# A PRACTICAL APPROACH TO BUILDING LEGAL KNOWLEDGE GRAPHS FROM LEGAL TEXTS FOR LEGAL CONSULTATION SYSTEMS

**Huynh Le Huy[1], Khuat Thanh Tung[2,3], Le Thi My Hanh[1]***

[1]*The University of Danang - University of Science and Technology, Vietnam*
[2]*Complex Adaptive Systems Lab, University of Technology Sydney, Australia*
[3]*NuverxAI - AI & Creative Innovation Company Limited, Vietnam*

*Corresponding author: ltmhanh@dut.udn.vn

(Received: September 13, 2025; Revised: December 05, 2025; Accepted: December 08, 2025)

DOI: 10.31130/ud-jst.2025.23(12).465E

**Abstract -** In the context of applying artificial intelligence to the legal domain, building legal question-answering (QA) systems requires a structured, queryable, and inferable knowledge foundation. This paper proposes a semi-automated method to extract and map legal knowledge from the Penal Code of Vietnam into a knowledge graph, supporting criminal law QA systems. The solution includes five main steps: (1) extracting legal text from the PDF files, (2) preprocessing and normalizing the text into individual articles, (3) using large language models (LLMs) to generate fundamental knowledge components based on the predefined rule set, (4) mapping these components into nodes and edges to construct the knowledge graph, and (5) optimizing and visualizing the graph. Preliminary results show that the model can accurately represent legal relationships such as violations, penalties, applicable subjects, and inter-article references, which will form the groundwork for the future automated legal QA applications.

**Key words -** Legal knowledge extraction; Knowledge graph; Neo4j; Large language models

## 1. Introducion

With the growing demand for legal information retrieval, many studies have focused on developing automated question-answering (QA) systems in the legal field. Traditional approaches mostly rely on keyword-based queries or expert rule systems, which often struggle with deep semantic processing and exhibit reduced accuracy when handling complex questions. Recently, the emergence of large language models (LLMs) such as OpenAI GPT [1] and Gemini [2] has opened new opportunities for understanding and processing natural language in legal contexts. At the same time, combining these models with knowledge graphs has proven to be an effective approach for enhancing reasoning and semantic querying capabilities [3, 4].

However, a significant challenge lies in accurately extracting and mapping knowledge from natural-language legal documents into a structured, queryable knowledge graph. Legal texts are often lengthy, employ complex domain-specific language, and contain numerous cross-references between articles, which makes knowledge extraction difficult. Previous research efforts have primarily focused on extracting isolated pieces of information or constructing small-scale sample graphs. These approaches have not yet fully leveraged LLMs for large-scale automation, nor have they assessed their effectiveness in supporting legal QA systems [5, 6].

To address these limitations, this paper proposes a five-step semi-automated method that integrates PDF text extraction, legal text preprocessing, knowledge generation using the Gemini LLM in JSON format, mapping into the Neo4j graph database, and constructing a complete legal knowledge graph. Notably, this study is conducted on the entire Penal Code of Vietnam, allowing for an evaluation of the method's practicality and scalability.

The main contributions of this paper are:

(i) Proposing a semi-automated legal knowledge extraction process based on LLMs and the Neo4j graph database management system;

(ii) Building a practical knowledge graph from the Penal Code of Vietnam;

(iii) Presenting a specific case study demonstrating the benefits of using a knowledge graph in a legal QA system;

(iv) Analyzing the advantages, limitations, and potential applications of the proposed method.

## 2. Background

### *2.1. Knowledge Graphs and Neo4j*

A knowledge graph is a method of representing structured knowledge, where entities (nodes) and their relationships are organized into an interconnected graph. Neo4j [7] is a popular Graph Database Management System (DBMS) designed for storing, managing, and querying graph-structured data. It is based on the property graph model, where data is represented through nodes (entities), relationships (connections between entities), and properties (attributes of nodes and relationships). Neo4j is widely used in domains such as social network analysis, recommendation systems, fraud detection, and knowledge graph construction [8].

In Neo4j, nodes can represent various real-world entities such as people, places, legal articles, or criminal behaviors, while relationships describe the links or interactions among these entities. Both nodes and relationships can carry multiple key-value pairs, allowing for flexible and semantically rich data modeling.

Cypher [9] is Neo4j's declarative query language that allows users to describe graph patterns intuitively. Common operations in Cypher include creating nodes, establishing relationships, querying patterns such as finding shortest paths, or filtering nodes based on specific

attributes. Neo4j's architecture is optimized for graph traversal and pattern matching operations, ensuring high performance even with complex queries.

Based on the powerful graph-processing capabilities, Neo4j is particularly suitable for representing legal knowledge, where legal documents inherently exhibit graph-like structures (e.g., references between articles, applicability relationships, regulatory conditions).

### *2.2. Characteristics of Vietnamese Legal Texts*

Vietnamese legal documents, particularly the Penal (Criminal) Code, possess unique characteristics that make knowledge extraction particularly challenging. Firstly, these texts are typically organized in a hierarchical structure consisting of chapters, sections, articles, clauses, and points. Each article may define multiple violations of varying severity, corresponding penalties, and affected subject groups. Additionally, legal language is highly specialized, using domain-specific terminology and long, complex sentence structures to ensure precision, comprehensiveness, and legal clarity.

Another notable feature is the high level of cross-referencing among articles within the same legal document, as well as with related legal documents. These interconnections create a dense, overlapping network of legal provisions that are interdependent, further complicating the process of extracting and reorganizing knowledge. Therefore, an effective extraction method must not only accurately identify individual legal entities but also reconstruct complete and semantically correct relationships across the entire legal document.

### *2.3. Knowledge Graphs in the Legal Domain*

In the legal domain, articles, regulations, and legal documents frequently reference, link, and constrain each other, making graph-based modeling a natural and effective approach. A typical legal knowledge graph includes nodes representing legal articles (Article), violations (Behavior), penalties (Penalty), and applicable subjects (Subject), along with relationships such as DEFINES BEHAVIOR, APPLIES TO, INCURS PENALTY, or REFERENCES other articles. Constructing such a legal knowledge graph enables deep semantic queries, such as finding all penalties related to a specific behavior, or identifying articles that reference a particular legal concept.

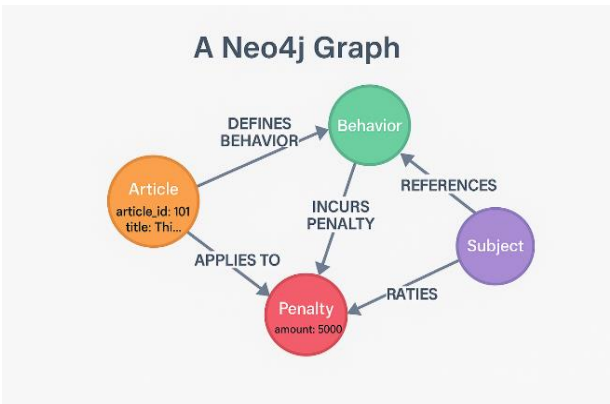


***Figure 1.*** *Legal Knowledge Graph Illustration*

In our proposed approach, Neo4j is used as the knowledge base system to store and manage the legal knowledge graph generated from the extraction process. Nodes and relationships are constructed based on JSON output from the knowledge generation model, while Cypher queries are used to explore, analyze, and develop legal QA applications. Figure 1 illustrates a simplified example of a legal knowledge graph built using Neo4j:

- Orange nodes represent legal articles, including attributes such as article ID and title.

- Green nodes denote the defined violations.

- Red nodes indicate the corresponding penalties.

- Purple nodes represent the applicable subjects regulated by the article.

Relationships in the graph describe the connections among entities, such as "DEFINES BEHAVIOR" from an article to a behavior, "INCURS PENALTY" from a behavior to a penalty, "APPLIES TO" from an article to a subject, and "REFERENCES" between articles or other entities. Modeling legal knowledge in graph form allows the system to easily query and analyze complex legal relationships, effectively supporting smart legal QA and advisory applications.

## 3. Methodology

The process of extracting and mapping legal knowledge into a knowledge graph proposed in this study consists of five main steps, illustrated in Figure 2. The steps are described in detail as follows:

### *3.1. Overview of the Knowledge Extraction and Mapping Process*

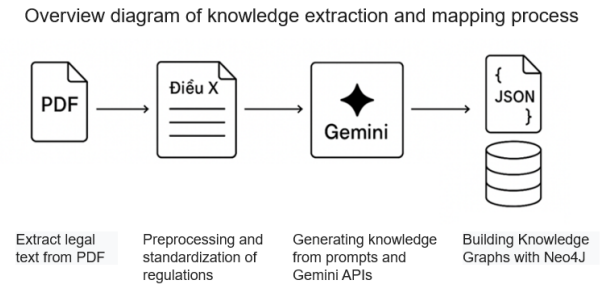


***Figure 2.*** *Overview of the Knowledge Extraction and Mapping Process from Legal Documents*

The process begins with extracting text from the PDF file of the Penal Code of Vietnam (in Vietnamese), followed by preprocessing and segmenting the content into individual legal articles. These articles are then fed into a large language model (Gemini) through a structured prompt, which includes detailed instructions regarding the types of legal entities (nodes) and the relationships (edges) between them, such as: DEFINES_BEHAVIOR, :APPLIES_TO, *etc*. These rules are predefined based on an analysis of the characteristics of Vietnamese legal texts to ensure that the output is consistently structured and suitable for mapping into a knowledge graph. The generated results are JSON files containing lists of legal entities and their relationships, which are then mapped into a graph database such as Neo4j to construct a complete, queryable, and visualizable knowledge graph.

### *3.2. Extracting Legal Text from PDF*

To extract text from the Criminal Code PDF file, the study used the PyMuPDF library to preserve formatting and full content. The extracted data is saved as plain .txt files for further processing.

### *3.3. Legal Ontology Design and Prompt Engineering Strategy*

To ensure structured and consistent extraction of legal knowledge from criminal law texts, we designed a lightweight ontology and a guided prompt engineering strategy. This ontology serves as the foundation for both the legal knowledge graph schema and the JSON-based outputs generated from large language models (LLMs).

*Ontology Design*

The ontology was developed based on an in-depth analysis of the Vietnamese Penal Code (2015, amended 2017), focusing on recurring legal concepts and their relationships. We defined five core entity types and six relation types, reflecting the semantics commonly embedded in legal provisions:

- Entity Types:

- Bo_Luat: The Penal Code itself (single root node).

- Dieu_Luat: Legal articles (e.g., Article 123).

- Hanh_Vi: Criminal behaviors or acts (e.g., murder, theft).

- Hinh_Phat: Legal penalties (e.g., imprisonment, death penalty).

- Doi_Tuong: Entities affected by or subject to the law (e.g., individual, legal entity).

- Relation Types:

- CHỨA: from Bo_Luat to each Dieu_Luat.

- QUY_ĐỊNH_HÀNH_VI: from Dieu_Luat to Hanh_Vi.

- QUY_ĐỊNH_HÌNH_PHẠT: from Dieu_Luat to Hinh_Phat.

- ÁP_DỤNG_CHO: from Dieu_Luat to Doi_Tuong.

- THAM_CHIẾU_ĐẾN: connecting related articles.

- ĐƯỢC_THỰC_HIỆN_BỞI / XỬ_PHẠT_ĐỐI_VỚI: linking behaviors or penalties to legal subjects.

This ontology is sufficient to express the majority of legal logic found in the Penal Code, while remaining simple enough to allow for scalable graph construction.

*Prompt Engineering Strategy*

We created a standardized prompt template to guide the LLM in extracting knowledge in alignment with our ontology. The prompt includes:

- Explicit listing of all allowed entity and relation types,

- Instructions to output only JSON data (no natural language explanations),

- A predefined format for nodes and edges,

- Reuse of previously created nodes (based on semantic equivalence),

- Insertion of each legal article's text as input.

### *3.4. Preprocessing and Normalizing Legal Articles*

The .txt document is split into individual articles based on headings containing the phrase "Điều X" (Article X). Each article is saved as a separate file to facilitate mapping with corresponding knowledge graph nodes. This step also includes Unicode normalization, removal of unnecessary characters, and separation of the article's title, content, and penalty clauses (if any).

### *3.5. Knowledge Generation via Prompt and Gemini API*

For each legal article, the system generates a standardized and structured prompt to be processed by the Gemini large language model. The prompt instructs the model to identify key entities such as: *Legal Article* (Dieu_Luat), *Behavior* (Hanh_Vi), *Penalty* (Hinh_Phat), and *Subject* (Doi_Tuong), as well as the relationships between them, including :DEFINES_BEHAVIOR, :APPLIES_TO, :REFERENCES, and so on. The result returned is a JSON file containing a list of nodes and relationships, normalized according to the preprocessing format for insertion into the Neo4j graph database.

### *3.6. Constructing the Knowledge Graph in Neo4j*

Finally, the system utilizes Neo4j and the Cypher query language to generate nodes and edges from the JSON files. The nodes are labeled according to their respective entity types, for example *:(Dieu_Luat {ten: "Article 123"})*, *(Hanh_Vi {mo_ta: "Murder"})*, and relationships such as *(:Dieu Luat) - [:DEFINES BEHAVIOR]→ (:Hanh Vi)*. All data is stored in graph format and can be visualized using the Neo4j Browser or other semantic query tools.

This proposed method adopts a semi-automated approach, combining traditional text processing techniques with large language models to extract and map legal knowledge. Legal texts are represented by four main entity types: *Legal Article* (Dieu_Luat), *Violation* (Hanh_Vi), *Penalty* (Hinh_Phat), and *Subject* (Doi_Tuong), along with their relationships such as :DEFINES_BEHAVIOR, :APPLIES_TO, and :REFERENCES.

To ensure accurate input for knowledge extraction, the Penal (Criminal) Code document is extracted using PyMuPDF, then preprocessed and split into separate articles. Each article is converted into an individual prompt for the Gemini API, generating JSON files containing nodes and edges. These JSON outputs are then mapped into the Neo4j graph database using Cypher, forming a complete legal knowledge graph capable of supporting semantic queries and question answering.

## 4. Experimental Results

### *4.1. The Illustration of the Constructed Graph*

Through the seamless integration of Neo4j and Python, we efficiently imported and stored Subject-Relation-Object triples into the graph database in a structured manner. This process not only validated Neo4j's effectiveness in handling large-scale knowledge graph data but also demonstrated its potential to automate the entire pipeline from data extraction to organized storage.

To visualize the structure and content of the legal knowledge graph, we utilized the built-in visualization tools provided by Neo4j. However, due to the large scale and complex nature of the comprehensive knowledge

graph, directly rendering the full dataset made it challenging to extract detailed information within specific legal domains. To address this issue, we adopted a focused visualization strategy applied across the entire Vietnamese Penal Code. The visualization process was performed using Neo4j's built-in tools, based on datasets extracted and processed from legal documents.

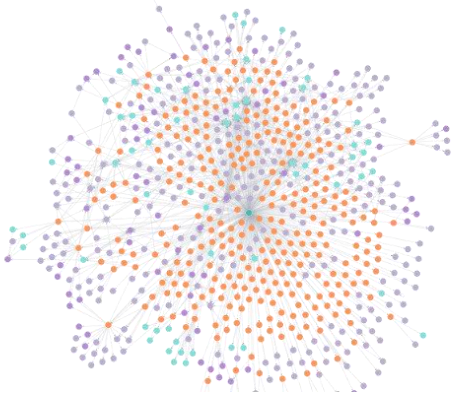

***Figure 3.*** *Visualization of the overall knowledge graph of the Penal Code of Vietnam*

The visualization results show that the legal knowledge graph constructed from the Vietnamese Penal Code comprises a total of 738 nodes and 1,452 relationships. The nodes are categorized into key groups, including: 1 *Bo_Luat* node, 314 *Dieu_Luat* (Article) nodes, 287 *Hanh_Vi* (Behavior) nodes, 57 *Hinh_Phat* (Penalty) nodes, and 79 *Doi_Tuong* (Subject) nodes. In terms of relationships, the system includes characteristic legal relations such as: *CHỨA* (CONTAINS) - 334, *QUY_ĐỊNH_HÀNH_VI* (DEFINES_BEHAVIOR) - 309, *QUY_ĐỊNH_HÌNH_PHẠT* (DEFINES_PENALTY) - 328, *ÁP_DỤNG_CHO* (APPLIES_TO) - 165, *XỬ_PHẠT_ĐỐI_VỚI* (PENALTY_FOR) - 75, *THAM_CHIẾU_ĐẾN* (REFERENCES) - 5, and *ĐƯỢC_THỰC_HIỆN_BỞI* (PERFORMED_BY). These statistics clearly reflect the richness and tightly interlinked nature of legal entities within the Penal (Criminal) Code document, demonstrating the capability to represent legal knowledge in a structured, visual, and semantically queryable format, which is ready to support intelligent legal question-answering systems.

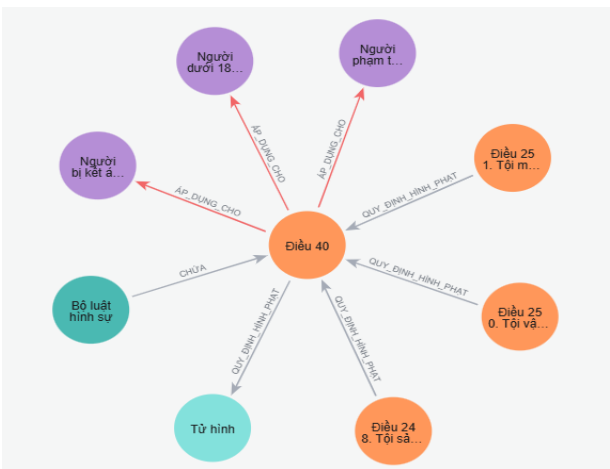


***Figure 4.*** *Local Subgraph Around Article 40*

As illustrated in Figure 3, the interconnected network of articles and their associated legal components, such as prohibited behaviors, penalty regulations, adjudication criteria, and subjects of criminal responsibility, can be clearly observed. Visualizing the entire code not only overcomes the limitations of traditional linear text processing methods but also opens up the possibility for multi-dimensional analysis of legal relationships, providing strong support for in-depth research and practical applications in the field of criminal law.

While the full visualization in Figure 3 provides a comprehensive overview of the global structure of the Penal Code knowledge graph, such a large-scale representation can make it difficult to observe fine-grained legal relationships at the article level. To address this, we include an enlarged subgraph focusing on a single legal provision, as shown in Figure 4. This detailed subgraph, centered on Article 40, illustrates how the knowledge graph encodes localized legal semantics through connections to applicable subjects, prescribed penalties, cross-referenced articles, and other legally relevant entities. By zooming into a specific article, the visualization makes the internal organization of the graph more interpretable, revealing how individual legal concepts interact and how semantic relationships are embedded within the encoded structure. This focused view not only complements the global visualization but also highlights the practical value of knowledge graph representations for legal reasoning and article-level analysis.

### *4.2. Comparative Evaluation: LLM-based QA vs. Knowledge Graph-Enhanced QA*

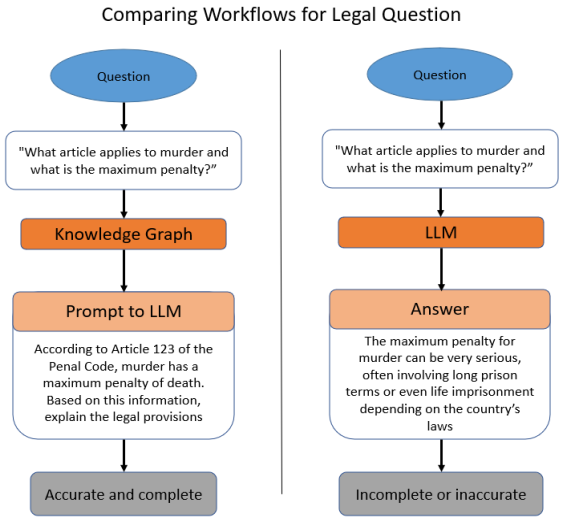


***Figure 5.*** *Comparing Workflows for Legal Question*

In the context of developing automated legal question-answering systems, the presence or absence of a knowledge graph results in a clear difference in answer accuracy, completeness, and content controllability. Without the use of a knowledge graph, the system relies entirely on the semantic inference capabilities of large language models such as Gemini or GPT-4o. For example, a natural-language question like: "*If a person*

*commits murder, which article applies, and what is the maximum penalty*?" is submitted directly to the model. The LLM then generates a response based solely on its internal knowledge and training data. This workflow, illustrated on the right side of Figure 5, highlights the limitations of relying solely on LLMs, as the model may produce incomplete answers (e.g., omitting the article number) or incorrect ones (e.g., referencing the wrong legal provision). These inaccuracies stem from the probabilistic nature of LLMs, which are not connected to an authoritative legal knowledge base.

Although in some cases the model can provide correct answers, such as identifying Article 123 and the maximum penalty of capital punishment, numerous inaccuracies still occur. For example, the model may return incomplete answers (e.g., stating only the penalty without referencing the applicable article) or cite the wrong article, such as Article 124, which refers to involuntary manslaughter, while the question clearly addresses intentional murder. Due to the probabilistic nature of LLMs and their lack of connection to an authoritative legal knowledge base, confusion between legal provisions is often unavoidable.

To ensure a fair and systematic comparison between the two approaches, we constructed a benchmark evaluation framework consisting of three components. First, we created a test set of 25 natural-language legal questions that cover a broad spectrum of Vietnamese criminal law topics, such as murder, theft, bribery, rape, and drug-related offenses. For each question, we established a ground truth reference including the correct legal article(s) and the maximum statutory penalty, based on expert verification from the Vietnamese Penal Code (2015, revised 2017). Second, each question was independently submitted to two different QA pipelines: (1) a baseline pipeline using only the LLM, and (2) an enhanced pipeline that retrieves relevant legal facts from a Neo4j-based knowledge graph and injects them into the prompt before passing it to the LLM. Third, each answer was manually assessed using two objective criteria: (i) correctness of the cited article and (ii) correctness of the stated penalty. This design allows us to quantify and compare the legal reasoning performance of both systems under identical input conditions.

To evaluate this, we experimented comparing the performance of legal question answering in two scenarios: using only a LLM and combining the LLM with a legal knowledge graph. Specifically, we created a set of **25 real-world legal questions** written in natural language, simulating common criminal law scenarios such as intentional murder, property theft, extortion, rape, bribery, drug possession, and engaging in sexual acts with minors. Each question was paired with a ground truth consisting of the applicable legal article and the maximum penalty, based on the *Vietnamese Penal Code* (2015, revised 2017). Although the current evaluation uses a test set of 25 manually curated questions, which provides an initial but limited view of system performance, we acknowledge that a larger benchmark dataset is necessary to fully capture the diversity and complexity of legal queries. Future work will expand this dataset to hundreds of questions covering broader legal contexts and multiple levels of legal reasoning. The results showed that the system using **only the LLM** achieved approximately 84% accuracy for complete and correct answers.

In contrast, when the system was integrated with the knowledge graph, the QA process became more structured and controllable. Instead of sending the question directly to the LLM, the system first queried the knowledge graph, which was built from the Penal (Criminal) Code, to identify relevant entities and relationships. In the murder case, for instance, the behavior "murder" was linked to *Article 123*, and then further queried to retrieve the corresponding maximum penalty, which is capital punishment. This information was then incorporated into a context-enriched prompt for the LLM, such as:

"*According to the Vietnamese Penal Code, the behavior 'murder' is defined in Article 123, with the maximum penalty being death. Based on this information, please explain the relevant legal provisions*."

With solid background knowledge provided, the model merely elaborated and explained the validated information, thus minimizing reasoning errors. The evaluation showed that the accuracy rate increased to approximately **92%**, and the responses became more complete, consistent, and legally grounded. The empirical outcomes confirm the role of the knowledge graph not only as an effective repository of legal information, but also as a critical support layer that significantly improves the quality of output in intelligent legal QA systems.

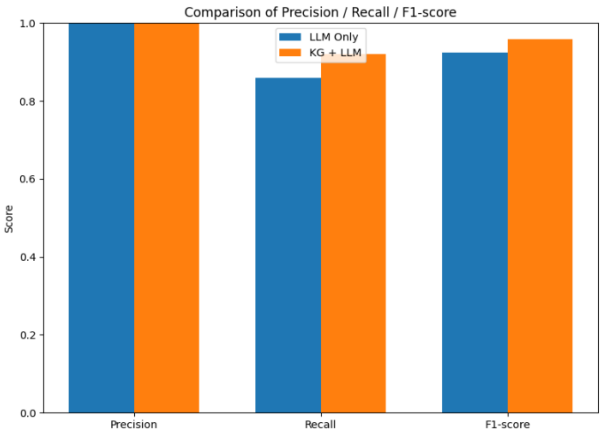


***Figure 6.*** *Comparison of Precision / Recall / F1-score between LLM-only and KG+LLM pipelines*

In addition to accuracy, we further extended the evaluation with more objective quantitative metrics, including Precision, Recall, and F1-score, to provide a clearer comparison between the two QA pipelines. A prediction was considered correct only when both the cited article and the corresponding maximum penalty matched the ground truth. As shown in Figure 6, the LLM-only baseline achieved a Precision of 1.00, a Recall of 0.86, and an F1-score of 0.9247. These results indicate that while the LLM produces highly accurate answers when confident, it frequently omits necessary legal details, leading to lower recall.

In contrast, the KG-enhanced pipeline achieved the same Precision of 1.00 but demonstrated a markedly higher Recall of 0.92 and an improved F1-score of 0.9583. This improvement reflects the stabilizing effect of incorporating verified legal facts retrieved from the knowledge graph, which helps the LLM avoid missing essential components of the answer and reduces reasoning errors. Together, these metrics provide a more comprehensive evaluation beyond accuracy alone and demonstrate that integrating a legal knowledge graph enhances both completeness and reliability of legal question answering.

Although both models achieve perfect precision, several failure patterns highlight their inherent limitations. The KG+LLM system mainly fails when the Knowledge Graph lacks complete or properly linked information. In such cases, the model cannot retrieve the required article or penalty, leading to missing or irrelevant outputs and a noticeable drop in recall. This demonstrates the strong dependency of KG-based reasoning on data completeness.

Conversely, the LLM-only system rarely predicts the wrong article but often produces long and unstructured explanations. This over-generation makes it difficult to extract the maximum statutory penalty and may introduce small inconsistencies when multiple sentencing tiers exist. In addition, the LLM-only model sometimes struggles with questions whose legal terms are semantically similar. A notable example is the confusion between *cưỡng đoạt tài sản (Article 170)* and *cướp giật tài sản (Article 171)*. Although both belong to related categories of property offenses, the model may initially drift toward the wrong legal group due to linguistic similarity before eventually producing the correct article through post-extraction. This semantic drift illustrates how LLMs may prioritize contextual similarity over strict legal classification.

Overall, the failure cases show that LLM-only suffers from verbosity, semantic confusion, and extraction noise, whereas KG+LLM is vulnerable to missing or incomplete graph data. The results suggest that improving KG coverage and enforcing more structured generation could further enhance system robustness.

## 5. Discussions

Initial experimental results indicated that the proposed method holds great potential for transforming legal texts into knowledge graphs to support legal query and question-answering applications. However, during implementation, several strengths and limitations were observed. One clear advantage of the method is its high level of automation: the entire process, from preprocessing to graph generation, requires minimal manual intervention, especially when a standardized prompt format is established based on carefully defined entity types and relationship categories derived from the structure of legal documents. In addition, the Gemini large language model demonstrated strong performance in processing lengthy and complex legal texts, showing the ability to accurately extract multiple behaviors and penalties within a single article. The scalability of the method is also promising, as it can be readily applied to other legal codes such as the Civil Code or the Traffic Law, which share similar structural characteristics.

Nonetheless, the method presents certain limitations. A major challenge is its reliance on prompt engineering, as the clarity and completeness of the defined entity types, semantic roles, and relationship categories directly influence the structure of the extracted knowledge. Ambiguous or inconsistently phrased prompts may produce redundant entities, missing penalties, or incorrect relationships, reducing the robustness and reproducibility of the system. Furthermore, some legal articles possess unique structural characteristics that can lead to errors in the knowledge generation process, such as mixing behaviors with penalties or omitting important cross-references. Another limitation is that knowledge validation is still largely conducted manually, which hinders scalability when applying the method to larger multi-document legal corpora. To address this limitation, in future work, we plan to automate the validation process using rule-based consistency checks and embedding similarity models to verify entity-relationship correctness, thus minimizing manual validation efforts.

Another important direction emerging from the experimental findings relates to the need for inter-document integration and automated cross-legal validation. While the current approach focuses primarily on transforming individual articles within the Penal Code into structured knowledge representations, many legal questions in real-world scenarios require reasoning across multiple legal documents, such as linking criminal liability to corresponding civil compensation obligations or aligning penal provisions with procedural requirements defined in the Code of Criminal Procedure. Without mechanisms for cross-referencing, the generated knowledge graph remains isolated within a single legal domain, limiting its ability to support more complex legal reasoning tasks.

To improve scalability and enhance reasoning quality, future extensions of this work should incorporate systematic inter-document linking across major legal codes, including the Civil Code, the Criminal Procedure Code, and domain-specific regulations. This could be achieved by harmonizing schema definitions, introducing canonical identifiers for shared legal concepts, and applying automated cross-checking techniques to detect contradictions or semantic mismatches between related provisions. Integrating such capabilities would not only strengthen the structural consistency of the knowledge graph but also provide a foundation for more advanced applications, such as multi-domain legal question answering and legal compliance analysis.

Despite these limitations, the proposed approach demonstrates strong potential for building legal knowledge bases that support semantic querying, developing legal consultation chatbots powered by knowledge graphs, and assisting in the analysis and detection of inconsistencies or overlaps in the current legal document system.

## 6. Conclusions and future work

In this paper, we proposed a semi-automated method for extracting and mapping legal knowledge from legal texts into a knowledge graph, to support intelligent legal question-answering systems. The method consists of five main steps: extracting text from PDF documents, preprocessing, generating knowledge using a large language model based on specifically defined entity types and relationships, mapping the generated knowledge into a knowledge graph, and visualizing the results. Initial experimental results on the Vietnamese Penal Code demonstrate the ability to accurately extract legal entities such as behaviors, penalties, articles, and applicable subjects, along with the relationships among them. The resulting knowledge graph can significantly support the development of legal query systems and legal advisory chatbots.

Looking forward, we plan to pursue several directions for further development. First, we will optimize prompt design and address knowledge generation errors through a deeper investigation into prompt engineering, entity and relationship modeling in legal texts, and output quality control from the LLM. Second, we aim to build an automated evaluation framework to assess the accuracy of extracted knowledge and reduce reliance on manual validation. Third, we intend to integrate natural language query technologies to develop a complete end-to-end legal QA system, in which the knowledge graph serves as the foundation for generating accurate and user-friendly responses. Lastly, we will expand the dataset to apply this method to other legal domains such as the Civil Code, Traffic Law, Administrative Law, and related legal areas.

## REFERENCES

[1] OpenAI, "GPT-4 Technical Report," *arXiv preprint* arXiv:2303.08774, 2023.

[2] G. Team *et al*. "Gemini: A Family of Highly Capable Multimodal Models," *arXiv preprint*, arXiv:2312.11805, 2023.

[3] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Neural approaches to conversational AI: Question answering, task-oriented dialogues and social chatbots,” *Information Fusion*, vol. 52, pp. 90-107, 2019.

[4] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, “A survey on knowledge graphs: Representation, acquisition, and applications,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494-514, 2021.

[5] T.-H.-Y. Vuong, M.-Q. Hoang, T.-M. Nguyen, H.-T. Nguyen, and H.-T. Nguyen, “Constructing a Knowledge Graph for Vietnamese Legal Cases with Heterogeneous Graphs,” *arXiv preprint*, arXiv:2309.09069, 2023.

[6] B. Dong, H. Yu, and H. Li, “A Knowledge Graph Construction Approach for Legal Domain,” *Tehnički vjesnik*, vol. 28, no. 2, pp. 357-362, 2021.

[7] Neo4j, "Neo4j Documentation," *neo4j.com*. [Online]. Available: https://neo4j.com/docs/ [Accessed September 10, 2025].

[8] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases: New Opportunities for Connected Data*, 2nd ed. Sebastopol, CA: O’Reilly Media, 2015.

[9] Neo4j, "Neo4j Cypher Manual," *neo4j.com*. [Online]. Available: https://neo4j.com/developer/cypher/ [Accessed September 10, 2025].