A Logic-Based Explainable Framework for Relation Classification of Human Rights Violations

Bimal Bhattarai[1,*], Rupsa Saha[1], Ole-Christoffer Granmo[1], Vladimir Zadorozhny[2] and Jiawei Xu[2]

[1]*University of Agder, Grimstad, Norway*
[2]*University of Pittsburgh, USA*

Abstract
Using a Relational Tsetlin Machine (RTM) for analysis of semi-structured data allows the use of inherent relational structures present in natural language text to get an explainable classification of data. A finite Herbrand model derives Horn Clauses from the model, which are simple yet powerful logical tools that can build an abstract view of the world. We use the same to analyze human rights violation data. We show concretely how natural language can be transformed into a relational structure, and further use the Relational Tsetlin Machine to not only classify incidents as serious and non-serious violations but also explore the patterns learned by the RTM in order to arrive that those decisions. Furthermore, the distilled Horn Clauses show a precise understanding of the concepts involved without the drawback of textual ambiguity.

Keywords
Tsetlin Machine (TM), Explainable, Natural Language Processing, Relational TM, Logic-based Reasoning

1. Introduction

Training Artificial Intelligence (AI) to answer natural language questions is a crucial part of the quest for encoding a human-equivalent understanding of the world in machines. Massive structured Knowledge Bases (KBs), such as Freebase [1] have been the cornerstone of such efforts. A common challenge lies in the appropriate interpretation of language by AI agents, both for building the KBs themselves (from existing natural language resources), as well as to identify the information required and provided from questions. The need for abstraction from specific (and limited) examples to build concepts and rules about the world, in general, is another challenge. What is a standard inductive reasoning problem if the KB is completely consistent and error-free, becomes extremely different when allowing for all the uncertainty, indeterminacy, errors, exceptions, and conflicts that are present in real-world data, more so when the data has been extracted and structured by AI even partially.

Various methods currently in use for Question Answering (QA) are broad: (a) Tokenization, POS tagging, parsing, and other linguistic approaches to derive a precise query from natural language questions, which can further be deployed on a structured database; (b) Support Vector Machines, Bayesian Classifiers, Maximum Entropy models, and similar statistical approaches, trained on large amounts of data; (c) Identifying and matching surface-level patterns with templates for response generation. Often hybrid approaches are preferred for higher performance. But most QA systems suffer from a lack of ability to generalize, and either have restrictive use cases or require massive amounts of knowledge. The absence of explanations of decisions taken by models also makes it difficult to identify problem areas and offer resolutions or improvements. [2, 3].

Tsetlin machines (TMs) [4] use propositional logic structures to build human-readable reasoning patterns from data. TMs's pattern recognition capabilities have been successfully demonstrated in natural language understanding [5, 6, 7, 8, 9, 10], though none have explored logical decision making. The propositional clauses constructed by a TM have high discriminative power and constitute a global description of the task learnt [11]. Apart from maintaining accuracy comparable to state-of-the-art machine learning techniques, the method also has provided a smaller memory footprint and faster inference than more traditional neural network-based models [12, 13, 14, 15, 16]. Furthermore, [17] shows that TMs can be fault-tolerant, able to mask stuck-at faults. However, although TMs can express any propositional formula by using a disjunctive normal form, first-order logic is required to obtain the computing power equivalent to a universal Turing machine. The more recently proposed Relational Tsetlin Machine introduces a *first order* TM framework with Herbrand semantics, with an eye towards QA applications [18].

In this paper, we aim to use the RTM model to approach categorization and QA on a Human Rights Violation

21st International Workshop on Nonmonotonic Reasoning, September 2–4, 2023, Rhodes, Greece
*Corresponding author.
bimal.bhattarai@uia.no (B. Bhattarai); rupsa.saha@uia.no (R. Saha); ole.granmo@uia.no (O. Granmo); vladimir@sis.pitt.edu (V. Zadorozhny); jix20@pitt.edu (J. Xu)

dataset, showcasing explainability with non-recursive first-order Horn clauses built from specific examples.

2. Background

2.1. Propositional Tsetlin Machine

The TM, first proposed in [4], is a revolutionary technique to pattern classification, regression, and novelty detection [19, 20, 6, 7]. The base unit of a TM is called a Tsetlin automaton (TA), which learns the best action from a set of available actions in its environment. A "regular" TM can also be referred to as a Propositional TM, due to the nature of its input and output operations.

In a "Propositional" TM, a team of base TAs collectively generates propositional formulas using conjunctive clauses. Its input is of the vector form $X = (x_1, \dots, x_o)$, and the TM learns clauses that decide if an input is to be in class $y = 0$ or $y = 1$. The learnt clauses are conjunctive combinations of elements from a subset of a literal set made of input features and their respective negations $\bar{x}_k = \neg x_k = 1 - x_k$. Each clause can be represented as:

$$C_j(X) = \bigwedge_{l_k \in L_j} l_k = \prod_{l_k \in L_j} l_k. \quad (1)$$

E.g., the clause $C_j(X) = x_1 \wedge x_2 = x_1 x_2$ consists of the literals $L_j = \{x_1, x_2\}$ and outputs 1 iff $x_1 = x_2 = 1$.

The number of clauses used is a user-defined parameter m. Each new configuration of clauses created is subjected to feedback, which controls the distribution of frequently occurring patterns, as well as increasing the discriminating power of individual patterns. Of the total number of clauses, half vote in favor of $y = 1$ i.e., positive polarity clauses (C_j^+), whereas the other half vote in favor of $y = 0$ i.e., negative polarity clauses (C_j^-). Classification is performed based on a majority vote using equation 2 and the unit step function: $\hat{y} = u(v) = 1 \textbf{ if } v \geq 0 \textbf{ else } 0$ (for details, see [19]).

$$v = \sum_{j=1}^{m/2} C_j^+(X) - \sum_{j=1}^{m/2} C_j^-(X). \quad (2)$$

E.g. the XOR-relation can be encoded as the classifier $\hat{y} = u\left(x_1\bar{x}_2 + \bar{x}_1 x_2 - x_1 x_2 - \bar{x}_1\bar{x}_2\right)$. The TM leverages a team of TA for learning, one TA per literalsl_k in L. Each TA performs one of two actions - Include or Exclude- and determines whether to include the literal l_k assigned in its clause. TM encompasses an online learning system that processes one training example X, y at a time. The TA generates a new configuration of clauses $C_1^+, \dots, C_{m/2}^-$, before computing a voting total v. Following that, feedback is distributed statistically to each TA team. The difference $epsilon$ between the clipped voting total v and a user-defined voting threshold T determines the likelihood that each TA team will get feedback. Take note that the voting amount is normalized by clipping the voting sum. For $y = 1$, the voting target is T, whereas for $y = 0$, the voting target is $-T$. Observe that when the vote aggregate approaches the user-specified threshold, the likelihood of reinforcing a sentence rapidly decreases to zero. This guarantees that clauses are distributed evenly throughout the frequently occurring patterns, rather than omitting some and focusing only on others. The TM makes use of both Type I and Type II feedback. Type I feedback is intended to generate frequent patterns and Type II feedback is intended to strengthen the discriminating capacity of the patterns (for details, see [19]). When $y = 1$, **Type I feedback** is supplied stochastically to clauses with positive polarity; when $y = 0$, **Type I feedback** is given to clauses with negative polarity. Each clause strengthens its TA, in turn, depending on the following criteria: (1) its output $C_j(X)$; (2) the TA's action — Include or Exclude; and (3) the value of the literal l_k allocated to the TA. Type I feedback is governed by two rules:

- When $C_j(X) = 1$ **and** $l_k = 1$ the *Include* is rewarded and *Exclude* is penalized with probability $\frac{s-1}{s}$. This reinforcement is powerful (triggered with high probability) and causes the clause to recall and refine the pattern it detects in X.[1]
- When $C_j(X) = 0$ **or** $l_k = 0$ the *Include* is penalized and *Exclude* is rewarded with probability $\frac{1}{s}$. This reinforcement is weak (activated with low probability) and coarsens infrequent patterns, hence increasing their frequency.

As mentioned before, the user-configurable parameter s determines pattern frequency; a larger s results in fewer patterns.

When $y = 0$, **Type II feedback** is supplied stochastically to sentences with positive polarity; when $y = 1$, **Type II feedback** is given to clauses with negative polarity. Whenever $C_j(X) = 1$ **and** $l_k = 0$, it penalizes *Exclude*. Thus, this feedback generates literals for differentiating between $y! = 0$ and $y = 1$ by evaluating the clause to 0 when confronted with its rival class.

While this vanilla TM setup operates on propositional input variables $X = (x_1 \dots, x_o)$, to generate propositional conjunctive clauses, the Relational Tsetlin Machine (RTM), described next, processes relations to generate Horn clauses.

3. Relational Tsetlin Machine

The work in [18] introduced the RTM as an extension to the vanilla TM, encoding relations found in natural language using a logic-based representation for the TM. The notion of RTM is based on a logic program using a

[1]Take note that when true positives are boosted, the probability $\frac{s-1}{s}$ is replaced by 1.

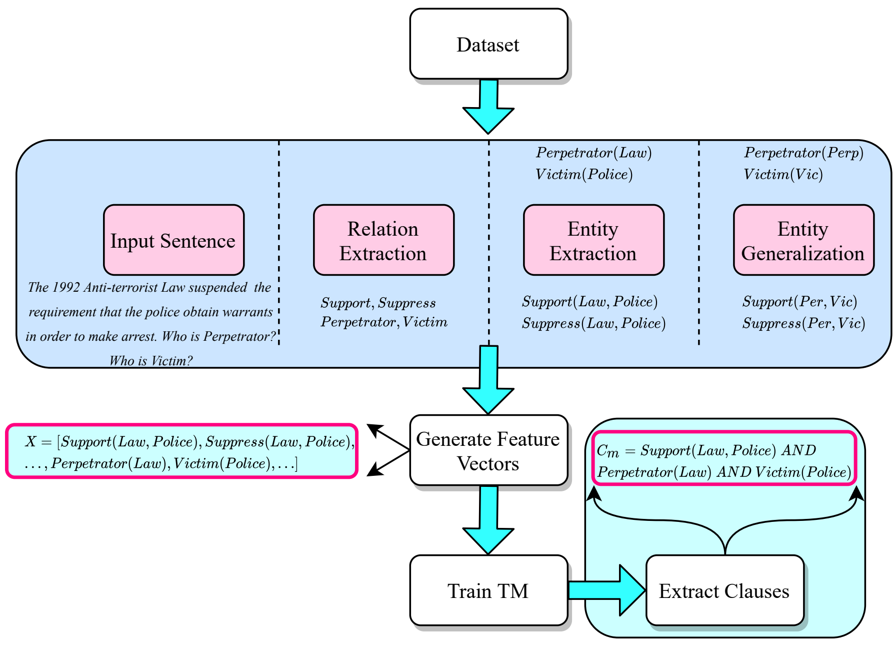
Dataset
Input Sentence
The 1992 Anti-terrorist Law suspended the requirement that the police obtain warrants in order to make arrest. Who is Perpetrator? Who is Victim?
Relation Extraction
Support, Suppress
Perpetrator, Victim
Perpetrator(Law)
Victim(Police)
Entity Extraction
Support(Law, Police)
Suppress(Law, Police)
Perpetrator(Perp)
Victim(Vic)
Entity Generalization
Support(Per, Vic)
Suppress(Per, Vic)
X = [Support(Law, Police), Suppress(Law, Police), ..., Perpetrator(Law), Victim(Police), ...]
Generate Feature Vectors
C_m = Support(Law, Police) AND Perpetrator(Law) AND Victim(Police)
Train TM
Extract Clauses

Figure 1: The Relational TM in operation with pipeline

finite *Herbrand model* [21, 22]. The ability to characterize learning using Horn clauses is particularly beneficial since Horn clauses are both simple and strong enough to describe any logical formula [22].

The RTM is a three-step process based on mapping the learning problem to a pattern recognition problem using vanilla TM:

- By mapping relations to propositional inputs, a method for dealing with relations and constants is devised. We begin with Horn clauses without variables. Let a set of constants $A = \{a_1, a_2, \dots, a_q\}$ be finite and a set of relations $R = \{r_1, r_2, \dots, r_p\}$ of arity $w_u \geq 1, u \in \{1, 2, \dots, p\}$ form the finite Herbrand base $HB = \{r_1(a_1, a_2, \dots, a_{w_1}), r_1(a_2, a_1, \dots, a_{w_1}), \dots, r_p(a_1, a_2, \dots, a_{w_p}), r_p(a_2, a_1, \dots, a_{w_p}), \dots\}$ consisting of all q^w ground atoms that can be expressed using A and R. Additionally, we have a logic program P with program rules defined as non-recursive Horn clauses. Each Horn clause has the following form:

$$H_0 \leftarrow H_1, H_2, \cdots, H_d. \tag{3}$$

Here, $H_l, l \in \{0, \dots, d\}$, is an atom $r_t(V_1, V_2, \dots, V_{w_t})$ with variables $V_1, V_2, \dots, V_{w_t}$, or its negation $\neg r_t(V_1, V_2, \dots, V_{w_t})$. The arity of r_t is denoted by w_t. We map every atom in HB to a propositional input x_k, getting the propositional input vector $X = (x_1, \dots, x_o)$ (cf. Section 2.1).
- The horn clauses with variables are introduced to decouple the TM from the constants, resulting in a more compact representation than is possible with merely propositional clauses. To illustrate this, assume $\mathcal{V} = \{V_1, V_2, \dots, V_v\}$ be v variables representing the constants occurring in an observation $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$. Here, v is the maximum number of distinct constants required for each observation $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$, each needing its own variable.

We map the atoms to propositional inputs to create a vanilla TM learning problem. That is, each propositional input x_k represents a unique atom with a specific variable configuration: $x_k \equiv r_t(V_{\alpha_1}, V_{\alpha_2}, \dots, V_{\alpha_{w_t}})$, with w_t being the arity of r_t. As a result, the number of constants in A has no effect on the number of propositional inputs x_k required to describe the problem. Rather than that, this is determined by the number of variables in $\mathcal{V}$ as well as the number of relations in R. For a particular observation $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$, we first replace the constants in $\tilde{\mathcal{Y}}$ with variables, from left to right. Accordingly, the corresponding constants in $\tilde{\mathcal{X}}$ are also replaced with the same variables. Remaining constants in $\tilde{\mathcal{X}}$ are arbitrarily replaced with additional variables. The propositional input vector is regenerated.
- A convolution approach with a standard TM as described in Section 2.1 handles a large number of alternative constant-to-variable mappings as a standard pattern recognition problem.

The RTM is summarized in Algorithm 3.

Decoupling the constants, i.e. step 2 above, serves to generalize the clause-based relations, and to allow quicker learning with less input. By seeking Horn clauses

Algorithm 1 Relational TM

```
    input Convolutional Tsetlin Machine TM, Example pool $D$, Number of training rounds $t_e$
1:  procedure TRAIN(TM, $D, t_e$)
2:      for $i \leftarrow 1, \ldots, t_e$ do
3:          $(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) \leftarrow \mathrm{ObtainTrainingExample}(D)$
4:          $A' \leftarrow \mathrm{ObtainConstants}(\tilde{\mathcal{Y}})$
5:          $(\tilde{\mathcal{X}}', \tilde{\mathcal{Y}}') \leftarrow \mathrm{VariablesReplaceConstants}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}, A')$
6:          $A'' \leftarrow \mathrm{ObtainConstants}(\tilde{\mathcal{X}}')$
7:          $Q \leftarrow \mathrm{GenerateVariablePermutations}(\tilde{\mathcal{X}}', A'')$
8:          $\mathrm{UpdateConvolutionalTM}(\mathrm{TM}, Q, \tilde{\mathcal{Y}}')$
9:      end for
10: end procedure
```

that employ variables rather than constants, we can prioritize atoms above variable configurations. If z is the largest number of unique constants involved in any particular observation, the number of atoms is bound by $O(z^w)$, where w is the largest arity of relations in R. Given the possibility of different methods to assign variables to constants, the preceding technique may result in duplicate rules. One may wind up with identical rules with just a syntactic variation, i.e., the same rules represented using different variable symbols. To prevent the creation of unnecessary rules, the Relational TM generates all feasible permutations of variable assignments. Finally, we conduct a convolution over the permutations to process them.

4. Using RTM to Analyse Data on Human Rights Violations

In this section, we present a case study for analyzing human rights violation data using the previously described Relational Tsetlin Machine. The data is derived from the PULSAR system [23].

PULSAR (Parsing Unstructured Language into Sentiment-Aspect Representations) aims at parsing human rights reports into sentence-level judgments and linking judgments to specific aspects of human rights. It handles a large corpus of yearly reports from human rights non-governmental organizations (HRNGOs), as well as the State Department and Amnesty International Annual Reports.

The parser uses aspect-based sentiment analysis (ABSA) to separate judgments (sentiments) from things being judged (aspects). As an example, consider the following sentence:

- *I like (sentiment, judgment) status of human rights in this country (aspect).*

PULSAR refines this approach by mapping specific words or phrases to specific parts of judgments and aspects using the following ontology:

- Perpetrator - what entity is being judged?
- JPr - judgment of presence or absence
- JQu - judgment of quantity or intensity
- JGT - judgment of giving or taking
- SAVP - what is the specific aspect being judged (protection/violation)
- GAHR - general aspect of human rights
- Victim - who is the victim of the action?
- Negation - words that negate the meaning of the sentence (usually *not*)

Using the above ontology PULSAR produces a series of judgments about human rights violations/protections, e.g.:

- *Security forces (Perpetrator) are (JPr) regularly (JQu) participating (JGT) in the abducting (SAVP, GAHR) of minorities (Victim).*

We use PULSAR output to produce either a ‘suppress’ or a ‘support’ relation between two entities, followed by a query. The entities are the subject and the object of the relation present in the sentence. In the case of a ‘suppress’ relation, Entity A and B are ‘Perpetrator’ and ‘Victim’. In a ‘support’ relation, they are ‘Supporter’ and ‘Beneficiary’. Based on this relation, one needs to identify whether a violation of rights has occurred.

To assist us in constructing the task, we make the following assumptions:

1. All statements only include information about relation “Support” or “Suppress”.
2. All questions are limited to information on the relation of subject and object.
3. Both “Support” and “Suppress” each entail two entities, such that $Support\ /\ Suppress(e, f) : e \in \{subject\}, f \in \{object\}$

Table 1
Relations and Entities in 2 examples

Sentence	*Relation*	*EntityA (Subj.)*	*EntityB (Obj.)*
The 1992 Anti-terrorist Law suspended the requirement that the police obtain warrants in order to make arrest.	**Support**	**Law**	**Police**

Support (Supporter-Beneficiary)

Sentence	*Relation*	*EntityA (Subj.)*	*EntityB (Obj.)*
The Government contends disciplinary action against police who are guilty of violating human rights.	**Suppress**	**Govt.**	**Police**

Suppress (Perpetrator-Victim)

Table 2
Relation Reduction by 2-part Entity Generalization

Sentence	*Relation*	*Entities*	*Entity Gen. I*	*Entity Gen. II*
The 1992 Anti-terrorist Law suspended the requirement that the police obtain warrants in order to make arrest.	**Support**	**Law, Police**	**Support**(X,)	**Support**(X,)
The Government contends disciplinary action against police who are guilty of violating human rights.	**Suppress**	**Govt, Police**	**Suppress (Govt,Police)**	**Suppress(Y,B)**
Who is supported by law?	**Query**	**Law, ?**	**Query-Object**(X, ?)	**Query-Object**(X, ?)

4. "Support" or "Suppress" is a time-bound relation, its impact is overtaken by a subsequent comparable action.

The first step is to convert the text into a machine-understandable relational representation. A relation in this context refers to a relationship between two (or more) elements of a text. Once identified, the entities may be generalized for further reduction of search space. Finally, the relations (whether reduced via generalization or not) are used as input features for a standard TM setup for the categorization of the text.

Relation Extraction: Our text is composed of simple sentences, each of which has a sentence containing only one relation. The relations found in the query are determined using other features of a dataset and are linguistically related. Table 1 illustrates examples of Relation Extraction on our dataset. Each statement is associated with the relation of either "Support" or "Suppress", while the query is associated with either the Subject or the Object. Using the query, we can extract the relation as well as identify the entity.

The relation between the entities are calculated based on positive valence count (V_{pos}) and negative valence count (V_{neg}). The assignment of the relation (Rel_A) is done as shown in Equation 4:

$$Rel_A = \begin{cases} Support & \text{if } V_{pos} > 0 \textbf{ and } V_{neg} = 0, \\ Suppress & \text{if } V_{pos} = 0 \textbf{ and } V_{neg} > 0, \\ Neutral & \text{if } V_{pos} = 0 \textbf{ and } V_{neg} = 0. \end{cases} \quad (4)$$

The relational factor is given by the difference in variance $\gamma = V_{pos} - V_{neg}$, resulting in:

$$Rel_A = \begin{cases} Support & \text{if } \gamma > 0, \\ Suppress & \text{if } \gamma < 0, \\ Neutral & \text{if } \gamma = 0. \end{cases} \quad (5)$$

Entity Extraction: After identifying the relations, we must determine the textual components that contribute to the formation of those relationships. This enables us to enhance the representation with restrictions, allowing the RTM to learn rules that most accurately reflect action and consequences in a logical manner. The retrieved entities can be combined with the information about the external world knowledge to create a richer representation. For example, the concept that the subject and object in a 'Suppress' relation can also be termed as 'Perpetrator' and 'Victim', is an example of external knowledge. Notably, as per Fig. 1, it is not feasible to begin answering the query until both Relation Extraction and Entity Extraction have been completed. Additionally, knowledge of the relation enables us to filter down the potential entities that will successfully respond to the query.

Entity Generalization: One disadvantage of the relational representation is that as more sentences are analyzed, the number of potential relations grows exponentially. One strategy to limit the spread is to relegate particular entities to a more generic identification. Consider the following two instances from Table 1: "Text: The 1992 Anti-terrorist Law suspended the requirement that the police obtain warrants in order to make an arrest. Q1: Who is Supporter? Q2: Who is Beneficiary?" and "Text: The Government contends disciplinary action against police who are guilty of violating human rights. Q1: Who is Perpetrator? Q2: Who is Victim?". Processing the texts as per the previous section, we end up with six distinct relations: Support(Law, Police), Supporter(Law), Beneficiary(Police), Suppress(Government, Police), Perpetrator(Government), Victim(Police). However, in order to answer any of the queries, we simply need the relations associated with that query. As a result, we can reduce both sentences to four relations: Support(subj$_1$, obj$_1$), Suppress(subj$_2$, obj$_2$), Subj(subj$_1$ / subj$_2$), and Obj(obj$_1$ / obj$_2$). Prioritization requires that the entities included in the query relation be generalized first. Any instances of those entities in the relations before the query are substituted. The entities present in other relations are subsequently substituted with variables irrelevant for answering the query (Table 2).

4.1. Classification

Having reduced the text into a relational feature set, we now proceed to the classification task. We train RTM with 650 clauses, threshold $= 600$, and specificity $= 25$ for 200 epochs. The LSTM with 100 memory units, 100 embedding sizes, spatial dropout, and softmax activation is used. The CNN with 32-dimensional embedding, convolution layer with max pooling, and sigmoid activation is used. Table 3 shows that the RTM outperforms LSTM, CNN, and vanilla TM considerably over 10 independent runs. The vanilla TM, due to its disregard for the relationship between variables, exhibits poor performance as it lacks the ability to generalize these relationships. Both LSTM and CNN demonstrate comparable performance. However, the utilization of the Herbrand model and horn clause in RTM enhances its robustness and generalization capability, surpassing the performance of other methods by approximately 20%.

4.2. Explainability

One of the major reasons for choosing TMs for this task is the inherent explainable structure built into the clauses produced during learning. At the end of the training, the relations captured by the TM constitute a global picture of the learning, i.e. what the model has learned in general. Additionally, the global picture can be seen as a description of the task itself, as understood by the machine. We also have access to a local snapshot that is unique to each input instance. This contains just the clauses that describe the relations associated with a particular instance.

We use instances from a human rights dataset to showcase our work. The dataset contains human rights-related sentences. Each sentence constitutes a perpetrator, a victim, and a query related to them. The relation that exists between them is either support or suppress. Based on this relation we determine whether the rights of the victim are violated and we put them in the respective label of "severity" or "non-severity". We show the clauses obtained for one such example chosen from the dataset:

Input: *The Government maintained that this waiting period was necessary to determine whether a woman may still be carrying the child of her former spouse.*
Output: no violation

To assist us in constructing the task, we make the following assumptions:

1. All statement sentences include just information about the relation "Support" or "Suppress".
2. All questions are limited to information on the relation between perpetrator and victim.
3. Relation "Support" entails two entities, such that $Support\ /\ Suppress(e, f) : e \in \{perpetrator\}, f \in \{victim\}$
4. Relation "Suppress" entails two entities, such that $Support\ /\ Suppress(e, f) : e \in \{perpetrator\}, f \in \{victim\}$
5. "Support" or "Suppress" is a time-bound relation, its impact is overtaken by a subsequent comparable action.

After Relation and Entity Extraction:

Input $=>$ Support(Govt., woman), Support(Govt., child), NotSuppress(Govt., woman), Not Suppress(Govt., child), Query (Subject), Query (Object). [2]

Clauses Without Entity Generalization: The complexity of tasks depends upon the collection of sentences from which the model has to identify the relations. For our experiment, we number each input for two potential relations (i.e., support or suppress). Based on this relation the classification is done into one

[2]Due to the fact that the TM requires binary features, each input is transformed to a vector with each element representing the existence (or lack) of the relationship instances.

Table 3
Performance comparison of RTM with baseline algorithms with Entity Generalization

Metrics	RTM		Vanilla TM		LSTM	CNN
	Mean	Max	Mean	Max		
Accuracy	94.71	96.34	68.22	70.65	73.9	74.77
F1 Score	0.940	0.958	0.49	0.60	0.72	0.73
Precision	0.937	0.959	0.54	0.61	0.73	0.74
Recall	0.945	0.970	0.70	0.83	0.74	0.75

Table 4
Horn Clause Representation of Example described in Subsec. 4.2

1	Perpetrator(Government)
2	Victim(Woman)
3	Victim(Child)
4	Support(Government,Woman)
5	Support(Government,Child)
6	$Support(Govt, Woman) \leftarrow Subj.(Govt),\ Obj(Woman), NotSuppress(Govt, Woman)$
7	$Support(Govt, Child) \leftarrow Subj.(Govt),\ Obj(Child), NotSuppress(Govt, Child)$

of two labels (i.e., severe or non-severe). The initial stage in preparing data for the TM is to reduce the input to relation-entity bindings. These bindings comprise the feature set against which the TM is trained.
Clauses are of the form: *Support(Govt, woman) AND Support(Govt, child) AND NotSuppress(Govt, woman) AND Query(Govt) AND Query(Child).*

Clauses With Entity Generalization: We follow the same procedure for making relation-entity binding for relational input features as explained above. Following that, we group the features by entity type to generalize the information. After all, entities are replaced with general placeholders, we train TM with binary features having general entities as features.

Performing Entity Generalization:

Input => Support(Subj1, Obj1), Support(Subj2, Obj2), NotSuppress(Subj1, Obj1), NotSuppress(Subj1, Obj2), Q(Subj1), Q(Obj1 / Obj2).

Clauses are of the form: *Query(Subj1) AND Query(Obj2) AND Support(Subj1, Obj1) AND Support(Subj1, Obj2) AND NotSuppress(Subj1, Obj2) AND NotSuppress(Subj1, Obj1).*

These clauses offer a more compact view of the task without distractions from unimportant constants.

5. Horn Clause Representation

Horn clause representation of the above example is given in Table 4.
Using generalization clauses 6 and 7 can be further replaced by the following single clause :

$$Support(Subj, Obj) \leftarrow Supporter(Subj), Beneficiary(Obj), notSuppress(Sub, Obj).$$

Characterizing learning using Horn clauses is interesting because they are simple but powerful enough to describe any logical formula [22]. Multiple learned Horn Clauses can form a deductive framework on a dataset.

6. Conclusion

A Relational Tsetlin Machine is used to reduce the text to relational input and classify human rights violations. In many real-world datasets, especially those with polarizing information such as human rights, it is imperative to gain a deeper understanding of the classification logic behind the actual classes. While the information can hide behind ambiguous language, the framework must have mechanisms to reduce the ambiguity as much as possible, so that the resultant classification is easy and straightforward to interpret. The use of RTM allows us to obtain precise explanations in terms of Horn Clauses that can form the basis of extended logical frameworks, without compromising on accuracy.

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1247–1250.

[2] M. A. C. Soares, F. S. Parreiras, A literature review on question answering techniques, paradigms and systems, Journal of King Saud University-Computer and Information Sciences 32 (2020) 635–646.

[3] A. M. Pundge, S. Khillare, C. N. Mahender, Question answering system, approaches and techniques: a review, International Journal of Computer Applications 141 (2016) 0975–8887.

[4] O.-C. Granmo, The tsetlin machine - a game theoretic bandit driven approach to optimal pattern recognition with propositional logic, ArXiv abs/1804.01508 (2018).

[5] R. K. Yadav, L. Jiao, O.-C. Granmo, M. Goodwin, Human-level interpretable learning for aspect-based sentiment analysis, in: The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), 2021.

[6] B. Bhattarai, O.-C. Granmo, L. Jiao, Measuring the novelty of natural language text using the conjunctive clauses of a tsetlin machine text classifier, in: Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,, 2021, pp. 410–417. doi:10.5220/0010382204100417.

[7] B. Bhattarai, O.-C. Granmo, L. Jiao, Word-level human interpretable scoring mechanism for novel text detection using tsetlin machines, Applied Intelligence (2022).

[8] B. Bhattarai, O.-C. Granmo, L. Jiao, Explainable tsetlin machine framework for fake news detection with credibility score assessment, in: LREC, 2022.

[9] R. Saha, O.-C. Granmo, M. Goodwin, Mining interpretable rules for sentiment and semantic relation analysis using tsetlin machines, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, 2020, pp. 67–78. doi:10.1007/978-3-030-63799-6_5.

[10] B. Bhattarai, O.-C. Granmo, L. Jiao, An interpretable knowledge representation framework for natural language processing with cross-domain application, in: Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part I, 2023, pp. 167–181.

[11] C. D. Blakely, O.-C. Granmo, Closed-Form Expressions for Global and Local Interpretation of Tsetlin Machines with Applications to Explaining High-Dimensional Data, arXiv preprint arXiv:2007.13885 (2020).

[12] A. Wheeldon, R. Shafik, A. Yakovlev, J. Edwards, I. Haddadi, O.-C. Granmo, Tsetlin Machine: A New Paradigm for Pervasive AI, in: SCONA Workshop at Design, Automation and Test in Europe (DATE 2020), 2020.

[13] J. Lei, A. Wheeldon, R. Shafik, A. Yakovlev, O.-C. Granmo, From Arithmetic to Logic Based AI: A Comparative Analysis of Neural Networks and Tsetlin Machine, in: 27th IEEE International Conference on Electronics Circuits and Systems (ICECS2020), IEEE, 2020.

[14] K. D. Abeyrathna, O.-C. Granmo, M. Goodwin, Extending the Tsetlin Machine With Integer-Weighted Clauses for Increased Interpretability, IEEE Access 9 (2021).

[15] J. Lei, T. Rahman, R. Shafik, A. Wheeldon, A. Yakovlev, O.-C. Granmo, F. Kawsar, A. Mathur, Low-Power Audio Keyword Spotting using Tsetlin Machines, arXiv preprint arXiv:2101.11336 (2021).

[16] K. D. Abeyrathna, A. A. O. Abouzeid, B. Bhattarai, C. Giri, S. Glimsdal, O.-C. Granmo, L. Jiao, R. Saha, J. Sharma, S. A. Tunheim, X. Zhang, Building concise logical patterns by constraining tsetlin machine clause size, in: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 2023.

[17] R. Shafik, A. Wheeldon, A. Yakovlev, Explainability and Dependability Analysis of Learning Automata based AI Hardware, in: IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), IEEE, 2020.

[18] R. Saha, O.-C. Granmo, V. I. Zadorozhny, M. Goodwin, A relational tsetlin machine with applications to natural language understanding, Journal of Intelligent Information Systems (2022) 1–28.

[19] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, G. T. Berge, The convolutional tsetlin machine, arXiv preprint arXiv:1905.09688 (2019).

[20] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, M. Goodwin, The regression tsetlin machine: a novel approach to interpretable nonlinear regression, Philosophical Transactions of the Royal Society A 378 (2019) 20190165. doi:10.1098/rsta.2019.0165.

[21] J. Lloyd, Foundations of Logic Programming, Springer-Verlag, New York, 1984.

[22] R. Kowalski, Logic programming, in: Computational Logic, volume 9, 2014, pp. 523–569. doi:10.1016/B978-0-444-51624-4.50012-5.

[23] B. Park, K. Greene, M. Colaresi, How to teach machines to read human rights reports and identify judgments at scale, Journal of Human Rights 19 (2020) 99–116.