Data Science Lab Report: Legal Tech

Anver Khusainov, Clémence Lanfranchi, Fernando Gonzalez

September 2021 - February 2022

Contents

1 Introduction

Legal decisions are complex tasks in a high-stakes setting. They require extensive analysis of a case, contrast the analysis to the law and past similar cases to eventually reach a decision. When a legal entity makes a decision, the losing party can file an appeal usually based on arguments that there were errors in the trial's procedure or errors in the judge's interpretation of the law. To do this, the appellant must submit a brief containing the legal arguments that justify why the decision should be reversed. The court then makes the decision based on the information presented. Specifically, the process is as shown in Figure 1.

This process is a time-consuming task: The court has to take into consideration the case and all the filed briefs. Machine learning NLP models are a useful tool to improve the efficiency of document analysis. With the rise of transformer models, e.g. [1], machine reading and understanding of natural language text has significantly improved. Thus, the goal of this project was to predict the court's outcome of an appeal (affirmed, reversed) based on the briefs and other variables. We explored different models from simple logistic regression to the latest deep learning models. We compared the models' results and worked on their interpretation. These models can be used as a tool for the court to have a quick overview of the appeal, select which cases are more ambiguous and decide which of them should be analyzed more carefully. Our models can also be helpful for lawyers to understand how to present their arguments and decide whether it is worth spending time preparing a brief for a specific case.

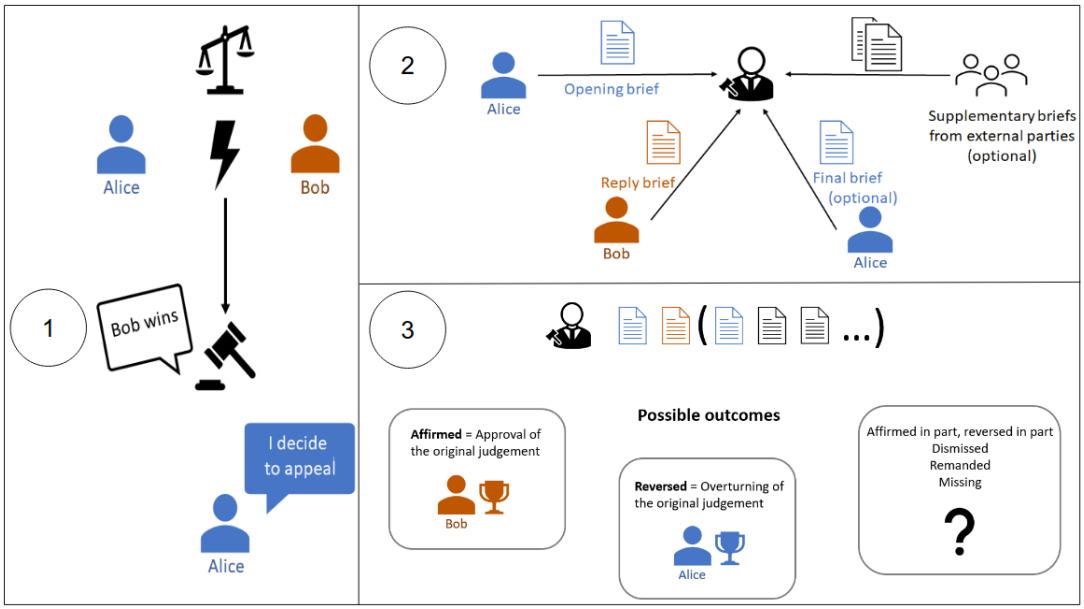


Figure 1: **Appeal process**. **1**.A decision is made by the court and one of the parties doesn't agree with the decision. **2**.The appellant submits a brief. The appellee then has a specified time to file an answering brief and the appellant may then file a second brief answering the appellee's brief. Other entities may also file briefs in support of one of the parties. **3**.The appellate court determines whether errors occurred in applying the law.

2 Related Work

One of the first approaches to predicting court decisions was Ruger et al. [2] They used classification trees to predict the votes for each Justice in a sequential manner, i.e. the predictions for some votes use as inputs predictions of the vote of other Justices. They used 6 variables (circuit, area of the case, type of petitioner, type of respondent, the ideological direction of the lower court and, whether or not the petitioner argued the constitutionality of a law). However, their model was not general in the sense that it was constructed for the 9 specific Justices sitting at the Court at that time. Then, Katz et al. [3] developed a time-evolving random forest where they included similar features but also features related to the past behavior of the Justices like the rate of reversal decisions. They also included a memory parameter "M" that controls how far in the past the model looks. One of the first approaches that uses textual content to predict Court's decisions was Aletras et al.[4] They use textual features like N-grams to train SVM classifiers to predict the decision of whether there has been a violation of an article of the convention of human rights. Finally, Medvedeva et al.[5] use SVM and TF-IDF to predict article violations but they removed some parts of the text that contained the verdict used by Alestras et al. We explore multiple representations of the text and types of models including deep learning architectures. To the best of our knowledge we are the first to use transformers to predict the outcome of the Court's decisions.

3 Presentation of the Data

3.1 Description of the Datasets

The dataset we used for our project consists of a corpus of about 160,000 legal briefs submitted to U.S. appellate courts from 1946 to 2016. The data was provided for the project by supervisor Dr. Ash. All of the 11 U.S. courts of appeal are represented in the dataset as well as the Federal circuit and the District of Columbia circuit. However, we observe some imbalance among circuits, some of them having more briefs than others, which could be explained by a natural imbalance in the number of appeal cases made in these courts or by a lack of representation of these courts on the data mining platform. The briefs are usually lengthy, with an average of about 5000 tokens (words) per brief. This constitutes one of the main challenges of this project. Another challenge of this dataset comprises the presence of technical language and sentence structures that are specific to the legal domain and the presence of many citations of law articles or past cases which might be hard to learn from. And lastly, the outcomes are imbalanced where the vast majority of the decisions gets affirmed (see further description in Section 3.2).

As already discussed, briefs can be of different types. For a single appeal case, two briefs are required: The opening brief that is filed by the appellant who states its case and the reasons for appealing the original court decision and the reply brief that is filed by the appellee as a response to the opening brief. Along with these mandatory briefs, other supplementary briefs or briefs from "Amicus Curiae" can also be filed to assist the court in deciding the matter. Therefore, the number of distinct appeal cases is much lower than the total number of briefs, since many briefs can belong to the same case.

Appeal cases are uniquely identified by the combination of three information which are the court of appeal, the year that the case was first appealed and the docket number of the case which is a five-digit number that is assigned to the case upon creation.

It is important to mention is that this dataset does not contain the outcomes of the appeal cases. A natural solution would be to map the briefs to their associated court decision using the identification key (court, year, docket) of each case. However, this join did not work well in our case. A possible explanation is that the dataset concerning court decisions had not been extracted in the same way the dataset with the court decisions was constructed. As a consequence, we relied on a

dataset containing the judicial opinion that judges write in order to explain their decision. There are 7 possible different outcomes of an appeal case, which are:

- Affirmed - Approval of the original judgment
- Reversed - Overturning of the original judgment
- Affirmed in part, reversed in part (self-explanatory)
- Dismissed - The court dismisses the appeal
- Remanded - The case is sent back to the lower court
- Other
- Missing - The opinion is missing

In this project, we decided to only focus on the appeal cases that were either Affirmed (0) or Reversed (1) since the other outcomes are more ambiguous and would probably decrease the performance of our models.

3.2 Data Preprocessing

Preprocessing of the briefs dataset For this dataset, the most important preprocessing tasks are to extract the docket number of the briefs and the type of each brief. For the first task, we were able to use the XML tags to locate the part of the brief containing the docket number. However, they are often mixed with other identification numbers, so we further used regular expressions to only extract the docket numbers. We decided to keep the first appearing docket number in case several such numbers are present in the same XML tag. Concerning the briefs' types, we rely on the lemmatized briefs' titles and we define some simple rules to categorize the briefs (for e.g., if the brief's title contains the word 'opening' then it is an opening brief). We decided to only keep the opening briefs to train and evaluate our NLP models in order to avoid high correlation between briefs. While our two extracting methods for the docket numbers and the briefs' types are not fully assured to return the right information, some manual checking on a portion of the data makes us confident that they provide sufficiently good results for the need of this project.

Then, concerning the main body of the brief, we remove some metadata and XML markup using Python's XML parser but we do not perform any further preprocessing. That is, we do not change any capitalization, punctuation, etc. In particular, we preserve the special legal citation notation used by U.S. courts.

Preprocessing of the judicial opinions Since judicial opinions can be quite lengthy, we first remark one important thing: In most cases, the final verdict is stated either at the very beginning or at the very end of the opinion. Therefore, we select only the first thousand tokens as well as the last thousand tokens of each opinion. We then lemmatize the text in order to finally apply brute force search to find which outcome is mentioned out of the 7 possible ones detailed above. However, it is worth mentioning that this dataset includes both lead opinions, which refer to the majority decision and addendum opinions, which are individual opinions from judges who wish to record their concurrence or dissent from the majority decision. This might have added some noise in extracting the outcomes.

Construction of the final dataset After preprocessing our two datasets, we can finally map the briefs to their outcomes, in case such an outcome exists. We then filter the opening briefs that have outcome Affirmed or Reversed, which leaves us with a dataset of 25,948 briefs. Among these briefs, about 86% are Affirmed and 14% are Reversed, meaning that our dataset is quite imbalanced, which is one of the challenges of this project.

4 Methodology

4.1 Modeling choices

Before we describe various approaches, we need to operationalize the evaluation process. There are two questions we want to answer. The first is how to split the data. The second one is how to measure the model's performance. Let's start with the data split.

It is tempting to use random split and, ideally, K-Fold validation, which is common practice in Machine Learning. However, Katz et al. [3] suggest a chronological split, i.e., train on the past to predict the future. The motivation is that, in practice, we want to use the model to predict future cases (after training the model on existing cases). A random split potentially may use information from the future itself. Furthermore, Medvedeva et al. [5] find supportive quantitative evidence of this hypothesis on the data from the European Court of Human Rights. They showed that a random split produces a too optimistic estimation of the model performance. Moreover, they investigated the importance of the "gap", i.e., the time between train and test data. Finally, they have shown that too old data in train may lead to worse results on the fixed test data. Thus, we decided to split it chronologically:

- train data: 1946 — 2011
- validation data: 2012 — 2013
- test data: 2014 — 2016

The number of cases vary significantly within different years. Moreover, until 2000, many years are skipped, and there is less than 230 cases before 2000. After 2000 the distribution of number of cases is shown in Figure 2a. Also, for some models, we used the suggestion from [5] and considered the difference in distributions for different years. As shown on Figure 2b the distribution has changed significantly in 2006. Thus, for some models (e.g., BERT on summaries), we only used train data from 2006. However, it did not affect the performance significantly. It is important to note that we fixed validation and test data, making the results comparable.

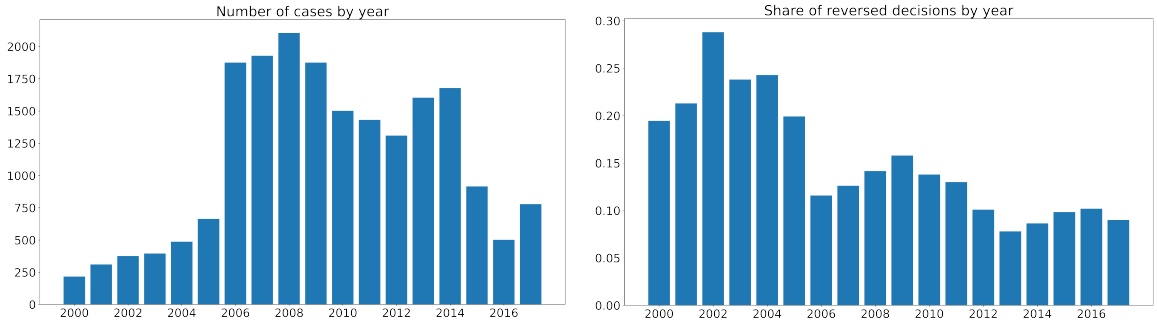
Chronological split is aimed to give a more realistic estimation of the model performance. It holds because it reduces unintended dependencies between cases. It is worthy to note that chronological split is not the only way to avoid such dependencies. Another approach we considered was to split the data by courts. For example, use one court for validation and test and all other courts for training. However, due to the limited time, we did not continue with this idea.

We also need to choose the metric (or set of metrics) to optimize. Due to the high class imbalance (share of reversed decisions is usually from 0.1 to 0.2 for different years), accuracy is not an informative metric for this task. Similarly to [3, 5], we use the F1 score. We also report ROC-AUC because, unlike the F1 score, this metric does not depend on (an arbitrarily set) threshold. It is helpful because the threshold may vary for different purposes. We also looked at other metrics, such as accuracy or average precision score, and have created confusion matrices. However, these additional metrics are mostly used for sanity checks, and we do not report them for all models.

4.2 Baseline/classical machine learning models

4.2.1 Non-NLP features

To build the first baseline model, we used hand-crafted features. This approach was proposed in Katz et al. [3]. Since these features are simple and do not make use of any possible advantages an NLP approach might yield, we refer to them as "non-NLP features". Although such features intuitively should not have significant predictive power, they are highly interpretable and can yield interesting insights. We readdress its interpretability in 6.2. In this section, we cover the choice of features and the model.



(a) (b)

Figure 2: Histograms of number of cases across years (a) and share of reversed cases (b)

For simplicity, let's divide these features in groups:

- information about the process (e.g., date, court)
- information about the parties (e.g., is appellant a person, race of appellee)
- simple text features (e.g., number of words in the brief)

Unfortunately, most of these features were not present in the data, so we involved other libraries for extracting them. To extract whether some party is a person or organization or GPE (Geopolitical entity), we used the NLP library spaCy [6]. Although this framework is common in NLP, one should not expect perfect labelling accuracy. We also extracted other binary features for the most frequent parties, such as "is appellant California" or "is appellee USA".

As mentioned, we use information about the race of both parties because this seems important from a social perspective. However, our dataset doesn't contain this information. We proposed a hypothesis that we can extract races from names. For this, we used the library ethnicolr [7]. The library yields a probability distribution over races using a name as input. However, there were some challenges: the library accepts first and last names, while we only had a full name. Anticipating 6.2, this can be one of the reasons why these features did not improve the score.

We investigated various baseline models, i.e. Support Vector Machines (SVM) [8], Random Forests [9] and XGBoost [10]. After manual hyperparameter optimization, we proceeded with XGBoost given its speed, interpretability and performance. Also, this model is resistant to small changes in hyperparameters and doesn't tend to overfit.

4.2.2 TF-IDF

Next, we investigaed models which also consider the text of a brief as input features. Our first model explored uses the TF-IDF transformation of a brief's text, that is, it creates a vector for each brief where each entry represents a word and the values are the importance of that word for the brief. Importance is defined as the product of the frequency of the word and the inverse document frequency over all briefs, that is the logarithmically scaled inverse fraction of the documents that contain the word.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{1}$$

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t,d}} \tag{2}$$

$$idf(t, D) = \log \frac{N}{|d \in D : t \in d|} \tag{3}$$

Where t is the word, d the document, and D is the collection of all documents.

The advantages of the resulting model is that it is simple and the interpretation of the features is straightforward. However, it only detects how important a word is in predicting the result in isolation, and does not take any sort of context or composition of language into account. We performed a grid search over the same algorithms as in the Non-NLP approach (SVM, Random Forsts and XGBoost) and the best resulting model for these features was the SVM.

4.2.3 Doc2Vec

Another way of transforming the text is using dense document embeddings. Specifically, we used Doc2Vec [11] to generate such vectors. Doc2Vec is an extension of Word2Vec [12], an approach that learns word embeddings that have semantic meaning by training a neural network with a single hidden layer to predict the neighboring words given a target word on large collections of text. The learned weights are used as the word vectors (or word embeddings). The extension in Doc2Vec is the inclusion of a separate input feature to the neural network which represents the document id. The rationale behind using these embeddings is that the different dimensions in the vector capture semantic meaning of the text contrary to the TF-IDF representation where each dimension represents the (weighted) occurrence frequency of a word and any pair of words have the same relation between them no matter how similar they are. We again gridsearched over our different algorithms and find that the best performing model using Doc2Vec as features was the SVM.

4.3 Transformers models

The Transformer [13] is a deep learning architecture that originally was proposed for sequence to sequence tasks in Machine Translation and has proven to be very effective in all of NLP. The original proposed architecture in [13] consists of an encoder and a decoder that use an attention-mechanism that can attend to an input sequence and decides at each decoding step which other parts of the source and (already generated) target sequence are important. The attention mechanism is fully connected, i.e., each word can pay attention to each other word in a sequence. Hence, the attention mechanism scales quadratically with sequence length, which was mititaged in early implementations of transformers by just allowing a maximum sequence length (typically 512 tokens).Our legal briefs are usually way longer than that, so we propose two different solutions to this problem.

4.3.1 Long-document transformers

LongFormer The first method is using transformers which do not compute the full attention matrix anymore (sparse transformers) and thus can handle longer sequences. In particular, Longformer [14] handles up to 4096 input tokens. Because only a partial (local) attention matrix is computed, the attention mechanism scales linearly with sequence length; it combines a local windowed attention with a task motivated global attention.

BigBird Similarly to LogFormer, BigBird [15] can handle sequences up to 4096 tokens, by also implementing sparse attention mechanisms. On top of the local and global attention in [14], BigBird also implements random attention, i.e. random connections between word pairs are sampled and these words can attend to each other. This combination enables BigBird to get rid of the quadratic dependency in the input sequence length that usual Transformers have while preserving the expressivity and flexibility of the quadratic full Transformers.

However, for both of these methods, even the increase in the maximum sequence length was not enough to use the entire text for all of the briefs. We tried two ways of manually shortening the briefs. The first one was selecting just some of the sections of the brief using the XML tags

(summary, statement of the case and argument). The second approach was to truncate the briefs after the first 4096 tokens, The second method (truncation) obtained better scores.

For both approaches, we used pretrained models that were fine-tuned on our briefs dataset.

4.3.2 Short-document transformers

Although long-document transformers are designed to work with large documents, they also have a significant drawback: It takes much more time to train these models. Hence, it is hard to experiment with them and find an optimal configuration. Also, research on short-document classifications receives more research attention nowadays, so we deduct that these models can be more robust. Thus, we also opted to approach the problem from the other side. First, we summarize a brief automatically. Then, then, we apply a standard, fully-connected Transformer architecture on the resulting (short) summary for classification.

Text summarization There are, broadly speaking, two ways of text summarization: extractive and abstractive. The former one extracts the most salient words or sentences from a source text and copies these into the summary. Abstractive summarization is a sequence to sequence task with the goal to decode the summary given the source text. Abstractive summarizers are more general, but using this approach will make the whole pipeline even more opaque. Indeed, transformers themselves are hard to apply in areas where transparency is crucial because their predictions are hard to explain. Using the model which rephrases the initial brief will make any attempt infeasible which tries to shed light in how these predictions were obtained. Given that interpretability is important in our setting, we limited ourselves to explore extractive summarizers.

We used a model called MemSum [16]. It outputs up to 20 sentences and accepts up to 400 sentences as input. Keep in mind that the briefs in our dataset are normally even longer than this. Thus, we also removed all but the longest 400 sentences before applying the summarization model. This throws away information and will introduce a bias towards longer sentences. However, manual inspection yields that most short sentences just contain formal language without semantically relevant information, so we perceive this approach to be reasonable.

BERT BERT [1] and its modifications are currently one of the most popular transformers in NLP. It has a limitation of 512 tokens (including special), so we truncated the summaries when needed. Due to BERT's performance in many applications, there are various fine-tuned models for different areas. One of the providers of such transformers is HuggingFace [17]. We used the model pre-trained on legal corpora: Legal BERT [18], which shows significant improvements on a range of legal NLP tasks.

BERT outputs for each token a contextualized embedding, i.e. a fixed size vector. These embeddings can then be used for a specific task, such as binary classification. So there is always a choice for the classifier on top of the embeddings. We tried to use XGBoost as in previous approaches, using contextualized BERT embeddings as input. We also experimented with a shallow neural network on top of these embeddings, but the performance was unsatisfactory in both cases. Thus, we tried another common approach: Fine-tuning the whole BERT model on the specific task. The idea is to train not only a classification model but also all BERT parameters (initialized from a pre-trained model). However, such an approach has a significant shortcoming for small datasets: BERT has hundreds of million parameters and is capable of memorizing the data.

To mitigate this, it is common practice to only update some parameters of the underlying model. We got the best results after freezing all parameters except those in the last two layers. The classification network does not need to be very large due to the dataset size. So we used a 2-hidden layer neural network (with RELU activation) with batch normalization [19] and dropout [20] layers. Batch normalization is a technique that speeds up and stabilizes the model training. Dropout is a common regularization in neural networks.

4.4 Model ensemble

Up to this point, we discussed different modelling algorithms and input features. Naturally, each of these combinations capture different information. While the Non-NLP model captures the importance of the general characteristics of the case like court number or type of appellant and appellee, the other models mainly capture how important some words and phrases are for predicting the outcome. That is the reason why we decided to combine the different models in an ensemble, to take advantage of the strengths of each model and complement them. We took the models described previously and used their probabilities as input features to a new model. Through a logistic regression, we obtained the weight of each classifier. For the final ensemble we used 3 models, the resulting weights are shown in Table 1

	XGB Non-NLP	**SVC TF-IDF**	**BERT summaries**
Weights	5.19	4.80	2.14

Table 1: Ensemble model weights

5 Results and Discussion

5.1 Results

Table 2 summarizes the evaluation results of all the algorithms. Transformer models are the ones that achieved the best results in both metrics. In specific, **BERT on summaries** achieved the best score if we look at individual models but the **Ensemble** model got the best performance overall. The confusion matrices for those two models are shown in Figure 3.

	ROC AUC	**F1 score**
XGB Non-NLP	0.704	0.260
SVC TF-IDF	0.722	0.270
SVC Doc2Vec	0.701	0.269
Longformer	0.701	0.272
Big Bird	0.710	0.272
BERT on summaries	0.726	0.300
Ensemble	0.752	0.310

Table 2: Performance comparison

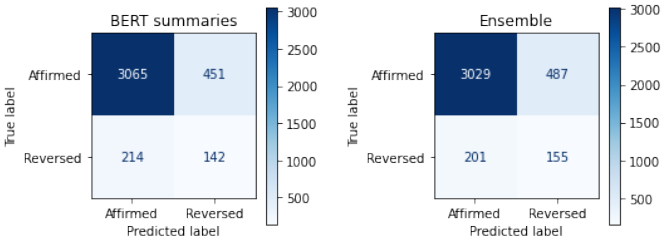


Figure 3: Confusion matrix for BERT (left) and Ensemble (right)

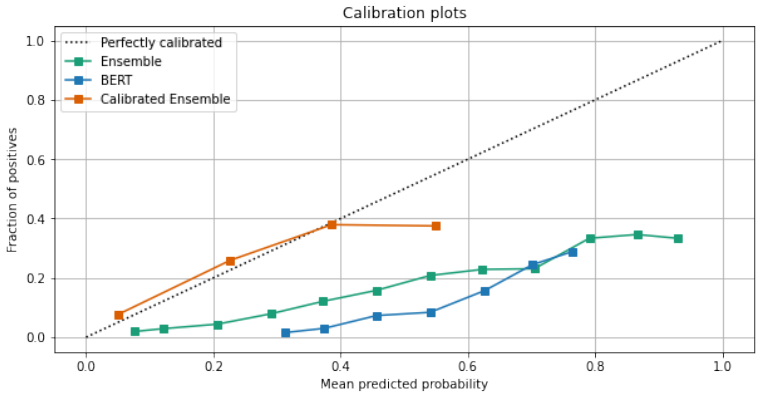


Figure 4: Calibration plots

In this project, we considered ROC AUC and F1 because they provide a general perspective of the performance of the model. However, the final selection of an evaluation metric depends on the specific use case. For instance, if the model is used to select which cases should be checked more in detail by a judge, a high precision score will be more important if judges do not want to spend time checking cases that will be affirmed anyway.

Another important aspect to consider when working with the outcome of the model is how much we can rely on the output probabilities. If our model assigns a probability of 0.4 to a case to be reversed we expect that for every 10 cases that our model predicts 0.4, the decision will be reversed in 4 of them. However, our models were not originally calibrated as shown in Figure 4. With a calibrated model we can have a better perspective of how our models will behave when predicting new cases. We calibrated our best performing model using a logistic regression model. The resulting final calibration plot is shown in Figure 4.

5.2 Discussion

We identified some modifications to the modeling process that can potentially improve the model's performance.

Better data labeling One of the first obstacles during the project was data labeling. We found inconsistencies in the original outcomes so we decided to extract them from the judicial opinions as explained in Section 3. This labeling process could be improved by performing a more detailed extraction by either manually labeling all the opinions or labeling a subset of them and training a model to do the rest.

Data inputs Briefs usually contain many citations, they refer to other briefs and legal acts. These references may contain valuable additional information for predicting the decision. It would be useful to include in the models not just the number of citations present in the text, but also a summary of the content of the cited case or law. Moreover, our current models only consider the opening briefs. We explored approaches appending the reply brief's text to the input of the transformer models, but the scores did not improve. However, additional approaches could be explored, like implementing Heterogeneous GNN as in [21] treating reply and additional briefs as separate entities that are linked to the opening brief or even to other cases.

Summary We considered the problem of classifying whether an opening brief leads to an affirmative or reversed decision of an existing case. Because such briefs are lengthy, we used a pre-trained

model to generate an extractive summary of a brief. We believe that some improvements can be made in this stage by either trying different parameters (number of sentences), modifying the text before inputting it to the summarizer, or using a different model altogether to generate the summaries.

Other approaches: Causal attribution model We explored the approach in [22, 23] where two parts of the model are trained jointly, one part learns word embeddings, and the other tries to predict the outcome using covariates, both using feed-forward neural networks. Control variables are used to encourage the model to capture information related to the narrative persuasion of the text rather than confound-related information that usually is not possible to modify. In this case, we used as covariates the court number and type of appellant and appellee. The model outputs an importance score for each word on how well it explains the outcome. Table 3 shows the most important words with their score. In general, we observe the presence of procedural words which could indicate that the judges look at technical rather than simple words explaining the argument. In future work, this approach could be extended by implementing more complex models for both parts: The covariates and the text.

Word	**Importance score**
view	0.195
parties	0.181
removable	0.177
jury	0.173
admissability	0.170
Cross-Motions	0.152
indefiniteness	0.152
loaned	0.151
conspiracy's	0.146
rejection	0.145
statutorily-required	0.140

Table 3: Word importance in causal attribution model

6 Interpretability

Since the long-term objective of this project would be to provide a tool to help judges in their decision-making process, it is all the more important to explore the interpretability of our models. Indeed, as with all predictive models that are applied to our every-day life, there is a strong need to explain in human understandable terms why a model is doing certain predictions. In the following sections, we explore three different ways to unveil the way our models work.

6.1 Local Interpretable Model-agnostic Explanations (LIME)

LIME [24] is an algorithm which explains the predictions of any black-box model in an interpretable and faithful manner. It learns an interpretable model locally around the prediction. More precisely, rather than generating an explanation for the whole model, this method focuses on individual predictions. Considering a data point that we want to explain, LIME samples instances around this instance, obtains predictions from the black-box model and weighs them by the proximity to the data point in question. Then, using this newly generated dataset, LIME trains an interpretable model (such as Lasso or decision trees) which provides local explanations.

Since this method is quite resource-intensive, we decide to apply it with TF-IDF and Logistic Regression considering that this model achieved comparable results to our best models (ROC AUC of 0.72) and is faster to run. For the predictions to investigate, we look at the examples that the model predicts with highest probability for each of the 4 cases: True Positives (the model predicts Reversed rightfully), True Negatives (the model predicts Affirmed rightfully), False Positive (the model predicts Reversed while the appeal case was Affirmed) and False Negative (the model predicts Affirmed while the appeal case was Reversed). This choice comes from the fact that we want to analyze which words have the strongest impact in leading to the right or wrong prediction and we take the examples predicted with highest probabilities so that we avoid some noise due to the model being uncertain about its decision.

We provide in Appendix A some visualization of the output of the LIME algorithm for some of our examples. It results from these examples that some words such as "sentence", "offense" or "appellant" tend to lead to an Affirmed prediction, whereas words like "federal", "district" or "court" tend to lead to a Reversed prediction. This observation can be quite simply explained by the fact that the appearance of the word "sentence" is usually linked to criminal cases which are very rarely reversed, whereas civil cases are more likely to be reversed.

After getting such insights from explanation, it appears that the model was able to deduce some simple rules about judges' decision behavior, however it does not provide us with more broad reasons to trust its predictions. Moreover, we also see that single letters like "s" or "v" appear among the most important words used for predictions, which have no connection to the appeal decision and thus indicate that the model should probably not be trusted too much.

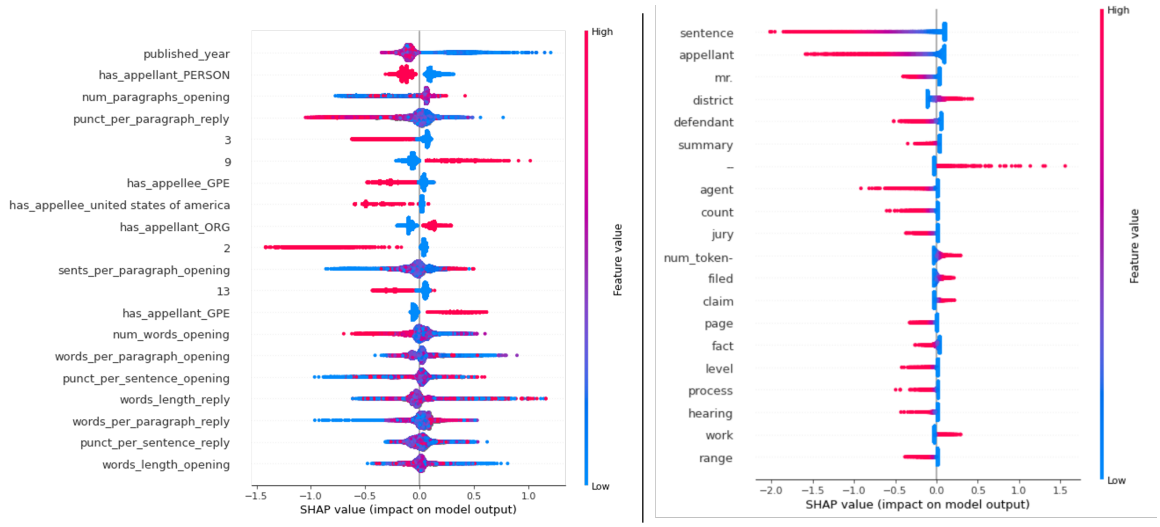
6.2 SHapley Additive exPlanations (SHAP)

SHAP [25] is an approach that unifies six previously existing methods of measure of feature importance (including LIME). It relies on Game Theory to compute Shapley values, which provide theoretical guarantees about accuracy (the explanation model matches the original model) and consistency (*i.e.*, if a model changes so that some feature's contribution increases, that feature's attribution should not decrease). Moreover, SHAP is model agnostic and can be used for global interpretation by calculating the Shapely values for a whole dataset and aggregating them.

We use SHAP both for the Non-NLP features and for the model with TF-IDF and Logistic Regression. We show the output of these two global interpretations in Figure 5.

Non-NLP Features The key insights from this visualization are the following:

- The feature `published_year` is very important. This resonates with our finding in 4.1 concerning the varying share of reversed cases across years. Similarly, we can see that the court number (features `3, 9, 2, 12`) also plays a key role, which indicates that the share of reversed cases is not equal among courts of appeal.
- People (feature `has_appelant_PERSON`) have less chance to win than organizations (`has_appelant_ORG`) or states (`has_appelant_GPE`). This finding appears quite natural since we can imagine that organizations and states usually have more experienced lawyers than regular people.
- Simple text features like total number of words or average word length are barely helpful, which indicates that judges are not much influenced by wordy briefs or the language register used in the brief (since longer words in English are usually linked with a more elevated and formal register).
- Features about race are not helpful at all, which might have two explanations: either judges are not influenced by the race of the parties involved in the case or the race features that we extrapolated from the full name of the parties are not reliable enough to be useful for



(a) Non-NLP features with XGBoost

(b) TF-IDF and Logistic Regression

Figure 5: SHAP values for our XGBoost model on Non-NLP features (a) and for Logistic Regression with TF-IDF (b)

prediction. The ground truth might be a combination of both explanations since previous research has found unconscious racial bias among judges [26] and that race could play a role in reversal rates in appellate courts [27, 28] but we also doubt that our extraction technique of racial features was completely right.

TF-IDF + Logistic Regression The global interpretation meets the previous local interpretation offered by LIME for the most important words like "sentence", "appelant" or "district". For the other words, it is quite difficult to draw insightful conclusions and the presence for instance of "mr" or "-" as important words could make us question the relevance of the model's predictions.

As a consequence, the interpretability reveals that, despite showing decent performance, our baseline NLP models might be hard to trust since they do not rely on solid features to make their predictions.

Next, we evaluate the interpretability of our Transformers model. However, LIME and SHAP approaches are both computationally intensive. Therefore, we need to rely on an other approach that we detail below.

6.3 Integrated gradients

To analyze output of BERT model, we used a (layer) integrated gradients method [29] implemented in library Captum [30]. In short, this method analyzes the gradient of the prediction by input tokens. However, unlike other older approaches [31, 32] two important axioms hold here: sensitivity and implementation invariance. The first axiom implies that if only one feature was changed and prediction changed too, then the gradient by this feature should be non-zero. The second axiom means that gradients should be the same for two classifiers that agree on all possible inputs. Since the integrated gradients method satisfies both of these axioms, this method is currently getting more popular and reliable.

After calculating the gradients by input, one can treat these gradients as a measure of importance. Captum also provides a visualizer that highlights words according to their contribution to

the decision.

In Appendix B, there are some examples of the method applied to the briefs. We want to highlight mainly two insights from it. First, the crime cases seem to be more likely to be approved. Also, the briefs which mention ambiguity in the initial decision are likely to be reversed. However, one should not simplify these results to just these two insights because the model is very complicated. And there is still a lack of knowledge about the interpretability of transformers models and BERT, in particular.

7 Conclusion

We proposed SOTA approaches in the legal area. In similar recent works [3, 5] authors used more classical approaches described in 4.2.1, 4.2.2. We also explored these methods on our briefs dataset as baselines. However, we explored in depth state-of-the-art approaches in NLP based on transformers and observe that these outperform other techniques. Although the performance of the models does not seem good enough yet for direct usage in practice, our results still may simplify the work of judges. To account for this, we adjusted a tool for interpretability that should attract judges' decisions on the highlighted parts of the brief.

We also believe that appeal cases are usually hard to decide, and judges' verdicts are not necessarily ground truth. We showed in Figure 2 that the share of reversed decisions fluctuates a lot in the current century, so probably external factors (that are independent of the briefs) can affect judges' decisions. Thus, it might be an interesting topic for social research to estimate human performance in appeal decisions. Not only is it interesting from a social perspective, but it will also help to get a more realistic "upper bound" for the "robot judges".

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Theodore W. Ruger, Pauline T. Kim, Andrew D. Martin, and Kevin M. Quinn. The supreme court forecasting project: Legal and political science approaches to predicting supreme court decisionmaking. *Columbia Law Review*, 104(4):1150–1210, 2004.

[3] Daniel Martin Katz, Michael J Bommarito, and Josh Blackman. A general approach for predicting the behavior of the supreme court of the united states. *PloS one*, 12(4):e0174698, 2017.

[4] Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preotiuc-Pietro, and Vasileios Lampos. Predicting judicial decisions of the european court of human rights: a natural language processing perspective. *PeerJ Computer Science*, 2:e93, 10 2016.

[5] Masha Medvedeva, Michel Vols, and Martijn Wieling. Using machine learning to predict decisions of the european court of human rights. *Artificial Intelligence and Law*, 28(2):237–266, 2020.

[6] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[7] Gaurav Sood and Suriyan Laohaprapanon. Predicting race and ethnicity from the sequence of characters in a name, 2018.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[11] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[15] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[16] Nianlong Gu, Elliott Ash, and Richard H. R. Hahnloser. Memsum: Extractive summarization of long documents using multi-step episodic markov decision processes, 2021.

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[18] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online, November 2020. Association for Computational Linguistics.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[21] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. *Are We Really Making Much Progress? Revisiting, Benchmarking and Refining Heterogeneous Graph Neural Networks*, page 1150–1160. Association for Computing Machinery, New York, NY, USA, 2021.

[22] Reid Pryzant, Kelly Shen, Dan Jurafsky, and Stefan Wagner. Deconfounded lexicon induction for interpretable social science. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1615–1625, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[23] Reid Pryzant, Sugato Basu, and Kazoo Sone. Interpretable neural architectures for attributing an ad's performance to its writing style. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 125–135, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[25] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[26] Jeffrey J Rachlinski, Sheri Lynn Johnson, Andrew J Wistrich, and Chris Guthrie. Does unconscious racial bias affect trial judges. *Notre Dame L. Rev.*, 84:1195, 2008.

[27] Jonathan P Kastellec. Racial diversity and judicial influence on appellate courts. *American Journal of Political Science*, 57(1):167–183, 2013.

[28] Maya Sen. Is justice really blind? race and reversal in us courts. *The Journal of Legal Studies*, 44(S1):S187–S229, 2015.

[29] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017.

[30] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

[31] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Mueller. How to explain individual classification decisions, 2009.

[32] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

Appendix A

We show below examples of the LIME interpretation for each of the following cases: True Positive (Figure 6), True Negative (Figure 7), False Positive (Figure 8) and False Negative (Figure 9).

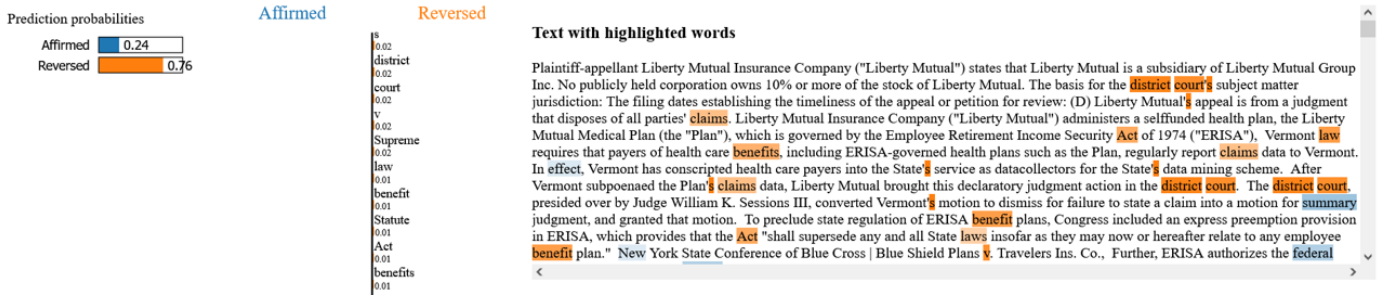


Figure 6: Example of a Reversed appeal case that is predicted as Reversed

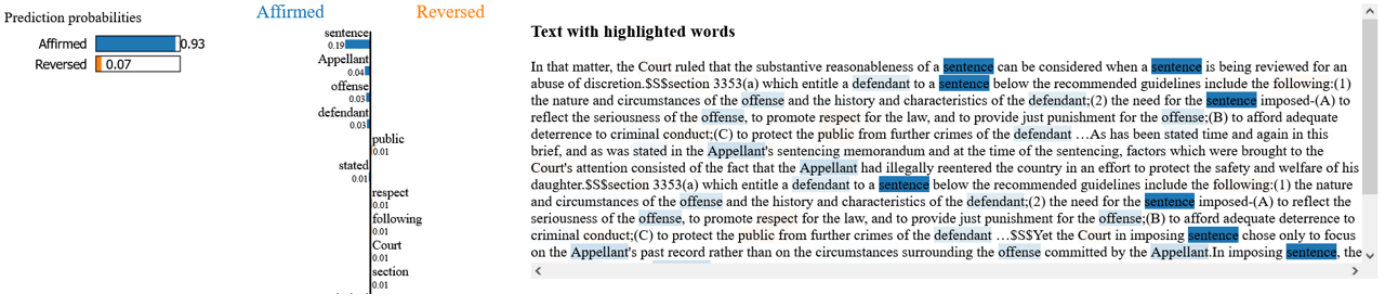


Figure 7: Example of an Affirmed appeal case that is predicted as Affirmed

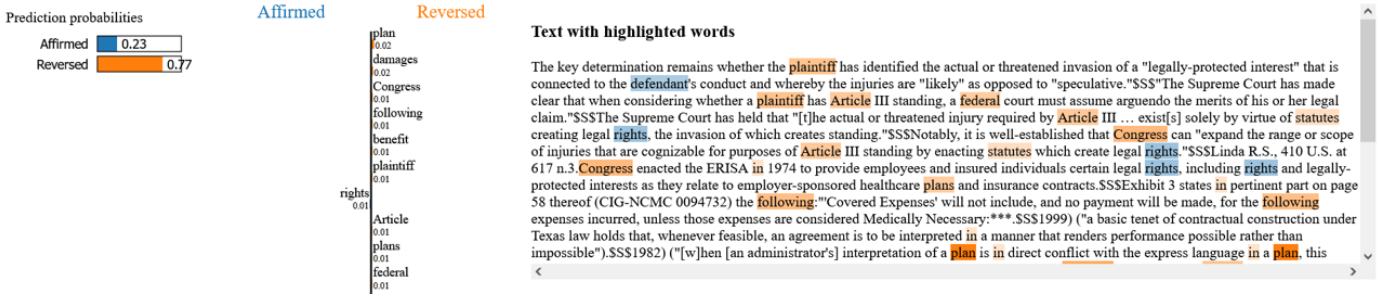


Figure 8: Example of an Affirmed appeal case that is predicted as Reversed

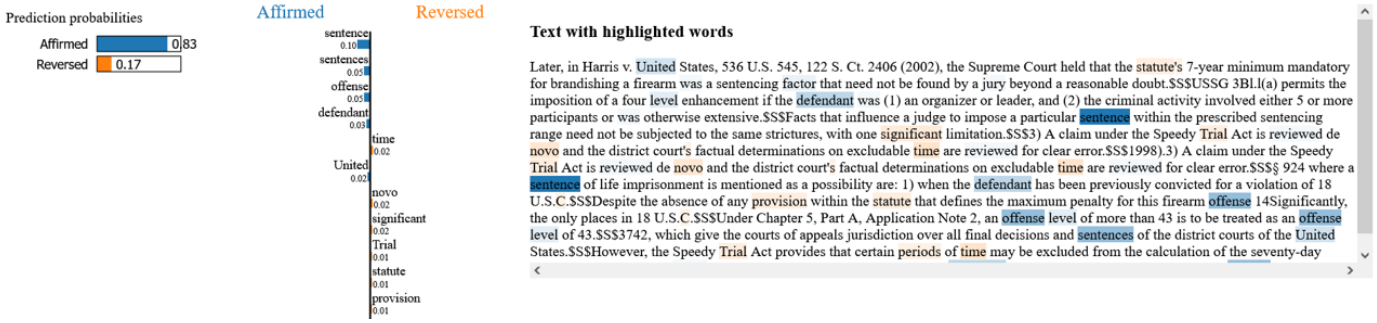


Figure 9: Example of a Reversed appeal case that is predicted as Affirmed

Appendix B

We show below examples of the Layer Integrated Method described in 6.3. Note that, unlike Appendix A, these results are both correct (TP and TN). The goal of these examples is to give some intuition of how the BERT model interprets the brief. The green highlighting corresponds to reverse strength, the read one to affirm. Note that BERT uses tokenizer and also splits some words on parts, these tokens are presented on the figures below.

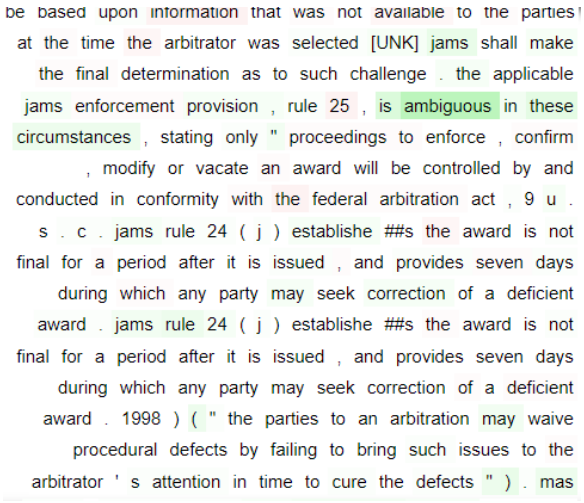


Figure 10: How ambiguity in the law affects the score (0.81 "probability" to reverse)

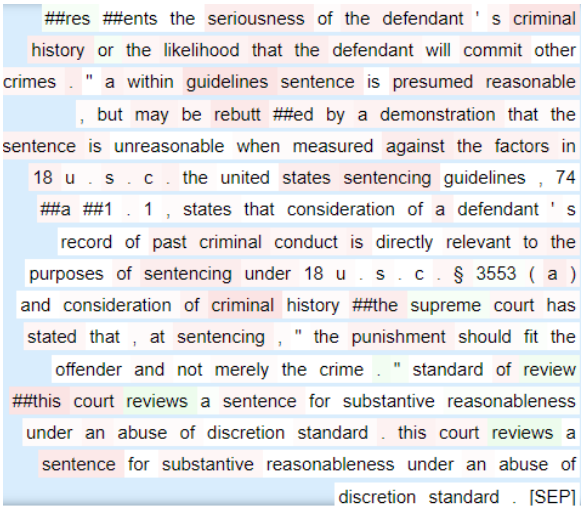


Figure 11: How crime cases affect the score (0.18 "probability" to reverse)