

Converting Natural Language From Complex Procedural Documents To Structured Diagrams

Nguyen Nhat Khiem

Supervisor: Dr. Tran Thanh Tung

Reviewer: Dr. Le Hai Duong

School of Computer Science and Engineering,
International University, Ho Chi Minh City, Vietnam

Thesis Defense

July 18, 2025



Table of Contents

- ① Introduction
- ② Related work
- ③ Proposed Method
- ④ Result
- ⑤ Discussion
- ⑥ Conclusion



Table of Contents

① Introduction

② Related work

③ Proposed Method

④ Result

⑤ Discussion

⑥ Conclusion



Motivation

- As procedural documents grow in volume, interpreting them remains **time-consuming** and **expert-dependent**.
- **Flowcharts** help reduce cognitive load and improve understanding.
- Advances in **LLMs** and retrieval enable smarter document visualization.
- Yet, there's **no standard metric** to evaluate text-to-flowchart transformation.



Objectives

- Develop a **method to transform procedural documents** into interactive flowcharts with support for **clarification queries**.
- Implement an **evaluation framework and experiment** to measure the **effectiveness of AI** in converting documents into flowcharts.
- Deliver a system that is **user-friendly** and supports **expert-guided refinement** through a human-in-the-loop approach.

Natural Language Understanding

LLMs apply **transformer-based attention** to understand context, semantics, and procedural flow, allowing them to **identify steps, conditions, and relationships** in unstructured text.



Actors & Features

Actors:

- **User:** a technical staff, analyst, or domain expert who uploads documents, explores flowcharts, validates results, and may refine the output manually.
In some cases, the user also plays the role of an expert verifier.
- **AI Engine (IUFlowGen):** automated system that uses LLMs and retrieval techniques to generate draft flowcharts and respond to clarification queries.

Features:

- **Send Document** — performed by **User**
- **Plot and Query** — performed by **AI Engine**
- **Interact and Ask** — performed by **User**



Contributions

Scientific Contribution

Formalize a **modular AI pipeline** that combines **retrieval, prompting, and DOT visualization**, and propose a **metric to evaluate the effectiveness** of transforming complex documents into structured flowcharts.

Real-world Contribution

Aid users in understanding procedural documents through **interactive visual flowcharts**, applicable in **education, healthcare, technical documentation, and many more**.



Table of Contents

- ① Introduction
- ② Related work
- ③ Proposed Method
- ④ Result
- ⑤ Discussion
- ⑥ Conclusion



Advancements

- GenFlowchart applies generative AI to understand and segment flowchart diagrams.
- Lester et al. introduce multi-phase summarization for long regulatory documents.
- ChatGPT identifies procedural steps and actors using advanced NLU capabilities.
- Eraser Flowchart Maker supports text-to-diagram generation with intuitive UI.



Research Gaps

Research gap 1

Current AI systems lack **interactive flowchart generation** that preserves **procedural logic and structure** from unstructured documents.

Research gap 2

The combination of **AI-guided generation with graph and vector representations** for storing and retrieving **steps, entities, and relationships** is still **under-discovered in procedural document understanding**.

⇒ **A new approach that builds graph-augmented prompts and renders interpretable, interactive flowcharts can fill this gap.**



Technical difficulties

- The amount of background knowledge required to combine **graph theory, vector search, and prompt engineering** is substantial.
- **LLMs are probabilistic**, which introduces **unpredictability in output consistency**, especially for structured generation tasks.
- Mapping **AI-generated content into structured and verifiable DOT code** demands careful post-processing and validation.



Table of Contents

- 1 Introduction
- 2 Related work
- 3 Proposed Method**
- 4 Result
- 5 Discussion
- 6 Conclusion



System overview

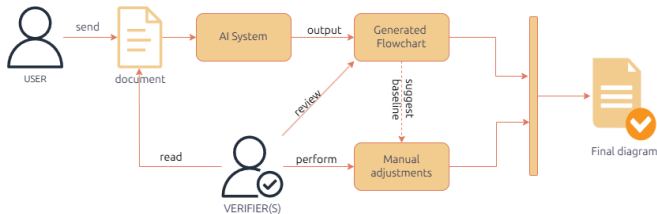
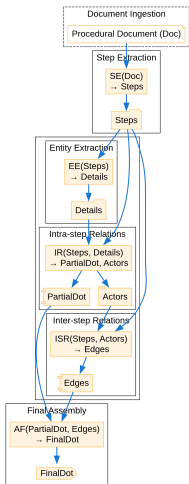


Figure: System overview of UIFlowGen Protocol

Process - Flowchart Generation

Algorithm

Pseudocode



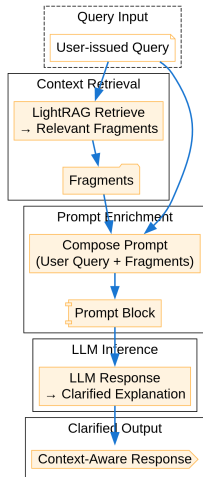
- **Goal:** convert procedural documents into **structured flowcharts** via automated step/entity extraction and reasoning.
- **Result:** generate a valid **FinalDot** representation used for rendering interactive flowcharts.
- **Step:** extract steps → enrich with entities → infer relations → assemble graph.
- **Scope:** execute locally with **LLMs**, **LightRAG**, **regex post-processing**, ensuring offline reproducibility.



Process - Query

Algorithm

Pseudocode



- **Goal:** support users and verifiers in exploring or confirming **specific flowchart elements or logic**.
- **Result:** generate a **context-aware explanation or inference** tied to queried nodes or actions.
- **Step:** receive query → retrieve relevant content → compose enriched prompt → return response.
- **Scope:** acts as an **interactive support layer** during review.



Table of Contents

- ① Introduction
- ② Related work
- ③ Proposed Method
- ④ Result
- ⑤ Discussion
- ⑥ Conclusion



Prototyping

GitHub repository: <https://github.com/Khim3/IUFlowGen>

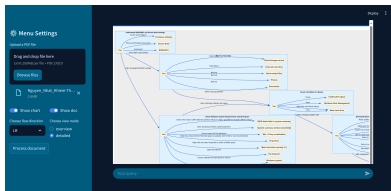


Figure: Generated flowchart from document.

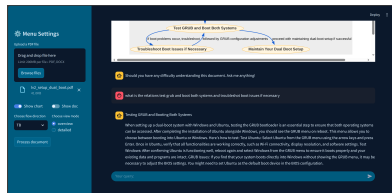


Figure: Clarification query interface

Experiment Setup

- An experiment was conducted with **10 participants**, divided into two groups:
 - **Group 1:** Without AI assistance
 - **Group 2:** With IUFlowGen (AI-assisted)
- Participants completed flowchart tasks from procedural documents across **four levels of increasing complexity**.
- All tasks were conducted under **time constraints** and **supervised by Dr. Tran Thanh Tung**.
- For the final two tasks, results were evaluated using five expert rubrics:
Rubrics: Step Coverage, Logical Flow, Relation Accuracy, Entity Representation, Structural Clarity



Benchmark Results

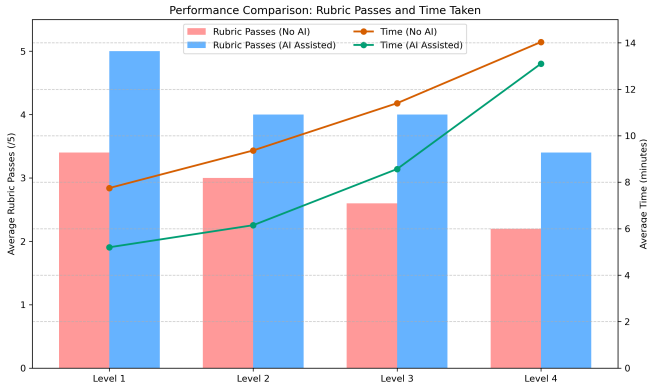


Figure: Comparison of average rubric and time across document complexity levels.



Table of Contents

- 1 Introduction
- 2 Related work
- 3 Proposed Method
- 4 Result
- 5 Discussion**
- 6 Conclusion



Analysis

- The system provides high **interpretability and transparency** by visualizing extracted procedural logic in a human-readable form.
- The **interactive human-in-the-loop model** empowers domain experts to validate and refine AI-generated flowcharts.
- IUFlowGen adapts well to varying document complexity, demonstrating strong **flexibility across synthetic and real-world texts**.
- Designed to operate on local infrastructure, the system performs reliably in **resource-constrained or offline environments**.



Limitations

- Entity extraction and semantic embedding are resource-intensive, causing latency on low-power machines.
- LLM-based generation is non-deterministic, leading to occasional inconsistencies that require human correction.
- The system supports only single-user, sequential document processing, limiting scalability in multi-user scenarios.
- Performance depends on the clarity and structure of the input document, especially for domain-specific texts.

⇒ Despite these limitations, IUFlowGen establishes a reliable base for AI-assisted procedural comprehension.



Table of Contents

- 1 Introduction
- 2 Related work
- 3 Proposed Method
- 4 Result
- 5 Discussion
- 6 Conclusion



Summary

- **Built for many applications:** IUFlowGen enables **automated, interactive flowchart generation** from procedural text, suitable for use in **education, healthcare, compliance, and technical documentation, etc.**
- **Ready for practical adoption:** Designed for **offline deployment**, IUFlowGen is accessible to **end users, organizations, and domain experts** needing secure document visualization tools.



Future Works

- Improve **UI/UX design** with real-time feedback, interactive previews, and adjustable layout controls for better user experience.
- Enable **direct flowchart editing**, including drag-and-drop, node repositioning, and in-place corrections.
- Package the system for **web or mobile deployment** to improve accessibility in education, legal, and enterprise settings.



Demonstration

Process A

Visualize the flowchart from an uploaded procedural document.

Scenario 1

Successful generation and rendering of a valid procedural flowchart.

Scenario 2

Clarification query triggered on a node to retrieve related explanation.

Process B

Perform a query from the generated chart to aid user understanding.



thank
you!

