Data Science Challenge Report

Subject:

Multi-Labels Classification for Legal Document

Completed on
Jun 24, 2025

Prepared by
Chuanliang Jiang

1. Introduction

Legal professionals routinely need to understand *how* and *why* a case has reached its current stage before a court. This summary—known as

the **procedural posture**—captures the decisions that preceded the appeal and anchors much of the legal reasoning that follows.

The objective of this report is to provide AI-driven annotated court opinions

1. **Characterize the dataset** – Quantify the number of documents, the set of unique procedural posture labels, and the total paragraph count. Highlight key dataset features such as label imbalance, variability in section length, and presence of overlapping postures, all of which may affect downstream modeling.
2. **Prototype an automated labeler** – Build an exploratory model that predicts procedural posture labels for previously unseen text. Evaluate the model using appropriate multi-label classification metrics (e.g., F1-score, ROC-AUC, Precision-Recall AUC,, hamming loss) and understand limitations in generalization.
3. **Additional experiments to improve/validate the model** – Explore alternative promising modeling strategies such as summarization, Retrieval Augmentation Generation(RAG), Instruction Fine-Tuning and Multi-Agent system for the next step model enhancement.

This report follows those objectives. It begins with an overview of the dataset and exploratory analysis, then details the modeling approach and evaluation, and finally presents recommendations and an implementation roadmap aimed at both technical and non-technical audiences.

2. Data Analysis and Visualization

*(All of the following analysis can be found in the notebook: **data_analysis_q1.ipynb**)*

Unzipping the `TRDataChallenge2023.zip` archive produces a single text file in **JSON Lines** format. Each line represents a JSON object with three primary keys: `documentId`, `postures`, and `sections`.

An example entry from the dataset is shown below:


```
{
  "documentId": "Ib4e590e0a55f11e8a5d58a2c8dcb28b5",
  "postures": [
    "On Appeal"
  ],
  "sections": [
    {
      "headtext": "",
      "paragraphs": [
        "Plaintiff Dwight Watson (“Husband”) appeals from the trial court’s equitable distributior
      ]
    },
    {
      "headtext": "Background",
      "paragraphs": [
        "Husband and Wife were married in November 19..."
      ]
    }
  ]
}
```


The table below presents the descriptive statistics for the primary quantitative attributes of the TR Data Challenge 2023 dataset. The dataset comprises a total of **18,000** documents, as indicated by the count value in the table. Each document is annotated with one or more posture labels (average is around 1.5 per document), reflecting the typical **multi-label** nature of the classification task. The cumulative number of posture assignments across all documents is **27,659**, as summarized by the sum of the num_postures column.

	num_postures	num_sections	text_length	word_count	num_paragraphs	num_headers
count	18000.00	18000.00	18000.00	18000.00	18000.00	18000.00
mean	1.54	5.09	17986.55	2891.67	30.12	4.45
std	0.75	5.73	19491.45	3119.72	34.08	5.88
min	0.00	1.00	0.00	0.00	0.00	0.00
25%	1.00	1.00	5642.25	905.00	8.00	0.00
50%	1.00	4.00	12809.50	2062.00	21.00	3.00
75%	2.00	7.00	23892.75	3838.25	40.00	7.00
max	7.00	91.00	785135.00	124134.00	1174.00	91.00

Total number of docs	18,000
Total number of postures	27,659
Total number of paragraphs	542,169

In addition, the dataset contains a total of **542,169** paragraphs, as shown by the sum of the num_paragraphs column. These paragraphs collectively form the full textual content of each legal document. The table also provides further insights into the distribution of text length(number of characters), word count, and the number of section

headers per document, offering a comprehensive overview of the dataset's structure and complexity.

These statistics are instrumental in guiding subsequent data preprocessing, model selection, and evaluation strategies, ensuring that the unique characteristics of the dataset are appropriately addressed throughout the analysis pipeline.

2.1. Multi-Label Postures Distribution Analysis

The Postures Label Analysis section explores the distribution and characteristics of procedural posture labels in the judicial opinions dataset. This analysis is crucial for understanding the complexity of the classification task and for informing model development and evaluation.

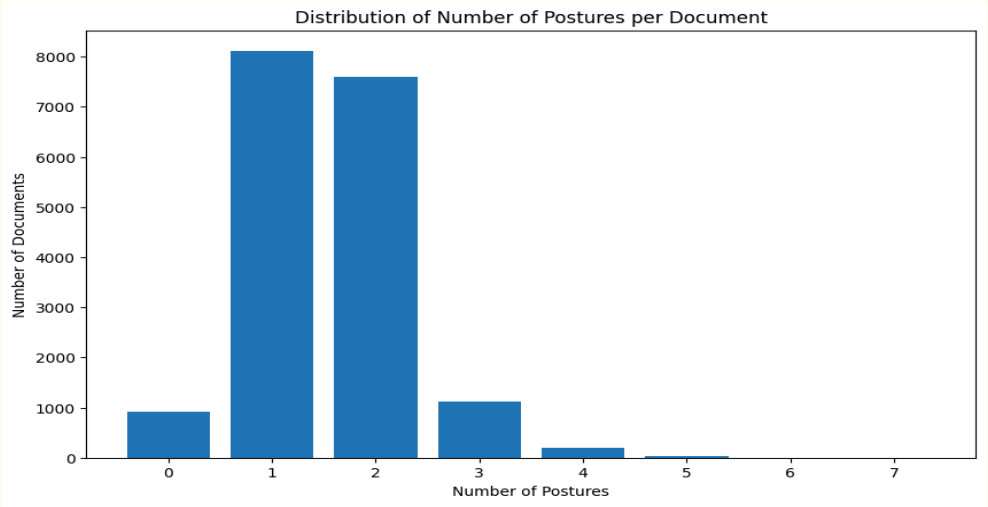
Based on the distribution charts below examining posture frequency and prevalence patterns, the following key insights emerge:

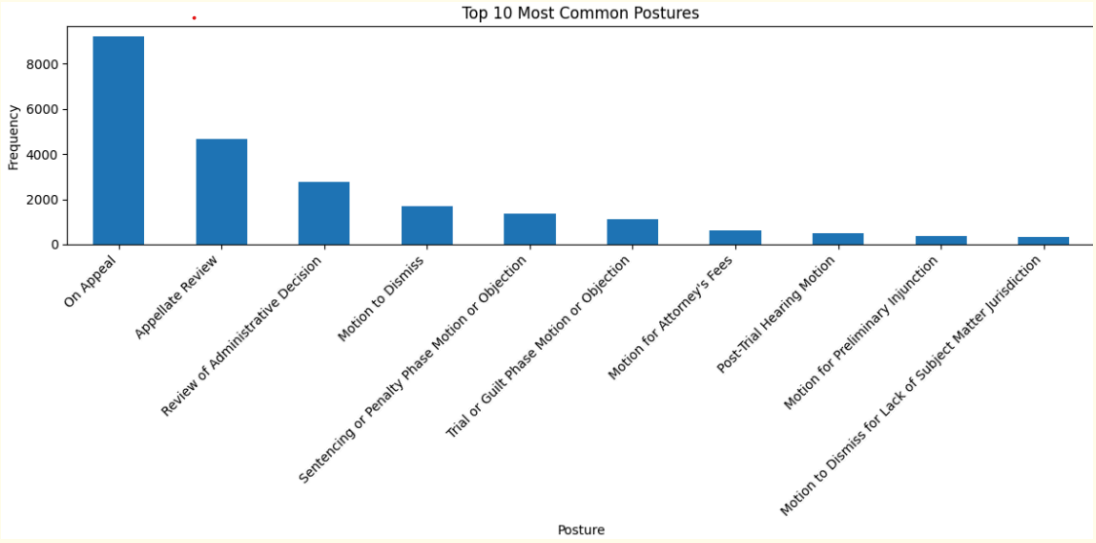
Label Distribution Characteristics:

- The majority of documents exhibit 1-2 posture labels, indicating a relatively sparse multi-label structure.
- Documents can contain up to 4+ concurrent posture labels, representing the complexity of multi-label density as models must predict all applicable postures for each document.
- Approximately 1,000 documents lack posture annotations (num_postures=0), requiring exclusion from the training dataset to prevent noise injection
- "On Appeal" and "Appellate Review" dominate the dataset as the most prevalent posture labels, while numerous categories exhibit extremely low frequency, creating a highly imbalanced multi-label distribution that poses significant challenges for model training.Without proper handling of this imbalance, the model may develop bias toward frequent categories while failing to adequately learn rare but potentially important legal

Data Quality Implications:

- The identified unlabeled instances necessitate data preprocessing to ensure training dataset integrity and model performance optimization.





Distribution of num_postures

num_postures	count	percentage
0	923	5.13%
1	8,118	45.10%
2	7,604	42.24%
3	1,129	6.27%
4	190	1.06%
5	32	0.18%
6	2	0.01%
7	2	0.01%

The dataset contains a large number of unique posture labels (e.g., 230), but only a subset of labels are common. Label frequency is highly imbalanced: a few postures (such as "On Appeal" and "Appellate

Review") appear in hundreds or thousands of documents, while many others are rare.

```
Total unique postures: 230

Most common postures:
On Appeal                                                          9197
Appellate Review                                                   4652
Review of Administrative Decision                                  2773
Motion to Dismiss                                                  1679
Sentencing or Penalty Phase Motion or Objection                    1342
Trial or Guilt Phase Motion or Objection                           1097
Motion for Attorney's Fees                                          612
Post-Trial Hearing Motion                                           512
Motion for Preliminary Injunction                                   364
Motion to Dismiss for Lack of Subject Matter Jurisdiction           343
Motion to Compel Arbitration                                        255
Motion for New Trial                                                226
Petition to Terminate Parental Rights                               219
Motion for Judgment as a Matter of Law (JMOL)/Directed Verdict      212
Motion for Reconsideration                                          206
```

The analysis typically filters to the most common postures (e.g., those appearing in at least 100 documents) to ensure sufficient data for model training and evaluation. The analysis provides a frequency table of the most common postures, such as:
"On Appeal"
"Appellate Review"
"Motion to Dismiss"
"Summary Judgment"
These common postures account for the majority of the dataset, while rare postures are often excluded from initial modeling.

Implications for Modeling:

- **Imbalanced label distribution** means that models may perform well on common postures but struggle with rare ones.
- **Filtering to common postures** helps ensure that the model has enough examples to learn from, but may limit the system's ability to recognize rare or specialized postures.
- **Multi-label evaluation metrics** (such as hamming loss and exact match accuracy) are necessary to properly assess model performance.

--	sample size	Total unique postures
raw_document	18,000	230
Removing Missing Postures	17,077	230
Filter to Common Postures	16,568	27

In summary,

- The initial dataset contains 18,000 samples and 230 unique posture labels.
- After removing samples with missing posture information, the sample size drops to 17,077, but the number of unique postures remains at 230.
- After filtering to only include common postures(those appearing in at least 100 documents), the sample size further reduces to 16,568, and the number of unique postures drops significantly to 27. This step focuses the dataset on the most frequent or relevant postures, likely for better model performance or analysis.
- The final 27 labels that the models will predict are as followings:

```
Labels: ['Appellate Review' 'Juvenile Delinquency Proceeding'
 "Motion for Attorney's Fees" 'Motion for Contempt' 'Motion for Costs'
 'Motion for Default Judgment/Order of Default'
 'Motion for Judgment as a Matter of Law (JMOL)/Directed Verdict'
 'Motion for New Trial' 'Motion for Permanent Injunction'
 'Motion for Preliminary Injunction' 'Motion for Protective Order'
 'Motion for Reconsideration' 'Motion to Compel Arbitration'
 'Motion to Dismiss' 'Motion to Dismiss for Lack of Jurisdiction'
 'Motion to Dismiss for Lack of Personal Jurisdiction'
 'Motion to Dismiss for Lack of Standing'
 'Motion to Dismiss for Lack of Subject Matter Jurisdiction'
 'Motion to Set Aside or Vacate' 'Motion to Transfer or Change Venue'
 'On Appeal' 'Petition for Divorce or Dissolution'
 'Petition to Terminate Parental Rights' 'Post-Trial Hearing Motion'
 'Review of Administrative Decision'
 'Sentencing or Penalty Phase Motion or Objection'
 'Trial or Guilt Phase Motion or Objection']
```

2.2. Text Length Distribution Analysis

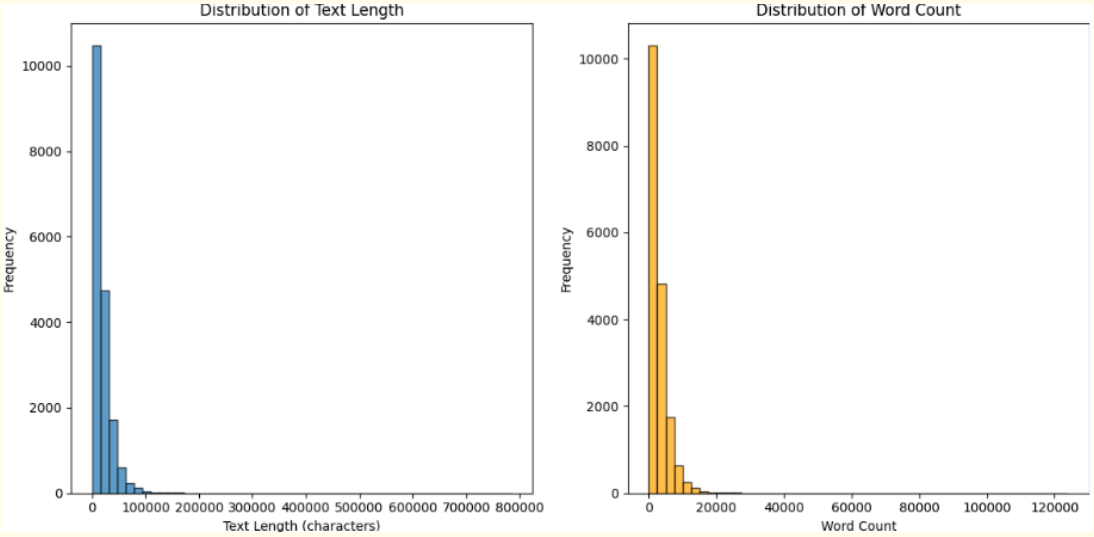
Analyzing text length distribution is a critical preprocessing step that directly influences model architecture selection and performance optimization. When deploying Large Language Models (LLMs) for this

multi-label classification task, context length becomes a pivotal consideration due to several key factors:

- **Context Length Constraints:** Each LLM architecture has a predefined maximum context length (e.g., 512 tokens for BERT, 4096 for Longformer, 8192 for MordernBERT, 10000+ for many decoder models(Llama, Mistral, Qwen family etc)). Understanding our data's text length distribution enables informed model selection that balances computational efficiency with information preservation.
- **Resource Optimization:** Oversized context windows lead to unnecessary computational overhead and increased memory consumption without performance gains. Conversely, insufficient context length results in critical information truncation, potentially degrading model accuracy.
- Strategic Implications: By examining the statistical distribution of text lengths across our dataset, we can:

    - Select appropriate model architectures with optimal context lengths
    - Implement efficient text preprocessing strategies (chunking, summarization, or truncation)
    - Estimate computational requirements and training costs
    - Ensure minimal information loss during model input preparation

This analysis forms the foundation for making data-driven decisions regarding model selection and hyperparameter configuration.



The above left plot shows the distribution of text length measured by the number of characters (the minimal unit of text). Whereas, the right plot shows the distribution of text length measured by word count (words separated by whitespace).

Both distributions are highly right-skewed, with most documents having relatively short lengths, but with a long tail extending far to the right. This indicates that while the majority of documents are short, there are a few rare documents that are extremely long, both in terms of character count and word count. These extreme outliers are visible as the long tails in both plots, highlighting the presence of unusually lengthy documents in the dataset.

Accurate Text Length Measurement Using Subword Tokenization:

The initial text length analysis employed a basic word-counting approach using whitespace as delimiters. However, this naive methodology fails to accurately represent how Large Language Models process and interpret textual input, leading to potential misestimation of context requirements.

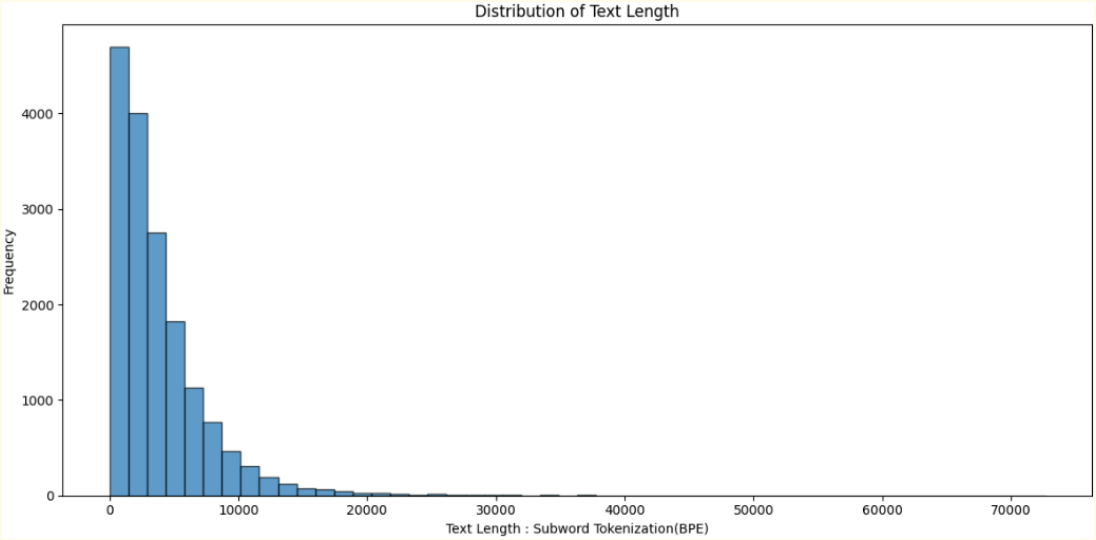
- **Tokenization Discrepancy:** Large Language Models(LLMs) do not process text at the word level as humans do. Instead, they utilize sophisticated subword tokenization algorithms including:
    - Byte Pair Encoding (BPE): Used by GPT models
    - WordPiece: Employed by BERT variants
    - SentencePiece: Utilized by T5 and many multilingual models
- **Technical Implementation:** To obtain precise context length measurements that align with actual LLM processing, we implement tokenization using ModernBERT, which employs BPE (Byte Pair Encoding) for subword segmentation. This approach provides several advantages:
    - Accurate Token Counting: Measures text length in the same units that the model processes
    - Context Window Optimization: Enables precise estimation of how much text fits within model constraints
    - Model-Agnostic Approach: The methodology can be adapted for different tokenizers (e.g., GPT-2/3/4 or many other LLMs' tokenizer) based on target model selection

We use the ModernBERT encoder model for subword tokenization. ModernBERT represents the latest advancement over the original (vanilla) BERT model, incorporating several key improvements. One of the main enhancements is the use of Byte-Pair Encoding (BPE) for tokenization, which allows the model to efficiently handle rare and out-of-vocabulary words by breaking them into subword units.

In addition to improved tokenization, ModernBERT introduces several innovative changes in its model architecture. These enhancements contribute to better overall performance, including increased accuracy and efficiency in various natural language processing tasks. Notably, ModernBERT significantly expands the maximum context length it can process—from the original BERT's limit of 512 tokens up to 8,192 tokens. This substantial increase enables the model to understand and generate much longer pieces of text, making it more suitable for tasks involving lengthy documents or conversations.

Distribution of text length(subword tokenization)

data_type	# of obs	Min	5%	10%	25%	Median	Average	75%	90%	95%	Max
full_text	16,568	2	562	739	1,311	2,725	3,826	5,065	8,143	10,646	72,647



The above two plots provide a detailed analysis of text length distribution after subword tokenization (using Byte-Pair Encoding, BPE).

The top table summarizes key statistics for the text length distribution (in subword tokens) for the dataset (16,568 documents). Most documents have a moderate length (median = 2,725 tokens), but the average is higher (3,826 tokens) due to the influence of a small

number of extremely long documents (as seen in the maximum value of 72,647 tokens). The 90th percentile is 8,143 tokens, showing that only 10% of documents exceed this length.

The bottom histogram displays the frequency distribution of text lengths (in subword tokens) across all documents.The distribution is highly right-skewed, with most documents having relatively short token lengths, but a long tail extending to the right. This indicates that while the majority of documents are concise, there are a few with extremely high token counts (outliers).

Both the histogram and the summary table highlight a right-skewed distribution of text lengths after subword tokenization.While the majority of documents are of moderate length, there are rare cases of extremely long documents, which significantly increase the average and maximum values. This insight is important for model design, especially when considering input length limitations for transformer-based models.

When considering ModernBERT as our target language model, which supports a maximum context length of 8,192 tokens, this configuration provides excellent coverage for our dataset. Approximately 90% of documents (up to 8,143 tokens) can be processed without truncation.

2.3. Data Split: Training, Validation and Test

The dataset consists of a total of 16,568 samples, which are divided into three subsets for model development and evaluation. The training set contains 11,597 samples (70% of the data) and is used to train the model. The validation set includes 2,485 samples (15%) and is used to tune model parameters and monitor performance during training. The test set also contains 2,486 samples (15%) and is reserved for evaluating the final model's performance on unseen data.

Total samples:	16,568	100%
Training set:	11,597	70%
Validation set:	2,485	15%
Test set:	2,486	15%

The screenshots below display the top 5 most frequent labels distribution among the training, validation and test sets, showing their counts and percentage distributions. The most common labels—such as "On Appeal," "Appellate Review," and "Review of Administrative Decision"—appear in the same order and with very similar proportions across all data sets. For example, "On Appeal" constitutes 55.2% of the training set , 55.6% of the validation set and 56.8% of the test set, while the other top labels also have nearly identical percentages.

This high level of consistency indicates that the label distribution is well-balanced among the training , validation and test sets, with no significant distribution shift. As a result, the patterns learned by the model during training are likely to generalize well to the validation and test sets, ensuring reliable model evaluation and reducing the risk of bias due to label imbalance.

```
Top 5 most frequent labels in training set:

On Appeal:                                      6,404     55.2%
Appellate Review:                               3,310     28.5%
Review of Administrative Decision:              1,940     16.7%
Motion to Dismiss:                              1,155     10.0%
Sentencing or Penalty Phase Motion or Objection:  953      8.2%
```

```
Top 5 most frequent labels in validation set:

On Appeal:                                      1,381     55.6%
Appellate Review:                               679       27.3%
Review of Administrative Decision:              406       16.3%
Motion to Dismiss:                              255       10.3%
Sentencing or Penalty Phase Motion or Objection:  204      8.2%
```

```
Top 5 most frequent labels in test set:

On Appeal:                                      1,412     56.8%
Appellate Review:                               663       26.7%
Review of Administrative Decision:              427       17.2%
Motion to Dismiss:                              269       10.8%
Sentencing or Penalty Phase Motion or Objection:  185      7.4%
```

3. Model Development

*(All of the following analysis can be found in the notebook: **model_development_q2.ipynb**)*

Multi-Label Posture Classification: Model Development Strategy

Two complementary modeling approaches are proposed to address the multi-label posture prediction task, each offering distinct advantages for legal document classification.

- The baseline model utilizes **Bag-of-Words** representations, specifically **TF-IDF** or **BM25**. This approach is well-suited for benchmarking due to several key advantages:

    1. **Computational Efficiency:** Bag-of-Words models are lightweight and fast, allowing for rapid prototyping and baseline performance assessment without the need for GPU resources.
    2. **Statistical Robustness:** By leveraging word-frequency features, these models offer interpretable and domain-agnostic representations, making them particularly suitable for analyzing specialized legal terminology.

    Using Bag-of-Words as a benchmark provides clear and interpretable performance metrics, which are essential for objectively evaluating the benefits of more complex, resource-intensive architectures.

- Advanced Approach: **Transformer-Based Models** (ModernBERT). The primary model leverages ModernBERT encoder architecture, a state-of-the-art transformer-based model specifically enhanced for multi-label classification tasks. This approach is chosen to overcome the inherent limitations of traditional bag-of-words and earlier BERT models.

    1. **Extended Context Coverage:** ModernBERT features an expanded context window of up to 8,192 tokens, which allows it to process and understand much longer

documents. In our dataset, this means that 90% of legal texts can be analyzed in their entirety without truncation, ensuring that important contextual information—often spread across entire documents—is preserved.

2. **Contextual and Semantic Understanding:**
   Unlike bag-of-words models, which rely solely on word frequency and ignore word order and context, transformer architectures like ModernBERT excel at capturing:
   - Long-range dependencies: They can model relationships between legal arguments and references that may be separated by hundreds or thousands of words.
   - Semantic nuances: Transformers distinguish between subtle differences in legal postures and categories, which is critical for accurate multi-label classification.
   - Pre-trained language knowledge: ModernBERT is pre-trained on large corpora, enabling it to better understand complex legal terminology, syntax, and document structure.

While bag-of-words models are efficient and interpretable, they lack the ability to capture context, sequence, and meaning beyond individual word counts. In contrast, transformer-based models provide a much richer and more nuanced understanding of legal documents, leading to improved classification performance, especially in complex, multi-label scenarios.

By leveraging ModernBERT, we ensure that our model can fully utilize the depth and complexity of legal texts, providing a significant advantage over traditional baseline methods.

Approaches to Multi-Label Classification

Before delving into specific model architectures, it is crucial to first consider the overall strategy for addressing multi-label classification problems, as well as the appropriate metrics for evaluating model performance in this context.

Several modeling strategies exist for multi-label classification, each with its own strengths and limitations:

- **Binary Relevance:**
  This approach decomposes the multi-label classification problem into k independent binary classification tasks, where k is the number of possible labels. Each classifier is trained to predict the presence or absence of a single label, treating all labels independently.

  Advantages: Simplicity, scalability, and compatibility with any binary classification algorithm.
  Limitations: Ignores potential correlations and dependencies between labels, which may be important in some domains.

- **Classifier Chains:**
  In this method, classifiers are linked in a chain, where the prediction of each label is conditioned not only on the input features but also on the predictions of previous labels in the chain.

  Advantages: Captures label dependencies and can improve performance when label correlations are significant.
  Limitations: Increased model complexity and sensitivity to the order of labels in the chain.

- **Label Powerset:**
  This strategy treats each unique combination of labels as a single class, effectively transforming the multi-label problem into a multi-class classification task.

  Advantages: Considers label correlations explicitly.
  Limitations: Can lead to a very large number of classes, especially in datasets with many labels, resulting in data sparsity and scalability issues.

For this study, the **binary relevance** approach is selected as the modeling strategy. The primary reasons for this choice are its conceptual simplicity, ease of implementation, and flexibility to work with any underlying binary classification algorithm. While it does not model label correlations, binary relevance provides a strong and interpretable baseline, especially when the primary goal is to establish

robust performance benchmarks or when label dependencies are weak or unknown.

Evaluation Metrics for Multi-Label Classification

Once the modeling strategy is established, it is important to select evaluation metrics that accurately reflect model performance in the multi-label setting. Commonly used metrics include:

- **Hamming Loss:**
  Measures the fraction of labels that are incorrectly predicted (either missed or wrongly assigned) across all samples. It is particularly suitable for problems with large label spaces, as it provides partial credit for partially correct predictions.

- **Jaccard Index (Intersection over Union, IoU):**
  Calculates the size of the intersection divided by the size of the union of the predicted and true label sets for each sample, then averages across all samples. This metric penalizes both false positives and false negatives, providing a balanced view of prediction quality.

- **Micro-Averaged Precision, Recall, and F1 Score:**
  Aggregates all true positives, false positives, and false negatives across all labels before computing the metrics. This approach is misleading when label frequencies are Imbalanced.

- **Macro-Averaged Precision, Recall, and F1 Score:**
  Computes the metrics independently for each label and then averages them, treating each label equally regardless of its frequency. This is especially important in scenarios with imbalanced label distributions, as it ensures that minority labels are given equal consideration.

- **ROC-AUC and Precision-Recall AUC:**
  The metrics discussed above—such as F1 score, Hamming Loss, and Jaccard Score—are all dependent on the choice of threshold used to convert predicted probabilities into binary (hard) predictions. The selection of this threshold can significantly influence the resulting metric values; for example, a

higher or lower threshold may lead to different balances between precision and recall, and thus different F1 or Jaccard scores. This sensitivity can make it challenging to fairly compare models or assess their true performance, especially when the optimal threshold is not obvious or varies across datasets.

In contrast, ROC-AUC (Receiver Operating Characteristic Area Under the Curve) and Precision-Recall AUC are threshold-independent metrics. They evaluate model performance across all possible threshold values, providing a more comprehensive and robust assessment. This property makes them particularly valuable in scenarios with imbalanced data, where the choice of threshold can have an even greater impact on traditional metrics. By summarizing performance over the full range of thresholds, ROC-AUC and PR-AUC offer a more reliable measure of a model's ability to distinguish between classes, regardless of class distribution or threshold selection.

In summary, the binary relevance approach is chosen for its simplicity and broad applicability, providing a solid foundation for multi-label classification. Careful selection of evaluation metrics—such as Hamming Loss, Jaccard Index, and both macro- and weighted-averaged scores in F1 and AUC—ensures a comprehensive assessment of model performance, accounting for both overall accuracy and the treatment of individual labels.

3.1 Bag-of-word (TFIDF): Benchmark

TFIDF feature extraction transforms the raw text into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features, which reflect the importance of words and word pairs in the documents.

```
# Create TF-IDF vectorizer
tfidf = TfidfVectorizer(
    max_features=10000,  # Limit features for computational
efficiency
    stop_words='english',
    ngram_range=(1, 2),  # Include unigrams and bigrams
    min_df=5,            # Ignore terms that appear in fewer than
5 documents
```

```
    max_df=0.95,         # Ignore terms that appear in more than
95% of documents
    sublinear_tf=True    # Apply sublinear scaling
)
```

Feature Limit: The number of features is capped at 10,000 (max_features=10000) to ensure computational efficiency.
Stop Words Removal: Common English stop words are removed (stop_words='english'), reducing noise in the data.
N-gram Range: Both unigrams (single words) and bigrams (pairs of consecutive words) are included (ngram_range=(1, 2)), capturing more context than single words alone.
Minimum Document Frequency: Terms appearing in fewer than 5 documents are ignored (min_df=5), filtering out rare, potentially uninformative terms.
Maximum Document Frequency: Terms appearing in more than 95% of documents are also ignored (max_df=0.95), removing very common terms that are unlikely to be discriminative.
Sublinear TF Scaling: Sublinear scaling is applied to term frequencies (sublinear_tf=True), which helps reduce the impact of very frequent terms.

As a result, this TF-IDF feature extraction process produces a 10,000-dimensional feature space, providing a compact and informative representation of the text data for subsequent model development.

After extracting TF-IDF features from the text data, we utilized four different machine learning models for multi-label classification: Logistic Regression, Random Forest, XGBoost, and LightGBM. These models were trained on the training set, and their performance was first evaluated on a validation set to monitor generalization and prevent overfitting. Subsequently, the trained models were assessed on a separate, synthetic test set to estimate their performance on truly unseen data.

The attached image below presents two comprehensive tables summarizing the evaluation metrics for each model on both the validation and test sets. The metrics reported include Accuracy, Hamming Loss, F1 scores (macro, and weighted), and ROC / Precision-Recall AUC (macro and weighted). These metrics provide a

holistic view of each model's strengths and weaknesses across different aspects of multi-label classification.

Model Evaluation in Validation Set

Model	Val Acc	Val Ham	val_f1_macro	val_f1_weighted	val_roc_auc_macro	val_roc_auc_weighted	val_pr_auc_macro	val_pr_auc_weighted
LOGISTIC	0.4978	0.0269	0.6277	0.8241	0.9800	0.9765	0.6757	0.8560
RANDOMFOREST	0.5127	0.0245	0.3862	0.7666	0.9655	0.9676	0.6136	0.8480
LIGHTGBM	0.6149	0.0181	0.5954	0.8183	0.9522	0.9753	0.6355	0.8891
XGBOOST	0.6205	0.0176	0.5861	0.8162	0.9778	0.9788	0.6918	0.9184

Model Evaluation in Test Set

Model	test Acc	test Ham	test_f1_macro	test_f1_weighted	test_roc_auc_macro	test_roc_auc_weighted	test_pr_auc_macro	test_pr_auc_weighted
LOGISTIC	0.4702	0.0281	0.6184	0.8188	0.9772	0.9746	0.6688	0.8556
RANDOMFOREST	0.5270	0.0242	0.3861	0.7671	0.9641	0.9649	0.6091	0.8438
LIGHTGBM	0.6070	0.0190	0.5729	0.8096	0.9621	0.9755	0.6197	0.8836
XGBOOST	0.6219	0.0179	0.5815	0.8133	0.9797	0.9780	0.6835	0.9148

The first table displays the performance of each model on the validation set. XGBoost achieves the highest accuracy (0.6205) and the lowest Hamming Loss (0.0176), indicating strong predictive performance and fewer label-wise errors. LightGBM also performs well, closely following XGBoost in most metrics. Logistic Regression and Random Forest show comparatively lower performance, especially in macro F1 and Jaccard scores, which suggests they may struggle more with less frequent labels.

The second table shows the results on the test set, which simulates unseen data. The performance trends observed in the validation set are largely mirrored here, demonstrating that the models generalize well and there is no significant distribution shift between the validation and test sets. XGBoost again leads in accuracy (0.6219) and Hamming Loss (0.0179), with LightGBM as the next best performer. The consistency across both sets reinforces the reliability of the evaluation process.

The similarity in performance metrics between the validation and test sets indicates that the models are not overfitting and are likely to perform similarly on new, real-world data. XGBoost consistently outperforms the other models across most metrics, followed by LightGBM. Logistic Regression and Random Forest lag behind, particularly in metrics that emphasize performance on minority labels.

It is important to note that these results were obtained without extensive hyperparameter tuning. With further optimization, it is likely

that all models, especially the tree-based ones, could achieve even better performance.

The tables provide a clear and detailed comparison of model performance, highlighting the strengths of advanced gradient boosting methods like XGBoost and LightGBM for multi-label text classification using TF-IDF features. The consistent results across validation and test sets underscore the robustness of the modeling pipeline.

3.2 Transformers Encoder Model (MordenBERT)

ModernBERT is an encoder-only, transformer-based model designed for efficient and effective text representation. As an encoder model, ModernBERT transforms input documents into dense numerical vectors that capture the essential semantic information of the text. These vectors serve as compressed representations, which is why encoder-only architectures are often referred to as representational models.

One of the key innovations in ModernBERT is the use of rotary positional embeddings (RoPE) instead of traditional positional encoding. RoPE enhances the model's ability to understand the relative positions of words within a sequence, enabling it to better capture word order and relationships. This improvement also allows ModernBERT to handle much longer input sequences, making it suitable for tasks that require processing extended contexts.

Another significant feature of ModernBERT is its Alternating Attention mechanism. Unlike standard transformer models that apply global attention at every layer, ModernBERT alternates between global and local attention. Specifically, global attention—where each token can attend to all others—is applied every third layer. In the remaining layers, a sliding window approach is used, restricting each token's attention to its 128 nearest neighbors. This design reduces computational complexity while still capturing both local and global dependencies in the text.

In our experiments, we evaluate ModernBERT (both based and large) with varying context lengths, ranging from 512 to 8,192 tokens. This

allows us to assess whether increasing the context window and reducing truncation leads to improved model performance. The results are then compared to those obtained from our benchmark models, providing a comprehensive evaluation of ModernBERT's capabilities in handling longer and more context-rich documents.

model_name	maximal_context_length	# of parameters	num_hidden_layer	embedding_size
ModernBERT-base	8,192	149,014,272	22	768
ModernBERT-large	8,192	394,781,696	28	1,024

The above attached table summarizes the key architectural details of the two ModernBERT models used in the multi-label classification experiments: ModernBERT-base and ModernBERT-large. Both models support a maximal context length of 8,192 tokens, allowing them to process long documents without truncation.

- **ModernBERT-base** has approximately 149 million parameters, 22 hidden layers, and an embedding size of 768.
- **ModernBERT-large** is a larger variant with about 395 million parameters, 28 hidden layers, and an embedding size of 1,024.

These two configurations enable a comparison of model capacity and performance in handling complex multi-label classification tasks.

Training Configuration and Strategy:

The training process leverages the HuggingFace Trainer API with a highly customized TrainingArguments configuration, tailored for multi-label classification using ModernBERT. Key aspects of the setup include:

- **Maximal Context Length:**
  For demonstration purposes, In the Jupyter notebook we set max_context_length=512 to accelerate model training. Experiments with longer context lengths were conducted in the background using L4 GPUs on the Runpod GPU Cloud platform: Runpod GPU Cloud. Please refer to train.py for other context length experiment

- **Learning Rate and Scheduling:**
  A learning rate of 2e-5 is used with linear decay and a 10% warmup phase, which helps stabilize early training and gradually reduces the learning rate for better convergence.

- **Batching and Gradient Accumulation:**

The batch size per device is set via args.train_batch and args.eval_batch, with gradient accumulation steps (args.gradient_accumulation_step) to achieve a larger effective batch size without exceeding GPU memory limits.

- **Evaluation and Checkpointing:**
  Evaluation is performed every 100 steps (eval_steps=100), and model checkpoints are saved at the same interval. Only the three best checkpoints are retained to save disk space. The best model is selected based on the lowest validation Hamming Loss, which is a suitable metric for multi-label problems.

- **Early Stopping and Callbacks:**
  Early stopping is enabled with a patience of 3 evaluations, halting training if no improvement is observed, thus preventing overfitting. Additional callbacks clear CUDA cache and log training progress.

- **Mixed Precision and Reproducibility:**
  Mixed precision (fp16=True) is enabled for faster training on compatible GPUs. Seeds are set for reproducibility.

- **Custom Logging:**
  Built-in logging is disabled in favor of custom logging and reporting, allowing for more flexible and detailed tracking of metrics.

[1201/1210 30:26 < 00:13, 0.66 it/s, Epoch 4.96/5]

Step	Training Loss	Validation Loss	Accuracy	Hamming Loss	F1 Macro	F1 Weighted	Roc Auc Macro	Roc Auc Weighted	Pr Auc Macro	Pr Auc Weighted	Jaccard Macro	Jaccard Weighted
100	No log	0.108869	0.324346	0.036277	0.098176	0.544194	0.757261	0.889413	0.177906	0.664161	0.077208	0.453510
200	No log	0.075353	0.500201	0.024547	0.191859	0.679791	0.874292	0.951123	0.344305	0.774936	0.150330	0.598609
300	No log	0.068367	0.543260	0.022938	0.347228	0.738232	0.925669	0.962463	0.446651	0.801689	0.268993	0.651860
400	No log	0.061599	0.554930	0.021149	0.355828	0.737189	0.940895	0.966702	0.521841	0.819050	0.280129	0.660274
500	No log	0.059241	0.574648	0.020762	0.451176	0.761073	0.946890	0.968905	0.581514	0.837406	0.352403	0.675321
600	No log	0.056793	0.591147	0.019674	0.480668	0.784267	0.951841	0.970376	0.598930	0.840041	0.381261	0.701362
700	No log	0.055119	0.592354	0.019510	0.527601	0.786582	0.958111	0.971808	0.616069	0.848114	0.406824	0.699286
800	No log	0.054728	0.610865	0.018735	0.536085	0.799105	0.956394	0.970696	0.629109	0.848413	0.420717	0.710891
900	No log	0.054200	0.615694	0.018451	0.541879	0.806194	0.954730	0.970917	0.642234	0.852992	0.424259	0.720777
1000	No log	0.053624	0.621328	0.018213	0.587482	0.818676	0.956112	0.971402	0.637450	0.852427	0.461486	0.730777
1100	No log	0.054034	0.615292	0.018213	0.576042	0.812174	0.955955	0.970973	0.640449	0.852279	0.453131	0.725022
1200	No log	0.053865	0.617706	0.018213	0.577485	0.814890	0.954598	0.970856	0.639752	0.852321	0.452617	0.727075

Training Progress and Validation Metrics:
The attached image above displays the validation metrics at each evaluation step (every 100 steps) throughout the training process.

- Validation loss decreases and accuracy increases throughout training, reflecting better generalization to unseen data.
- All F1 , Jaccard and AUC scores (across macro, and weighted variants) improve, demonstrating that the model is learning to predict multiple labels more accurately and with better overlap.
- The use of early stopping and checkpointing ensures that the best-performing model (based on Hamming Loss) is retained, reducing the risk of overfitting.

The training process is robust and well-optimized for multi-label classification. The model demonstrates strong and consistent improvements across all validation metrics, indicating effective learning and generalization. The careful choice of evaluation metrics, frequent validation, and early stopping contribute to a reliable and reproducible training pipeline, suitable for technical reporting and further experimentation.

The following table summarizes the performance of ModernBERT models (Base and Large) across different context lengths on the test set. The metrics reported include Accuracy, Hamming Loss, F1 macro, F1 micro, Jaccard macro, and Jaccard weighted.

Model Performance Across Different Model Size and Context Length
Test Set

ModerBERT	Context-Length	Accuracy	Hamming_Loss	F1_macro	ROC_AUC_macro	PR_AUC_macro	Jaccard_macro	Jaccard_weighted
Base	512	61.42%	0.0187	0.5769	0.9552	0.6367	0.4520	0.7273
Base	1,024	63.52%	0.0177	0.6054	0.9634	0.6400	0.4806	0.7419
Base	2,048	64.88%	0.0171	0.6154	0.9700	0.6596	0.4936	0.7499
Base	4,096	65.69%	0.0165	0.6354	0.9739	0.6806	0.5113	0.7572
Base	8,192	65.77%	0.0163	0.6475	0.9744	0.6914	0.5231	0.7607
Large	512	65.93%	0.0166	0.6360	0.9680	0.6933	0.5078	0.7565
Large	1,024	67.98%	0.0154	0.6716	0.9706	0.7327	0.5509	0.7698
Large	2,048	68.02%	0.0151	0.6675	0.9777	0.7413	0.5447	0.7767
Large	4,096	68.50%	0.0151	0.7031	0.9769	0.7476	0.5788	0.7851
Large	8,192	68.58%	0.0149	0.7086	0.9798	0.7526	0.5836	0.7822

Key Observations:

- **Context Length:**
  For both the Base and Large models, increasing the context length from 512 to 8,192 tokens consistently improves performance across all metrics. Accuracy, F1 scores, and Jaccard indices all increase, while Hamming Loss decreases. This indicates that providing the model with more context (i.e., longer input sequences) allows it to make more accurate and comprehensive predictions, likely because it can utilize more relevant information from the document without truncation.

- **Model Size:**
  The Large model outperforms the Base model at every context length. For example, at the maximum context length of 8,192 tokens, the Large model achieves the highest accuracy (68.58%), lowest Hamming Loss (0.0149), and the best F1 and Jaccard scores. This demonstrates that increasing model capacity (more parameters, larger embedding size, and more layers) leads to better representation learning and improved classification performance.

It turns out that both increasing the context length and using a larger model size contribute to better multi-label classification performance. The best results are achieved with the ModernBERT-Large model at the maximum context length, suggesting that for tasks involving long documents and complex label structures, investing in models that can process more context and have greater capacity is highly beneficial. This table clearly demonstrates the value of scaling both input length and model size for advanced NLP tasks.

3.3 Performance Comparison (TFIDF vs MordenBERT)

The following table presents a holistic performance comparison between traditional decision tree-based models using TF-IDF features (Logistic Regression, Random Forest, XGBoost, LightGBM) and transformer-based encoder models (ModernBERT-Base and ModernBERT-Large) for multi-label classification.

Performance Comparison: TFIDF vs MordenBERT

ModerBERT	Accuracy	Hamming_Loss	F1_macro	ROC_AUC_macro	PR_AUC_macro	Jaccard_macro	Jaccard_weight
Logistic-Reg	47.02%	0.0281	0.6184	0.6688	0.9772	0.4815	0.722743
Random-Forest	52.70%	0.0242	0.3861	0.6091	0.9641	0.2972	0.672294
XGBoost	60.70%	0.0190	0.5729	0.6197	0.9621	0.4471	0.721178
LightGBM	62.19%	0.0179	0.5815	0.6835	0.9797	0.4584	0.725522
MordenBERT-Base	65.77%	0.0163	0.6475	0.6914	0.9744	0.5231	0.760700
MordenBERT-Large	68.58%	0.0149	0.7086	0.7526	0.9798	0.5836	0.782200

The table reports several key evaluation metrics, including Accuracy, Hamming Loss, F1 macro, ROC-AUC macro, PR-AUC macro, Jaccard macro, and Jaccard weighted. Among all models, ModernBERT-Large achieves the best performance across nearly all metrics, with the highest accuracy (68.58%), lowest Hamming Loss (0.0149), and the

strongest F1 macro, ROC-AUC, PR-AUC, and Jaccard scores. ModernBERT-Base also outperforms all traditional models, ranking second in every metric.

This comparison highlights the superior capability of transformer-based encoder models, particularly ModernBERT, in handling complex multi-label classification tasks. Their ability to capture long-range contextual information in legal documents enables them to predict procedural posture more accurately than decision tree-based models, which are limited by their reliance on bag-of-words features and lack of contextual understanding.

4. Additional Model Experiments

The model experiments in section 3 indicates that context length is essential for the model performance since longer context length means more relevant information exposed to the model from the document without truncation.

However, the text length in our corpus is quite long, even though the language model can handle a maximum input of 8,192 tokens or even longer context, such lengthy contexts can dilute the essential information that the LLM is able to process effectively. When the input text is too long, the model may struggle to focus on the most relevant

details, which can negatively impact its performance and the quality of its predictions.

In order to address this challenge, it is beneficial to provide the language model with only the most relevant information from the large corpus, rather than overwhelming it with the entire text. By filtering and condensing the input, we can help the LLM focus on the critical content, thereby enhancing its ability to extract essential information and make more accurate predictions.

There are two promising approaches that are worthy of further exploration:

4.1 Summarization:

Rather than feeding the entire corpus into the LLM, first summarize the information relevant to the procedural posture within the legal document, and then prompt the LLM to predict the posture based on that focused summary. Summarization techniques can be used as a form of feature extraction. By generating concise summaries of the original text, we can distill the most important points and reduce the input length, making it easier for the LLM to process and understand the core information.

The following prompt template is used to instruct either OpenAI's GPT-4o or open-source language models such as Qwen 3 to summarize the procedural posture of legal documents, including judicial opinions and legal briefs.

```
# Instruction

## Context
- **Goal**: You are tasked with summarizing the procedural posture for the legal document in judicial opinion or briefs.
The procedural posture of a case is a summary of how the case arrives before the court.
- **Data**: Your input data is a legal document which describes the procedural history including any prior decisions under appeal.

# Data
<data>
{content}
```

```
</data>

# Questions
## Q1. Summarize the input text in {summary_length} words or less.
Write the summary between <summary> </summary> tags.

Tips:
- The summary should contain the relevant information for the procedural posture in as much detail as possible.
- Be concise and clear. Do not add phrases like "This is the summary of the data ..." or "Summarized text: ...".
- Similarly, do not reference the user ('the user asked XYZ') unless it's absolutely relevant.
- Within {summary_length} words, include as much relevant information as possible.
- Do not include any line breaks in the summary.
- Provide your answer in **English** only.

## Q2. Explain how you wrote the summary in {explanation_length} words or less.

## Provide your answers between the tags <summary>your answer to Q1</summary>, <explanation>your answer to Q2</explanation>

# Output
```

To enhance the quality of summarization, we adopt a chain-of-thought prompting strategy. Rather than requesting a summary directly, we first ask the model to generate an explanation or rationale. This intermediate reasoning step allows the model more time to ”think” through the relevant aspects before producing the final summarization, leading to more coherent and contextually accurate results.

Furthermore, both the summary and explanation lengths are dynamically adjusted based on the length of the input text. Longer documents are paired with proportionally longer explanations and summaries to ensure adequate coverage of key procedural information.

In the jupyter notebook, I randomly sampled 10 legal documents to demonstrate how to build the entire summarization workflow using **LangGraph** platform.

The final output follows a structured format as shown in the table below. Given an input text field named **"content"**, we prompt an LLM to generate both a summary focused on the procedural posture and its corresponding rationale. This summary serves as a distilled representation of the key procedural information, enabling us to use it in place of the original raw content—which is often lengthy and noisy—for more efficient and accurate procedural posture prediction.

id	content	posture	summary	explanation
0	In a proceeding pursuant to Mental Hygiene Law...	On Appeal	Roderick L. appeals from a Supreme Court, Dutc...	To write the summary, I focused on capturing t...
1	Defendant-appellant Ronald Epps appeals from a...	Appellate Review, Trial or Guilt Phase Motion ...	Defendant-appellant Ronald Epps appeals from a...	In writing the summary, I focused on capturing...
2	Proceeding pursuant to CPLR article 78, in eff...	Review of Administrative Decision	The case involves a CPLR article 78 proceeding...	I wrote the summary by focusing on the procedu...
3	In an action, inter alia, to recover damages f...	Motion to Dismiss, On Appeal	In a legal action involving claims of conversi...	I wrote the summary by focusing on the procedu...
4	Appeal from an order of the Family Court of Ch...	On Appeal	This case is an appeal from a Family Court ord...	I focused on the procedural history, detailing...
5	¶ 1 Travis Padgett appeals trial court orders ...	Appellate Review	Travis Padgett appealed trial court orders den...	To write the summary, I focused on capturing t...
6	Desima James appeals his 63-month sentence aft...	Appellate Review, Sentencing or Penalty Phase ...	Desima James appealed his 63-month sentence af...	To write the summary, I focused on capturing t...
7	It is hereby ORDERED that the judgment so appe...	Appellate Review	The procedural posture of the case involves th...	In writing the summary, I focused on capturing...
8	Appeal from a judgment of the Supreme Court (N...	On Appeal, Review of Administrative Decision	Petitioner, a Connecticut resident, had his Ne...	I focused on the key procedural events: the re...
9	In an action to recover damages for personal i...	On Appeal	In a personal injury case, the plaintiff appea...	I focused on summarizing the procedural histor...

Below is a screenshot illustrating an example of the summary and rationale generated by the LLM. It demonstrates what procedural information was extracted from the original content and how the model reasoned through the text to produce the final summarization.

```
========================================
Original Content:

In a proceeding pursuant to Mental Hygiene Law article 10, Roderick L.
appeals from an order of the
Supreme Court, Dutchess County (Peter M. Forman, J.), dated January 13,
2017.  The order, upon a
finding, made after a jury trial, that Roderick L. suffers from a mental
abnormality as defined in
Mental Hygiene Law § 10.03(i) and a determination, made after a
dispositional hearing, that he is a
dangerous sex offender requiring civil confinement, in effect, granted
the petition and directed
that he be committed to a secure treatment facility for care and
treatment. ORDERED that the order
is affirmed, without costs or disbursements. In June 2000, the appellant
was convicted, upon his
plea of guilty, of attempted rape in the first degree for a sexual
assault he committed on December
23, 1999, against the eight-year-old daughter of his then-girlfriend.
 Prior to the release of the
appellant from prison, the State of New York commenced this proceeding
pursuant to Mental Hygiene
Law article 10, alleging that the appellant is a sex offender requiring
civil management (see Mental
Hygiene Law § 10.06[a]).  A jury trial was held, after which the jury
found that the appellant
```

suffers from a "mental abnormality," as defined in Mental Hygiene Law § 10.03(i) (see Mental Hygiene Law § 10.07). Following a dispositional hearing, the Supreme Court determined that the appellant is a dangerous sex offender requiring civil confinement (see Mental Hygiene Law § 10.07[f]; see also Mental Hygiene Law § 10.03[e]), in effect, granted the petition, and directed that the appellant be committed to a secure treatment facility and receive further treatment. Contrary to the appellant's contention, legally sufficient evidence supported the jury verdict since there was a valid line of reasoning by which the jury could conclude that the appellant suffered from a mental abnormality as defined in Mental Hygiene Law § 10.03(i) (see Matter of State of New York v. David B., 156 A.D.3d 793, 793, 67 N.Y.S.3d 650; Matter of State of New York v. Ian I., 127 A.D.3d 766, 767–768, 7 N.Y.S.3d 199; Matter of State of New York v. Raul L., 120 A.D.3d 52, 59, 988 N.Y.S.2d 190).

Additionally, the verdict was not contrary to the weight of the evidence, as it was supported by a fair interpretation of the evidence (see Matter of State of New York v. David B., 156 A.D.3d at 793, 67 N.Y.S.3d 650; Matter of State of New York v. Ian I., 127 A.D.3d at 768, 7 N.Y.S.3d 199). We agree with the Supreme Court's determination to admit certain hearsay evidence regarding allegations in the indictment of sexual assaults committed by the appellant against the eight-year-old child other than the count to which he was allowed to plead guilty. That evidence was sufficiently reliable and the probative value of the evidence in assisting the jury to evaluate the experts' opinions substantially outweighed its prejudicial effect (see Matter of State of New York v. John S., 23 N.Y.3d 326, 345, 991 N.Y.S.2d 532, 15 N.E.3d 287; Matter of State of New York v. Kerry K., 157 A.D.3d 172, 187, 67 N.Y.S.3d 227). Although the court should not have admitted hearsay evidence regarding uncharged allegations of sexual assaults against the eight-year-old child's six-year-old sister, the error was harmless (see Matter of State of New York v. John S., 23 N.Y.3d at 347–348, 991 N.Y.S.2d 532, 15 N.E.3d 287; Matter of State of New York v. James M., 140 A.D.3d 1178, 1179, 35 N.Y.S.3d 374). We agree with the Supreme Court's determination to permit the grand jury minutes to be released pursuant to Mental Hygiene Law § 10.08(c) and relied upon by the experts (see Matter of State of New York v. John S., 23 N.Y.3d at 341–342, 991 N.Y.S.2d 532, 15 N.E.3d 287; Matter of State of New York v. Kerry K., 157 A.D.3d at 184, 67 N.Y.S.3d 227). We also agree with the Supreme Court's determination to grant the State's application to admit into evidence the written report of its unavailable expert (see Matter of State of New York v. Leroy P., 122 A.D.3d 638, 640, 995

N.Y.S.2d 596). Contrary to the appellant's contention, the evidence was not cumulative to the testimony of the State's testifying expert (see Matter of State of New York v. Justin D., 145 A.D.3d 735, 736, 43 N.Y.S.3d 399; Matter of State of New York v. Leroy P., 122 A.D.3d at 640, 995 N.Y.S.2d 596). Furthermore, we agree with the Supreme Court's finding, after the dispositional hearing, by clear and convincing evidence, that the appellant's level of dangerousness requires confinement (see Mental Hygiene Law § 10.07[f]; Matter of State of New York v. Larry B., 113 A.D.3d 865, 867, 979 N.Y.S.2d 397).

==========
Summary:

Roderick L. appeals from a Supreme Court, Dutchess County order dated January 13, 2017, which affirmed his civil confinement under Mental Hygiene Law article 10. Initially, Roderick L. was convicted in June 2000 for attempted rape of an eight-year-old. Before his release, New York State initiated proceedings under Mental Hygiene Law article 10, claiming he required civil management. A jury found Roderick L. had a "mental abnormality" as per Mental Hygiene Law § 10.03(i). Following a dispositional hearing, the court deemed him a dangerous sex offender needing civil confinement. The court's decision was supported by sufficient evidence and was not against the weight of the evidence. The court admitted certain hearsay evidence and grand jury minutes, which were deemed reliable and probative. The court also allowed the State's expert report despite the expert's unavailability. The court's findings were supported by clear and convincing evidence, affirming Roderick L.'s confinement necessity.

==========
Explanation:

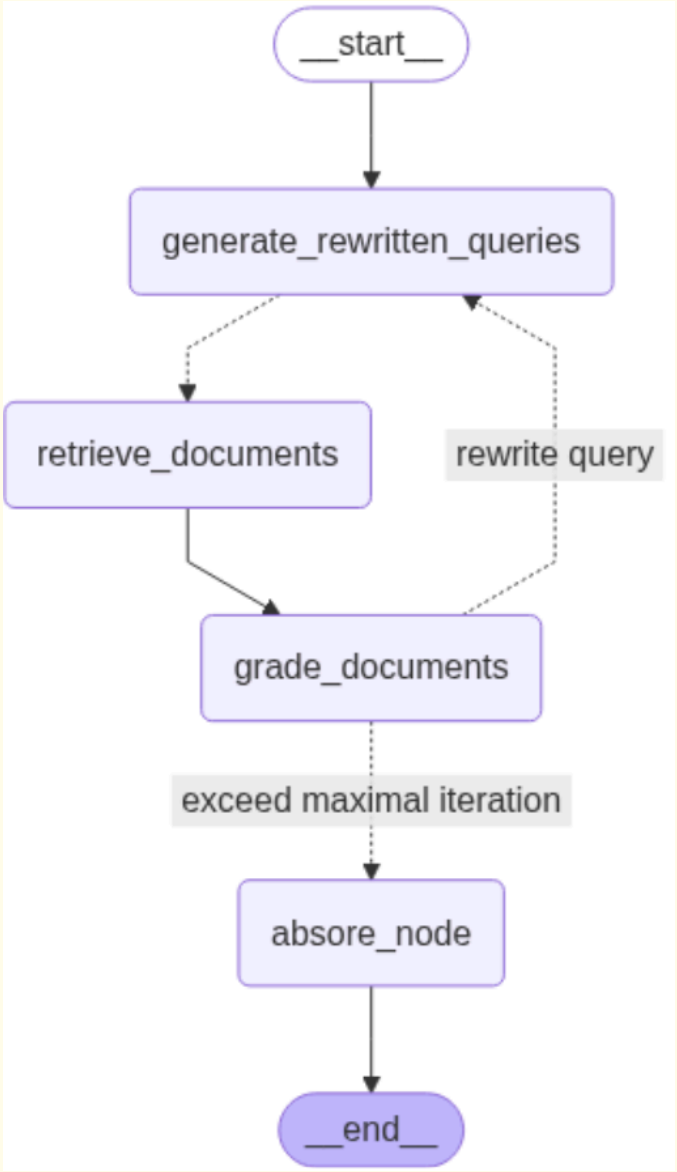
To write the summary, I focused on capturing the key procedural events and decisions that led to the appeal. I included the initial conviction, the State's initiation of proceedings under Mental Hygiene Law, the jury's finding of a mental abnormality, and the court's determination of Roderick L. as a dangerous sex offender requiring confinement. I also noted the court's evidentiary rulings and the affirmation of the order based on sufficient evidence. The summary was crafted to be concise yet comprehensive, ensuring all critical procedural details were included within the 260-word limit.
==========

4.2 Retrieval Augmentation Generation(RAG)

Alternatively, we can adopt a Retrieval-Augmented Generation (**RAG**) approach to extract relevant information prior to prediction. This method involves the **retrieval component** of the RAG system, retrieving the most pertinent sections from the corpus before passing them to the language model. Techniques such as semantic search or keyword-based search (e.g., BM25) are employed to identify contextually relevant passages that are likely to contain procedural posture cues. The retrieved content is then supplied to the LLM, ensuring the model operates on focused, high-signal input rather than the entire document, thereby improving both efficiency and prediction accuracy.

For demonstration purposes, I randomly sampled a small subset of data within a Jupyter notebook to illustrate how to build a LangGraph workflow that iteratively retrieves document chunks relevant to procedural posture.

The following graph structure illustrates the Retrieval-Augmented Generation (RAG) workflow with iterative query refinement.



Starting from the original user question associated with procedural posture, the node: "**generate_rewritten_query**" in the graph, generates multiple rewritten versions of the original query, which create diverse variations to improve retrieval coverage.

```
========================================
Original Questions:

what is the procedural posture in the document where the
procedural posture describes the procedural
history including any prior decisions under appeal.

========================================
Rewritten Questions:

1. Can you explain the procedural posture in the document,
detailing the procedural history and any
previous decisions that are under appeal?
```

```
2. What does the procedural posture in the document reveal about
the procedural history and any prior
decisions that are currently being appealed?
3. How is the procedural posture outlined in the document,
including the procedural history and any
earlier decisions that are subject to appeal?
```

Then for each query(original + rewritten), the node:
retrieve_documents, retrieves relevant document chunks using semantic search from embedding vector databases(ChromDB). The workflow only retrieves information from the specific document_id being processed. Multiple document chunks that match the queries semantically will be retrieved for later evaluation.

Each retrieved document chunk will be evaluated for relevance in the node "**Grade Document**", those retrieved chunk which are relevant to the user's question will be reserved, therefore the workflow filtered a list of only relevant document chunks.

These processes iterate multiple times.

- If the filter_documents is not empty, relevant information related to procedural posture has been found, we proceed to output these relevant documents.
- If the number of iteration exceeds the maximal number allowed, the workflow terminates the entire retrieval process forcefully in order to avoid infinite loop and output whatever documents were found(may be empty).
- If no relevant document was found and maximal attempts have not been hit, the workflow will loop back to try again with new queries and generate new rewritten queries and restart the retrieval process

The key workflow conditions are displayed in the following script. This system implements adaptive query refinement. It creates an intelligent, self-improving retrieval system that adapts its search strategy based on the quality of results found.

```python
# Simplified decision logic
def decide_to_rewrite_query(state):
    filtered_documents = state["documents"]
    attempted_generations = state["attempted_generations"]
```

```
    if filtered_documents:
        return "output relevant"  # Found something useful
    elif attempted_generations > MAX_ATTEMPTS:
        return "exceed maximal iteration"  # Give up
    else:
        return "rewrite query"  # Try again with new queries
```

The final output is organized into a structured format, as illustrated in the table above. For each legal document, the input field “full_text” contains the original content. Using an adaptive LangGraph workflow, we retrieve only the segments most relevant to procedural posture, which are stored in the “relevant_text” field.

This filtering and condensing step ensures that only high-signal, contextually important information is passed to the language model, rather than overwhelming it with the full, often noisy document. As a result, the LLM can focus on critical procedural details, leading to improved comprehension and more accurate predictions of posture labels.

	document_id	full_text	postures	relevant_text
0	I55973ee08ab111e88d669565240b92b2	In a proceeding pursuant to Mental Hygiene Law...	On Appeal	In a proceeding pursuant to Mental Hygiene Law...
1	I1722882095ad11e8809390da5fe55bec	Defendant-appellant Ronald Epps appeals from a...	Appellate Review, Trial or Guilt Phase Motion ...	guilt. Thus, we conclude that Epps's Brady cha...
2	I09ec1a27edff11e6b92bf4314c15140f	Proceeding pursuant to CPLR article 78, in eff...	Review of Administrative Decision	Proceeding pursuant to CPLR article 78, in eff...
3	If2fec26a8e1411e5a795ac035416da91	In an action, inter alia, to recover damages f...	Motion to Dismiss, On Appeal	In an action, inter alia, to recover damages f...
4	Id199ee2b373411e380938e6f51729d80	Appeal from an order of the Family Court of Ch...	On Appeal	15 N.Y.3d 703, 2010 WL 2606035 [2010]; Rossit...
5	I5f180eb0a72911e8ba29f178bdd7ef1e	¶ 1 Travis Padgett appeals trial court orders ...	Appellate Review	appellate counsel, there was no need for Mr. P...
6	I4abc65509c4411e89b71ea0c471daf33	Desima James appeals his 63-month sentence aft...	Appellate Review, Sentencing or Penalty Phase ...	meaningfully limit his right of allocution, es...
7	I4e847ba0e49711e8aec5b23c3317c9c0	It is hereby ORDERED that the judgment so appe...	Appellate Review	We also reject the contention in defendant's p...

4.3 Further Exploration:

While the current study presents a strong foundation, there remain several promising avenues for future experimentation and enhancement.

In this study, I primarily focused on using an encoder-only model, specifically ModernBERT, to predict procedural posture labels by treating the task as a multi-labels classification problem. Each label is handled as a discrete category, and the model is trained to select the

most appropriate label(s) based on the input representation. However, this is only one of several viable strategies for modeling this task.

An alternative and potentially more powerful approach involves leveraging decoder-based language models—such as the LLaMA or Qwen families—and reframing the problem as a text generation task. Instead of forcing the model to classify from a fixed set of categories, we can fine-tune these models to generate the procedural posture labels directly as natural language outputs. This approach opens the door for instruction tuning, particularly using Parameter-Efficient Fine-Tuning (PEFT) methods like LoRA or QLoRA, where we can guide the model to "think" and "respond" with label predictions in a more human-like and explainable way.

Moreover, this generation-based strategy allows for multi-label prediction without needing to predefine rigid label boundaries. The model can generate one or more appropriate postures in free-text format, which may be better suited for capturing nuanced legal language and overlapping procedural contexts. Unfortunately, due to time constraints, I was not able to implement and evaluate this promising direction within the current scope of work.

Additionally, the generation process could be further optimized using Reinforcement Learning from Human Feedback (RLHF) or synthetic preference data. By rewarding the model for generating accurate, relevant, and interpretable posture labels, we can teach it not just what to predict, but also how to reason through complex procedural information.

Lastly, while the prior discussion of Retrieval-Augmented Generation (RAG) focused primarily on the retrieval component—extracting relevant sections from long legal documents—it's important to note that the generation component offers another compelling opportunity. Once we retrieve the most relevant content, we can prompt the LLM to generate the corresponding procedural posture(s) directly, rather than relying on classification over predefined categories. This could further improve model interpretability, adaptability, and performance in handling real-world legal texts.

In summary, future work could significantly benefit from exploring decoder-based generative models, instruction tuning, RLHF, and full

RAG pipelines and even multi-agents systems, all of which hold strong potential to enhance the accuracy and expressiveness of procedural posture prediction in NLP.