THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Final Project Report

Brunda Mariswamy

The George Washington University

DATS 6312: NLP

Professor AmirJafari

Table of Contents

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.

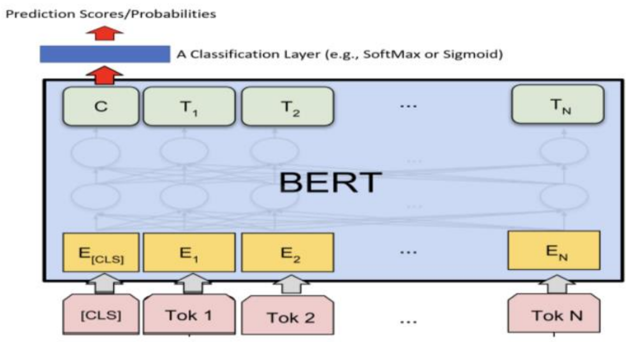

Prediction Scores/Probabilities
A Classification Layer (e.g., SoftMax or Sigmoid)
C
T_1
T_2
...
T_N
BERT
$E_{[CLS]}$
E_1
E_2
...
E_N
[CLS]
Tok 1
Tok 2
...
Tok N

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.

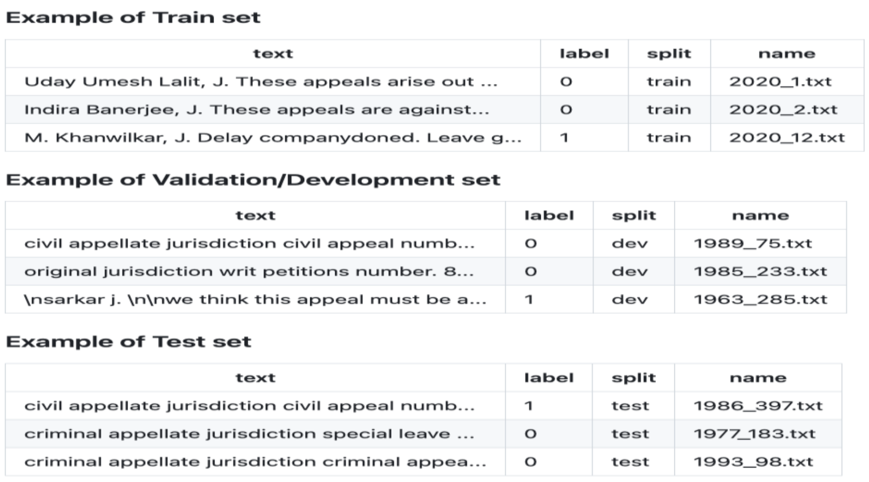
3. Detail description.

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.



Example of Train set

text	label	split	name
Uday Umesh Lalit, J. These appeals arise out ...	0	train	2020_1.txt
Indira Banerjee, J. These appeals are against...	0	train	2020_2.txt
M. Khanwilkar, J. Delay companydoned. Leave g...	1	train	2020_12.txt

Example of Validation/Development set

text	label	split	name
civil appellate jurisdiction civil appeal numb...	0	dev	1989_75.txt
original jurisdiction writ petitions number. 8...	0	dev	1985_233.txt
\nsarkar j. \n\nwe think this appeal must be a...	1	dev	1963_285.txt

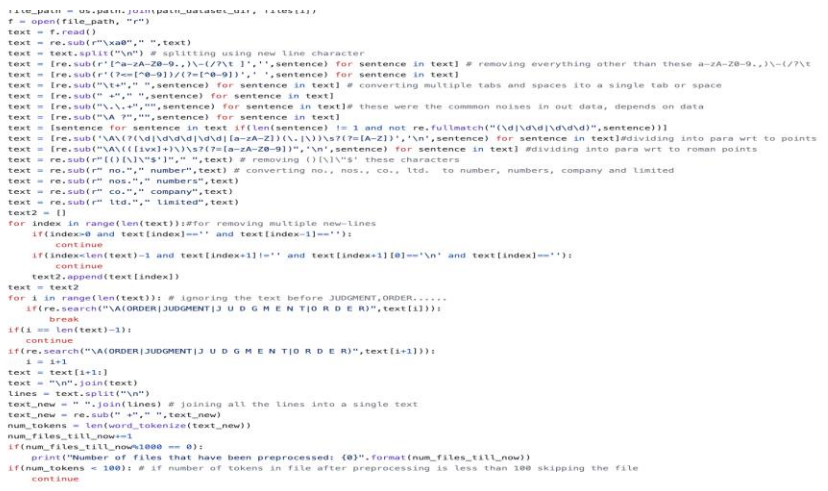
Example of Test set

text	label	split	name
civil appellate jurisdiction civil appeal numb...	1	test	1986_397.txt
criminal appellate jurisdiction special leave ...	0	test	1977_183.txt
criminal appellate jurisdiction criminal appea...	0	test	1993_98.txt

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.



```python
file_path = os.path.join(path_dataset_dir, files[i])
f = open(file_path, "r")
text = f.read()
text = re.sub(r"\xa0"," ",text)
text = text.split("\n") # splitting using new line character
text = [re.sub(r'[^a-zA-Z0-9.,)\-(/?\t ]','',sentence) for sentence in text] # removing everything other than these a-zA-Z0-9.,)\-(/?\t
text = [re.sub(r'(?<=[^0-9])/(?=[^0-9])',' ',sentence) for sentence in text]
text = [re.sub("\t+"," ",sentence) for sentence in text] # converting multiple tabs and spaces ito a single tab or space
text = [re.sub(" +"," ",sentence) for sentence in text]
text = [re.sub("\.\.+","",sentence) for sentence in text]# these were the commmon noises in out data, depends on data
text = [re.sub("\A ?","",sentence) for sentence in text]
text = [sentence for sentence in text if(len(sentence) != 1 and not re.fullmatch("(\d|\d\d|\d\d\d)",sentence))]
text = [re.sub('\A\(?(\d|\d\d\d|\d\d|[a-zA-Z])(\.|\))\s?(?=[A-Z])','\n',sentence) for sentence in text]#dividing into para wrt to points
text = [re.sub("\A\(([ivx]+)\)\s?(?=[a-zA-Z0-9])",'\n',sentence) for sentence in text] #dividing into para wrt to roman points
text = re.sub(r"[()[\]\"$']"," ",text) # removing ()[\]\"$' these characters
text = re.sub(r" no."," number",text) # converting no., nos., co., ltd.  to number, numbers, company and limited
text = re.sub(r" nos."," numbers",text)
text = re.sub(r" co."," company",text)
text = re.sub(r" ltd."," limited",text)
text2 = []
for index in range(len(text)):#for removing multiple new-lines
    if(index>0 and text[index]=='' and text[index-1]==''):
        continue
    if(index<len(text)-1 and text[index+1]!='' and text[index+1][0]=='\n' and text[index]==''):
        continue
    text2.append(text[index])
text = text2
for i in range(len(text)): # ignoring the text before JUDGMENT,ORDER......
   if(re.search("\A(ORDER|JUDGMENT|J U D G M E N T|O R D E R)",text[i])):
       break
if(i == len(text)-1):
   continue
if(re.search("\A(ORDER|JUDGMENT|J U D G M E N T|O R D E R)",text[i+1])):
   i = i+1
text = text[i+1:]
text = "\n".join(text)
lines = text.split("\n")
text_new = " ".join(lines) # joining all the lines into a single text
text_new = re.sub(" +"," ",text_new)
num_tokens = len(word_tokenize(text_new))
num_files_till_now+=1
if(num_files_till_now%1000 == 0):
    print("Number of files that have been preprocessed: {0}".format(num_files_till_now))
if(num_tokens < 100): # if number of tokens in file after preprocessing is less than 100 skipping the file
    continue
```

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 hyperparameter set for tuning the model with epoch 3 with learning rate ‘2e-5’.

Maximum Gradient Norm used for gradient clipping to prevent exploding gradients during backpropagation. Gradients are scaled down if their norm exceeds 1.0. Total number of steps is calculated by multiplying the number of batches in the training dataset (len(train_dataloader)) by the number of epochs (epochs). Number of warmups steps the number of initial steps where the learning rate is gradually increased. It allows the model to stabilize before applying aggressive updates. Seed value is set to 32. The model is saved and pushed to Hugging face repo(“brundamariswamy/Indian-Custom-Bert”) can be downloaded and used for predicting legal judgements.

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.

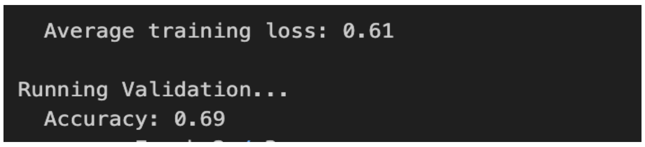
4.Results

Results for Prediction of Legal Judgement

Accuracy Report

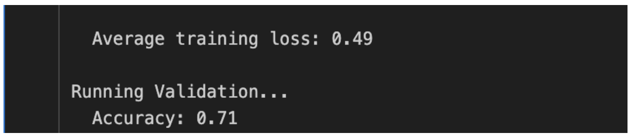
In the initial phase, our pre-trained model, with a baseline accuracy of 65%, demonstrated its adaptability across various tasks. However, through a meticulous fine-tuning process on our task-specific dataset for just 3 epochs, we observed a remarkable surge in accuracy to 72%.

Epoch 1: Training loss – 0.61, Accuracy - 0.69



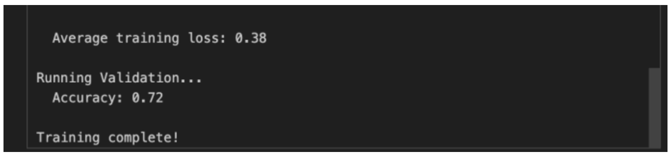
```
  Average training loss: 0.61

Running Validation...
  Accuracy: 0.69
```

Epoch 2: Training loss – 0.49 Accuracy - 0.71



```
  Average training loss: 0.49

Running Validation...
  Accuracy: 0.71
```

Epoch 3: Training loss – 0.38 Accuracy – 0.72

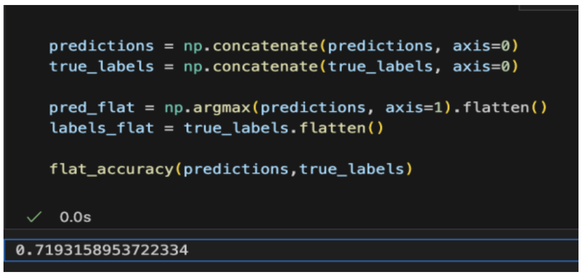


```
  Average training loss: 0.38

Running Validation...
  Accuracy: 0.72

Training complete!
```

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.

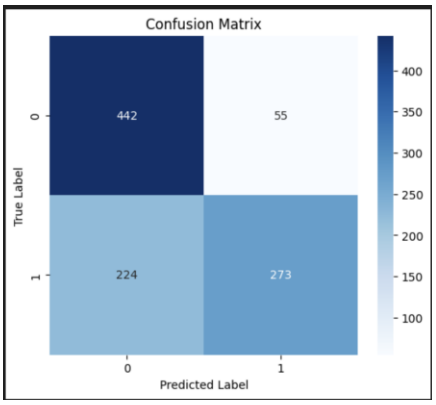
Evaluating on validation set of data.

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.



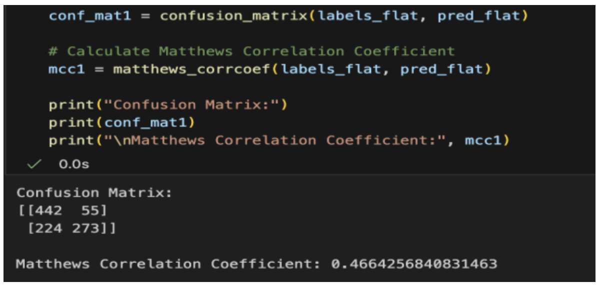
```
predictions = np.concatenate(predictions, axis=0)
true_labels = np.concatenate(true_labels, axis=0)

pred_flat = np.argmax(predictions, axis=1).flatten()
labels_flat = true_labels.flatten()

flat_accuracy(predictions,true_labels)

✓ 0.0s

0.7193158953722334
```

Confusion matrix for validation set.



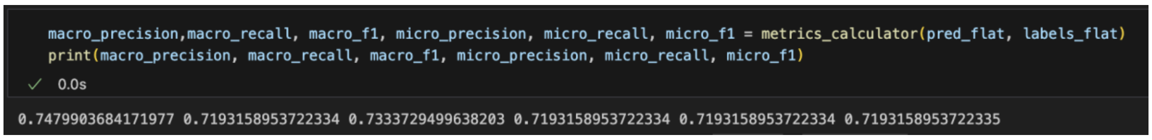
True Positives (TP): 273: This is the number of instances where the model correctly predicted positive (class 1) when the actual label was positive.
True Negatives (TN): 442:This is the number of instances where the model correctly predicted negative (class 0) when the actual label was negative.
False 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.

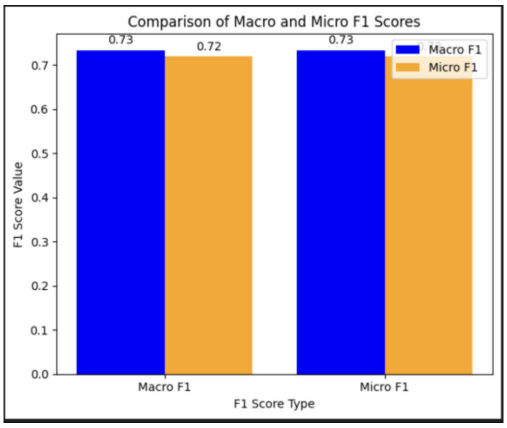


```
conf_mat1 = confusion_matrix(labels_flat, pred_flat)

# Calculate Matthews Correlation Coefficient
mcc1 = matthews_corrcoef(labels_flat, pred_flat)

print("Confusion Matrix:")
print(conf_mat1)
print("\nMatthews Correlation Coefficient:", mcc1)
✓ 0.0s

Confusion Matrix:
[[442  55]
 [224 273]]

Matthews Correlation Coefficient: 0.4664256840831463
```

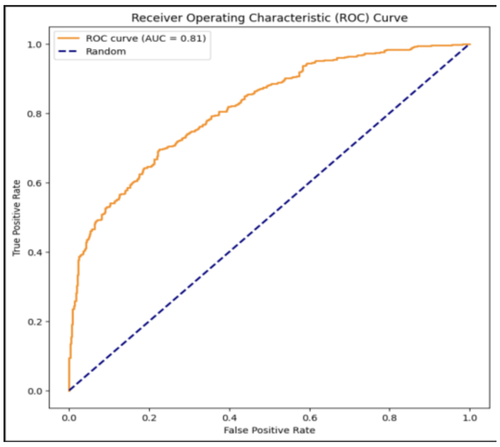
A macro perspective considers the average performance per class, ensuring a balanced evaluation. On the other hand, micro metrics, which account for the global count of true positives, false positives, and false negatives, provide an overview of the model's overall precision, recall, and F1-Score. The 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.

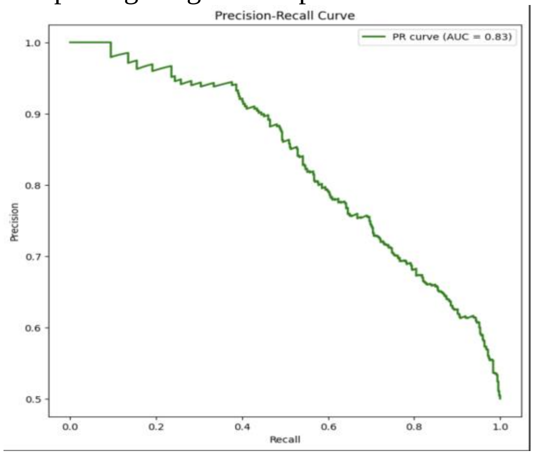


```
macro_precision,macro_recall, macro_f1, micro_precision, micro_recall, micro_f1 = metrics_calculator(pred_flat, labels_flat)
print(macro_precision, macro_recall, macro_f1, micro_precision, micro_recall, micro_f1)
✓ 0.0s

0.7479903684171977 0.7193158953722334 0.7333729499638203 0.7193158953722334 0.7193158953722334 0.7193158953722335
```



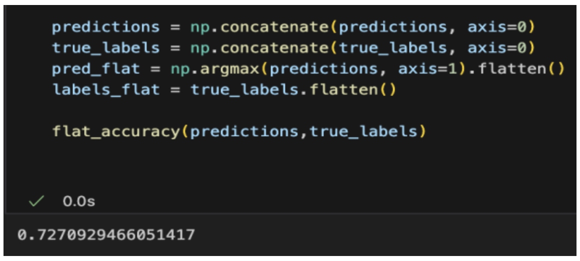
An AUC of 0.83 in the Precision-Recall (PR) curve and on ROC auc=0.81indicates strong model performance, striking a good balance between precision and recall. This suggests the model is effective at accurately identifying positive instances while capturing a significant portion of all actual positive instances.





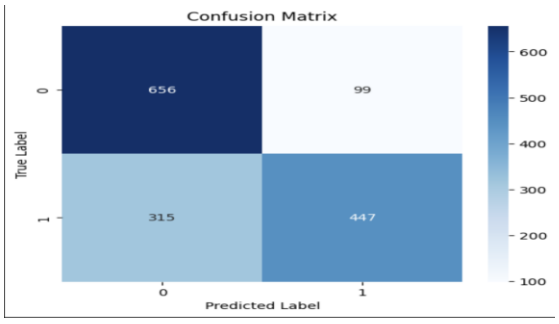
Evaluating on Test set of data

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.



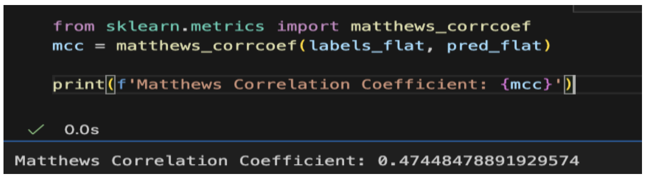
```
predictions = np.concatenate(predictions, axis=0)
true_labels = np.concatenate(true_labels, axis=0)
pred_flat = np.argmax(predictions, axis=1).flatten()
labels_flat = true_labels.flatten()

flat_accuracy(predictions,true_labels)

✓ 0.0s
0.7270929466051417
```

Confusion matrix for Test set.



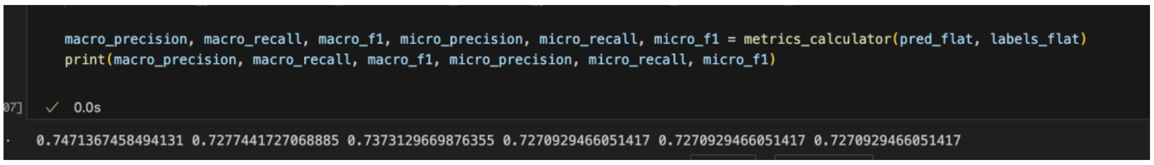
True Positives (TP): 447 instances were correctly predicted as Class 1.
True Negatives (TN): 656 instances were correctly predicted as Class 0.
False Positives (FP): 99 instances were predicted as Class 1 but belonged to Class 0.
False Negatives (FN): 315 instances were predicted as Class 0 but belonged 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.

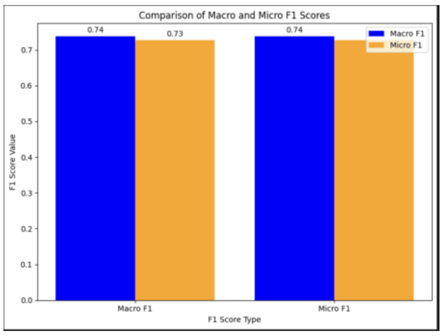


```
from sklearn.metrics import matthews_corrcoef
mcc = matthews_corrcoef(labels_flat, pred_flat)

print(f'Matthews Correlation Coefficient: {mcc}')

✓ 0.0s

Matthews Correlation Coefficient: 0.47448478891929574
```

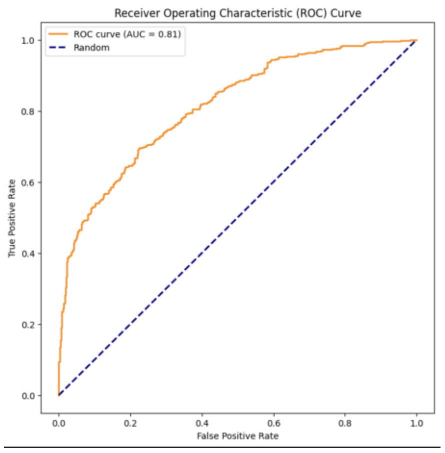
A macro perspective considers the average performance per class, ensuring a balanced evaluation. On the other hand, micro metrics, which account for the global count of true positives, false positives, and false negatives, provide an overview of the model's overall precision, recall, and F1-Score. The 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.

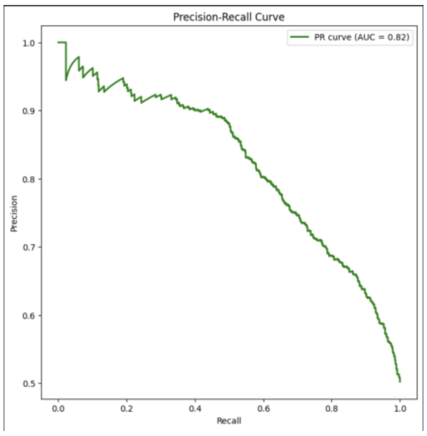


```
macro_precision, macro_recall, macro_f1, micro_precision, micro_recall, micro_f1 = metrics_calculator(pred_flat, labels_flat)
print(macro_precision, macro_recall, macro_f1, micro_precision, micro_recall, micro_f1)

07]  ✓ 0.0s

0.7471367458494131 0.7277441727068885 0.7373129669876355 0.7270929466051417 0.7270929466051417 0.7270929466051417
```



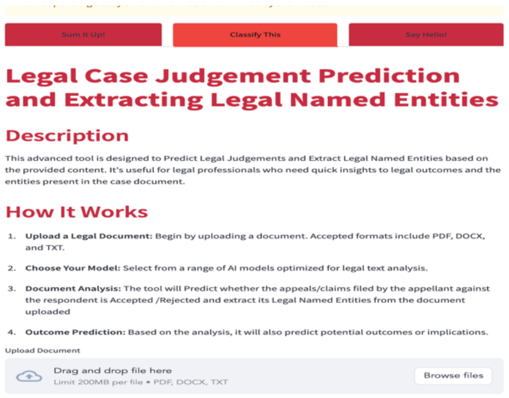
An AUC of 0.83 in the Precision-Recall (PR) curve and on ROC AUC =0.82 indicates strong model performance, striking a good balance between precision and recall. This suggests the model is effective at accurately identifying positive instances while capturing a significant portion of all actual positive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





Inferencing results of streamlit app

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



Legal Case Judgement Prediction and Extracting Legal Named Entities

Description

This advanced tool is designed to Predict Legal Judgements and Extract Legal Named Entities based on the provided content. It's useful for legal professionals who need quick insights to legal outcomes and the entities present in the case document.

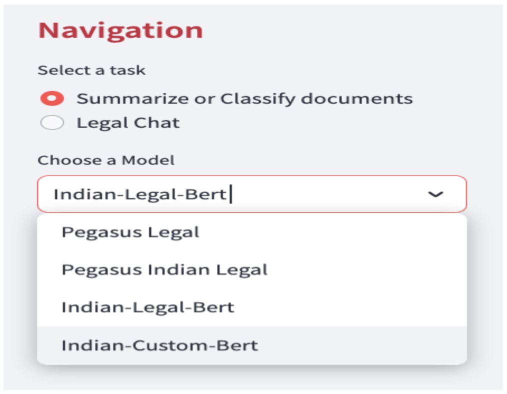
How It Works

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.
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.

Upload Document

Drag and drop file here
Limit 200MB per file • PDF, DOCX, TXT

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



Navigation

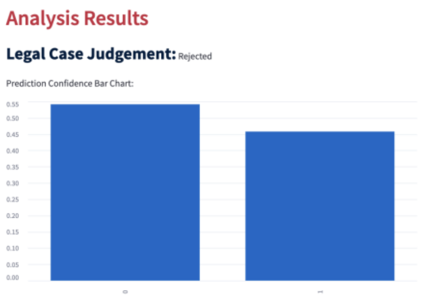
Select a task

[✓ Summarize or Classify documents]
Legal Chat

Choose a Model

Indian-Legal-Bert

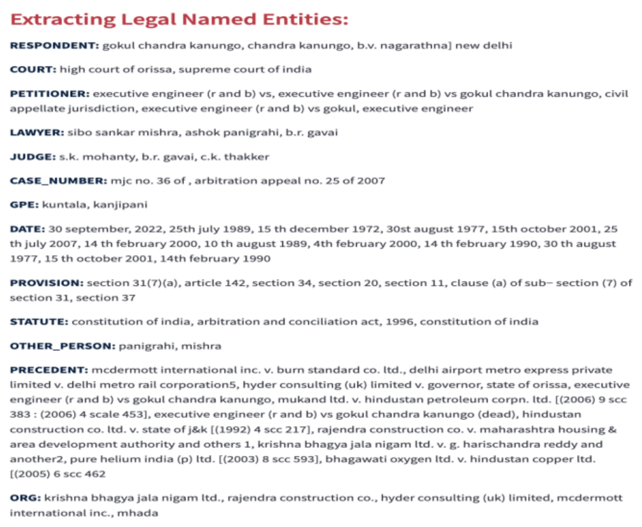
Document Analysis: The tool will Predict whether the appeals/claims filed by the appellant against the respondent is Accepted /Rejected and extract its Legal Named Entities from the document uploaded.



Analysis Results

Legal Case Judgement: Rejected

Prediction Confidence Bar Chart:



Extracting Legal Named Entities:

RESPONDENT: gokul chandra kanungo, chandra kanungo, b.v. nagarathna] new delhi

COURT: high court of orissa, supreme court of india

PETITIONER: executive engineer (r and b) vs, executive engineer (r and b) vs gokul chandra kanungo, civil appellate jurisdiction, executive engineer (r and b) vs gokul, executive engineer

LAWYER: sibo sankar mishra, ashok panigrahi, b.r. gavai

JUDGE: s.k. mohanty, b.r. gavai, c.k. thakker

CASE_NUMBER: mjc no. 36 of , arbitration appeal no. 25 of 2007

GPE: kuntala, kanjipani

DATE: 30 september, 2022, 25th july 1989, 15 th december 1972, 30st august 1977, 15th october 2001, 25 th july 2007, 14 th february 2000, 10 th august 1989, 4th february 2000, 14 th february 1990, 30 th august 1977, 15 th october 2001, 14th february 1990

PROVISION: section 31(7)(a), article 142, section 34, section 20, section 11, clause (a) of sub– section (7) of section 31, section 37

STATUTE: constitution of india, arbitration and conciliation act, 1996, constitution of india

OTHER_PERSON: panigrahi, mishra

PRECEDENT: mcdermott international inc. v. burn standard co. ltd., delhi airport metro express private limited v. delhi metro rail corporation5, hyder consulting (uk) limited v. governor, state of orissa, executive engineer (r and b) vs gokul chandra kanungo, mukand ltd. v. hindustan petroleum corpn. ltd. [(2006) 9 scc 383 : (2006) 4 scale 453], executive engineer (r and b) vs gokul chandra kanungo (dead), hindustan construction co. ltd. v. state of j&k [(1992) 4 scc 217], rajendra construction co. v. maharashtra housing & area development authority and others 1, krishna bhagya jala nigam ltd. v. g. harischandra reddy and another2, pure helium india (p) ltd. [(2003) 8 scc 593], bhagawati oxygen ltd. v. hindustan copper ltd. [(2005) 6 scc 462

ORG: krishna bhagya jala nigam ltd., rajendra construction co., hyder consulting (uk) limited, mcdermott international inc., mhada

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) https://huggingface.co/law-ai/CustomInLawBERT
2) https://aclanthology.org/2021.acl-long.313.pdf
3) https://arxiv.org/pdf/2209.06049.pdf
4) https://arxiv.org/pdf/2211.03442v1.pdf