkScribe only implements one fraction of a fraction of the total capabilities that I envisioned and have prototyped in various scripts.

5.3 Future Work

There are several avenues for future enhancement for INK, building upon the InkScribe foundation:

- **Expand Source Support and Scraper Robustness.**
- **Refine Dynamic Glossary and Contextual Translation:** Improve the PageRank-inspired glossary generation and explore more advanced context mechanisms for translation. Develop and integrate robust quantitative metrics for translation consistency.
- **Full Integration and Enhancement of Semantic Taggers:**
  - Fully integrate GLiNER-based [8] independent tagging into the InkScribe CLI workflow.
  - Develop the GraphRAG-inspired [2] dependent tagging pipeline, including narrative knowledge graph construction and querying.
  - Operationalize the FLARE-inspired [5] active retrieval for contextual enrichment during analysis.
  - Mature the Agentic Tagger for meta-tagging.
- **Enhanced Web UI:** Add features like in-browser search, annotation, progress tracking, and interactive exploration of semantic tags and knowledge graphs.
- **Improved Offline LLM Integration.**
- **Comparative Translation and Analysis Tools.**
- **Multimodal Translation for Graphic Narratives:** Extend capabilities to handle manga/manhua/manhwa, including OCR of stylized text and style-consistent re-rendering.
- **Performance Optimization and Cost Management:** Implement batching, parallel processing, advanced caching, and user-configurable cost controls.
- **GUI for CLI Operations.**
- **Incremental Processing:** Allow updating indexes, translations, and analyses only for new or modified content.
- **Codebase Consolidation and Refinement:** Integrate more of the experimental scripts and advanced functionalities into the main InkScribe package for better maintainability, usability, and a more unified user experience, furthering the INK vision.

6 Conclusion

INK (for Narrative Knowledge) represents an ambitious endeavor to create a comprehensive, open-source toolkit for interacting with the rich and complex world of web novels. Its current primary implementation, InkScribe, successfully demonstrates the potential of integrating web scraping, advanced LLM-based translation, and sophisticated semantic analysis techniques to address significant "pain points" for users. The project's exploration and application of principles from recent research like GLiNER [8] for flexible tagging, HyDE [4] for enhanced semantic search, and concepts from GraphRAG [2] and FLARE [5] for deeper narrative understanding, showcase a forward-looking approach to personal information management for specialized literary content.

The development journey of InkScribe, a solo effort within a single academic semester, has resulted in a functional core CLI tool and a clear architectural vision for the broader INK project. It is acknowledged that INK's ambition somewhat outpaced the practical constraints of a single-semester solo development for InkScribe, leading to varying levels of completion and integration for the more advanced conceptualized features. Nonetheless, the foundational components for fetching, translation (with a novel dynamic glossary approach), and semantic search are operational and demonstrate the core vision of INK. InkScribe provides a solid foundation for future development, with numerous avenues for

enhancing its capabilities, usability, and performance. It fulfills the core goals of the development track by delivering a useful software artifact that addresses a clear user need and by detailing its innovative design and implementation as part of the INK project.

Team Contributions

- **David Liang:** Was responsible for the entire lifecycle of the INK project and its InkScribe implementation, including conceptualization, system architecture design, research and adaptation of NLP techniques, full-stack development of all components (CLI, web scrapers, NLP/translation pipeline, semantic analysis modules, FastAPI web UI), testing, documentation, and report writing. Pretty much everything.

Software Availability

The primary public codebase for InkScribe, the current software artifact of the INK project, containing the most conglomerated and stable components at this stage, is available on GitHub:

- **Main InkScribe Repository:** https://github.com/davliang/inkscribe-public

The tool is designed to be run locally. The web interface is also served locally by the `inkscribe serve start` command. Instructions for installation and usage are provided in Section 3 and in the repository's `README.md` file.

It is important to note that the INK project was an ambitious undertaking for a single person within a single semester. Consequently, while the core functionalities of InkScribe are present in the public repository, a significant amount of exploratory work, testing, and development of more advanced or experimental features (particularly related to comprehensive graph-based analysis, fully automated complex tagging pipelines, and robust scraping of numerous sources) exists in the form of partial, scattered scripts and private experimental repositories. This separation was maintained due to the sensitive nature of some web scraping techniques and the potential for copyright infringement if all developmental code were made public prematurely.

Other related development work and experimental repositories (that were published to GitHub and does not include additional local packages on my computer) include:

- A publicly available fixed version of the Doccano annotation platform, used for manual supervised dataset creation for tagging experiments: https://github.com/davliang/doccano
- `auto-labeling-pipeline` (Doccano's auto-labeling with provided model): https://github.com/davliang/auto-labeling-pipeline
- `doccano-client` (Client for Doccano API): https://github.com/davliang/doccano-client
- `inkflow` (FastAPI/Celery based exploration): https://github.com/davliang/inkflow (Private)
- `ink-scribe` (Older Python implementation attempt): https://github.com/davliang/ink-scribe (Private)
- `ink-flow` (JavaScript/Prisma based exploration): https://github.com/davliang/ink-flow (Private)
- `ink-nlp` (Basic NLP library exploration): https://github.com/glyphs-ink/ink-nlp (Private)
- `xuanze` (Novel management pipeline exploration): https://github.com/davliang/xuanze (Private)
- `xique` (BentoML services exploration): https://github.com/davliang/xique (Private)
- `shensuo` (Next.js frontend exploration): https://github.com/davliang/shensuo (Private)
- `novel-faiss` (FAISS and PageRank exploration for glossary/search): https://github.com/davliang/novel-faiss (Private)

Future work will involve further consolidation of these scattered elements into the main public InkScribe repository as features mature and are vetted for broader release, advancing INK.

References

[1] BerriAI. 2025. *LiteLLM: Python SDK and Proxy Server for Unified LLM Access.* https://github.com/BerriAI/litellm Accessed: 2025-05-15.

[2] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL] https://arxiv.org/abs/2404.16130

[3] Inc. Eto Labs. 2025. *LanceDB: The Database for Multimodal AI.* https://github.com/lancedb/lancedb Accessed: 2025-05-15.

[4] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. arXiv:2212.10496 [cs.IR] https://arxiv.org/abs/2212.10496

[5] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. arXiv:2305.06983 [cs.CL] https://arxiv.org/abs/2305.06983

[6] Jason Liu and Contributors. 2024. *Instructor: A library for structured outputs from large language models.* https://github.com/instructor-ai/instructor

[7] Google LLC. 2025. *Google Gen AI Python SDK.* https://github.com/googleapis/python-genai Accessed: 2025-05-15.

[8] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2023. GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer. arXiv:2311.08526 [cs.CL] https://arxiv.org/abs/2311.08526