SemEval 2023 Task 6: Rhetorical Roles Prediction (RR): Legal Document Segmentation

Rohan Chaudhury, Shubham Gupta, Hemal Mamtora, Abhishek Maurya, Upasana Mishra
Department of Computer Science
Texas A&M University
{rohan.chaudhury, guptashubham1798, hemal, tamuabhi-123, umishra24}@tamu.edu

Abstract

In the legal domain judgment summarizing, judgment outcome prediction, precedent search, etc require the segmentation of long and unstructured legal documents into semantically coherent text segments. This paper attempts to segment a given legal document into topically semantic and coherent units and assign a label to each segment which is referred to as Rhetorical Roles (RR). The legal document needs to be considered as a set of sequential sentences as evident from our experiments. The classification of the sentences requires the prediction of a single RR label from 13 different labels. Thus we considered the task of classifying sentences in Legal Documents based on their respective rhetorical roles. We tried different kinds of approaches ranging from Statistical models, Information Retrieval techniques, Deep Neural Network architectures, and fine-tuning pretrained Large Language Models. The model that resulted in the best performance considered contextual relationships between sequential sentences using T5 encoders combined with Bidirectional LSTM and Conditional Random Field.

1 Introduction

The number of open cases in populous nations (like India) has been increasing rapidly. For instance, the National Judicial Data Grid for India estimates that as of December 2021, there were about 40 million cases pending in the nation's courts (National Judicial Data Grid, 2021) (Malik et al., 2021). India has a common-law system, so it may not be possible to fully automate the judicial pipeline due to the subjectivity involved. However, many intermediate tasks can be automated to support legal practitioners and speed up the system. Natural Language Processing (NLP) techniques, for instance, can be used to process legal documents in order to organize and structure the data so that it is suitable for automatic search and retrieval.

Legal information retrieval systems that can successfully meet user-specific information demands are becoming more and more popular as a result of the complexity of legal situations becoming more and more sophisticated. Nevertheless, these downstream systems frequently demand that papers be correctly prepared and split, which is sometimes accomplished with rather straightforward preprocessing processes, neglecting the thematic coherence of parts. Systems often rely on sentence or paragraph-level representations, which may be missing important context, or document-level representations, which are too lengthy for useful search results. Written texts frequently consist of a series of semantically related paragraphs that are intended to smoothly flow from one subtopic to another. Typically, a user's information demands are met by accessing only the pertinent subtopic.

Over two million recent court rulings from the United States that were made available online serve as an illustration. These papers are often presented in a human-readable format, meaning that they include formatting and text alignment that makes it easier for humans to understand a document's structure and organization and makes it easier to navigate and locate specific legal provisions within the text. This is consistent with how legal documents have traditionally been used. The user receives whole documents (usually web pages) via standard generic web search engines consisting of hundreds of pages. Users only expect the pertinent portions of papers, therefore rapid and efficient solutions require systems that can identify and extract these pertinent portions of documents.

In this project, we try different approaches to compete with the baseline given in SemEval task 6 for Rhetorical role prediction. We compare our approaches and provide analysis for the same.

The Following rhetorical roles were given in the dataset (Kalamkar et al., 2022a):

Preamble: This includes the information about

the legal judgment record's metadata. A typical ruling would start with the court's name, the parties' finer details, the names of the lawyers and judges, and a headnote (rundown). This section would frequently conclude with a catchphrase like (JUDGMENT or Arrange). Some documents also have HEADNOTES and ACTS sections at the beginning which are included in the Preamble as well.

Facts (FAC): These are the relevant case facts. It describes the sequence of actions that resulted in the case being filed and how it progressed (for example, submitting a First Information Report (FIR) at a police station or making an appeal to the Magistrate). Depositions, current court processes, and summaries of earlier court actions.

Ruling by Lower Court (RLC): Cases are appealed from lower courts rather than being filed directly in higher courts. In light of the current appeal, the documents include decisions made by the lower courts (Trial Court and High Court) (to the Supreme Court or high court).

ISSUES: The main issues on which the verdict must be delivered are mentioned in some rulings. These legal issues that the court has framed are mostly in the form of questions.

Argument by Petitioner (ARG PETITIONER): This category includes precedent cases raised by petitioner attorneys, but when the court analyzes them later, they fall into either the relied-upon or not-relied-upon category.

Argument by Respondent (ARG RESPONDENT): Lawyers for the respondents' arguments. This refers to precedent cases that respondent attorneys argue, but when the court analyzes them later, they fall into one of two categories: relied upon or not.

Analysis (ANALYSIS): These are the court's viewpoints. Courts' discussions of the evidence, facts provided, earlier cases, and laws are included in this. Discussions on whether the law is or is not relevant to the situation at hand. It serves as the parent tag for the following three tags: PRE RELIED, PRE NOT RELIED, and STATUTE.

Statute (STA): Texts that the court uses to explain existing laws may include Acts, Sections, Articles, Rules, Orders, Notices, Notifications, and Quotations taken verbatim from the relevant Act. Both the tags Analysis and Statute will be included in the statute.

Precedent Relied (PRE RELIED): Texts in which the court analyzes earlier case materials, deliberations, and rulings that were cited in making final conclusions. Both the tags Analysis + Precedent will be present on Precedent.

Precedent Not Relied (PRE NOT RELIED): Texts in which the court considers earlier case materials, exchanges, and rulings that were not considered in its final findings. It may be because the circumstances of the previous case are unrelated to the current case.

Ratio of the decision (Ratio): Texts that the court debates. This contains the primary justification offered for applying any legal theory to the legal question. It is the outcome of the court's analysis. Usually, it shows up just before the choice is made. It differs from "Ratio Decidendi," which is taught in legal academic curricula.

Ruling by Present Court (RPC): Final decision, the conclusion along with the order of the Court resulting from the logical/natural conclusion of the reasoning.

None (NONE): A sentence is marked as NONE if it does not fit into any of the aforementioned categories.

2 Background work

Past works have centered on the creation of annotated corpora and the task of automatic rhetorical role labeling. (Venturi, 2012) developed a corpus, TEMIS of 504 sentences annotated both syntactically and semantically. The work of (Wyner et al., 2013) focuses on the process of annotation and conducting inter-annotator studies. (Savelka and Ashley, 2018) using Conditional Random Fields (CRF) with handcrafted features conducted Document segmentation of U.S. court documents to segment the documents into functional and issue-specific parts. (Saravanan et al., 2008) developed automatic labeling of rhetorical roles, where CRFs were used to label seven rhetorical roles. (Bhatia, 2014) created Genre Analysis of Legal Texts to create seven rhetorical categories. (Nejadgholi et al., 2017) developed a method for the identification of factual and nonfactual sentences using fastText. (Walker et al., 2019) compares different Machine Learning based approaches and rule-based scripts for rhetorical role identification. The baseline used is a closely related work by (Ghosh and Wyner, 2019), where they label rhetorical roles in Indian Supreme Court documents by using the BiLSTM-CRF model with sent2vec features.

3 Problem Statement

AI applications in the legal domain such as Judgment summarizing, judgment outcome prediction, precedent search, etc require the segmentation of long and unstructured legal documents into semantically coherent text segments. The text segmentation and classification problem are modeled as a single-sentence prediction task. Given a legal document D, containing the sentences $[s_1, s_2, ..., s_n]$, the task of rhetorical role prediction is to predict the label (or role) y_i for each sentence $s_i \in$ D.

4 Dataset

The following link contains the dataset used.

1. Training Data
2. Validation Data

The dataset contained a total of 245 legal documents in the training set and a total of 30 legal documents in the validation set. The test set of the data is not released yet. Individually there are a total of 28986 sentences in the documents for training and 2890 sentences for validation. These sentences along with their assigned rhetorical roles are present in the documents. The test dataset will be released in January 2023. So For our testing purpose, we used the validation dataset.

5 Technical Challenges

1. Unavailability of large language models trained on Indian Legal Documents and the unavailability of additional labeled Indian domain-specific legal documents.
2. The initial issue was to figure out how to batch sentences in the documents so that the contextual relationships between sequential sentences are maintained. The solution to this issue is described in Section 6.6.1.
3. Computational resources required to fine-tune the entire t5 model encoders along with additional layers for classification were huge and so we only trained the classification layers and set require_grad for the layers of t5 as false.

6 Approaches

The text segmentation and classification problem are modeled as a single-sentence prediction task. Multiple approaches have been tried and tested on the given dataset to classify the sentences as corresponding rhetorical roles. **The team members who worked together on the approaches are mentioned before the approach title. All team members contributed equally to the project.** The model which resulted in the best performance (discussed in the last few sections of approaches) considered contextual relationships between sequential sentences using T5 encoders combined with Bidirectional LSTM and Conditional Random Field.

6.1 Abhishek Maurya and Upasana Mishra: Naive Bayes Approach

Naive Bayes classifiers are a family of classifiers with each of them sharing a common principle of independence of all feature pairs being classified from each other. These algorithms work on the principle of Bayes' Theorem which can be mathematically stated as follows:

$$P(X|Y) = (P(Y|X) * P(X))/P(Y)$$

P(X|Y): the probability of event X given event Y is true. P(Y|X): the probability of event Y given event X is true. P(X), P(Y): independent probabilities of events X and Y respectively.

Here, X and Y are events with X appearing while Y has already happened. By following the above equation, we can make sure that the inference of one occupancy of an event has no influence or dependency on the other event. This is what is called a naive approach. The above equation calculates the probability of the occurrence of event X given the probability of event Y which has already occurred.

Taking into account the conditional independence, and assuming all features of X being mutually independent, conditional on the category y:

$$P(y|x_1, x_2, ..., x_d) = \frac{P(y)\prod_{i=1}^{d} P(x_i|y)}{P(x_1)P(x_2)....P(x_d)}$$

For a given input, the denominator will be constant, so we may remove that term.

$$P(y|x_1, x_2, ..., x_d) \propto P(y)\prod_{i=1}^{d} P(x_i|y)$$

Hence, the probability of the given sample will be the output with maximum probability:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes classifiers though not very complex, have widely been used for the purpose of text classification and analysis. Text classification is one of the major tasks in the field of Natural Language Processing. However, the datasets in this task involve text that is unstructured and hence it is critical to extract useful and meaningful information from such data or in other cases classify them under the correct class. Naive Bayes Algorithms can incorporate various methods like Multinomial, Gaussian, or Bernoulli. (McCallum and Nigam, 1998)

In order to feed the text data to our classifier, we need to format the data as per the acceptable input format of our classifier in use. Hence, the dataset is primarily divided into two parts called the feature matrix and the other called the target vector. The feature matrix, denoted by, 'X', contains all the vectors of the dataset. If, say, the number of total features is d then,

$$X = (x_1, x_2, x_3, ..., x_d)$$

The target vector is also called the response vector and is denoted by 'y'. It is named so that it represents the target class (response) for each of the vectors in X. As stated earlier, The Naive Bayes algorithm assumes that each of the features of the same class has an independent and equal influence on the outcome. However, it is to be noted that, in real-world scenarios, these assumptions might not hold true, and hence this approach is 'naive'.

Now, the text dataset we have is raw data which is nothing more than a sequence of strings and this cannot be directly fed to our model. The model accepts data in the form of numerical feature vectors having fixed sizes. To help with this, scikit-learn library has been used to provide for the utilities to extract the numerical features from the text. Two tasks namely - firstly, tokenizing the strings and then assigning each possible token with an integer id is done, and secondly, counting the occurrences of these generated tokens in each document. Hence, we now say that an individual token occurrence frequency will become one feature and the vector of all these token frequencies for one document will become a multivariate sample.

6.2 Upasana Mishra and Hemal Mamtora: Doc2Vec Approach

For the task of text classification, the data that we deal with is text data which can be in the form of strings, words, sentences, or even documents. One of the strong tools of NLP is Doc2Vec which involves the technique of representing the documents in the form of a vector and is considered a generalization of the word2vec method.

Since the technique doc2vec gets its base from the technique of word2vec, it is important to take note of a few things about word2vec. Wrod2vec is a method of producing the numeric representations of each word in the text in a way that these representations will also be able to capture the relationships among themselves as well. In this way, we can encapsulate various relations between words like synonyms, antonyms, analogies, etc. The word2vec representations technique incorporates two algorithms namely: Continuous Bag-of-Words (CBOW) and Skip-Gram model. The CBOW model forms a sliding window (hence continuous) around the current word and predicts it from the "context" - the surrounding words. Each word here is a feature vector which then becomes a word vector after training. The vectors that represent similar words, as stated before, are close to each other in distance. They are also additionally encapsulating numeric relations. Now, the next algorithm - Skip Gram uses one word to predict all surrounding words ("context") unlike the CBOW which uses one word each time. Skip gram is slower than CBOW but more accurate with infrequent words. (Le and Mikolov, 2014)

In doc2vec, we deal with documents instead of words. Documents, unlike words, do not possess logical structures. Hence, we add another vector called paragraph ID. So, as in the CBOW algorithm for word2vec, now, instead of using only words to predict the next word, we consider another feature vector, i.e., document-unique. So, in the process of training the word vectors W, the document vector D is also trained, and at the end of this training, it holds a numeric representation of the document. This algorithm is called the Distributed Memory version of the Paragraph Vector (PV-DM). Hence, the document vector intends to represent the concept of the document.

The other algorithm of doc2vec which is like the Skip-Gram model is the Distributed Bag of Words version of Paragraph Vector (PV-DBOW). This is the technique that has been used in our code. This algorithm is faster and consumes less memory as there will be no need to save the word vectors in this algorithm. A neural network is trained to ob-

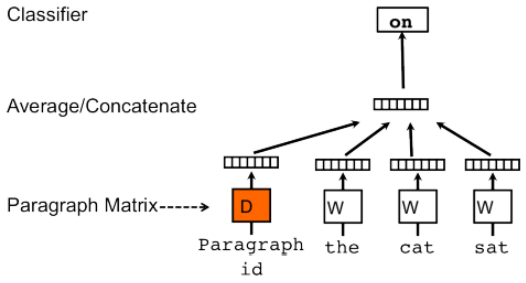

Figure 1: PV-DM Model of Doc2Vec

tain these paragraph vectors. The training is done on the task of prediction of the probability distribution of words in a paragraph given a word from the paragraph that is randomly sampled. Gensim library has been used to implement the doc2vec. We implement the DBOW model of doc2vec. For the implementation of the task, a Logistic regression classifier has been used along with the Doc2Vec.

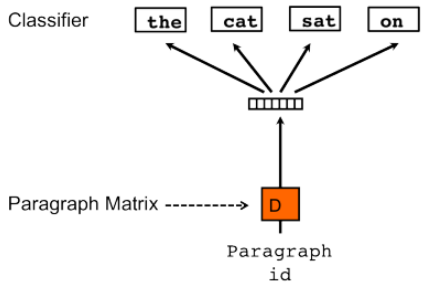

Figure 2: PV-DBOW Model of Doc2Vec

6.3 Shubham Gupta and Abhishek Maurya: Wiki-bm25-lexmodel Approach

Search engines employ the ranking function in the domain of information retrieval, known as Okapi BM25, or best matching (BM) abbreviated, to determine the relevancy of content to a particular search query. It is based on the probabilistic retrieval paradigm created by Stephen E. Robertson, Karen Spärck Jones, and others in the 1970s and 1980s (Robertson et al., 2009).

As legal documents are concerned with the legal phrases used, these phrases are the keywords that act as deciding factors for a sentence being assigned to a particular class for classification. In TF-IDF (term frequency-inverse document frequency) method, TF is a measure of how often a phrase appears in a document, and IDF is about how important that phrase is. Hence, we decided to try out the improved TF-IDF method, with refinements of TF and IDF components using the BM25 function (Larson, 2010). Tokenizer and query encoder of bm25 is used to generate the embeddings of the legal text (Lee and Lee, 2010). Sentence-wise generation of embedding of the text is targeted with the mentioned tokenizer and encoder.

6.3.1 Wiki-bm25 + Multinomial Logistic Regression (MLR)

In the legal text, every sentence is used for the generation of word embeddings. Using "facebook/sparwiki-bm25-lexmodel-context-encoder", a tokenizer is used for truncated token generation and then the output is fed to the wiki-bm25 query encoder to generate the word embeddings.

As the word embeddings are in the form of a vector, the problem statement boils down to performing a multiclass classification on a vector. Multinomial logistic regression is used to predict categorical placement in or the probability of category membership on a dependent variable based on multiple independent variables. Multinomial logistic regression is a simple extension of binary logistic regression that allows for more than two categories of the dependent or outcome variable. Like binary logistic regression, multinomial logistic regression uses maximum likelihood estimation to evaluate the probability of categorical membership. Therefore, multinomial logistic regression is chosen to perform the classification on word embeddings after training and fitting the model with training data of text embeddings (Shah et al., 2020). Once we have trained the MLR model, then in the test set, each sentence is encoded, and using the trained classifier model, the rhetorical role is predicted. Hence with this approach, each sentence of a legal text is labeled as rhetorical roles as a result of classification.

Model training: Max iteration of 100 is chosen to perform the fitting of the model. L2 regularization is used for normalizing the weights and cost function. The “SAG” solver is used to perform stochastic gradient descent as an optimizer of the cost function.

6.4 Hemal Mamtora and Rohan Chaudhury: GloVe Embedding Approach

GloVe stands for Global Vectors for word representation. The Stanford University academics that created it wanted to create word embeddings by combining global word co-occurrence matrices from a specific corpus. (Pennington et al., 2014)

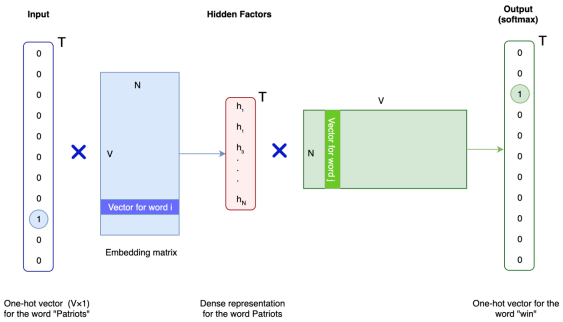

Figure 3: Glove Embedding (Pennington et al., 2014)

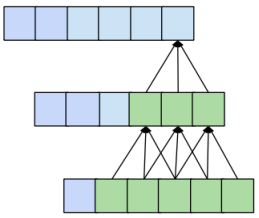

Figure 4: 1D Convolution

The GloVe word embedding's fundamental premise is to infer the link between words using statistical data. The co-occurrence matrix, as opposed to the occurrence matrix, reveals the frequency of a certain word pair occurring when it is combined with another. An instance of two words occurring together is represented by each value in the co-occurrence matrix.

Rationale, sometimes simpler models provide better results. The multilayer feedforward network cannot capture long-range dependencies, but to check how much accuracy is gained when using a bigger, better model, this approach was tried.

Glove Embedding requires very little preprocessing and fine-tuning. It does not take into consideration the context of the text. So this could be inferior as compared to the transformer-based approaches. But this is a lightweight approach, hence, to find out, how much f1 gain occurs when trying a simpler model as compared to a heavyweight model, this approach was considered. Preprocessing and fine-tuning for glove embedding was extremely minimal. Consequently, this might not be as effective as transformer-based methods. To determine how much f1 gain occurs when attempting a simpler model as opposed to a heavy-weight model, this approach was tried.

The approach: It involved preprocessing, model training, and hyperparameter tuning. With exploratory data analysis, we found the distribution of labels, and where exactly the model is failing.

6.4.1 Glove Embedding + 1D Convolution

1D convolution layer (e.g. temporal convolution). This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. (Kiranyaz et al., 2021)

The next step in the approach was trying some method that could find repetitive patterns. This was because the law cases would have recurring themes for different categories of labels. So 1D convolution was tried. It is a sliding window approach.

First, the dataset was loaded which contained sentences and paragraphs of legal documents. After loading the dataset, it was converted from JSON to pandas dataframe. Using the preprocessing and analysis techniques in natural language processing, the text was converted to sequences. For feeding it to the glove embedding, the longest sequence length was found. For inference, the input would be padded with empty tokens. Using a tokenizer, the input was converted to a set of tokens. Next, the data is fed to glove embeddings. There are 2 approaches to the direct feed-forward network with cross-entropy loss. Both approaches use a softmax activation for multiclass classification for each sentence. The 2nd approach uses a 1D convolution layer to capture longer-term dependencies and repeated patterns in sentences.

The results were also tabulated in the result section. The additional insight is given here. It performs poorly as compared to the baseline. A macrof1 score of 0.26 and 0.31 are indicators of poor capture of sequence information. Hence, we moved on to using contextual embeddings for our next approaches.

The model got confused mostly between the preamble and statute. The model was finding difficulty primarily in understanding the statute since it was similar to other legal sections of the paper. The prologue and law caused the most confusion for the model. The statute was particularly difficult for the model to comprehend because it was comparable to other legal sections of the text.

6.5 Rohan Chaudhury and Shubham Gupta: T5 introduction and usage based approaches

T5 or Text-to-Text Transfer Transformer (Raffel et al., 2019), is a Transformer-based architecture that employs a text-to-text method. The model text

is fed as input into each task, such as translation, question answering, and classification, and trained to produce some target text. This enables us to reuse the same model, loss function, hyperparameters, and so on across our wide range of workloads. Changes of the T5 model from BERT include:

1. the addition of a causal decoder to the bidirectional design.

2. substituting a variety of other pre-training activities for the fill-in-the-blank cloze activity.

For our application, we have used only encoders of T5 models of several sizes including T5-base, T5-large, T5-xl, and T5-xxl. We tried several approaches using the T5 encoders some of whom are discussed below:

6.5.1 T5-Encoder+Probabilistic Linear Discriminant Analysis(PLDA)

PLDA (Ioffe, 2006) is a generative model in which we assume that the data samples of a class are generated by a Gaussian distribution. The Gaussian mean represents the class variable and is derived from a different Gaussian distribution known as the prior. By comparing the likelihood of instances from one class vs. instances from other classes, we may compare the unseen samples for the task of recognition using PLDA scores. Using PLDA scores between all pairs of instances from the whole collection of examples, we can cluster examples into classes.

As shown in Fig 5, the sentences in the documents are passed through a T5-Encoder (t5-xxl) to get the sentence representations which along with the training labels are then used to train a PLDA classifier. This classifier then uses the PLDA scores to classify any unseen samples to any one of the 13 Rhetorical Role labels. The model results are shown in the table in the "Results" section.

6.5.2 T5-Encoder+FAISS (Facebook AI Similarity Search)

FAISS (Johnson et al., 2017) is a library that enables users to efficiently search for comparable embeddings of multimedia documents. It overcomes the limits of typical query search engines that are specialized for hash-based searches and delivers more scalable similarity search functionalities.

As shown in Fig 6, the sentences in the training documents are passed through a T5-Encoder (t5-xxl) to get the sentence representations which are then used to learn the FAISS indices. When a sample from the validation set is introduced, the same

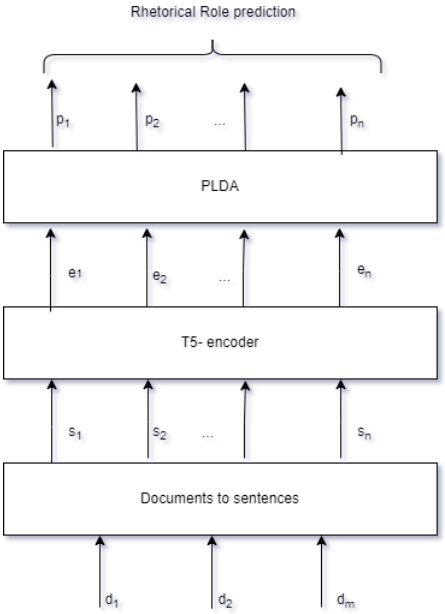

Figure 5: T5-encoder with PLDA

T5 Encoder is used to get the sentence embeddings and the FAISS indices are used to search for the most similar training sample. The label of the most similar training sample is then provided as the label of the unseen sample.

For the above 2 approaches, we have also experimented with the Sentence Transformer - 'paraphrase-xlm-r-multilingual-v1' to get the sentence representations instead of T5-Encoder in these model architectures and captured the corresponding results which are presented in the table in the "Results" section.

In the previous 2 approaches, the model got confused primarily between the following labels:

1. "Argument by Petitioner" and "Argument by Respondents"

2. "Precedent Not Relied", "Precedent Relied" and "Analysis" labels

This was primarily because the labels were almost similar to each other in terms of sentence representations and to distinguish between them we needed to consider the contextual relationships between sets of sequential sentences.

6.5.3 T5-encoder with Bi-Directional LSTM Approach

Based on the analysis done on the training data, it was observed that the rhetorical roles of sentences do not change abruptly. So, based on this observation using a Bi-Directional LSTM model made

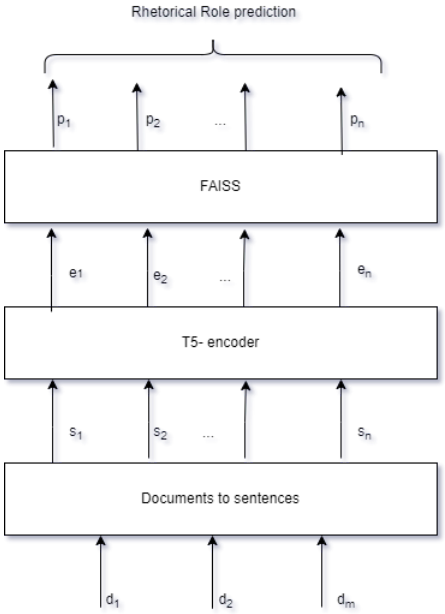

Figure 6: T5-encoder with FAISS

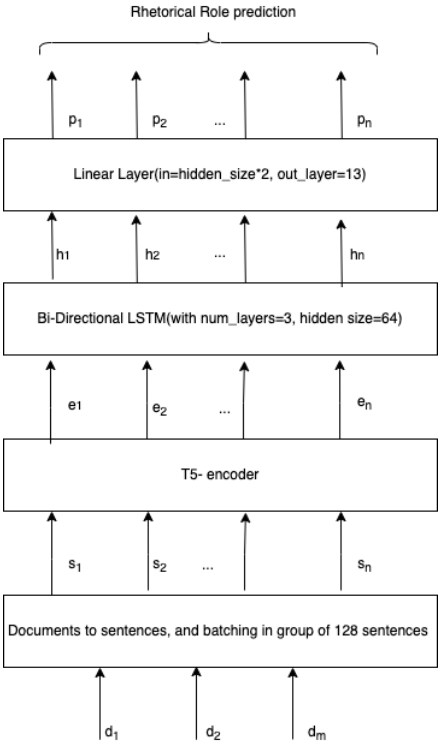

Figure 7: T5-encoder with Bi-Directional LSTM

sense as it will be able to pass information to neighboring sentences to predict labels.

Bi-Directional LSTM is a neural network-based model which passes the information in a sequence in both forward and backward directions. These were introduced to remove the problem faced by traditional RNN models in terms of exploding and vanishing gradient problems leading to poor transfer of information from the first sentence to the later sequence(Melamud et al., 2016).

So, for this approach as seen in figure 7, all the documents were first divided into sentences. These sentences were batched with batch_size=128. Now, we want to convert each of the sentences into embeddings. The current state-of-the-art sentence encoder(T5-base) encoder was used to get the embedding of each sentence. Once we obtain the sentence embedding it is passed to a Bi-Directional LSTM followed by a linear layer which is used to predict the label of the Rhetorical role.

6.6 The approaches contributed equally by all members: Models considering the contextual relationship between neighboring sentences

We so far have tried models which use some form of sentence embeddings, but the issue with the approach previously discussed was that they didn't focus on the label predicted for the previous sentence. So, for this approach, we used an additional Conditional Random Field layer to model the relationship between sentences.

We have tried various tokenizers, encoders, and hyperparameters for our final approach. The following section is explained using the T5-base tokenizer, and T5-base encoder.

6.6.1 Data Pre-Processing

The data pre-processing technique is similar to that used by (Kalamkar et al., 2022b) First, each sentence from the training JSON file and the validation JSON file is passed to the T5-base tokenizer which converts each word of each sentence into tokens. From this step, the output is a document number followed by the label of each sentence and the word tokens for that sentence. Depending on the max_seq_length parameter the maximum number of tokens for each sentence is constrained. Also, based on the max_seq_length appropriate number of pad tokens are added to each sentence.

Now we obtain the tokenized data to create batches for the finetuning of the model. The logic we used here was that a single document should always be in a single batch and not split into two documents. So, to do this efficiently we make use of the number of sentences in each sentence to split various documents into batches. We follow the following steps for each document (consider the batch size of the document is batch_size):

1. Each document is converted into a record that stores two things: the data of the document and the number of sentences in the document.

2. Now, we create buckets based on the number of sentences in each document. If the (number of sentences)*(number of records + 1) > batch_size, then we add a new bucket for the number of sentences and add the current record to the new bucket. If (number of sentences)*(number of records + 1) <= batch_size means that the current bucket can hold an additional record, we add the new record to the existing bucket.

3. Now, that we have processed all the documents and added them to buckets, now we club buckets that are empty. To do this we start with the buckets which have the largest number of sentences and add other records from the smaller buckets till the (max_length_bucket)*(number of records+1) <=batch_size.

4. The final buckets from the above step form our batches and are passed to the model for predictions.

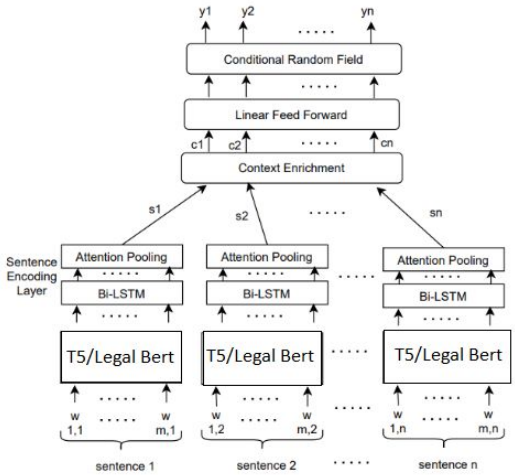

Figure 8: T5-encoder with Bi-Directional LSTM and CRF (Kalamkar et al., 2022b)

6.6.2 Model Architecture

The pre-processed data is obtained in form of batches. As shown in the figure 8 above:

1. The batches are passed through an encoder layer to get the word embeddings. The embedding size of the output from this layer depends on the different models that are used in the experiments. The models used in this layer are:

   (a) T5-small - output embedding size: 512

   (b) T-base - output embedding size: 768

   (c) T-large - output embedding size: 1024

   (d) LegalBERT - output embedding size: 768

2. The obtained embeddings are passed through a Bi-LSTM layer to capture contexts from both sides of the words. Different numbers of layers and hidden layer sizes are used in our experiments.

3. The outputs from the Bi-LSTM layers are passed to the Attention Pooling layers to get the sentence representations.

4. The sentence representations are passed through a Context Enrichment Layer that encodes contextual information by taking a sequence of sentence representations and converting them into contextualized sentence representations.

5. This is followed by MLP (Multi layer perception) layers and CRF (Conditional Random Field) that leverage the distributed representation features to predict the Rhetorical Role label for each sentence via softmax activation

   This model architecture is fine-tuned using the training data provided and tested on the validation data provided (test data is not yet released).

7 Results

Macro F1 and Precision scores achieved on validation data for different models are compiled as a result in Table 1.

Model	Macro F1	Precision
Naive Bayes	0.3202	0.5217
Doc2Vec	0.3567	0.4607
Wiki BM25 Lexical Model + PLDA	0.3391	0.3882
Wiki BM25 Lexical Model + LR	0.3829	0.5159
Glove Embedding + FCN	0.2694	0.3142
Glove Embedding + 1DCNN	0.3136	0.3562
T5 Encoder + (FAISS)	0.3524	0.4806
T5 Encoder + (PLDA)	0.4319	0.5
T5-base + LSTM	0.3923	0.4703
Sentence Transformer (paraphrase-xlm-r-multilingual-v1)+ PLDA	0.3931	0.4640
(paraphrase-xlm-r-multilingual-v1)+ FAISS	0.3329	0.6103
Baseline modified with T5-base encoder + BiLSTM with 1024 hidden layer	0.6103	0.8127
T5-base encoder + BiLSTM with three hidden layer	0.5796	0.7964
Baseline modified with Legal-Bert + 1DCNN	0.5873	0.8089
Baseline modified with Legal-Bert Large 1.7m + 1DCNN	0.5777	0.7919
Baseline	0.5962	0.8082
Our Approach (Best) - Baseline modified with T5-large tokenizer and encoder	**0.6161**	**0.8166**

Table 1: Results on Validation data

7.1 Best Model Parameters

"model": "t5-large",
"model_trainable": false,
"dropout": 0.5,
"lr": 3e-05,
"batch_size": 32,
"max_seq_length": 128,
"max_epochs": 20,
"early_stopping": 5
"num_layers_lstm": 1
"num_hidden_units_lstm": 758

Further analysis of our best model and confusion matrix for various approaches is added in the Appendix section.

8 Analysis

8.1 Statistical models

Traditional methods like Naive Bayes and embedding-based approaches like Glove, and Doc2Vec performed decently. This is because they could not discern the relationships between terms in the documents.

As multinomial logistic regression is a generalized linear model, it cannot fit non-linear data points. The embedding vector data can be seen as a distribution of non-linear points hence, it does not draw a perfect line as a separation between the data points.

8.2 Information Retrieval Based model

Wiki-bm25 tried to find the relevant terms to classify the sentences, but it was not able to embed the contextual information

8.3 Temporal Models

Next trials with 1D-Conv and LSTM captured long-term dependencies leading to better performance. There was an improvement in the f1 score since now the model considers long-term dependency, but the improvement was not significant enough.

The model got confused most between the preamble and statute. The model was finding difficulty primarily in understanding the statute since it was similar to other legal sections of the paper.

8.4 Neural Language Models without contextual relationship without sentences

In our experiments with T5-encoder with PLDA, FAISS, and LSTM we shuffled the sentences before passing them to the T5-encoder. This caused the model to not learn the relationship between

the current and the next label. This can be seen from the performance difference in approach in this section(Macro F1 0.43) vs the next section where context relationship is used(Macro F1 0.59). This shows that the observation that legal roles do not change abruptly is correct.

8.5 Neural Language Models with Context relationship between sentences

The T5-large encoder with Bi-Directional LSTM and CRF resulted in the best performance.

We also observed that Legal-BERT/Legal-BERT-1.7m did not perform better than the T5-large encoder which has a similar number of parameters and is fine-tuned on legal documents. This can be attributed to two facts. First, LEGAL-BERT is trained on EU documents and not trained on Indian law documents. Second, the T5 encoder is able to encode the sentences better than the traditional Bert Model on which Legal-Bert is based.

9 Limitations

From each approach:

- BM25 is based on the bag-of-words model, therefore it does not capture the position in the text, semantics, or co-occurrences in different documents. It does not capture word semantics in the text hence only useful as a lexical-level feature.
- GLOVE embedding does not contain contextual information which could hurt performance.
- 1D Conv and FCN do not capture long-range dependencies.
- In our best approach, the labels with per-label-precision and per-label-F1 scores for Argument by Petitioner and Argument by Respondents are low because they are very similar to each other.
- Also, the model was not able to identify "Precedent Not Relied" class and classified them as Precedent Relied and Analysis. This can be attributed to the incorrect identification of boundaries because they are almost similar to one another.

10 Contribution

All team members contributed equally to the development and delivery of the project. We initially split the statistical, information retrieval, and deep neural network approaches among ourselves. As the computation required for large language models (including t5/BERT configuration) was significant we individually trained different configurations using various approaches and reported the results above. The report and presentation were also contributed to equally by all the members of our team.

References

Vijay Kumar Bhatia. 2014. *Analysing genre: Language use in professional settings*. Routledge.

Saptarshi Ghosh and Adam Wyner. 2019. Identification of rhetorical roles of sentences in indian legal judgments. In *Legal Knowledge and Information Systems: JURIX 2019: The Thirty-second Annual Conference*, volume 322, page 3. IOS Press.

Sergey Ioffe. 2006. Probabilistic linear discriminant analysis. In *Computer Vision – ECCV 2006*, pages 531–542, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Prathamesh Kalamkar, Aman Tiwari, Astha Agarwal, Saurabh Karn, Smita Gupta, Vivek Raghavan, and Ashutosh Modi. 2022a. Corpus for automatic structuring of legal documents. *arXiv preprint arXiv:2201.13125*.

Prathamesh Kalamkar, Aman Tiwari, Astha Agarwal, Saurabh Karn, Smita Gupta, Vivek Raghavan, and Ashutosh Modi. 2022b. Corpus for automatic structuring of legal documents. *CoRR*, abs/2201.13125.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 2021. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398.

Ray R Larson. 2010. Logistic regression for ir4qa. In *NTCIR*, pages 126–129.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.

Yong-Hun Lee and Sang-Bum Lee. 2010. A research on enhancement of text categorization performance by using okapi bm25 word weight method. *Journal of the Korea Academia-Industrial cooperation Society*, 11(12):5089–5096.

Vijit Malik, Rishabh Sanjay, Shouvik Kumar Guha, Shubham Kumar Nigam, Angshuman Hazarika, Arnab Bhattacharya, and Ashutosh Modi. 2021. Semantic segmentation of legal documents via rhetorical roles. *arXiv preprint arXiv:2112.01836*.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, page 41–48. AAAI.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Isar Nejadgholi, Renaud Bougueng, and Samuel Witherspoon. 2017. A semi-supervised training method for semantic search of legal facts in canadian immigration cases. In *JURIX*, pages 125–134.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

M Saravanan, Balaraman Ravindran, and S Raman. 2008. Automatic identification of rhetorical roles using conditional random fields for legal document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

Jaromir Savelka and Kevin D Ashley. 2018. Segmenting us court decisions into functional and issue specific parts. In *JURIX*, pages 111–120.

Kanish Shah, Henil Patel, Devanshi Sanghvi, and Manan Shah. 2020. A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research*, 5(1):1–16.

Giulia Venturi. 2012. Design and development of temis: a syntactically and semantically annotated corpus of italian legislative texts. In *proceedings of the workshop on semantic processing of legal texts (SPLeT 2012)*, pages 1–12.

Vern R Walker, Krishnan Pillaipakkamnatt, Alexandra M Davidson, Marysa Linares, and Domenick J Pesce. 2019. Automatic classification of rhetorical roles for sentences: Comparing rule-based scripts with machine learning. In *ASAIL@ ICAIL*.

Adam Z Wyner, Wim Peters, and Daniel Katz. 2013. A case study on legal case annotation. In *JURIX*, pages 165–174.

A Appendix

A.1 Confusion Matrix

A.1.1 Stastical Approach

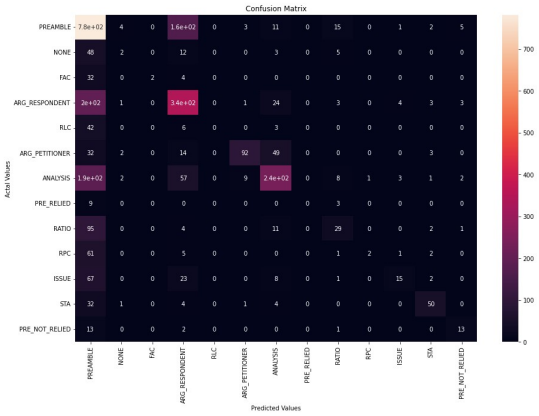

Figure 9: Confusion Matrix: Naive Bayes

A.1.2 Information Retrieval Approach

Figure 10: Confusion Matrix: wiki-bm25-lexmodel with Logistic Regression classifier

A.1.3 Deep Learning Approach

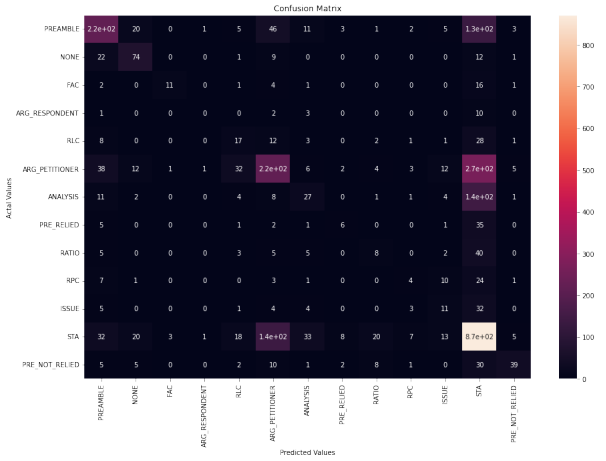

Figure 11: Confusion Matrix: Deep Learning - Glove Embedding + 1D Conv

A.1.4 T5-base Encoder + Classifier Approach

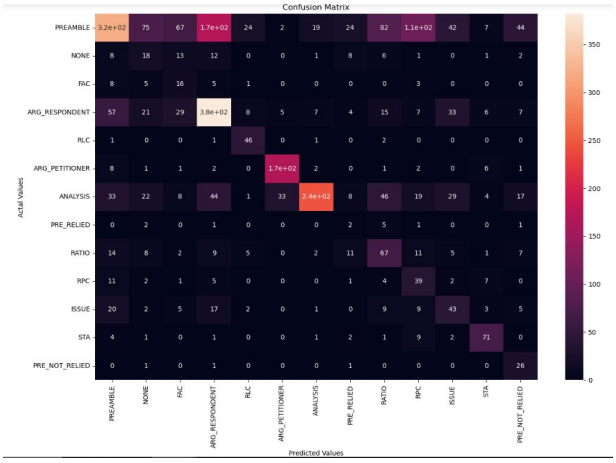

Figure 12: Confusion Matrix: sentence-t5-xxl with PLDA classifier

A.1.5 Best Approach

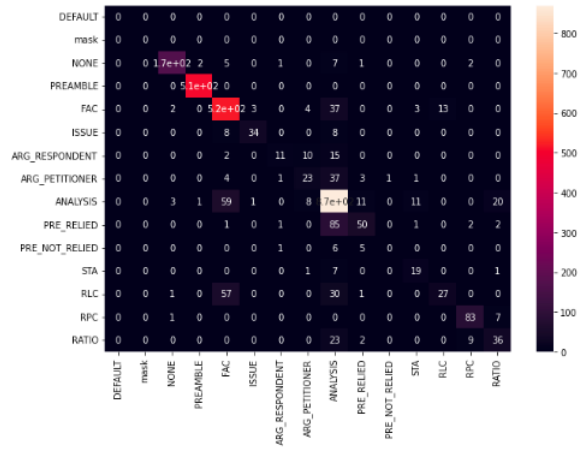

Figure 13: Confusion Matrix: T5-large tokenizer with modified baseline

The following are the results for our best model: The same results are also included in the code submission along with training parameters

Label	Per-Label-F1	Per-Label-Precision
NONE	0.9322493225	0.9608938547
PREAMBLE	0.9970559372	0.9941291585
FAC	0.8395461912	0.7920489297
ISSUE	0.7727272727	0.8947368421
ARG_RESPONDENT	0.4150943396	0.7333333333
ARG PETITIONER	0.3965517241	0.5
ANALYSIS	0.8250355619	0.7733333333
PRE_RELIED	0.4651162791	0.6849315068
PRE NOT RELIED	0	0
STA	0.6031746032	0.5428571429
RLC	0.3461538462	0.675
RPC	0.8877005348	0.8645833333
RATIO	0.5294117647	0.5454545455

We also observed that the performance improvement were not significant after 15 epochs.

Epochs	Test macro F1
0	0.3057565813
3	0.4360708977
5	0.5216789151
6	0.5731321915
9	0.5799649748
11	0.60534519
13	0.5988133323
15	0.608058479
18	0.6064139999
19	0.6061397982