

Final Project Report

**Brunda Mariswamy**

**The George Washington University**

**DATS 6312: NLP**

**Professor AmirJafari**

**Table of Contents**

[1.Introduction 2](#_Toc153208093)

[2.Description of individual work 2](#_Toc153208094)

[3. Detail description. 3](#_Toc153208095)

[1.Data Collection 3](#_Toc153208096)

[2.Data Pre-Processing 4](#_Toc153208097)

[3.Split data for pre-training 4](#_Toc153208098)

[4.Implementation 4](#_Toc153208099)

[4.Results 7](#_Toc153208100)

[Results for Prediction of Legal Judgement 7](#_Toc153208101)

[Accuracy Report 7](#_Toc153208102)

[Evaluating on validation set of data. 8](#_Toc153208103)

[Evaluating on Test set of data 10](#_Toc153208104)

[5.Summary 13](#_Toc153208105)

[6.Coding Calculation 13](#_Toc153208106)

[7.References 13](#_Toc153208107)

**Legal Document Summarization and Analysis**

# 1.Introduction

The problem our group has selected is the summarization and analysis of legal documents. The legal industry is inundated with large volumes of complex documents such as contracts, court rulings, and legislation. These documents are often lengthy and difficult for nonexperts to understand. Efficiently summarizing and analyzing these texts can significantly benefit legal professionals and the public by saving time, reducing costs, and enhancing understanding. We chose this problem because there is a growing demand for tools that can simplify the consumption of legal information, and we believe that Natural Language Processing (NLP) offers promising solutions.

We as a team worked on different tasks Text summarization (both extractive and abstractive), Sentiment analysis to gauge the tone of the document, Q&A of legal documents, Named Entity Recognition to identify legal entities.

# 2.Description of individual work

I worked on two NLP tasks such as Predicting Legal Judgment and Extracting Legal Named Entities.

Prediction of Legal Judgement

Automatically predicting the outcome of court cases has been widely studied in recent years. This has mostly been done via the task of Legal Judgment Prediction (LJP), where, given the facts of a case, we are to predict the laws/statutes violated, applicable charges and terms of penalty. Along these lines, the CJPE (Court Judgement Prediction and Explanation) task aims to predict the final decision of the court, based on the rest of the case judgement (i.e., the input is the case judgement document with the final decision removed). In the Indian scenario, a court case usually consists of one or multiple appeals/claims filed by the appellant against the respondent, and the judge needs to provide an ‘accept’ or ‘reject’ decision for each of these claims. Thus, this can be modeled as a binary text classification task.

NLP in the legal domain has seen increasing success with the emergence of Transformer-based Pre-trained Language Models (PLMs) pre-trained on legal text. PLMs trained over European and US legal text are available publicly; however, legal text from other domains (countries), such as India, have a lot of distinguishing characteristics.

I experimented with BERT, used Hugging Face library to fine tune BASE models of transformers from Hugging Face on ILDC dataset and I fine-tuned a BERT base Legal pretrained model on Indian legal pre-training corpus, which has a custom vocabulary specifically tailored for Indian legal text.

Among the combinations of input tokens, the best performance was obtained by using last 512 tokens as input to the BERT Base model. I observed t the trend that the more the tokens from the final parts of the document are taken as input, the better is the prediction performance. This observation agrees with the fact that there are more clues towards the correct prediction in the final parts of the document the decision etc. most aligned to the judgment are expected to appear more towards the end, closer to the judgment.

A diagram of a computer

Description automatically generated

Figure 1. The binary classification model employs BERT for contextualized embeddings, followed by a linear transformation layer applied to the **[CLS]** token's embedding for efficient binary document classification.

# 3. Detail description.

Given a case proceeding from the SCI, the task of COURT JUDGMENT PREDICTION is to automatically predict the decision for the case (with respect to the appellant)

**Prediction:** Given a case proceeding D, the task is to predict the decision y ∈ {0, 1}, where the label 1 corresponds to the acceptance of the appeal/petition of the appellant/petitioner.

## 1.Data Collection

The Indian Legal Documents Corpus for Court Judgment (ILDC) family of datasets, contain different types of training data. Among them, the ILDC-multi dataset consists of approx. 35K Supreme Court of India case documents each having multiple claims/appeals; a single label is assigned for each document – ‘accept’ if at least one appeal in that case document is accepted, ‘reject’ otherwise. For our current experiment, I considered the ILDC-multi dataset. The dataset has 4 columns text, label, split and name.

A screenshot of a test

Description automatically generated

Figure 2. Sample Dataset

## 2.Data Pre-Processing

Indian legal text is known to be noisy. There are many odd artefacts in the text, such as sequences of dashes or lines, page numbers and metadata in random positions of the text, etc. Most of such anomalies have crept into the text due to digitization of hand-written documents. We remove such artefacts during pre-training using Regular Expression matching.

A screenshot of a computer program

Description automatically generated

Figure 3. Preprocessing using regular expressions.

## 3.Split data for pre-training

The documents in ILDC data are split based on the column “split. Where values of the columns are 11042 Train,1517 Test number of legal documents.

## 4.Implementation

Worked on fine tuning the base model “law-ai/CustomInLawBERT”.The original base model is trained by considering (begin tokens, middle tokens, end tokens) to predict the judgement.But after reading through multiple legal case documents I saw that the information at the end portion of the documents contains information that can be helpful for Judgement Prediction task, hence To accommodate the model's input constraints, sequences longer than 510 tokens are truncated, and special tokens are added. The resulting sequences are then padded to a maximum length of 512 tokens. For each token in a sequence, the function generates an attention mask, assigning a value of 1 if the token is an actual content word (token ID > 0) and 0 otherwise. Notably, this process effectively identifies the positions of content tokens while distinguishing them from padding tokens. the BERT base model (12 layers, 768 hidden dimensionality, 12 attention heads, 110M parameters).

* The model was fine-tuned using PyTorch framework with below Hyperparameter values.
* Pretrained Model Used:” law-ai/CustomInLawBERT'' from hugging face hub,the model specifically designed for the Indian Legal domain. It features a custom.
* tokenizer with a vocabulary tailored to legal language. The model's architecture mirrors that of the popular "Bert-base-uncased" model, encompassing 12 hidden layers, a hidden dimensionality of 768,12 attention heads, and approximately 110 million parameters.
* Data Split: The dataset as column which identifies the data for train,test.Hence there are 33,299 number of legal documents for training and 1,517 for testing.
* Tokenization: The tokenization process involves breaking down input text into individual tokens using a BERT tokenizer. To adhere to BERT's maximum input token limit of 512, the code retains the last 510 tokens of each sequence. These tokenized sequences are then encoded into input IDs, and the function returns padded input IDs and lengths. This preparation is crucial for BERT-based sequence classification.
* Due to memory limitations, a minibatch size of 6 was chosen. Minibatch size represents the number of data samples processed together in each iteration during training, and a smaller size like 6 is selected to accommodate memory constraints.
* For optimizing the model's parameters during training, the Adam optimizer is employed with a learning rate of 2e-5. Various optimization techniques are implemented to address overfitting, including weight decay and gradient clipping. Weight decay helps prevent the model from becoming too specialized to the training data, while gradient clipping limits the size of the gradients during backpropagation, preventing potential exploding gradient issues.
* A learning rate scheduler is connected to the learning rate, dynamically adjusting it during training. In this case, the learning rate is adjusted every 3 epochs, aiding in the convergence of the model over the training iterations.
* Performance metrics: The code calculates accuracy and confusion matrix along with macro- and micro-precision, recall, and F1 score.
* Model saving: The model is saved after the training process,which can be later used for inference and can recreate the same model for later.
* During inference, a detection threshold was set at 0.5 to predict class 0 or 1.If threshold >0.5 then its class 1 else 0. The model is available in hugging face hub <https://huggingface.co/brundamariswamy/Indian-Custom-Bert> .

Hugging repo link for pretrained model and fine-tuned model.

<https://huggingface.co/law-ai/CustomInLawBERT>

<https://huggingface.co/brundamariswamy/Indian-Custom-Bert>

Extracting Legal Named Entities

Identification of named entities from legal texts is an essential building block for developing other legal Artificial Intelligence applications. Named Entities in legal texts are slightly different and more fine-grained than commonly used named entities like Person, Organization, Location etc. used pretrained transformer-based legal NER baseline model to extract the entities, where the model was trained on corpus of 14444 Indian court judgment sentences and 2126 judgment preambles annotated with 14 legal named entities.

A typical Indian court judgment can be split into two parts viz., preamble and judgment. The preamble of a judgment contains formatted metadata like names of parties, judges, lawyers, date, court etc. The text following the preamble till the end of the judgment is called "judgment". The preamble typically ends with keywords like JUDGMENT or ORDER etc. In case these keywords are not found, we treated the first occurrence of 2 consecutive sentences with a verb as the start of the judgment part. This is because the preamble typically contains formatted metadata and not grammatically complete sentences.

Some entities are extracted from the preamble, and some from the judgment text. Some entities are extracted from both the preamble and judgment, and their definitions may change depending on where they are extracted from. I did not fine tune this model instead used pretrained model to extract entities (Model - opennyaiorg/en\_legal\_ner\_trf).

|  |  |  |
| --- | --- | --- |
| Named Entity | Extract From | Description |
| COURT | Preamble, Judgment | Name of the court which has delivered the current judgement if extracted from the preamble. Name of any court mentioned if extracted from judgment sentences. |
| PETITIONER | Preamble, Judgment | Name of the petitioners/appellants/revisionist from current case |
| RESPONDENT | Preamble, Judgment | Name of the respondents/defendants/opposition from current case |
| JUDGE | Preamble, Judgment | Name of the judges from the current case if extracted from the preamble. Name of the judges of the current as well as previous cases if extracted from judgment sentences |
| LAWYER | Preamble | Name of the lawyers from both the parties |
| DATE | Judgment | Any date mentioned in the judgment |
| ORG | Judgment | Name of organizations mentioned in text apart from the court. |
| GPE | Judgment | Geopolitical locations which include names of states, cities, villages |
| STATUTE | Judgment | Name of the act or law mentioned in the judgement |
| PROVISION | Judgment | Sections, sub-sections, articles, orders, rules under a statute |
| PRECEDENT | Judgment | Precedent consists of party names + citation(optional) or case number (optional) |
| CASE\_NUMBER | Judgment | All the other case numbers mentioned in the judgment (apart from precedent) where party names and citation are not provided |
| WITNESS | Judgment | Name of witnesses in current judgment |
| OTHER\_PERSON | Judgment | Name of all the persons that are not included in petitioner, respondent, judge and witness |

Table 1: Legal Named Entities Definitions

Implementing streamlit app for Legal Judgment and Extracting Legal Named Entities.

To inference the results of the fine-tuned model I created a demo page on streamlit.

How It Works

**1.Upload a Legal Document:** Begin by uploading a document. Accepted formats include PDF, DOCX, and TXT.

**2.Choose Your Model:** Select from a range of AI models optimized for legal text analysis (Indian-Legal-Bert or Indian-Custom-Bert)

**3.Document Analysis:** The tool will Predict whether the appeals/claims filed by the appellant against the respondent is Accepted /Rejected and extract its Legal Named Entities from the document uploaded.

**4.Outcome Prediction:** Based on the analysis, it will also predict potential outcomes or implications.

# 4.Results

## Results for Prediction of Legal Judgement

### Accuracy Report

In the initial phase, our pre-trained model, with a baseline accuracy of 65%, demonstrated its adaptability across various tasks. However, through a meticulous fine-tuning process on our task-specific dataset for just 3 epochs, we observed a remarkable surge in accuracy to 72%.

Epoch 1: Training loss – 0.61, Accuracy - 0.69

A screenshot of a computer

Description automatically generated

Epoch 2: Training loss – 0.49 Accuracy - 0.71

A screen shot of a computer

Description automatically generated

Epoch 3: Training loss – 0.38 Accuracy – 0.72

A screenshot of a computer

Description automatically generated

This 6% improvement not only underscores the effectiveness of leveraging pre-trained knowledge but also highlights the model's ability to refine its parameters for enhanced task performance. The noteworthy boost in accuracy substantiates the importance of fine-tuning in tailoring a pre-trained model to suit the intricacies of our specific dataset, thereby showcasing its potential for superior predictive capabilities.

### Evaluating on validation set of data.

The Validation set consisting of 994 legal documents, the model achieved an accuracy of approximately 71.93%. This implies that the model made correct predictions for the class labels in nearly 72% of instances.

A screen shot of a computer program

Description automatically generated

Confusion matrix for validation set.

A blue squares with numbers and labels

Description automatically generated

**True Positives (TP):** 273: This is the number of instances where the model correctly predicted positive (class 1) when the actual label was positive.

**True Negatives (TN):** 442:This is the number of instances where the model correctly predicted negative (class 0) when the actual label was negative.

**False Positives (FP):** 55: This is the number of instances where the model incorrectly predicted positive when the actual label was negative.

**False Negatives (FN):** 224: This is the number of instances where the model incorrectly predicted negative when the actual label was positive.

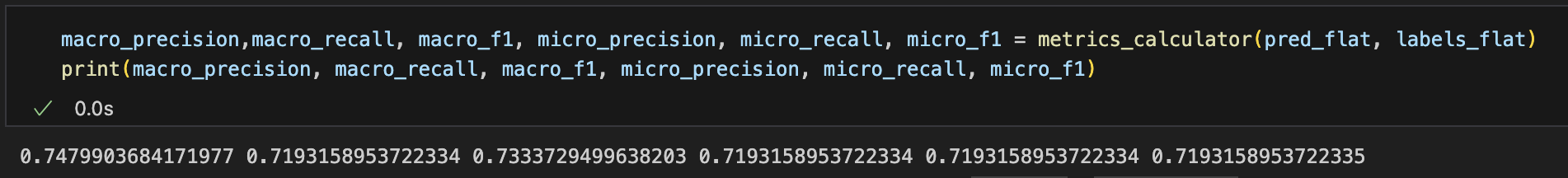
Mathews Correlation

The Matthews Correlation Coefficient (MCC) value of 0.4664 which is greater than 0 and indicates a moderately positive correlation between your predicted and true labels.

A computer screen with colorful text

Description automatically generated

A macro perspective considers the average performance per class, ensuring a balanced evaluation. On the other hand, micro metrics, which account for the global count of true positives, false positives, and false negatives, provide an overview of the model's overall precision, recall, and F1-Score. The consistently high values for both macro and micro metrics (precision: 0.748, recall: 0.719, F1-Score: 0.733) suggest that the model achieves a well-rounded performance across individual classes and the entire dataset, demonstrating its effectiveness in classification tasks.



A graph of a comparison of micro and micro

Description automatically generated

An AUC of 0.81 in the Precision-Recall (PR) curve and on ROC auc=0.81indicates strong model performance, striking a good balance between precision and recall. This suggests the model is effective at accurately identifying positive instances while capturing a a significant portion of all actual positive instances.

A green line graph with white text

Description automatically generated A graph of a function

Description automatically generated

### Evaluating on Test set of data

The test set consisting of 1,517 legal documents, the model achieved an accuracy of approximately 72.70%. This implies that the model made correct predictions for the class labels in nearly 72.7% of instances.

A screen shot of a computer code

Description automatically generated

Confusion matrix for Test set.

A blue squares with white text

Description automatically generated

**True Positives (TP):** 447 instances were correctly predicted as Class 1.

**True Negatives (TN):** 656 instances were correctly predicted as Class 0.

**False Positives (FP):** 99 instances were predicted as Class 1 but belonged to Class 0.

**False Negatives (FN):** 315 instances were predicted as Class 0 but belonged to Class 1.

Mathews Correlation

The Matthews Correlation Coefficient (MCC) value of 0.4745which is greater than 0 and indicates a moderately positive correlation between your predicted and true labels.

A screen shot of a computer program

Description automatically generated

A macro perspective considers the average performance per class, ensuring a balanced evaluation. On the other hand, micro metrics, which account for the global count of true positives, false positives, and false negatives, provide an overview of the model's overall precision, recall, and F1-Score. The consistently high values for both macro and micro metrics (precision: 0.748, recall: 0.727, F1-Score: 0.737) suggest that the model achieves a well-rounded performance across individual classes and the entire dataset, demonstrating its effectiveness in classification tasks.

A screen shot of a computer code

Description automatically generated

A screenshot of a graph

Description automatically generated

An AUC of 0.83 in the Precision-Recall (PR) curve and on ROC AUC =0.82 indicates strong model performance, striking a good balance between precision and recall. This suggests the model is effective at accurately identifying positive instances while capturing a significant portion of all actual positive instances.

A graph of a function

Description automatically generated A graph of a curve

Description automatically generated

Inferencing results of streamlit app

**Upload a Legal Document:** Begin by uploading a document. Accepted formats include PDF, DOCX, and TXT.

A screenshot of a legal document

Description automatically generated

**Choose Your Model:** Select from a range of AI models optimized for legal text analysis (Indian-Legal-Bert or Indian-Custom-Bert)

A screenshot of a phone

Description automatically generated

**Document Analysis:** The tool will Predict whether the appeals/claims filed by the appellant against the respondent is Accepted /Rejected and extract its Legal Named Entities from the document uploaded.

A graph of a bar chart

Description automatically generated

A document with text on it

Description automatically generated

# 5.Summary

Predicting Legal Judgment and Extracting Legal Named Entities using pretrained Bert model to fine-tune BERT models on an Indian Legal Documents. The results highlight a notable accuracy improvement in fine -tuned Legal Judgment Prediction model, reaching 72% while the pretrained model had an accuracy of 65%, with robust performance on validation and test sets. The implementation of a Streamlit app allows users to upload legal documents, providing a practical tool for legal professionals to gain insights into legal outcomes and identify entities within case documents. In future I would like to use ROBERTA and DISTILBERT to train the model for Indian Legal Documents and compare results all three BERTS.

# 6.Coding Calculation

Number of lines of code found in internet - ~327(Train and streamlit)

Number of lines modified - ~220.

Number of lines added - ~ 80.

Calculation – ~30%

# 7.References

1. Pre-trained Language Models for the Legal Domain: A Case Study on Indian Law Shounak Paul, Arpan Mandal, Pawan Goyal, Saptarshi Ghosh.<https://arxiv.org/pdf/2211.03442v1.pdf>
2. ILDC for CPJE: Indian Legal Documents Corpus for Court Judgment Prediction and    Explanation Vijit Malik, Rishabh Sanjay, Shubham Kumar Nigam, Kripabandhu Ghosh, Shouvik Kumar Guha, Arnab Bhattacharya, Ashutosh Modi.<https://arxiv.org/pdf/2209.06049.pdf>

**Image References**

BERT architecture <https://www.semanticscholar.org/paper/Effectively-Leveraging-BERT-for-Legal-Document-Limsopatham/5bdfbded52da393eb3f7a0270a4f135bb564ee4a>