An Interpretable Knowledge Representation Framework for Natural Language Processing with Cross-Domain Application

Bimal Bhattarai, Ole-Christoffer Granmo, and Lei Jiao

Centre for AI Research, University of Agder, Grimstad, Norway
{bimal.bhattarai, ole.granmo, lei.jiao}@uia.no

Abstract. Data representation plays a crucial role in natural language processing (NLP), forming the foundation for most NLP tasks. Indeed, NLP performance highly depends upon the effectiveness of the preprocessing pipeline that builds the data representation. Many representation learning frameworks, such as Word2Vec, encode input data based on local contextual information that interconnects words. Such approaches can be computationally intensive, and their encoding is hard to explain. We here propose an interpretable representation learning framework utilizing Tsetlin Machine (TM). The TM is an interpretable logic-based algorithm that has exhibited competitive performance in numerous NLP tasks. We employ the TM clauses to build a sparse propositional (boolean) representation of natural language text. Each clause is a class-specific propositional rule that links words semantically and contextually. Through visualization, we illustrate how the resulting data representation provides semantically more distinct features, better separating the underlying classes. As a result, the following classification task becomes less demanding, benefiting simple machine learning classifiers such as Support Vector Machine (SVM). We evaluate our approach using six NLP classification tasks and twelve domain adaptation tasks. Our main finding is that the accuracy of our proposed technique significantly outperforms the vanilla TM, approaching the competitive accuracy of deep neural network (DNN) baselines. Furthermore, we present a case study showing how the representations derived from our framework are interpretable. [1]

Keywords: natural language processing (NLP) · Tsetlin Machine (TM)· propositional logic · knowledge representation · domain adaptation · interpretable representation.

1 Introduction

The performance of machine- and deep learning in NLP heavily relies on the representation of natural language text. Therefore, succeeding with such models

[1] We use an asynchronous and parallel version of Tsetlin Machine (available at https://github.com/cair/PyTsetlinMachineCUDA)

requires efficient preprocessing pipelines that produce effective data representations. Firstly, data representation influences the accuracy of the classifier by determining how much helpful information it can extract from raw data. Secondly, dense and high-dimensional representation models can be more costly to compute. Indeed, recent advances in deep neural networks (DNNs) have brought forward both the accuracy benefits and the complexity of NLP data representation.

Since natural language data is unstructured, encompassing multiple granularities, tasks, and domains, achieving sufficient natural language understanding is still challenging. Simultaneously, state-of-the-art language models like BERT and GPT-3 struggle with high computational complexity and lack of explainability [2,35]. One might argue that attention is an explanation. However, attention merely highlights which part of the input the model used to produce its output. It does not break down the focus area into semantically meaningful units and cannot explain the ensuing reasoning process leading to an output decision [36]. Further, computation-wise, the complexity of attention is quadratic.

DNN NLP models usually represent words in vector space. Word2Vec is one early and widely used vector-based representation approach introduced by Mikolov et al. in 2013 [29]. In Word2Vec, a single-layer neural network learns the context of words and relates the words based on the inner product of context vectors. Similarly, GloVe is a popular unsupervised model incorporating corpus-wide word co-occurrence statistics [31]. The cornerstone of the latter two approaches is the distributional hypothesis, which states that words with similar contexts have similar meanings. While boosting generalization ability by co-locating similar words in vector space, the dense vectors are expensive to compute and difficult to interpret.

The Tsetlin Machine (TM) is a logic-based pattern recognition approach that blends summation-based (cf. logistic regression) and rule-based approaches (cf. decision trees). Recent studies on TMs report promising performance in NLP, including sentiment analysis [44], novelty detection [6,9], fake news detection [8], semantic relation analysis [34], and robustness toward counterfactual data [46].

The TM leverages propositional logic for interpretable modeling and bitwise operation for efficiency. Yet, recent TM research reports increasingly competitive NLP accuracy compared to deep learning at reduced complexity and increased efficiency. Simple AND rules, referred to as clauses, give these properties, employing set-of-words (SOW) as features. The clauses are self-contained, hence parallelizable [1]. Simultaneously, they can capture discriminative patterns that are interpretable [10].

Contributions: In this paper, we propose a representation learning framework for NLP classification utilizing TM. We use the TM clauses for supervised pretraining, building an abstract logical representation of the training data. We then show that the logical representation may be effective already after three epochs of training for six NLP classification tasks. We also evaluate our logic-based approach on twelve domain adaptation tasks from the Amazon dataset.

Furthermore, as the learning of TM is human-interpretable, we provide a case study to explore the explainability of our representation.

2 Related Work

Conventional representation learning mostly focuses on feature engineering for data representation. For example, [23] introduced distributed representation for symbolic data, further developed in the context of statistical language modelling [4] and in neural net language models [3]. The neural language models are based on learning a distributed representation for each word, termed as a *word embedding*. One of the most common techniques in NLP is the bag of words (BOW) representation [49], extended to n-grams, topic modelling [42], and fuzzy BOW [50]. Other techniques include representing text as graphs [28]. However, because these models lack pre-trained knowledge, the representations produced are in general not robust, and consequently, they have degraded performance [47].

In recent years, there has been tremendous progress in NLP models employing pretrained language models [33,19,24]. Most of the state-of-the-art NLP solutions are today initialized using various pre-trained input data representations such as word2vec [29], GloVe [31], and FastText [12]. These word embeddings map words into informative low-dimensional vectors, which aid neural networks in computing and understanding languages. While the initialization of input using such embeddings has demonstrated improved performance in NLP tasks, adopting these sophisticated pretrained language models for data representation comes with a cost. First, the models are intrinsically complicated, being trained on immense amounts of data through fine-tuning of a very large number of parameters [2]. Second, as complexity rises, the interpretability of the input representation becomes more ambiguous [21]. One interpretation of such models is based on the attention mechanism, which assigns weights to input features. However, a more extensive investigation demonstrates that attention weights do not in general provide useful explanations [36].

TMs [22] are a recent rule-based machine learning approach that demonstrates competitive performance with DNN, providing human-interpretable rules using propositional reasoning [5]. Several studies have demonstrated the interpretability of TM, with competitive accuracy in comparison with other deep learning approaches. Examples of applications for TM include regression [17] , natural language understanding [44,6,8,34,9,45,7], and speech understanding [26]. Furthermore, [10] analyzed the local and global interpretability of TM clauses, showing how the TM discrimination capability can be interpreted by inspecting each clause. However, these studies generally employ TM as a classifier. In this work, we exploit the data representations created by a TM to train computationally simple machine learning classifiers such as Support Vector Machine (SVM) and Logistic Regression (LR). Our intent is to develop rich context-specific language representations by using the clauses of a TM to capture the patterns

and sub-patterns of data, utilized for later classification and domain adaptation tasks.

3 Data Representation Framework

3.1 Tsetlin Machine

A TM consists of dedicated teams of two-action Tsetlin Automata (TA) [41] that operate with boolean input and its negations. Each TA has $2N$ states (i.e., N states per action) and performs an action depending on its current state, which is either an "Include" action (in state 1 to N) or "Exclude" action (in state $N+1$ to $2N$). The TA states update based on reinforcement feedback in the form of rewards and penalties. Rewards strengthen the TA action, enhancing the confidence of the TA in the current action, whereas a penalty suppresses the action. The feedback helps the TA reach the optimal action, which is the one that maximizes the expected number of rewards. The learning of TM comes from multiple teams of TAs that build conjunctive clauses in propositional logic. During learning, each TM clause captures a specific sub-pattern, comprising negated and non-negated inputs. The output is decided by counting the number of matching sub-patterns recognized by the clauses.

The TM accepts a vector $\mathbf{x} = [x_1, \ldots, x_o]$ of o propositional features as Boolean input, to be categorized into one of Cl classes, $Y = (y_1, y_2, \ldots, y_{Cl})$, where Cl is the total number of classes. These features are then turned into a set of literals that comprises of the features themselves as well as their negated counterparts: $L = \{x_1, \ldots, x_o, \neg x_1, \ldots, \neg x_o\}$.

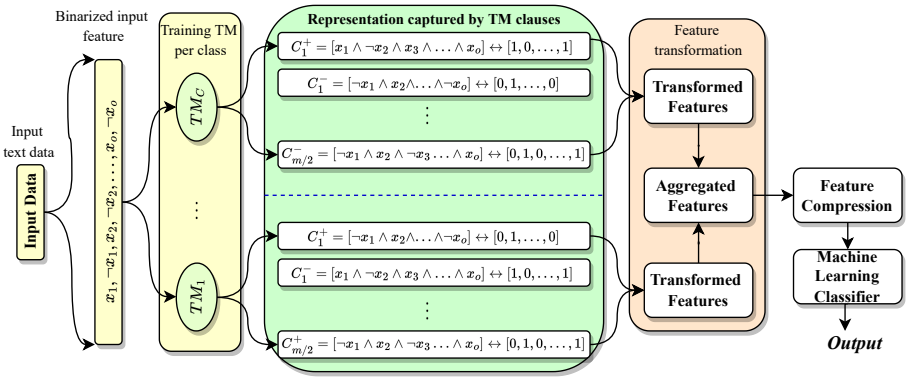


Fig. 1: Knowledge representation framework.

If there are Cl classes and m sub-patterns per class, a TM employs $Cl \times m$ conjunctive clauses to express the sub-patterns. For a given class[2], we index its clauses by $j, 1 \leq j \leq m/2$, each clause being a conjunction of literals. In general, half of the clauses are assigned positive polarity, i.e., C_j^+, and the other half are assigned negative polarity, i.e., C_j^-. A clause C_j^ψ, $\psi \in \{-, +\}$, is produced by ANDing a subset of the literals, $L_j^\psi \subseteq L$:

$$C_j^\psi(\mathbf{x}) = \bigwedge_{l_k \in L_j^\psi} l_k. \tag{1}$$

Here, $l_k, 1 \leq k \leq 2o$, is a feature or its negation. L_j^ψ is a subset of the literal set L. For example, a particular clause $C_j^+(\mathbf{x}) = x_1 \wedge \neg x_2$ consists of literals $L_j^+ = \{x_1, \neg x_2\}$ and it outputs 1 if $x_1 = 1$ and $x_2 = 0$.

The number of clauses m assigned to each class is user-configurable. The clause outputs are merged into a classification decision by summation and thresholding using the unit step function $u(v) = 1$ **if** $v \geq 0$ **else** 0:

$$\hat{y} = u\left(\sum_{j=1}^{m/2} C_j^+(\mathbf{x}) - \sum_{j=1}^{m/2} C_j^-(\mathbf{x})\right). \tag{2}$$

From Eq. (2), we can see that the classification is accomplished based on a majority vote, with the positive clauses voting for the class and the negative ones voting against it. For example, the classifier $\hat{y} = u\left(x_1\bar{x}_2 + \bar{x}_1x_2 - x_1x_2 - \bar{x}_1\bar{x}_2\right)$ captures the XOR-relation. The TM learning involves guiding the TAs to take optimal actions. Each clause receives feedback for each round of training, which is transmitted to its individual TAs. The TM utilizes Type I and Type II feedback that governs the rewards and penalties distributed to the TAs. In short, Type I feedback is designed to develop frequent patterns, eliminate false negatives, and make clauses evaluate to 1. Type II feedback, on the other hand, enhances pattern discrimination, suppresses false positives, and makes clauses evaluate to 0. Both types of feedback allow clauses to learn numerous sub-patterns from data. The details of the learning process can be found in [22].

3.2 Data Representation

The trained TM is comprised of clauses that express sub-patterns in the data. In our NLP tasks, sub-patterns typically contain contextual combinations of words that explicitly characterize a specific class. In essence, the operation of TM in NLP consists of building rules by *ANDing* groups of word literals that occur together in similar contexts. As such, the clauses (rules) are contextually rich and specific, which we here exploit to build accurate representations. By being modular and decomposable, our representation also signifies which clauses are relevant for a given input. Since our representations are based on logical rules, we will refer to them as *knowledge representations* (cf. knowledge-based systems).

[2] Without loss of generality, we consider only one of the classes, thereby simplifying the notation. Any TM class is modeled and processed in the same way.

Our overall procedure is illustrated in Fig. 1 and can be detailed as follows. In brief, consider a trained TM with m clauses. Given the input text $\mathbf{x} = [x_1, \ldots, x_o]$, we transform it into a representation consisting of logical rules:

$$TM^{\mathbf{x}}_{trans} = \zeta^{\mathbf{x}} = \|^{Cl}_{y=1}[C^{y,+}_1(\mathbf{x}), \ldots, C^{y,+}_{m/2}(\mathbf{x}), C^{y,-}_1(\mathbf{x}), \ldots, C^{y,-}_{m/2}(\mathbf{x})]. \quad (3)$$

Here, $\|^{Cl}_{y=1}$ denotes the array concatenation of the positive- and negative polarity clauses for class 1 to Cl. Each clause can be computed using Eq. (1).

Next, we perform feature compression since the transformed feature array produced in Eq. (3) can be too sparse for many machine learning algorithms. Assume that the total number of input examples is E and that each example is converted to a vector $\zeta^{\mathbf{x}} = (x_1, x_2, \ldots, x_{d_\zeta})$, $\zeta^{\mathbf{x}} \in R^{d_\zeta}$ of dimensionality $d_\zeta = Cl{\cdot}m$. The dimensionality of the input matrix then becomes $E \times d_\zeta$. We transform this input matrix further by centering the data: $A = [\Omega_1, \Omega_2, \ldots, \Omega_E]$. The center can be determined as follows:

$$\mathbf{x}_r = \frac{\sum_{e=1}^{E} \mathbf{x}_e}{E}, \quad (4)$$

$$\Omega_e = \mathbf{x}_e - \mathbf{x}_r. \quad (5)$$

The covariance matrix of A is $Cov(A, A) = E[(A - E[A])(A - E[A])^T]$ and it contains eigenvalues arranged in decreasing orders i.e., $\gamma_1 > \gamma_2 > \ldots > \gamma_E$ with corresponding eigenvectors $v_1, v_2, \ldots, v_E$. The set of original vectors can then be presented in the eigen space as follows:

$$\Omega = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_E v_E = \sum_{i=1}^{E} \alpha_i v_i. \quad (6)$$

After picking the top $\mathcal{P}$ eigenvectors v_i and corresponding eigenvalues γ_i, we have:

$$\Omega = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_{\mathcal{P}} v_{\mathcal{P}} = \sum_{i=1}^{\mathcal{P}} \alpha_i v_i, \quad (7)$$

where $\mathcal{P} << \mathcal{E}$. In the above equation, a vector of coefficients $[\alpha_1, \alpha_2, \ldots, \alpha_{\mathcal{P}}]$ represents the final representation formed in a Principal Component Analysis (PCA) space. We can observe that the number of dimensions is reduced while the most important features are retained by eigenvectors corresponding to large eigenvalues. Also, $\mathcal{P}$ eigenvalues in $\mathcal{E}$ are selected as follows:

$$\frac{\sum_{i=1}^{\mathcal{P}} \gamma_i}{\sum_{i=1}^{\mathcal{E}} \gamma_i} \geqslant \theta. \quad (8)$$

Here, θ can be 0.9 or 0.95. Now, each input example Ω_i can be expressed by a linear combination of $\mathcal{P}$ eigen vectors $\alpha_i = v_j^T \Omega_i$, where $j = 1, 2, \ldots, \mathcal{P}$, which is a compressed representation of the input given to the attached classifier, such as SVM or LR.

4 Experiments and Results

In this section, we evaluate the performance of the logical data representation created by transforming the input using the trained TMs[3].

4.1 Datasets

We conduct our experiments on six publicly accessible benchmark text classification datasets.

- TREC-6 [13] is an open-domain, fact-based question classification dataset.
- WebKB [16] comprises manually classified web pages gathered from the computer science departments of 4 universities and classified into 5 categories.
- MPQA [43] is a dataset for detecting opinion polarity.
- CR [20] is a customer review dataset with each sample labeled as positive or negative.
- SUBJ [30] is a review classification into subjective or objective.
- R8 [18] is a subset of the Reuters-21578 with 8 classes.

4.2 Implementation Details

We utilize the publicly accessible predefined train and test splits for all the datasets. The TM model is first initialized with three parameters: the number of clauses m, threshold T, and specificity s. We generate the TM representation under 3 settings: early stopping, mid stopping, and best stopping. The early stopping, mid stopping, and best stopping correspond to running our experiment with 3, 10, and 250 epochs. This enables us to observe the effect of quick TM convergence on the representation and classification performance. Following that, the TM model is trained for the different settings, and the training and testing input are transformed into respective representations using the trained model. The representations obtained at this point are uncompressed and in sparse Boolean form. Thereafter, the representation compression is done using PCA and Linear Discriminant Analysis (LDA) [4]. Finally, the compressed representation is fed into a simple machine learning classifier such as linear SVM and LR, where the only features are the transformed vectors from TM. We repeat this procedure for each setting, and the results are reported in Subsection 4.4.

[3] Classification is done using SVM from scikit-learn with default parameters.
[4] We use the default scikit-learn parameters for PCA and LDA for feature compression.

Table 1: Performance comparison (in accuracy) of vanilla TM with and without Knowledge representation in three stopping settings.

Datasets	$TM_{vanilla}$	$TM_{representation}$		
		TM_{best}	TM_{mid}	TM_{early}
TREC	91.6	95	**95.6**	92.2
MPQA	74.55	**87.3**	82.75	81.33
SUBJ	86.8	88.4	89.9	**90.1**
WebKB	91.69	**93.05**	92.19	92.47
CR	80.55	**83.06**	77.76	81.48
R8	95.93	**96.84**	96.71	96.29

4.3 Baselines

We compared the performance of our framework with deep learning and general pre-trained models. We adopted BERT [14] as a general pre-trained baseline. These models achieve state-of-the-art performance on a variety of NLP tasks. For deep learning models, we also included long short-term memory (LSTM) and convolutional neural network (CNN). We present the results of all the baseline models from the original papers. The LSTM model in our work is from [27], which represents the entire text based on the last hidden state. We use a BiLSTM model from [15,48,39]. The CNN models are taken from [25,15], which employ pretrained word embedding for initialization. The result for DiSAN, which adopts directional self-attention, is taken from [38]. The BERT model is from [14].

Table 2: Performance comparison of our model with baseline algorithms. We reproduce the results with the same hyperparameter configurations for all baselines for a fair comparison and report average accuracy across 10 different random seeds.

Models	TREC	MPQA	SUBJ	WebKB	CR	R8
LSTM	87.19	89.43	85.66	85.32	80.06	96.09
BiLSTM	91.0	89.5	92.3	-	-	96.31
CNN-non-static	93.6	89.05	93.4	-	84.3	95.71
CNN-static	92.0	89.06	93.0	-	84.7	94.02
CNN-multichannel	92.2	89.4	93.2	-	85.0	-
DiSAN	94.2	90.1	94.2	-	84.8	-
BERT	**97.6**	**90.66**	**97.0**	79.0	**86.58**	96.02
TM	91.6	74.55	86.8	91.69	80.55	95.93
TM_{rep}	95.6	87.3	90.1	**93.05**	83.06	**96.84**

4.4 Results and analysis

The performance of our representation is compared to vanilla TM under 3 different settings (as explained in Subsection 4.2) shown in Table 1. The $TM_{vanilla}$

is the text classification using legacy TM. And TM_{rep} makes use of the representation generated by TM under various settings. On all datasets, we observe that the classification accuracy using the representation outperforms vanilla TM. For MPQA, we can see a massive improvement of around 13% followed by approximately 4% for TREC. With only 3 epochs, we can observe that the TM representation performs well, with the highest accuracy in the SUBJ dataset. This also demonstrates how the quick convergence of TM enables the generation of richer representation within a small number of epochs, hence benefiting representation production time.

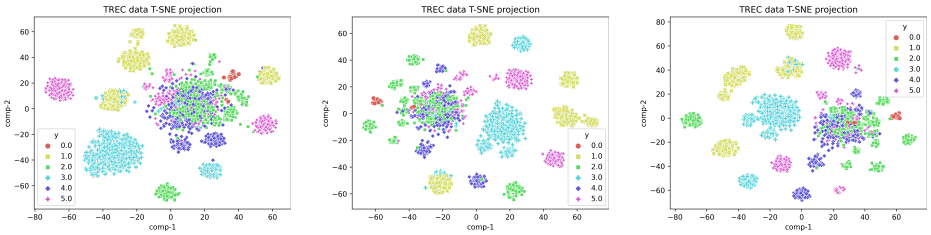
Table 3: Computation for TM vs BERT in TREC dataset.

Parameter	$BERT$	TM
10 Epochs	297s	96s
Memory in (MB)	4637	1131

The performance result of our representation framework is compared with the other baselines (from Subsection 4.3) in Table 2. We observe that our framework performs competitively with other baselines. The model beats all other baselines in WebKB and R8, with an accuracy of 93.05% 96.84% respectively. WebKB largely entails classifying personal attributes captured by sparse data, such as categorizing individual students and professors from academia. The sparseness of the data may explain the poor performance of BERT according to a study that reports that BERT completely ignores the minority class at test time for low-resource tasks such as few-shot learning and rare entity recognition [40]. For TREC, MPQA, and CR, BERT outperforms all other models. Our model, on the other hand, achieves 95.6 percent on TREC, placing it as the second-best model in terms of performance. LSTM performs the worst of all models, whereas BiLSTM performs competitively. Surprisingly, a basic CNN model with static vectors gives competitive results against the more sophisticated attention-based DiSAN model. We see that the performance of vanilla TM falls short when compared with models initialized with pre-trained word embeddings. Overall, the TM representation performs competitively compared with computationally intensive models such as BERT, which is trained on a big text corpus. Training time and memory consumption of TM and BERT are shown in Table 3. To calculate training time, we run both TM and BERT for 10 epochs. Note that the TM is trained entirely from scratch, whereas the BERT is simply fine-tuned. We observe that the TM outperforms BERT by 3× in terms of training time and memory utilization.

4.5 Visualization

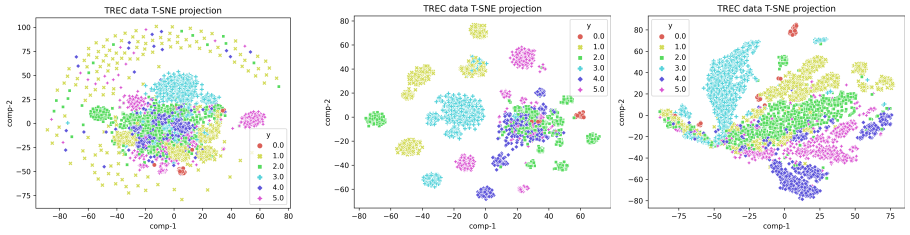
To investigate how our representation enhances the performance on NLP tasks, we plot the learned representation using t-SNE. We visualize the input with



(a) 3 *epochs* (b) 10 *epochs* (c) *best epochs*

Fig. 2: Visualization of t-SNE projection of representation produced in 3 training settings on TREC dataset.

and without the TM representation in Fig. 3. Additionally, the figure contains the corresponding BERT hidden layer representation. [5]. We observe that both representations provide richer and more precise information because the clusters get more separated and clear-cut. Further, notice how the TM clauses compress the data, significantly reducing the number of distinct data points (Fig. 3b). Each data point relates important features, formulated in propositional logic. Additionally, we demonstrate in Fig. 2 how the number of epochs influences the representation. We observe that as the number of epochs increases, the cluster becomes increasingly compact and distinct.



(a) *Raw Data* (b) *TM* (c) *BERT*

Fig. 3: Visualization of t-SNE projection for raw data, TM, and BERT representation on TREC dataset.

4.6 Concluding remarks

From the above empirical results and visual analysis, our conclusion is that the TM representation considerably enhances the input feature space, resulting in

[5] For BERT representation, the pretrained "BERT Base Uncased" model is fine-tuned with 3 epochs, and hidden states from the 11^{th} layer are visualized.

enhanced performance. As explored further below in the case study on interpretability, the advantage of using TM can at least partially be explained by its ability to capture both semantic and structural word representations from the input. Additionally, unlike DNNs, our model provides a reasonable trade-off between performance and explainability. That is, the TM representation is computationally simple and explainable through the logic-based propositional rules composed by the clauses.

5 A Case Study: Interpretability

In this section, we demonstrate a case study showing how the representation produced by our framework is interpretable. Let us assume the following input sentence from the TREC dataset: "what is the highest waterfall in the united states?" with the label "Location" and "what is the date of boxing day?" with the label "Entity". After tokenization, the vocabulary will consist of the following tokens: ["what", "highest", "waterfall", "united", "states", "date", "boxing", "day"]. During training, the clauses capture the distinctive pattern to designate each label. Fig. 4 contains some sample clauses that support "Location".

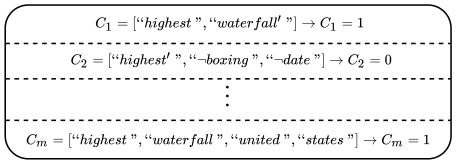


Fig. 4: Literals captured by clauses.

Referring to Fig. 1, the given input can be represented using TM clauses from a trained model. The representation can be written in an array: $[C_1, C_2, \ldots, C_m] \rightarrow [1, 0, \ldots, 1]$. For a given input, the representation consists of an array of clauses that are activated. And the clauses that are activated encapsulate the propositional rules necessary to make the correct classification decision. As a result, the representation is dense with information and can be completely interpretable. For example, for the above input, C_1 and C_m in Fig. 4 are activated in the representation. The vocabulary encompassed by these clauses are ["highest", "waterfall", "united", "states"]. That is, these clauses encapsulate the propositional rules associated with the label "Location".

6 Application: Domain Adaptation

We here demonstrate that the input representations produced from our framework can be used in domain adaptation tasks. These results thus reinforce our

Table 4: Domain adaptation performance (accuracy %) on Amazon review dataset.

	S-only	MMD	DANN	CORAL	WDGRL	ACAN	BERT	TM_{rep}
B → D	81.09	82.57	82.07	82.74	83.05	83.45	**86.75**	84.94
B → E	75.23	80.95	78.98	82.93	83.28	81.20	82.80	**86.21**
B → K	77.78	83.55	82.76	84.81	85.45	83.05	86.20	**87.57**
D → B	76.46	79.93	79.35	80.81	80.72	82.35	81.55	**85.06**
D → E	76.24	82.59	81.64	83.49	83.58	82.80	80.60	**86.81**
D → K	79.68	84.15	83.41	85.35	86.24	78.60	83.00	**87.75**
E → B	73.37	75.72	75.95	76.91	77.22	79.75	81.85	**84.83**
E → D	73.79	77.69	77.58	78.08	78.28	81.75	**83.85**	83.43
E → K	86.64	87.37	86.63	87.87	88.16	83.35	**90.80**	87.88
K → B	72.12	75.83	75.81	76.95	77.16	80.80	82.10	**82.30**
K → D	75.79	78.05	78.53	79.11	79.89	82.10	82.05	**83.07**
K → E	85.92	86.27	86.11	86.83	86.29	86.60	**88.35**	88.31
AVG	77.84	81.22	80.74	82.16	82.43	82.15	84.13	**85.68**

previous conclusion that the representations are rich, and also applicable as contexts in cross-domain applications. We employ Amazon reviews datasets [11], which comprises 4 domains: Books (B), DVD (D), Electronic (E), and Kitchen & Housewares (K), with 12 adaptation scenarios. Each domain has around 2000 labeled and approximately 4000 unlabeled reviews. We follow the transductive setting in [32] to train in the source domain and test in the target domains. For a fair comparison, the results for the baseline algorithms are obtained directly from [37,51]. The results are summarized in Table 4. As shown, the new approach can outperform baseline algorithms in 9 out of 12 tasks. And on average, our model beats all the other algorithms.

7 Conclusion

In this paper, we propose a data representation framework that enhances the performance of Tsetlin Machines (TMs). Our approach is capable of producing more sophisticated data representation through the utilization of semantic and contextual patterns captured by clauses in TMs. We conduct extensive experiments on NLP classification and domain adaptation using publicly available datasets. In NLP classification, our experimental findings suggest that our method is competitively equal to complicated and non-transparent DNNs, including BERT. In domain adaptation, we outperform all other baselines, illustrating that the representation produced from our framework can be employed in cross-domain applications. Additionally, using a t-SNE plot, we visualize how the representation can enhance input features by utilizing distinctive decision boundaries for each class. Finally, we present a case study demonstrating the interpretability of TM-generated representation.

References

1. Abeyrathna, K.D., Bhattarai, B., Goodwin, M., Gorji, S., Granmo, O.C., Jiao, L., Saha, R., Yadav, R.K.: Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling. In: The Thirty-eighth International Conference on Machine Learning (ICML 2021). ICML (2021)
2. Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. pp. 610–623 (2021)
3. Bengio, Y.: Neural net language models. Scholarpedia **3**(1), 3881 (2008)
4. Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. Advances in Neural Information Processing Systems **13** (2000)
5. Berge, G.T., Granmo, O.C., Tveit, T.O., Goodwin, M., Jiao, L., Matheussen, B.V.: Using the tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications. IEEE Access **7**, 115134–115146 (2019)
6. Bhattarai, B., Granmo, O.C., Jiao, L.: Measuring the novelty of natural language text using the conjunctive clauses of a Tsetlin machine text classifier. In: Proceedings of ICAART (2021)
7. Bhattarai, B., Granmo, O.C., Jiao, L.: Convtexttm: An explainable convolutional tsetlin machine framework for text classification. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference (2022)
8. Bhattarai, B., Granmo, O.C., Jiao, L.: Explainable tsetlin machine framework for fake news detection with credibility score assessment. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference (2022)
9. Bhattarai, B., Granmo, O.C., Jiao, L.: Word-level human interpretable scoring mechanism for novel text detection using tsetlin machines. Applied Intelligence (2022)
10. Blakely, C.D., Granmo, O.C.: Closed-Form Expressions for Global and Local Interpretation of Tsetlin Machines. In: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2021). Springer (2021)
11. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th annual meeting of the association of computational linguistics. pp. 440–447 (2007)
12. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the association for computational linguistics **5**, 135–146 (2017)
13. Chang, E., Seide, F., Meng, H.M., Chen, Z., Shi, Y., Li, Y.C.: A system for spoken query information retrieval on mobile devices. IEEE Transactions on Speech and Audio processing **10**(8), 531–541 (2002)
14. Chen, Q., Zhang, R., Zheng, Y., Mao, Y.: Dual contrastive learning: Text classification via label-aware data augmentation. arXiv preprint arXiv:2201.08702 (2022)
15. Chen, T., Xu, R., He, Y., Wang, X.: Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. Expert Systems with Applications **72**, 221–230 (2017)
16. Craven, M.W., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T.M., Nigam, K., Slattery, S.: Learning to extract symbolic knowledge from the world wide web. In: AAAI/IAAI (1998)

17. Darshana Abeyrathna, K., Granmo, O.C., Zhang, X., Jiao, L., Goodwin, M.: The regression tsetlin machine: a novel approach to interpretable nonlinear regression. Philosophical Transactions of the Royal Society A **378**(2164), 20190165 (2020)
18. Debole, F., Sebastiani, F.: An analysis of the relative hardness of reuters-21578 subsets. Journal of the American Society for Information Science and technology **56**(6), 584–596 (2005)
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
20. Ding, X., Liu, B., Yu, P.S.: A holistic lexicon-based approach to opinion mining. In: Proceedings of the 2008 international conference on web search and data mining. pp. 231–240 (2008)
21. Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: An overview of interpretability of machine learning. In: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA). pp. 80–89. IEEE (2018)
22. Granmo, O.C.: The tsetlin machine–a game theoretic bandit driven approach to optimal pattern recognition with propositional logic. arXiv preprint arXiv:1804.01508 (2018)
23. Hinton, G., McClelland, J., Rumelhart, D.: Distributed representations. In: The Philosophy of Artificial Intelligence, pp. 248–280. Oxford University Press (1990)
24. Ilic, S., Marrese-Taylor, E., Balazs, J.A., Matsuo, Y.: Deep contextualized word representations for detecting sarcasm and irony. In: WASSA@ EMNLP (2018)
25. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751 (2014)
26. Lei, J., Rahman, T., Shafik, R., Wheeldon, A., Yakovlev, A., Granmo, O.C., Kawsar, F., Mathur, A.: Low-power audio keyword spotting using tsetlin machines. Journal of Low Power Electronics and Applications **11**(2), 18 (2021)
27. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 2873–2879 (2016)
28. Luo, Y., Uzuner, Ö., Szolovits, P.: Bridging semantics and syntax with graph algorithms—state-of-the-art of extracting biomedical relations. Briefings in bioinformatics **18**(1), 160–178 (2017)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems **26** (2013)
30. Pang, B., Lee, L.: A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04). pp. 271–278 (2004)
31. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
32. Qu, X., Zou, Z., Cheng, Y., Yang, Y., Zhou, P.: Adversarial category alignment network for cross-domain sentiment classification. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 2496–2508 (2019)
33. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)

34. Saha, R., Granmo, O.C., Goodwin, M.: Mining interpretable rules for sentiment and semantic relation analysis using tsetlin machines. In: Proceedings of International Conference on Innovative Techniques and Applications of Artificial Intelligence (2020)
35. Schwartz, R., Dodge, J., Smith, N., Etzioni, O.: Green AI. Communications of the ACM **63**, 54 – 63 (2020)
36. Serrano, S., Smith, N.A.: Is attention interpretable? In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 2931–2951 (2019)
37. Shen, J., Qu, Y., Zhang, W., Yu, Y.: Wasserstein distance guided representation learning for domain adaptation. In: Thirty-second AAAI conference on artificial intelligence (2018)
38. Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. In: Proceedings of the AAAI conference on artificial intelligence (2018)
39. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 1556–1566 (2015)
40. Tänzer, M., Ruder, S., Rei, M.: Memorisation versus generalisation in pre-trained language models. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 7564–7578 (2022)
41. Tsetlin, M.L.: On behaviour of finite automata in random medium. Avtomat. i Telemekh **22**(10), 1345–1354 (1961)
42. Wallach, H.M.: Topic modeling: beyond bag-of-words. In: Proceedings of the 23rd international conference on Machine learning. pp. 977–984 (2006)
43. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of human language technology conference and conference on empirical methods in natural language processing. pp. 347–354 (2005)
44. Yadav, R., Jiao, L., Granmo, O.C., Goodwin, M.: Human-level interpretable learning for aspect-based sentiment analysis. In: Proceedings of AAAI (2021)
45. Yadav, R.K., Jiao, L., Granmo, O.C., Goodwin, M.: Interpretability in word sense disambiguation using tsetlin machine. In: Proceedings of ICAART. pp. 402–409 (2021)
46. Yadav, R.K., Jiao, L., Granmo, O.C., Goodwin, M.: Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation. In: The 31st International Joint Conference on Artificial Intelligence (IJCAI) (2022)
47. Yang, J., Xiao, G., Shen, Y., Jiang, W., Hu, X., Zhang, Y., Peng, J.: A survey of knowledge enhanced pre-trained models. arXiv preprint arXiv:2110.00269 (2021)
48. Zhang, T., Huang, M., Zhao, L.: Learning structured representation for text classification via reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2018)
49. Zhang, Y., Jin, R., Zhou, Z.H.: Understanding bag-of-words model: a statistical framework. International Journal of Machine Learning and Cybernetics **1**(1), 43–52 (2010)
50. Zhao, R., Mao, K.: Fuzzy bag-of-words model for document representation. IEEE transactions on fuzzy systems **26**(2), 794–804 (2017)

51. Zhou, J., Tian, J., Wang, R., Wu, Y., Xiao, W., He, L.: Sentix: A sentiment-aware pre-trained model for cross-domain sentiment analysis. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 568–579 (2020)