IRIT_IRIS_A at SemEval-2023 Task 6: Legal Rhetorical Role Labeling Supported by Dynamic-Filled Contextualized Sentence Chunks

Alexandre G. de Lima[1,2,3], Jose G. Moreno[3], Eduardo H. da S. Aranha[2]
[1]Instituto Federal do Rio Grande do Norte, Natal, Brazil
[2]Universidade Federal do Rio Grande do Norte, Natal, Brazil
[3]Institut de Recherche en Informatique de Toulouse, UMR 5505 CNRS, Toulouse, France
alexandre.lima@ifrn.edu.br, jose.moreno@irit.fr,
eduardoaranha@dimap.ufrn.br

Abstract

This work presents and evaluates an approach to efficiently leverage the context exploitation ability of pre-trained Transformer models as a way of boosting the performance of models tackling the Legal Rhetorical Role Labeling task. The core idea is to feed the model with sentence chunks that are assembled in a way that avoids the insertion of padding tokens and the truncation of sentences and, hence, obtain better sentence embeddings. The achieved results show that our proposal is efficient, despite its simplicity, since models based on it overcome strong baselines by 3.76% in the worst case and by 8.71% in the best case.

1 Introduction

SemEval-2023 Task 6 (LegalEval) focuses on the understanding of texts from the legal domain and it comprises three sub-tasks: Legal Rhetorical Role Labeling, Legal Named Entity Recognition, and Court Judgment Prediction with Explanation (Modi et al., 2023). The Legal Rhetorical Role Labeling (Legal RRL) task aims to assign semantic roles to the sentences from a legal document. Examples of such roles are *facts*, *arguments*, and *rulings*. The exact roles are defined by the documents' nature and by the following utilization of the labeled sentences. Legal documents are usually long and complex, so the task is relevant because the labeled sentences assist the document interpretation and the extraction of information. The labeled sentences are useful for downstream tasks, such as summarization and judgment prediction. The task is not trivial even for professionals from the legal domain, mainly because of its inherent subjectivity. The Legal RRL sub-task dataset comprises judgment documents written in English from the Indian legal system. The documents' sentences were manually labeled by students from Indian law universities, whose each sentence was assigned with one of 13 pre-defined rhetorical roles (Modi et al., 2023).

The Legal RRL can be tackled as a sentence classification task, and this approach requires the utilization of an encoder to convert text sentences into numeric vectors. Recent Legal RRL works exploit pre-trained Transformer models to encode sentences. However, they do not fully leverage the context exploitation ability of Transformer models: sentences are usually encoded in isolation and the inter-sentence context is lost (Savelka et al., 2020; Aragy et al., 2021), the inter-sentence context is yielded by a Recurrent Neural Network (RNN) (Li et al., 2021; Bhattacharya et al., 2021) or by using data augmentation (de Lima et al., 2022), or only cover *facts* instead of the full range of rhetorical roles (de Lima et al., 2023).

This work presents the approach exploited by the IRIT_IRIS team to tackle the SemEval-2023 Task 6, sub-task A. We propose a novel strategy to encode sentences that fully leverages the context exploitation ability of Transformer and hence yields richer sentence embeddings. Instead of encoding sentences in isolation, we exploit sentence chunks that avoid the truncation of core sentences and the utilization of padding tokens. Achieved results lead us to deem that our proposal is efficient as the respective models overcome strong baselines by 3.76% in the worst case and by 8.71% in the best case. A secondary finding is that models based on pre-trained RoBERTa and Longformer models benefit more from the proposed approach than BERT-based models[1].

2 Background

To leverage the context exploitation ability of Transformers, we just need to feed the encoder with a sequence of sentences separated by a special token such as the `[SEP]` one from BERT (Devlin et al., 2019). Because most Transformer models

[1]The code of our models is available at https://github.com/alexlimatds/SemEval_2023_Task_6A

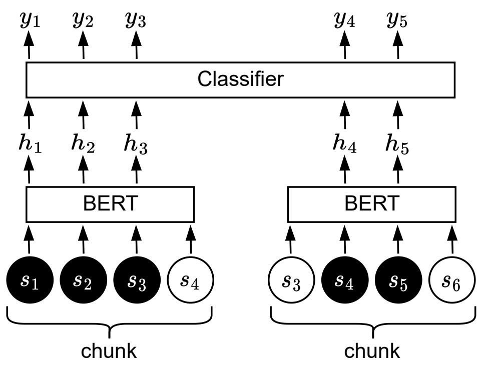


Figure 1: Model based on sentence chunks with overlapped edge sentences. Black and white circles respectively represent core sentences and edge sentences. This example illustrates the processing of two sentence chunks sharing only one sentence at each edge. Remark that only core sentences are forwarded to the classifier.

have a limit regarding the sequence length (e.g., 512 for BERT) and legal documents are usually long, feeding the encoder with a whole document is not computationally feasible. A workaround is breaking the documents into sentence chunks and feeding the encoder with such chunks as proposed by Cohan et al. (2019). A downside of this approach is that inter-sentence context is restricted to chunks (sentences in a chunk are not aware of sentences in other chunks). Mullenbach et al. (2021) and van den Berg and Markert (2020) rely on overlapped chunks to overcome this: consecutive chunks share some of their edge sentences as illustrated in Figure 1.

Only two works about Legal RRL leverage sentence chunks and Transformers to encode sentences. In Malik et al. (2022) a chunk comprises up to three sentences: the core sentence (i.e., the sentence to be classified) and its left and right neighboring sentences (remark that the first sentence does not have a left neighbor and the last sentence does not have a right neighbor). Nevertheless, this model performs worse than one that encodes sentences in isolation. Kalamkar et al. (2022) exploit chunks without sharing sentences as proposed by Cohan et al. (2019) but the model performs significantly inferior to one that leverages inter-sentence context through RNNs.

3 Dynamic-Filled Contextualized Sentence Chunks

The approaches from Cohan et al. (2019), Mullenbach et al. (2021), and van den Berg and Markert (2020) differ in the number of shared sentences, number of core sentences, and how the sentences are distributed among chunks.

In Cohan et al. (2019) there are no shared sentences and the number of chunks is dynamically set from the number of sentences in the document and from a hyperparameter delimiting the maximum number of sentences in a chunk. The chunks are filled so that they contain almost equal amounts of sentences. To assure that, sentences can be truncated. To establish chunks with the same number of tokens, which allows batch processing, the chunks are often filled with padding tokens.

Mullenbach et al. (2021) and van den Berg and Markert (2020) propose chunks with a fixed number of core sentences and edge sentences that must be set in advance. The drawback of this approach is that it yields chunks with a small number of content tokens when the concerned sentences are short. A large number of core sentences could avoid this, but it may lead to many truncated core sentences.

Chunks with many padding tokens are context-poor and truncation of core sentences results in data loss. Such facts may result in poor sentence embeddings that harm the performance of classification models.

Algorithm 1 Assembly logic of DFCSCs

```
Require: ${s_1, s_2, ..., s_n}$ : sentences in a document
Require: $m$ : minimum desired number of edge tokens
Require: $c\_len$ : number of tokens in a chunk
 1: $chunkList \leftarrow$ new List
 2: $chunk \leftarrow$ new chunk
 3: $i \leftarrow 1$
 4: while $i \leq n$ do
 5:     $t_s \leftarrow$ number of tokens in $s_i$
 6:     $t_c \leftarrow$ number of tokens in $chunk$
 7:     if $t_c = 0$ then
 8:         Put $chunk$ into $chunkList$
 9:         Put $s_i$ into $chunk$
10:         if $c\_len - t_s < m$ then
11:             Fill the edges of $chunk$ until reach $c\_len$
12:             $chunk \leftarrow$ new chunk
13:         $i \leftarrow i + 1$
14:     else
15:         if $c\_len - t_s - t_c \geq m$ then
16:             Put $s_i$ into chunk
17:             $i \leftarrow i + 1$
18:         else
19:             Fill the edges of $chunk$ until reach $c\_len$
20:             $chunk \leftarrow$ new chunk
21: return $chunkList$
```

To overcome the limitations of the three cited approaches, we propose *Dynamic-Filled Contextualized Sentence Chunks* (DFCSC). These are chunks with a variable number of core and edge sentences, without truncated core sentences, and that usually do not require padding tokens. These characteristics lead to better leverage of the context exploitation ability of Transformer models. The procedure of filling a DFCSC has two hyperparameters, the number of tokens in a chunk c_len, and the minimum desired number of edge (shared) tokens, m. The goal is to get a chunk with the maximum number of consecutive sentences while maintaining a desired minimum number of shared tokens among successive chunks. A minimum number of edge tokens assures that inter-sentence context is not restricted to a chunk and the more content tokens in a chunk, the better the representations of core sentences. Working oriented to tokens in the chunk edges, instead of sentences, avoids the insertion of padding tokens. To fully leverage the context exploitation ability of a PTLM, c_len must be set equal to the maximum sequence length supported by the model (e.g., 512 for BERT).

Algorithm 1 presents the assembly logic of DFCSCs. A sentence s is put into a chunk if it does not harm m (lines 15 to 16). Harming m means that the remaining capacity of the chunk after inserting s is lesser than m. Otherwise, the remaining capacity in the chunk is filled with edge tokens extracted from the neighboring sentences (lines 18 to 19). Then, an empty chunk is created (line 20) and s is put into it (lines 7 to 9). When the chunk is empty, s is always inserted into the chunk (lines 7 to 9). This is the unique case in which harming m is allowed, and the chunk is filled with edge tokens when it happens (lines 10 to 11). Remark that a core sentence is truncated only when its number of tokens is larger than the chunk supports (c_len).

Algorithm 1 abstracts some details such as the special tokens. In practice, a chunk starts with a special token to mark its beginning (e.g., `[CLS]`), and sentences and context edges are separated by another special token (e.g., `[SEP]`). The examples in Figure 2 present the resulting DFCSCs, with special tokens, assembled from a set of five sentences. The first chunk has s_1 and s_2 as core sentences. Because there are no sentences at left, only the right edge is filled with tokens from neighboring sentences (s_3 and s_4). The second chunk has only s_3 as core sentence. This is because inserting s_4 would harm m. The edges are filled with tokens from s_1 and s_2 at left, and from s_4 at right. The third chunk has only s_4 as core sentence and it harms m. This happens because the length of s_4 is greater than c_len, but since s_4 is the first sentence in the chunk, the insertion is allowed. The fourth chunk contains only s_5 as core sentence since this sentence is the remaining one. Because there are no sentences at right, only left edge is filled with tokens from the neighboring sentence (s_4).

4 Experimental Setup

4.1 Dataset

The available dataset (Kalamkar et al., 2022) comprises 277 judgment documents written in English from the Indian legal system and it is split into two sets, train (247 documents and 28,986 sentences) and validation (30 documents and 2,879 sentences). The documents' sentences were labeled by law students and each sentence is labeled with one of 13 pre-defined rhetorical roles. The Fleiss Kappa score is 0.59, which indicates a moderate agreement among annotators and highlights the inherent subjectivity of the task. The dataset is highly unbalanced. The most and lesser frequent rhetorical roles are ANALYSIS and PRE_NOT_RELIED with respectively 11,609 and 170 sentences. The dataset also presents a high variation in sentences' length. The shortest and longest training sentences respectively have 1 and 669 tokens. We refer the reader to Kalamkar et al. (2022) for more details about the dataset and its development.

4.2 Models and Evaluation

We employ two sets of models relying on DFCSCs, namely DFCSC-SEP and DFCSC-CLS.

1) DFCSC-SEP: the three models in this set rely on DFCSCs to get sentence embeddings from a pre-trained Transformer model. The first model exploits InCaseLaw (Paul et al., 2022) as a sentence encoder ($c\_len = 512$). The second model exploits RoBERTa-base (Liu et al., 2019) ($c\_len = 512$) while the third one exploits Longformer-base (Beltagy et al., 2020) ($c\_len = 1,024$). InCaseLaw is a BERT model pre-trained with corpora from the Indian legal system. Longformer is a model that relies on a sparse self-attention mechanism which allows it to deal with longer sentences than the BERT-based models. In all models, core sentences are represented by the hidden states of the respective separator tokens (the token at the right of a core sentence). We set $m = 350$ for the three models.

2) DFCSC-CLS: as before, but each core sentence is represented by the concatenation of hidden states of the special token marking the beginning of the

$s_1 =$ 'the cunning fox'
$s_2 =$ 'jumped the lazy dog'
$s_3 =$ 'hello world!'
$s_4 =$ 'Bob is the best player of chess in the world for sure'
$s_5 =$ 'My name is Bond'

$c\_len = 18$
$m = 6$

1st chunk: [CLS][SEP]**the cunning fox[SEP]jumped the lazy dog[SEP]**hello world ! bob is the[SEP]
2sd chunk: [CLS]cunning fox jumped the lazy dog[SEP]**hello world ![SEP]**bob is the best player[SEP]
3rd chunk: [CLS]![SEP]**bob is the best player of chess in the world for sure[SEP]**my[SEP]
4th chunk: [CLS]the best player of chess in the world for sure[SEP]**my name is bond[SEP]**[SEP]

Figure 2: Example of a set of DFCSCs assembled from a set of five sentences. The core sentences' tokens are formatted in bold. The employed tokenizer is the one from the DistilBERT-base-uncased model (Sanh et al., 2019).

chunk ([CLS] or <s>) and of the respective separator token ([SEP] or </s>).

We also employ SingleSC, Cohan, and Sharing edges as baselines. Each model is presented bellow.

SingleSC: single sentence classification models, that is, models that do not rely on chunks. Each sentence is individually fed to the model and it is represented by the hidden state of the respective token marking the beginning of the sentence (<s> or [CLS]). The exploited pre-trained Transformers are BERT-base-uncased, RoBERTa-base, and InCaseLaw.

Cohan: models that follows the chunk design from (Cohan et al., 2019), that is, chunks without sentence sharing. The exploited pre-trained Transformers are InCaseLaw, RoBERTa, and Longformer. Models based on InCaseLaw and RoBERTa have the following hyperparameter values: 85 as the maximum sentence length, 512 as the chunk length, and 7 as the maximum number of sentences in a chunk. The model based on Longformer has the following hyperparameter values: 90 as the maximum sentence length, 1,024 as the chunk length, and 16 as the maximum number of sentences in a chunk. Each sentence is represented by the hidden state of the following separator token.

Sharing edges: models relying on chunks with a fixed number of core sentences and edge sentences. The exploited pre-trained Transformers are InCaseLaw, RoBERTa-base, and Longformer-base. Models based on InCaseLaw and RoBERTa have the following hyperparameter values: 80 as the maximum sentence length, 512 as the chunk length, 2 as the number of core sentences in a chunk, and 2 as the number of edge sentences. The model based on Longformer has the following hyperparameter values: 85 as the maximum sentence length, 1,152 as the chunk length, 9 as the number of core sentences in a chunk, and 2 as the number of edge sentences. Each core sentence is represented by the hidden state of the following separator token.

Appendix A presents the remaining hyperparameters.

For all models, a fully-connected layer is employed as a classifier. All models are fine-tuned over the train set for four epochs and evaluated over the validation set. For each model, the fine-tuning is repeated four times with a different random seed for each repetition (the same four seeds for all models). The evaluation metric is the weighted F1 score.

5 Results

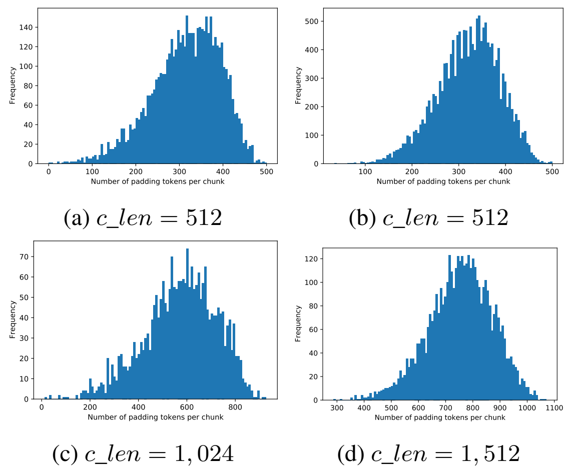
5.1 Experimental Results

Table 1 presents the scores achieved by all models. We start our analysis by considering the models employing the same pre-trained Transformer model as a sentence encoder. Firstly, and as expected, we verify that inter-sentence context exploitation is advantageous as all SingleSC models perform lower than their counterparts.

Secondly, we verify that both DFCSC approaches are really effective as they always overcome the baselines: DFCSC-SEP InCaseLaw and DFCSC-CLS InCaseLaw are respectively 4.29% and 3.76% better than Cohan InCaseLaw; DFCSC-SEP RoBERTa and DFCSC-CLS RoBERTa are respectively 4.12% and 4.52% better than Cohan RoBERTa; and DFCSC-SEP Longformer and DFCSC-CLS Longformer are respectively 6.32% and 8.71% better than Cohan Longformer.

Model	Epoch	F1	std
SingleSC BERT	2/4	0.6402	0.0068
SingleSC InCaseLaw	1/4	0.6774	0.0055
SingleSC RoBERTa	4/4	0.6666	0.0028
Cohan InCaseLaw	2/4	0.6992	0.0086
Cohan RoBERTa	4/4	0.7573	0.0042
Cohan Longformer	4/4	0.7487	0.0045
Sharing edges InCaseLaw	1/4	0.6923	0.0182
Sharing edges RoBERTa	1/4	0.6910	0.0055
Sharing edges Longformer	4/4	0.6537	0.0103
DFCSC-SEP InCaseLaw	2/4	0.7293	0.0076
DFCSC-SEP RoBERTa	4/4	0.7886	0.0043
DFCSC-SEP Longformer	4/4	0.7960	0.0054
DFCSC-CLS InCaseLaw	4/4	0.7256	0.0040
DFCSC-CLS RoBERTa	4/4	0.7915	0.0018
DFCSC-CLS Longformer	4/4	**0.8139**	0.0026

Table 1: Performance of models. The scores are averages of four evaluations and concern the best fine-tuning epoch. **std** stands for Standard Deviation. The best score is formatted in bold.



(a) $c\_len = 512$ (b) $c\_len = 512$ (c) $c\_len = 1,024$ (d) $c\_len = 1,512$

Figure 3: Histograms of the number of padding tokens per chunk according to the respective model and the training set. (a) Cohan InCaseLaw/RoBERTa, (b) Sharing edges InCaseLaw/RoBERTa, (c) Cohan Longformer, (d) Sharing edges Longformer.

Thirdly, we realize that DFCSCs provide more gains to RoBERTa and Longformer-based models than to InCaseLaw-based ones. This suggests that the formers exploit contextual data in a better way than the last, and we believe this is due to the pre-training process of such models: InCaseLaw follows the same procedure of BERT whereas Longformer and RoBERTa are pre-trained with another and the same procedure.

DFCSC-SEP Longformer and DFCSC-CLS Longformer achieve the two highest scores among all models. We hypothesize this is because Longformer-based models work with longer chunks that result in richer sentence embeddings since they are yielded from chunks with more context, and in no truncation of core sentences.

Histograms in Figure 3 show the number of padding tokens inserted into the chunks following Cohan and Sharing edges layouts. We can realize that most of such chunks have more padding tokens than content tokens. In our experiments, the DFCSCs do not have padding tokens because the edges of a DFCSC are always filled with content tokens. We deem that this is the main factor of the superior performance of DFCSC-based models: chunks with fewer padding tokens allow the Transformer encoders to yield better sentence embeddings.

The DFCSC-CLS approach appears to be better than DFCSC-SEP. Besides the improvements that the DFCSC-CLS approach provides to RoBERTa and Longformer-based models, the standard deviation scores of DFCSC-CLS models indicate that they have higher training stability.

The performance differences among Cohan and Sharing edges models are remarkable. We believe this is mainly due to a bad tuning of Sharing edges models' hyperparameters. This highlights another advantage of the DFCSC-based approach: it defines fewer hyperparameters and hence it presents an easier tuning.

5.2 DFCSC Hyperparameter Analysis

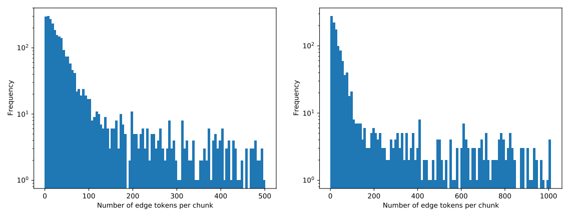
Based on the hypothesis that a richer context leads to better sentence embeddings, we deem that a good practice of hyperparameter optimization is by setting c_len with the maximum sequence length supported by the employed sentence encoder (e.g., 512 for BERT) and by the hardware platform.

Regarding the m hyperparameter, it is not obvious whether the larger its value, the larger the model performance. In order to assess the impact of m, we perform experiments on selected models with several values of m whose results are presented in Table 2. From such results, we see that 250 is the best value for both models. This suggests that the absolute amount is more important than the proportion of edges tokens in a chunk.

The assembly logic of DFCSC was not designed to deal with $m = 0$ and so we employ $m = 2$ to assure at least one token in each edge. This value leads to the worst results for each model. Although, the achieved results (0.7856 and 0.8009) are still better than the ones achieved by the models based on the Cohan layout (0.7573 and 0.7487

Model	m	c_len	m/c_len	**Epoch**	**F1**	**std**
DFCSC-CLS RoBERTa	2	512	0.00	4/4	0.7856	0.0025
	100	512	0.20	4/4	0.7969	0.0045
	250	512	0.49	4/4	0.8105	0.0051
	350	512	0.68	4/4	0.7915	0.0018
DFCSC-CLS Longformer	2	1,024	0.00	4/4	0.8009	0.0055
	100	1,024	0.10	4/4	0.8155	0.0038
	250	1,024	0.24	4/4	0.8188	0.0035
	350	1,024	0.34	4/4	0.8139	0.0026
	512	1,024	0.50	4/4	0.8163	0.0008
	700	1,024	0.68	4/4	0.8167	0.0034

Table 2: Impact of m on the models' performance. The scores are averages of four evaluations and concern the best fine-tuning epoch. **std** stands for Standard Deviation.



(a) $c\_len = 512$ (b) $c\_len = 1,024$

Figure 4: Histograms of the number of shared tokens per chunk regarding the DFCSC layout with $m = 2$ and the training set.

from Table 1). We remark that, despite the value assigned to m, many chunks will have more than 2 edge tokens as can be verified in Figure 4. As consequence, rich context is often employed when generating sentence embeddings.

There are no large differences in performance from the exploited values of m. We believe this is due to the characteristics of the DFCSC approach: core sentences are usually not truncated and hence there is no data loss; context is always provided since the filling of edges is oriented to tokens instead to sentences; and edges are filled with content tokens instead padding tokens. The low sensitivity to m is a good characteristic of the approach since it eases the hyperparameter optimization effort and leads to more predictable models.

5.3 Shared Task Results

The resulting board of the shared task shows 27 participants from which our team took 9th place with a 0.8076 weighted F1 score. The best-achieved score is 0.8593 by the AntContentTech team. We made five submissions and our best model was a DFCSC-CLS Longformer fine-tuned for four epochs over the train and development sets ($c\_len = 1,280$ and $m = 350$). This model overcomes the hierarchical baseline model reported by the authors of the dataset (Kalamkar et al., 2022) that achieves a 0.7900 F1 score.

6 Conclusions

This work presents a novel approach to leverage the context exploitation ability of Transformers and employs it to tackle the Legal RRL subtask of SemEval-2023 Task 6. The proposal relies on DFCSCs as a way of achieving better sentence embeddings from Transformers' hidden states. Models based on DFCSCs are compared to baselines based on similar approaches and the results show that our proposal is effective: the best DFCSC model achieves a 0.8139 F1 score whereas the best baseline does 0.7573. In the future, we plan to employ DFCSCs over other Legal RRL datasets as a way of reinforcing our findings.

Acknowledgements

This work has been supported by the Coordination for the Improvement of Higher Education Personnel - Brasil (CAPES) and was partially supported by the LawBot project (ANR-20-CE38-0013), granted by ANR the French Agence Nationale de la Recherche.

References

Roberto Aragy, Eraldo Rezende Fernandes, and Edson Norberto Cáceres. 2021. Rhetorical role identification for portuguese legal documents. In *Intelligent Systems - 10th Brazilian Conference, BRACIS 2021, Virtual Event, November 29 - December 3, 2021, Proceedings, Part II*, volume 13074 of *Lecture Notes in Computer Science*, pages 557–571. Springer.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Paheli Bhattacharya, Shounak Paul, Kripabandhu Ghosh, Saptarshi Ghosh, and Adam Wyner. 2021. Deeprhole: deep learning for rhetorical role labeling of sentences in legal case documents. *Artificial Intelligence and Law*, pages 1–38.

Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Dan Weld. 2019. Pretrained language models for sequential sentence classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3693–3699, Hong Kong, China. Association for Computational Linguistics.

Alexandre G de Lima, Mohand Boughanem, Eduardo Henrique da S Aranha, Taoufiq Dkaki, and Jose G Moreno. 2022. Exploring sbert and mixup data augmentation in rhetorical role labeling of indian legal sentences. In *Joint Conference of the Information Retrieval Communities in Europe (CIRCLE) 2022*.

Alexandre G de Lima, Jose G Moreno, Mohand Boughanem, Taoufiq Dkaki, and Eduardo Henrique da S Aranha. 2023. Leveraging positional encoding to improve fact identification in legal documents. In *First international workshop on Legal Information Retrieval (LegalIR) at ECIR 2023*, pages 11–13.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Prathamesh Kalamkar, Aman Tiwari, Astha Agarwal, Saurabh Karn, Smita Gupta, Vivek Raghavan, and Ashutosh Modi. 2022. Corpus for automatic structuring of legal documents. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4420–4429, Marseille, France. European Language Resources Association.

Dongjin Li, Ke Yang, Lijun Zhang, Dawei Yin, and Dezhong Peng. 2021. Class: A novel method for chinese legal judgments summarization. In *Proceedings of the 5th International Conference on Computer Science and Application Engineering*, CSAE '21, New York, NY, USA. Association for Computing Machinery.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Vijit Malik, Rishabh Sanjay, Shouvik Guha, Angshuman Hazarika, Shubham Kumar Nigam, Arnab Bhattacharya, and Ashutosh Modi. 2022. Semantic segmentation of legal documents via rhetorical roles. In *Proceedings of the Natural Legal Language Processing Workshop, EMNLP 2022*, Abu Dhabi, UAE. Association for Computational Linguistics.

Ashutosh Modi, Prathamesh Kalamkar, Saurabh Karn, Aman Tiwari, Abhinav Joshi, Sai Kiran Tanikella, Shouvik Guha, Sachin Malhan, and Vivek Raghavan. 2023. SemEval-2023 Task 6: LegalEval: Understanding Legal Texts. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, Toronto, Canada. Association for Computational Linguistics (ACL).

James Mullenbach, Yada Pruksachatkun, Sean Adler, Jennifer Seale, Jordan Swartz, Greg McKelvey, Hui Dai, Yi Yang, and David Sontag. 2021. CLIP: A dataset for extracting action items for physicians from hospital discharge notes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1365–1378, Online. Association for Computational Linguistics.

Shounak Paul, Arpan Mandal, Pawan Goyal, and Saptarshi Ghosh. 2022. Pre-training transformers on indian legal text. *CoRR*, abs/2209.06049.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Jaromír Savelka, Hannes Westermann, and Karim Benyekhlef. 2020. Cross-domain generalization and knowledge transfer in transformers trained on legal data. In *Proceedings of the Fourth Workshop on Automated Semantic Analysis of Information in Legal Text held online in conjunction with the 33rd International Conference on Legal Knowledge and Information Systems, ASAIL@JURIX 2020, December 9, 2020*, volume 2764 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Esther van den Berg and Katja Markert. 2020. Context in informational bias detection. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6315–6326. International Committee on Computational Linguistics.

A Training and Implementation Details

Python 3.8.8, PyTorch 1.9.0, and Hugging Face 4.17.0 are the programming language and main libraries utilized by us to develop all models. Hugging Face is mainly used to load pre-trained Transformer models.

Pre-trained Transformer models are utilized with default parameters, except for Longformer ones whose global attention is set to take `<s>` and `</s>` tokens into account.

Model	**Batch Size**	**Dropout**	**LR**	**Training Time**
SingleSC BERT	8	0.1	10^{-5}	07h27m29s
SingleSC InCaseLaw	8	0.1	10^{-5}	07h21m36s
SingleSC RoBERTa	8	0.1	10^{-5}	07h30m46s
Cohan InCaseLaw	4	0.1	10^{-5}	01h35m00s
Cohan RoBERTa	4	0.1	10^{-5}	01h37m18s
Cohan Longformer	2	0.1	$2 \cdot 10^{-5}$	03h34m54s
Sharing edges InCaseLaw	4	0.1	10^{-5}	04h45m33s
Sharing edges RoBERTa	4	0.1	10^{-5}	04h52m32s
Sharing edges Longformer	2	0.1	$2 \cdot 10^{-5}$	08h55m11s
DFCSC-SEP InCaseLaw	4	0.1	10^{-5}	02h21m03s
DFCSC-SEP RoBERTa	4	0.1	10^{-5}	02h33m24s
DFCSC-SEP Longformer	2	0.1	$2 \cdot 10^{-5}$	02h24m12s
DFCSC-CLS InCaseLaw	4	0.1	10^{-5}	02h22m07s
DFCSC-CLS RoBERTa	4	0.1	10^{-5}	02h32m05s
DFCSC-CLS Longformer	2	0.1	$2 \cdot 10^{-5}$	02h16m28s

Table 3: Training hyperparameters and training time of models referenced in Table 1. Training time concerns four executions of a model. **LR** indicates the initial Learning Rate.

For all models, we employ the Cross-Entropy loss function and Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $weight\_decay = 10^{-3}$). The learning rate is linearly scheduled from an initial value to zero and there are no warm-up steps.

A single NVIDIA GeForce GTX 1080 Ti GPU with 11GB is used to train all models.

Table 3 presents the training hyperparameters values employed to train the models presented in Table 1. The indicated dropout rate is applied only in the fully-connected layer.