ClauseEase: Simplifying Legal Language Using Artificial Intelligence

Arnab Ghosh

1. ABSTRACT

Legal contracts are often filled with complex, jargon-heavy language that is difficult for non-lawyers to comprehend. This complexity can lead to misunderstandings, delays, and legal risks. **ClauseEase** aims to address this challenge by developing an AI-powered system that automatically extracts, analyzes, classifies, and simplifies legal contract clauses without losing their legal intent. Using natural language processing (NLP) and machine learning (ML), ClauseEase converts complex contract language into plain English while retaining accuracy and context. The system is designed to assist small businesses, clients, and legal professionals in understanding contractual terms more easily.

2. INTRODUCTION

Contracts are essential legal instruments but are often written in complicated and technical language that can be overwhelming to ordinary readers. Existing tools typically paraphrase content without maintaining legal precision. **ClauseEase** bridges this gap by providing a modular, automated system for simplifying and explaining legal contracts.

The project leverages AI technologies like **Legal-BERT**, **spaCy**, and **transformers** to identify clauses, detect legal terms, and generate simplified versions of each clause. This ensures accessibility and transparency while preserving legal integrity. The overall system has been divided into multiple modules, each performing a distinct function in the processing pipeline.

3. MODULES IMPLEMENTED

3.1 Document Ingestion Module

3.1.1 Objective:

Extract text data from legal contracts in **PDF** and **DOCX** formats.

3.1.2 Description:

This module acts as the input layer of ClauseEase. It extracts clean, structured text from uploaded contract files using libraries like PyMuPDF (fitz) and python-docx. The extracted text serves as the foundation for all further NLP tasks.

3.1.3 Technologies Used:

- Python
- PyMuPDF (for PDFs)
- python-docx (for DOCX files)

- os, glob (for file handling)

3.2 Text Preprocessing Module

3.2.1 Objective:
Clean, normalize, and segment the extracted text into meaningful clauses.

3.2.2 Description:

This module processes the raw text by removing unwanted characters, spaces, and formatting inconsistencies. It uses **regex** for pattern-based clause segmentation, **NLTK** for sentence tokenization, and **spaCy** for named entity extraction. The output is a structured set of clauses, each containing clean text, sentences, and identified entities — ready for analysis.

3.1.3 Technologies Used:

- Python
- re (regular expressions)
- nltk (sentence tokenization)
- spacy (named entity recognition)

3.3 Legal Clause Detection Module

Objective:
Automatically detect and classify clauses into legal categories using AI models.

Description:
This module employs the **Legal-BERT** model (nlpaueb/legal-bert-base-uncased) to understand and label each clause based on its type — such as **Confidentiality, Termination, Indemnity, Dispute Resolution, or Governing Law.** Using **transformers** and **PyTorch**, it tokenizes input clauses and predicts the most probable class. This step helps structure large contracts into labeled sections, simplifying analysis.

Technologies Used:

- transformers
- torch (PyTorch)
- scikit-learn
- numpy

Model Used:
`nlpaueb/legal-bert-base-uncased`

3.4 Legal Term Recognition Module

Objective:

Identify and highlight important legal terms or jargon inside each clause.

Description:

This module works at the **word or phrase level**, detecting specific legal terminology within the contract. It can be implemented in two ways:

1. **Using Pretrained Models (spaCy)** – Detects entities automatically (e.g., "Arbitration," "Jurisdiction").
2. **Using a Custom Legal Dictionary** – Uses pattern matching and regex to locate pre-defined legal terms and provide their meanings.

This feature is crucial for simplifying and explaining contracts in plain English later.

Technologies Used:

- Python
- re
- spacy (NER models)

3.5 Language Simplification Module

Objective:

Simplify complex legal clauses into plain, easily understandable English.

Description:

This module transforms complex legal text into simplified versions while preserving meaning and legal intent. It uses **transformer-based text generation models** such as **PEGASUS** or **BART**, accessed via the transformers pipeline.

Each clause is simplified sentence by sentence using the "text2text-generation" pipeline, producing concise and legally accurate paraphrases.

Technologies Used:

- transformers
- torch
- nltk (for sentence splitting)

Model Used:
`tuner007/pegasus_paraphrase`

Example Output:
Original:

"The Lessee shall indemnify and hold harmless the Lessor from any liability arising out of the use of the premises."
Simplified:
"The tenant must protect the owner from any losses caused by using the property."

4. Software Requirements

Python 3.10+	**:**	Programming Language
PyMuPDF	**:**	Extract text from PDF
python-docx	**:**	Extract text from DOCX
re	**:**	Text cleaning and regex pattern matching
nltk	**:**	Sentence tokenization
spacy	**:**	Named Entity Recognition
torch	**:**	Deep learning model support
transformers	**:**	Load and run pretrained NLP models
scikit-learn	**:**	Evaluation and data handling
numpy	**:**	Numerical operations

5. Conclusion

ClauseEase successfully integrates multiple natural language processing modules to automate the understanding and simplification of legal contracts. From text extraction to clause classification and simplification, each module plays a crucial role in transforming dense legal content into readable, structured, and accurate text.
The project demonstrates how **AI and NLP can bridge the gap between legal complexity and human comprehension**, ultimately empowering users to interpret contracts confidently without requiring legal expertise. Future improvements can include adding more clause categories, fine-tuning simplification models, and enhancing glossary-based contextual linking for better interpretability.