Ashoka University

Department of Computer Science

Capstone Project

Using Large Language Models for Legal Text Analytics

Suyog Joshi

(1020211358)

Supervisors: Dr. Lipika Dey, Dr. Partha Pratim Das

Submitted December 8, 2024

Abstract

Legal text analysis proves to be a lengthy, expensive, and time-consuming process. By using Large Language Models (LLMs), this process has been simplified but just this approach is unable to provide a complete picture of legal texts, which tend to be interconnected and complex. This project demonstrates the feasibility of integrating LLMs with knowledge graphs in order to understand large corpora of legal texts in context with each other, thereby providing a comprehensive understanding of a legal field. Focusing specifically on food safety cases in India from January 2022 - December 2023, this project compares traditional Named Entity Recognition methods with LLM-based entity extraction and summarisation. From the LLM-based entity extraction, knowledge graphs are created and used to generate insights about food safety cases, thereby illustrating the utility of combining LLM output with knowledge graphs to understand legal texts. Future work and limitations are discussed.

Keywords: legal text analysis, natural language processing, large language models, knowledge graphs, food safety.

Report's total word count: 5120 words

Program code: GitHub

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Lipika Dey, for her invaluable guidance and support throughout the course of this project. Dr. Dey's expertise and insights have been significant in shaping this work and helping me navigate its challenges.

I would also like to thank my peers for their support and insights as I developed this project. Their constructive feedback, collaborative discussions, and willingness to share perspectives were instrumental in refining my approach and overcoming technical challenges.

Contents

List of Figures

List of Tables

Introduction

"Language is to lawyers what a piano is to the pianst: the tool of her trade."

- Andrei Marmor

1.1 Background

Legal documents, such as court judgements, are complex in nature, documenting various aspects of a case. These documents capture the facts of the case, detailed arguments from all parties involved, and the judge's reasoning and final decision. They are lengthy, due to their precise and repetitive nature which ensures clarity. Legal professionals have to handle large volumes of legal text regularly, which is a challenging and time-consuming task. Computational tools and natural language processing (NLP) techniques offer methods to mitigate this challenge, allowing for more efficient case analysis and information extraction.

Legal Text Analytics is the practice of automatically extracting knowledge from legal data. This includes a vast range of tasks, including named entity recognition (NER), summarization, argument mining, and question answering, among others. Prior tools created for these tasks have largely been developed using supervised learning methods, which requires large amounts of carefully annotated data needed for training. This is a time-consuming and expensive task.

Large Language Models (LLMs) have shown the ability to handle large amounts of unstructured text data, and extract information quickly. They are useful when analysing singular documents, but cannot provide quantitative insights from multiple documents on their own. Tools like retrieval-augmented generation allow LLMs to integrate external knowledge sources, enabling them to answer queries based on large datasets. However, this approach is still limited in providing structured representations of relationships between entities across multiple documents, which are essential for comprehensive legal analysis.

To overcome these limitations, this project uses a hybrid approach combining LLMs with knowledge graphs. LLMs are used to both summarize documents and extract key information, such as the statutes cited, names of respondents/appellants, key facts, and the decision taken by the judge. Some of these entities are then used to populate a knowledge graph, which was designed following a legal ontology that stores the core legal concepts and relationships among them. This graph can be queried to gather corpus-level insights, as opposed to insights extracted from individual documents.

1.2 Objectives

1.2.1 Automated summarization of documents

A typical court judgement can span dozens to hundreds of pages. This complexity means manually processing information is a challenge. LLMs can generate concise, accurate sum-

maries of documents that capture case details, reasoning, and outcomes, allowing for faster comprehension of documents. This project uses GPT-4o-mini for summarization.

1.2.2 Extraction of relevant legal entities

To construct a knowledge graph, relevant entities must be extracted from the corpus of documents. This includes entities such as the statutes, provisions, and precedents[1] cited, the names of key actors (judges, appellants, and respondents), and the outcome of the case. These entities can then be linked to each other using pre-defined relationships following a legal ontology.

1.2.3 Definition and creation of a Knowledge Graph

A Knowledge Graph (KG) represents a structured semantic representation of entities and their relationships. For this project, the creation of a knowledge graph involves extracting entities from unstructured legal text, and representing the relationships between them using a pre-defined ontology. To ensure that the system is generalizable, this project chooses a design which models relationships common across all legal cases. For example, representing the food items mentioned in a case would be useful for food safety cases, but irrelevant for tax code-based cases.

1.2.4 Knowledge graph queries

To extract meaningful quantitative insights from the corpus, the knowledge graph can be queried using a graph query language like `Cypher` from `Neo4j`. Some example insights that are valuable for analysis are:

1. Top-n most common statutes cited across the database.
2. Cases which have statute x and precedent y in common.
3. Cases which are most similar to case x, based on similarity algorithms such as the Jaccard index.
4. In cases involving food safety, what are the most common food items involved?

1.3 Dataset

For the purposes of testing the pipelines defined in this project, 196 cases related to food safety are used. (Indian Kanoon, 2024) is a repository of digitized court proceedings with advanced search features, which allowed for cases specifically mentioning the *Food Safety Standards and Standards Act, 2006* to be scraped.

[1] Statutes refer to the laws enacted by a legislative body such as the Parliament, and provisions are the specific sections and clauses within a larger legal document, including statutes, contracts, and the Constitution.

Literature Review

Prior work on legal text analytics has largely involved statistical NLP or supervised learning methods. Dozier et al. (2010) proposed a statistical method to extract named entities from U.S. legal judgements. However, statistical models require large volumes of annotated data to maintain high accuracy. Supervised learning models have also been developed for the legal domain, such as the one proposed in Pais et al. (2021), which uses a BiLSTM model for information extraction. Transformer based models, such as BERT, have also been used for tasks like information extraction from legal text (Darji et al., 2023). The problem of annotated data remains, however, as the models learn from manually labeled data. Although this is possible, it is not always feasible, given time-consuming and possibly expensive nature of the task. Recent approaches to information extraction have explored the use of pretrained Large Language Models, both for general use (Peng et al., 2024), and for field-specific documents such as medical texts (Wiest et al., 2024). These methods have shown promising results, as they are able to extract information with few-shot learning from prompts. Thus, this project uses LLMs for entity extraction in legal texts, and compares the results to the BERT-based NER model proposed by Kalamkar et al. (2022).

Summarization is a key task in legal text analytics, due to the complexity of legal documents. Summaries of legal texts can allow users to quickly understand the facts of a case, but in a field as sensitive as law, accuracy is vital. Thus, the summaries generated have to be accurate and trustworthy. Recent advancements in large language models (LLMs) have provided significant improvements in the accuracy and efficiency of legal text summarization. Unlike traditional models, LLMs are trained on vast pretraining datasets and can use few-shot learning to generate summaries that are concise and semantically rich. For example, models such as GPT-4 have demonstrated their ability to produce detailed yet accurate summaries of legal cases (Laban et al., 2023), capturing the complex reasoning presented in legal documents.

Several domain-specific approaches have been proposed to ensure the accuracy and trustworthiness of summaries in fields like law. LawLLM (Shu et al., 2024), for example, focuses on generating case summaries that include important legal details while minimizing ambiguities in the text. Chowdhury et al. (2024) propose a summary-of-summaries method, which combines chunk-wise summaries to generate summaries of documents with high granularity. This project uses LLMs for summarization, and explores the ability of LLMs to summarize entire documents without needing to use methods like summary-of-summaries.

Although LLMs are useful for tasks like summarization and information extraction, they are not able to analyse many documents at once. Thus, to extract insights from a large corpus of documents, other methods must be used. Recent developments in document analysis have explored the use of knowledge graphs to represent the relationships between and within documents. Storti (2022) uses an ontological model to represent information related to musical documents, which can be queried, navigated, and explored. Similarly, van Sonsbeek et al. (2023) employs knowledge graphs to represent radiology reports, which helps clinical professionals understand the underlying relationships between medical terms.

By combining the extractive and generative capabilities of LLMs with the structured, semantic representation provided by knowledge graphs, this project aims to address some of the challenges in legal text analytics. Specifically, the project seeks to overcome limitations such as the inability of LLMs to quantitatively analyze large corpora and the challenge of extracting insights across related documents. Knowledge graphs, by structuring legal relationships into nodes and edges, enable cross-document analysis, while LLMs add value by generating summaries and extracting entities efficiently.

Methodology

3.1 Data Collection and Preprocessing

Court proceedings related to food safety were collected from IndianKanoon using a webscraper based on `Selenium` and `BeautifulSoup`. Prior consent to use judgements available on the website was received from the IndianKanoon team. First, a search term was created, querying for judgements mentioning the *Food Safety and Standards Act, 2006*. The query was limited to judgements passed between January 1, 2022 and December 31, 2023. 200 cases were collected, out of which 3 were dropped due to incorrect formatting, and 1 was dropped due to its length exceeding the `GPT-4o-mini` context length of 128,000 tokens. Table 3.1 presents the statistical measures for the corpus.

Table 3.1: Statistical measures for judgement corpus

Statistic	Value (in tokens)
Count	196
Mean	5064.65
Standard Deviation	9909.77
Maximum	53941
Minimum	453
Median	1121
Sum	992672

Minimal pre-processing was conducted. Links common to all documents, such as links pointing to IndianKanoon or judis.nic.in were removed. The remaining text was passed downstream as-is.

3.2 Knowledge Graph Design

A knowledge graph design was created that aims to capture some of the key information that is relevant for insightful analysis. The design was based on concepts and relationships presented in Leone et al. (2019). The main features are the provisions and statutes cited within each document, as they outline the legal framework followed across cases. The statutes represent the broader issues being discussed—for example, a tax-related case may cite the Income Tax Act—and the provisions represent the specific clauses and sub-sections that provide the precise context for the case's arguments and judicial reasoning. Other information is also extracted, such as the judge overseeing the case, names of petitioners and respondents, the names of locations and organizations mentioned in the document, and the final judgement in the case.

Figure 3.1 depicts the entities and relationships extracted.

Including statutes and provisions in the knowledge graph allows for meaningful insights to be extracted about the nature of the cases present in a corpus. The graph can be queried for quantitative insights. This includes information like the most commonly cited statutes across cases, clustering of cases based on citation similarity, and finding the cases most similar to a chosen case. These queries can provide information about patterns across the corpus, such as understanding the frequency of specific legal interpretations, or identifying the most common reasoning provided in specific types of cases.

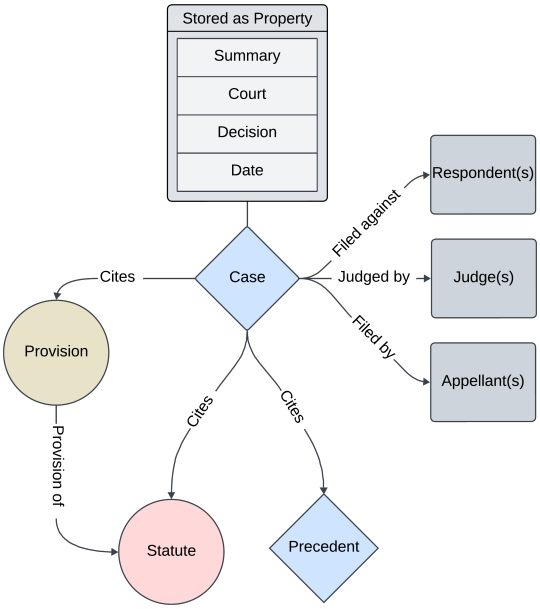

Figure 3.1: Knowledge Graph Design

3.3 Named Entity Recognition for Legal Entity Extraction

3.3.1 Model design

The first method implemented to extract the relevant legal entities was the “traditional" method, named entity recognition (NER). Figure 3.2 shows the pipeline developed for NER-based extraction. A general purpose NER model trained on general domain text was not feasible, as legal text has a highly specialized vocabulary and domain-specific terminology. The model, developed by Kalamkar et al. (2022), was trained on a dataset of 14444 Indian court judgements, labeled with 14 entity types. Table 3.3 lists the entity types extracted by this model. The NER model consists of a transition-based dependency parser (Honnibal and Johnson, 2015) on top of RoBERTa-base (Liu et al., 2019), a transformer model based on BERT. The model was fine-tuned on the labeled judgement dataset, and accuracy scores were calculated, with the model achieving an F1 score of 0.911. This model's output was used as the baseline to calculate the LLM-extracted output.

For the purpose of this project, the pre-trained model was downloaded from GitHub and used as-is. All 14 entity types extracted from each document were stored. A fault of using the NER model was that it extracted all the mentions of each entity. That is, the model was not able to recognize two differing mentions of a singular entity as the same entity. For

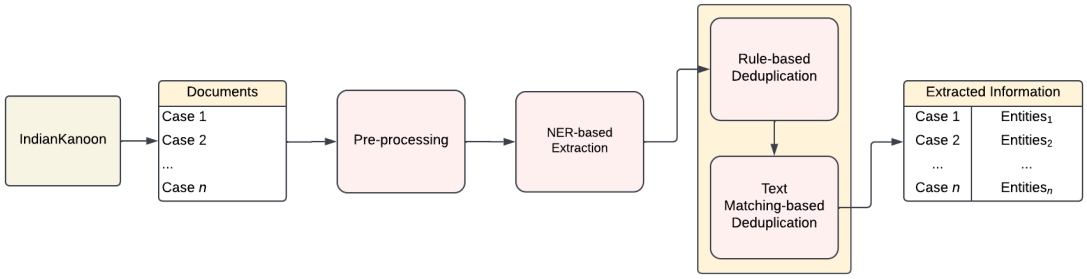

Figure 3.2: Pipeline for NER-based knowledge extraction

example, a legal document often uses abbreviations of statute names. Table 3.2 lists some examples of this effect. Including all of these duplicate entities would be problematic, as the LLM-extracted output only included one instance of each entity. Thus, a deduplication method was created for accurate comparison.

Table 3.2: Common Variations of Statute and Legal Entity Names

Entity/Statute Name	Common Variations
Food Safety and Standards Act, 2006	FSS, Food Safety Act, Food Safety Act 2006
Indian Penal Code, 1860	IPC, Penal Code, IPC 1860
Constitution of India	Constitution, Indian Constitution, COI
High Court	HC, High Court of [State Name]
Code of Criminal Procedure, 1973	CrPC, Criminal Procedure Code, CrPC 1973

3.3.2 Detection and removal of duplicate entities

The first method for deduplication was a simple, rule-based technique. Since some abbreviations and variations are consistent across documents, they can be identified using regular expressions. For example, *FSS Act* is always a reference to the *Food Safety and Standards Act, 2006*. Similarly, any extracted statutes that included the words *food safety* and *act* within them were assumed to be references to the *FSS Act*. Such rules were created for the most commonly-occurring laws, including the *Indian Penal Code*, and the *Code of Criminal Procedure*.

However, this was not possible for all the laws cited in the corpus, given the vast range of citations. Thus, a method based on similarity, outlined in Alg. 1 was used. The method identifies similar entities based on the overlap of characters between them. If the similarity is above a certain threshold, one of the similar entities is removed. To ensure that non-duplicate names were not accidentally removed, this threshold was set to be relatively high (0.7, meaning names with $\geq$ 70% of characters being identical were removed).

Though the NER-based extraction method is useful to extract certain entities, there are various challenges in converting that extracted data to usable knowledge, as shown by the deduplication issue. Another major issue with NER is that if entities not labeled in the training data need to be extracted, the model must be re-trained with a vast amount of labeled data. Using LLMs sovles this problem, as illustrated in the next section.

Algorithm 1 Text-Matching-Based Deduplication

Input: List of extracted entities $E = [e_1, e_2, \ldots, e_n]$, similarity threshold θ
Output: Deduplicated list of entities E'

$E' \leftarrow []$ ▷ Initialize the deduplicated list
for each entity e_i in E **do**
    Normalize e_i (lowercase, remove spaces/punctuation, tokenize)
    $is\_duplicate \leftarrow$ False
    for each entity e_j in E' **do**
        Compute overlap similarity Similarity(e_i, e_j)
        if Similarity(e_i, e_j) $\geq \theta$ **then**
            $is\_duplicate \leftarrow$ True
            break
        end if
    end for
    if $is\_duplicate =$ False **then**
        Add e_i to E'
    end if
end for
for each entity e_i in E' **do**
    Apply rule-based mappings for abbreviations and variations
end for
return E'

3.4 Using LLMs for Entity Extraction and Summarization

Due to the challenges associated with using NER for legal entity extraction, this project explores the use of LLMs for the same task. LLMs are trained on vast amounts of data, meaning they can generalize across domains better than models trained on a specific dataset. Unlike traditional NER models, LLMs can recognize entities based on context without requiring exhaustive labeling for each entity type. This is particularly beneficial in the legal domain, where new statutes, amendments, and case-specific terminologies may emerge. This project uses `GPT-4o-mini` through the OpenAI API. The API is used due to the high computational costs of LLM inference. `GPT-4o-mini` is chosen over `GPT-4o` as it is cheaper while providing better performance than the older `GPT-3.5` model. Document length was also a factor when choosing the model, as most documents were longer than the context length of models like `GPT-3.5`. Previous approaches (Chowdhury et al., 2024) have proposed methods like using document summaries to get around this, but given the long context lengths (up to 128k tokens) of newer models, long documents can directly be provided to the model for extraction. Figure 3.3 depicts the pipeline for LLM-based knowledge graph population.

3.4.1 Entity Extraction

The project uses a prompt-engineering approach, where a legal document is provided as input along with a structured prompt to extract specific entity types. OpenAI's API provides support for structured output, meaning the LLM can output data in a structured format following a schema provided by the caller. This allowed for certain entities to be extracted in string format (i.e. entities that have only one occurrence, such as the date or the judge's name), while others were directly returned in lists (statutes, provisions, etc). The prompt passed to the API for extraction is listed in Appendix A.1. The prompt attempts to ensure the output is

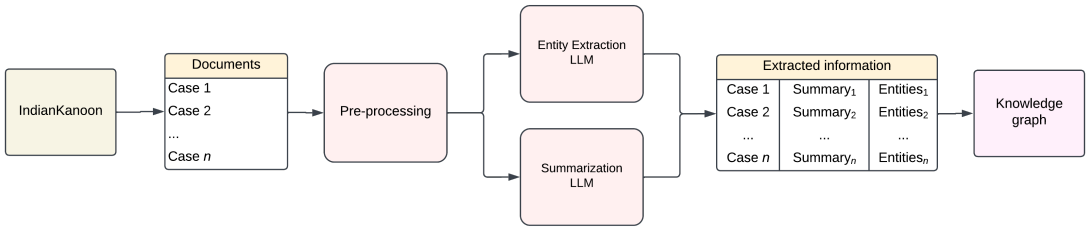

Figure 3.3: Pipeline for LLM-based summarization and knowledge extraction

standardized across API calls, so that no further processing is required after extraction. The schema for the LLM output is as shown in Code Snippet B.3.

```
court: str
petitioners: list[str]
respondents: list[str]
judges: list[str]
date: str
org: list[str]
gpe: list[str]
provisions: list[str]
statutes: list[str]
precedents: list[str]
key_facts: str
type_of_case: str
decision: str
```

Listing 3.1: Entity Extraction Data Structure

The output from the LLM is stored in a DataFrame along with the summaries. Other than the lack of duplicate entities, another advantage of using LLMs for information extraction is that any new type of entity can be added to the outputs without needing to re-train the model with labeled data. Table 3.3 compares the entity types extracted by the LLM and the NER model. The accuracy of the LLM relative to the NER output and the documents themselves is presented in Chapter 4.

3.4.2 Summarization

Another advantage of using LLMs for legal text analytics is their ability to quickly generate summaries. Summaries of legal documents allow the user to quickly understand the context, argumentation, and judicial reasoning present in the document. This project uses `GPT-4o-mini` to generate summaries of each document, and stores them as properties of the document's respective node in the knowledge graph. This means that when the user is browsing the graph for insights, they can click on a case node they are interested in, and quickly learn about the case.

As in the extraction step, a summarization prompt (outlined in Appendix A.2) is passed to the LLM along with the full document text. This prompt aims to keep the summary concise while ensuring the key information a user would need for analysis remains. No further processing is done to the output of the LLM.

Table 3.3: Comparison of Entity Types Extracted by LLMs and NER Models

Entity Type	LLM	NER Model
Petitioners	✓	✓
Respondents	✓	✓
Judges	✓	✓
Court	✓	✓
Date of judgement	✓	All dates mentioned in text
ORG (organizations)	✓	✓
GPE (locations)	✓	✓
Provisions	✓	✓
Statutes	✓	✓
Precedents	✓	✓
Key Facts	✓	–
Type of Case	✓	–
Decision	✓	–
Summary	✓	–

3.5 Knowledge Graph Population and Querying

After the extraction and summarization using `GPT`, the entities are used to populate a knowledge graph in `Neo4j`. For some basic analysis, only the statutes and sections associated with each case are added to the graph as nodes. Information like the summary, court, decision, and date are added as properties of the graph, since they are not going to be used for querying. The statute and section entities are chosen to be graphed because they are central to understanding the legal arguments in each case. Statutes and sections form the foundation of legal reasoning, and mapping their connections to cases provides an easy way to uncover trends, relationships, and areas of frequent litigation.

By representing statutes and sections as nodes in a graph, it's possible to answer questions such as:

1. Which statutes are cited most frequently across cases?
2. How are specific sections of a statute used in different legal contexts?
3. What patterns emerge in cases that involve the same legal provisions?

Using a tool like `Neo4j` enables easy visualization and querying of these relationships. Edges in the graph represent connections between cases and the statutes or sections they reference, making it easier to trace legal precedents or discover clusters of related cases. Some interesting insights extracted are presented in Chapter 4.

Results

4.1 Measuring LLM Performance

4.1.1 Extraction

To measure the performance of the LLM on the extraction task, the output was compared to the NER extracted entities. Specifically, the accuracy measurements were calculated as follows:

- True Positive: The number of statutes that were identified by both the LLM and the NER results. These are the statutes that both the LLM and NER agree on.
- False Positive: The number of statutes that were identified by the LLM but were not present in the NER results. These are the statutes that the LLM identified incorrectly (relative to the NER results)
- The number of statutes that were present in the NER results but were not identified by the LLM. These are the statutes that the LLM missed.

The outputs were considered matching if and only if there was a direct match between the LLM and NER outputs. It is important to consider that the NER output, despite the deduplication methods, had a large number of duplicate entities. This was due to there being many variations of each entity, including abbreviations and misspellings. This means the accuracy measurements may not accurately capture the true performance of the LLM. Table 4.1 lists the accuracy measures (precision, recall, f1) of each of the various entities extracted by both the LLM and the NER model.

Table 4.1: Comparison of Metrics for Extracted Entities (LLM vs. NER)

Entity Type	Precision	Recall	F1-Score
GPE	0.2883	0.2026	0.2380
Judges	0.7706	0.4363	0.5571
ORG	0.2911	0.0339	0.0607
Petitioners	0.4722	0.3730	0.4168
Precedents	0.0665	0.0268	0.0382
Provisions	0.1786	0.0919	0.1214
Respondents	0.1329	0.0734	0.0945
Statutes	0.4479	0.2211	0.2960

The LLM output was also compared to manually labeled data from a small subset of the corpus (15 documents). Each document was read and only the statutes were manually

extracted. Even if there were multiple mentions of the same statute, the statute was counted only once. Table 4.2 shows the accuracy measurement scores for the LLM when compared to the ground truth. The LLM performs very well, matching the performance of the NER model as tested by Kalamkar et al. (2022). However, for a more accurate picture, the LLM should be tested on a larger corpus.

Table 4.2: Metrics for Extracted Statutes (LLM vs. Ground Truth)

Entity Type	Precision	Recall	F1-Score
Statutes	0.9615	0.909	0.9346

4.1.2 Summarization

To calculate the accuracy of the LLM-generated summaries, two standard measures, BERT and ROUGE-L, were used. Table 4.3 shows the average scores over the entire corpus. The BERT score of 0.624 represents a moderate amount of semantic similarity between the summaries and the documents. The ROUGE-L score represents structural similarity by considering the longest common subsequence between the document and its summary. The approach in Chowdhury et al. (2024) used a summary-of-summaries method to be able to fit documents in small context length, and achieved BERT and ROUGE-L scores of 0.83 and 0.21 respectively when using `GPT-3.5`. This shows that the summary-of-summaries method may generate better summaries, which may occur because the LLM has to condense less text per summary.

Table 4.3: Average BERT and ROUGE-L Scores Over the Corpus

Metric	Average Score
BERT	0.624
ROUGE-L	0.275

4.2 Knowledge Graph

The knowledge graph was created in `Neo4j`. Once all nodes (cases, statutes, provisions) were updated, the graph represented a total of 699 nodes and 1503 relationships. Figure 4.1 depicts the knowledge graph in its entirety. To make the nodes visible more clearly, Figure 4.2 shows the knowledge graph when limited to 100 relationships. Blue nodes represent cases, yellow nodes represent provisions, and red nodes represent statutes. It is clear from both figures that most of the case nodes are centered around one statute node. This is the *Food Safety and Standards Act, 2006*. It is naturally the most prominent node, given the cases were filtered to include the term *Food Safety* during data collection.

This knowledge graph can now be queried to extract insights from the corpus. While this project includes some example queries and their results, there are many possible insights that can be extracted from the data.

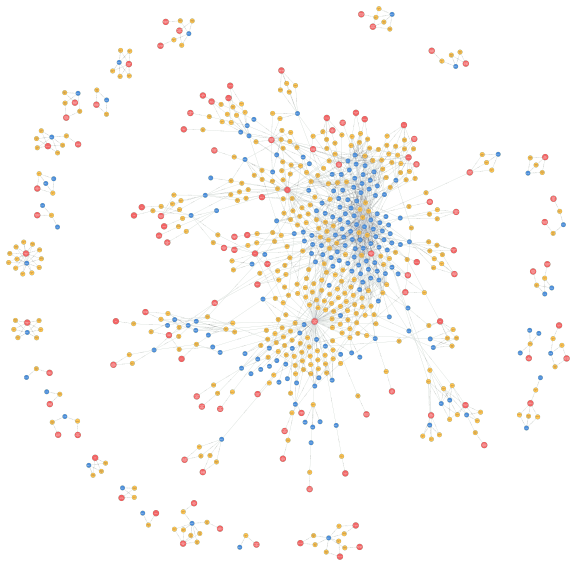

Figure 4.1: Knowledge graph generated from 196 food safety-related cases

4.3 Knowledge Graph Queries

This section presents some possible queries and their outputs. These include straightforward statistics about the corpus, such as the most common provisions cited, and also some interesting insights that can be extracted using more advanced techniques. Appendix B lists the `Neo4j Cypher` queries used to get these insights.

1. **Most commonly cited provisions**

   Table 4.4 outlines the most commonly cited provisions within the corpus. The *IPC* is the most common reference, taking up the first 4 rows, followed by Section 59 of the *Food Safety and Standards Act*. The only statute in the top 15 which is not the *Food Safety Act* or the *IPC* is the *Cigarettes and Other Tobacco Products (Prohibition of Advertisement and Regulation of Trade and Commerce, Production, Supply and Distribution) Act, 2003* (shortened to COTP).

2. **Similar cases**

   Figure 4.3 shows the three most similar cases based on the number of citations they have. These documents are from three judgements made together by the same judge for the same crime committed by 3 people. Queries like these can be valuable for identifying connections between related cases and understanding how legal principles are applied across similar judgments. In this specific example, the shared citations highlight the consistency in judicial reasoning when addressing the same incident involving multiple individuals.

3. **Similar cases - Jaccard similarity**

   Figure 4.4 shows the output when the graph is queried to calculate pairwise Jaccard similarity between case nodes, and return the top 10 most similar pairs along with the statutes and provisions associated with them. The interesting aspect of these cases is that they belong to distinct categories of legal matters. For instance, the top cluster of cases in the graph are all bail applications, while the cluster at the bottom right consists

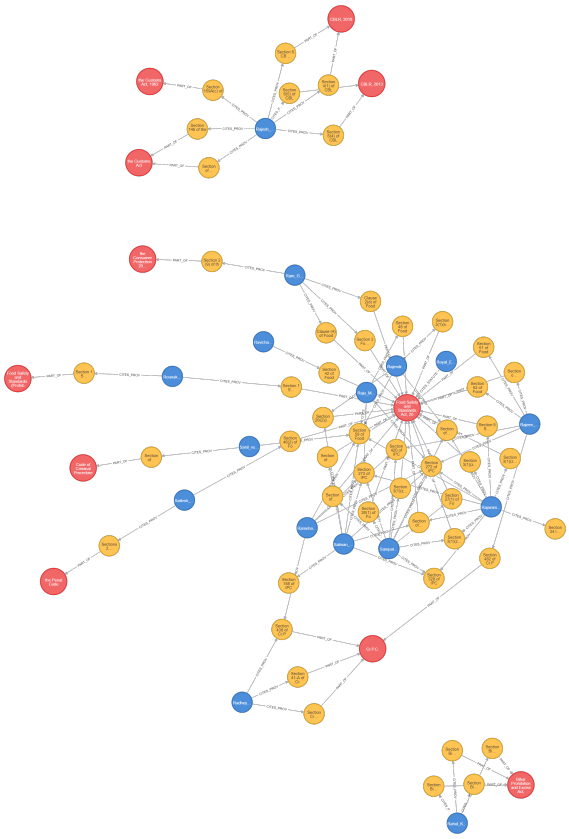

Figure 4.2: Knowledge graph limited to 100 nodes

Provision	**Citation Count**
Section 273 of IPC	76
Section 272 of IPC	63
Section 188 of IPC	49
Section 328 of IPC	49
Section 59 of Food Safety and Standards Act, 2006	47
Section 420 of IPC	33
Section 482 of Cr.P.C.	22
Section 63 of Food Safety and Standards Act, 2006	19
Section 34 of IPC	19
Section 270 of IPC	18
Section 59(i) of Food Safety and Standards Act, 2006	14
Section 26(2)(iv) of Food Safety and Standards Act, 2006	13
Section 51 of Food Safety and Standards Act, 2006	13
Section 7(1) of COTP Act	13
Section 58 of Food Safety and Standards Act, 2006	12

Table 4.4: Provisions and Their Citation Counts

of public interest litigations (PILs). This clustering demonstrates how cases with similar citations often share similar legal contexts or case types.

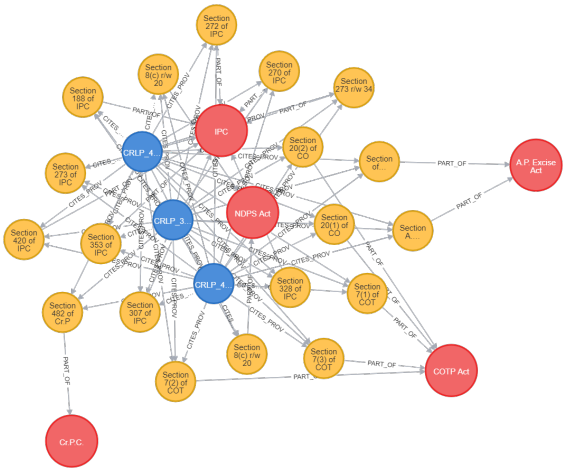

Figure 4.3: The three most similar cases based on common citation count

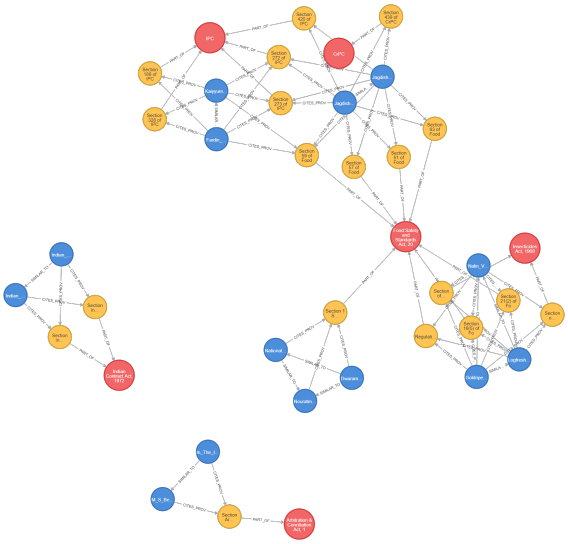

Figure 4.4: Top 10 most similar cases from pairwise Jaccard similarity

These examples show the range of insights that can be gained from querying the legal knowledge graph. From basic statistics like the most commonly cited provisions to more advanced queries identifying similar cases based on shared citations, the graph helps uncover meaningful patterns in legal data.

Beyond the examples presented here, there are many other types of queries that can be explored. For instance, one could investigate the co-occurrence of statutes across different cases to identify frequently paired provisions or analyze the timelines of citations to study how the relevance of certain laws evolves over time.

Conclusion and Future Work

5.1 Limitations and Future Work

While the current knowledge graph is a useful tool for analyzing food safety cases, expanding the corpus would enhance its utility further. Incorporating judgments from more years would enable the study of broader trends in food safety violations. For example, future analyses could explore:

1. Trends in the types of food violations and their frequency over time.
2. Regional variations in how food safety laws are enforced and interpreted.
3. Patterns in penalties imposed for specific offenses, providing insights into judicial consistency.
4. Connections between food safety violations and public health outcomes, using additional data sources.

Another area for future work is enhancing the knowledge graph to include more detailed relationships, such as links between parties, industries, and specific outcomes. This would provide a richer context for exploring the implications of legal decisions.

Although the current querying system is useful, it requires technical knowledge of graph databases and query languages. This limitation can be addressed by integrating a natural language querying interface into the system. A natural language interface would allow users to interact with the knowledge graph using plain language rather than specialized query languages like `Cypher`. For example, a user could ask, *"Which provisions of the Food Safety and Standards Act, 2006 are most frequently cited in judgments?"* or *"Show me cases where penalties were imposed under Section 59."* LLMs or other NLP methods could be used to translate natural language queries into query language.

5.2 Conclusion

This project explores the use of large language models (LLMs) and knowledge graphs for analyzing legal texts, particularly focusing on food safety-related cases. By integrating natural language processing techniques with graph-based representations, the project demonstrates a novel approach to structuring and analyzing legal data. The knowledge graph enables detailed querying and analysis by allowing querying that may range from identifying commonly cited statutes to exploring relationships between cases based on certain similarities. The insights gained highlight the potential of this methodology for automating labor-intensive legal research tasks and extracting valuable patterns from large corpora of judgments.

References

S. Chowdhury, S. Joshi, and L. Dey. Cross examine: An ensemble-based approach to leverage large language models for legal text analytics. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 194–204, Miami, FL, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.nllp-1.16. URL https://aclanthology.org/2024.nllp-1.16.

H. Darji, J. Mitrović, and M. Granitzer. German bert model for legal named entity recognition. In *International Conference on Agents and Artificial Intelligence*, 2023. URL https://api.semanticscholar.org/CorpusID:257356390.

C. C. Dozier, R. Kondadadi, M. Light, A. Vachher, S. Veeramachaneni, and R. Wudali. Named entity recognition and resolution in legal text. In *Semantic Processing of Legal Texts*, 2010. URL https://api.semanticscholar.org/CorpusID:17899708.

M. Honnibal and M. Johnson. An improved non-monotonic transition system for dependency parsing. In *Conference on Empirical Methods in Natural Language Processing*, 2015. URL https://api.semanticscholar.org/CorpusID:1267472.

Indian Kanoon. Indian kanoon - search engine for indian law, 2024. URL https://indiankanoon.org/. Accessed: 2024-12-06.

P. Kalamkar, A. Agarwal, A. Tiwari, S. Gupta, S. Karn, and V. Raghavan. Named entity recognition in Indian court judgments. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 184–193, Abu Dhabi, United Arab Emirates (Hybrid), Dec. 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.nllp-1.15. URL https://aclanthology.org/2022.nllp-1.15.

P. Laban, W. Kryscinski, D. Agarwal, A. R. Fabbri, C. Xiong, S. R. Joty, and C.-S. Wu. Summedits: Measuring llm ability at factual reasoning through the lens of summarization. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:266164172.

V. Leone, L. Di Caro, and S. Villata. Taking stock of legal ontologies: a feature-based comparative analysis. *Artificial Intelligence and Law*, 28(2):207–235, June 2019. ISSN 1572-8382. doi: 10.1007/s10506-019-09252-1. URL http://dx.doi.org/10.1007/s10506-019-09252-1.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

V. F. Pais, M. Mitrofan, C. L. Gasan, V. Coneschi, and A. Ianov. Named entity recognition in the romanian legal domain. In *NLLP*, 2021. URL https://api.semanticscholar.org/CorpusID:241583154.

L. Peng, Z. Wang, F. Yao, Z. Wang, and J. Shang. Metaie: Distilling a meta model from llm for all kinds of information extraction tasks. *ArXiv*, abs/2404.00457, 2024. URL https://api.semanticscholar.org/CorpusID:268819314.

D. Shu, H. Zhao, X. Liu, D. Demeter, M. Du, and Y. Zhang. Lawllm: Law large language model for the us legal system. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 4882–4889, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3680020. URL https://doi.org/10.1145/3627673.3680020.

E. Storti. Towards a knowledge graph representation of fair music content for exploration and analysis. In *TPDL Workshops*, 2022. URL https://api.semanticscholar.org/CorpusID:253161663.

T. van Sonsbeek, X. Zhen, and M. Warring. Knowledge graph embeddings for multi-lingual structured representations of radiology reports. *ArXiv*, abs/2309.00917, 2023. URL https://api.semanticscholar.org/CorpusID:261530892.

I. C. Wiest, F. Wolf, M.-E. Leßmann, M. van Treeck, D. Ferber, J. Zhu, H. Boehme, K. K. Bressem, H. Ulrich, M. P. Ebert, and J. N. Kather. Llm-aix: An open source pipeline for information extraction from unstructured medical text based on privacy preserving large language models. *medRxiv*, 2024. URL https://api.semanticscholar.org/CorpusID:272333918.

Prompts for Entity Extraction and Summarization

A.1 Entity Extraction

You will be provided with a legal judgement of a food safety-related case. Your goal is to extract key information following the schema provided. Please ensure that the extracted information is accurate and complete.

Any references to the Food Safety and Standards Act should be extracted as "Section x of the Food Safety and Standards Act, 2006" or "Food Safety and Standards Act, 2006".

Always use full forms of abbreviations, e.g. "Supreme Court" instead of "SC", or "Indian Penal Code" instead of "IPC".

Names of courts should be standardized and follow correct capitalization, e.g. "Supreme Court of India", "High Court of Delhi", NOT "Supreme Court", "Delhi High Court" or "DELHI HIGH COURT".

Here is a description of the parameters:

- court: name of the court that issued the judgement.

- petitioners: array of strings containing the names of ALL appellant(s) in the case.

- respondents: array of strings containing the names of ALL respondent(s) in the case.

- judges: array of strings containing the names of the judge(s) in the case.

- date: date of the judgement, as a string in the format "DD-MM-YYYY".

- org: array of strings containing the names of all organizations, companies, or government entities mentioned in the case, if any.

- gpe: array of strings containing the names of all geographical locations mentioned in the case, if any.

- provisions: array of strings containing the provisions of ALL statutes cited in the judgement. Provide these in the format "Section x of y". In case of references to multiple sections of the same statute, list them all separately.

- statutes: array of strings containing the names of ALL acts or laws cited in the judgement.

- precedents: array of strings containing the names of ALL precedents cited in the judgement.

- key_facts: key facts about the case. This should be very concise, and include the background of the case, and the main arguments made by both parties. If no information is provided, leave this field empty.

- type_of_case: the type of case, e.g. bail application, civil appeal, criminal appeal, public interest litigation, etc.

- decision: the decision of the court in the case, if provided. If there is a verdict, respond with 'in favour of appellant' or 'in favour of respondent'. If not, leave this field empty.

A.2 Summarization

You will be provided with a legal judgement of a food safety-related case. Your goal is to provide a detailed summary of the judgement. Only include information explicitly stated in the document. Do not infer, interpret, or add new information. Use concise language while preserving all critical legal details, such as statute names, provisions, case outcomes, and involved parties. Organize the summary logically, grouping related points. For example:

- Case overview

- Key facts

- Legal issues and arguments

- Court's reasoning

- Decision or judgment

Return plain text, do not include any markdown or HTML formatting.

Knowledge Graph Queries

This appendix lists the `Neo4j Cypher` queries used to extract the insights as presented in Chapter 4.

B.1 Most commonly cited provisions

```
MATCH (p:Case)-[:CITES_PROV]->(sec:Section)
WITH sec.name AS Provision, COUNT(p) AS CitationCount
ORDER BY CitationCount DESC
RETURN Provision, CitationCount
```

B.2 Similar cases based on common citations

```
MATCH (c1:Case)-[:CITES_PROV]->(sec:Section)<-[:CITES_PROV]-(c2:Case)
WHERE id(c1) < id(c2)
WITH c1, c2, COUNT(sec) AS sharedProvisionsCount
ORDER BY sharedProvisionsCount DESC
LIMIT 2
WITH COLLECT(c1) + COLLECT(c2) AS similarCases
UNWIND similarCases AS case
MATCH (case)-[r]->(n)-[r2]->(n2)
RETURN case, r, n, r2, n2
```

B.3 Jaccard similarity

```
MATCH (c1:Case)-[:CITES_PROV]->(sec:Section)<-[:CITES_PROV]-(c2:Case)
WHERE id(c1) < id(c2)
WITH c1, c2, collect(sec) AS sharedProvisions, count(sec) AS
    intersectionSize
MATCH (c1)-[:CITES_PROV]->(sec1:Section)
WITH c1, c2, sharedProvisions, intersectionSize, count(sec1) AS c1Size
MATCH (c2)-[:CITES_PROV]->(sec2:Section)
WITH c1, c2, sharedProvisions, intersectionSize, c1Size, count(sec2) AS
    c2Size
WITH c1, c2, sharedProvisions, intersectionSize, c1Size, c2Size,
     (intersectionSize * 1.0) / (c1Size + c2Size - intersectionSize) AS
    jaccardSimilarity
ORDER BY jaccardSimilarity DESC
LIMIT 10
WITH c1, c2, jaccardSimilarity, sharedProvisions
MATCH (c1)-[:CITES_PROV]->(sec1:Section)-[:PART_OF]->(stat1:Statute)
MATCH (c2)-[:CITES_PROV]->(sec2:Section)-[:PART_OF]->(stat2:Statute)
```

```
15 RETURN c1, c2, jaccardSimilarity, collect(DISTINCT sec1) AS Sections1,
       collect(DISTINCT stat1) AS Statutes1, collect(DISTINCT sec2) AS
       Sections2, collect(DISTINCT stat2) AS Statutes2
```