Tsetlin Machine Literal Streaming with Absorbing Automata

Youmna Abdelwahab
Centre for AI Research
University of Agder
Grimstad, Norway
yhabdelfat@uia.no

Ole-Christoffer Granmo
Centre for AI Research
University of Agder
Grimstad, Norway
ole.granmo@uia.no

Lei Jiao
Centre for AI Research
University of Agder
Grimstad, Norway
lei.jiao@uia.no

2024 International Symposium on the Tsetlin Machine (ISTM) | 979-8-3315-0498-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISTM62799.2024.10931700

***Abstract*—Sparse Tsetlin Machines (STMs) open up for assigning a small sample of the literals to each clause, and then permanently pruning those literals during learning through absorbing automata states. By trimming the STM in this manner, one can achieve a tenfold speed increase. However, when each clause is limited to the literal sample selected during clause creation, accuracy can drop. To reduce accuracy loss, we here introduce a scheme for streaming unallocated literals through the clauses. That is, the literals left out during clause construction are gradually integrated into the appropriate clauses throughout learning.That is, the literals left out during clause construction are gradually integrated into the appropriate clauses throughout learning. Each time an absorbing state eliminates a literal from a clause, a new unallocated one is added to that clause, placed in an *insertion* state. We study the effectiveness of the scheme at various degrees of sampling, varying the absorbing and insertion states. In particular, we investigate the effect of incorporating unallocated literal during learning. Across several benchmark datasets, we observe a boost in accuracy with sampling rates 5% and 20%. However, without literal streaming, the accuracy drops markedly for sampling rates 1% to 4% , which confirms the positive effect literal streaming has on STMs. In conclusion, literal streaming makes the Tsetlin Machine more scalable, yielding higher accuracy with fewer resources.**

***Index Terms*—Tsetlin Machine, Literal Streaming, Absorbing Tsetlin Automata, Natural Language Processing, Interpretable AI**

I. Introduction

The Tsetlin Machine (TM) is a pattern recognition approach that employs groups of Tsetlin Automata (TA) to produce logical rules by formulating conjunctive clauses. Concisely, the TM combines summation-based and rule-based techniques (logistic regression and decision trees) respectively, allowing for the learning of nonlinear patterns similar to neural network learning. The TM is based on TA to generate logical rules through conjunctive clauses. It combines the rules for the final decision through majority voting.

TMs have been used for several paradigms of machine learning such as classification [1], convolution [2], regression [3], auto-encoding [4], and reinforcement learning [5], [6]. Its parallel architecture further facilitates constant-time scaling through GPU-based acceleration [7]. Recent research investigates various application of TMs, such as sentiment analysis [8], novelty detection [9], [10], fake news detection [11], counterfactual robustness [12], and representation learning [13].

While the vanilla TM performs competitively across various applications and paradigms, a recent study proposes a Sparse Tsetlin Machine (STM) with absorbing TA states [14], illustrated in Fig. 1. That construct TM learning absorbing rather than ergodic. That is, the absorbing TA state locks the transitions of the TA once it goes to this state. If a TA excluded its literal in the absorbing state, the literal is excluded permanently. Similarly, if the TA included the literal, it permanently incorporates the literal into the clause. Because the TA is the learning entity in a TM, the absorbing functionality immediately reduces training time and energy consumption. Additionally, STMs open up for assigning a small sample of the literals to each clause, with a tenfold speed increase. However, when each clause is limited to the literal sample selected during clause creation, accuracy can drop. To reduce accuracy loss, this paper introduces a scheme for streaming unallocated literals through the clauses. That is, the literals left out at clause creation are gradually introduced to the corresponding clauses during learning. Each time an absorbing state eliminates a literal from a clause, a new unallocated one is added to that clause, placed in an *insertion* state. We then investigate the effect TM streaming has on classification effectiveness across various absorbing and insertion states. Particularly, we compare the streaming approach against pure sampling with absorbing states. We find that incorporating unassigned literals improves classification accuracy on benchmark datasets, with a significant boost observed for sampling rates 1% to 4%.This suggests that unallocated literal streaming enhances the capabilities of the STM.

Our paper is organized as follows. In Section II, we give an overview of the vanilla TM. Then, in Section III, we show how STM can accommodate for literal streaming by introducing a pool of unallocated literals per clause, with absorption-driven streaming into an insertion state. We examine the new scheme empirically in Section IV, before we conclude in Section V.

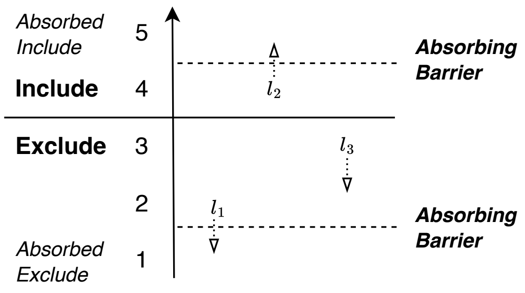

Fig. 1. Clause with absorbing states [14].

II. Tsetlin Machine Architecture

A TM employs groups of TAs with $2N$ states. Each TA has two actions, namely, include or exclude, each corresponds to N states. The states from 1 to N denote the include action, and $N+1$ to $2N$ for exclude action (Fig. 2, top). The feedback system in a TM acts as a guide for the TAs. Reward motivates TAs to move towards to the edge states of the chain. Conversely, penalties encourages the transitions towards center states.

The input vector of TM is in the form of true/false sequence (Boolean vectors), $\mathbf{X} = [x_1, \ldots, x_K]$, to be classified into one of the H classes where $Y \in \{y_i\}_{i=1}^{H}$. Each element in the sequence is denoted by x_i, $i \in \{1, \ldots K\}$. These elements of sequence[1] are transformed into a literal set that consist of both the original features and their negations: $L = \{x_1, \ldots, x_K, \neg x_1, \ldots, \neg x_K\}$. To capture different aspects of a class, the TM uses a user-defined number of clauses (N_c), $N_c = H \times \mathcal{S}$ based on the complexity of the patterns (represented by $\mathcal{S}$ sub-patterns per class). Half of these clauses are designed to favor the positive polarity class, while the other half aim to mitigate the occurrence of false positives (negative polarity). For any class h, the clauses $C^{p}_{n,h}$ is given by, $C^{p}_{n,h}(\mathbf{X}) = \bigwedge l_k, l_k \in L^{p}_{n,h}$, where n $(1 \leq n \leq N/2)$ is the clause index and $p \in \{+, -\}$ is the polarity. After training, each clause $C^{p}_{n,h}$ considers a specific subset of features of the class h, i.e., the included features in its original form or its negation, denoted by $L^{p}_{n,h}$. For example, the clause $C^{+}_{n,h}(\mathbf{X}) = \neg x_1 \wedge x_2$ considers sub pattern "$not\ x_1$ and x_2".

For any input data, each clause outputs a 0 or 1 based on whether all its chosen features match the input. These individual clause outputs are then added together. The concluding classification decision articulate on this sum. A special function (unit step function) takes this sum as input. If the sum is 0 or negative, the function outputs 0, indicating the data not belonging to the class the clauses were trained for. However, if the sum is positive the function outputs 1 signifying the data likely belongs to that class. Simply, a positive sum indicates a match to the desired class. The model does this by adding up the "votes" from each clause.Simply, the model sums the "votes" from each clause and a positive sum suggests a match to the targeted class.

Each literal l_k in each clause C^{p}_{n} has its own dedicated TA. The TA's main task is to decide whether to include that particular literal in the clause. The TA makes this decision based on a feedback system that provides rewards, inaction, or penalties in the training process of TM. TM learning leverages two main types of feedback: Type I and Type II. Type I feedback encourages the TAs to choose features that frequently appear together aiming to reduce false negatives (cases where the model misses a valid pattern). In simpler terms, it stimulates the recognition of common characteristics of the target class. On the contrary, Type II feedback is oriented towards enhancing the differentiation between the target class and alternative categories. It penalizes TAs for including features that might lead to false positives which is classifying something as belonging to a particular class when it belongs to another class. This feedback type helps the clauses become more specific and avoid incorrectly classifying data points. Through this integrated feedback system, the TAs within each clause acquire the ability to discern diverse sub-patterns present within the dataset. When effectively integrated, these sub-patterns collectively enhance the overall accuracy of the TM in classifying new data [1].

III. Sparse Tsetlin Machine With Streaming

Our STM with streaming is a variation of the STM. Like the standard STM, it uses a sparse representation of the TA states instead of dense arrays. That is, it utilizes lists to store the TAs and their states. Then, instead of simply sampling the literals available to build a clause, the literals are divided into two subsets. One fraction is made available to building the clause, referred to as *allocated* literals. The other faction is yet to be allocated, referred to as *unallocated* literals. An unallocated literal can possibly be added to the allocated subset when a TA permanently excludes (the concept of permanently excluding a literal is to be explained later) its allocated literal. In this way, the literals are streamed to the allocated subset, one by one. Accordingly, each literal, in turn, gets its chance to take part in the clause.

The sparse representation makes the approach efficient for handling high-dimensional data, while the streaming balances feature exploration against computation time. Further, by keeping some of the literals out of the loop, the approach may have a regularization effect.

To achieve the above effects, we first modify the TA itself, introducing the concept of *insertion state*. As soon as one literal is absorbed, another literal changes status from unallocated to allocated. This mechanism allows the clause construction process to focus on a few literals at a time, rather than all at once. As previous work already have covered the effect of absorbing states, we will concentrate on the effect

[1]The Boolean elements in the sequences and the features are used interchangeably in this paper.

of literal streaming. In what follows, we will summarize the concept of absorbing TA and TM, and then explain the literal streaming in detail.

A. Absorbing Tsetlin Automaton

Fig. 2 depicts the standard TA and absorbing TA, as well as TA for streaming. The top part of the figure shows a standard TA with 256 states in total similar to [14]. States 1 to 127 signify the Exclude action, while states 128 to 256 signify the Include action. The absorbing TA is shown below the standard TA. For the absorbing TA, we configure a user-defined depth for the absorbing state, which was chosen as state 117 in the figure, marked in red. Whenever the TA reaches this state, it permanently excludes the literal from the clause, discarding the TA. Finally, notice the new type of state that we introduce in this paper, marked in green in the figure. This is the insertion state, which we discuss next.

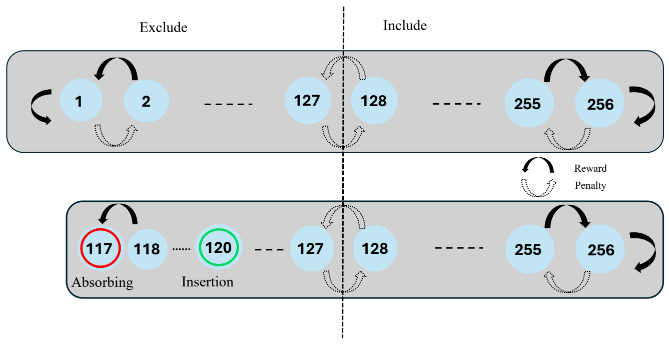

Fig. 2. Standard Tsetlin Automaton *vs* Absorbing Tsetlin Automaton. The absorbing boundary set at the 120^{th} state and Insertion boundary set at the 117^{th} state.

B. Literal Streaming

As illustrated in Fig. 3, we divide the literals for a clause into allocated and unallocated ones. The ratio between them is controlled by a hyper-parameter named sampling percentage. The unallocated literals represent feature-class combinations not yet investigated by the specific clause. Once an allocated literal is permanently excluded (i.e., absorbed in the TA), a randomly selected literal in the unallocated category can be inserted to the clause. The insertion state, which is state 120 and indicated in green in Fig. 2, is used as an initial state when an unallocated literal becomes allocated to the clause.

As further demonstrated in the figure, the left side showcase a snapshot of the status of a clause in the current form $l_2 \wedge l_3 \wedge l_7 \wedge l_9$. Its literals and TA states are also illustrated, where the number after each literal indicates the current state number in its corresponding TA. There are 5 kinds of literals in this illustration, including permanently included literals (absorbed as included), included literals (included but not absorbed), excluded literals (excluded but not absorbed), discarded literals (absorbed as excluded) and unallocated literals. The first 4 kinds have been allocated literals. After certain number of iterations, the clause evolves to $l_2 \wedge l_3 \wedge l_7 \wedge l_8 \wedge l_9$ and the status of the clause is shown on the right side. The arrows from left to right illustrate the changes of the literals. Clearly, one included literal, l_3, becomes permanently included, while another literal, l_4 becomes permanently excluded. As the permanently excluded literal, l_4, makes room in the clause, an unallocated literal, l_{12}, is allocated and inserted as an excluded literal, shown in the green part in the right figure.

C. The Merit of STM

The advantage STMs lies firstly in their dynamism. The absorption-driven streaming offers a versatile approach to pattern learning, enabling robust adaptation to diverse datasets without the need for predefined feature lists. The dynamic learning process facilitates the STM's continual update of its comprehension of the data, thereby enhancing its classification accuracy progressively. In essence, the adaptive learning mechanism empowers STM to iteratively refine their understanding of complex data patterns, leading to progressively improved classification accuracy over time. This flexibility makes STM's a promising solution for addressing complex data scenarios and advancing the field of machine learning.

The concept of absorption also refines the literal streaming process. By introducing absorbing states within the TA, the system can permanently exclude literals that consistently demonstrate irrelevance to the classification task. When an unallocated literal ventures into the Exclude side it reaches the designated absorbing state and becomes permanently removed from the pool. This mechanism prevents perpetually unsuccessful feature-class combinations from cluttering the clause creation process, ultimately enhancing the overall efficiency and focus of the STM. In essence, literal streaming with absorption offers a more dynamic, data-driven, and concise clause creation within STMs. Incorporating unallocated literals and permanently removing irrelevant ones through absorption may lead to a more sparse and concise representation compared with the traditional predefined clause setting.

IV. Empirical Results

In this section, we provide the experimental setup, variations and results for the literal sampling across absorption and insertion states. We further present and discuss our our empirical findings. We consider three aspects. We first demonstrate the impact of changing the literal sampling rate from 1% to 50% on accuracy across different absorption and insertion state variations. Additionally, we examine how accuracy is influenced by the presence or absence of literal streaming.

A. Datasets

Throughout this study, we run the experiments on 7 publicly available datasets. We have utilized the same datasets used across the previous study [14] and the IMDB dataset

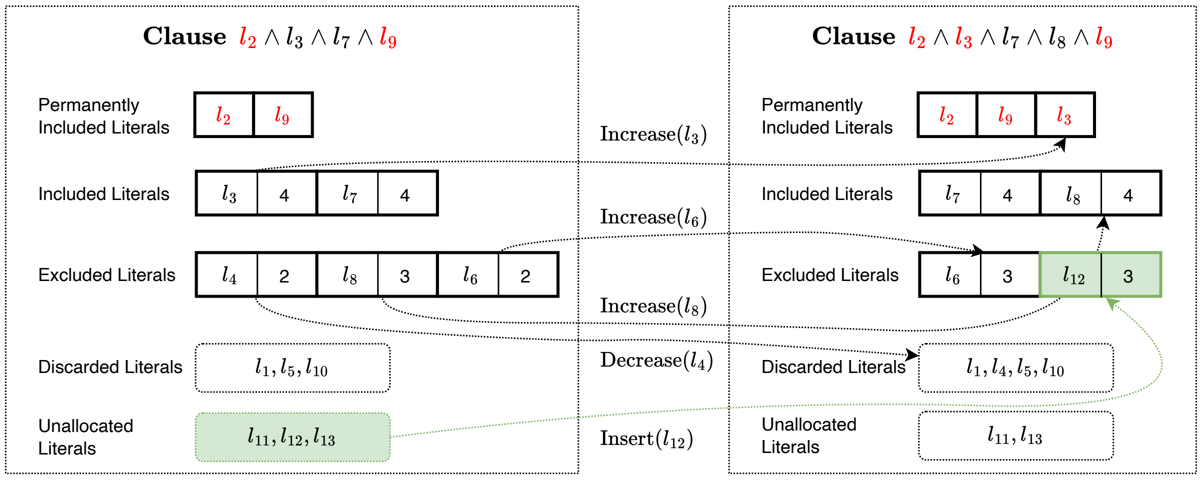

Fig. 3. Insertion of Unallocated Literals to the Excluded literals lists.

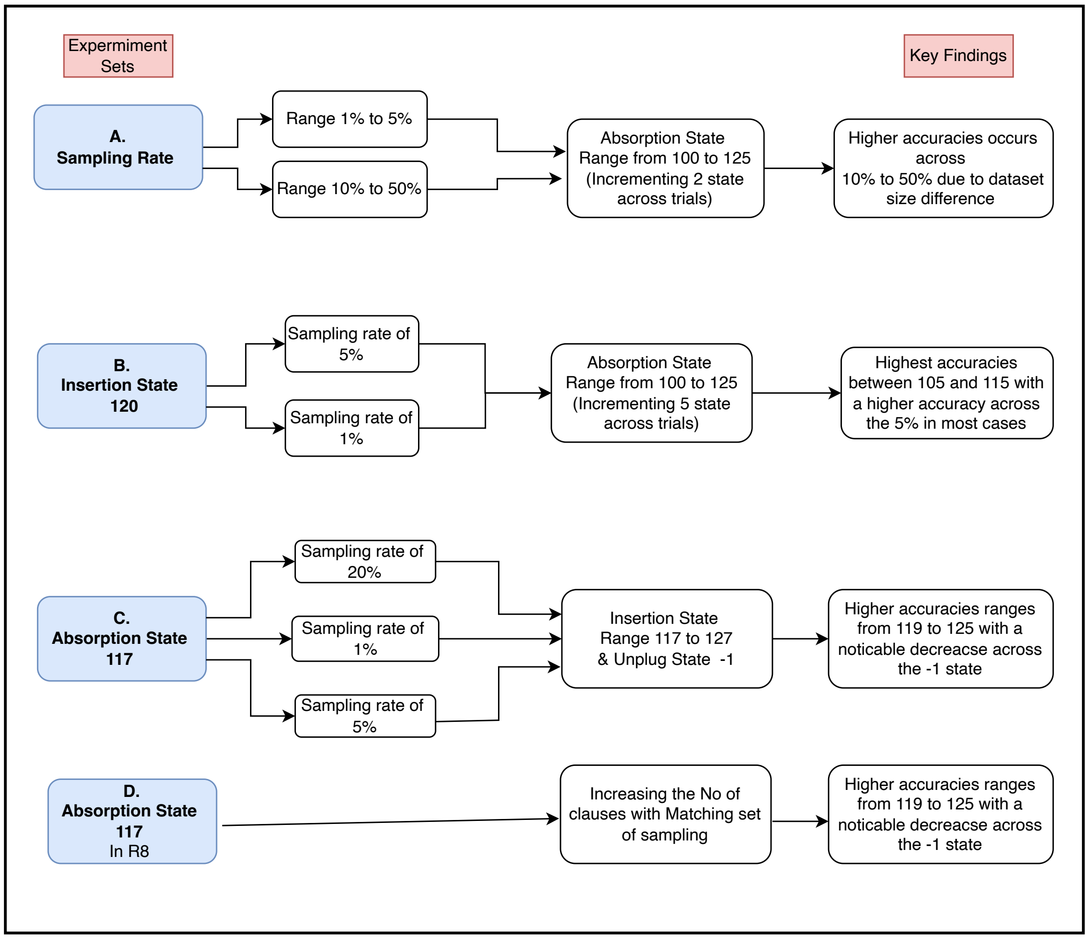
Expermiment Sets
Key Findings
A. Sampling Rate
Range 1% to 5%
Range 10% to 50%
Absorption State Range from 100 to 125 (Incrementing 2 state across trials)
Higher accuracies occurs across 10% to 50% due to dataset size difference
B. Insertion State 120
Sampling rate of 5%
Sampling rate of 1%
Absorption State Range from 100 to 125 (Incrementing 5 state across trials)
Highest accuracies between 105 and 115 with a higher accuracy across the 5% in most cases
C. Absorption State 117
Sampling rate of 20%
Sampling rate of 1%
Sampling rate of 5%
Insertion State Range 117 to 127 & Unplug State -1
Higher accuracies ranges from 119 to 125 with a noticable decreacse across the -1 state
D. Absorption State 117 In R8
Increasing the No of clauses with Matching set of sampling
Higher accuracies ranges from 119 to 125 with a noticable decreacse across the -1 state

Fig. 4. Illustrates the set of Experiments utilised across the 8 datasets mentioned in the dataset.Experimenting within the effect of literal streaming with 3 sets A. sampling rate across the 8 datasets. B. Insertion state with incrementally changing the absorption state. C.Absorption state with incrementally changing insertion state of literals and unplugging it too. D.Absorption state with incrementally changing insertion state of literals and unplugging it across R8

in addition. The predefined training and testing splits were utilized across all datasets.

- IMDB [15] is an open-domain, Movie review dataset available in the Keras Library.
- TREC-6 [16] is a fact-based question classification dataset.
- MPQA [17] is a dataset for detecting opinion polarity.
- CR [18] is a dataset that contain customer reviews with labels of either positive or negative.
- SUBJ [19] is a dataset that contain reviews classified as subjective or objective.
- R8 and R52 [20] is a subset of the Reuters-21578 with consist of 8 and 52 classes, respectively.

B. Implementation Details

The TM model is initialized with the following parameters: the number of clauses (N_c), voting margin (T), specificity (s), maximum included literals, literal sampling, and an absorbing state. The TM performance is initially fixed across the datasets utilizing N_c, T, and s with initial numbers 1000, 10000 and 1 respectively. We then vary the literal sampling values ranging from 0.01 to 0.5, while keeping the absorption and insertion state fixed on 117 and 120 respectively, to show the sampling rate effect across datasets. All of our experiments are run over 25 epochs.

We investigate four aspects of STM with streaming. Briefly illustrating the 4 sets of experiments with the findings in Fig 4.

- We evaluate literal streaming across several sampling percentages.
- We experiment with a stable sampling rate, varying the absorption state. For instance, we usesampling rate 1% and 5% across several absorbing states, with a stable insertion state at 120.
- We keep the absorption state stable at 117 and the sampling rate at 1%, 5%, and 20%, investigating the effect of varying the insertion state. We also turn off the insertion state, shown as insertion state -1 in our results.
- For the R8 dataset, we finally investigate the effect of increasing the number of clauses N and the voting margin T ten times, across sampling rates 1%, 5% and 20% and across absorbing states from 100 to 125. We here keep the insertion state table at 117

C. Results and Discussions

We first investigate how accuracy is affected by sampling percentage (in decimal form) across the datasets, illustrated in Table I. Each dataset (IMDB, R8, R52, SUBJ, CR, MPQA, and TREC) is evaluated according to multiple metrics, including accuracy and training time, at different sampling rates ranging from 0.01 to 0.5. For most of the datasets, increasing the sampling rate to 0.1 and 0.2 was required to get the maximize accuracy. A sampling rate of 0.5 occasionally gave a decrease in accuracy across, which can be attributed to over-fitting when too many literals are available for clause construction. For instance, the IMDB dataset achieves an accuracy of 91.02 at a sampling rate of 0.1, but drops to 90.84% at a sampling rate of 0.5. Also notice how the smaller sampling rates ranging from 0.01 to 0.04 gave a significant accuracy loss.

We next investigate the effect of absorption state in Tables II and III, focusing in a sampling rate of 1% and 5%. The table shows the performance of the TM with absorption states ranging from 100 to 125, with the latter being very close to the Include action boundary at state 128. For the majority of the datasets, there is a notable accuracy decrease as the absorption state increases. For instance, with the IMDB dataset, the accuracy is 90.06% at an absorption state of 120, but only 82.18% at an absorption state of 125. The R52 dataset shows a similar trend, with an accuracy of 86.88% at 120 and 84.03% at 125.However, the CR dataset appears less impacted by the absorption state, with an accuracy of 72.34% at 120 and 69.68% at 125.

With a similar pattern of performance, when upgrading the sampling to 5% the accuracy increases between an absorption state that ranges as previously mentioned. A lower absorption state in the TM leads to better performance on most datasets, which we take into consideration in the rest of the experimental trials. As well as, it was noted a decrease in training time with an increasing absorbing state.

In this manner, we further investigate the effect of sampling and the results of accuracy across different insertion state with two different absorption state of 117 and 120 which is a middle ground from previous experiments across different insertion states ranging from 117 to 127. Additionally, the -1 insertion state for the investigation into halting the streaming of unallocated literal.

The second set of experiments was conducted across different insertion states to investigate the impact of sampling with 1%, 5% and 20%, respectively across insertion state ranging from 117 to 127 and an insertion to -1 (turning literal streaming off) These results are compiled in Tables IV, V and VI with an absorption rate of 117. The highest accuracy occurred between insertion states 119 and 123 with a couple of exceptions that occurred at 125. In larger datasets, such as the IMDB, R52 and R8, the highest accuracy across the insertion state was from 119 to 123, while smaller datasets obtained the largest accuracy mostly with insertion states 119 and 120 with 120.

Importantly, observe the loss of accuracy when turning literal streaming off (insertion state -1), which is the key result of this paper. For all the datasets, the accuracies drop from 1% to 4%. For instance, in the CR dataset there was a visible decrease in accuracy when turning off literal streaming in terms of accuracies and also a decrease in the training time. This result also pinpoints that the size of the dataset has a slight effect on the accuracy. Throughout the three tables, there

TABLE I
PERFORMANCE OF TM WITH DIFFERENT SAMPLING WITH ABSORPTION STATE 117 AND INSERTION STATE 120. THE PERFORMANCE RESULT IS PRESENTED IN PERCENTAGE AND TRAINING TIME IN SECONDS FROM THE LAST EPOCH.

Datasets	**Sampling**									
	0.01	**0.02**	**0.03**	**0.04**	**0.05**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
IMDB	90.73(69.2)	90.23(40.3)	90.24(38.3)	90.41(27.8)	90.75(26.0)	**91.02(23.3)**	90.99(25.5)	90.90(18)	90.86(17.4)	90.84(17.2)
R8	92.46(2.2)	94.14(2.4)	93.80(2.4)	93.97(2.4)	94.47(11.12)	94.30(2.5)	94.47(2.5)	**94.81(10.56)**	94.47(2.6)	94.47(2.6)
R52	87.69(4.2)	89.1(4.5)	89.99(4.5)	90.69(4.9)	91.08(4.5)	92.64(5.8)	92.33(5.5)	93.46(5.3)	93.46(5.4)	**93.57(5.2)**
SUBJ	85.15(2.8)	88.10(3.9)	88.95(4.6)	88.60(6.1)	89.0(5.5)	**89.50(6.8)**	89.0(7.6)	89.15(8.2)	88.70(7.2)	88.95(8.6)
CR	72.07(3.4)	72.07(7.3)	68.88(7.3)	69.41(7.9)	71.28(8.0)	73.40(8.6)	**75.27(7.5)**	73.94(8.1)	69.41(8.4)	68.88(8.4)
MPQA	69.37(1.7)	69.75(2.2)	69.43(3.6)	69.18(3.0)	69.79(3.5)	70.78(5.5)	83.69(5.5)	84.73(5.2)	**85.34(5.6)**	84.92(5.8)
TREC	82.80(6.6)	79.60(9.2)	84.0(10.5)	84.60(6.4)	86.20(9.4)	87.40(5.5)	**89.20(7.2)**	88.80(5.3)	87.60(7.9)	88.40(9.2)

TABLE II
PERFORMANCE OF TM OF 120 AS AN INSERTION STATE WITH 1% PERCENT SAMPLING

Datasets	**Absorption State**					
	100	**105**	**110**	**115**	**120**	**125**
IMDB	90.12(106.9)	90.04(105.4)	90.06(111.4)	**90.39(114.9)**	89.64(126.9)	82.18(14.3)
R8	93.47(2.6)	92.80(2.8)	92.96(2.7)	**93.97(2.6)**	93.30(2.4)	92.80(2.0)
R52	85.32(6.6)	85.32(6.8)	86.88(6.6)	86.45(6.3)	**89.33(6.3)**	84.03(6.3)
SUBJ	**88.55(13.4)**	86.35(14.1)	83.10(13.3)	84.70(13.3)	86.20(13.0)	87.50(12.9)
CR	72.61(3.3)	70.74(3.3)	71.54(2.2)	72.34(3.3)	**74.20(3.3)**	69.68(3.3)
MPQA	69.37(3.1)	69.37(4.0)	71.23(7.6)	**73.04(11.9)**	71.49(11.7)	69.37(12.2)
TREC	76.40(5.6)	**80.20(6.1)**	76.20(5.0)	79.80(5.9)	76.0(5.8)	75.20(4.9)

TABLE III
PERFORMANCE OF TM OF 120 AS AN INSERTION STATE WITH 5% PERCENT SAMPLING

Datasets	**Absorption State**					
	100	**105**	**110**	**115**	**120**	**125**
IMDB	91.06(44.0)	**91.11(43.5)**	90.86(41.8)	90.29(39.2)	90.69(41.2)	81.25(10.1)
R8	93.47(4.0)	93.97(2.7)	94.30(3.0)	**94.47(2.4)**	93.97(2.4)	92.46(1.2)
R52	91.28(7.6)	91.28(7.9)	90.81(7.7)	**91.82(7.4)**	91.20(6.3)	86.95(4.5)
SUBJ	88.80(30.4)	**89.20(33.1)**	89.05(29.1)	89.0(31.1)	88.85(22.1)	88.55(19.9)
CR	71.01(1.5)	70.74(1.4)	71.24(1.4)	**73.67(1.5)**	72.61(1.0)	69.95(0.8)
MPQA	68.90(37.3)	69.46(33.2)	68.99(33.5)	**69.60(32.2)**	69.23(31.8)	69.13(34.1)
TREC	85.40(2.4)	**86.40(2.4)**	84.20(2.4)	84.0(2.3)	83.0(2.2)	79.20(2.1)

is an accuracy decrease comparing the 117 insertion state to the -1 state.

We then experiment with an increase of the N and T to 10000 clauses and, 100000 respectively, which led to a slight increase of 1% to 2% percent of higher accuracy throughout the several insertion states mentioned in the table. The most visible would be the difference between the 117 insertion state and the -1 insertion state, with a 2% increase. Meanwhile throughout the 3 set of experiments, the training time tends to fluctuate within the insertion states.

However, upon comparing the first and last insertion states, a reduction in training time was observed. For further proof we demonstrate a comparison between previous studies.In [14] and [21]they reported the average of the epochs across experiments. While in this study we report the average and (maximum) values across VII with comparison of previous results throughout a subset of the datasets and reported the maximum values across the experiments throughout the paper.

V. CONCLUSION

This study investigated the impact of sampling unallocated literal streaming on the performance of STM. Our findings

TABLE IV
PERFORMANCE OF TM OF 117 AS AN ABSORPTION STATE WITH 20% PERCENT SAMPLING

Datasets	**Insertion State**						
	117	**119**	**121**	**123**	**125**	**127**	**-1**
IMDB	90.93(23.1)	91.0(22.2)	91.11(25.1)	**91.12(28.1)**	91.09(32.6)	91.07(37.6)	90.0(23.6)
R8	94.64(2.2)	94.14(2.6)	94.14(2.6)	93.63(2.6)	**94.81(2.7)**	94.64(2.3)	93.47(2.2)
R52	92.95(8.4)	**93.85(8.7)**	92.72(8.4)	92.83(7.7)	92.99(7.9)	92.60(7.5)	92.20(8.0)
R8(10000 clauses)	94.47(18.1)	94.30(17.3)	93.97(16.4)	94.47(18.5)	**94.97(18.5)**	93.97(15.9)	94.4716.9)
SUBJ	89.25(15.0)	**89.85(14.5)**	88.85(12.2)	88.95(11.2)	89.25(12.8)	89.0(13.2)	88.80(15.4)
CR	70.74(1.4)	71.01(1.5)	**73.94(1.6)**	73.67(1.6)	73.67(1.4)	73.94(1.2)	69.41(1.2)
MPQA	73.80(35.7)	83.55(49.6)	**85.25(44.9)**	85.11(51.1)	84.78(48.6)	80.87(44.6)	69.46(46.2)
TREC	87.0(13.1)	**89.0(12.4)**	87.80(13.7)	86.60(12.8)	85.80(14.0)	88.20(13.0)	87.0(13.6)

TABLE V
PERFORMANCE OF TM OF 117 AS AN ABSORPTION STATE WITH 1% PERCENT OF SAMPLING

Datasets	**Insertion State**						
	117	**119**	**121**	**123**	**125**	**127**	**-1**
IMDB	89.84(14.5)	90.62(13.2)	90.58(13.9)	**88.08(12.9)**	89.29(10.3)	88.56(11.7)	88.52(13.6)
R8	93.13(2.3)	93.80(2.4)	92.96(2.4)	**93.97(2.4)**	92.46(2.3)	90.45(2.6)	92.1(2.2)
R52	**88.63(4.7)**	88.07(4.0)	87.73(5.0)	87.7(5.1)	87.93(6.0)	85.51(5.1)	80.41(4.0)
R8(10000 clauses)	**94.14(20.8)**	93.47(20.4)	93.80(23.1)	92.29(23.6)	93.13(22.4)	91.62(25.5)	92.29(16.8)
SUBJ	88.65(3.0)	86.25(3.1)	**89.0(3.1)**	88.30(2.9)	87.65(2.9)	86.70(2.9)	76.55(2.6)
CR	72.07(1.8)	71.07(1.8)	**74.47(1.9)**	69.86(1.8)	71.28(1.9)	70.40(1.8)	65.43(1.7)
MPQA	**75.35(8.1)**	73.85(8.4)	73.09(8.2)	72.43(8.1)	71.21(8.6)	72.34(8.2)	71.68(6.0)
TREC	72.36(8.1)	71.80(8.4)	74.40(8.4)	**76.60(1.4)**	70.60(1.3)	71.0(1.28)	61.20(1.13)

TABLE VI
PERFORMANCE OF TM OF 117 AS AN ABSORPTION STATE WITH 5% PERCENT SAMPLING ACROSS SEVERAL INSERTION STATES

Datasets	**Insertion State**						
	117	**119**	**121**	**123**	**125**	**127**	**-1**
IMDB	90.81(25.1)	90.84(25.4)	90.95(34.3)	90.75(6.9)	**90.92(8.1)**	90.55(8.9)	90.75(45.2)
R8	**94.64(2.4)**	94.47(2.9)	94.47(2.9)	94.47(2.9)	92.80(4.2)	93.97(2.1)	94.30(2.2)
R52	90.85(4.8)	91.24(4.9)	91.0(5.1)	**91.51(4.8)**	90.54(4.8)	88.94(6.0)	89.95(5.0)
R8(10000 clauses)	94.64(23.1)	**94.81(22.7)**	93.97(21.0)	94.47(20.9)	93.97(27.1)	94.47(20.0)	93.87(20.0)
SUBJ	88.75(5.6)	88.65(5.8)	88.65(5.7)	**89.20(5.8)**	88.55(5.8)	88.55(5.8)	84.65(4.4)
CR	73.14(1.0)	69.95(1.2)	**73.67(1.3)**	72.61(1.5)	70.74(1.4)	72.34 (1.5)	69.95(0.9)
MPQA	75.78(22.8)	74.46(28.1)	75.87(28.6)	**76.0(28.4)**	75.31(28.4)	76.0(27.7)	73.80(15.3)
TREC	82.40(10.5)	82.40(11.4)	82.20(11.9)	**83.00(32.7)**	81.20(34.8)	80.0(33.5)	77.20(23.1)

suggest that the sampling significantly affects the accuracy of the STM, illustrated across the results on different datasets with more prominent effects observed in smaller datasets compared to larger datasets such as the IMDB.

This implies that sampling is an efficient process within the learning process. Nevertheless, additional investigation is necessary to evaluate scalability, as briefly explored with the R8 dataset in this study.A comprehensive examination across larger datasets in future research endeavors. By comprehending the influence of literal streaming, we can optimize TM training strategies and improve their overall effectiveness in pattern recognition tasks.

TABLE VII
A COMPARISON OF PREVIOUS STUDIES VS CURRENT STUDY. RESULTS ACROSS

Datasets	[14]	Current	[21]
MPQA	73.85	75.9(76.0)	X
IMDB	X	90.73(91.12)	89.9
CR	75.95	72.42(75.27)	X
Subj	81.78	83.07(89.28)	X

ACKNOWLEDGMENT

This work is supported in part by the project "Logic-based Artificial Intelligence Everywhere: Tsetlin Machines in Hardware" and funded under grant number 312434 of the Research Council of Norway.

REFERENCES

[1] O.-C. Granmo, "The tsetlin machine–a game theoretic bandit driven approach to optimal pattern recognition with propositional logic," *arXiv preprint arXiv:1804.01508*, 2018.

[2] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The Convolutional Tsetlin Machine," *arXiv preprint arXiv:1905.09688*, 2019. [Online]. Available: https://arxiv.org/abs/1905.09688

[3] K. D. Abeyrathna, O.-C. Granmo, X. Zhang, L. Jiao, and M. Goodwin, "The Regression Tsetlin Machine - A Novel Approach to Interpretable Non-Linear Regression," *Philosophical Transactions of the Royal Society A*, vol. 378, 2020.

[4] B. Bhattarai, O.-C. Granmo, L. Jiao, R. Yadav, and J. Sharma, "Tsetlin machine embedding: Representing words using logical expressions," in *Findings of the Association for Computational Linguistics: EACL 2024*, Y. Graham and M. Purver, Eds. St. Julian's, Malta: Association for Computational Linguistics, Mar. 2024, pp. 1512–1522. [Online]. Available: https://aclanthology.org/2024.findings-eacl.103

[5] S. R. Gorji and O. Granmo, "Off-policy and on-policy reinforcement learning with the tsetlin machine," *Appl. Intell.*, vol. 53, no. 8, pp. 8596–8613, 2023. [Online]. Available: https://doi.org/10.1007/s10489-022-04297-3

[6] R. Seraj, J. Sharma, and O. C. Granmo, "Tsetlin Machine for Solving Contextual Bandit Problems," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[7] K. D. Abeyrathna, B. Bhattarai, M. Goodwin, S. Gorji, O.-C. Granmo, L. Jiao, R. Saha, and R. K. Yadav, "Massively Parallel and Asynchronous Tsetlin Machine Architecture Supporting Almost Constant-Time Scaling," in *International Conference on Machine Learning (ICML)*, 2021.

[8] R. Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, "Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis," in *AAAI*, 2021.

[9] B. Bhattarai, O.-C. Granmo, and L. Jiao, "Word-level Human Interpretable Scoring Mechanism for Novel Text Detection Using Tsetlin Machines," *Applied Intelligence*, 2022.

[10] B. Bimal, G. Ole-Christoffer, and J. Lei, "Measuring the Novelty of Natural Language Text Using the Conjunctive Clauses of a Tsetlin Machine Text Classifier," in *Proceedings of ICAART*, 2021.

[11] B. Bhattarai, O.-C. Granmo, and L. Jiao, "Explainable Tsetlin Machine Framework for Fake News Detection with Credibility Score Assessment," in *Proceedings of the 13th Conference on Language Resources and Evaluation*, 2022, pp. 4894–4903.

[12] R. K. Yadav, L. Jiao, O. C. Granmo, and M. Goodwin, "Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation," in *IJCAI International Joint Conference on Artificial Intelligence*, 2022.

[13] B. Bhattarai, O.-C. Granmo, and 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.

[14] B. Bhattarai, O.-C. Granmo, L. Jiao, P.-A. Andersen, S. A. Tunheim, R. Shafik, and A. Yakovlev, "Contracting tsetlin machine with absorbing automata," in *2023 International Symposium on the Tsetlin Machine (ISTM)*, 2023, pp. 1–7.

[15] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[16] E. Chang, F. Seide, H. M. Meng, Z. Chen, Y. Shi, and Y.-C. Li, "A system for spoken query information retrieval on mobile devices," *IEEE Transactions on Speech and Audio processing*, vol. 10, no. 8, pp. 531–541, 2002.

[17] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, 2005, pp. 347–354.

[18] X. Ding, B. Liu, and P. S. Yu, "A holistic lexicon-based approach to opinion mining," in *Proceedings of the 2008 international conference on web search and data mining*, 2008, pp. 231–240.

[19] B. Pang and L. Lee, "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)*, 2004, pp. 271–278.

[20] F. Debole and F. Sebastiani, "An analysis of the relative hardness of reuters-21578 subsets," *Journal of the American Society for Information Science and technology*, vol. 56, no. 6, pp. 584–596, 2005.

[21] G. T. Berge, O.-C. Granmo, T. O. Tveit, M. Goodwin, L. Jiao, and B. V. Matheussen, "Using the tsetlin machine to learn human-interpretable rules for high-accuracy text categorization with medical applications," *IEEE Access*, vol. 7, pp. 115 134–115 146, 2019.