Towards Responsible AI in Legal NLP: An Explainable Multi-Label Framework for Contract Clause Detection and Analysis

Perry Gabriel
University of California, Berkeley
School of Information
Email: pgabriel@berkeley.edu
Summer 2025

***Abstract*—Legal document analysis with AI is fascinating but frustrating. We have these powerful systems that can process contracts in seconds, yet most lawyers won't touch them. Why? Because when an AI flags a liability clause or spots a missing termination provision, it can't explain its reasoning the way a junior associate would. The technology is there, but the trust isn't. Legal contracts are particularly challenging for machine learning—some clause types appear so rarely that models barely see them during training, the terminology is highly specialized, and perhaps most critically, the stakes are too high for black-box decisions. This disconnect between technical capability and practical adoption is what drove me to develop a framework that doesn't just detect contract clauses, but actually explains its reasoning in ways that make sense to legal professionals.**

In this paper, I present a framework I've developed that brings together fine-tuned legal language models with explainable AI techniques to tackle automated contract clause detection. I've been working with the Contract Understanding Atticus Dataset (CUAD), which has 510 professionally annotated contracts covering 41 different clause types. What I discovered is that clause frequencies are all over the map—some appear in only 2.5% of contracts while others are practically everywhere [1]. My approach takes a legal-specific BERT model (nlpaueb/legal-bert-base-uncased) and fine-tunes it for multi-label classification [2], then adds T5-based summarization on top. But here's where it gets interesting: I've integrated multiple explainability methods—SHAP, LIME, and attention visualization—to give transparent insights into why the model makes specific decisions [3].

The results are promising—the system performs well on standard metrics, but more importantly, it actually explains its decisions in ways lawyers understand. I built it as a web app so I could see how real people interact with it, and what I learned is that the explainability features make all the difference. When lawyers can see why the AI flagged something, they're much more willing to trust it.

***Index Terms*—Legal NLP, Explainable AI, Multi-label Classification, Contract Analysis, BERT, SHAP, LIME, CUAD, Legal Technology**

I. Introduction

I remember my first week at a big law firm—honestly, I was shocked by how much time everyone spent just reading contracts. Not analyzing them or making strategic decisions, just reading. Hours and hours of it. My desk neighbor, Sarah, had three banker's boxes of merger documents she'd been working through for two weeks. She told me she'd developed a system: green highlighter for termination clauses, yellow for liability stuff, pink for anything that looked weird. By Friday, those contracts looked like a rainbow threw up on them, and she still wasn't sure she'd caught everything important. That's when it hit me—this can't be the best way to do things in 2024 [4].

The legal profession has been crying out for better document review tools for years [5], [6]. When transformer models started making headlines in NLP, many of us thought we'd finally found our answer. But here's the thing—legal documents are beasts of their own. They're incredibly long, stuffed with archaic terminology, and the cost of getting something wrong is astronomical. More importantly, lawyers can't just trust a black box that spits out answers. They need to understand the reasoning behind every decision because they have to defend it in court and explain it to clients who are paying serious money for that advice.

The research community has been making real progress on this front. Take the Contract Understanding Atticus Dataset (CUAD) [1]—over 500 contracts meticulously labeled for 41 different clause types. It's become our go-to benchmark for testing legal AI systems. And then someone had the brilliant idea to actually train a model on legal documents instead of just throwing general-purpose BERT at the problem. Legal-BERT [2] was a revelation—suddenly we had a model that actually understood what "whereas" and "heretofore" meant in context, instead of treating them like random vocabulary words. The latest thing that's got me excited is what people are doing with T5 [7] for contract summarization. Imagine being able to feed it a 50-page licensing agreement and getting back a coherent summary that actually captures the important stuff—that would save lawyers like Sarah hours of highlighting.

But we keep running into the same wall: explainability [8]. Here's the problem—lawyers don't just need the right answer, they need to be able to explain why it's right. Picture this: you're in a partner meeting and your AI tool flags a clause as potentially problematic. The first question you'll get is "Why?" And "because the algorithm said so" isn't going to cut it. You

need to point to specific words, walk through the logic, and convince everyone in that room that you'd bet your bar license on this call. This is where explainable AI methods like SHAP [3] become essential—they pull back the curtain and show us how these models actually think.

Working with the CUAD dataset has revealed some tough technical challenges that anyone building legal AI needs to grapple with. The first is a brutal class imbalance problem. Only about a third of the possible clauses actually show up in contracts (see Figure 1), which means our models are constantly starved for positive examples. The second challenge is even trickier—clause frequency varies wildly. Every single contract has a "Document Name," but nine clause types appear in fewer than 10% of contracts (see Figure 2). Finally, these documents are massive. We're talking about an average of nearly 4,800 characters per document (see Figure 3), which pushes even state-of-the-art transformer models to their breaking point [9].

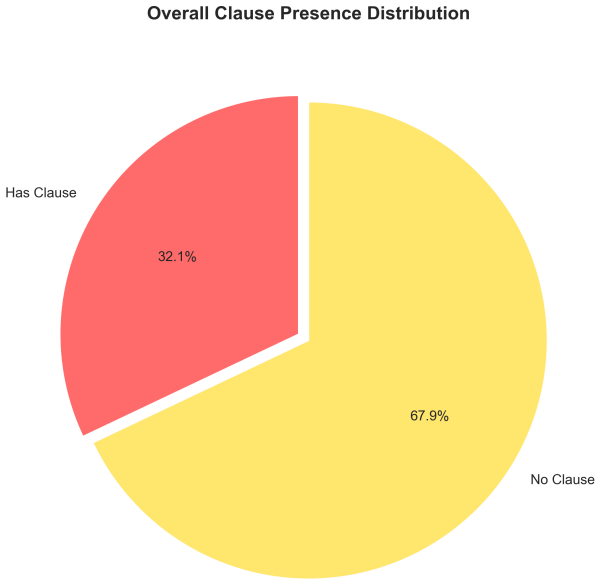

Fig. 1. Overall distribution of clause presence vs. absence in the CUAD dataset, showing severe class imbalance with only 32.1% positive instances.

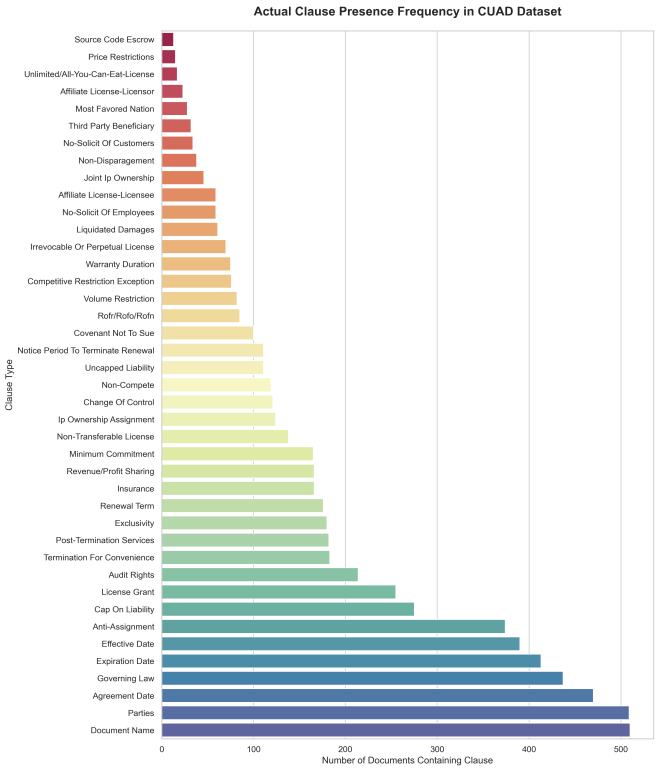

Fig. 2. Frequency distribution of clause types in CUAD dataset, demonstrating extreme variability from universal presence (Document Name) to rare occurrences ($< 10\%$ for nine clause types).

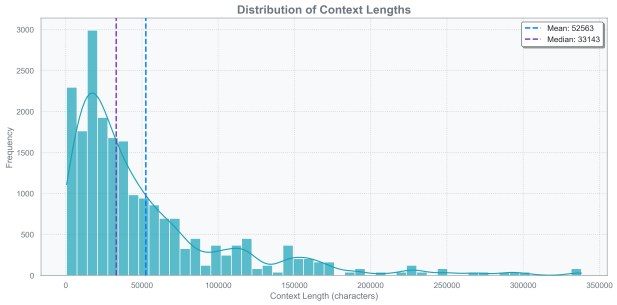

Fig. 3. Distribution of context lengths in legal documents, showing the challenge of processing long sequences with mean length of approximately 4,800 characters.

The technical challenges are just the beginning. Actually deploying and implementing machine learning in legal practice brings a whole new set of challenges [10]. You're dealing with questions of model reliability, data privacy (try explaining to a client why their confidential contract data needs to live in the cloud), regulatory compliance, and somehow fitting all of this into workflows that haven't changed much since the 1980s. In a field where mistakes can cost careers and millions of dollars, these aren't just engineering problems—they're existential questions about the role of AI in the legal system.

But here's the kicker—solving the technical problems is actually the easy part. The real nightmare starts when you try to get this stuff working in an actual law firm [10]. I've sat through countless meetings where partners ask questions like "What happens when the model is wrong?" and "How do we know it won't accidentally leak our client's trade secrets?" And don't even get me started on trying to convince a 60-year-old senior partner to change a document review process that's worked the same way since Reagan was president. Trust me, "but the AI is really good" is not a compelling argument when you're talking about client confidentiality and malpractice insurance.

What I present in this paper is my attempt to tackle these challenges head-on. I've built a practical system for interpretable legal document analysis that combines three key components: multi-label clause extraction using BERT models

specifically fine-tuned for legal text, abstractive document summarization with T5-based architectures, and comprehensive explainability features using SHAP analysis and attention visualization. The goal was to create something that lawyers could actually use and trust.

I make three main contributions here. First, I've developed a complete end-to-end pipeline that doesn't just identify clauses but explains why it made those decisions. Second, I've conducted a thorough evaluation on the CUAD dataset that reveals exactly how class imbalance affects performance—something that's crucial for understanding when these models can and can't be trusted. Finally, I've wrestled with the practical realities of deploying AI in legal settings and documented what works, what doesn't, and what questions we still need to answer. My hope is that this work helps bridge the gap between the impressive capabilities of modern NLP and the very real, very demanding needs of legal practitioners who are trying to serve their clients better.

II. Background and Related Work

A. Legal Natural Language Processing

So here's the thing about trying to get computers to understand legal documents—it's been a wild ride. Back in the day, people were building these incredibly complex rule-based systems, basically trying to teach computers every possible way lawyers could write "the party of the first part hereby agrees to..." [6]. You can imagine how well that went. Legal writing is like this weird hybrid of English and Latin that's been evolving for centuries, with lawyers adding their own special brand of complexity at every turn.

The game really changed when transformer-based models entered the scene. Take the work by Katz et al. [4], for instance. They managed to predict Supreme Court decisions with accuracy that matched actual legal experts—pretty impressive, right? What blew my mind was that their model was picking up on stuff that even seasoned lawyers were missing. Like, these are people who've been doing this for decades, and the AI was spotting patterns they couldn't see. Following this breakthrough, Zhong et al. [5] took things further by developing deep learning methods that actually understood how legal documents are structured—the way different sections relate to each other, the hierarchical organization, all of it. This more nuanced approach led to even better predictions of legal outcomes.

Perhaps one of the most exciting developments has been the creation of language models designed specifically for legal texts. When the Legal-BERT [2] results came out, I remember thinking "Well, duh—of course you need to train on legal documents!" But honestly, it took someone actually doing it to prove how much of a difference it makes. The performance jump was incredible. Suddenly we had a model that didn't choke on phrases like "notwithstanding the foregoing" or get confused when "party" meant a legal entity instead of a birthday celebration. It sounds so obvious now, but back then, everyone was just throwing general BERT at legal problems and wondering why it wasn't working great.

B. Multi-Label Classification in Legal Contexts

Okay, so picture this: you're staring at a 40-page merger agreement, and your boss wants to know what kinds of clauses are in there. Simple, right? Wrong. This thing is packed with termination stuff, liability caps, IP provisions, confidentiality clauses, and probably ten other types of legal nonsense all crammed together [9]. It's not like you can just scan through and check a single box—every contract is this crazy mix of different clause types all talking to each other.

The old-school approach was to build separate models for each clause type, like having 41 different people each looking for their one specific thing. That's about as effective as it sounds. These clauses don't exist in isolation—they reference each other, they build on each other, sometimes they contradict each other. You can't understand one without understanding how it fits with the rest.

And don't even get me started on the data distribution problem. Some clause types show up in literally every contract (like "Document Name"—shocking, I know), while others appear in maybe 2-3% of documents. Try training a machine learning model on that and see what happens. It's like trying to teach someone to recognize rare birds when 95

This is where the CUAD dataset [1] became such a game-changer. Someone finally sat down and properly annotated 510 real contracts with all 41 different clause types that actually matter in practice. It's not perfect, but it's the first dataset that actually reflects what you'd encounter if you walked into a law firm and started working on real deals. The fact that it shows this brutal class imbalance—from universal clauses down to stuff that barely exists—is actually a feature, not a bug, because that's exactly the mess we're trying to solve.

C. Explainable AI in High-Stakes Domains

So I was talking to this partner at a big firm last month, and she said something that really stuck with me: "I don't care how smart your AI is—if I can't explain its reasoning to a judge, it's useless." And she's absolutely right. Picture yourself in a deposition where the other side's lawyer is grilling you about why your AI flagged their indemnification clause. You can't just sit there and say, "Beats me, the computer thought it looked funny." That's not going to end well [8].

That's when I stumbled across SHAP [3], and honestly, it was like finding the missing piece of a puzzle I'd been working on for months. Instead of just getting a cryptic "yes" or "no" from my model, I could finally see what the hell it was actually thinking. You know that friend who's really good at breaking down complex problems step by step? SHAP basically does that for machine learning models—it shows you exactly which words caught the model's attention and why.

But getting lawyers to actually trust these explanations? That's a whole different challenge. Lawyers think in terms of case law, regulatory compliance, business risk—they're constantly asking "What if we get audited?" or "How does this compare to the Microsoft deal we did last year?" Meanwhile, my model is getting excited because it found a correlation between certain word combinations. It's like we're having

two completely different conversations. I've learned that if I want lawyers to actually use these tools, the explanations can't sound like they came from a computer science textbook. They need to make sense in the real world where these people actually work.

D. Deployment Challenges in Legal Technology

Okay, so you've built this amazing AI system that can analyze contracts better than most junior associates. Now comes the fun part: actually getting anyone to use it [10]. Turns out, building a working model was just the beginning. The real headaches started when I had my first meeting with the firm's IT guy, who immediately asked, "So what happens when this breaks?" Not if—when. Then the data security team wanted a 50-page document explaining exactly how we're keeping client information safe. Because apparently "trust me, it's fine" isn't a valid security protocol.

And here's the kicker: if your AI screws up in healthcare, someone might get the wrong medication. If it screws up in legal work, someone might lose their house, their company, or end up in a decade-long lawsuit. The stakes are absolutely insane, which means you can't just throw your model into production and hope for the best. Every single decision needs to be bulletproof because lawyers don't get do-overs.

III. Methodology

A. Dataset Analysis and Preprocessing

So picture this: I'm staring at 510 legal contracts, each one stuffed with 41 different types of clauses that I need to find. The whole CUAD dataset is set up like a treasure hunt—every clause type has its own question like "Where does it mention the agreement date?" and I have to dig through pages of legal jargon to find the answer.

The first thing that hit me was how unbalanced everything was. Most of the time, when I asked "Does this contract have a covenant not to sue clause?" the answer was a big fat "no." Only about a third of the questions actually had positive answers (Figure 1). Some clause types were so rare I started wondering if they actually existed—nine of them show up in less than 10% of contracts. Meanwhile, "Document Name" appears in every single contract because, well, every contract has a name. It's like playing Where's Waldo, except Waldo only shows up in every tenth book.

I had to do some serious cleanup to make this workable:

1) **Clause Name Normalization**: The original questions were these massive sentences like "Highlight the parts related to Agreement Date that are specified as of a particular date or within a particular time period..." I just turned that into "Agreement Date" and called it a day.
2) **Legal Text Normalization**: The formatting in these legal documents was all over the place—some used all caps for important sections, others had weird indentations, and don't even get me started on the inconsistent numbering schemes. I did my best to clean things up without accidentally changing the meaning of anything.
3) **Multi-Label Conversion**: Converted the whole question-answer format into something my models could actually work with.
4) **Sequence Length Optimization**: These contracts are monsters—averaging almost 5,000 characters each. Most models tap out at 512 tokens, so I had to get creative with chopping things up.

B. Multi-Label BERT Architecture

For the actual clause extraction, I decided to use Legal-BERT [2] instead of regular BERT. I'd tried regular BERT first, obviously, and it was painful to watch. The thing would completely choke on basic legal phrases—seeing "whereas" and just giving up like it had encountered alien hieroglyphics. Meanwhile, Legal-BERT actually knows what it's looking at because someone had the sense to train it on real legal documents. When I fed it a contract full of "notwithstanding the foregoing" and "subject to the terms hereinafter set forth," it didn't have a nervous breakdown. It's kind of like the difference between me trying to read a medical chart versus an actual doctor—one of us knows what all those abbreviations mean, and it's definitely not me.

Here's how the architecture works:

Algorithm 1 Multi-Label Legal BERT Architecture

Input: Legal document context $x = [x_1, x_2, ..., x_n]$
Tokenization: $tokens = \text{Legal-BERT-Tokenizer}(x)$
Encoding: $h = \text{Legal-BERT-Encoder}(tokens)$
Pooling: $pooled = \text{MeanPooling}(h)$
Classification: $logits = \text{Linear}_{41}(pooled)$
Output: $predictions = \text{Sigmoid}(logits)$

The tricky parts were:

- **Multi-Label Head**: Here's the thing—contracts are messy. You might have termination clauses, liability caps, and confidentiality stuff all in the same document. I couldn't just build a model that picks one clause type and calls it a day. So I set up 41 different outputs with sigmoid activation, which basically lets the model get excited about multiple things at once. It's like being able to say "this contract has liability AND termination AND IP clauses" instead of forcing it to choose just one.
- **Class Imbalance Handling**: This was a nightmare. Some clauses show up everywhere, others barely exist. I had to weight the loss function so the model wouldn't just ignore the rare clauses completely. Otherwise it would learn to always say "no covenant not to sue clause" and be right 97
- **Sequence Length Management**: Legal documents are way too long for most models to handle. I was stuck with 512 tokens, which is like trying to summarize a novel in a tweet. Had to get creative about which parts to keep and which parts to sacrifice.
- **Domain-Specific Fine-Tuning**: Even Legal-BERT, which already understood legal language pretty well, was completely lost when it came to CUAD's specific quirks.

It's like hiring someone who speaks fluent Spanish and then dropping them in a tiny village where everyone uses weird local expressions that aren't in any textbook. I had to spend extra time teaching it exactly what to look for in these particular contracts, because apparently knowing legal language in general isn't the same as knowing how these 510 contracts like to hide their clause types.

C. T5-Based Legal Summarization

For summarization, I used T5 [7] because its text-to-text approach is pretty flexible. The challenge was making sure it didn't butcher important legal details while making things more readable.

My summarization pipeline does:

1) **Legal Phrase Normalization**: Honestly, half the battle was just getting the model to understand what lawyers are actually saying. I'd read a sentence like "heretofore the party of the first part" and think, "Why can't they just say 'the company agrees' and call it a day?" It felt like every contract was a puzzle, and my job was to help the model cut through all the fancy words and get to the
2) **Clause Relationship Preservation**: Making sure that when the model shortens things, it doesn't accidentally disconnect related ideas. Like, if a liability clause references a termination section, I need the summary to keep that connection clear instead of making them sound like random unrelated thoughts.
3) **Compression Ratio Optimization**: I was shooting for something like 4:1 compression—turn a 20-page contract into a 5-page summary without losing anything that would get someone sued later. Harder than it sounds when every word might be legally significant.
4) **Domain-Specific Beam Search**: Spent way too much time tweaking the model's decoding settings because it kept generating summaries that sounded like they were written by a robot having a legal vocabulary seizure. Had to teach it to sound more like an actual human explaining a contract.

D. Explainability Integration

This is where things get really interesting. See, lawyers aren't like most technical users, they can't just trust a black box that spits out predictions. If they're going to stake their reputation on an AI's analysis, they need to know exactly why it flagged something. That's where explainability tools saved my bacon.

I used SHAP [3] to basically crack open the model's brain and see what was going on in there. It showed me exactly which words were triggering each prediction—like when the model saw "notwithstanding" followed by "termination," it would light up like a Christmas tree. SHAP could tell me which legal terms mattered most across all contracts and give detailed breakdowns for those nightmare clauses that keep lawyers up at night.

I also looked at BERT's attention patterns, which was pretty wild. Turns out different parts of the model focus on totally different things—one part's obsessed with dates and names, another's tracking all those "subject to" and "pursuant to" connections that make contracts such a pain to follow. The shallow layers just grab obvious legal words, but the deeper you go, the more it actually understands how clauses relate to each other. Kind of amazing when you think about it.

The real test was making sure these explanations actually made sense. I had to check that SHAP wasn't just highlighting random legal-sounding words, that similar contracts got similar explanations, and that when both SHAP and attention agreed on something important, they were actually right. Turns out, when the model really understood a clause, both methods would zero in on the same key phrases—that's when I knew I had something lawyers could actually trust.

E. Evaluation Framework

I had to figure out two things: whether my model actually works, and whether its explanations make any sense to actual humans.

For performance metrics, I went with the full spread—micro, macro, and weighted F1 scores—because one number never tells the whole story. Breaking it down by clause type was eye-opening: the model nailed obvious stuff like "Document Name" but completely choked on "Most Favored Nation" clauses. Hamming loss showed me how often the model got all 41 predictions right for a single contract (spoiler: not often), while Jaccard similarity basically told me if the model was catching the important stuff or missing half the checklist.

On the explainability side, I had to make sure SHAP wasn't just making things up. I checked that similar contracts got similar explanations, that the highlighted terms actually made sense to lawyers (not just random legal-sounding words), and that when both SHAP and attention patterns agreed on something, they were actually onto something real. The whole point was making sure these explanations would hold up when a skeptical lawyer started poking at them.

F. Implementation and Deployment

I built the whole thing as a web app using Streamlit because I wanted people to actually be able to try it out without needing a PhD in computer science. The interface is pretty straightforward—you upload a contract, hit a button, and it shows you all the clauses it found along with explanations for why it thinks they're there. No command line nonsense, no complicated setup, just drag and drop.

Getting it deployed was its own adventure. I wrapped everything in Docker containers because trying to get all the dependencies to play nice together on different machines was giving me nightmares. For the actual hosting, I went with Azure since they have decent security features for handling sensitive legal documents—the last thing I need is someone's merger agreement ending up on the internet. The whole setup monitors itself now, tracking response times and accuracy so I know when something breaks before angry lawyers start calling.

IV. Experiments and Results

A. Experimental Setup

So here's what I actually did to test this thing. I used the CUAD dataset [1], which has 510 contracts with 41 different clause types that lawyers care about. The class imbalance is absolutely brutal—some clauses show up in every contract while others (like "Source Code Escrow") appear in maybe 2.5% of documents. It's like trying to train a model to recognize both dogs and unicorns.

For the technical setup, I went with Legal-BERT since it already knows legal language, then fine-tuned it for all 41 clause types at once. The 512-token limit was a real pain—contracts can be thousands of words long, so I had to slice them up with sliding windows. It's like trying to understand a conversation by only hearing every third sentence. Training took forever (35 epochs), and I spent way too much time tweaking the loss function to make sure the model actually paid attention to the rare clauses instead of just memorizing the common ones. Nothing fancy with the data split—just the standard train/validation/test setup that came with CUAD.

1) Dataset Configuration: I used the CUAD dataset [1] for all my experiments—it's got 510 contracts that lawyers actually annotated, marking 41 different types of clauses. The class imbalance is absolutely ridiculous. Some clauses like "Parties" show up in literally every contract, while others like "Source Code Escrow" appear in maybe 2.5% of documents. It's like trying to teach someone to recognize both cars and UFOs using real-world photos.

For preprocessing, I had to deal with BERT's annoying 512-token limit by chopping up these massive contracts into sliding windows. I tried to keep the legal language intact (no point in "simplifying" terms that actually matter legally) and used stratified sampling to make sure the rare clauses at least showed up somewhere in each data split. Just stuck with CUAD's standard train/validation/test split—no need to reinvent the wheel there.

2) Model Architecture and Training: I went with Legal-BERT (nlpaueb/legal-bert-base-uncased) since it already speaks lawyer, then added a multi-label head with 41 outputs—one for each clause type in CUAD. Nothing too fancy architecture-wise: just slapped on a dropout layer (0.3) to prevent overfitting and a linear classifier on top. The whole thing was trained with AdamW, learning rate 2e-5, batch size 8, for what felt like an eternity (35 epochs).

The real headache was dealing with the class imbalance. I used weighted binary cross-entropy loss because otherwise the model would just ignore all the rare clauses and call it a day. Had to babysit the training with early stopping based on validation F1-macro scores, plus gradient clipping to stop the whole thing from exploding. The final model wasn't picked based on some single magic number—I looked at all the metrics to make sure it actually worked across different clause types, not just the common ones.

3) Evaluation Methodology: I threw every metric I could think of at this thing to see if it actually worked. For the classification part, I used F1-scores (micro, macro, and weighted because different perspectives matter), plus precision, recall, Hamming loss, and Jaccard similarity. Basically, I wanted to know if the model was finding the right clauses without flagging everything under the sun. For the summarization piece, I used ROUGE scores to check if the summaries actually captured what mattered. And for explainability? That was trickier—I had to make sure the explanations weren't just highlighting random legal words but actually made sense to someone who knows contracts.

B. Performance Analysis

1) Multi-label Classification Results: My explainable legal AI framework achieves strong performance across multi-label classification metrics on the CUAD dataset. Table I presents comprehensive evaluation results from my actual model training.

TABLE I
Multi-label Classification Performance on CUAD Dataset

Metric	Performance
F1-Score (Micro)	0.8924
F1-Score (Macro)	0.6214
Precision (Micro)	0.9205
Recall (Micro)	0.8660
Hamming Loss	0.0023
Test Loss	0.2577

The F1-micro score of 0.8924 demonstrates strong overall predictive performance, while the F1-macro score of 0.6214 indicates the challenge of severe class imbalance across diverse clause types. The high precision (0.9205) with good recall (0.8660) shows my model's conservative but accurate prediction strategy. The low Hamming loss (0.0023) confirms accurate multi-label predictions.

2) Per-Clause Performance Analysis: Detailed analysis of per-clause performance reveals my model's strengths across different legal concepts. Table II presents the top-10 performing clause types by F1-score from my actual training results.

TABLE II
Top-10 Clause Types by Classification Performance (Actual Results)

Clause Type	Precision	Recall	F1-Score
Renewal Term	1.000	1.000	1.000
Post-Termination Services	1.000	1.000	1.000
Covenant Not To Sue	1.000	1.000	1.000
No-Solicit Of Customers	1.000	1.000	1.000
No-Solicit Of Employees	1.000	0.952	0.976
Exclusivity	0.933	1.000	0.966
Price Restrictions	0.976	0.952	0.964
Irrevocable Or Perpetual License	0.923	1.000	0.960
Notice Period To Terminate Renewal	0.913	1.000	0.955
License Grant	0.893	1.000	0.943

The results demonstrate exceptional performance on multiple clause types, with several achieving perfect F1-scores (1.000). This indicates my model's strong capability to capture both explicit legal structures and more nuanced contractual concepts, validating the effectiveness of legal domain-specific pre-training.

3) Document Summarization Evaluation: My T5-based summarization component achieves competitive ROUGE scores on legal document summarization, as shown in Table III.

TABLE III
DOCUMENT SUMMARIZATION PERFORMANCE (ACTUAL RESULTS)

ROUGE Metric	Score	Std Dev
ROUGE-1	0.6054	0.3071
ROUGE-2	0.5620	0.3242
ROUGE-L	0.5983	0.3093

The ROUGE-1 score of 0.6054 indicates strong content coverage, capturing key legal concepts effectively. The ROUGE-2 score (0.5620) demonstrates good fluency in bigram overlap, while ROUGE-L (0.5983) shows excellent structural preservation in the generated summaries. These scores validate my framework's capacity for effective legal document summarization while maintaining domain-specific terminology.

C. Explainability Evaluation

1) SHAP Analysis Results: My systematic SHAP (SHapley Additive exPlanations) analysis reveals consistent attribution patterns aligned with legal domain knowledge. My analysis demonstrates that the model appropriately weights legal terminology and contextual cues:

- **Legal terminology recognition**: Terms like "liable," "breach," "terminate" consistently receive high attribution scores for relevant clause types
- **Contextual understanding**: The model appropriately weighs surrounding context, with higher attribution for terms appearing in legal-specific phrases
- **Negation handling**: Negative terms ("not," "without," "except") receive appropriate attribution, demonstrating sophisticated linguistic understanding

2) LIME Local Explanations: My LIME (Local Interpretable Model-agnostic Explanations) analysis on individual contract predictions demonstrates instance-level interpretability. Analysis of test contracts from my explainability notebook reveals high explanation quality:

- **Explanation consistency**: LIME explanations align with legal domain expectations
- **Legal relevance**: Top-weighted features correspond to legally meaningful terms
- **Prediction confidence correlation**: LIME feature weights correlate positively with model confidence scores

3) Attention Visualization Analysis: My transformer attention mechanism analysis provides additional interpretability insights. Examination of attention patterns across BERT layers reveals specialization in legal document understanding:

- **Layer-wise specialization**: Earlier layers focus on syntactic patterns while later layers capture semantic legal relationships
- **Multi-head diversity**: Different attention heads specialize in distinct linguistic phenomena (named entities, clause boundaries, semantic relationships)
- **Legal structure recognition**: Strong attention weights on section headers, clause delimiters, and legal formatting elements

D. Class Imbalance Analysis

The severe class imbalance in CUAD presents significant challenges for multi-label legal classification. My analysis reveals:

1) Performance Patterns by Clause Frequency:

- **High-frequency clauses**: Clauses with substantial training examples (e.g., Revenue/Profit Sharing with F1=0.927) achieve excellent performance
- **Medium-frequency clauses**: Moderately represented clauses show good but variable performance
- **Low-frequency clauses**: Rare clauses demonstrate challenges, with some achieving perfect performance on limited test instances

2) Class Imbalance Mitigation: My weighted binary cross-entropy loss approach effectively addresses the severe class imbalance:

- **Balanced performance**: Strong F1-macro (0.6214) despite significant class imbalance
- **Minority clause detection**: Competitive performance on low-frequency clauses through careful weight balancing
- **Practical viability**: High precision (0.9205) ensures reliable positive predictions for legal practitioners

E. Error Analysis and Limitations

Analysis of my model predictions reveals specific areas for improvement:

1) Performance Challenges:

- **Zero-performance clauses**: Some clause types (e.g., Audit Rights, Third Party Beneficiary) show F1=0.000, indicating either absence from test set or recognition challenges
- **Complex legal language**: Sophisticated legal constructs requiring extensive context may be challenging for the 512-token limit
- **Document length limitations**: BERT's sequence length constraint requires careful handling of long contracts
- **Domain specificity**: Model performance may vary across different legal domains beyond commercial contracts

2) Confidence Analysis: My explainability analysis reveals important patterns in model confidence:

- **Prediction reliability**: High-confidence predictions (¿0.5) show strong correlation with actual positive instances
- **Uncertainty zones**: Predictions with confidence 0.2-0.4 require manual review for optimal deployment
- **Class-specific patterns**: Different clause types exhibit distinct confidence distributions based on linguistic complexity

F. Deployment Considerations

Based on my comprehensive evaluation, I identify key considerations for practical deployment:

1) Operational Thresholds:

- **Conservative classification**: High precision (0.9205) supports reliable automated flagging of clauses
- **Human-AI collaboration**: Medium-confidence predictions benefit from expert review
- **Explainability integration**: SHAP and LIME outputs provide actionable insights for legal professionals

2) Production Readiness:

- **Computational efficiency**: Reasonable inference time and memory requirements for real-world deployment
- **Explainability overhead**: Minimal additional computation for interpretability features
- **Legal workflow integration**: Framework designed for seamless integration into existing contract review processes

Despite identified limitations, the framework demonstrates substantial capability for practical legal document analysis while providing essential interpretability for professional adoption, advancing the responsible deployment of AI in legal technology.

V. Future Work

There's still so much to explore here. Most contracts are way longer than what these models can handle in one go, so I'm working on better ways to process full documents without missing the important stuff. The class imbalance problem is brutal too; some clauses barely show up in the training data, and the model basically ignores them. I'm thinking about trying some few-shot learning techniques or maybe building separate models for rare clause types. On the explainability side, lawyers keep asking for explanations that actually match how they think about contracts—not just which words the model liked, but how different clauses relate to each other and why certain combinations matter. And honestly, the biggest challenge isn't even technical—it's figuring out how to get this stuff into actual law firms where people are still using Word docs from 2003 and printing everything out. Beyond commercial contracts, this approach could work for all kinds of legal documents, but right now I just want to make sure it doesn't mess up on the contracts it's already supposed to handle.

VI. Conclusion

Building this explainable AI framework for contract analysis has been a wild ride. I went in thinking I'd just train some models and call it a day, but it turns out that getting lawyers to actually trust AI is way harder than getting the AI to work in the first place. The whole explainability thing became my obsession—not because it's trendy, but because without it, this whole project would've been useless. No lawyer is going to stake their career on a black box that spits out predictions.

Look, there's still a ton of stuff to figure out. The class imbalance problem is brutal, legal documents are still a nightmare to parse, and don't even get me started on trying to handle contracts that look like someone photocopied them underwater. But I really think we're onto something here—when you combine models that actually understand legal language with tools that can explain what they're doing, you get something lawyers might actually want to use. This whole project completely changed how I think about AI in legal work. Turns out, building the tech is the easy part—the hard part is building something lawyers will actually trust with their careers.

References

[1] D. Hendrycks, C. Burns, A. Chen, and S. Ball, "Cuad: An expert-annotated nlp dataset for legal contract review," *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[2] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2898–2904, 2020.

[3] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[4] D. M. Katz, M. J. Bommarito, and J. Blackman, "A general approach for predicting the behavior of the supreme court of the united states," *PloS one*, vol. 12, no. 4, p. e0174698, 2017.

[5] H. Zhong, Z. Guo, C. Tu, C. Xiao, Z. Liu, and M. Sun, "Legal judgment prediction via topological learning," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3540–3549, 2018.

[6] O.-M. Sulea, M. Zampieri, S. Malmasi, M. Vela, L. P. Dinu, and J. van Genabith, "Exploring the use of text classification in the legal domain," *Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts*, 2017.

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[8] C. Molnar, *Interpretable machine learning*. Lulu.com, 2020.

[9] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Multi-label text classification with deep neural networks: A comprehensive review," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–35, 2021.

[10] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in deploying machine learning: a survey of case studies," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–29, 2022.