Proceedings of LATERAISSE Workshop @LREC2022, pages 42–50
Marseille, 25 June 2022

An Open Source Contractual Language Understanding Application Using Machine Learning

Afra Nawar*, Mohammed Rakib*, Salma Abdul Hai*, Sanaulla Haq*
North South University
Dhaka-1229, Bangladesh
{afra.nawar05, mohammed.rakib, salma.hai,sanaulla.haq}@northsouth.edu

Abstract

Legal field is characterized by its exclusivity and non-transparency. Despite the frequency and relevance of legal dealings, legal documents like contracts remains elusive to non-legal professionals for the copious usage of legal jargon. There has been little advancement in making legal contracts more comprehensible. This paper presents how Machine Learning (ML) and Natural Language Processing (NLP) can be applied to solve this problem, further considering the challenges of applying ML to the high length of contract documents and training in a low resource environment. The largest open-source contract dataset so far, the Contract Understanding Atticus Dataset (CUAD) is utilized. Various pre-processing experiments and hyperparameter tuning have been carried out and we successfully managed to eclipse SOTA results presented for models in the CUAD dataset trained on RoBERTa-base. Our model, A-type-RoBERTa-base achieved an AUPR score of 46.6% compared to 42.6% on the original RoBERT-base. This model is utilized in our end to end contract understanding application which is able to take a contract and highlight the clauses a user is looking to find along with it's descriptions to aid due diligence before signing. Alongside digital, i.e. searchable, contracts the system is capable of processing scanned, i.e. non-searchable, contracts using tesseract OCR. This application is aimed to not only make contract review a comprehensible process to non-legal professionals, but also to help lawyers and attorneys more efficiently review contracts.

Keywords: Contract Review, Machine Learning, CUAD

1. Introduction

As transactions and interpersonal or business relationships are legalized to an extent greater than ever in precedence, contracts have become one of the most widely utilized legal documents today, enabling legal repercussions, and thus, informed agreement is critical. For legal professionals, who work with numerous clients and documents, the contract review process is a routine and time-consuming task making it is easy to overlook crucial information. For individuals without legal knowledge and unable to attain legal services, the process is esoteric.

Despite the advances of machine learning (ML) and natural language processing (NLP), applied technology in the legal industry which addresses these issues are scarce. Majority of the legal documents, such as judgment papers and contracts are in text format, some even hundreds of pages in length, and difficult to review quickly and accurately. (Hegel et al., 2021). Legal AI is important in the legal industry because it helps save time and effort by reducing the amount of work lawyers have to perform (Dabass and Dabass, 2018). In this paper we apply Machine Learning to contract agreements in order to simplify contract understanding process for the average person and attorneys. As shown in Fig 1, our application allows users to input a contract, and the model provides a labelled contract with the types of clauses recognized, assisting users in making educated legal decisions in a matter of minutes.

*All authors contributed equally to this work

2. Literature Review

Legal judgment prediction, legal entity recognition, document classification, legal question answering, and legal summarization are some of the tasks which have been explored using Machine learning and NLP. Legal Artificial Intelligence (LegalAI) is a branch of artificial intelligence that focuses on assisting lawyers with legal duties.

Authors in (Zhong et al., 2020) demonstrate numerous embedding- and symbol-based approaches and discuss LegalAI's future path. They have gone through three common applications in detail, including judgment prediction, similar case matching, and legal question answering, to show why these two types of techniques are critical to LegalAI. Malik et al. (Malik et al., 2021) introduce the INDIAN LEGAL DOCUMENTS CORPUS (ILDC), a collection of Supreme Court of India case processes (SCI). Their best prediction model presents a 78% accuracy compared to 94% for human legal experts.

In another study (Holzenberger et al., 2020), the authors present the Statutory Reasoning Assessment dataset (SARA), which consists of a collection of rules taken from the US Internal Revenue Code (IRC) laws, as well as a set of natural language questions that can only be answered properly by referring to the rules. They have offered a legal statutes resource, a collection of hand-curated natural language rules and cases, and a symbolic solver capable of representing these rules and solving the challenge task. This study is intended

to be a contribution to legal-domain natural language processing, in addition to the fascinating challenge provided by statutory reasoning.

In (Roegiest et al., 2018) the authors propose different models for the due diligence problem to find specific clauses in legal contractual documents and quantify the risk associated with each. They also introduce a new dataset, a subset of their production dataset, with 15 million sentences in 4200 contracts. This goal is quite similar to ours, however they did not approach the problem with Deep Learning, rather with linear classifiers, Conditional Random Fields (CRF), hybrid models of SVM and Hidden Markov Models. The best and most reliable result was achieved through CRF.

In (Hendrycks et al., 2021) they've compiled a high-quality dataset of annotated contracts to aid contract analysis research and to learn more about how well NLP models work in highly specialized domains. Over 13,000 annotations by legal experts are included in CUAD through 41 labels. On CUAD, we tested ten pre-trained language models and discovered that their performance is promising, but there is still much space for improvement. They have also discovered that data is a major bottleneck, as reducing data by an order of magnitude drastically reduces efficiency, emphasizing the importance of CUAD's large number of annotations. They also discovered that model design has a significant impact on efficiency, implying that algorithmic advances from the NLP group would aid in resolving this issue. They concluded that the CUAD has the potential to speed up research into a major real-world issue while also acting as a benchmark for evaluating NLP models on specialized domains in general. The authors in (Leivaditi et al., 2020) provide another dataset for contract review, however, with fewer categories and annotations than CUAD.

One recent work done in this field, (Hegel et al., 2021), shows that the visual cues like layout, style, and placement of text in a document are significant elements that are important to obtaining an acceptable degree of accuracy on long documents.

3. METHODOLOGY

The system diagram, delineating the work flow of our application, is shown in Fig-1. The dataset is first split into three parts: training, validation and testing. Next, the text of each of the parts is converted to tokens and then the tokens are embedded to tensors. After this, the dataset is trained using pretrained transformer models and then evaluated. Subsequently, the best performing model is selected and quantized in order to deploy the model with the backend. Finally, the user can do inference on any legal contract. Each of the blocks below will be briefly explained in this section, along with their significance.

3.1. CUAD Dataset

We are using Contract Understanding Atticus Dataset (CUAD) for training and evaluating our model. It contains 510 contracts with 13101 labeled clauses. There are 25 different types of contracts with varying lengths ranging from a few pages to over one hundred pages. But, most parts of a contract should not be highlighted. Labeled clauses make up about 10% of each contract on average. Since there are 41 label categories, this means that on average, only about 0.25% of each contract is highlighted for each label. We have divided the dataset into two parts — 80% for training and 20% for testing. In terms of feeding the data to our model, there are 22450 training samples and 4182 test samples. Each sample has four keys: 'id', 'title', 'context', 'question', 'answers'. The 'answers' key has two parts: the answer itself as text and the starting index of the answer in the context. Our model will predict the starting index and text of the answer after completion of its training.

3.2. Tokenizing Inputs and Embedding Tokens

Before feeding the texts of our document to any model, we need to preprocess them. This is done by the Transformer Tokenizer (Wolf et al., 2019). Each model has its own tokenizer, which converts input text to tokens, including converting the tokens to their corresponding IDs in the vocabulary and put it in a format the model expects, as well as generate the other inputs that the model requires. Since we will be using pre-trained models, we will utilize the vocabulary and tokenizer used while pre-training these models. The next step in the workflow is to breathe meaning to the tokens so that the model can understand the relationship among the tokens and make sense out of them. This is done by converting each token into a vector representation of numbers called a tensor. But before embedding the tokens into tensors, the tokens for very long documents are handled by truncating the context to the max length that our model can fit. Moreover, in order to account for the case in which the answer lies at the point we split a long context, we allow some overlap between the features.

3.3. Training and Evaluating Pretrained Transformer Models

Pretrained models are models that have already undergone extensive training on massive datasets, which are then applied to downstream tasks through fine-tuning. We are using pre-trained transformer models since these significantly improve results, compared to training from scratch, for many NLP tasks like Question Answering, Machine Translation, Named Entity Recognition, etc. We aim to fine-tune pre-trained transformer models by retraining them for Question Answering Task since our CUAD dataset is based on it. Besides, we perform various experiments by trying different combinations of hyperparameters to evaluate and improve the performance of these models, which are broadly explained in the experiment and results section.

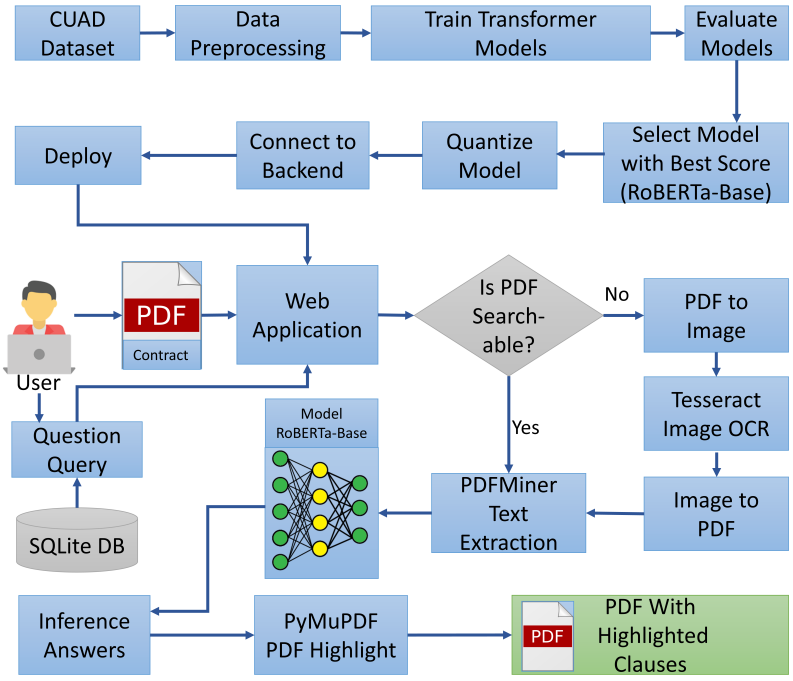

Figure 1: System Diagram of Our Project.

3.4. Quantization

Quantization refers to a method of computing and storing tensors with smaller bitwidths instead of floating point precision to enable deep learning models to run quicker and use less memory. There are 3 types of quantization: 1) Dynamic Quantization 2) Static Quantization 3) Quantization-Aware Training. We implemented dynamic quantization which quantizes the weights beforehand whereas the activations are quantized at runtime (Peng et al., 2007). We have used Dynamic quantization as it is the most simple one of the three and can be applied on-the-fly without requiring to retrain the model.

3.5. Application

3.5.1. Backend

In order to make the trained model useful in the real world an end to end system was implemented wherein users may upload their contracts and perform clause-wise inference on the contracts to allow them to easily and quickly evaluate the contract and perform due diligence.

The web application returns inferences from our trained model underneath the user interface. The user may upload their contract and select out of the 41 clauses they wish to quickly identify, starting the backend inference process. When the user uploads their PDF it goes straight to our text extraction module with PDFMiner Library. It extracts text from PDF documents with 97% accuracy calculated using Levenshtein Distance compared to the extracted texts in the test set of original CUAD dataset. If, however, the PDFS are scanned, i.e. non searchable, they are passed to OCR PDF conversion function to generate a searchable PDF. This PDF is passed to the PDFMiner module like before, the consequent processes being identical.

The extracted text is passed as the context along with the selected clause or question query to the trained Roberta-Base model as features. It performs the inference and returns it to our Highlighting module which calculates the rectangular positions of the answers within the user's PDF and draws highlights around it. The app is reloaded to show the user the answer.[1]

3.5.2. Searchable PDF Conversion

In order for the model to deliver an inference, the extracted texts from the input contract must be fed into it. To extract the contract's contents, we utilized PDFminer. However, this approach only works with native/searchable PDF files, not scanned/non-searchable ones. We tested Tesseract (Tesseract-Ocr,) and EasyOCR (JaidedAI,) and compared their accuracy to guarantee that the texts are accurat5ely retrieved from scanned files. For Tesseract to perform well, it is essential that the quality of the images are enhanced before it is input into the OCR. To provide a general solution, image preprocessing techniques were used to eliminate any distorted or potentially poor images. As shown in the Fig-2 below, converting the image to grayscale, binarization, noise removal using di-

[1]The web application and model are available at github.com/afra-tech/defactolaw

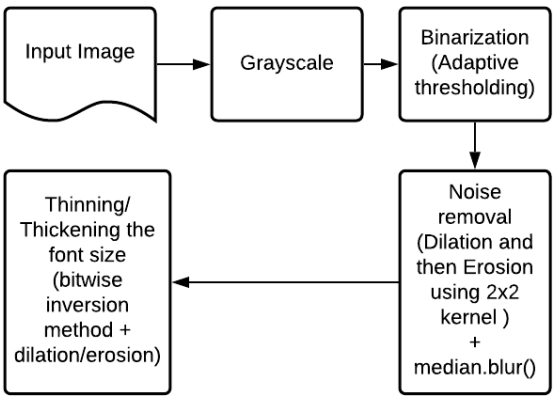

Figure 2: Flowchart of Image Preprocessing.

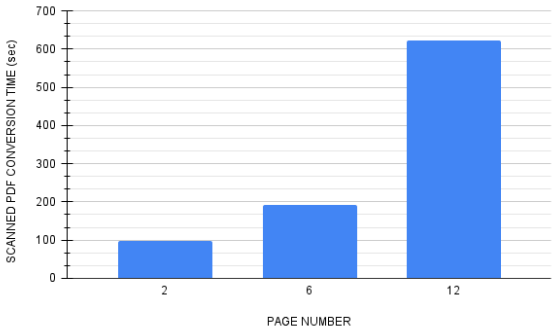

Figure 3: Scanned to Searchable PDF Conversion Time.

lation and erosion, median blur, and thinning or thickening the fonts using dilation, erosion, and bitwise invert methods were all part of the preprocessing.

The adaptive threshold was chosen for the experiment because it is considerably better for situations involving text extraction. The block size, which is one of the parameters in adaptive threshold, determines the size of the neighbourhood area and C is a constant that is subtracted from the mean or weighted sum of the neighbourhood pixels. To determine these two parameters has been a challenge. Since, low noise (Higher value of C) resulted in fainted texts and higher noise (Low value of C) resulted in illegible images. Because only certain noise is required for proper text readability, it was necessary to eliminate it using morphological operations such as dilation and erosion. In order to remove pixels that do not correspond to text that are still surrounding text items or perhaps the noise, the binarized images were first dilated and then eroded. In the dilation, A 2x2 kernel or matrix was created, and the entire image was convolutioned. Dilation increases the amount of whitespace in the image, which reduces noise and tiny dark areas. Erosion works in a similar way, except it increases the darkness of the letters and makes them easier to read. For erosion, the same kernel size was utilized. The image was slightly blurred after the two processes and the overall salt and pepper noise in the image was eliminated as a result of this. For thinning/thickening the font, the image was initially inverted to make dilation and erosion make sense. The background is now black, but the text is white. Now, erosion with a kernel size of 2x2 was employed to make the typefaces narrower. The same technique was used to thicken the font size, only the erosion process was substituted with dilation with the same kernel size.

However, the extracted texts' accuracy was not sufficient (less than 50%). Various issues in the input data, such as different layouts, skewness, and typefaces, hindered successful recognition. In comparison to Tesseract, EasyOCR did slightly better at extracting texts. Using EasyOCR speeds up the process since Tesseract requires pre-processing of non-scanned images to make them seem like scanned images in order to function well, which increases the overall inference time. However, the accuracy of the retrieved texts can not always be guaranteed. The overall application time is increased since extracting texts from scanned PDF takes time and converting the scanned PDF to a searchable PDF for highlighting the text clauses also takes some time. As a result, a general approach was offered, which is described below.

We have tested the use of tesseract output HOCR. HOCR is an open data representation standard for structured text generated by optical character recognition (OCR). Because the text, style, layout information, recognition confidence metrics, and other data are encoded using Extensible Markup Language (XML) in the form of Hypertext Markup Language (HTML) or XHTML (Breuel, 2007), this worked perfectly for our system. The scanned PDF is converted to searchable PDF using tesseract output in HOCR and storing the result as PDF. This is now used for text extraction by applying the corresponding implementation of extracting texts from searchable PDF as discussed before. The retrieved texts using this technique have an accuracy that ranges from 93.99% to 99.09%. Fig-3 illustrates the time it takes to convert a scanned PDF to searchable PDF vs. the amount of pages it contains. As it can be observed, the time grows exponentially as the PDF's content increases. This is due to the internal process of converting the PDF to images, and subsequently from images to HOCR and lastly to a searchable PDF.

4. Experiments and Results

4.1. Transformer Models

4.1.1. Evaluation Metric

We have used four evaluation metrics which are: Exact Match, F1 Score, Area under Precision Recall Curve (AUPR) and Precision @80% Recall. In the dataset's paper (Hendrycks et al., 2021), Precision @90% Recall is only non-zero for DeBERTa x-large, which we

have not trained due to insufficient resources, hence the metric is not evaluated.

1. **Exact Match:** This metric is as simple as it sounds. For each question-answer pair, EM=1 if the characters of the model's prediction exactly match the characters of the True Answer, otherwise EM=0.

2. **F1 Score:** F1 score is a typical metric for classification problems and is generally utilized in QA. The number of shared words between the prediction and the truth is the basis of the F1 score: precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth. We will be evaluating F1 Scores of questions that have answers since this tells us how accurately our model could highlight the desired labels in the contract. F 1 Score =2 $\times \frac{Precision \times Recall}{Precision + Recall}$

3. **Area Under Precision Recall Curve (AUPR):** A precision-recall curve shows the relationship between precision (= positive predictive value) and recall (= sensitivity) for every possible selected cut-off. The area under this curve is called AUPR. It represents the average precision of a model. The more the area the better the model. AUPR is suitable for imbalanced datasets as this metric is not affected by a large no. of negative or positive examples compared to their counterparts.

4. **Precision at 80% Recall:** As our main goal is to review legal contracts, it is of huge importance that our model does not misclassify any positive labels (important parts which are required to be highlighted). For this reason, a high recall (no. of positives predicted correctly out of all the actual positives in the dataset) is necessary and so we fix our recall at 80% threshold and then measure precision which is a good indicator of how well our model can review contracts.

4.1.2. Major Contributions in Improving Performance of Models

1. **Training Large No. of Models:** We have trained a total of 75 models of different types using various hyperparameters to find out the best model. Fig-4 shows the performance (Exact-Match Score) of all the 75 models trained during this process. Observing the figure we see that there are a lot of models with very good, very bad as well as average performances. So we set a baseline score of 60% shown by the red line in Fig-4. So, in this paper we will broadly explain only the models that have an Exact Match score of 60% or more.

2. **Balancing Features:** A contract, on average, has 10% of labeled clauses. So, after converting to features, more than 99% of them do not contain any of the 41 relevant labels. To mitigate this imbalance, we drop a significant portion of the features that do not contain any relevant labels so that features are approximately balanced between having highlighted clauses and not having any highlighted clauses. As a result, we see a significant improvement in training times and performance gains as there is a balance between highlighted and unhighlighted parts. To be more specific, observing Fig-5 we can see that training time was reduced by 48 times and performance of models also increased by 1.5 times. So balancing features played a crucial role in training the models in a resource-efficient manner as well as improving performance scores.

4.1.3. Optimal Hyper-parameters

The optimal hyperparameters for all our experiments are shown in Table-1. We have tried numerous combinations of various hyperparameters as shown in Table-2 and observed that the hyper-parameter values in Table-1 give the best results. It is important to note that all the optimal values are based on a single GPU with 12GB or 16GB VRAM. Besides, all our experiments were conducted in a resource constrained environment.

Table 1: Optimal Hyperparameters

Hyperparameter	Default Value
Learning Rate	3x10-5
Batch Size	16
Epochs	4
Weight Decay	0.01
Gradient Accumulation Step	2
Eval Accumulation Steps	1
Max Length	384
Doc Stride	128

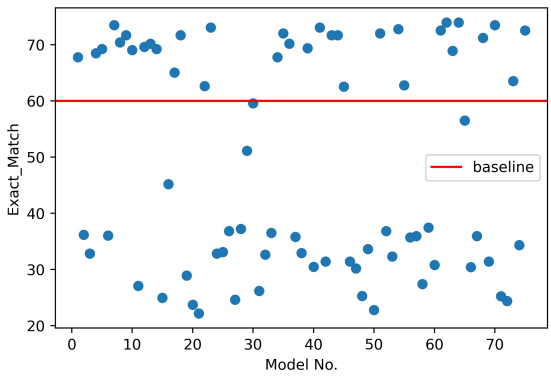

Figure 4: Scatter plot of Exact Match Performances of All Models Trained

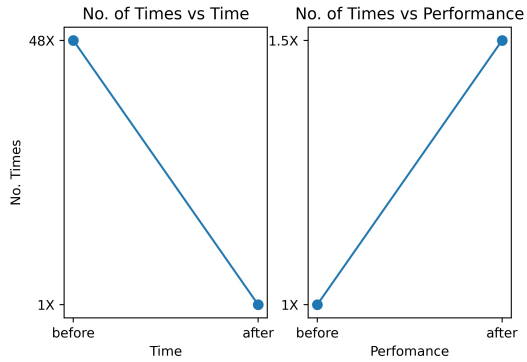

Figure 5: Before-After Plot of Performance and Training Time for Balancing Features.

Table 2: Range of Hyperparameters Tested to Find Optimal Hyperparameters

Hyperparameter	Range of Values
Learning Rate	1x10-4, 3x10-4, 3x10-3, 3x10-5, 3x10-7
Batch Size	2, 4, 8, 16, 24
Epochs	2, 4, 8
Weight Decay	0.01
Gradient Accumulation Step	1, 2, 4, 8, 16, 20
Eval Accumulation Steps	1, 2, 4, 8, 16, 20
Max Length	256, 384, 512
Doc Stride	128, 256

4.1.4. Original Models

Original models are the ones provided by the owners of the CUAD dataset. They have performed grid search to find out optimum parameters and then trained these models on their dataset. Table-3 shows the performance of these models and Table-4 shows the optimum hyperparameters. They have used multiple GPUs in parallel for achieving such results with no limit on computational power.

Table 3: Original CUAD Models

Model Name	AUPR	Precision at 80% Recall	Exact Match	F1 Score
albert-xlarge	37.8	20.5	-	-
roberta-base	42.6	31.1	73.5	81.8
roberta-large	48.2	38.1	74.0	84.8

4.1.5. RoBERTa

We have selected RoBERTa (Liu et al., 2019) since it is an improved version of BERT and the base version has suitable parameters for our computationally restricted environment. The RoBERTa models performed best on both AUPR scores and Exact Match scores. Since we will be reviewing contracts, the AUPR score and Precision at 80% recall is quite important. Comparing RoBERTa with other models like Longformer and ALBERT in Fig-6 we see that no other transformer type comes close to RoBERTa in terms of AUPR. Observing Table-3 and Table-5, we can see that, the B-type-roberta-base (AUPR-46.8) has eclipsed the AUPR score of the original SOTA RoBERTa base (AUPR-42.6) by a healthy margin of 4.2% which is commendable. This was possible due to the proper tuning and optimal selection of hyperparameters from various combinations. The original SOTA score for precision at 80% recall is 31.1% for RoBERTa-base and we have managed to get a score of 29.6% which is also quite good. If we compare these scores to the original SOTA RoBERTa-large, we see that the RoBERTa-large has an AUPR of 48.2% which is 1.4% higher than our RoBERTa base. But if we check the Precision at 80% Recall scores then our best RoBERTa-base model is behind by a lot. The RoBERTa-large model has a Precision at 80% Recall score of 38.1% whereas our best model has a score of 29.6%. Now, if we come to the Exact Match scores, we see that the SOTA RoBERTa-base model has an exact match score of 73.5% whereas our best model (roberta-base-squad2) has an exact match score of 74% which is a minor improvement. Now, in order to get these results, we had to specifically tune the model to get best results at that particular metric. This resulted in multiple models each giving high scores at a particular metric. For example: best AUPR score of 46.8% was given by B-type-roberta-base model, best Exact Match score of 74% was given by roberta-base-squad2 model and best Precision at 80% Recall of 29.6% was given by A-type-roberta-base model. However, if we were to select a model which gives decent scores in all of the criteria then roberta-base-squad2-nq model should be selected. This model has an AUPR score of 43.4%, Precision at 80% Recall score of 28.1% and Exact Match score of 70.12%

4.1.6. ALBERT

We tried our dataset on Albert (Lan et al., 2020) since it has a good record in question-answering tasks and performs pretty well in the SQUAD-v2 dataset outperforming BERT. But unfortunately, we see that the results are not quite satisfactory. We have tried various ALBERT models and the best performing one among them is ALBERT-xlarge-v2. It has an AUPR of 37.5%, Precision at 80% Recall of 25.2% and Exact Match of 72% as shown in Table-5. Out of all, only the Exact Match score is close to the SOTA model. The rest are quite far off. So, we stopped further pursuing this model due to poor performance and resource constraints.

Table 4: Hyperparameters of Models Used in Experiments

Hyperparameter / Model	Learning Rate	Batch Size	Epochs	Decay	Gradient Accumulation Step	Eval Accumulation Steps	Max Length	Doc Stride
DistilRoBERTa-base	3x10-5	4	4	0.1	8	8	384	128
Longformer-base-squad2								
ALBERT-xlarge-v2								
RoBERTa-base-squad2								
RoBERTa-base-squad2-nq								
A-typeRoBERTa-base-squad2-nq	3x10-5	16	4	0.1	2	1	384	128
A-type-RoBERTa-base								
B-type-RoBERTa-base	3x10-5	24	4	0.1	1	1	384	128

Table 5: Performance of All Models

Model Names	AUPR	Precision @ 80% Recall	Exact_Match	F1 Score
DistilRoBERTa-base	35.9	18.6	67.8	79
Longformer-base-squad2	36.4	21.3	73.1	84.3
ALBERT-xlarge-v2	37.5	25.2	72	83.6
RoBERTa-base-squad2	41.4	22.3	74	84.5
A-type-RoBERTa-base-squad2-nq	42.7	27.3	72.7	83
RoBERTa-base-squad2-nq	43.4	28.1	70.12	80.9
A-type-RoBERTa-base	46.6	29.6	65	74.4
B-type-RoBERTa-base	46.8	25	59.6	68.2

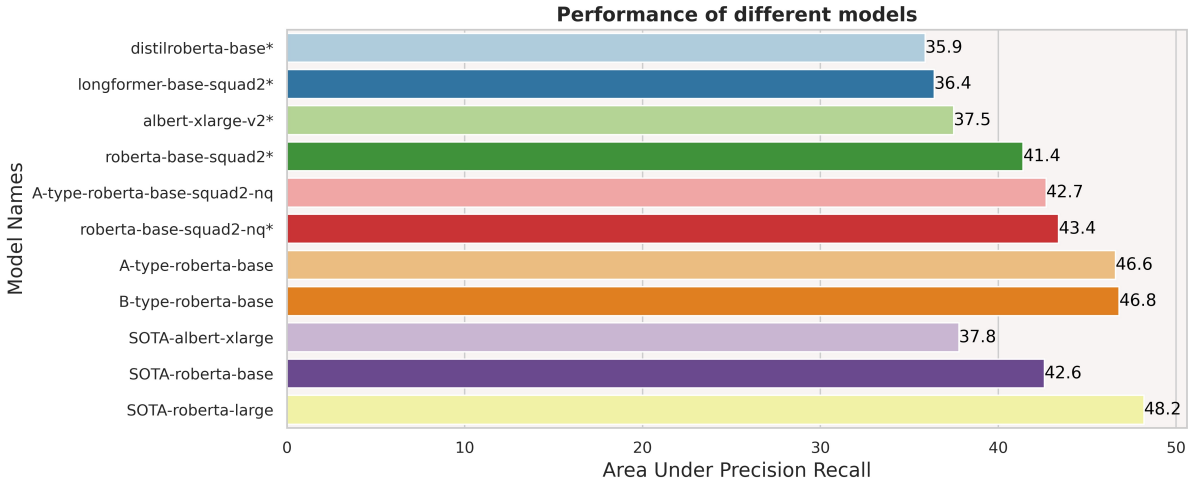

Figure 6: Comparison of AUPR Scores of All Models

4.1.7. Longformer

Longformer (Beltagy et al., 2020) is the extended version of the RoBERTa specifically trained on long documents from where the checkpoint of the RoBERTa model ends. Since legal documents can be very long and our dataset has documents that consist of more than 100 pages, we decided to use the Longformer model. The results from the Longformer model is also not upto the mark. We experimented with various longformer models and are reporting only the best one among them. The best longformer with our default hyperparameters is longformer-base that has been fine tuned on squad-v2 and then trained on our dataset. It has an AUPR score of 36.4%, Precision at 80% Recall of 21.3% and Exact Match score of 73.1% as shown in Table-5. This model has a noteworthy Exact Match score since it is only 0.4% behind the original RoBERTa base model. The results of the other metrics are very poor.

4.2. Distil-RoBERTa-base

Knowledge distillation is the process of transferring knowledge from a large model to a smaller model without loss of validity. Distil-RoBERTa-base is a model where the knowledge of RoBERTa-base was trans-

Table 6: Quantization Results

Roberta Base	Exact Match	F1 Score	Size
Before Quantization	65	74.4	500MB
After Quantization	41.4	44.7	240MB

ferred to it while reducing the number of trainable parameters and size. So, for our case the model size was reduced by about 160MB and parameters decreased by 28M compared to the regular RoBERTa-base model. The performance of this model with respect to its reduced size and parameters is good but not sufficient enough. It has an AUPR score of 35.9%, Precision at 80% Recall of 18.6% and Exact Match score of 67.8% as shown in Table-5.

4.2.1. Comparison Among All Models

Since the main purpose of these models is to review contracts, a high recall is a must. We don't want the model to misclassify any positive labels (important parts which are required to be highlighted). For this reason, a high recall (no. of positives predicted correctly out of all the actual positives in the dataset) is necessary along with high precision (no. of positives predicted correctly out of all the positives predicted by the model). For this reason, the models with high Precision@ 80% Recall and high AUPR (average precision of the model) should be selected. Analyzing Fig-6 and Table-5, we can come to a conclusion that for our task, A-type-RoBERTa-base is the best model as it has the highest Precision @80% Recall of 29.6% and second-highest AUPR score of 46.6% (highest 46.8%). Although A-type-RoBERTa-base has the highest AUPR score of 46.8% it falls significantly behind in the Precision @80% Recall metric with a score of 25%. So, we decided to use the A-type-RoBERTa base model for our project.

4.2.2. Quantized Model Performance

To reduce the model size for more space conscious deployment, we applied quantization on our best model, A-type-RoBERTa-base model. This allowed reduction of the model size from 500mb to 240mb which is a 50% decrease in size just by quantizing the linear layers. However, while evaluating the results are not quite up to the mark. The results in Table-6 are for dynamic quantization which quantizes the weights beforehand whereas the activations are quantized at runtime. So we see that, the exact match score decreases by 36% and the F1 Score decreases by 39%.

5. Conclusion

In this paper, we present a machine learning and NLP powered application for automatic contract review utilizing the open source CUAD question answering dataset. We presented the logical workflow of the application along with our trials and experiments and with different tools and technologies for each functional step.

Legal documents are inherently lengthy, and we've outlined the challenges of applying ML and NLP processing to them under resource constraints, due to which we were unable to carry out our experiments utilizing larger transformers, such as DeBERTa-xlarge. Nevertheless, we achieve a higher AUPR score for the RoBERTa base model compared to the results of the CUAD paper (Hendrycks et al., 2021). We also present our experiments into text extraction from contracts which are both searchable, i.e. digitally created or text overlaid documents, and non-searchable i.e. scanned documents to allow users to upload contracts from different sources.

Our application is open source and available on github, as cited in section 3.5.1. With our work we aim to contribute towards open source tools and technologies in the legal field for legal professionals and the general public without legal education and means to afford legal services for professional contract review.

Appendix: Application Overview

Fig-7 portrays the pictorial shots of our front-end.

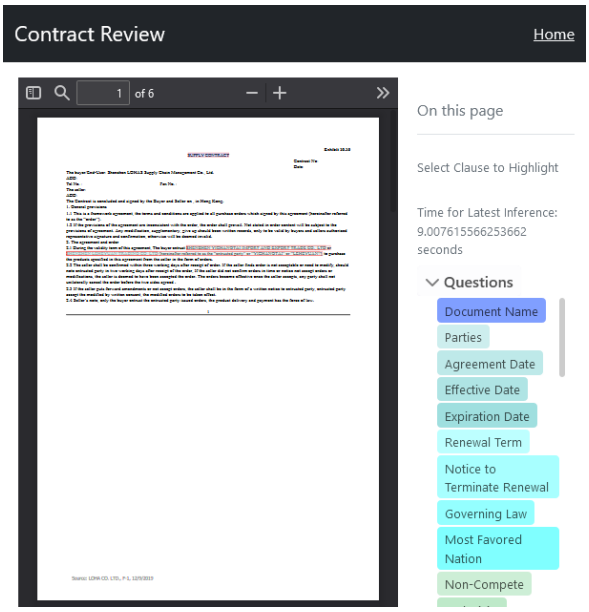

Figure 7: Screenshot of Legal Contract Review Application

6. Bibliographical References

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The Long-Document Transformer.

Breuel, T. M. (2007). The hOCR Microformat for OCR Workflow and Results. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1063–1067. IEEE.

Dabass, J. and Dabass, B. S. (2018). Scope of Artificial Intelligence in Law.

Hegel, A., Shah, M., Peaslee, G., Roof, B., and Elwany, E. (2021). The Law of Large Documents: Understanding the Structure of Legal Contracts Using Visual Cues. *arXiv preprint arXiv:2107.08128*.

Hendrycks, D., Burns, C., Chen, A., and Ball, S. (2021). CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. *arXiv preprint arXiv:2103.06268*.

Holzenberger, N., Blair-Stanek, A., and Van Durme, B. (2020). A Dataset for Statutory Reasoning in Tax Law Entailment and Question Answering. *arXiv preprint arXiv:2005.05257*.

JaidedAI.). JaidedAI/EasyOCR: Ready-to-use OCR with 80 Supported Languages and All popular Writing Scripts.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.

Leivaditi, S., Rossi, J., and Kanoulas, E. (2020). A Benchmark for Lease Contract Review. *arXiv preprint arXiv:2010.10386*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Malik, V., Sanjay, R., Nigam, S. K., Ghosh, K., Guha, S. K., Bhattacharya, A., and Modi, A. (2021). ILDC for CJPE: Indian Legal Documents Corpus for Court Judgment Prediction and Explanation. *arXiv preprint arXiv:2105.13562*.

Peng, H., Prodic, A., Alarcón, E., and Maksimovic, D. (2007). Modeling of Quantization Effects in Digitally Controlled dc–dc Converters. *IEEE Transactions on power electronics*, 22(1):208–215.

Roegiest, A., Hudek, A. K., and McNulty, A. (2018). A Dataset and an Examination of Identifying Passages for Due Diligence. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 465–474.

Tesseract-Ocr.). tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository).

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR*, abs/1910.03771.

Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., and Sun, M. (2020). How does NLP Benefit Legal System: A Summary of Legal Artificial Intelligence. *arXiv preprint arXiv:2004.12158*.