IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 3, 2019, accepted July 11, 2019, date of publication August 14, 2019, date of current version August 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935416

Using the Tsetlin Machine to Learn Human-Interpretable Rules for High-Accuracy Text Categorization With Medical Applications

GEIR THORE BERGE[1,2,3], OLE-CHRISTOFFER GRANMO[1], TOR ODDBJØRN TVEIT[1,3,4], MORTEN GOODWIN[1], LEI JIAO[1], (Senior Member, IEEE), AND BERNT VIGGO MATHEUSSEN[1]

[1]Centre for Artificial Intelligence Research, University of Agder, 4879 Kristiansand, Norway
[2]Department of Information Systems, University of Agder, 4630 Kristiansand, Norway
[3]Department of Technology and eHealth, Sørlandet Hospital Trust, 4604 Kristiansand, Norway
[4]Department of Anesthesia and Intensive Care, Sørlandet Hospital Trust, 4604 Kristiansand, Norway

Corresponding author: Geir Thore Berge (geir.thore.berge@sshf.no)

This work was supported in part by the Norwegian Research Council under Grant 241277 and was also approved by the Norwegian Regional Committees for Medical and Health Research Ethics.

ABSTRACT Medical applications challenge today's text categorization techniques by demanding both high accuracy and ease-of-interpretation. Although deep learning has provided a leap forward in regard to accuracy, this leap comes at the sacrifice of interpretability. In this paper, we introduce a text categorization approach that leverages the recently introduced Tsetlin Machine to address this accuracy-interpretability challenge. Briefly, we represent the terms of a text as propositional variables. From these variables, we capture categories using simple propositional formulae, such as: IF "rash" AND "reaction" AND "penicillin" THEN Allergy. The Tsetlin Machine learns these formulae from labeled text, utilizing conjunctive clauses to represent the particular facets of each category. Therefore, also the absence of terms (negated features) can be used for categorization purposes. Our empirical comparisons with Naïve Bayes classifiers, decision trees, linear support vector machines (SVMs), random forest, long short-term memory (LSTM) neural networks, and other techniques, are quite conclusive. Using relatively simple propositional formulae, the accuracy of the Tsetlin Machine either outperforms or performs approximately on par with the best evaluated methods on both the 20 Newsgroups and IMDb datasets, as well as on a clinical dataset containing authentic electronic health records (EHRs). On average, the Tsetlin Machine delivers the best recall and precision scores across the datasets. The main merit of the proposed approach is thus its capacity for producing human-interpretable rules, while at the same time achieving acceptable accuracy. We believe that our novel approach can have a significant impact on a wide range of text analysis applications, providing a promising starting point for deeper natural language understanding with the Tsetlin Machine.

INDEX TERMS Classification algorithms, health informatics, machine learning, supervised learning, text categorization, Tsetlin machine.

I. INTRODUCTION

Understanding natural language text involves interpreting linguistic constructs at multiple levels of abstraction: words form phrases that interact to form sentences, which in turn are interweaved into paragraphs that carry implicit and explicit meaning [1], [2]. Because of the complexity inherent in the formation of natural language, text understanding has traditionally been a difficult area for machine learning algorithms [3]. Medical text understanding is no exception. First, medical language is typically intricate. In addition, the method used for text understanding must be interpretable for health professionals to inform medical decision-making [4], [5].

Although deep learning in the form of convolutional neural network (CNN), recurrent neural network (RNN), and Long Short-Term Memory (LSTM) recently has provided a leap ahead in text categorization accuracy [2], [6]–[8], this leap

The associate editor coordinating the review of this article and approving it for publication was Bora Onat.

has come at the expense of interpretability and computational complexity [9]. Simpler techniques such as the Naïve Bayes classifier, logistic regression, decision trees, random forest, k-nearest neighbors (kNN), and the support vector machine (SVM) are still widely used, arguably because they are simple and efficient, yet, provide reasonable accuracy, in particular when data is limited [5], [10].

A. CHALLENGES ADDRESSED IN THIS PAPER

Despite recent successes, natural language understanding continues to challenge machine learning [11], [12].

First, realistic vocabularies can be surprisingly rich, leading to high-dimensional input spaces. Depending on the feature selection strategy, text categorization has to deal with thousands to tens of thousands (or even more) of features. Even though natural language text typically contains high volumes of duplicated words (e.g., stop words), few are irrelevant [13]. Low frequency terms further carry considerable information that may be relevant for categorization [14]. As a result, a vocabulary can be so diverse that documents covering the same category may not even share a single medium frequency term [15]. As an example, the electronic health record (EHR) dataset that we target in the present paper abounds with jargon, misspellings, acronyms, abbreviations, and a mixture of Latin, English, and mother tongue [4].

Another challenge is the inherently complex composition of natural language [3], [9]. For example, determining whether a patient is allergic to drugs or not based on EHRs has proven to be very difficult without introducing handcrafted rules [4]. Rather than relying on the additivity of individual features to discriminate among categories, full sentences, paragraphs and even complete records must be considered in context. Indeed, the temporal dynamics of EHRs is a strong confounding variable. Simultaneously, EHRs may also contain ambiguous considerations and speculations which must be treated as false positives despite their similarity to true positives on the individual feature level [4], [16].

A third challenge is interpretability. While rule-based methods such as decision trees are particularly easy to understand, other techniques tend to be more complex [5], [17]. Neural networks are arguably among the most complicated and have been criticized for being ''black boxes'' [9]. That is, it is very difficult for humans to understand how they produce their results. Even trained data scientists may have problems explaining their outcomes, or such explanations may demand comprehensive and time-consuming analysis. It has been argued that as doctors or nurses are expected to justify their decisions, so should machine learning algorithms be able to explain *their* decisions [5], [17]. There has been recent progress on postprocessing techniques for neural networks to address this issue [12]. However, such approaches add to the already inherent complexity of the neural networks by introducing a new abstraction layer. How to provide both an exhaustive and compact explanation of neural network reasoning is currently an open question.

At the same time, learning propositional formulae from data has been notoriously difficult [18], with the space of candidate formulae growing exponentially with the number of propositional variables involved [19]. This exponential growth becomes particularly severe in text categorization involving extensive vocabularies. A text categorization problem may consist of thousands, or even tens of thousands, of propositional variables. The full range of possible propositional formulae is thus immense. As a result, extreme care must be taken during training to maximize the precision and recall of the categorization, while simultaneously combating overfitting.

In this paper, we attack the above challenges by introducing the first approach to text categorization that leverages the recently introduced *Tsetlin Machine* [19].

B. THE TSETLIN MACHINE

The Tsetlin Machine has garnered significant interest because it facilitates human-understandable pattern recognition by composing patterns in propositional logic. Without losing the important property of interpretability, it has outperformed state-of-the-art pattern recognition techniques in benchmarks involving pattern discrimination, image recognition, and optimal move prediction for board games [19].

The Tsetlin Machine builds on the Tsetlin Automaton, a pioneering solution to the multiarmed bandit problem developed by M. L. Tsetlin in the Soviet Union in the early 1960s [19], [20].

At the heart of the Tsetlin Machine, we find a novel game theoretic scheme that orchestrates *decentralized teams* of Tsetlin Automata. The orchestration guides the Tsetlin Automata to jointly learn arbitrarily complex propositional formulae in conjunctive normal form, capturing the various facets of the patterns faced [19]. Such formulae have turned out to be particularly suited for human interpretation, while still allowing complex nonlinear patterns to be formed [21].

Apart from being simple to interpret, the Tsetlin Machine represents both inputs, patterns, and outputs as bit sequences. Recognition of patterns, in turn, involves decentralized manipulation of those bits. This provides the Tsetlin Machine with an advantage computationally, compared to methods that rely on more complex mathematical modeling.

C. PAPER CONTRIBUTIONS AND ORGANIZATION

The contributions and organization of the paper can be summarized as follows: In Section 2, we provide the details of our Tsetlin Machine-based method to text categorization. We further demonstrate how the conjunctive clauses are formed to capture complex patterns in natural language. Next, in Section 3, we evaluate our scheme empirically on the 20 Newsgroups and the IMDB movie review (IMDb) datasets, as well as on a newly published clinical dataset, in comparison with selected state-of-the-art techniques. In addition, we conduct experiments to investigate the learning behavior and the execution speed of the Tsetlin Machine. We also provide examples of clauses formulated

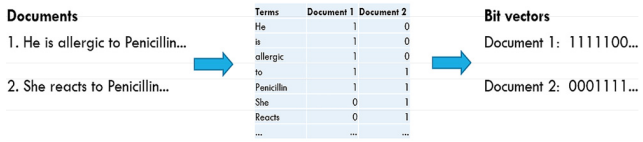


FIGURE 1. Example of bit vector representation of document sentences, taken from the medical dataset used in one of the experiments.

by the Tsetlin Machine, assessing their interpretability. Furthermore, we examine how parameter selection influences the classification performance. Finally, in Section 4, we conclude and provide directions for further research, including contextual language models for the Tsetlin Machine.

II. TEXT CATEGORIZATION WITH THE TSETLIN MACHINE

In this section, we first present our Tsetlin Machine based framework for text categorization. To highlight the key characteristics of the framework, we provide a running example from the medical domain – detection of an Allergy in the EHRs. Thereafter, we present the learning procedure itself, demonstrating step-by-step how the Tsetlin Machine extracts sophisticated linguistic patterns from labeled documents and simultaneously combats overfitting. The overall algorithm for text categorization is shown in Algorithm 1 and explained in the following.

A. REPRESENTING TEXT AS PROPOSITIONAL VARIABLES

The Tsetlin Machine takes a vector of k propositional variables, $\boldsymbol{X} = [x_1, x_2, \ldots, x_k]$, as input, each taking the value 0 or 1 (or equivalently, False or True). As illustrated in Fig. 1, we represent text (sentences, paragraphs, or in our case, documents) as follows. First, we form a vocabulary, $\boldsymbol{V} = \{t_1, \ldots, t_k\}$, consisting of each unique term t_o found in the target text corpus, $\boldsymbol{D}$. A propositional variable, x_o, $o \in \{1, \ldots, k\}$, is then introduced to represent each term, $t_o \in \boldsymbol{V}$. This allows us to model a document, $d \in \boldsymbol{D}$, as a vector, $\boldsymbol{X}$, of $|\boldsymbol{V}|$ propositional variables (stored as a vector of bits). The vector tells us which terms are present and which are absent in d.

Example: For the topic Allergy, terms such as "penicillin" and "reacts" would for instance be of importance, providing the propositional variables $x_{\text{"penicillin"}}$ and $x_{\text{"reacts"}}$.

B. LINGUISTIC PATTERNS IN CONJUNCTIVE NORMAL FORM

To build patterns that accurately capture categories, we compose propositional formulae in *conjunctive normal form*. These relate the propositional variables into compounds that trigger on evidence for or against a category.

Each compound, i.e., a conjunctive clause, is built by employing the conjunction operator on the propositional variables and their negations (referred to as *literals*):

$$\boldsymbol{C}_j(\boldsymbol{X}) = \boldsymbol{x}_{q_1} \wedge \boldsymbol{x}_{q_2} \wedge \ldots \wedge \boldsymbol{x}_{q_r} \wedge \neg \boldsymbol{x}_{q_{r+1}} \wedge \ldots \wedge \neg \boldsymbol{x}_{q_s}. \quad (1)$$

Algorithm 1 Function Categorize Text $(\mathbf{X}, \underline{\mathbf{C}})$

Input: A feature vector $\mathbf{X}$ marking the absence or presence of phrases; A set of clauses $C_j \in \underline{\mathbf{C}}$ produced by the Tsetlin Machine

Output: Categorization y

Begin

1. Evaluate each clause, $C_j \in \underline{\mathbf{C}}$, with $\mathbf{X}$ as input.
2. Sum up output of clauses (votes with positive and negative polarity [see Section II.C]):
$$f_\Sigma(\boldsymbol{X}) = \left(\sum_{j=1,3,\ldots}^{m-1} C_j(\boldsymbol{X})\right) - \left(\sum_{j=2,4,\ldots}^{m} C_j(\boldsymbol{X})\right).$$
3. The final output, y, is decided by thresholding $f_\Sigma(\boldsymbol{X})$ to identify the category:
$$y = 1 \text{ if } f_\Sigma(\boldsymbol{X}) > 0, \text{ otherwise } 0.$$
4. Return y.

End

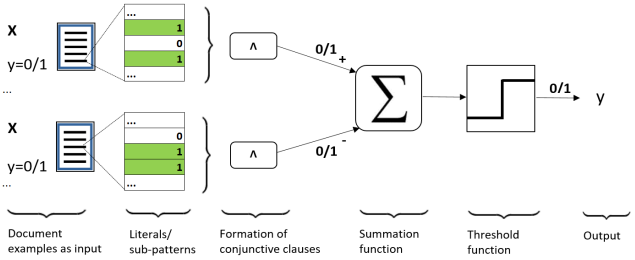


FIGURE 2. The Tsetlin Machine architecture, introducing clause polarity, a summation operator collecting "votes," and a threshold function arbitrating the final output.

In (1), q_u are indexes from $\{1, \ldots, k\}$ that identify which propositional variables take part in the conjunction. Note that empty clauses (those without literals) output 1 during training and 0 during categorization.

Example: For instance, the clause "rash" $\wedge$ "reaction" $\wedge$ "penicillin" can act as evidence (a vote) for the category Allergy.

The beauty of the conjunctive normal form is that it can capture arbitrarily refined patterns by looking at multiple terms in conjunction, forming a global view of the input vector. This is opposed to modeling features independently, as done by linear models like the Naïve Bayes classifier.

Finally, note the similarity between the conjunctive clauses in our model and the Boolean model of information retrieval, where Boolean expressions of terms are used to form queries by means of the AND-, OR-, and NOT operators [11].

C. CATEGORIZATION: ADDING UP EVIDENCE FROM CLAUSES

As illustrated in Fig. 2, in order to provide a rich and robust representation of each category, a Tsetlin Machine utilizes multiple clauses, $\underline{\boldsymbol{C}} = \{C_1, \ldots, C_m\}$, forming an ensemble of patterns. Each clause is further assigned a polarity (+/−). We use odd indexed clauses to capture patterns that provide evidence on the *presence* of a category (positive polarity),

while even indexed clauses capture evidence on the *absence* of the category (negative polarity).

As further depicted in Fig. 2, evidence obtained from clauses are aggregated by summation:

$$f_{\Sigma}(\boldsymbol{X}) = \left(\sum_{j=1,3,\ldots}^{m-1} C_j(\boldsymbol{X}) \right) - \left(\sum_{j=2,4,\ldots}^{m} C_j(\boldsymbol{X}) \right) \quad (2)$$

The concluding categorization is decided from the sign of $f_{\sum}(X)$:

$$y = 1, \quad \textbf{if } f_{\sum}(\boldsymbol{X}) > 0; \text{ otherwise } y = 0. \quad (3)$$

That is, only when the number of clauses providing a positive output outweigh those with negative output is the document assigned the target category ($y = 1$, i.e., Allergy in our case).

Altogether, this means that the Tsetlin Machine both learns what a category looks like, and what it *does not* look like. By summing evidence for and against the category, the threshold mechanism arbitrates the final decision.

Example: Recall our clause: "rash" ∧ "reaction" ∧ "penicillin." If the terms "rash," "reaction," and "penicillin" are all present in the document being categorized, the clause would evaluate to 1. This outcome would count as evidence for Allergy if the clause has positive polarity, and if none of the clauses with negative polarity evaluates to 1, the document would be assigned the category Allergy ($y = 1$).

D. LEARNING THE COMPOSITION OF CLAUSES

We now turn to learning of the propositional formula $\Phi(\boldsymbol{X}) = C_1 \bigvee \ldots \bigvee C_m$ for categorizing text. The Tsetlin Machine addresses this challenge by decomposing the problem into a large number of simple decisions. These decisions are coordinated by a novel decentralized game between independent Tsetlin Automata – one automaton per clause per literal [19]. Each automaton decides whether to include or exclude the assigned candidate literal in the given clause. That is, whether a literal should take part in a clause is collectively regulated by the competing presence of multiple candidate literals.

As shown in Algorithm 2, the learning is performed online, processing one training example, $(\boldsymbol{X}, y)$ at a time. The input vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_o]$ dictates which terms are present and which are absent in the current document. The target, y, to be predicted is simply the category of the document (Allergy or No Allergy in our case).

The Tsetlin Automaton: The Tsetlin Automaton that we use can be defined as a quintuple [22]: $\{\underline{\boldsymbol{\Phi}}, \underline{\boldsymbol{\alpha}}, \underline{\boldsymbol{\beta}}, F(\cdot,\cdot), G(\cdot)\}$.

$\underline{\boldsymbol{\Phi}} = \{\varphi_1, \varphi_2, \ldots, \varphi_{2N}\}$ is the set of internal states; $\underline{\boldsymbol{\alpha}} = \{\alpha_{Exclude}, \alpha_{Include}\}$ is the set of automaton actions; and $\underline{\boldsymbol{\beta}} = \{\beta_{Inaction}, \beta_{Penalty}, \beta_{Reward}\}$ is the set of inputs that can be given to the automaton. An output function $G[\varphi_t]$ determines the next action performed by the automaton given the current automaton state: (1) $G[\varphi_t] = \alpha_{Exclude}$ **for** $1 \le t \le N$; and (2) $G[\varphi_t] = \alpha_{Include}$ **for** $N + 1 \le t \le 2N$. Finally, a transition function $F[\varphi_t, \beta_u]$ determines the new automaton state from:

(1) the current automaton state and (2) the response of the environment to the action performed by the automaton.

Algorithm 2 Function Learning Propositional Formula ($\mathbf{D}$, m, T, s)

Input: Training data $(\boldsymbol{X}, y) \in \boldsymbol{D}$, number of clauses m, target T, specificity s

Output: Propositional formula $\Phi(\boldsymbol{X})$

Begin

The Tsetlin Machine learns $\Phi(\boldsymbol{X})$ by means of a *game*:

1. The arrival of a labeled document $(\boldsymbol{X}, y)$ from $\boldsymbol{D}$ signals the start of new game round.
2. Each Tsetlin Automaton decides whether to include or exclude its designated literal, leading to a new configuration of clauses $\boldsymbol{C}$. Recall that each decision is based on the state of the Tsetlin Automaton, as exemplified in Fig. 3.
3. Each clause, $C_j \in \underline{\boldsymbol{C}}$, is then evaluated with $\boldsymbol{X}$ as input.
4. We use the parameter T as a target for the summation $f_{\Sigma}(\boldsymbol{X})$. This parameter decides the update probability of each clause (see Section II.D).
5. If a clause is selected for updating, each of its Tsetlin Automata are independently and randomly given either Reward, Inaction, or Penalty feedback, according to Table 1 or Table 2, parameterized by s:
   a. If $y = 1$ the Type I Feedback Table is activated, while the Type II Feedback Table is activated if $y = 0$ (reversed for negative clauses).
   b. The probability of each kind of feedback is decided by the action of the automaton (either include or exclude literal x_k / $\neg x_k$), the value of the target clause, C_j, and the value of the literal x_k / $\neg x_k$.
6. The Tsetlin Automata update their states based on the feedback, exemplified in Fig. 3.
7. Goto 1 if the stopping criteria (e.g., the number of epochs) is unfulfilled.
8. Return propositional formula $\Phi(\boldsymbol{X}) = C_1 \bigvee \ldots \bigvee C_m$ obtained from the last Tsetlin Automata decisions.

End

Briefly, we have: (1) $\varphi_{t+1} = F[\varphi_t, \beta_{Penalty}]$ **for** $1 \le t \le N$; (2) $\varphi_{t-1} = F[\varphi_t, \beta_{Penalty}]$ **for** $N + 1 \le t \le 2N$; (3) $\varphi_{t-1} = F[\varphi_t, \beta_{Reward}]$ **for** $1 < t \le N$; $\varphi_{t+1} = F[\varphi_t, \beta_{Reward}]$ **for** $N + 1 \le t < 2N$; and (4) $\varphi_t = F[\varphi_t, \cdot]$ **otherwise** [19]. The crucial issue is to design automata that can learn the optimal action when interacting with the environment.

Example: Fig. 3 depicts six Tsetlin Automata with $N = 100$ states per action, collaborating to build a single clause. The three Tsetlin Automata to the left (TA) control the terms "to," "Voltaren," and "reacts," while the three to the right (TA') control the negation of the same terms. Those moving away from the central states in the figure ("Voltaren," "reacts," ¬"reacts," ¬"to," and ¬"Voltaren") have received a β_{Reward}, while those moving toward the center have received a $\beta_{Penalty}$.

TABLE 1. The type I feedback table has been designed to combat false negative output. It is triggered when a document correctly evaluates to true (i.e., both output value and target value are 1), or wrongly to false (i.e., output value is 0, while target value is 1).

Document→ Target Clause evaluates to / Target Literal evaluates to		1		0	
		1	0	1	0
Include literal	P (Reward)	$\frac{s-1}{s}$	NA	0	0
	P (Inaction)	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P (Penalty)	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
Exclude literal	P (Reward)	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$
	P (Inaction)	$\frac{1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P (Penalty)	$\frac{s-1}{s}$	0	0	0

TABLE 2. The type II feedback table combats false positive output. It is triggered when a document has wrongly evaluated to false, i.e. output value is 1, while the target value is 0.

Document→ Target Clause evaluates to / Target Literal evaluates to		1		0	
		1	0	1	0
Include literal	P (Reward)	0	NA	0	0
	P (Inaction)	1.0	NA	1.0	1.0
	P (Penalty)	0	NA	0	0
Exclude literal	P (Reward)	0	0	0	0
	P (Inaction)	1.0	0	1.0	1.0
	P (Penalty)	0	1.0	0	0

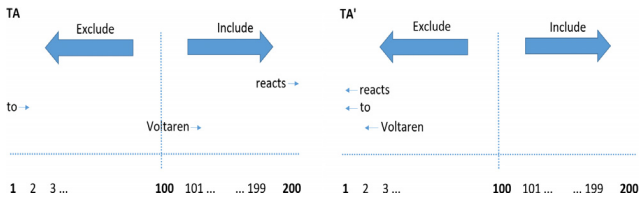


FIGURE 3. Six tsetlin automata with 100 states per action. Each automaton learns to either exclude or include a candidate literal (a term or its negation) in a clause.

Remark: Note that a state near 1 conveys very strong support for $\alpha_{Exclude}$, while a state close to $2N$ means strong support for $\alpha_{Include}$. The center reflects uncertainty.

The Nature of the Tsetlin Machine Game: The whole team of Tsetlin Automata, across all the categories and clauses, is orchestrated by means of a novel game [19]. In this game, the Tsetlin Automata partake as independent players. Two simple tables specify the complete multidimensional game matrix, where the Tsetlin Automata are the decision makers: (1) *Type I Feedback* and (2) *Type II Feedback*.

As shown in Fig. 3, a Tsetlin Automaton that receives a Reward moves away from the center (the Tsetlin Automaton becomes more confident in the action it currently selects). Conversely, when receiving a Penalty, the state of a Tsetlin Automaton shifts toward the center states, i.e., the states N and $N+1$, signaling increased uncertainty.

The *Type I Feedback Table* is activated when a document is either correctly assigned the target category (true positive), or mistakenly ignored (false negative). The feedback given by this table has two effects. First, clauses are refined by introducing more literals from the document. Left alone, this effect will make the clause *memorize* the document, by including *every* literal. However, this effect is countered by the second effect. The second effect is weaker by a factor of s, seeking to make all clauses evaluate to 1, whenever a document of the target category appears. This effect counters overfitting. Indeed, the "granularity" of the clauses produced can be controlled by means of s, decided by the user.

The second table, the *Type II Feedback Table*, is activated when a document is wrongly assigned the target category (a false positive categorization). The effect of this table is to introduce literals that render the clauses false whenever they face a document of the incorrect category.

Note that the above reasoning applies to clauses with positive polarity. For clauses with negative polarity, Type I Feedback takes the role of Type II Feedback, and vice versa.

Both of the tables are thus interacting, making the whole game of Tsetlin Automata converge toward robust pattern recognition configurations.

As detailed further in [19], in order to effectively utilize sparse pattern representation capacity (a constrained number of clauses), we introduce the user set parameter T as a target for the summation $f_\Sigma(\boldsymbol{X})$. In all brevity, the probability of activating *Type I Feedback* is: $(T - max(-T, min(T, f_\Sigma(\boldsymbol{X}))))/2T$, while *Type II Feedback* is activated with probability: $(T + max(-T, min(T, f_\Sigma(\boldsymbol{X}))))/2T$. If the votes accumulate to a total of $+/-T$ or more, neither rewards or penalties are handed out to the involved Tsetlin Automata. This mechanism thus helps to alleviate the vanishing signal-to-noise ratio problem [19], as it effectively removes clauses (and their literals) that already are able to capture a pattern (supporting the correct classification) from further feedback.

Remark: The computational simplicity and small memory footprint of Tsetlin Automata, combined with the decentralized nature of the above game, makes the Tsetlin Machine particularly attractive for execution on GPUs. This is explored further in the empirical results section.

E. DETAILED WALKTHROUGH OF THE LEARNING STEPS

Fig. 4 provides a step-by-step walkthrough that exemplifies how the Tsetlin Machine is able to gradually learn the target concept Allergy, building and refining clauses, document-by-document. For the sake of clarity, we focus on a substring of each document, although the steps we now describe occur for each term of the vocabulary, for each document processed. We explain the figure from left to right, starting with Example 1 (from the top).

Processing of Document 1: The first document, d_1, contains the substring "reacts to Voltaren." Each of these terms are associated with propositional variables, say, x_{29123}, x_{32232}, *and* x_{37372}. Again, for clarity, we focus on two of the clauses

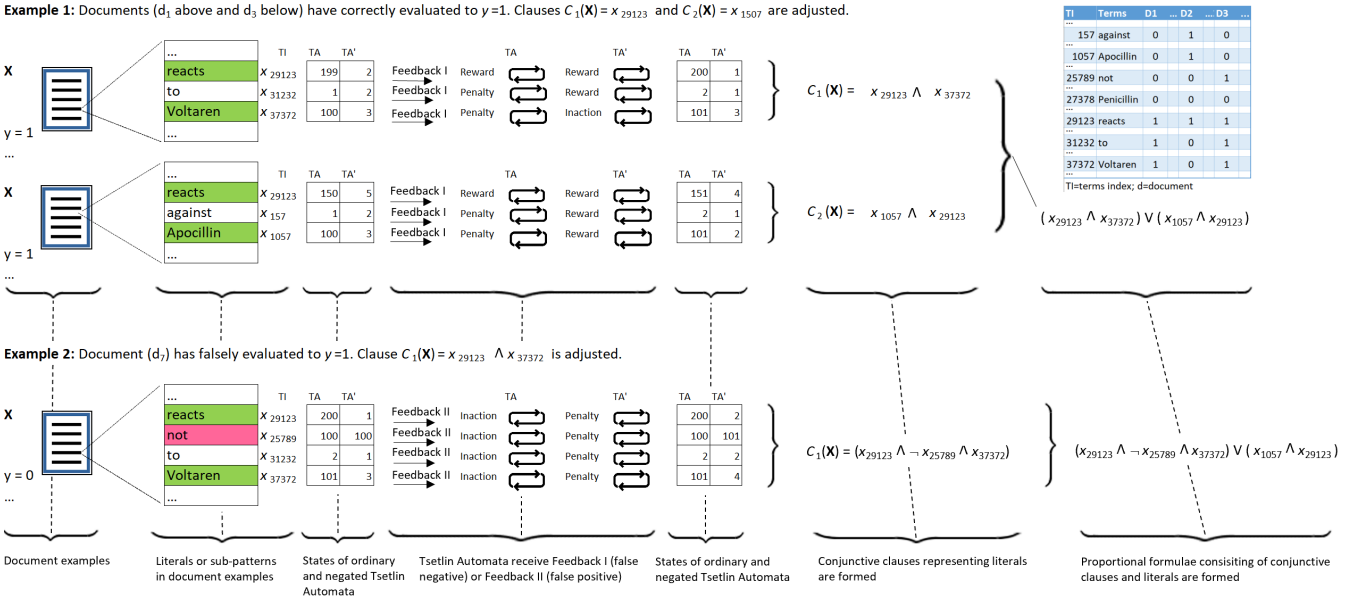


FIGURE 4. **Step-by-step example of Tsetlin Machine-based learning, demonstrating how clauses are composed and refined.**

associated with the target concept, $C_1(\boldsymbol{X})$ and $C_2(\boldsymbol{X})$, both with positive polarity. Each clause may potentially include any of the propositional variables, as well as their negation. Thus, we have two Tsetlin Automata for each term, for every clause. The first Tsetlin Automaton in the pair (referred to as TA) decides whether to exclude or include the propositional variable as is. The second one (referred to as TA') decides whether to exclude or include the *negation* of the variable.

Starting from the left, the columns TA and TA' list the state of each of the associated automata controlling $C_1(\boldsymbol{X})$, before they have been updated. As seen, the term "reacts" is already included in $C_1(\boldsymbol{X})$ because the state of the corresponding automaton is greater than or equal to $N + 1$ (i.e., 101). It is further included with confidence because the state is almost at the extreme end, i.e., $2N$. The other two terms, "to" and "Voltaren," are excluded: "to" is excluded with confidence (state 1), and "Voltaren" is excluded with uncertainty (state 100). Altogether, this means that the clause takes the form $C_1(X) = x_{29123}$ initially.

With $y = 1$, document d_1 has been flagged as an example of the target category. Accordingly, the *Feedback Type I Table*, described previously, is activated as depicted in the figure. The new updated states of the Tsetlin Automata are listed next. As seen, after the update, "Voltaren" is now also included in the clause, increasing categorization precision: $C_1(X) = x_{29123} \wedge x_{37372}$.

Processing of Document 3: Further on in the processing, another document, d_3, with target $y = 1$ comes along. This time $C_1(\boldsymbol{X})$ evaluates to 0, because "Voltaren" is absent from d_3. However, another clause $C_2(X) = x_{29123}$ happens to only include "reacts" and, thus, evaluates to 1. This in turn, triggers *Feedback Type I* for clause $C_2(\boldsymbol{X})$, again leading to an update of the Tsetlin Automata states. As seen, the "Apocillin" is now included in the clause because the associated Tsetlin Automaton changed action by going from state 100 to 101: $C_2(X) = x_{29123} \wedge x_{1057}$.

Processing of Document 7: Document d_7 is an example of a document not belonging to the category, with $y = 0$. This example highlights the difficulty of the text categorization task at hand. Indeed, $C_1(\boldsymbol{X})$ evaluates to 1 because the terms "reacts" and "Voltaren" are present, leading to a false positive categorization. However, this activates Feedback Type II. This time, clause $C_1(\boldsymbol{X})$ is updated by attempting to modify the clause so that it evaluates to 0 instead. This is done by penalizing those exclude actions that would make the clause evaluate to 0 had they been replaced by an include action. In our example, the most uncertain of those being penalized is the literal that negates "not." After receiving *Feedback Type II*, this literal is now being included in the clause: $C_1(X) = x_{29123} \wedge x_{37372} \wedge \neg x_{25789}$. In this case, the false positive categorization was eliminated by this simple change. In general, however, the game needs to be played for several epochs before all of the clauses find their final form.

III. EXPERIMENTS

In this section, we describe the experimental setup, including construction of the datasets and NLP preprocessing techniques, and the experimental setup. We further evaluate our Tsetlin Machine-based text categorization approach empirically, in comparison with other techniques. To this end, we use two public datasets and a proprietary one, accompanying this paper: 1) The 20 Newsgroups dataset [23], [24]; 2) The IMDb dataset [25]; and 3) A clinical dataset with authentic EHRs from a hospital [4]. The first two datasets facilitate comparisons with previous research, while the third focuses on real-life performance on a challenging medical application. We also conduct experiments to investigate the learning behavior and the execution speed of the Tsetlin

Machine, including performance statistics per epoch. We further discuss examples of clauses formulated by the Tsetlin Machine, and finally examine how parameter selection influences classification results.

A. DATASET CONSTRUCTION

1) THE 20 NEWSGROUPS DATASET

This publicly available dataset is a collection of 20000 newsgroup documents, partitioned (nearly) evenly across six categories (i.e., computers, recreation, science, politics, religion, and forsale) and 20 subcategories [23], [24]. The dataset has been widely used to evaluate text classification algorithms. We use the ''bydate'' version of the dataset containing 18 846 documents, recommended by the data provider as being more realistic. This specific version of the dataset has document duplicates and newsgroup-identifying headers removed. It is further split into a training set containing 11 314 documents and a test set containing 7532 documents, separated in time.

2) THE IMDb DATASET

The IMDb dataset was originally created by Maas et al. [25] for learning word vectors targeting sentiment analysis. The balanced dataset contains 100 000 reviews of movies, where half of them are labeled (negative and positive sentiment) and the other half are unlabeled. We refer to Maas *et al.* [25] for further details on the dataset.

3) THE CLINICALLY DERIVED DATASET

This dataset contains complete authentic EHRs and is derived from the Sørlandet Hospital Trust's enterprise-wide integrated EHR system in Norway. The dataset has been manually curated by health professionals to create a gold-standard for allergy-relevant information. It contains approximately 20 000 clinical notes belonging to patients who were admitted to the hospital for orthopedic surgical procedures performed between January 1, 2014 and December 31, 2015. Based on the annotated allergy information, each of the patient cases was manually assigned either the category Allergy or the category No Allergy. The dataset was also randomly reduced in size to 431 patient EHRs (76 319 documents, ∼15 951 K words, and ∼104 M tokens in total) to achieve a sufficiently balanced dataset. Finally, classification of whether a patient has allergies or not is performed by processing the collective content of each patient's EHR. To facilitate reconstruction of our experiments with the Tsetlin Machine, an anonymized bit vector representation of the dataset is available at https://github.com/cair/TextUnderstandingTsetlinMachine.

B. NLP PREPROCESSING

The datasets were converted into a sparse matrix binary format for the Tsetlin Machine to process further. This also included building a dictionary of all the dataset-specific terms. NLP preprocessing was an iterative process where we experimented with different techniques and configurations to understand how it affected the Tsetlin Machine. NLP preprocessing for the final results included lowering case, removal of noninformative punctuation, and tokenization. We did not remove stop words in our classification experiments, and we considered a token to be any sequence of symbols, separated by white space. For the IMDb dataset, we calculated term frequency–inverse document frequency (tf–idf) for all features first, computed threshold values [26], and finally applied chi-squared statistics to determine the 7500 most significant features. A chi-squared test statistic was used to determine the 15 000 most significant features for the 20 Newsgroups dataset, while mutual information was employed to identify the top 38876 features for the clinical dataset [27]. NumPy [28], Scikit-learn [29], and the Natural Language Toolkit (NLTK) [30] Python libraries were used for programming the data transformation steps.

For the Weka processed datasets, the StringToWordVector filter was used to perform cleaning and tokenization of the input data into unigrams, and tf–idf together with the InfoGainAttributeEval method (used to calculate mutual information) were employed to determine the most significant attributes.

C. EXPERIMENTAL SETUP AND EVALUATION METRICS

The experiments were conducted on server A (NVIDIA DGX-2 with dual Intel Xeon Platinum 8168, 2.7 GHz, 24-cores, 30 TB SSD, 1.5 TB memory, 16X NVIDIA Tesla V100 512 GB, and Ubuntu 18.04 LTS x64) and B (Intel i7-8700K 6-cores CPU, 64GB of memory, GeForce GTX 1080 Ti, and Ubuntu 18.04 LTS x64).

We adopted the Python (with Cython C/C++ extensions) or CUDA/GPU capable C++ version of the Multiclass Tsetlin Machine to run the experiments.

We further used Weka (v3.9) [31], StarSpace [32], Scikit-learn, Keras [33], and TensorFlow [33] to produce the results of the other evaluated methods. This included the Naïve Bayes classifier (multinomial Naïve Bayes), logistic regression, decision tree, random forest, linear SVM, kNN, multilayer perceptron (MLP), LSTM, LSTM CNN, bidirectional LSTM (Bi-LSTM), and Bi-LSTM CNN.

The classifiers we employed in Weka were the Naïve Bayes classifier, decision tree (J48/C4.5), logistic regression (multinomial logistic regression), random forest, kNN, linear SVM (sequential minimal optimization), and MLP (DL4jMlp) [34].

Note that we used both CPU and GPU implementations of the Tsetlin Machine for the classification experiments. The fast GPU version of the Tsetlin Machine employs bitwise operators and executes up to 37 times faster than the CPU implementation, depending on the dataset. Our CUDA source code for text classification with the Tsetlin Machine can be found at https://github.com/cair/TextUnderstanding Tsetlin-Machine.

Although we conducted extensive hyperparameter optimization for the Tsetlin Machine (manual grid search) and the other evaluated algorithms (e.g., automatic tuning by

use of the CVParameterSelection metafilter in Weka, GridSearchCV in Keras, and Hyperas/Hyperopt in Python) [35], even better configurations may potentially exist. We have, however, strived to put an equal effort into optimizing each approach to facilitate a fair comparison. Concerning the neural network methods specifically, parameters such as embedding size, optimization algorithm, learning rate, dropout, batch size, sequence length, the number of LSTM cells, and CNN filter and kernel sizes, were tuned for each dataset. The neural networks were further configured to train for a maximum of 200 epochs. Ten percent of the training data was randomly selected as validation data, and early stopping was implemented as a strategy to stop the training if the validation loss did not increase for 15 consecutive epochs.

We used the authors' published training/test data split for the experiments on the 20 Newsgroups (60/40) and IMDb (50/50) datasets [23], [25], and performed 100 independent runs to facilitate statistically robust comparisons of the different techniques. For the clinical dataset we performed a 10-fold cross-validation, repeated 10 times to produce representative averages with small variance. The clinical dataset is characterized by its complexity and relatively few samples. We experimented with how to best partition the available data into a test and training set, before finally settling on a 90/10 split, which in our tests delivered the best overall results.

The evaluation metrics we employed in the experiments were macro-averaged accuracy, recall, precision, and F-measure. The results are presented as the mean percentages, with 95% confidence intervals.

D. RESULTS

1) THE 20 NEWSGROUPS DATASET

The results for the 20 Newsgroups dataset with 20 classes are reported in Table 3. The Tsetlin Machine we employed here was configured using 15 000 selected features, 15 000 clauses, and 100 states. Furthermore, we used an s-value of 50 and a summation target T of 60. We report a conservative estimate of performance – performance after the 200th epoch. This performance estimate is conservative because, as seen in Fig. 5, the Tsetlin Machine is continuously exploring the configuration space, and the best performing configuration is not always manifested in the final epoch. However, the estimate reflects the average behavior of the steady state of the underlying learning process.

As Table 3 shows, except for the SVM and the logistic regression, the Tsetlin Machine outperformed the other baselines. The recall scores of the SVM and the logistic regression are higher than the Tsetlin Machine's recall score, while their precision scores are almost equal. Considering the three methods' F-measure scores in light of the confidence intervals, they perform quite similarly.

The macro-averaged accuracy of the Tsetlin Machine in this experiment was 81.5% ± 0.1%, with 82.7% being the highest accuracy over all the epochs and experiments.

TABLE 3. Average classification results (in percent) for the 20 newsgroup dataset.

Method	Precision	Recall	F-measure
Multinomial Naïve Bayes	82.8±0.0	80.0±0.0	79.8±0.0
Logistic regression	**83.1±0.0**	81.7±0.0	81.9±0.0
Decision tree	56.8±0.0	61.8±0.4	59.2±0.0
Random forest	69.9±0.1	68.2±0.1	68.3±0.1
kNN	56.0±0.0	43.3±0.0	45.9±0.0
Linear SVM	82.6±0.0	**82.0±0.0**	**82.2±0.0**
MLP	80.9±0.0	78.1±0.0	76.9±0.0
LSTM	80.4±0.0	72.6±0.1	76.3±0.0
LSTM CNN	82.8±0.0	72.8±0.1	76.7±0.0
Bi-LSTM	80.9±0.0	72.6±0.1	76.5±0.0
Bi-LSTM CNN	81.8±0.1	72.3±0.3	76.7±0.2
Tsetlin Machine	82.6±0.1	80.9±0.1	81.7±0.1

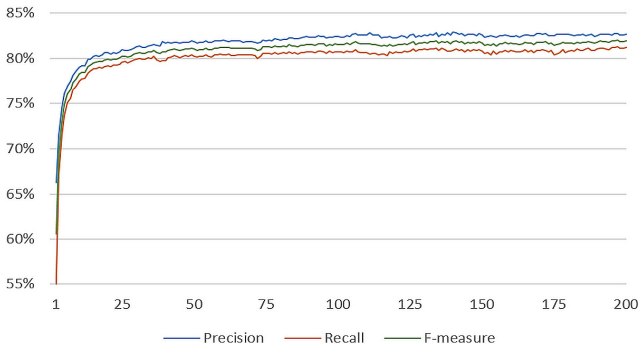


FIGURE 5. The learning behavior (y-axis is score) of the Tsetlin Machine on the 20 newsgroups dataset across 200 epochs (x-axis is epochs).

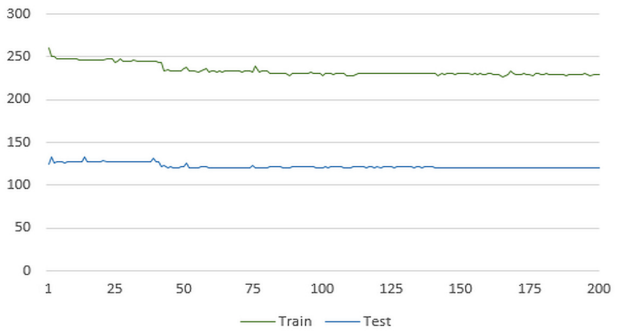


FIGURE 6. The execution time in seconds (y-axis) for the fast GPU version of the Tsetlin Machine on the 20 newsgroups dataset across 200 epochs (x-axis) on server A.

Fig. 5 depicts the learning behavior of the Tsetlin Machine on the dataset over 200 epochs in a single run. As seen, the accuracy increases consistently across the epochs, without any indication of overfitting. As further observed, the precision and recall scores are quite similar, with the difference of the two mostly falling below 3% over the epochs. The time spent per epoch to train and evaluate the fast GPU version of the Tsetlin Machine is plotted in Fig. 6.

As observed, when compared to Fig. 5, the training time drops when the Tsetlin Machine is close to convergence. While spending approximately 260 seconds for the initial epoch, the training time gradually falls to 243 seconds in

epoch 42, after which it suddenly drops to approximately 233 seconds and remains around 230 seconds for the remaining epochs.

To our knowledge, the best accuracy score (86.1%) on this particular dataset (the ''bydate'' version) was recently reported by Papagassi et al. using an end-to-end multiscale CNN framework [36]. Zheng *et al.* [37] employed a bidirectional hierarchical skip-gram (BHSG) model for text topic embedding and recorded an accuracy score of 85.4%. They also included a baseline result for SVM (82.8%) that is similar to our linear SVM result (82.2%). Using a regularized least squares classifier (optimal regularization chosen post hoc on test set), Rennie [38] achieved an accuracy score of 84.9%, while Feng et al. reported an accuracy score of 82% with AdaBELM [39]. In a comparative study on term weighting schemes for text classification, Mazyad et al. recorded macro-averaged F-measure of 69% for SVM, 68.9% for stochastic gradient descent (SGD), and 49.1% for a decision tree [40]. By implementing error-correcting output coding (ECOC) together with multinomial Naïve Bayes, Li and Vogel [41] reported average macro-accuracy of 81.8%. Using discriminative restricted Boltzmann machines to classify the 20 Newsgroups dataset, Larochelle and Bengio recorded an accuracy score of 76.2% [42].

Note that the highest performing scores on this dataset were achieved by including more features (e.g., Papagassi et al. used a vocabulary of 53 160 words) than what we used in our experiments [36]. Our results also indicate that the accuracy of the Tsetlin Machine increases with the number of features. However, a sparse version of the Tsetlin Machine, or alternatively more memory, is needed to go significantly beyond the 15 000 features we used for this dataset.

2) THE IMDB DATASET

Table 4 shows the results for the IMDb dataset. The Tsetlin Machine was here configured to use 7500 selected features, 20 000 clauses, and 256 states. Furthermore, we used an s-value of 27.0, a target T of 80, and ran the Tsetlin Machine for 200 epochs. For each run, we select the Tsetlin Machine configuration that obtains the highest accuracy on the training data, and report the corresponding performance on the test data.

As observed in Table 4, the Tsetlin Machine outperformed the other methods on this dataset. The macro-averaged accuracy of the Tsetlin Machine in this experiment was 89.2% $\pm$ 0.2%, with 89.7% being the highest accuracy over all the epochs and folds.

Utilizing a CNN and a LSTM network to classify the same data, Yenter and Verma [43] achieved maximum accuracy scores of between 89.22% and 89.5% on five different models, and thus concluded that their proposed model outperformed prior relevant research [25], [44], [45].

Recently, several authors have also reported even better results by using ensemble methods or by introducing combinations of techniques such as entropy minimization loss, as well as adversarial and virtual adversarial training [46]–[55]. For example, Wang and Manning reported an accuracy of 91.22% by implementing SVM with NB features (NBSVM-bi) [46]. Using deep learning, Miyato *et al.* [51] were able to achieve 94.09% accuracy by using adversarial training methods. Finally, by implementing a mixed objective function (L-mixed) and word embedding, Singh Sachan et al. obtained an accuracy of 95.68% and concluded that the good performance of their model mostly comes from fine-tuning the pretrained embedding layer and by adjusting the gradient norm clipping threshold value [52]. In our forthcoming work, we therefore intend to investigate how we can enhance our relatively plain modeling of text (presence and absence of terms) to further increase accuracy, for example by introducing pretrained word embedding.

TABLE 4. Average classification results (in percent) for the IMDb dataset.

Method	Precision	Recall	F-measure
Multinomial Naïve Bayes	85.9±0.0	86.1±0.0	86.0±0.0
Logistic regression	86.5±0.0	87.6±0.0	87.1±0.0
Decision tree	71.1±0.0	68.4±0.0	69.7±0.0
Random forest	78.9±0.1	78.1±0.1	78.5±0.1
kNN	58.4±0.0	63.5±0.0	60.8±0.0
Linear SVM	88.0±0.0	89.1±0.0	88.5±0.0
MLP	82.6±0.1	82.6±0.1	82.6±0.1
LSTM	87.2±0.7	84.3±0.9	85.6±0.6
LSTM CNN	89.5±0.2	86.8±0.4	88.1±0.1
Bi-LSTM	87.6±0.8	83.9±1.1	85.5±0.6
Bi-LSTM CNN	88.3±0.2	87.5±0.5	87.9±0.2
Tsetlin Machine	**89.7±0.0**	**89.7±0.0**	**89.7±0.0**

Example Clauses: Table 5 shows five examples of typical clauses formulated by the Tsetlin Machine on the IMDb dataset (grouped according to type as A, B, and C). Clauses in group A are examples of positive and negative single phrase clauses. Group B gives examples of clauses where two phrases interact to form more intricate meanings. For example, the phrase ''Wonderful movie'' is modified by the phrase ''but then,'' which in combination have a negative meaning in this case. Finally, the group C clause is an example of a positive clause containing the phrase ''a unique,'' further modified or restricted by a range of negations that effectively serve to eliminate different textual contexts.

3) THE CLINICAL DATASET

Table 6 displays results for the clinical dataset. To produce the final results presented here, the Tsetlin Machine was configured to use 38 876 features, 500 clauses, 100 states, an s-value of 3.0, a summation target T of 25, and to run 2 epochs per fold. When considering the F-measure results, the Tsetlin Machine beats the Naïve Bayes classifier by a small margin, with logistic regression in third place. Given the confidence intervals, however, the difference in performance of the three top contenders becomes statistically insignificant. Interestingly, the two linear algorithms outperformed all the other methods, including linear SVM and the usually more capable nonlinear algorithms (i.e., the neural networks).

TABLE 5. Examples of clauses formulated by the Tsetlin Machine on the IMDb dataset.

Negative clauses	Positive clauses
A. "pretty lame"	A. "endearing"
B. "wonderful movie" AND "but then"	C. "a unique" NOT "it brought" NOT "lumet"
B. "2 out" AND "garbage br"	NOT "passion" NOT "there's no" NOT "really enjoyed" NOT "sex" NOT "supporting" NOT "individual" NOT "profound" NOT "his finest" NOT "storyline" NOT "poorly" NOT "most underrated" ...

TABLE 6. Average classification results (in percent) for the clinical dataset.

Method	Precision	Recall	F-measure
Naïve Bayes	65±1.4	**76.2±1.8**	69.9±1.3
Logistic regression	**73.4±1.8**	66.9±1.8	69.6±1.4
Decision tree	59.7±1.8	57.3±2.4	57.9±1.7
Random forest	63±1.8	56.1±2.0	58.9±1.7
kNN	68±2.9	36.8±2.1	47.1±2.2
Linear SVM	67.5±1.3	66.5±1.3	66.3±1.3
MLP	70.3±3.5	67.1±2.4	67.2±2.1
LSTM	68±1.2	65.9±1.2	65.6±1.3
LSTM CNN	65.8±2.0	66.4±1.1	65.1±1.5
Bi-LSTM	65.9±1.5	65.2±1.5	65.6±1.5
Bi-LSTM CNN	67.4±1.3	66.1±1.3	65.2±1.4
StarSpace	61.7±2.6	59.8±1.9	59.3±1.8
Tsetlin Machine	70.7±1.1	69.6±1.1	**70.2±1.1**

A peak accuracy of 79.1% was achieved in epoch 294 when running the Tsetlin Machine for an extended total of 2000 epochs, possibly reflecting temporary overfitting on the test set (in later epochs, the accuracy results stabilized at approximately 68–69%). It is reasonable to believe that the Naïve Bayes and logistic regression classifiers in this experiment gained from their simple (linear) structure, preventing them from fitting the training data too closely [56]. Similar to Naïve Bayes, the Tsetlin Machine has also previously demonstrated accuracy advantages when data is sparse, which may possibly explain why the Tsetlin Machine is able to outperform the other nonlinear capable algorithms employed in this experiment [19].

The results reported in Table 7 reflect how selected configurations of clauses, target T, and s parameters impact the F-measure (the other parameters are kept unchanged). On this particular dataset, an s-value from 1.5–9.5 gives the best results. Increasing the number of clauses to 2000 also seems to favor lower s-values. Additionally, note that we were able to achieve a second-best F-measure of 68.9% by using a configuration of 1000 clauses, a summation target T of 50, and an s-value of 2. In general, most of the configurations provide competitive results, indicating robust learning with the Tsetlin Machine across a variety of configurations. However, the results in Table 7 also indicate that the proposed method is sensitive to parameter selection. In our experience, tuning of the number of clauses, s, and T did not require more effort than manually configuring different deep learning techniques. However, because the Tsetlin Machine currently lacks the automatic hyperparameter optimization methods that are available for deep learning algorithms (e.g., Hyperopt), manual grid search has to be conducted. This means that it may take more time to tune the Tsetlin Machine models in practice. We also note that the clinical dataset is complex and has relatively few samples.

This makes the results on this dataset particularly prone to fluctuations.

As observed in Table 8, there are similarities between the explanatory power of the Tsetlin Machine-produced clauses and a decision tree. Both are produced in human-readable

TABLE 7. Average classification results (in percent) for the Tsetlin Machine using selected configurations of clauses, summation target *T*, and *s* parameters.

s	Clauses 500, *T* 25	Clauses 1K, *T* 50	Clauses 2K, *T* 100
s 1.5	66.1±0.8	64.4±1.6	**66.6±1.2**
s 2	64.8±2.0	**68.9±1.4**	64.4±1.1
s 3	**70.2±1.1**	62.9±1.2	65.6±1.5
s 6	66.3±1.5	66.6±1.5	65.9±1.5
s 9.5	65.2±1.1	66.6±1.2	63.5±0.9
s 12	64.5±1.6	63.2±1.7	61.6±1.9
s 15	58.6±1.7	62.1±1.8	62.8±1.5

TABLE 8. Comparison of a J48/C4.5 decision tree (right column) with example clauses formulated by the Tsetlin Machine (left column) on the clinical dataset.

Clauses	Decision tree
"Penicillin allergy" AND …	"allergy of pollen" <= 0
	"feets" <= 1.715296
"urticaria" AND …	"stiansand" <=0
	"carotid" <= 1.084065
"Cortisone" AND	"andr"<=0
"subcutaneous" AND …	"Zyrtec" <0
	"complicated" <=0
"Ventoline" and "performance ability" AND "work pressure" AND "triggered by effort" AND "grass" AND …	
"antibodies" AND "allergic" AND "circulatory" AND …	
"grass" AND "asthmatic" AND "short of breath" AND "itchy skin" AND "symptom picture" AND "Cetirizin" AND "hyperventilating" AND "causes" AND "episodes" AND "gluten-free" AND "hay-fever" AND ...	

form, making them easy to interpret. The results they produce here are informative from a medical domain perspective. For example, the drugs Zyrtec, Ventoline, Cortisone, and Cetirizin are all drugs that are relevant in the context of patient allergy. They also identify allergens (e.g., grass and pollen) and signs/symptoms/words (e.g., urticaria, complicated, and itchy skin) consistent with patient allergy. Both decision trees and clauses may further naturally be translated into an equivalent set of rules for incorporation into clinical decision support systems (CDSS). As shown in Table 5 and Table 8 (see also Fig. 4), the clauses already contain operator structures that may easily be included in e.g., if-then rules. Unlike decision trees, though, the Tsetlin Machine is able to combine interpretability with high accuracy performance (see Table 6), both of which are important to healthcare.

Similar to decision trees, phrases may falsely be included, such as the word ''stiansand'' in Table 8 (part of a city's name). Another example is the name of a hospital doctor that showed up as a phrase in some of the Tsetlin Machine produced clauses. Because of the approaches' transparency, falsely included phases can be manually pruned in a postprocessing step before inclusion into medical applications.

IV. CONCLUSION AND FUTURE WORK

This paper proposed a text categorization approach based on the recently introduced Tsetlin Machine. In brief, we represent the terms of a text as propositional variables. From these, we capture categories using simple propositional formulae that are easy to interpret for humans. The Tsetlin Machine learns these formulae from a labeled text, utilizing conjunctive clauses to represent the particular facets of each category. Our empirical results were quite conclusive. The Tsetlin Machine outperformed all of the evaluated methods on the IMDb dataset, while performing approximately on par with the best evaluated methods on the 20 Newsgroups and a clinical dataset containing authentic EHRs. On average, the Tsetlin Machine delivered the best recall and precision scores across the datasets. Furthermore, we observed that the explanatory power of the Tsetlin Machine-produced clauses seems to equal that of decision trees.

In our further work, we plan to examine how to use the Tsetlin Machine for unsupervised learning of word embeddings. Furthermore, we will investigate how the sparse structure of documents can be taken advantage of to speed up learning. We further plan to leverage local context windows to learn structure in paragraphs and sentences for more precise text representation. We are currently also conducting research into developing a convolutional version of the Tsetlin Machine to further increase accuracy performance [57]. Finally, based on our promising results we envision implementing the Tsetlin Machine in a CDSS [4]. In particular, we are interested in how structured medical data can be combined with the medical narrative to support precision medicine.

Some of the linear based algorithms (i.e., SVM, logistic regression, and Naïve Bayes) considered in this paper showed strong accuracy performance on all three included datasets. The experiments may therefore not have revealed the full power of the Tsetlin Machine, which also has the capability to address nonlinear patterns [19]. However, we note that the experiments conducted here show that the Tsetlin Machine is competitive also on simpler linear patterns, typical for text analysis tasks. Further experiments with the Tsetlin Machine should also be conducted on datasets that are unsolvable for linear algorithms.

In conclusion, we believe that our novel Tsetlin Machine-based approach can have a significant impact on a wide range of text analysis applications. Furthermore, we believe that the approach and results presented in this paper can provide a promising starting point for deeper natural language understanding.

REFERENCES

[1] P. Norvig, ''Inference in text understanding,'' in *Proc. AAAI*, 1987, pp. 561–565.
[2] X. Zhang, J. Zhao, and Y. LeCun, ''Character-level convolutional networks for text classification,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.
[3] P. Linell, ''The written language bias in linguistics: Its nature, origins and transformations,'' Dept. Commun. Stud., Univ. Linköping, Linköping, Sweden, Tech. Rep. SIC 2, 1982.
[4] G. T. Berge, O.-C. Granmo, and T. O. Tveit, ''Combining unsupervised, supervised, and rule-based algorithms for text mining of electronic health records—A clinical decision support system for identifying and classifying allergies of concern for anesthesia during surgery,'' in *Proc. ISD*, 2017. [Online]. Available: https://aisel.aisnet.org/isd2014/proceedings2017/CogScience/2/
[5] Y. Wang, ''Clinical information extraction applications: A literature review,'' *J. Biomed. Inform.*, vol. 77, pp. 34–49, Jan. 2018.
[6] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, ''Very deep convolutional networks for text classification,'' in *Proc. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, 2017, pp. 1107–1116.
[7] J. Schmidhuber, ''Deep learning in neural networks: An overview,'' *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[8] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal, ''Learning general purpose distributed sentence representations via large scale multi-task learning,'' 2018, *arXiv:1804.00079*. [Online]. Available: https://arxiv.org/abs/1804.00079
[9] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, ''Deep learning for healthcare: Review, opportunities and challenges,'' *Brief. Bioinform.*, vol. 19, no. 6, pp. 1236–1246, 2017.
[10] G. Valdes, J. M. Luna, E. Eaton, C. B. Simone, II, L. H. Ungar, and T. D. Solberg, ''MediBoost: A patient stratification tool for interpretable decision making in the era of precision medicine,'' *Sci. Rep.*, vol. 6, Nov. 2016, Art. no. 37854.
[11] C. D. Manning, and P. Raghavan, H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge Univ. Press, 2008.
[12] M. Wu, M. C. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez, ''Beyond sparsity: Tree regularization of deep models for interpretability,'' 2017, *arXiv:1711.06178*. [Online]. Available: https://arxiv.org/abs/1711.06178
[13] G. K. Zipf, *Selected Studies of the Principle of Relative Frequency in Language*. Cambridge, MA, USA: Harvard Univ. Press, 1932.
[14] T. Joachims, ''Text categorization with support vector machines: Learning with many relevant features,'' in *Proc. Eur. Conf. Mach. Learn.*, 1998, pp. 137–142.
[15] T. Joachims, ''A statistical learning model of text classification for SVMs,'' in *Learning to Classify Text Using Support Vector Machines*. Boston, MA, USA: Springer, 2002, pp. 45–74.
[16] Ö. Uzuner, B. R. South, S. Shen, and S. L. DuVall, ''2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text,'' *J. Amer. Med. Informat. Assoc.*, vol. 18, no. 5, pp. 552–556, Jun. 2011.
[17] B. D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter, and L. Floridi, ''The ethics of algorithms: Mapping the debate,'' *Big Data Soc.*, vol. 3, no. 2, 2016. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/2053951716679679

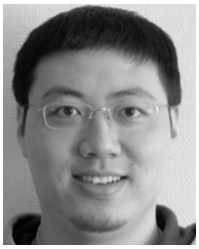
[18] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[19] O.-C. Granmo, "The Tsetlin Machine—A game theoretic bandit driven approach to optimal pattern recognition with propositional logic," 2018, *arXiv:1804.01508*. [Online]. Available: https://arxiv.org/abs/1804.01508
[20] M. L. Tsetlin, "On behaviour of finite automata in random medium," *Avtomatika I Telemekhanika*, vol. 22, no. 10, pp. 1345–1354, 1961.
[21] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "A Bayesian framework for learning rule sets for interpretable classification," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2357–2393, 2017.
[22] M. A. L. Thathachar and K. Narendra, *Learning Automata: An Introduction*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.
[23] (1995). *20 Newsgroups Data Set*. [Online]. Available: http://qwone.com/~jason/20Newsgroups/
[24] K. Lang, "Newsweeder: Learning to filter netnews," in *Proc. Mach. Learn.* Amsterdam, The Netherlands: Elsevier, 1995, pp. 331–339.
[25] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2011, pp. 142–150.
[26] K. D. Abeyrathna, M. Goodwin, X. Zhang, and O.-C. Granmo, "A scheme for continuous input to the Tsetlin Machine with applications to forecasting disease Outbreaks," in *Proc. Int. Conf. Ind., Eng. Appl. Appl. Intell. Syst.*, 2019, pp. 564–578.
[27] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. ICML*, vol. 97, 1997, pp. 412–420.
[28] *NumPy*. (2019). Accessed: Jan. 24, 2019. [Online]. Available: http://www.numpy.org/
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[30] E. Loper and S. Bird, "NLTK: The natural language toolkit," in *Proc. ACL Workshop Effective Tools Methodol. Teach. Natural Lang. Process. Comput. Linguistics*, 2002, pp. 63–70.
[31] E. Frank, M. A. Hall, and I. H. Witten. (2016). *The WEKA Workbench Online Appendix for, Data Mining: Practical Machine Learning Tools and Techniques*. Accessed: Jan. 24, 2019. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/
[32] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "StarSpace: Embed all the Things!" 2017, *arXiv:1709.03856*. [Online]. Available: https://arxiv.org/abs/1709.03856
[33] *Keras*. (2019). Accessed: Jan. 24, 2019. [Online]. Available: https://keras.io/
[34] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, and E. Frank, "WekaDeeplearning4j: A deep learning package for Weka based on Deeplearning4j," *Knowl.-Based Syst.*, vol. 178, pp. 48–50, Aug. 2019.
[35] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Comput. Sci. Discovery*, vol. 8, no. 1, 2015, Art. no. 014008.
[36] R. Pappagari, J. Villalba, and N. Dehak, "Joint verification-identification in end-to-end multi-scale CNN framework for topic identification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 6199–6203.
[37] S. Zheng, H. Bao, J. Xu, Y. Hao, Z. Qi, and H. Hao, "A bidirectional hierarchical skip-gram model for text topic embedding," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2016, pp. 855–862.
[38] J. D. Rennie, "On the value of leave-one-out cross-validation bounds," in *Computers in Biology and Medicine*. Amsterdam, The Netherlands: Elsevier, 2003, pp. 123–129.
[39] X. Feng, Y. Liang, X. Shi, D. Xu, X. Wang, and R. Guan, "Overfitting reduction of text classification based on AdaBELM," *Entropy*, vol. 19, no. 7, p. 330, 2017.
[40] A. Mazyad, F. Teytaud, and C. Fonlupt, "A comparative study on term weighting schemes for text classification," in *Proc. Int. Workshop Mach. Learn., Optim., Big Data*, 2017, pp. 100–108.
[41] B. Li and C. Vogel, "Improving multiclass text classification with error-correcting output coding and sub-class partitions," in *Proc. Can. Conf. Artif. Intell.*, 2010, pp. 4–15.
[42] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 536–543.
[43] A. Yenter and A. Verma, "Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis," in *Proc. IEEE 8th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2017, pp. 540–546.
[44] L. Rahman, N. Mohammed, and A. K. Al Azad, "A new LSTM model by introducing biological cell state," in *Proc. 3rd Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, 2016, pp. 1–6.
[45] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Syst. Appl.*, vol. 57, pp. 117–126, Sep. 2016.
[46] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2012, pp. 90–94.
[47] G. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio, "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews," 2015, *arXiv:1412.5335*. [Online]. Available: https://arxiv.org/abs/1412.5335
[48] Z. Zhao, "Learning document embeddings by predicting N-grams for sentiment classification of long movie reviews," presented at the Workshop Contribution (ICLR), 2016. [Online]. Available: https://zhezhaoa.github.io/pub/learning_document_embeddings_by_predicting_n-grams_for_sentiment_classification_of_long_movie_reviews.pdf
[49] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," in *Proc. Conf. North Amer. Chapter Assoc. for Comput. Linguistics, Hum. Lang. Technol.*, 2015, pp. 103–112.
[50] A. B. Dieng, C. Wang, J. Gao, and J. Paisley, "TopicRNN: A recurrent neural network with long-range semantic dependency," 2016, *arXiv:1611.01702*. [Online]. Available: https://arxiv.org/abs/1611.01702
[51] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016, *arXiv:1605.07725*. [Online]. Available: https://arxiv.org/abs/1605.07725
[52] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting LSTM networks for semi-supervised text classification via mixed objective function," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 6940–6948.
[53] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," 2017, *arXiv:1704.01444*. [Online]. Available: https://arxiv.org/abs/1704.01444
[54] S. Gray, A. Radford, and D. P. Kingma, "GPU kernels for block-sparse weights," OpenAI, San Francisco, CA, USA, Tech. Rep., 2017. [Online]. Available: https://openai.com/blog/block-sparse-gpu-kernels/
[55] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2018, pp. 328–339.
[56] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 841–848.
[57] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional Tsetlin Machine," 2019, *arXiv:1905.09688*. [Online]. Available: https://arxiv.org/abs/1905.09688

GEIR THORE BERGE received the B.S. degree in nursing and the B.S. and M.S. degrees in information systems from the University of Agder, Norway, in 1997 and 2002, where he is a staff member of the Centre for Artificial Intelligence Research (CAIR) and a part time-Ph.D. Research Fellow with the Department of Information Systems. He has been with the Technology and eHealth Department, Sørlandet Hospital Trust, since 2001. Through the various roles, he has held in the healthcare domain, since 1997, he has accumulated extensive clinical and technical knowledge. He has been the Leader of the clinical systems unit, Sørlandet Hospital Trust, and also the Leader of the South-Eastern Norway Regional Health Authority EHR system administrator management board. His main interests include health informatics, clinical decision support systems, artificial intelligence, machine learning, deep learning, computational linguistics, and information structuring.

OLE-CHRISTOFFER GRANMO received the M.S. and Ph.D. degrees from the University of Oslo, Norway, in 1999 and 2004, respectively. He is currently a Professor and the Director of the Centre for Artificial Intelligence Research (CAIR), University of Agder, Norway. He develops theory and algorithms for systems that explore, experiment, and learn in complex real-world environments. His research interests include artificial intelligence, machine learning, learning automata, bandit algorithms, deep reinforcement learning, Bayesian reasoning, and computational linguistics. Within these areas of research, he has written more than 115 refereed journal and conference publications.

TOR ODDBJØRN TVEIT received the Ph.D. degree in clinical medicine from the University of Bergen, Norway, in 2013. He became a Medical Doctor (MD) at the University of Oslo, Norway, in 1992, and a Specialist in anesthesiology (anesthesia, intensive care, emergency medicine, and pain therapy), in 2000. He is currently with the Sørlandet Hospital Trust and is a staff member of the Centre for Artificial Intelligence Research (CAIR), University of Agder. He is a Specialist in anesthesiology and a Consultant with the Anesthesia and Intensive Care Department. He is also a Medical Professional Advisor with the Technology and eHealth Department, and a Research Supervisor with the Research and Development Department, Sørlandet Hospital Trust. He has more than 15 years of experience as an Advisor for the development, implementation, and use of clinical ICT-systems, EHR-solutions, and eHealth. This includes major development projects, both at the Sørlandet Hospital Trust and the South-Eastern Norway Regional Health Authority. His research interests include perioperative medicine, acute pain, clinical pharmacology, eHealth, artificial intelligence, machine learning, pattern recognition, computational linguistics, and information structuring.

MORTEN GOODWIN received the B.S. and M.S. degrees from the University of Agder, Norway, in 2003 and 2005, respectively, and the Ph.D. degree from the Department of Computer Science, Aalborg University, Denmark, in 2011, on applying machine learning algorithms on eGovernment indicators which are difficult to measure automatically. He currently holds several positions at the University of Agder, Norway. He is an Associate Professor with the Department of ICT, the Deputy Director of the Centre for Artificial Intelligence Research (CAIR), and the Coordinator of the International Master's Programme in ICT. He is also a Public Speaker and an Active Researcher. His main research interests include machine learning, swarm intelligence, deep learning, and adaptive learning in the fields of accounting, medicine, games, and chatbots. He has more than 60 peer-reviewed scientific publications and has supervised more than 110 student projects, including master's and Ph.D. theses within his topics of interest. He has also spoken at more than 90 popular science public events, mostly in the area of artificial intelligence.



LEI JIAO received the B.E. degree in telecommunication engineering from Hunan University and the M.E. degree in communication and information system from Shandong University, China, in 2005 and 2008, respectively, and the Ph.D. degree in information and communication technology from the University of Agder, Norway, in 2012, where he is currently an Associate Professor with the Department of Information and Communication Technology. His research interests include resource allocation in wireless communications, smart grid, and artificial intelligence.

BERNT VIGGO MATHEUSSEN received the Ph.D. degree from the Norwegian University of Science and Technology, Trondheim, in 2004. He is currently the Manager of Agder Energi Kraftforvaltning technology and development group, and holds a part time researcher position (20%) at the University of Agder. His research interests include hydrological modelling, hydropower scheduling and optimization, neural networks, Bayesian networks, and industrial applications of various types of artificial intelligence and machine learning techniques.