A Comparative Study of Transformer Architectures for Legal Clause Extraction

Student Name: Aman Jaiswal
Student Number: 240318897
Supervisor Name: Haris Bin Zia
Programme of Study: MSc. Artificial Intelligence

Abstract—**Automating contract review faces two key challenges: extreme document length and the "long tail" problem, where critical clause types are often rare. This dissertation conducts a direct comparison of three transformer architectures for this task, framed as question answering on the benchmark CUAD dataset. We evaluate three distinct architectures: a zero-shot decoder-only model (Gemini 2.5 Pro), a fine-tuned encoder-only model (LEGAL-BERT), and a fine-tuned encoder-decoder model (FLAN-T5). On a consistent, stratified data split, our experiments show that task-specific fine-tuning yields a significant performance advantage over the zero-shot approach. The fine-tuned encoder-decoder model (FLAN-T5) achieved the highest scores (F1 $75.91\,\%$, EM $73.44\,\%$), confirming its effectiveness for this task.[1] Further analysis reveals how clause frequency impacts performance and highlights architectural differences in handling multi-span answers and unanswerable questions. Our findings confirm that fine-tuning is essential for high-precision legal NLP and suggest that generative, encoder-decoder models are a highly promising direction for building auditable extraction systems.**

Index Terms—**Legal AI, Contract Analysis, Clause Extraction, Question Answering, Transformers, Fine-Tuning, Evaluation**

I. Introduction

Large-scale transformer models have fundamentally reshaped specialized fields, turning natural language processing from a theoretical discipline into a practical tool for expert workflows [3]. Legal technology has been particularly affected, as AI that can comprehend, analyze, and generate complex text promises to boost efficiency and mitigate risk. However, applying these models in a high-stakes domain like law requires a level of precision, auditability, and reliability that far exceeds the demands of general-purpose tasks. The central challenge, therefore, is not simply to deploy existing architectures but to rigorously test their fitness for purpose [4], [5].

At the heart of this challenge is the manual review of legal contracts, a foundational yet formidable task that is a critical bottleneck in legal operations. This process is complicated by dense legal language, varied formatting, and the classic "long-tail" problem, where many critical clause types are rare but carry significant weight. To guide progress, the research community developed the Contract Understanding Atticus Dataset (CUAD) [6], but a clear, comparative analysis of modern architectural approaches for this benchmark is lacking.

[1]**EM** is the percentage of QA items whose predicted answer exactly matches a gold answer after SQuAD-style normalisation; **F1** is the harmonic mean of token-level precision and recall against each gold answer (maximum across golds) [1], [2].

In practice, three dominant transformer architectures are applied to this problem. The first are **encoder-only** models (e.g., LEGAL-BERT), well-suited for extractive tasks that predict an answer's start and end position in the text [7]. The second are **encoder-decoder** models (e.g., FLAN-T5), which are trained to generate the answer as a new text string [8]. The third are large, **decoder-only** models (e.g., Gemini), which can perform the task in a zero-shot setting via carefully designed prompts.

This dissertation addresses the gap in comparative analysis by conducting a rigorous, quantitative experiment designed to answer three primary research questions:

RQ1: How do specialized, fine-tuned models compare against a general-purpose, zero-shot Large Language Model for legal clause extraction?

RQ2: Between fine-tuned architectures, does a generative, **encoder-decoder** model offer a performance advantage over a purely extractive, **encoder-only** model?

RQ3: What architectural weaknesses are revealed by domain-specific challenges, such as rare clauses, complex answer structures, and unanswerable questions?

The primary goal of this research is to answer these questions through a direct comparison of the three transformer families. The key contributions are:

- **Reproducible Benchmark:** A rigorous benchmark comparing three distinct AI architectures on a stratified, held-out test set from the CUAD dataset.
- **Performance Gap Analysis:** A quantitative analysis demonstrating the significant performance gap between specialized, fine-tuned models and a general-purpose zero-shot LLM for this specialized domain task.
- **Architectural Insight:** A specific finding that an encoder-decoder model (FLAN-T5) outperforms an encoder-only model (LEGAL-BERT), offering new insights into the optimal architecture for legal document analysis systems.
- **Diagnostic Analysis:** A detailed analysis of model failure modes, evaluating how each architecture handles rare clauses, multi-span answers, and unanswerable questions.

II. Related Work

This research connects legal technology with applied natural language processing, building on prior work in automated contract analysis by distinguishing our contribution through a specific focus on a precise, auditable evaluation paradigm.

A. Evolution of Clause Extraction Methods

Early systems for contract review relied on handcrafted rules and regular expressions. While interpretable, these methods are brittle and difficult to scale. Classical machine learning approaches like Support Vector Machines (SVMs) offered improvements but struggled to grasp the long-range context required for complex legal documents.

Transformer architectures [3] represented a fundamental change. The field has seen three dominant architectural families emerge: **encoder-only** models like LEGAL-BERT [7], which are effective for extractive QA; **encoder-decoder** models like FLAN-T5 [8], which generate answers textually; and large **decoder-only** models like Gemini, which can perform tasks in a zero-shot setting. Our study provides a direct comparison of these three families.

B. Benchmarks in Legal NLP

To properly situate this research, it is helpful to understand where the **CUAD** dataset fits within the broader ecosystem of legal NLP benchmarks. While CUAD is designed for extracting specific clause text, other datasets tackle different legal tasks. For instance, the **LEDGAR** dataset focuses on classifying entire provisions within contracts, a broader task than pinpointing exact clause language. The **ContractNLI** dataset introduces a more complex reasoning challenge, asking models to determine if a statement is logically supported by a contract clause. For a holistic measure of a model's legal understanding, comprehensive benchmark suites like **LexGLUE** and **LegalBench** combine numerous tasks and datasets. We chose **CUAD** for this study because it remains the standard for the practical, real-world task of verbatim clause extraction.

C. Methodological Approaches on CUAD

When reviewing prior work on CUAD, a key distinction is the evaluation methodology, which reflects different end-goals. Most high-performing systems, such as ConReader [9] and PeerDA [10], frame the task as clause identification and report recall-focused metrics like **AUPR** (Area Under the Precision-Recall curve). This approach is well-suited for a "do-not-miss-anything" review paradigm. In contrast, our study adopts the stricter question-answering metrics of **Exact Match (EM)** and token-level **F1-score**, prioritizing the precision and direct usability of the extracted text.

D. Handling Long Legal Documents

The significant length of legal contracts poses a challenge for standard transformers with fixed-length context windows. Naive truncation is ineffective, as it can sever clauses and harm recall. The literature proposes two primary solutions: models with sparse-attention mechanisms like Longformer [11], and hierarchical models like Hi-Transformer [12]. Our baselines use a sliding-window approach, a practical and widely used strategy for managing long contexts.

III. METHODOLOGY

Our methodology is designed to ensure a rigorous, reproducible, and fair comparison across the different model architectures. We frame clause extraction as a span-level question-answering (QA) task, supported by a systematic data processing and evaluation pipeline.

A. Dataset and Data Handling

This research uses the Contract Understanding Atticus Dataset (CUAD) v1 [6], the de facto benchmark for this task. It was created by The Atticus Project and legal professionals to foster research in automated contract review.

a) Dataset Structure: The dataset consists of 510 commercial contracts from the EDGAR database, expertly annotated for 41 distinct clause types. The task is structured as question-answering, where each clause type corresponds to a question (e.g., "What is the governing law?"). For a given contract, the answer is the exact text span of the clause. A crucial feature is that questions can be unanswerable if a contract does not contain the relevant clause, reflecting real-world review scenarios. The data schema is shown in Listing 1.

Listing 1: CUAD Dataset Schema.

```
{
  "context": "The full text of a legal contract. Lengths vary
             significantly, often exceeding standard model
             input limits.",

  "question": "A fixed prompt for each of the 41 clause types
              (e.g., 'Which law governs the agreement?').",

  "answers": "A list containing one or more gold answer strings
      .
             Answers may be disjoint (multi-span).",

  "is_impossible": "A boolean flag, true if the clause is
                   absent from the contract."
}
```

b) Data Splitting and Verification: We created a single, fixed **80/20 train-test split** from the 20,910 question-answer pairs in the dataset. To ensure the split was representative, we stratified it using a combined key of (`clause_type`, `is_impossible`), which preserves the distribution of both clause types and the answerable/unanswerable balance. For model selection during training, 10% of the training data was held out as a validation set. Our verification analysis of the final splits confirmed a maximum distributional difference of just **0.012%** for any category. All experiments use the fixed test set for reproducibility. Two items that caused API errors during zero-shot inference were excluded from all model evaluations to maintain alignment, resulting in a final test set of $N = 4,180$.

B. Experimental Pipeline

The experimental pipeline consists of six stages:

1) **Data Ingestion:** We load and flatten the CUAD dataset into the standardized format shown in Listing 1.
2) **Text Normalization:** We perform light text cleaning while preserving original casing. SQuAD-style normalization is applied only by the evaluation scorer.

3) **Long-Document Chunking:** To manage long contracts, we use a sliding-window strategy, splitting contexts into overlapping chunks.
4) **Model-Specific Preprocessing:** Each model family receives a specifically formatted input (e.g., tokenized pairs for the encoder-only model, instruction-formatted strings for the encoder-decoder model).
5) **Inference and Post-processing:** Predictions are generated, and minimal post-processing (e.g., trimming whitespace) is applied.
6) **Evaluation:** We compute EM and F1 scores using a SQuAD-v2-compliant script.

IV. Model Implementation

We implemented one model from each of the three main transformer families. To ensure a fair comparison, all models were evaluated using an identical interface for data-handling and evaluation. This section details the configuration for each model.

A. Zero-Shot Model: Decoder-Only (Gemini 2.5 Pro)

This approach requires no fine-tuning. Each question-context pair is formatted as a strict prompt that requests either an exact copy of the clause or a specific marker if the clause is absent.

Listing 2: The zero-shot prompt template.

```
Task: Extract the {CLAUSE-TYPE} clause verbatim from the
    context.
If absent, return exactly: Clause not found

Question: {question_text}
Context: {contract_text_chunk}

Rules:
- Return only the clause text (verbatim) or the sentinel.
- No paraphrasing, no labels, no quotes, no extra words.
- If multiple fragments form the clause, copy them contiguously
  as they appear; do not invent text.
```

TABLE I: Inference policy for Gemini 2.5 Pro (zero-shot).

Parameter	Value
Prompting	Strict template in Listing 2
Context handling	Overlapping chunks; deterministic top-1 selection
Decoding	Deterministic (temperature=0)
Post-processing	Trim whitespace only

B. Fine-Tuned Extractive Model: Encoder-Only (LEGAL-BERT)

This model uses a domain-adapted encoder with a question-answering head to predict the start and end indices of an answer within the text. During training, context windows that do not contain a correct answer serve as explicit negative examples, which helps calibrate the model's ability to identify unanswerable questions.

TABLE II: Final training settings for LEGAL-BERT (extractive).

Parameter	Value
Backbone	`legal-bert-base-uncased` [7]
Input window (train)	Max length 512; stride 128; offset maps retained
Input window (test)	Max length 384; stride 128
Batch / precision	16 per-device (effective 64 with accumulation); fp16
Optimizer / LR	AdamW; 2×10^{-5}
Epochs / selection	3; best by validation F1

C. Fine-Tuned Generative Model: Encoder-Decoder (FLAN-T5)

This sequence-to-sequence model is fine-tuned to generate the answer string (or the "not found" marker) directly from an instruction-based input. Using short answer targets during training encourages the model to generate concise spans without including irrelevant text, while a small beam size for inference improves stability.

TABLE III: Final training and inference settings for FLAN-T5 (generative).

Parameter	Value
Backbone	`flan-t5-base` [8]
Input format	Instruction + question + chunk
Input/target caps	512 / 32 (train); 512 / 256 (test)
Batch / precision	8 per-device (effective 16 with accumulation); bf16
Optimizer / LR	AdamW; 3×10^{-5}
Epochs / selection	3; best by validation F1
Decoding (test)	Beam search (4 beams); minimal post-processing

V. Performance Evaluation

We designed our evaluation protocol to be strict, transparent, and aligned with best practices in question-answering research, prioritizing the precision and usability of the extracted text.

A. Evaluation Framework and Metrics

A key distinction in prior work on CUAD is the evaluation methodology. Most benchmark papers frame the task as clause identification and report recall-focused metrics like AUPR. In contrast, our study adopts the standard framework from question-answering for its focus on auditability. We chose this for three reasons: (1) it directly measures whether the text returned to a legal reviewer is precisely correct and ready to use; (2) it provides a strict, unambiguous criterion for unanswerable questions; and (3) it aligns with widely understood QA evaluation tools.

Our two primary metrics, calculated using a SQuAD-v2-compliant scorer, are:

- **Exact Match (EM):** The percentage of predictions that match one of the ground-truth answers exactly after normalization.

- **Token-level F1:** The harmonic mean of token-level precision and recall, computed against each ground-truth answer and taking the maximum.

For unanswerable questions, a prediction is scored as correct (EM=1, F1=1) only if it matches the specific ”not found” marker, `Clause not found`.

B. *Experimental Protocol*

We evaluated all models on the same held-out test set ($N = 4,180$). To ensure a fair comparison, the data-processing, chunking, and normalization procedures were identical for all models. Decoding settings for the fine-tuned models (e.g., beam size) were fixed based on validation set performance, while we used deterministic settings for the zero-shot model to ensure reproducibility.

VI. Results and Analysis

This section presents the quantitative results of our study. We first report the overall performance of our three models to directly answer our primary research questions. We then provide a detailed diagnostic analysis of their architectural behaviors, followed by a comparison that situates our findings within the context of existing systems.

A. *Overall Performance (RQ1 and RQ2)*

Our primary results reveal a clear performance hierarchy among the three architectural families, directly answering RQ1 and RQ2. As shown in Table IV and Figure 1, both fine-tuned models substantially outperform the strong zero-shot LLM baseline. Furthermore, the generative, encoder-decoder model (FLAN-T5) achieves the highest performance, surpassing the extractive, encoder-only model (LEGAL-BERT) by over 4 EM points and 5 F1 points.

TABLE IV: Overall results of our three models on the CUAD test split, scored with SQuAD-style EM and F1.

Model	Architecture Family	EM (%)	F1 (%)
FLAN-T5	**Encoder-Decoder**	**73.44**	**75.91**
LEGAL-BERT	Encoder-only	69.40	70.74
Gemini 2.5 Pro	Decoder-only (zero-shot)	57.66	67.98

These findings lead to two key conclusions. First, for high-precision domains like legal text, task-specific fine-tuning remains critical. Second, the generative text-to-text approach appears better suited for this task than purely extractive methods, likely due to its ability to generate clean spans without including adjacent, irrelevant text.

B. *Diagnostic Analysis (RQ3)*

To address RQ3 and understand the underlying architectural behaviors, we conducted several diagnostic analyses of model failure modes.

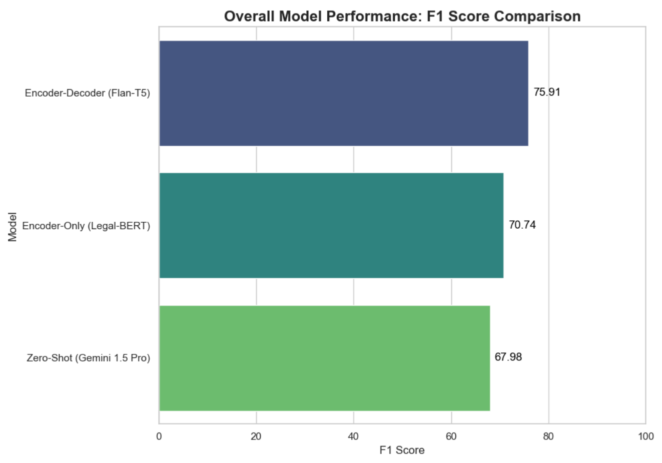

Fig. 1: Visual comparison of overall F1 scores. The results highlight the significant performance advantage of fine-tuning and the superiority of the encoder-decoder architecture for this task.

a) The Long-Tail Challenge: The CUAD dataset has a classic long-tail distribution, where a few clause types are common while many are rare (Figure 2). This directly impacts model performance (Figure 3). All models show a clear correlation between the number of training examples for a clause and its resulting F1 score. Clauses with fewer than 50 training examples, such as *Source Code Escrow*, remain a significant source of errors.

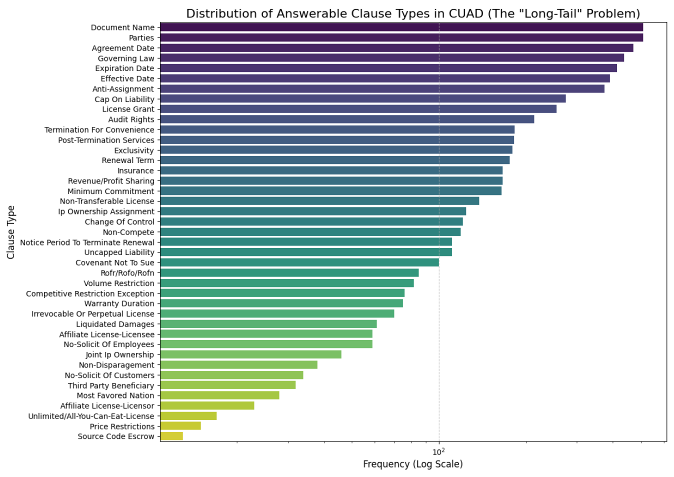

Fig. 2: Distribution of answerable clause types in the CUAD training set.

b) Architectural Behavior on Answer Structure: We analyzed how each architecture handles different answer structures (Figure 4). The encoder-only LEGAL-BERT model, which can only predict one contiguous span, is inherently disadvantaged on multi-span questions (e.g., a list of parties). In contrast, the zero-shot LLM performs relatively well, as it can synthesize lists. FLAN-T5 shows a significant performance drop on multi-

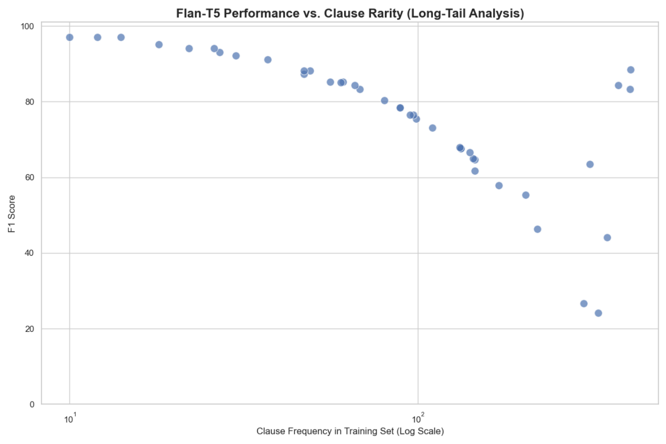

Fig. 3: FLAN-T5 F1 score versus clause frequency in the training set.

span items, suggesting its fine-tuning process optimized for the more common single-span case.

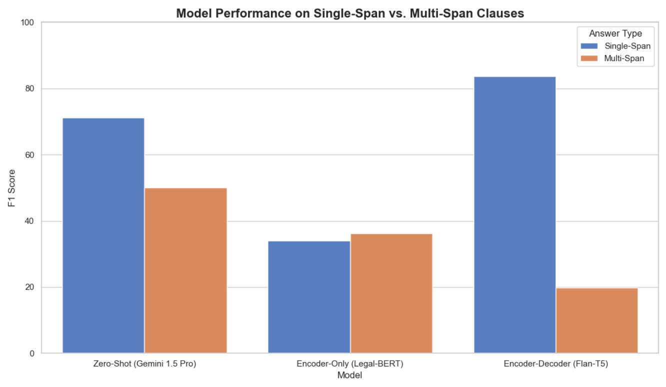

Fig. 4: F1 score comparison on single-span versus multi-span questions.

c) Reliability on Unanswerable Questions: A model's ability to identify when a clause is absent is as critical as its ability to find one that is present. Figure 5 shows that the fine-tuned models achieve extremely high True Negative (TN) rates, correctly identifying absent clauses over 96% of the time. This precision, however, comes at the cost of a higher False Negative (FN) rate, where they miss clauses that are present. The zero-shot LLM shows a more balanced, though less precise, profile, highlighting a classic precision-recall trade-off.

C. Comparison with Existing Systems

To situate our results within the broader literature, we compare them to prior work. It is important to note that a direct numerical comparison is often inappropriate, as many systems are evaluated with a different, recall-focused paradigm.

Table V shows results for high-performing systems like PeerDA that use the AUPR metric. These scores reflect the state-of-the-art for the clause identification task. For a more direct comparison on the F1 metric, Table VI presents scores

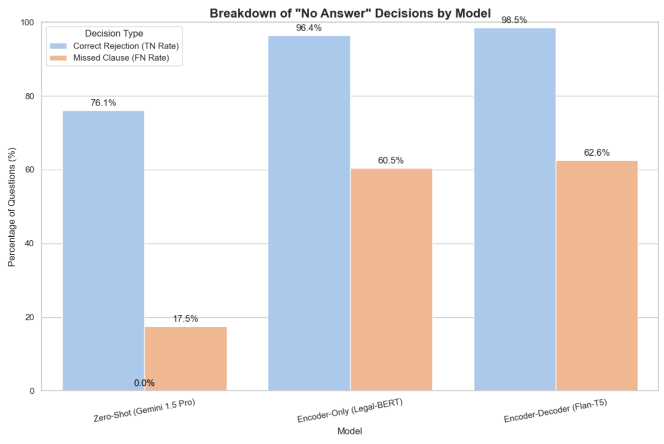

Fig. 5: Breakdown of ”No Answer” decisions.

for several large language models **as reported by Liu et al. in their ContractEval study** [13]. These models were not part of our experiment. Our fine-tuned FLAN-T5's F1 score of **75.91%** is highly competitive against these cited zero-shot results, demonstrating the powerful effect of task-specific fine-tuning.

TABLE V: Performance of prior systems on CUAD using recall-focused metrics. These were not part of our experiment.

Model (from prior work)	AUPR	P@0.8R
RoBERTa-base [9]	43.2	32.2
ConReader-large [9]	49.1	44.2
PeerDA (RoBERTa-base) [10]	**52.3**	**45.5**

TABLE VI: F1 scores of LLMs on CUAD, as reported in the ContractEval paper [13]. These were not part of our experiment.

LLM (from prior work)	F1
GPT-4.1	0.641
GPT-4.1 mini	0.644
Claude Sonnet 4	0.523
Gemini 2.5 Pro (Preview)	0.497

Conclusion

Our rigorous, direct comparison of three dominant transformer architectures for legal clause extraction has delivered a clear verdict: **task-specific fine-tuning is indispensable** for achieving the precision required in legal document analysis.

The generative, **encoder-decoder** model **FLAN-T5 was the top performer**. Its ability to generate clean, well-defined text spans proved to be a significant advantage over extractive, **encoder-only** models like LEGAL-BERT. While extractive methods guarantee the output is an exact copy from the source text, they risk including irrelevant surrounding language or failing on clauses split into multiple parts. Zero-shot, **decoder-only** LLMs provide a powerful baseline, but their tendency toward higher variance and format drift makes them less reliable for production systems where auditable, exact accuracy is non-negotiable.

Ultimately, this work provides empirical evidence that complements the existing CUAD literature. By prioritizing auditability-focused metrics (EM and F1) over recall-oriented ones (AUPR/P@0.R), we offer a practical guide for practitioners developing robust and trustworthy legal AI systems.

Future Work

The findings from this study highlight several promising avenues for future research to address the remaining challenges in automated contract review:

- **Addressing the Long-Tail Problem:** The performance drop on rare clauses is the most significant limitation. Future work should investigate advanced **data-augmentation** techniques, such as those explored by PeerDA [10], to create synthetic training data for low-resource clause types. Furthermore, exploring **parameter-efficient fine-tuning (PEFT)** methods could enable effective adaptation without the cost of full model training.
- **Improving Long-Context Handling:** Our use of a sliding-window approach is practical but can introduce errors at chunk boundaries. A key next step is to evaluate models with **dedicated long-context architectures** (e.g., Longformer). A highly promising alternative is to develop a **Retrieval-Augmented Generation (RAG)** pipeline, where a specialized retriever first identifies relevant passages to feed to the generative model for precise extraction.
- **Exploring Hybrid Systems:** Our analysis suggests that an optimal system might combine architectural strengths. Future research could evaluate **two-stage hybrid models** that use a high-recall model for initial candidate retrieval, followed by a high-precision generative model like our fine-tuned FLAN-T5 to produce a final, review-ready answer.
- **Investigating Prompt Engineering:** Our zero-shot evaluation relied on a single, fixed prompt to ensure a controlled comparison of architectures. Future work could explore the impact of **prompt engineering**, analyzing how different instructions or the inclusion of few-shot examples could further enhance the performance of large, decoder-only models on this task.

Acknowledgment

The author thanks Supervisor Haris Bin Zia for invaluable guidance and support throughout this research project.

References

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv:1606.05250*, 2016.

[2] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *ACL*, 2018.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[4] I. Chalkidis, A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. M. Katz, and N. Aletras, "Lexglue: A benchmark dataset for legal language understanding in english," in *ACL*, 2022.

[5] N. Guha, J. Nyarko, D. E. Ho, C. Ré, A. Chilton *et al.*, "LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models," *arXiv:2308.11462*, 2023.

[6] D. Hendrycks, C. Burns, A. Chen, and S. Ball, "CUAD: An expert-annotated NLP dataset for legal contract review," *NeurIPS Datasets and Benchmarks*, 2021, arXiv:2103.06268.

[7] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The Muppets straight out of law school," in *Findings of EMNLP*, 2020, pp. 2898–2904.

[8] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani *et al.*, "Scaling instruction-finetuned language models," *arXiv:2210.11416*, 2022.

[9] W. Xu, Y. Deng, W. Lei, W. Zhao, T.-S. Chua, and W. Lam, "Conreader: Exploring implicit relations in contracts for contract clause extraction," in *EMNLP*, 2022, pp. 2581–2594. [Online]. Available: https://aclanthology.org/2022.emnlp-main.166.pdf

[10] W. Xu, X. Li, Y. Deng, W. Lam, and L. Bing, "PeerDA: Data augmentation via modeling peer relation for span identification tasks," in *ACL*, 2023. [Online]. Available: https://aclanthology.org/2023.acl-long.484.pdf

[11] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv:2004.05150*, 2020.

[12] C. Wu, F. Wu, T. Qi, and Y. Huang, "Hi-transformer: Hierarchical interactive transformer for long document modeling," *arXiv:2106.01040*, 2021.

[13] S. Liu, Z. Li, R. Ma, H. Zhao, and M. Du, "Contracteval: Benchmarking llms for clause-level legal risk identification in commercial contracts," *arXiv:2508.03080*, 2025, uses CUAD test set; reports F1 and Jaccard for 19 LLMs.