Neuro-Argumentative Learning with Legal Text

Petros Vasileiadis[1], Emanuele De Angelis[2,*], Maurizio Proietti[2] and Francesca Toni[1]

[1]Imperial College London, UK
[2]IASI-CNR, Rome, Italy

Abstract

We introduce a neuro-argumentative pipeline for legal outcome classification on plain-text legal case descriptions, which integrates the flexibility of neural machine learning methods with the inherent explainability and reasoning capabilities of Assumption-Based Argumentation (ABA). This approach addresses the limitations of opaque machine learning models by producing interpretable logical rules from case data, which can be used to justify the predicted outcome. The pipeline consists of a neural BERT-based feature extractor, which processes the case description to generate logical facts, and a symbolic component, which applies ABALearn, a form of symbolic learning, to these facts to derive an ABA framework that captures domain-specific rules. The trained feature extractor can then be used alongside the learned framework to predict the outcome of new legal cases. The learned rules serve as explicit justifications for each prediction, resulting in an inherently explainable decision-making process. When evaluated using a synthetically generated legal dataset, our proposed pipeline achieves performance comparable to state-of-the-art models in terms of F1 score and other standard classification metrics, while also introducing a transparent, symbolic reasoning layer.

Keywords

Legal reasoning, Neuro-symbolic machine learning, Assumption-Based Argumentation, Explainable AI

1. Introduction

Legal decision-making often involves binary or categorical outcomes, such as *guilty or not guilty*, or *application granted or denied*, based on case facts, applicable laws, and supporting arguments. This process requires careful manual analysis by legal professionals, who must identify relevant facts, interpret complex statutory language, and reason about their implications. The repetitive and time-consuming nature of this work has motivated research into AI-assisted legal decision-making systems [1].

Large language models (LLMs) such as BERT [2] and GPT [3] have demonstrated impressive natural language understanding capabilities and show promise for legal tasks such as case classification [4]. However, their black-box nature raises serious concerns, especially since these models can hallucinate, generating plausible-sounding but incorrect outputs [5], and often fail to perform the kind of structured, statutory reasoning required in law [6]. In the legal field, where decisions carry significant consequences and require justification, explainability is essential. Systems that provide decisions without transparent reasoning undermine trust, accountability, and fairness. Therefore, any system used for legal decision-making must prioritise explainability and transparency.

We propose a neuro-argumentative pipeline (see Figure 1) that enables Assumption-Based Argumentation (ABA) [7, 8] on natural language legal case descriptions. A neural feature extractor identifies relevant facts from case descriptions, which are translated into a symbolic form suitable for ABALearn [9]. This is a form of symbolic learning which obtains ABA frameworks from background knowledge and positive/negative examples, to then make case-based inferences given the facts of new cases.

One of the key benefits of symbolic learning is its inherent explainability. Each derived rule explicitly shows how particular facts lead to a legal decision or classification outcome. This allows users to trace which rules apply to a given case and see the justification for the prediction output.

We evaluate our pipeline on two datasets [10], extended with synthetic case descriptions. One is based on tort law, involving liability for damages [11] and the other concerns eligibility for a welfare benefit based on criteria like age, gender, and distance [12].

We compare our pipeline's performance to other state-of-the-art classification systems, using standard classification metrics such as accuracy and F1 score. The final performance of the pipeline is comparable to that of the state-of-the-art, while also introducing explainability and reasoning.

Overall, this paper contributes to the growing field of neuro-symbolic and explainable AI by demonstrating how neural feature extraction and argumentation-based reasoning can be effectively combined in the legal domain, offering both performance and explainability.

2. Preliminaries

2.1. Assumption-Based Argumentation

Assumption-Based Argumentation is a structured argumentation formalism that generalises various non-monotonic reasoning approaches. In ABA, arguments and attacks are defined through rules, assumptions, and their contraries. It provides a formal basis for reasoning, allowing us to construct arguments that attack or defend assumptions in order to draw acceptable conclusions [8].

An *ABA framework* (as originally proposed in [7], but presented here following [8]) is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ such that

- $\langle \mathcal{L}, \mathcal{R} \rangle$ is a deductive system, where $\mathcal{L}$ is a *language* and $\mathcal{R}$ is a set of *(inference) rules* of the form $s_0 \leftarrow s_1, \ldots, s_m$ ($m \geq 0, s_i \in \mathcal{L}$, for $1 \leq i \leq m$);
- $\mathcal{A} \subseteq \mathcal{L}$ is a (non-empty) set of *assumptions*;[1]
- $\overline{\ }$ is a *total mapping* from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{a}$ is the *contrary* of a, for $a \in \mathcal{A}$.

[1]The non-emptiness requirement can always be satisfied by including in $\mathcal{A}$ a *bogus assumption*, with its own contrary, neither occurring elsewhere.

ANSyA 2025: 1st International Workshop on Advanced Neuro-Symbolic Applications, co-located with ECAI 2025.
*Corresponding author.
emanuele.deangelis@iasi.cnr.it (E. De Angelis)
0000-0002-7319-8439 (E. De Angelis); 0000-0003-3835-4931 (M. Proietti); 0000-0001-8194-1459 (F. Toni)

Given a rule $s_0 \leftarrow s_1, \ldots, s_m$, s_0 is the *head* and $s_1, \ldots, s_m$ is the *body*; if $m = 0$ then the body is said to be *empty* (represented as $s_0 \leftarrow$ or $s_0 \leftarrow true$) and the rule is called a *fact*. In this paper we focus on *flat* ABA frameworks, where assumptions are not heads of rules. Elements of $\mathcal{L}$ can be any sentences, but in this paper we focus on ABA frameworks where $\mathcal{L}$ is a finite set of ground atoms. However, we will use *schemata* for rules, assumptions and contraries, using variables to represent compactly all instances over some underlying universe.

Example 1. *An simple ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ may be as follows, for $X, Y \in \{alex, bob, carol, john, mary\}$:*

$\mathcal{L} = \{innocent(X), person(X), not\_guilty(X), liar(X)$
$witness\_con(X,Y), caught\_in\_the\_act(X)\}$
$\mathcal{R} = \{innocent(X) \leftarrow person(X),\ not\_guilty(X),$
$guilty(X) \leftarrow caught\_in\_the\_act(X),$
$witness\_con(mary, alex) \leftarrow,$
$witness\_con(john, carol) \leftarrow,$
$liar(alex) \leftarrow, \quad caught\_in\_the\_act(bob) \leftarrow,$
$person(alex) \leftarrow, \quad person(bob) \leftarrow,$
$person(carol) \leftarrow, \quad person(john) \leftarrow$
$person(mary) \leftarrow\},$
$\mathcal{A} = \{not\_guilty(X)\}$ *with*
$\overline{not\_guilty(X)} = guilty(X)$

Without loss of generality, we will leave the language component of ABA frameworks implicit, and use, e.g., $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ to stand for $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, where $\mathcal{L}$ is the set of all sentences in $\mathcal{R}$, $\mathcal{A}$ and in the range of $\overline{\ }$. We will also write facts as rules with equalities in the body, e.g. we may write $liar(alex) \leftarrow$ as $liar(X) \leftarrow X = alex$.

Arguments are deductions of claims using rules and supported by assumptions, and attacks between arguments target assumptions in their support, as follows.

- An argument for (claim) $s \in \mathcal{L}$ supported by $A \subseteq \mathcal{A}$ and $R \subseteq \mathcal{R}$ (denoted $A \vdash_R s$) is a finite tree with nodes labelled by sentences in $\mathcal{L}$ or by $true$, where: (i) the root is labelled by s, (ii) leaves are either $true$ or assumptions in A (iii) non-leaves, s', have as children the elements of the body of some rule in R with head s'.
- Argument $A_1 \vdash_{R_1} s_1$ *attacks* argument $A_2 \vdash_{R_2} s_2$ if and only if $s_1 = \overline{a}$ for some $a \in A_2$.

In ABA, conclusions are drawn by determining acceptability of sets of arguments (or *extensions* [8]). In this paper, we use the following notion of extension:

- A set Δ of arguments is a *stable extension* iff (i) no argument in Δ attacks any argument in Δ (i.e. Δ is *conflict-free*) and (ii) every argument not in Δ is attacked by an argument in Δ.

Example 2. *The following are some of the arguments that can be constructed in the ABA framework F of Example 1 (for simplicity, we omit to indicate the set of rules that have been used):*

Arg_1: $\{not\_guilty(mary)\} \vdash innocent(mary)$
Arg_2: $\{not\_guilty(bob)\} \vdash innocent(bob)$
Arg_3: $\{\} \vdash guilty(bob)$

Argument Arg_1 is not attacked by any other argument, as no argument can be constructed for the claim $guilty(mary)$. Instead, Arg_2 is attacked by Arg_3, whose claim is the contrary of the assumption that supports Arg_2. Arg_3 cannot be attacked, as no assumption supports it. Thus, Arg_1 and Arg_3 are accepted in the stable extension of F, while Arg_2 is not. It can be seen that F has a unique stable extension (because the attack relation is acyclic). In particular, this stable extension includes accepting arguments for $innocent(alex), innocent(carol), innocent(mary)$, and $innocent(john)$, while it does not include an accepting argument for $innocent(bob)$.

We say that an ABA framework F is *satisfiable* if it admits at least one stable extension, and *unsatisfiable* otherwise. We also say that a sentence s is *accepted*, or *covered*, in a stable extension Δ of F, written $F \models_\Delta s$, if it is the claim of an argument in Δ.

2.2. Learning ABA Frameworks

ABA frameworks can be automatically learned from a given *background knowledge*, in the form of an ABA framework, a set $\mathcal{E}^+$ of *positive examples*, and a set $\mathcal{E}^-$ of *negative examples*. To this end, we follow the *ABALearn* method [9]:

- Given a satisfiable background knowledge $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, positive examples $\mathcal{E}^+ \subseteq \mathcal{L}$ and negative examples $\mathcal{E}^- \subseteq \mathcal{L}$, with $\mathcal{E}^+ \cap \mathcal{E}^- = \emptyset$, a *solution* of *ABA learning* is a new ABA framework, $F' = \langle \mathcal{R}', \mathcal{A}', \overline{\ }' \rangle$, with $\mathcal{R} \subseteq \mathcal{R}'$, $\mathcal{A} \subseteq \mathcal{A}'$, and, for all $\alpha \in \mathcal{A}$, $\overline{\alpha}' = \overline{\alpha}$, such that:

  (*Existence*) F' admits at least one stable extension Δ,

  (*Completeness*) for all $e \in \mathcal{E}^+$, $F' \models_\Delta e$, and

  (*Consistency*) for all $e \in \mathcal{E}^-$, $F' \not\models_\Delta e$.

The *ABALearn* method is based on the use of *transformation rules*, which can be used for deriving new ABA frameworks. These transformation rules include: (1) *rote learning*, which adds a new rule $p(X) \leftarrow X = a$; (2) *folding*, which, given rules $H \leftarrow B, C$ and $K \leftarrow B$, replaces $H \leftarrow B, C$ by the new rule $H \leftarrow K, C$; (3) *assumption introduction*, which, given rule $H \leftarrow B$, introduces an assumption α, with contrary $\overline{\alpha}$, and adds the new rule $H \leftarrow B, \alpha$; and (4) *fact subsumption*, which deletes any fact of the form $p(a) \leftarrow$ if there is an accepted argument with claim $p(a)$ in the ABA framework $\langle \mathcal{L}, \mathcal{R} \setminus \{p(a) \leftarrow\}, \mathcal{A}, \overline{\ } \rangle$.

The ABALearn algorithm iterates four steps (according to the pattern: $(1); (2; 3; 4)^*$):

1. **Generating initial rules**. This step applies *rote learning* to cover positive examples and avoid to cover negative examples.
2. **Generalising facts**. This step selects a fact obtained by rote learning and applies *fact subsumption*. If the fact is not subsumed, it applies *folding* with the goal of generating a new, more general, rule that makes no explicit references to the constants occurring in the ABA framework.
3. **Introducing new assumptions**. This step applies *assumption introduction* to any rule obtained by step (2) if it supports the an argument for a negative example.
4. **Learning facts for contraries**. This step applies *rote learning* to generate facts for the contrary of the new assumption introduced by step (3).

Example 3. *Let us consider the background knowledge consisting in the ABA framework F of Example 1, and the following sets of positive and negative examples:*

$\mathcal{E}^+ = \{innocent(mary)\}$
$\mathcal{E}^- = \{innocent(john)\}$

The positive example $innocent(mary)$ is already covered in the unique stable extension of F. However, also the negative example $innocent(john)$ is covered. Thus, by rote learning, ABALearn introduces the new rule:

$$\rho_1.\ guilty(X) \leftarrow X = john$$

which avoids covering $innocent(john)$. By folding twice, rule ρ_1 is replaced by:

$$\rho_2.\ guilty(X) \leftarrow witness\_con(X, Y), person(Y)$$

Now, the positive example $innocent(mary)$ is no longer covered, because $guilty(mary)$ is covered. By assumption introduction, we introduce a new assumption $\alpha(Y)$ with contrary $c\_\alpha(Y)$ and, by rote learning, a new fact:

$$\rho_3.\ guilty(X) \leftarrow witness\_con(X, Y), person(Y), \alpha(Y)$$
$$\rho_4.\ c\_\alpha(Y) \leftarrow Y = alex$$

Finally, by folding, rule ρ_4 is replaced by:

$$\rho_5.\ c\_\alpha(Y) \leftarrow liar(Y)$$

The ABA framework with $\mathcal{R}' = \mathcal{R} \cup \{\rho_3, \rho_5\}$ is a solution of ABA Learning, as it covers all positive examples and no negative example.

ABALearn is implemented in *ASP-ABAlearn* [13] (available at https://github.com/ABALearn/aba_asp). The implementation takes advantage of the existence of a mapping between ABA frameworks under stable extensions and *Answer Set Programming* (ASP) [14], and in particular, it uses SWI-Prolog [15] and the Clingo [16] ASP solver. The mapping of ABA frameworks to ASP is exploited, among other things, for determining the facts to be learned at Steps 1 and 4 of the ABALearn algorithm.

2.3. Datasets

We will use two datasets [10]. The first is based on tort law [11], and consists of cases represented by ten boolean features:

1. **dmg**: Someone has suffered damages by someone else's act.
2. **cau**: The act caused the suffered damages.
3. **vrt**: The act is a violation of someone's right.
4. **vst**: The act is a violation of a statutory duty.
5. **vun**: The act is a violation of unwritten law against proper social conduct.
6. **jus**: There exist grounds of justification.
7. **ift**: The act is imputable to someone because of the person's fault.
8. **ila**: The act is imputable to someone because of law.
9. **ico**: The act is imputable to someone because of common opinion.
10. **prp**: The violated statutory duty does not have the purpose to prevent the damages.

In addition, each case has a label **dut** of the verdict on whether there is a duty to repair damages, that is, the final case outcome.

Hence, the dataset consists of tuples of the following form, where column **dut** represents the outcome of the case.

dmg	cau	vrt	vst	vun	jus	ift	ila	ico	prp	dut
1	1	0	1	0	0	0	0	1	1	1
1	0	0	1	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...

The second dataset is based on welfare benefit applications [12], with cases described in terms of features about applicants' eligibility:

- **Age**: Integer. The applicant's age.
- **Gender**: Categorical. The applicant's gender.
- Con_1**,...,**Con_5: Boolean. Indicates which years the applicant has paid contributions.
- **Spouse**: Boolean. Indicates whether the applicant is the spouse of the patient.
- **Absent**: Boolean. Indicates whether the applicant is absent from the UK.
- **Resources**: Integer. The applicant's amount of capital resources.
- **Patient Type**: Categorical. Indicates whether the patient is an in-patient or an out-patient.
- **Distance (to the hospital)**: Integer. The distance to the hospital in km.

Synthetic Data Generation We have implemented a synthetic data generator to obtain plain text descriptions of cases based on their features and outcomes in the datasets. This is based on the Mistral-Small-24B-Instruct-2501 model[2]. For instance, the generated description of the first case in the tort law dataset above is as follows:

"During the proceedings, the Plaintiff, Ms. Emily Harris, alleged that on January 5, 2022, she suffered severe burns while using a defective electric kettle manufactured by the Defendant, KettleTech Inc. The Plaintiff testified that the kettle, purchased new from a local retailer, malfunctioned when she turned it on, causing hot water to spray onto her. . . "

3. Neuro-argumentative pipeline

3.1. Pipeline overview

As shown in Figure 1, the neuro-argumentative pipeline consists of two main components:

1. a **neural feature extractor**, responsible for extracting a background knowledge, along with positive and negative examples, from labeled legal cases, described as unstructured text, and
2. a **symbolic learning module**, which learns an ABA framework from the background knowledge and the examples using the ABALearn method, and employs the learnt ABA framework to predict the outcome of new cases.

Training Process The dataset, which contains plain text case descriptions labelled with the ground truth feature values and the final case outcome, is split into two parts:

- One subset (3000 cases) is used to train the neural feature extractor, mapping text to logical features.
- The other subset (1000 cases) is used by ABALearn to learn new rules, using the features extracted by the trained extractor alongside the known case outcomes.

[2] https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501

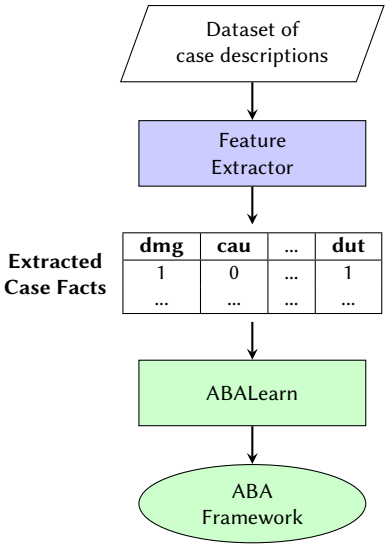


Figure 1: Outline of neuro-argumentative pipeline

Training the neuro-argumentative pipeline is a two-stage process, training each component separately. In the first stage, we train the neural feature extractor to predict the logical features present, using the plain-text case descriptions as input.

Then we can use the trained feature extractor to extract the logical facts from the case descriptions of the second training subset. These predicted facts, along with their associated case outcomes, are used as input to ABALearn to learn new rules and construct an ABA framework.

The output of the training process is the trained feature extractor model and the learned ABA framework, which can be used to make predictions on the outcome of unseen cases using argumentation-based inference.

Inference At inference time, the neuro-argumentative pipeline operates in a fully end-to-end manner. The process is composed of two sequential steps: neural feature extraction and symbolic inference.

1. **Feature Extraction:** Given a plain-text case description, the trained neural feature extractor is used to predict the logical facts relevant to the case.
2. **Symbolic Inference:** The extracted logical facts are added to the learned ABA framework, which uses argumentation-based reasoning to determine the legal outcome for the new case.

3.2. Feature Extraction

This section presents the neural architectures explored for extracting symbolic features from plain-text legal case descriptions.

3.2.1. BERT-based feature extractor

In our baseline architecture, we utilise the pre-trained Legal-BERT [17] encoder to extract features from plain-text legal case descriptions. This implementation was developed only to handle datasets with exclusively boolean features.

Architecture Overview The feature extractor consists of:

1. A shared Legal-BERT encoder, which processes the input case description and produces contextual embeddings.
2. A separate independent multilayer perceptron (MLP) head for each target feature, which predicts binary labels from the [CLS] token embedding produced by Legal-BERT, indicating the presence of the target feature.

Training Procedure During training, the Legal-BERT encoder parameters are frozen to preserve the encoder's pretrained representations and reduce computational overhead. Only the individual MLP heads for each feature are trained using a sigmoid output activation and binary cross-entropy loss.

Each feature head is trained independently. This means that in every training step, each feature's MLP head is updated using the loss computed for that feature only.

Benefits and Limitations This architecture provides a simple and computationally efficient way to extract multiple symbolic features using a shared transformer-based encoder and avoids interactions between heads during backpropagation. It benefits from the generalisation capabilities of a large pre-trained model while keeping the parameter count and training time low.

However, since the encoder is not updated during training, it cannot fully adapt to the specific legal domain, which can limit the feature extractor's accuracy, especially for more abstract or context-dependent features.

3.2.2. Fine-tuning the BERT encoder

Architecture Overview - Updates Building upon the feature extractor described above, this approach retains the same architecture, however, the encoder parameters are no longer frozen. During training, the gradients from the loss function are propagated all the way through the encoder, enabling it to adapt to the specific legal domain, whether that is tort law or welfare benefit applications.

Training Procedure All MLP heads and the Legal-BERT encoder are trained together. Given an input case x with n binary features, the model outputs $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$, and the total loss is calculated as the sum of binary cross-entropy losses across all heads:

$$\mathcal{L} = \sum_{i=1}^{n} \text{BCE}(y_i, \hat{y}_i)$$

where y_i is the ground truth for feature i. Gradients from this loss flow through the MLP heads and into the encoder, allowing the model to learn domain-specific contextual embeddings.

Benefits and Limitations Fine-tuning the Legal-BERT model allows the encoder to adapt to the legal domain and learn better contextual representations, leading to improved classification performance.

However, this approach significantly increases the number of trainable parameters, leading to higher computational

and memory requirements during training. It also introduces a greater risk of overfitting, especially when working with small or imbalanced datasets, as the model may learn to memorise patterns that do not generalise.

3.2.3. Multi-Task Feature Extractor

The previous architectures rely on a single shared encoder, which can limit task-specific learning and make it difficult to handle heterogeneous feature types. To address these challenges, we propose a multi-task feature extractor architecture where each feature is handled by a dedicated model.

Architecture Overview The proposed architecture consists of multiple independent BERT encoders, one for each target feature. Each encoder processes the same input case description but independently learns to represent it in a way that is most useful for predicting its assigned feature. The encoder output is then passed through the corresponding feature-specific MLP head that produces the prediction.

Training Procedure Our implementation supports three types of target features:

1. **Binary:** The MLP head outputs a single neuron with a sigmoid activation function and is optimised using the binary cross-entropy loss.
2. **Categorical:** The MLP head outputs a probability vector over possible classes using softmax activation and categorical cross-entropy loss.
3. **Numerical:** The MLP head outputs a scalar value without activation and is trained using mean squared error (MSE) loss.

Each encoder-head pair is trained independently using only the label corresponding to its target feature. The input case description is shared across all feature models, but there is no parameter sharing between the encoders. At each training step, we calculate the appropriate loss specific to that feature, and use backpropagation only on the corresponding encoder and head.

Benefits and Limitations The main advantage of this architecture is improved performance through task specialisation. This independent training ensures that the learning for one feature does not conflict with another, preserving task-specific representations and gradients, and allowing the encoder to capture subtle patterns relevant to its target.

However, this comes at a cost, as the number of parameters increases linearly with the number of features, resulting in higher memory requirements.

3.3. Symbolic learning

In this section, we show how ABALearn is used as the symbolic component of the pipeline shown in Figure 1. In particular, we show how the features of the cases[3] are encoded to be given as input to ABALearn. Then, we present an excerpt of the learned ABA framework. Finally, we describe the inference process for predicting case outcomes and discuss how the learned ABA framework, in conjunction with the case-specific facts, can be used to provide justifications for each prediction.

The encoding process is shown in Figure 2. Given any case i and feature f in {**dmg**, **cau**, …, **prp**}, if $f(i)$ is 1, then '`f(i)`' is added as a fact to the background knowledge of ABALearn. In addition to background knowledge, ABALearn takes as input a set of positive and negative examples, denoted by the pair $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$. If **dut**$(i)$ is 1, then '`dut(i)`' is added to $\mathcal{E}^+$; otherwise, it is added to $\mathcal{E}^-$.

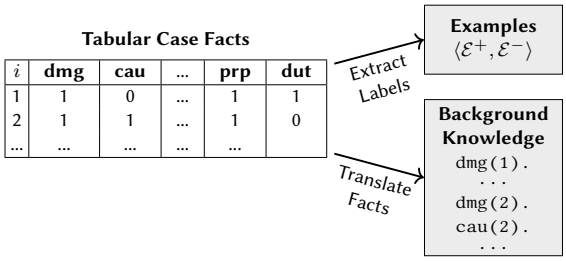


Figure 2: Encoding cases into ABALearn

```
% Facts for case with id 1
dmg(1). vst(1). prp(1).
% Facts for case with id 2
dmg(2). cau(2). vst(2). prp(2).
...
% Command to run ABALearn:
aba_asp('tort.bk',     % (1)
        [dut(1),...],  % (2)
        [dut(2),...]). % (3)
% (1) background knowledge file
% (2) a list of positive examples
% (3) a list of negative examples
```

Given the input ABA framework shown above, ABALearn aims to generate a set of rules that covers all the positive examples and none of the negative examples. This ABA framework learned using ABALearn is shown below.

```
% Learnt rules
dut(A):- alpha_1(A), dmg(A), vst(A), prp(A).
c_alpha_1(A):- dmg(A), cau(A), vst(A), prp(A).
...
% Assumptions
assumption(alpha_1(A)).
...
% Contraries
contrary(alpha_1(A),c_alpha_1(A)) :-
  assumption(alpha_1(A)).
...
```

The learned ABA framework can be used to predict the outcome of new unseen cases. Let us consider a new case 42 from the tort law dataset with the following features:

dmg	cau	vrt	vst	vun	jus	ift	ila	ico	prp
1	1	0	1	0	0	0	0	1	1

The new case is translated into facts using the same process described to encode the background knowledge given as input to ABALearn.

The new facts are then added to the learned ABA framework, thereby getting the encoding shown below.

[3] In the pipeline, these are the features extracted from textual descriptions of the cases. We focus here, for illustration, on the features in the original tort law dataset.

```
% Learnt rules
dut(A):- alpha_1(A), dmg(A), vst(A), prp(A).
c_alpha_1(A):- dmg(A), cau(A), vst(A), prp(A).

% Facts of the new case with id 42
dmg(42). cau(42). vst(42). ico(42). prp(42).
```

We can predict whether there is a duty to repair damages by checking if the claim `dut(42)` belongs to the stable extension of the ABA framework augmented with the new facts. The new facts enable the deduction of `c_alpha_1(42)`, which attacks the argument supporting the deduction of the claim `dut(42)`. Hence, the ABA framework predicts that there is no duty to repair damages.

4. Evaluation

4.1. Feature Extraction

We begin our evaluation by comparing the performance of the feature extraction architectures discussed above. We focus on the tort law dataset, the only one lending itself to all feature extractor architectures, as it contains binary features.

Figure 3 compares the average F1 scores of all architectures on the tort law dataset. The baseline Legal-BERT model achieves an F1 score of 0.66. Fine-tuning the encoder yields a substantial performance gain of 22.8%, while the multi-task architecture adds a modest further improvement of 2.3%.

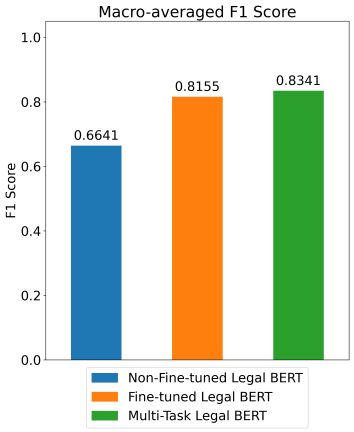


Figure 3: Average feature extraction F1 score for each feature extractor architecture (tort law dataset)

While the small gain from using multiple encoders may suggest that a single Legal-BERT encoder already produces sufficiently rich embeddings for most features, a closer look at per-feature performance reveals a different picture. The multi-task model exhibits more consistent F1 scores across features, whereas the single-encoder model seems to perform well on some features while underperforming on others.

Downstream Task Performance We also assess the feature extractor architectures based on their utility in downstream symbolic learning and classification tasks. Figure 4 presents the pipeline classification F1 scores for predicting the `dut` (duty) label for each feature extractor architecture. The baseline architecture achieves an F1 score of 0.59. Fine-tuning the encoder improves this by 33%, while the multi-task architecture yields an additional 3.5% improvement.

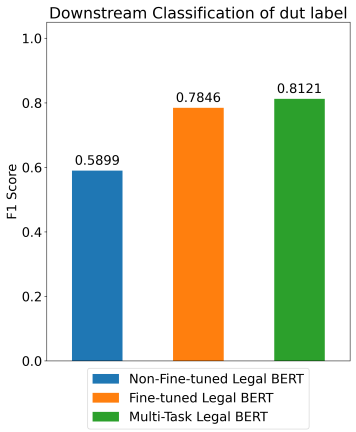


Figure 4: F1 scores for downstream classification of `dut` label using our symbolic module (tort law dataset)

The larger improvements in downstream classification, relative to standalone feature extraction, show the importance of feature quality in symbolic learning. The symbolic module depends on consistent features to learn meaningful rules. Poor-quality features can lead to learning meaningless or misleading rules, exemplifying the "garbage in, garbage out" principle. Even small gains in feature extraction can yield significant benefits in downstream reasoning.

In the pipeline evaluation we adopt the *multi-task architecture* as it demonstrates more consistent performance across features.

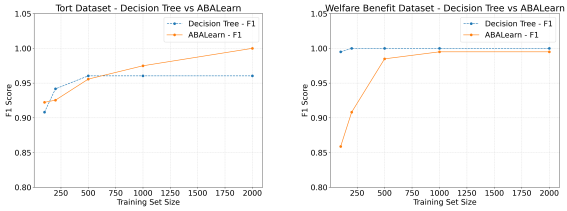
4.2. Symbolic Learning (ABALearn)

We evaluate ABALearn by training it on fragments of the datasets of varying sizes to examine how training data volume affects symbolic learning. We use *ground truth* features to isolate the symbolic module's performance from that of the feature extractor. As a baseline, we compare ABALearn to a decision tree classifier, using F1 score on case outcome predictions.

Figure 5 shows F1 scores across different training set sizes for both datasets. On the tort law dataset, ABALearn closely tracks the performance of the decision tree and begins to outperform it with training sets of 1000 examples or more. In contrast, ABALearn initially underperforms on the welfare benefit dataset when trained on smaller sets, but matches the decision tree's performance once the training size exceeds 500 examples. This slower convergence is likely due to the added complexity of handling numerical features, which significantly expands the logical space that the ABA framework must represent.

The strong F1 scores across both datasets suggest that ABALearn can learn meaningful rules for case outcome classification with relatively limited training examples. This efficiency allows us to allocate more examples to feature extraction. For our pipeline experiments, we fix the symbolic training set size at 1000 examples, which is the training set

size where ABALearn surpasses the decision tree on the tort law dataset and levels off on the welfare benefit dataset.



(a) ABA vs decision tree for the tort law dataset

(b) ABA vs decision tree for the welfare benefit dataset

Figure 5: F1 score comparison of ABALearn models vs decision tree across our two datasets

4.3. Neuro-argumentative Pipeline

Building on the strong individual performance of the feature extractor and symbolic learning modules, we now evaluate the end-to-end neuro-argumentative pipeline, comparing its performance against several BERT-based classifier approaches for our synthetic datasets. Figures 6 and 7 show the case outcome classification F1 scores across the different classifier architectures. For both datasets, the neuro-argumentative pipeline outperforms both the baseline BERT-base and Legal-BERT classifiers, but fails to surpass the performance of their fine-tuned counterparts.

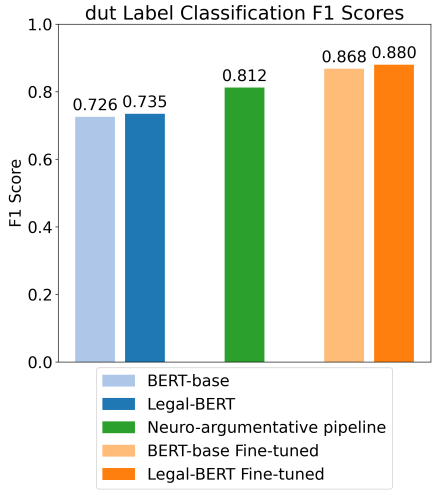


Figure 6: F1 scores for the tort law dataset

While the 10% relative F1 improvement over baseline classifiers on the tort law dataset is notable, the consistently strong performance across all models suggests this domain may be too simple to pose a meaningful challenge. In contrast, the welfare benefit dataset is more challenging, causing baseline classifiers to struggle. Here, our neuro-argumentative pipeline achieves a 39.5% relative F1 improvement over the domain-specific Legal-BERT classifier and performs within 5% of the fine-tuned model.

It is worth noting that the pipeline's F1 score is significantly lower than that of the standalone symbolic module when it is given clean, ground truth features, which indicates that the main performance bottleneck lies in feature

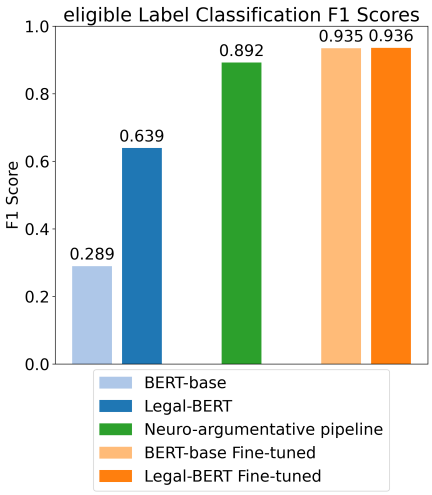


Figure 7: F1 scores for the welfare benefit dataset

extraction. This means that improving the accuracy of the extracted features could lead to substantial gains in symbolic reasoning and overall pipeline performance.

4.4. Explainability

One of the main objectives of this paper is to introduce an explainable system for legal decision-making. While Assumption-Based Argumentation is, in principle, an inherently transparent and explainable argumentation framework, the symbolic rules learned by ABALearn in practice may fall short of this ideal.

By using ABA, users can trace which rules and assumptions lead to a particular conclusion, providing a transparent decision-making process We show below some of the rules learned by the neuro-argumentative pipeline.

```
% Learned rules
dut(A) :- alpha_1(A), dmg(A), ico(A), vun(A).
dut(A) :- alpha_2(A), cau(A), dmg(A), ila(A).
...
c_alpha_29(A) :- dmg(A), ico(A), ift(A),
            ila(A), jus(A), prp(A),
            vst(A), vun(A).
```

We can see that the learned ABA frameworks can sometimes include a large number of assumptions and contraries, and rules with long and complex bodies, which complicates the overall structure of the framework. The reasoning behind a prediction can become difficult to follow, and the large number of rules makes it challenging to isolate those that are relevant to a specific case outcome. These issues appear even when learning the framework using clean, ground truth feature values and are even worse when using features extracted from the case descriptions, as noise and variability in the input data lead ABALearn to introduce workarounds during learning, ultimately resulting in rules that are less coherent and more difficult to interpret.

Although the foundation for explainability is present in the form of the learned rules, further work is needed to improve the interpretability thereof and make them more accessible to human users.

5. Conclusion

This work presents a neuro-argumentative pipeline for legal decision-making that combines the flexibility of neural feature extraction with the transparency of assumption-based argumentation. The pipeline demonstrates good classification performance, with particularly strong results in the symbolic learning component, which achieves high theoretical accuracy when provided with clean feature inputs.

Future Work As demonstrated in our experiments, the quality of feature extraction significantly affects both symbolic learning and overall pipeline performance. While our proposed architecture performs well on many binary features, it struggles with more nuanced features, leading to suboptimal performance across certain feature types. Exploring new feature extraction architectures to improve upon the performance of the proposed ones could lead to large overall improvements in the pipeline's classification performance.

Additionally, even though the rules learned by ABALearn can be inspected to trace the reasoning behind a prediction, their interpretability remains limited. The learned frameworks are often large and complex, even when derived from a relatively small number of features. Future work could focus on developing automated simplification tools to reduce redundancy and improve clarity, making the learned rules more accessible to users. For example, merging logically equivalent rules or pruning unnecessary literals could yield more concise representations.

Finally, while the current pipeline has demonstrated strong performance on synthetic datasets, futher work is required to apply it to real-world datasets that are often characterised by incomplete, noisy, or inconsistently structured inputs.

This paper demonstrates the promise of combining neural learning with symbolic argumentation for legal classification tasks. Notably, the results suggest that explainability and performance do not need to be mutually exclusive. This work lays a foundation for further exploration of neuro-symbolic systems in the legal domain, where transparency in the decision-making process is paramount.

Acknowledgments

We thank support from the Royal Society, UK (IEC\R2\222045). Toni was partially funded by the ERC (grant agreement No. 101020934) and by J.P. Morgan and the RAEng, UK, under the Research Chairs Fellowships scheme (RCSRF2021\11\45). De Angelis and Proietti were supported by the MUR PRIN 2022 Project DOMAIN funded by the EU – NextGenerationEU (2022TSYYKJ, CUP B53D23013220006, PNRR, M4.C2.1.1), by the PNRR MUR project PE0000013-FAIR (CUP B53C22003630006), and by the INdAM - GNCS Project *Argomentazione Computazionale per apprendimento automatico e modellazione di sistemi intelligenti* (CUP E53C24001950001). De Angelis and Proietti are members of the INdAM-GNCS research group. Finally, we would like to thank the anonymous reviewers for their constructive remarks.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] J. Lai, W. Gan, J. Wu, Z. Qi, P. S. Yu, Large language models in law: A survey, 2023. arXiv:2312.03718.

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[4] S. Vatsal, A. Meyers, J. E. Ortega, Classification of US Supreme Court Cases using BERT-Based techniques, 2023. arXiv:2304.08649.

[5] M. Dahl, V. Magesh, M. Suzgun, D. E. Ho, Large legal fictions: Profiling legal hallucinations in large language models, Journal of Legal Analysis 16 (2024) 64–93. doi:10.1093/jla/laae003.

[6] A. Blair-Stanek, N. Holzenberger, B. Van Durme, Can GPT-3 perform statutory reasoning?, in: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law, ICAIL '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 22–31. doi:10.1145/3594536.3595163.

[7] A. Bondarenko, P. Dung, R. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artif. Intell. 93 (1997) 63–101. doi:10.1016/S0004-3702(97)00015-5.

[8] K. Cyras, X. Fan, C. Schulz, F. Toni, Assumption-based argumentation: Disputes, explanations, preferences, FLAP 4 (2017). URL: https://www.collegepublications.co.uk/downloads/ifcolog00017.pdf.

[9] M. Proietti, F. Toni, Learning assumption-based argumentation frameworks, 2023. arXiv:2305.15921.

[10] C. Steging, S. Renooij, B. Verheij, T. Bench-Capon, Arguments, rules and cases in law: Resources for aligning learning and reasoning in structured domains, Argument & Computation 14 (2023) 235–243.

[11] B. Verheij, Formalizing arguments, rules and cases, in: Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law, ICAIL '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 199–208. doi:10.1145/3086512.3086533.

[12] T. Bench-Capon, Neural networks and open texture, in: Proceedings of the 4th International Conference on Artificial Intelligence and Law, ICAIL '93, Association for Computing Machinery, New York, NY, USA, 1993, p. 292–297. doi:10.1145/158976.159012.

[13] E. De Angelis, M. Proietti, F. Toni, Learning brave assumption-based argumentation frameworks via ASP, in: Proceedings of ECAI 2024, volume 392 of *FAIA*, IOS Press, 2024, pp. 3445–3452. doi:10.3233/FAIA240896.

[14] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: Proceedings IJCSLP, MIT Press, 1988, pp. 1070–1080. URL: http://www.cs.utexas.edu/users/ai-lab?gel88.

[15] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager, SWI-Prolog, Theory and Practice of Logic Programming 12 (2012) 67–96. doi:10.1017/S1471068411000494.

[16] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Clingo = ASP + Control: Preliminary report, 2014. arXiv:1405.3694.

[17] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, I. Androutsopoulos, LEGAL-BERT: The Muppets straight out of law school, 2020. arXiv:2010.02559.