Exploring Large Language Models and Hierarchical Frameworks for Classification of Large Unstructured Legal Documents

Nishchal Prasad, Mohand Boughanem, Taoufiq Dkaki

Exploring Large Language Models and Hierarchical Frameworks for Classification of Large Unstructured Legal Documents

Nishchal Prasad[0009−0000−4712−3540], Mohand Boughanem[0000−0001−7004−0807], and Taoufiq Dkaki[0000−0003−3962−7663]

Institut de Recherche en Informatique de Toulouse (IRIT), Toulouse, France
{Nishchal.Prasad, Mohand.Boughanem, Taoufiq.Dkaki}@irit.fr

Abstract. Legal judgment prediction suffers from the problem of long case documents exceeding tens of thousands of words, in general, and having a non-uniform structure. Predicting judgments from such documents becomes a challenging task, more so on documents with no structural annotation. We explore the classification of these large legal documents and their lack of structural information with a deep-learning-based hierarchical framework which we call MESc; "Multi-stage Encoder-based Supervised with-clustering"; for judgment prediction. Specifically, we divide a document into parts to extract their embeddings from the last four layers of a custom fine-tuned Large Language Model, and try to approximate their structure through unsupervised clustering. Which we use in another set of transformer encoder layers to learn the inter-chunk representations. We analyze the adaptability of Large Language Models (LLMs) with multi-billion parameters (GPT-Neo, and GPT-J) with the hierarchical framework of MESc and compare them with their standalone performance on legal texts. We also study their intra-domain(legal) transfer learning capability and the impact of combining embeddings from their last layers in MESc. We test these methods and their effectiveness with extensive experiments and ablation studies on legal documents from India, the European Union, and the United States with the ILDC dataset and a subset of the LexGLUE dataset. Our approach achieves a minimum total performance gain of approximately 2 points over previous state-of-the-art methods.

Keywords: Legal judgment prediction · Long document classification · Multi-stage hierarchical classification framework.

1 Introduction

A legal case proceeding cycle[1] involves analyzing vast amounts of data and legal precedents, which can be a time-consuming process given the complexity and length of the case. The number of legal cases in a country is also proportionally related to its population. This leads to a backlog of cases, especially in highly

[1] https://www.law.cornell.edu/wex/civil_procedure

populated countries, ultimately setting back the progress of its legal system[2][17]. Automating such legal case procedures can help speed up and strengthen the decision-making process, saving time and benefiting both the legal authorities and the people involved. One of the fundamental problems that deal with this larger component is the prediction of the outcome based just on the case's raw texts (which can include facts, arguments, appeals, etc. except the final decision), as in a typical real-life (raw) setting.

Several machine learning techniques have been applied to legal texts to predict judgments as a text classification problem ([14], [12]). While it seems like a general text classification task, legal texts differ from general texts and are rather more complex, broadly in two ways, i.e. structure and syntax and, lexicon and grammar ([43], [7], [23]). The structure of legal case documents is not uniform in most settings and their complex syntax and lexicon make it more difficult and expensive to annotate, requiring only legal professionals. This adds to another challenge of the long lengths of these documents, reaching more than 10000 words (Table 2). This lack of structure information and the long lengths of these legal documents pose a challenge in predicting judgments. In our work we explore this problem on four fronts, by (a) developing a hierarchical framework for the classification of large unstructured legal documents, (b) exploring the adaptability of billion-parameter large language models (LLMs) to this framework, (c) analyzing the performance of these LLMs without this framework and (d) checking the intra-domain(legal) transfer learning capability of domain-specific pre-trained LLMs. This is summarized below:

- We explore the problem of judgment prediction from large unstructured legal documents and propose a hierarchical multi-stage neural classification framework named "Multi-stage Encoder-based Supervised with-clustering" (MESc). This works by extracting embeddings from the last four layers of a fine-tuned encoder of a large language model (LLM) and using an unsupervised clustering mechanism to approximate the structure. Alongside the embeddings, these approximated structure labels are processed through another set of transformer encoder layers for final classification.
- We show the effect of combining features from the last layers of transformer-based LLMs (BERT[13], GPT-Neo[3], GPT-J[32]), along with the impact on classification upon using the approximated structure.
- We study the adaptability of domain-specific pre-trained multi-billion parameter LLMs to such documents and study their intra-domain(legal) transfer learning capability (both with fine-tuning and in MESc).
- We performed extensive experiments and analysis on four different datasets (ILDC[20] and LexGLUE's [9] ECtHR(A), ECtHR(B), and SCOTUS) and achieved a total gain of ≈ 2 points in classification on these datasets.

[2] `https://www.globaltimes.cn/page/202204/1260044.shtml`

2 Related works

Several strategies have been investigated to predict the result of legal cases in specific categories (criminal, civil, etc.) with rich annotations (Xiao et al. [34], Xu et al. [35], Zhong et al. [42], Chen et al. [10]). These studies on well-structured and annotated legal documents show the effect and importance of having good structural information. While creating such a dataset is both time and resource (highly skilled) demanding, researchers have worked on legal documents in a more general and raw setting. Chalkidis et al. [5] presented a dataset of European Court of Human Rights case proceedings in English, with each case assigned a score indicating its importance. They described a Legal Judgment Prediction (LJP) task, which seeks to predict the outcome of a court case using the case facts and law violations. They also create another version of this dataset [8] to give a rational explanation for the predictions. In the US legal case setting, Kaufman et al. [18] used AdaBoost decision tree to predict the U.S. Supreme Court rulings. Tuggener et al. [30] proposed LEDGAR, a multilabel dataset of legal provisions in US contracts. Malik et al. [20] curated the Indian Legal Document Corpus (ILDC) of unannotated and unstructured documents, and used it to build baseline models for their Case Judgment Prediction and Explanation (CJPE) task upon which Prasad et al. [25] showed the possibility of intra-domain(legal) transfer learning using LEGAL-BERT on Indian legal texts.

Pretrained large language models (LLMs) based on transformers (Devlin et al.[13], Vaswani et al.[31]) have shown widespread success in all fields of natural language processing (NLP) but only for short texts spanning a few hundred tokens. There have been several approaches to handle longer sequences with long sequence transformer-based LLMs (Beltagy et al.[2], Kitaev et al.[19], Zaheer et al.[39], Ainslie et al. [1]). These architectures display similar performance as the hierarchical adaptation of their vanilla counterparts ([9],[6],[11]), and since we try to learn and approximate the structure information of the document, we choose to process the document in short sequences rather than as a whole. So, we take a different approach to handle large documents with LLMs (such as BERT [13]) based on the hierarchical idea of "divide, learn and combine" (Chalkidis et al.[6], Zhang et al. [40], Yang et al. [37]), where the document is split (into parts then sentences and words, etc.) and features of each component are learned and combined hierarchically from bottom-up to get the whole document's representation. Also with the unavailability of the domain-specific pre-trained checkpoints of these long sequence LLMs and considering their expensive pretraining, we choose to use the vanilla models and develop a hierarchical adaptation.

Moreover, the domain-specific pre-training of transformer encoders has accelerated the development of NLP in legal systems with better performance as compared to the general pre-trained variants (Chalkidis et al.[7]'s LEGAL-BERT trained on court cases of the US, UK, and EU, Zheng et al. [41]'s BERT trained on US court cases dataset CaseHOLD, Shounak et al. [24]'s InLegal-BERT and InCaseLawBERT trained on the Indian legal cases). Recently, with the emergence of multi-billion parameter LLMs such as GPT-3 [4], LLaMA [28], LaMDA [27], and their superior performance in natural language understanding,

researchers have tried to adapt their variants (with few-shot learning) to legal texts (Trautmann et al. [29], Yu et al. [38]). In this paper, with full-fine tuning, we check the adaptability of these billion parameter LLMs with the hierarchical framework and also their intra-domain(legal) transfer-learning compared to the intra-domain pre-training (as done in LEGAL-BERT, InLegal-BERT). To do so we use three such variants of GPT (GPT-Neo (1.3 and 2.7)[3], GPT-J[32]) pre-trained on Pile[15], which has a subset (FreeLaw) of court opinions of US legal cases.

3 Method: Classification Framework (MESc)

To handle large documents MESc architecture shares the general hierarchical idea of divide, learn, and combine ([6], [40], [37]) but differs from the previous works in the following: (a) It uses the last four layers of the fine-tuned transformer-based LLM for extracting global representations for parts(chunks) of the document. (b) Approximating the document structure by applying unsupervised learning (clustering) on these representations' embeddings and using this information alongside, for classification. (c) Instead of only RNNs, different combinations of transformer encoder layers are tested to get a global document representation. (d) Divide the process into four stages, fine-tuning, extracting embeddings, processing the embeddings (supervised + unsupervised learning), and classification.

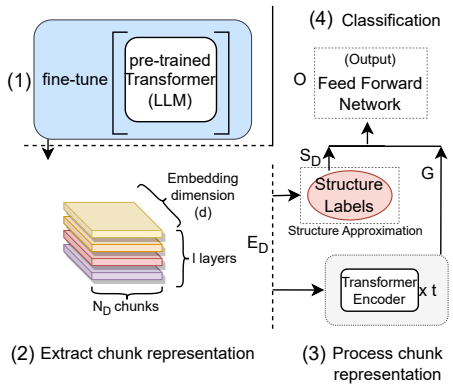

Fig. 1. Multi-stage Encoder-based Supervised with-clustering (MESc) framework.

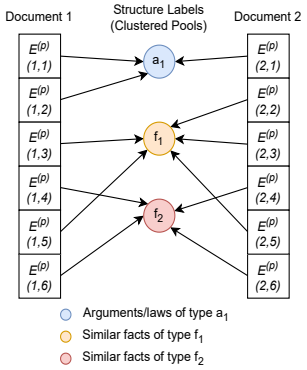

Fig. 2. An example of clustering of chunk representations of two documents to generate structure labels.

An overview of MESc can be seen in Fig. 1. An input document D is tokenized into a sequence of tokens, $D = \{t_{1,D}, t_{2,D}, \cdots, t_{L_D,D}\}$ via a tokenizer specific to a chosen pre-trained transformer-based language model (BERT, GPT etc.), where $t \in \mathbb{N}$ and $\mathbb{N}$ is the vocabulary of the tokenizer. This token sequence is split into

a set of blocks (chunks) $\{C_{1,D}, C_{2,D}, \cdots, C_{N_D,D}\}$ with overlaps(o). Where each chunk, $C_{i,D} = \{t_{((i-1)\times(c-o))}, \cdots, t_{((i-1)\times(c-o)+c)}\}$ with c being the maximum number of tokens in the chunk, which is a predefined parameter for MESc (e.g. 512). $N_D = \lceil \frac{L_D}{c-o} \rceil$ is the total number of chunks for a document having L tokens in total, with $o \ll c$. N_D varies with the length of the document.

Stage 1 - Fine-tuning: To each chunk of a document, we associate the gold class label of the document l_D, and combine them to form a token matrix:

$$I_D \in \mathbb{R}^{N_D \times c \times 1} \leftarrow [\{C_{1,D}, l_D\}, \{C_{2,D}, l_D\}, \cdots, \{C_{N_D,D}, l_D\}] \tag{1}$$

This is used as input for the document for fine-tuning the pre-trained LLM, where N_D is the batch size for one pass through the encoder. This allows the encoder to adapt to the domain-specific legal texts, which helps get richer features for the next stage.

Stage 2 - Extracting chunk embeddings: Different layers of transformer models learn varied representations of the input sentence ([36,16,26]). When simultaneously used alongside each other these representations can be used to give varied features for further learning. Since the last pre-trained LLM layer captures the final representation of a chunk, we use it alongside its immediate lower layers to extract the chunk representation. We use the immediate lower layers with the intuition that the representations learned are not heavily but enough varied from the final layer.

For a document, we pass its chunks C_i through the fine-tuned encoder and extract its representation embeddings ($E_{i,D}$) from the last l layers. $E_{i,D} \in \mathbb{R}^{l \times d}$, where d is the dimension of the features (we use $l = 4$). The representation embeddings can be either the first token (as in BERT) or the last token for causal language models (as in GPT). We accumulate all $E_{i,D}$ of a document to form an embedding matrix:

$$E_D \in \mathbb{R}^{N_D \times l \times d} \leftarrow \left[E_{1,D}, E_{2,D}, \cdots, E_{N_D,D}\right] \tag{2}$$

The $E_{i,D}$ acts as a representation of the chunk in this context, and combining them yields an approximate representation of the entire document. Doing this for all the documents gives us generated training data.

Stage 3 - Processing the extracted representations: Since the features extracted from the last layers of a fine-tuned encoder have different embedding spaces, they can contribute to give varied features. So for this stage, we choose to combine the last $p < l$ layers in E_D for further training. We experiment with different p before fixing one value as discussed in section 4. This gives $E_D^{(p)} \in \mathbb{R}^{N_D \times p \times d}, p \in \{1, 2, 3, 4\}$. We used p=1, 2, and 4 in our experiments to compare their effects. We concatenate together the representations from these p layers to get,

$$E_{i,D}^{(p)} \in \mathbb{R}^{pd \times 1} \leftarrow \left[E_{i,D}^{(l)} | E_{i,D}^{(l-1)} | \cdots | E_{i,D}^{(l-p-1)}\right] \tag{3}$$

This gives,

$$\widehat{E}_D^{(p)} \in \mathbb{R}^{N_D \times pd} \leftarrow \left\{ E_{1,D}^{(p)} | E_{2,D}^{(p)} | \cdots | E_{N,D}^{(p)} \right\} \tag{4}$$

We also experimented with the element-wise addition of representations in $E_D^{(p)}$ and found their performance to be lower in most of the experiments of section 4, hence we exclude it here.

a. **Approximating the structure labels (S_D)** (Unsupervised learning): To get the information on the document's structure i.e. its parts (facts, arguments, concerned laws, etc.), we use a clustering algorithm (HDBSCAN [21]). We cluster the p chosen extracted chunk embeddings, $\widehat{E}_D^{(p)}$ to map similar parts of different documents together where the labels of one part of a document are learned by its similarity with another part of another document. The idea is that the embeddings of similar parts from different documents will group forming a pool of labeled clusters that can help identify its part in the document. A synthetic example can be seen in Fig. 2, where the $E_{i,D}^{(p)}$ of documents 1 and 2 learn their cluster (label) pool for, arguments of type a_1=$\{E_{1,1}^{(p)}, E_{1,2}^{(p)} E_{2,1}^{(p)}\}$, facts of type f_1=$\{E_{1,3}^{(p)}, E_{1,5}^{(p)}, E_{2,2}^{(p)}, E_{2,3}^{(p)}, E_{2,5}^{(p)}\}$, facts of type f_2=$\{E_{1,4}^{(p)}, E_{1,6}^{(p)}, E_{2,4}^{(p)}, E_{2,6}^{(p)}\}$. So for document 1 the approximated structure then becomes S_1=$\{a_1,a_1,f_1,f_2,f_1,f_2\}$ and for document 2 it is S_2=$\{a_1,f_1,f_1,f_2,f_1,f_2\}$. It is to be noticed that this distinction if it's a fact or an argument etc. is done here for representation. In an actual setting, this is unknown and the labels don't carry any specific name or meaning except for the model to give an approximation of its structure.
Since the performance of the HDBSCAN clustering mechanism decreases significantly with an increase in data dimension, we use a dimensionality reduction algorithm (pUMAP[22]), before clustering. For all the chunks of a document, their approximated structure labels are combined with the output of stage 3(b), before processing through the final classification stage (4).

b. **Global document representation** (Supervised learning): For intra-chunk attention, we use transformer encoder layers (Vaswani et al. [31]), for a chunk to attend to another through its multi-head attention and feed-forward neural network (FFN) layer. This helps the chunk representations to attend to one another in parallel. For a chunk's position in the document, we add its positional embeddings ([13]) in $E_D^{(p)}$ and process it through t transformer layers $T_{\{h,d_f\}}^{(t)}$, with h attention heads and $d_f = pd$ as the dimension of the FFN. t and h are both hyperparameters whose choice depends upon the input feature lengths. Section 4 evaluates different values of these parameters, but $t \geq 3$ sometimes overfits the model in our experiments, hence we fix $t = 2$ for MESc. The output is max-pooled and passed through a feed-forward neural network FFN_T of 128 nodes to get:

$$G\left(\widehat{E}_D^{(p)}\right) = FFN_T\left(maxpool\left(T_{\{h,d_f\}}^{(t)}\left(\widehat{E}_D^{(p)}\right)\right)\right) \in \mathbb{R}^{128} \tag{5}$$

Stage 4 - Classification: The structure labels along with the output of the feed-forward network of stage 3(b) are concatenated together and passed through

an internal feed-forward network FFN_i (32 nodes, with softmax activation) and an external feed-forward network FFN_e (u label/class nodes with task-specific activation function sigmoid or softmax) giving the output $O(D)$ for a document D (Eq. 6). O and G are learnt together while S_D is learnt independently.

$$O\left(D\right) = FFN_e\left(FFN_i\left(\left(\left[G\left(\widehat{E}_D^{(p)}\right)|S_D\right]\right)\right)\right) \in \mathbb{R}^u \tag{6}$$

The code and trained models for the above implementation can be found at GitHub[3] and our finetuned LLMs at HuggingFace[4].

3.1 Experimental setup

Table 1 lists the major details for the experimental setup. For our backbone transformer-based language model, we used domain-specific models LEGAL-BERT[7], InLegalBERT[24], and for multi-billion parameter LLMs we chose GPT-Neo[3], GPT-J[32]. The tokenizers used are from the respective backbone transformer encoders. We abbreviate the encoders fine-tuned on 512 input length as (α) and, for ones fine-tuned with 2048 input length as (γ).

These hyperparameters (Table 1) were used based on the guidelines of the respective language models and several of our previous experiments and dataset analyses. We list out some of them further in the paper and in discussions while referring to Table 4, Table 5, Table 2, and Fig. 3.

3.2 Dataset:

We chose the legal datasets having large documents with a nonuniform structure throughout and without any structural annotations. The ILDC dataset [20] includes highly unstructured 39898 English-language case transcripts from the Supreme Court of India (SCI), where the final decisions have been removed from the document. Upon analyzing the documents from their sources and the dataset we found that they are highly unstructured and noisy. The initial decision between "rejected" and "accepted" made by the SCI judge(s) is used to classify each document and serves as their decision label. The LexGLUE dataset [9] comprises a set of seven datasets from the European Union and US court case setting, for uniformly assessing model performance across a range of legal NLP tasks, from which we choose ECtHR (Task A), ECtHR (Task B), and SCOTUS as they are classification tasks involving long unstructured legal documents. ECtHR (A and B) are court cases from the European Convention on Human Rights (ECHR) for articles that were violated or allegedly violated. The dataset contains factual paragraphs from the description of the cases. SCOTUS consists of court cases from the highest federal court in the United States of America,

[3] `https://github.com/NishchalPrasad/MESc`
[4] `https://huggingface.co/nishchalprasad`
[5] `https://umap-learn.readthedocs.io/en/latest/parametric_umap.html`
[6] `https://www.deepspeed.ai/`
[7] `https://huggingface.co/docs/accelerate/index`

Table 1. Experimental setup for different stages of MESc architecture.

Stage 1	
BERT-based LLM: chunk-size = 512 tokens (90 token overlaps), [CLS] token to test. *GPT-based LLM*: max input length=2048, chunk-size∈{512, 2048}, last token to test. For all (α) GPT we compare with (α) BERT-based LLM on 512 input length. Finetuned for e=4 epochs, chose best e for Stage 2 and evaluation.	
Stage 2 (Embedding Extraction)	Stage 3 & 4
BERT-based LLM: [CLS] token for each chunk	*Optimizer* = Adam (learning rate = $3.5e^{-6}$) *Loss func.:* multi-label: categorical cross-entropy binary & multi-class: binary cross-entropy t={1, 2, 3}, h=8, e=5 epochs (best e for evaluation)
GPT-based LLM: last token for each chunk	*Structure approximation:* pUMAP[5] (64 dimensions) HDBSCAN (15 min clusters)
GPU used: Nvidia V100 & A100, with ZERO-3 in Deepspeed[6] with Accelerate[7]. Maximum fine-tune time (hours/epoch) for GPTs (6 Nvidia A100): GPT-Neo-1.3B = 2.1, GPT-Neo-2.7B = 4, GPT-J = 8	

Table 2. Dataset statistics

Name		ILDC	ECtHR(A)	ECtHR(B)	SCOTUS
No. of Docs.	Train	37387	9000	9000	5000
	Val.	994	1000	1000	1400
	Test	1517	1000	1000	1400
Average tokens	Train	4120	2011	2011	8291
		501275	46500	46500	126377
Max tokens	Val.	8048	2210	2210	12639
		51045	18352	18352	56310
	Test	5238	2401	2401	12597
		55703	20835	20835	124955
No. of labels		2	10	10	13
Problem Type		Binary	Multi-Label	Multi-Label	Multi-Class

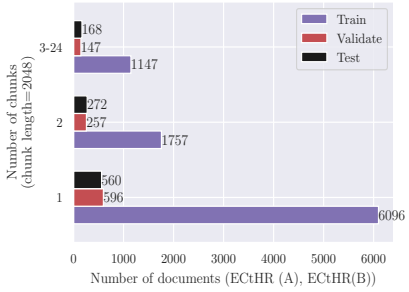

Fig. 3. Number of documents vs. the number of chunks for ECtHR.

Table 3. Class distribution of the datasets.

(Problem type)		class : # documents												
ILDC (Binary)	Train	0: 22067	1: 15320											
	Val.	0: 497	1: 497											
	Test	0: 755	1: 762											
ECtHR (A) (Multi-label)	Train	0: 505	1: 1349	2: 1368	3: 4704	4: 710	5: 41	6: 291	7: 110	8: 141	9: 1421			
	Val.	0: 57	1: 193	2: 187	3: 300	4: 87	5: 4	6: 42	7: 33	8: 18	9: 139			
	Test	0: 56	1: 189	2: 166	3: 299	4: 123	5: 5	6: 77	7: 37	8: 16	9: 122			
ECtHR (B) (Multi-label)	Train	0: 623	1: 1740	2: 1623	3: 5437	4: 1056	5: 81	6: 441	7: 162	8: 444	9: 1558			
	Val.	0: 75	1: 236	2: 219	3: 394	4: 153	5: 9	6: 64	7: 39	8: 34	9: 168			
	Test	0: 76	1: 234	2: 196	3: 394	4: 188	5: 11	6: 106	7: 43	8: 32	9: 155			
SCOTUS (Multi-class)	Train	0: 1011	1: 811	2: 423	3: 193	4: 45	5: 35	6: 255	7: 1043	8: 717	9: 191	10: 53	11: 220	12: 3
	Val.	0: 360	1: 218	2: 108	3: 70	4: 22	5: 35	6: 51	7: 226	8: 165	9: 83	10: 14	11: 38	12: 10
	Test	0: 372	1: 222	2: 88	3: 51	4: 28	5: 17	6: 24	7: 260	8: 200	9: 83	10: 15	11: 37	12: 3

with metadata from SCDB [8]. The details of the number of labels, the document lengths (in tokens), task description, and class distribution can be found in Table 2 and Table 3. The tokenization Table 2 is done using the tokenizer of GPT-J.

For performance comparison on LexGLUE, we used the SOTA benchmark of Chalkidis et al. [9], Condevaux et al.'s LSG [11], Chalkidis et al.'s HAT [6]

[8] `http://scdb.wustl.edu/`

and for ILDC we used its benchmark from [20] and of Shounak et al. [24]'s experiments.

4 Results and discussion

μ-F1 (micro) and m-F1 (macro) are used to measure the performance for the LexGLUE dataset, and accuracy(%) and macro-F1 for the ILDC dataset. These metrics were chosen partly to compare with the previous benchmark models (stated in Table 4) conforming to their original results and metrics. We list out the detailed experimental results for best configurations of MESc in Table 5 and the fine-tuned performance of the LLMs used in Table 4.

Intra-domain(legal) transfer learning: Based on the analysis of ILDC by Malik et al. [20] we use the last chunk of the documents for evaluation. As can be seen from Table 4, for LexGLUE's subset, all the GPTs used here adapt better than the BERT-based models with a minimum of ≈ 3 points gain on μ-F1 and a minimum of ≈ 6 points on m-F1 score. On the other hand in the ILDC dataset, for the α variants with 512 input lengths for evaluation, the performance dropped or remained similar to the InLegalBERT, while upon increasing the evaluation input length to 2048 we can see an increase of more than 1 point in the performance. When fine-tuned with 2048 input length, the performance of GPT-J (γ) compared to its α and β variant is at least ≈ 2 points higher for all the datasets. We can see that an increase in the input length for fine-tuning helps to capture more feature information for such documents. Also going from GPT-Neo-1.3B to GPT-Neo-2.7B to GPT-J-6B, the performance increases by a margin of 2 points at minimum, here we see the parameter count playing an important role in adapting and understanding these documents. Even though GPT-Neo and GPT-J are pre-trained on US legal cases (Pile [15]) they adapt better to the European and Indian legal documents, with a minimum gain of $\approx$ 7 points (γ) on the ECtHR(A & B) and the ILDC dataset over their domain-specific pre-trained counterparts LEGAL-BERT and InLegalBERT respectively. These results show the transfer learning capacity of LLMs between different legal domains with different settings, which can be a better alternative with limited resources compared to expensive domain-specific pre-training.

Performance with MESc: Looking at Table 5 we interpret the results in two directions.

(a) Encoders fine-tuned on 512 input length (α)**:** For LEGAL-BERT and InLegalBERT in all datasets, MESc achieves a significant increase in performance by at least 4 points in all metrics than their fine-tuned LLM counterparts with just the last layer (p=1). Combining the last four layers with t=1 encoder layer yields a performance boost of 4 points or more in ECtHR datasets while there is not much improvement in ILDC and SCOTUS. With S_D, the approximated structure labels, there is a slight performance increase in the ILDC. The same goes for SCOTUS with ≈ 1 point increase. With the same configuration

Table 4. Fine-tuned results on the last chunk for the chosen LLMs (Section 3.1)

α: fine-tuned and evaluated with 512 input length, β: evaluating α on its maximum input length, γ: fine-tuned and evaluated with its maximum input length. All measures are in (%). e = epoch.					
Dataset		**LEGAL-BERT** (μ-F1/m-F1)	**GPT-Neo 1.3B** (μ-F1/m-F1)	**GPT-Neo 2.7B** (μ-F1/m-F1)	**GPT-J 6B** (μ-F1/m-F1)
LexGLUE's subset	ECtHR (A)	(α) 62.85/48.66 (e = 4)	(α)66.19/56.59 (β)66.20/57.16 (e = 2)	(α)68.49/54.45 (β)68.11/56.49 (e = 2)	(α)71.42/59.27 (β)73.30/62.45 (γ)**74.51/64.67** (e = 3)
	ECtHR (B)	(α) 70.89/64.05 (e = 3)	(α)75.42/70.91 (β)75.74/70.09 (e = 2)	(α)74.48/68.26 (β)75.13/70.72 (e = 2)	(α)77.15/73.26 (β)80.49/76.31 (γ)**83.16/79.27** (e = 3)
	SCOTUS	(α) 68.76/53.57 (e = 6)	(α)71.14/60.35 (β)73.71/63.10 (γ)75.02/64.38 (e = 2)	(α)70.57/60.25 (β)73.64/65.64 (γ)76.36/66.19 (e = 1)	(α)72.00/62.76 (β)75.71/66.25 (γ)**78.50/71.96** (e = 3)
		InLegalBERT (Acc./m-F1)	Accuracy (Acc.) / m-F1		
ILDC		(α) **76.00/76.10** (e = 4)	(α)72.91/72.91 (β)77.26/77.25 (e=1)	(α)74.29/74.24 (β)81.21/81.18 (e=1)	(α)73.96/73.96 (β)81.93/81.92 (γ)**83.72/83.66** (e=1)

Table 5. Test results for different configurations of MESc. We show the maximum scores attained in all the runs. The bold-faced values also signify statistically significant findings in 5 different runs. (The baseline results are from their original papers.)

* is the fine-tuned LLM used for embedding extraction (Table 4). p = last p layers of the * model; t transformer encoder layers; S_D = approximated structure.							
			ECtHR (A)	**ECtHR (B)**	**SCOTUS**		**ILDC**
			(%) **μ-F1/m-F1**				(%) **Accuracy/m-F1**
Chalkidis et al.[9]			71.2/64.7	80.4/74.7	76.6/66.5		
LSG [11]			71.7/63.9	81.0/75.1	74.5/62.6	Malik et al.[20]	77.78/77.79
HAT [6]			-	80.8/79.8	-	Shounak et al.[24]	-/83.09
MESc							
	p, t	S_D					
LEGAL-BERT* (α)	p=1, t=1	✗	68.25/58.06	74.18/68.90	71.36/59.16	InLegalBERT* (α)	83.72/83.73
		✓	-	-	-		83.65/83.65
	p=1, t=2	✗	69.23/59.35	73.86/67.42	71.52/58.17		83.45/83.47
		✓	-	-	-		83.78/83.78
	p=4, t=1	✗	75.46/62.26	81.02/75.73	73.96/58.65		83.41/83.41
		✓	75.82/63.78	81.22/77.25	75.25/61.94		**84.15/84.15**
	p=4, t=2	✗	75.43/63.37	81.18/75.64	74.31/60.54		83.72/83.68
		✓	**76.18/65.08**	**81.57/76.70**	**75.50/62.08**		**84.11/84.13**
	p=4, t=3	✗	75.23/63.11	81.32/76.99	73.99/56.35		-
		✓	75.10/63.09	81.00/76.21	73.92/57.83		-
GPT-Neo 1.3B* (α)	p=2, t=2	✗	71.15/63.59	80.30/77.02	75.36/64.79		-
		✓	**72.73/64.48**	**80.40/78.08**	**76.46/65.92**		-
	p=4, t=2	✗	71.46/62.77	80.86/76.64	74.29/63.52		-
		✓	70.68/64.10	80.60/77.57	74.18/63.77		-
GPT-Neo 2.7B* (α)	p=2, t=2	✗	74.57/62.24	79.49/76.20	76.76/65.70	GPT-Neo 2.7B* (α)	82.97/82.79
		✓	**75.67/66.44**	**80.72/76.96**	**76.27/66.30**		83.65/83.64
	p=4, t=2	✗	75.24/63.55	79.40/75.03	75.77/65.54		83.01/83.00
		✓	75.87/65.61	79.35/76.35	76.41/67.75		83.22/83.21
GPT-J 6B* (α)	p=2, t=2	✗	72.22/62.63	79.31/76.92	75.05/66.58	GPT-J 6B* (α)	82.84/82.78
		✓	**71.63/64.06**	**79.77/77.60**	**75.98/67.15**		83.21/83.19
	p=4, t=2	✗	71.56/61.18	78.00/76.05	74.90/63.33		82.73/82.73
		✓	72.19/64.37	77.95/76.25	74.85/65.93		83.37/83.36
GPT-J 6B* (γ)	p=2, t=2	✗	73.84/64.34	80.94/76.75	76.88/67.73		
		✓	**74.70/65.71**	**81.69/78.01**	78.14/68.53		
	p=4, t=2	✗	72.96/63.33	81.13/77.63	77.28/67.86		
		✓	74.84/65.48	81.34/78.02	**78.67/69.66**		

and t=2 encoder layers, we can see a much bigger performance with the structure labels achieving new baseline scores in ECtHR (A), ECtHR (B), and ILDC datasets. For SCOTUS, this improvement from the baseline is not much. This is because of the high skew of class labels in the test dataset (for example label 5 has only 5 samples). With these results, we fixed certain parameters in MESc for further experiments with the extracted embeddings from GPT-Neo and GPT-J. For them, we ran experiments with t=2 encoders and the last layer (p=1) and gained lesser performance than p=2 (or 4) layers and t=2 encoders, which we exclude in this paper. For ECtHR(A&B) and SCOTUS, concatenating the embeddings from the last two layers of GPT-Neo or GPT-J had a significant impact above their vanilla fine-tuned variants by a minimum margin of 3 points for GPT-Neo-1.3B, and 1 point for GPT-Neo-2.7B and GPT-J. This increases further by a minimum of 1 point when including the approximated structure labels, showing the impact of having structural information. For ILDC, concatenating the last four layers didn't have much improvement in the performance while including the generated structure labels increased the performance.

(b) Encoders fine-tuned on 2048 input length (γ)**:** Referring to Table 4 and Table 5 for the documents we did a comparative study of MESc(on GPT-J 6B* (γ))'s performance with its backbone fine-tuned LLM (GPT-J 6B (γ)) to see the effect of increasing the number of parameters and the input length. GPT-J 6B (γ) fine-tuned on its maximum input length (2048) achieves better (or similar) performance than its MESc overhead trained on its extracted embeddings. For SCOTUS, MESc achieves better performance (2 points, m-F1) in the test set. Almost similar performance (m-F1) in ECtHR(B), 1 point higher (m-F1) in ECtHR(A)'s test set, and lesser in ILDC. To check if this is the case with GPT-Neo-1.3B and GPT-Neo-2.7B we fine-tuned them with their maximum input length (2048) on SCOTUS (which through our experiments can be seen are more difficult to classify). We found that fine-tuning GPT-Neo (1.3B and 2.7B) on its maximum input length didn't show the same results as with the GPT-J. We find that for both GPT-Neo-1.3B(γ) and GPT-Neo-2.7B(γ) even the MESc (GPT-Neo-1.3B*(α)) and MESc (GPT-Neo-2.7B*(α)) performs better ($>$ 1 point m-F1) respectively. To analyze this, we plot the distribution of the number of documents with respect to their chunk counts (chunk length = 2048) in the datasets, one such example of ECtHR can be found in Fig. 3 (we accumulate the document counts for chunks 3 to 24 for clarity). As observed, most of the documents can fit in 1 or 2 chunks (median = 1), which means that with the longer input of 2048, most of the important information is not fragmented during the fine-tuning process (stage 1) and prediction. Along with this, the higher number of parameters in GPT-J helps it adapt better to most of the documents. We observe that most ($>$ 90%) of the documents can fit in very few chunks, deepening the models with extra layers (stages 3 & 4) does not have any added value.

The results obtained are statistically significant[9] [33].

[9] We performed student's t-test (p-value $<$ 0.05).

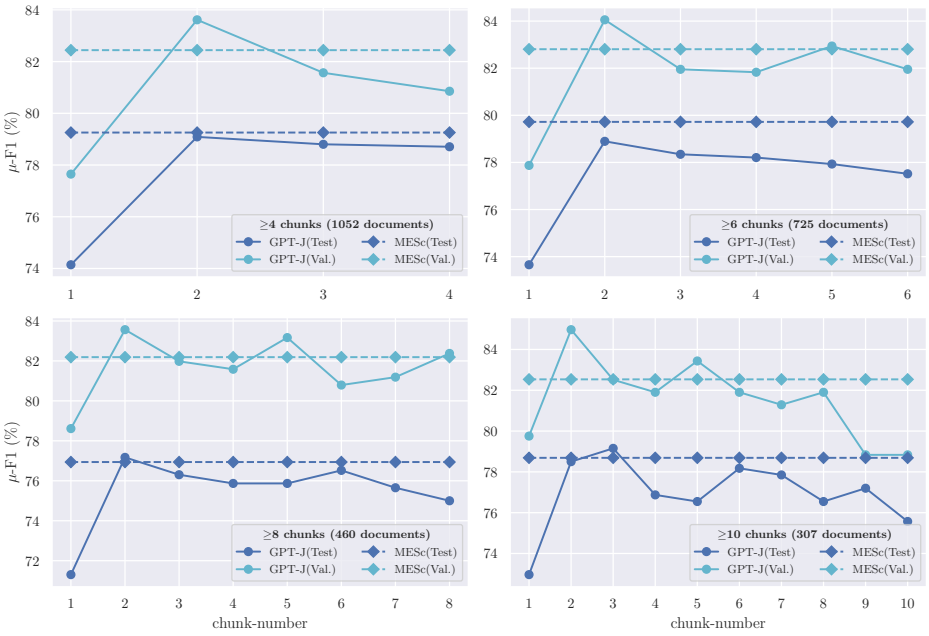

Fig. 4. μ-F1 for chunk-number for GPT-J (γ) vs MESc (GPT-J* (γ)) in SCOTUS on both Validation(Val.) and Test set.

Analysis on long documents: To analyze the performance of MESc and its standalone LLM with the document length we ran experiments with GPT-J (γ) on documents with minimum lengths of 4, 6, 8, and 10 chunks of the documents. The predictions are made using the nth chunk of all the documents. We show these results from the SCOTUS dataset in Fig. 4, where, for the input chunk of the documents, we plot its corresponding μ-F1 score. Since MESc has no input length limit and takes all the chunks at once we plot its prediction for all chunks considered as constant lines. The performance with GPT-J (γ) fluctuates with the input chunk, with the worst performance when using the first chunk and the best on the second/third chunk. This shows that for these documents in the test set the second/third chunk has a higher probability of containing the important information for a more robust prediction. Choosing which chunk to use for an unseen test set becomes more difficult as the document length increases and there is no prior information on its important parts. The fluctuations become worse for documents with a minimum of 10 chunks. While for MESc the performance is overall better than GPT-J (γ) in all the lengths considered. This shows that the hierarchical framework (such as MESc) is more reliable than its LLM counterpart on longer documents and when the important parts of the document are unknown.

With these results on MESc, we find that:

1. Concatenating embeddings from the last two layers in GPT-Neo (1.3B, 2.7B) or GPT-J or, the last 4 layers in BERT-based models, provides the optimum number of feature variances. Globally, concatenating the embeddings helped to get a better approximation of the structure labels and improved performance.

2. MESc works better than its counterpart LLM under the condition that the length of most of the documents in the dataset is much greater than the maximum input length of the LLM.

3. For long documents when their important parts are unknown MESc performs better than its counterpart LLMs.

5 Conclusion

We explore the problem of classification of large and unstructured legal documents and develop a multi-stage hierarchical classification framework (MESc). We test the effect of including our approximated structure and the impact of combining the embeddings from the last layers of a fine-tuned transformer-based LLM in MESc. Along with BERT-based LLMs, we explored the adaptability of LLMs with billion parameters (GPT-Neo and GPT-J) to MESc and analyzed its limits (section 4) with these LLMs suggesting the optimal condition for its performance. The benchmark performance of GPT-Neo and GPT-J on the legal cases from India and Europe shows the intra-domain(legal) transfer learning capability of these billion-parameter language models. Most of all, our experiments achieve a new baseline in the classification of the ILDC and the LexGLUE subset (ECtHR (A), ECtHR (B), and SCOTUS). In our future work, we aim to analyze the clusters and how they contribute to the prediction. We aim to develop an explanation algorithm to explain the predictions while also leveraging this work in-domain, on the French and European legal cases to further explore the problem of length and the non-uniform structure.

6 Ethical Considerations

This work conforms to the ethical consideration of the datasets (ILDC [20] and LexGLUE [9])) used here. The framework developed here is in no way to create a "robotic" judge or replace one in real life, but rather to help analyze how LLMs and our hierarchical framework can be applied to long legal documents to predict judgments. These methods are not foolproof to predict judgments, and should not be used for the same in real-life settings (courts) or used to guide people unfamiliar with legal proceedings. The results from our framework are not reliable enough to be used by a non-professional to make high-stakes decisions in one's life concerning legal cases.

Acknowledgements

This work is supported by the LAWBOT project (ANR-20-CE38-0013) and was performed using HPC resources from GENCI-IDRIS (Grant 2022-AD011013937).

References

1. Ainslie, J., Ontanon, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., Yang, L.: ETC: Encoding long and structured inputs in transformers. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 268–284. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.19, https://aclanthology.org/2020.emnlp-main.19
2. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer (2020). https://doi.org/10.48550/ARXIV.2004.05150, https://arxiv.org/abs/2004.05150
3. Black, S., Leo, G., Wang, P., Leahy, C., Biderman, S.: GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow (Mar 2021). https://doi.org/10.5281/zenodo.5297715, https://doi.org/10.5281/zenodo.5297715
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
5. Chalkidis, I., Androutsopoulos, I., Aletras, N.: Neural legal judgment prediction in English. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 4317–4323. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1424, https://aclanthology.org/P19-1424
6. Chalkidis, I., Dai, X., Fergadiotis, M., Malakasiotis, P., Elliott, D.: An exploration of hierarchical attention transformers for efficient long document classification (2022), https://arxiv.org/abs/2210.05529
7. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., Androutsopoulos, I.: LEGAL-BERT: The muppets straight out of law school. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 2898–2904. Association for Computational Linguistics, Online (Nov 2020), https://aclanthology.org/2020.findings-emnlp.261
8. Chalkidis, I., Fergadiotis, M., Tsarapatsanis, D., Aletras, N., Androutsopoulos, I., Malakasiotis, P.: Paragraph-level rationale extraction through regularization: A case study on European court of human rights cases. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 226–241. Association for Computational Linguistics, Online (Jun 2021). https://doi.org/10.18653/v1/2021.naacl-main.22, https://aclanthology.org/2021.naacl-main.22
9. Chalkidis, I., Jana, A., Hartung, D., Bommarito, M., Androutsopoulos, I., Katz, D., Aletras, N.: LexGLUE: A benchmark dataset for legal language understanding in English. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 4310–4330. Association for Computational Linguistics, Dublin, Ireland (May 2022), https://aclanthology.org/2022.acl-long.297

10. Chen, H., Cai, D., Dai, W., Dai, Z., Ding, Y.: Charge-based prison term prediction with deep gating network. 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). pp. 6362–6367. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1667, https://aclanthology.org/D19-1667
11. Condevaux, C., Harispe, S.: Lsg attention: Extrapolation of pretrained transformers to long sequences. In: Kashima, H., Ide, T., Peng, W.C. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 443–454. Springer Nature Switzerland, Cham (2023)
12. Cui, J., Shen, X., Nie, F., Wang, Z., Wang, J., Chen, Y.: A survey on legal judgment prediction: Datasets, metrics, models and challenges. ArXiv **abs/2204.04859** (2022)
13. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423, https://doi.org/10.18653/v1/n19-1423
14. Feng, Y., Li, C., Ng, V.: Legal judgment prediction: A survey of the state of the art. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22. pp. 5461–5469. International Joint Conferences on Artificial Intelligence Organization (7 2022). https://doi.org/10.24963/ijcai.2022/765, https://doi.org/10.24963/ijcai.2022/765, survey Track
15. Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., Leahy, C.: The pile: An 800gb dataset of diverse text for language modeling. CoRR **abs/2101.00027** (2021), https://arxiv.org/abs/2101.00027
16. Jawahar, G., Sagot, B., Seddah, D.: What does BERT learn about the structure of language? In: Korhonen, A., Traum, D., Màrquez, L. (eds.) Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3651–3657. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1356, https://aclanthology.org/P19-1356
17. Katju, J.M.: Backlog of cases crippling judiciary (2019), https://www.tribuneindia.com/news/archive/comment/backlog-of-cases-crippling-judiciary-776503
18. Kaufman, A.R., Kraft, P., Sen, M.: Improving supreme court forecasting using boosted decision trees. Political Analysis **27**(3), 381–387 (2019). https://doi.org/10.1017/pan.2018.59
19. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: The efficient transformer. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=rkgNKkHtvB
20. Malik, V., Sanjay, R., Nigam, S.K., Ghosh, K., Guha, S.K., Bhattacharya, A., Modi, A.: ILDC for CJPE: indian legal documents corpus for court judgment-prediction and explanation. CoRR **abs/2105.13562** (2021), https://arxiv.org/abs/2105.13562

21. McInnes, L., Healy, J., Astels, S.: hdbscan: Hierarchical density based clustering. Journal of Open Source Software **2**(11), 205 (2017). https://doi.org/10.21105/joss.00205, https://doi.org/10.21105/joss.00205
22. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2018). https://doi.org/10.48550/ARXIV.1802.03426
23. Nallapati, R., Manning, C.D.: Legal docket classification: Where machine learning stumbles. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. pp. 438–446. Association for Computational Linguistics, Honolulu, Hawaii (Oct 2008), https://aclanthology.org/D08-1046
24. Paul, S., Mandal, A., Goyal, P., Ghosh, S.: Pre-training transformers on indian legal text (2022). https://doi.org/10.48550/ARXIV.2209.06049
25. Prasad, N., Boughanem, M., Dkaki, T.: Effect of hierarchical domain-specific language models and attention in the classification of decisions for legal cases. In: Proceedings of the 2nd Joint Conference of the Information Retrieval Communities in Europe (CIRCLE 2022), Samatan, Gers, France, July 4-7, 2022. CEUR Workshop Proceedings, vol. 3178. CEUR-WS.org (2022), http://ceur-ws.org/Vol-3178/CIRCLE_2022_paper_21.pdf
26. Song, Y., Wang, J., Liang, Z., Liu, Z., Jiang, T.: Utilizing BERT intermediate layers for aspect based sentiment analysis and natural language inference. CoRR **abs/2002.04815** (2020), https://arxiv.org/abs/2002.04815
27. Thoppilan, R., Freitas, D.D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.T., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H.S., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhao, V., Zhou, Y., Chang, C.C., Krivokon, I., Rusch, W., Pickett, M., Srinivasan, P., Man, L., Meier-Hellstern, K., Morris, M.R., Doshi, T., Santos, R.D., Duke, T., Soraker, J., Zevenbergen, B., Prabhakaran, V., Diaz, M., Hutchinson, B., Olson, K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., Aguera-Arcas, B., Cui, C., Croak, M., Chi, E., Le, Q.: Lamda: Language models for dialog applications (2022)
28. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models (2023)
29. Trautmann, D., Petrova, A., Schilder, F.: Legal prompt engineering for multilingual legal judgement prediction. CoRR **abs/2212.02199** (2022). https://doi.org/10.48550/arXiv.2212.02199, https://doi.org/10.48550/arXiv.2212.02199
30. Tuggener, D., von Däniken, P., Peetz, T., Cieliebak, M.: LEDGAR: A large-scale multi-label corpus for text classification of legal provisions in contracts. In: Proceedings of the Twelfth Language Resources and Evaluation Conference. pp. 1235–1241. European Language Resources Association, Marseille, France (May 2020), https://aclanthology.org/2020.lrec-1.155
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762
32. Wang, B., Komatsuzaki, A.: GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax (May 2021)

33. Welch, B.L.: The generalization of 'student's' problem when several different population variances are involved. Biometrika **34**(1/2), 28–35 (Jan 1947), http://www.jstor.org/stable/2332510
34. Xiao, C., Zhong, H., Guo, Z., Tu, C., Liu, Z., Sun, M., Feng, Y., Han, X., Hu, Z., Wang, H., Xu, J.: CAIL2018: A large-scale legal dataset for judgment prediction. CoRR **abs/1807.02478** (2018), http://arxiv.org/abs/1807.02478
35. Xu, N., Wang, P., Chen, L., Pan, L., Wang, X., Zhao, J.: Distinguish confusing law articles for legal judgment prediction. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 3086–3095. Association for Computational Linguistics, Online (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.280, https://aclanthology.org/2020.acl-main.280
36. Yang, J., Zhao, H.: Deepening hidden representations from pre-trained language models for natural language understanding. CoRR **abs/1911.01940** (2019), http://arxiv.org/abs/1911.01940
37. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489. Association for Computational Linguistics, San Diego, California (Jun 2016). https://doi.org/10.18653/v1/N16-1174, https://aclanthology.org/N16-1174
38. Yu, F., Quartey, L., Schilder, F.: Legal prompting: Teaching a language model to think like a lawyer (2022)
39. Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A.: Big bird: Transformers for longer sequences. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 17283–17297. Curran Associates, Inc. (2020)
40. Zhang, X., Wei, F., Zhou, M.: HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5059–5069. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1499, https://aclanthology.org/P19-1499
41. Zheng, L., Guha, N., Anderson, B.R., Henderson, P., Ho, D.E.: When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law. p. 159–168. ICAIL '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3462757.3466088, https://doi.org/10.1145/3462757.3466088
42. Zhong, H., Guo, Z., Tu, C., Xiao, C., Liu, Z., Sun, M.: Legal judgment prediction via topological learning. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 3540–3549. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018). https://doi.org/10.18653/v1/D18-1390, https://aclanthology.org/D18-1390
43. Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., Sun, M.: How does NLP benefit legal system: A summary of legal artificial intelligence. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5218–5230. Association for Computational Linguistics, Online (Jul 2020), https://aclanthology.org/2020.acl-main.466