Human-Level Interpretable Learning for Aspect-Based Sentiment Analysis

Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin
Department of Information and Communication Technology
Faculty of Engineering and Science, University of Agder
4879, Grimstad, Norway
E-mails: {rohan.k.yadav, lei.jiao, ole.granmo, morten.goodwi}@uia.no

Abstract — **This paper proposes a human-interpretable learning approach for aspect-based sentiment analysis (ABSA), employing the recently introduced Tsetlin Machines (TMs). We attain interpretability by converting the intricate position-dependent textual semantics into binary form, mapping all the features into bag-of-words (BOWs). The binary-form BOWs are encoded so that the information on the aspect and context words are retained for sentiment classification. We further adopt the BOWs as input to the TM, enabling learning of aspect-based sentiment patterns in propositional logic. To evaluate interpretability and accuracy, we conducted experiments on two widely used ABSA datasets from SemEval 2014: Restaurant 14 and Laptop 14. The experiments show how each relevant feature takes part in conjunctive clauses that contain the context information for the corresponding aspect word, demonstrating human-level interpretability. At the same time, the obtained accuracy is on par with existing neural network models, reaching 78.02% on Restaurant 14 and 73.51% on Laptop 14.**

B.1 Introduction

Sentiment analysis, which identifies people's opinion on specific topics, is a classic problem in natural language processing (NLP). Under the umbrella of sentiment analysis, aspect-based sentiment analysis (ABSA), which is a fine-grained evaluation framework for sentiment classification [1], has become a hot research topic [2]. Among various tasks in ABSA, this paper focuses on the sentiment polarity (positive, neutral, negative) of a target word in given comments or reviews. For example, let us consider a review: *"Certainly not the best **sushi** in New York, however, it is always fresh and the **place** is very clean, sterile"*. The target word *"sushi"* is closely associated with its context words *"not best"*, assorting it as a negative polarity. The target word, *"place"*, is associated with its context words *"clean"* and *"sterile"*, classifying it as a positive sentiment. Such a complex form of sentiment classification is highly dependent on where the word appears in the sentence. To address this challenge, several recent approaches to ABSA have been based on attention mechanisms [3]. Although the accuracy of attention-based ABSA approaches are progressively improved, the interpretability of these models is still questionable, making them less trust-worthy. Not surprisingly, little research has been done on ABSA learning techniques that are interpretable at a human level [4].

Recently, interpretable AI has taken a big leap in industrial application [5]. Indeed, the scientific community has performed extensive research on ways to interpret neural networks. In a modern neural network, one can use the fact that the variants of attention [6] assign soft weights to the input representations, and then extract highly weighted tokens as rationales. However, these attention weights do not provide faithful explanations for classification [7, 8, 9, 10]. On the other hand, certain classic models, like Decision Trees, are particularly easy to understand, yet still compromise on accuracy compared with neural networks. Hence, an effective trade-off between accuracy and interpretability has still not been achieved.

In this article, we propose a Tsetlin Machine (TM) [11] based ABSA that employs a binary representation of the input features. The resulting architecture is *interpretable* and achieves competitive accuracy compared with state-of-the-art techniques. The ABSA task has two important inputs: a context word and an aspect word. Such aspect-based classification usually relies heavily on the position of the aspect word in the context. Such position information can be easily embedded in the neural network models. However, in TM, as all patterns and outputs are expressed in bits, learning and classification depend on bit manipulation, making it a challenging task to embed all the information into binary form. We therefore also aim to propose an extensive pre-processing approach for the ABSA inputs so that the binary form retains as much useful information as possible for the classification.

Our main contributions can be summarized as follows:

- We propose a novel pre-processing scheme to convert the ABSA inputs into binary form with limited information loss.
- We design an interpretable learning architecture using TM. The architecture offers human-level interpretable results with comparable classification accuracy.
- We employ additional knowledge from SentiWordnet [12] to enhance the accuracy of the architecture. It provides additional knowledge to the model and has significant impact on accuracy as explained later.

The remainder of the paper is organized as follows: We summarize related work in Section 2. The proposed pre-processing and TM architecture along with its learning process are described in Section 3. In Section 4, we report the experiment results and the comparisons with state-of-the-art. The interpretability of trained models is demonstrated in Section 5 before we conclude this work in Section 6.

B.2 Related Work

Sentiment analysis operates at three levels: document level, sentence level and aspect level. This work focuses on aspect level. Most of traditional supervised approaches depend heavily on handcrafted features to identify the sentiment of a word based on its context [13, 14]. However, these models fail to capture the semantic relatedness between the aspect word and its context. This problem gives rise to the attention-based models that are able to capture such a relationship [15, 6, 16, 17]. Furthermore, it is shown in [18] how an attention layer captures the weightage of the context words for predicting the sentiment of an aspect word. However, existing models cannot leverage the syntactic structure of the sentence, thereby making it difficult to distinguish various sentiments for multiple aspects of the sentence. To address this challenge, the RepWalk neural network model was recentely proposed [19]. It performs a replicated random walk on a syntax graph, effectively focusing on the descriptive contextual words.

Despite the fact that neural network-based models with attention, including BERT and contextualized embedding [6, 20, 21], capture the semantic relatedness among words in the context, they still lack interpretability. This arguably makes them black box models [22]. Many applications of attention mechanisms show, however, that a model can interpreted based on the weight assigned by the attention vector to each input, but they do not provide a faithful explanation of classification [7, 8]. Many researchers have attempted to replicate human learning behavior in neural networks [23], but have failed to answer the question of making the learning interpretable. In order to overcome the issue of interpretability in NLP, we explore the recently introduced Tsetlin Machine (TM), which recognizes patterns in the form of propositional logic [11, 24]. TMs have demonstrated promising results in various classification tasks involving numerical data, image data, text data, and board games [25, 26].

In this paper, we aim to reduce the gap between interpretability and accuracy with a significant margin on the ABSA task. To the best of our knowledge, this is the first study using TM to explore how each word in the context includes or excludes themselves to form conjunctive clauses for sentiment classification. Once the model is trained, clauses in the TM hold the information about which individual features in the context take part in the sentiment classification of the aspect word.

B.3 Methodology

B.3.1 Input Binarization

For both datasets, the ABSA tasks have a context word and an aspect word whose polarity is to be classified. Usually, the sentiment of the aspect word is reflected by its surrounding words in a

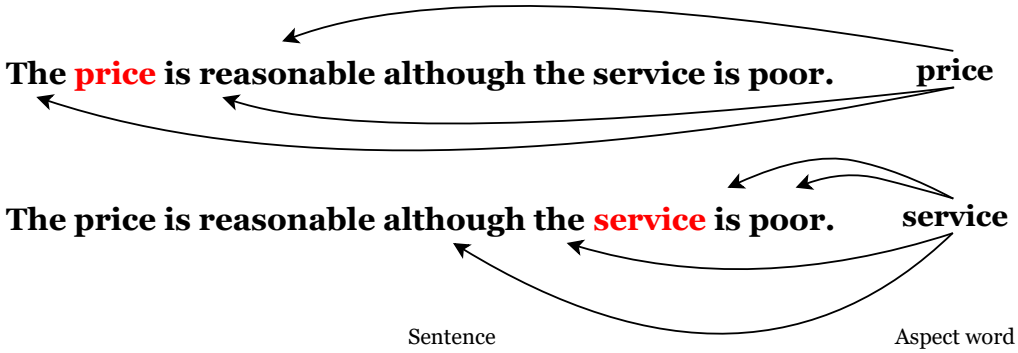

Figure B.1: Representation of an aspect word and its surrounding words.

sentence, as shown in Figure B.1. In this example, the aspect word "price" has positive sentiment due to the word "reasonable" in the context. Similarly, for "service", the context word "poor" describes its negative sentiment. This reveals that the sentiment of aspect words heavily relies on its position in a sentence and thus position embedding [27] is necessary. Such embedding creates a probability distribution of the sentence based on the aspect word. Recently, position-aware modelling has shown promising results on ABSA tasks [28].

Since TM requires binary inputs, to utilize TM for interpretability, the inputs must be binarized. It is challenging to incorporate the required position-based word relations in binary form, to allow for ABSA. In particular, since a TM does not employ any world knowledge like Word2vec [29], Elmo [21] or BERT [20], so as to retain the interpretability of the model, we reduce the size of vocabulary by replacing the sentiment carrying words with a common token. Understandably, without pre-trained embeddings, a model cannot find the similarity between two semantically related words such as "excellent" and "good". Hence, we adopt Opinion Lexicon [30], which is a list of English positive and negative sentiment words. In more details, we replace every possible word in the dataset by the common token "positive" or "negative", as shown in Figure B.2. Such external knowledge also helps to reduce the vocabulary size thereby decreasing the sparsity of BOW representations.

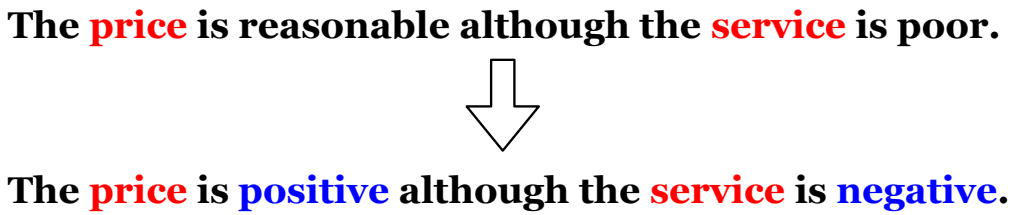

Figure B.2: Replacement of sentiment-carrying words with a common sentiment token using Opinion Lexicon.

Once the vocabulary size is determined, the context word and the aspect word can be converted into binary form, named as $BOW_{context}$ and BOW_{aspect} respectively. Since BOW in binary form does not consider the frequency of the replaced common tag (i.e., "positive" and "negative"), it becomes a rough representation of those tokens. In order to determine the location of these sentiment-carrying tokens, the sentence is split into two parts, divided by the aspect word. More specifically, we create additional binary vectors LOC^1_{vec} and LOC^2_{vec}, representing

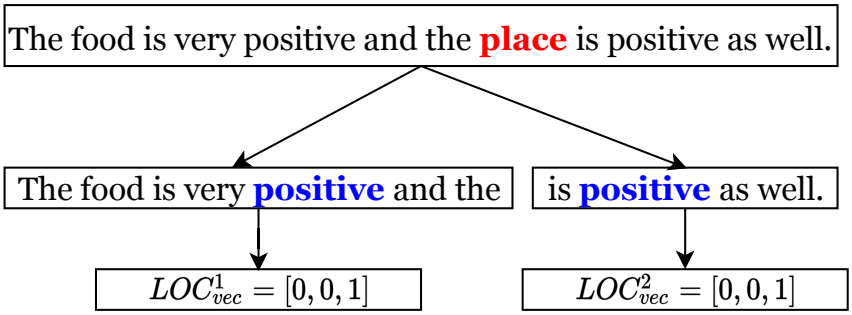

Figure B.3: 3-bit input feature representing the location of common sentiment-carrying tokens: negative, no sentiment, and positive.

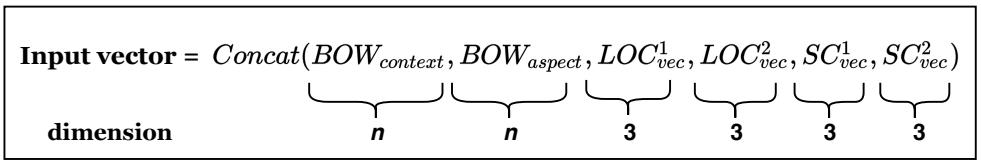

Figure B.4: Construction of binary input by concatenating all the pre-processed features.

the location of the common tokens. The dimension of LOC^1_{vec} and LOC^2_{vec} is three (the 1^{st} bit: negative, the 2^{nd} bit: no sentiment, the 3^{rd} bit: positive) as shown in Figure B.3. LOC^1_{vec} represents the presence of the common tokens "positive" or "negative" in the first part. If there are no sentiment tags, this is represented by "no sentiment". Similarly, LOC^2_{vec} represents the presence of the common tokens in the second part.

After the pre-processing of inputs, we use SentiWordNet to obtain the sentiment score (SC) of the 1^{st} part and the 2^{nd} part of the split sentence. This involvement of such additional knowledge enrich the input information. We adopt the sentiment score in a 3-D binary form for each part of the sentence. The SC vector SC^1_{vec} for the 1^{st} part of the context is given by Eq. (B.1). Similarly, vector SC^2_{vec} is utilized for the second part of the context.

$$SC^1_{vec} = \begin{cases} [0, 0, 1](positive), & \text{if } SC > 0, \\ [1, 0, 0](negative), & \text{if } SC < 0, \\ [0, 1, 0](no\ sentiment), & \text{if } SC = 0. \end{cases} \tag{B.1}$$

After processing all these binary representations, we concatenate them all to make a final input vector of size $(2n + 12)$ as shown in Figure B.4.

B.3.2 The Tsetlin Machine Based ABSA

TM is a recent classification method that manipulates expressions in propositional logic based on a team of Tsetlin Automata (TA) [11]. TA is a fixed structure deterministic automaton that learns the optimal action from a set of actions suggested by the environment. In TM, each input bit corresponds to two TAs, i.e., *TA* and *TA'*. *TA* controls the original bit of the input sample whereas *TA'* controls its negation. Here we use TA to represent a general Tsetlin automata that can be a *TA* or a *TA'*. Each TA corresponds to one literal. A literal here indicates an input bit or its negation. For example, if the bit represents the word "food", *TA* controls "food" itself and

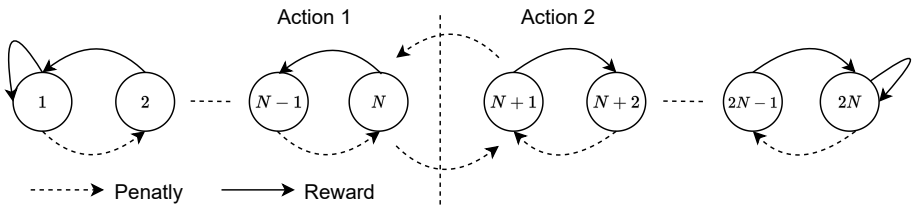

Figure B.5: The two-action TA and its transition in TM.

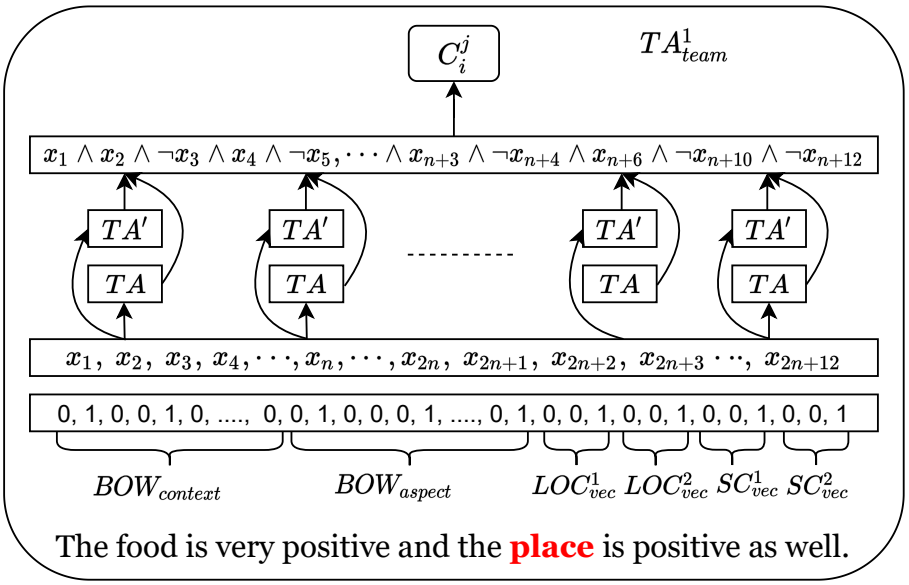

Figure B.6: TA team forms a Clause C_i^j by either including or excluding the input features.

then *TA'* handles "not food". Any TA employed by a TM has two actions with $2N$ states in total, as shown in Figure B.5. When it operates in states from 1 to N, action "exclude" is selected while action "include" is adopted for states from $N+1$ to $2N$. For each iteration, a TA performs "include" or "exclude" based on the current state. This in turn triggers a reward or penalty. If a reward is received, the TA moves to the deeper side of the action whereas if it obtains a penalty, it moves towards the center and eventually jumps to the other side of the action. Clearly, a TA, through its actions, decides whether to include or exclude its corresponding literal.

TM has a novel game theoretic strategy that regulates a decentralized team of TAs. This strategy guides the TAs to learn an arbitrarily complex propositional formula by including or excluding certain literals. More specifically, the included literals, by the operation of conjunction, formulate clauses. Each clause, after training, is expected to capture a sub-pattern. The overall pattern is decided by summing up the output of all clauses for any unknown input. The architecture for ABSA using TM is shown in Figs. B.6 and B.7.

Let us consider the input feature as a vector with a vocabulary size of n words, which is represented in BOW as $X_s = [x_1, x_2, x_3, \cdots, x_n, \cdots, x_{2n}, x_{2n+1}, x_{2n+2}, \cdots, x_{2n+12}]$ with $x_k \in \{0,1\}$ and $k \in \{1, \ldots, 2n+12\}$. Here, $[x_{2n+1}, x_{2n+2}, x_{2n+3}]$ and $[x_{2n+4}, x_{2n+5}, x_{2n+6}]$ represent LOC^1_{vec} and LOC^2_{vec} respectively. Similarly, $[x_{2n+7}, x_{2n+8}, x_{2n+9}]$ and $[x_{2n+10}, x_{2n+11}, x_{2n+12}]$ represent SC^1_{vec} and SC^2_{vec} respectively. Let q be the number of classes ($q = 3$ in ABSA task: positive, neutral and negative). If a pattern has m sub-patterns, the pattern can be captured using

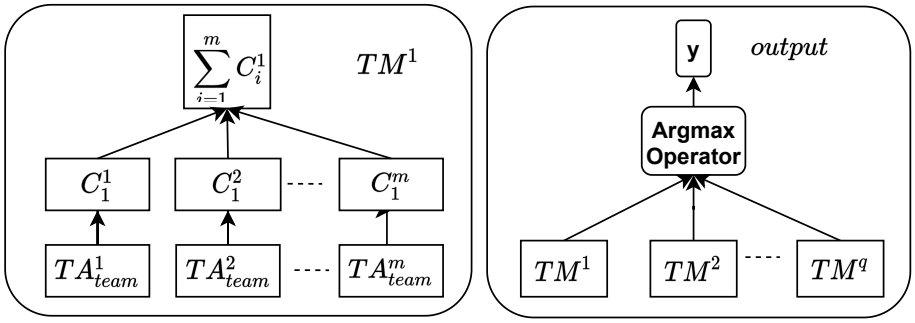

Figure B.7: (a). The sum of the votes for the clauses offers a score for a particular class. (b). Argmax operator decides the output class based on the score of the clauses in each class.

$q \times m$ conjunctive clauses C_i^j, $1 \leq j \leq q$, $1 \leq i \leq m$:

$$C_i^j = \left(\bigwedge_{k \in I_i^j} x_k \right) \wedge \left(\bigwedge_{k \in \bar{I}_i^j} \neg x_k \right), \tag{B.2}$$

where I_i^j and $\bar{I}_i^j$ are non-overlapping subsets of the input variable indices, $I_j^i, \bar{I}_j^i \subseteq \{1, \cdots, 2n+12\}, I_j^i \cap \bar{I}_j^i = \emptyset$. The subsets decide which of the input variables take part in the clause, and whether they are negated or not. The indices of input variables in I_j^i represent the literals that are included as is, while the indices of input variables in $\bar{I}_j^i$ correspond to the negated ones. Among m clauses in each class, clauses with odd indexes are assigned to positive polarity (+) whereas those with even indices are assigned to negative polarity (-). The clauses with positive polarity vote for the target class and those with the negative vote against it.

$$f^j(X_s) = \Sigma_{i=1,3,\ldots}^{m-1} C_i^j(X_s) - \Sigma_{i=2,4,\ldots}^{m} C_i^j(X_s). \tag{B.3}$$

For q number of classes, the final output y is given by the argmax operator to classify the input based on the highest sum of votes, as shown in Eq. (B.4).

$$y = \text{argmax}_j \left(f^j(X_s) \right). \tag{B.4}$$

B.3.3 The Learning Process of TM Based ABSA

In this section, we will detail the learning process of TM for the ABSA task. We explain the learning process with a walk-through of a specific sample context: "The food is very good and the place is clean as well", using the aspect word "place" whose sentiment is to be predicted. The context is first changed to "The food is very **positive** and the place is **positive** as well." For ease of explanation, we use the text word as a feature instead of the index in its binary form. For additional features, we will use the index of the binary input so as to differentiate the features that take part in classification. The indexes for additional features are LOC_{vec}^1 = $[2n+1, 2n+2, 2n+3]$, LOC_{vec}^2 = $[2n+4, 2n+5, 2n+6]$. Since the sentiment scores for both the first part of the context ("The food is very good and the") and that for the second part ("is clean as well") are greater than zero, we have SC_{vec}^1 =$[2n+7, 2n+8, 2n+9]$= [0,0,1], and SC_{vec}^2 = $[2n+10, 2n+11, 2n+12]$= [0,0,1], according to Eq. (B.1).

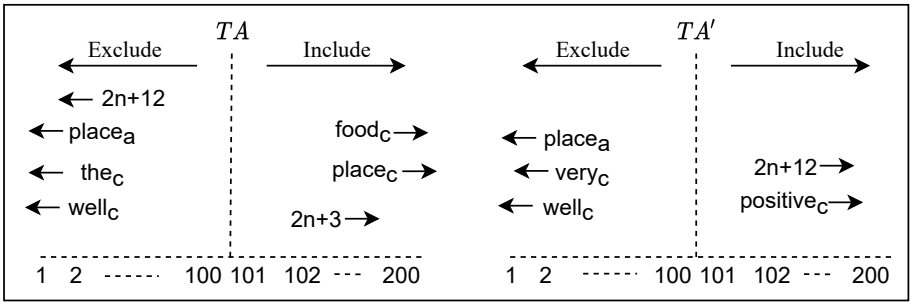

Figure B.8: TAs with 100 states per action that learn whether to exclude or include a specific word (or its negation), location of common token (or its negation) and the sentiment score information (or its negation) in a clause at time step 1.

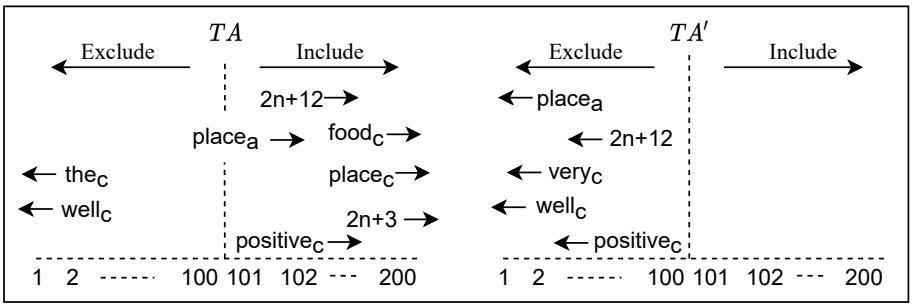

Figure B.9: TAs with 100 states per action that learn whether to exclude or include a specific word (or its negation), location of common token (or its negation) and the sentiment score information (or its negation) in a clause at time step t.

Figures B.8, B.9, and B.10 show the learning process of the ABSA task with the TM model. The subscripts c and a in the figures represent the word from context and aspect respectively. The *TA* or *TA'* that received reward will move away from the center while those that received penalty will move towards the center. In this way, the *TA* (or *TA'*) can be trained to either "include" or "exclude" a word (or its negation), helping the clauses, which are composed by the literals, learn different subpatterns. Consequently, the TM, composed by clauses, will gradually converge to the intended pattern. The feedback (reward or penalty) given to the TM follows two types: Type I and Type II feedback. Based on these feedback types, rewards or penalties are fed to the TA for the training samples. Type I Feedback is activated when a given input feature is either correctly assigned to the target sentiment (true positive) or mistakenly ignored (false negative). This feedback provides two countering effects: (1) involving more literals from the sample to refine the clauses; (2) trimming of the clauses by a factor specificity s that makes all clauses eventually evaluate to 1. The s-parameter is also responsible for avoiding overfitting. Type II Feedback is activated when an input feature is wrongly assigned to the target sentiment (false positive). It is responsible for introducing literals that make the clause evaluate to false, every time a false positive occurs. Type I Feedback and Type II Feedback are summarized in Tables B.1 and B.2 respectively.

Let us consider an example: a clause $C_1^1 = [food_c \wedge \neg positive_c \wedge place_c \wedge (2n{+}3) \wedge \neg(2n{+}12)]$ that is formed at time step $t = 1$, as shown in Figure B.8. Here, the time step indicates the instant the clause is updated during training iterations. The clause is composed by a combination

Input features	time	Clause formed and its learning in each step	Clause Output	Pred Class for true class = 1	Feedback type
the_c $food_c$ $very_c$ $positive_c$ $place_c$ $well_c$ $place_a$ $2n+3$ $2n+12$	1	$C_1^1 = food_c \wedge \neg positive_c \wedge place_c \wedge (2n+3) \wedge \neg(2n+12)$ $C_1^1 = 1 \wedge 0 \wedge 1 \wedge 1 \wedge 0$	$C_1^1 = 0$	$y = 0$	Feedback I
	2	$C_1^1 = food_c \wedge place_c \wedge (2n+3)$ $C_1^1 = 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 0$	Feedback I
	3	$C_1^1 = food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge (2n+12)$ $C_1^1 = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 1$	Feedback I
	⋮	⋮	⋮	⋮	⋮
	t	$C_1^1 = food_c \wedge positive_c \wedge place_c \wedge place_a \wedge (2n+3) \wedge (2n+12)$ $C_1^1 = 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1$	$C_1^1 = 1$	$y = 1$	Feedback I

Figure B.10: The illustration of the clause update until reaching to an intended pattern at time step t.

Input	Clause	1		0	
	Literal	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 B.1: The Type I Feedback.

of literals that are "included" by its associated TAs. At the current step, the excluded literals in this case (i.e, $2n+12, place_a, the_c, well_c$) are controlled by *TA*, and the negated literals (i.e, $place_a, very_c, well_c$) are governed by *TA'*. Clearly, this clause evaluates to 0, thereby contributing to predict class 0 despite the true class being 1, as shown in Figure B.10. This indeed triggers the Type I feedback. With Type I feedback, the reward or penalty for each literals is decided by Table B.1. Since the literal $\neg positive_c$ is included, its feature is 0 ($\neg 1$) and the clause output is 0. Therefore, it receives penalty for being included with the probability of $\frac{1}{s}$, making it slowly move towards the center and eventually jumping to the side with action "exclude". Similarly, the literal $\neg(2n+12)$ also receives the penalty with probability $\frac{1}{s}$, making it slowly moving towards the center, as well, eventually jumping to exclude action. Once this happens, the clause C_1^1 becomes $[food_c \wedge place_c \wedge (2n+3)]$ as shown in time step $t = 2$ that outputs 1, making a prediction of class 0 as depicted in Figure B.10. Table B.1 shows if the clause output is 1, the literals are of value 1, and the actions of the literals are "excluded", such literals obtain inaction or penalty with probability $\frac{1}{s}$ or $\frac{s-1}{s}$ respectively, making them slowly move towards

Input	Clause	1		0	
	Literal	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

Table B.2: The Type II Feedback.

the center and eventually jump to "include" action. Once it happens, the clause becomes C_1^1 = $[food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge (2n+12)]$ as shown in time step $t = 3$. Based on reward and penalty, TM reaches to the intended pattern at time step t by the arrangement of literals controlled by their respective TAs, as shown in Figure B.9. The final clause is given by C_1^1 = $[food_c \wedge positive_c \wedge place_c \wedge (2n+3) \wedge place_a \wedge (2n+12)]$. The clause will still obtain Type I feedback when more training samples are given and they reinforce the true positive occurrences until the sum of the votes by these clauses reaches a threshold parameter T.

The overall training and testing processes of TM-based ABSA are summarized in Algorithm 1 and Algorithm 2 respectively. For conciseness, we present, in Algorithm 1, the training procedure for the clauses with positive polarity, i.e., the clauses with odd index number. Clearly, the feedback types for the negative ones are just opposite. The complete training approach of a TM can be found in [11].

Once the class is predicted, we can explore its clauses for interpretability. The clauses that are triggered (i.e., $C_i^j(X_{s,te})$ = 1) are explored and their literals are converted into the original words for interpretation with the help of the additional information like LOC_{vec}^1, LOC_{vec}^2, SC_{vec}^1, or SC_{vec}^2.

B.4 Experiment Results

B.4.1 Datasets

The datasets are obtained from SemEval-2014 Task 4. The task has two domain-specific datasets, namely, Restaurant 14 (res14) and Laptop 14 (lap14). These datasets are provided with training and testing data. The statistics of the two datasets is shown in Table B.3. The code and the datasets are available online[1].

[1] https://github.com/rohanky/tm_absa

Algorithm 1 Training Process of TM based ABSA

Require: Given Input = [Context sentence, Aspect Word, Sentiment Score]

Pre-processed Input = Concat($BOW_{context}$, BOW_{aspect}, LOC^1_{vec}, LOC^2_{vec}, SC^1_{vec}, SC^2_{vec})
Final Input: $X_{s,tr}$ = [x_1, $\cdots$, x_n, $\cdots$, x_{2n}, $\cdots$, x_{2n+12}] and y ▷ y is the label of the input sample
Output: trained TM.
for Each training sample **do**
    $\hat{y}$ = TM($X_{s,tr}$, T, s) ▷ Current sentiment estimate for the input sample
    if $y = 1$ **then**:
        for each clause C^j_i with odd index **do**
            Use Type I Feedback($X_{s,tr}$, $\hat{y}$, T, C^j_i, s) to update all Tsetlin automata in C^j_i.
        end for
    else ▷ if $y = 0$
        for each clause C^j_i with odd index **do**
            Use Type II Feedback($X_{s,tr}$, $\hat{y}$, T, C^j_i, s) to update all Tsetlin automata C^j_i.
        end for
    end if
end for
return Trained TM.

Algorithm 2 Testing Process of TM based ABSA

Require: Given Input = $X_{s,te}$

Output: predicted class
$f^j(X_{s,te})$= 0, for all j
for all j **do** ▷ For all classes
    for all i in class j **do** ▷ For all clauses in this class
        $f^j(X_{s,te})$ = $f^j(X_{s,te})$+$(-1)^{i+1}C^j_i(X_{s,te})$
    end for
end for
return $\text{argmax}_j f^j(X_{s,te})$

B.4.2 Baselines

In our experiment, we evaluate the proposed method and compare it with related approaches for ABSA as baselines.

- **ContextAvg** averages the word embedding to form a context embedding [16].
- **LSTM** uses the last hidden vector of the LSTM for classification [31].
- **TD-LSTM** utilizes two LSTMs to learn the language model from the left and the right contexts of the aspect [16].
- **ATAE-BiLSTM** is an attention-based LSTM with Aspect Embedding model [32].

Dataset	Positive	Negative	Neutral	Total
res14 (train)	2164	807	637	3608
res14 (test)	728	196	196	1120
lap14 (train)	994	870	464	2238
lap14 (test)	341	128	169	638

Table B.3: The statistics of SemEval-2014 dataset.

- **MemNet** integrates the content and the position of the aspect word into a deep neural network [16].
- **RAM** is a multi-layer architecture where each layer consists of attention-based aggregation of word features and a GRU cell [33].
- **IAN** is an Interactive Attention Network model that calculates the attention weights of the word in its sentiment and aspect interactively [3].
- **PRET+MULT** uses two approaches of transfer knowledge from document level using pretraining and multitask training [34].
- **HCSN** proposes a Human-like Semantic Cognition network for the ABSA task, motivated by the human beings' reading cognitive process [23]. We show that performance of our proposed scheme is quite similar to this technique with high interpretability.
- **TNet** employs a CNN layer instead of attention layer to extract features from the transformed word representations originated from a bi-directional RNN layer [35].
- **AGDT** is an Aspect-Guided Deep Transition model that uses the given aspect to direct the sentence encoding from scratch with specially designed deep transition architecture. This model generates the aspect-based sentence representation and hence predicts sentiment more accurately [36].

B.4.3 Results

In our experiment, the main selling-point of the architecture is transparent learning and interpretability rather than accuracy. Better accuracy may be achieved when grid search is adopted. As we have used the integer weighted TM [37], the parameters available are the number of clauses, the threshold T, and the specificity s, which are configured as 700, 90×100, and 15 respectively for both datasets. For pre-processing of text, we substitute the short form to its full form, such as "isn't" to "is not". Additionally, we stem the words to reduce the vocabulary size created due to spelling mistakes and variants of words[2]. The remaining pre-processing procedure has already been explained before. We train the TM model on both the datasets for 100 epochs each.

Since the output sentiment label has imbalanced training samples, we use two evaluation metrics: Accuracy and Macro-F1 [38]. Following most of the related studied within the ABSA

[2]In this work, we adopt the Porter Stemmer.

Methods	Restaurant 14		Laptop 14	
	Accuracy	Macro-F1	Accuracy	Macro-F1
ContextAvg	71.5	58.0	61.5	53.9
LSTM	74.3	63.0	66.5	60.1
TD-LSTM	75.6	64.5	68.1	63.9
ATAE-BiLSTM	77.6	65.3	68.7	64.2
MemNet	76.9	66.4	68.9	62.8
RAM	78.5	68.5	72.1	68.4
IAN	78.6	NA	72.1	NA
PRET+MULT	79.1	69.7	71.2	67.5
HCSN	77.8	70.2	76.1	72.5
TNet	80.79	70.84	76.01	71.47
AGDT	78.85	NA	71.50	NA
TM based ABSA	78.02 (76.40 $\pm$ 1.0)	67.85 (64.01 $\pm$ 0.8)	73.51 (71.47 $\pm$ 0.9)	70.82 (67.48 $\pm$ 1.5)

Table B.4: Experiment results of various approaches for SemEval-2014 dataset. The upper results show the best reproducible accuracy and lower ones represent the mean and standard deviation of the last 50 epochs when running the model for five times.

task, we report the best reproducible results by running the ABSA TM for 100 epochs, as shown in Table B.4. We have reported the highest reproducible accuracy along with its mean and standard deviation obtained during 5 experiments. As we can see, Context2vec and LSTM perform quite poorly as they do not consider the aspect information when deciding the sentiment polarity. However, due to the consideration of left and right context information, TD-LSTM performs slightly better than LSTM. The variants of attention perform consistently better than LSTM and TD-LSTM. This is due to the fact that attention captures important information with regard to the aspect word. Other methods like RAM and MemNet perform slightly better because of the integrated memory in sentiment modeling. Another kind of the neural network-based model is HCSN. HCSN utilizes a human-being-like cognitive network for ABSA, which is motivated by the principles of human beings' reading cognitive processes. Its pre-reading, active reading, and post-reading technique mimics the human behavior, which is then fed to the GRU network. As interesting as it seems, the involvement of the neural network still brings this below human-level interpretation on what drives the model to make the decision. Our model, which offers a transparent view of the learning process, obtains quite similar or higher accuracy compared to HCSN and PRET+MULT techniques. However, the TNet architecture with a CNN layer, which extracts salient features from transformed word representation, achieves higher accuracy compared to TM. AGDT is a model that uses Aspect guided GRU along with Max pooling to obtain Aspect Concatenated Embedding. It obtains quite similar accuracy compared to TM on Restaurant 14, whereas accuracy is lower on Laptop 14. Note that we do not use any pre-trained word2vec or glove embedding for TM and our model still performs better than LSTM, TD-LSTM as well as attention based BilSTM for both datasets. The Macro-F1 score shows that TM does not only greedily learn a particular class but also creates a set of features for each and every class. Even though the performance of our proposed model does not outperform

the state-of-the-arts models, it reaches to comparable accuracy and Macro-F1 with transparent learning and interpretable prediction.

In addition to the above comparisons, we demonstrate here the necessity of including both $LOCs$ and SCs vectors. First we only used the $LOCs$ in the model and observed that the accuracy of the model reaches 76.51%. Secondly, we replaced $LOCs$ with SCs and the performance of the model decreased to 75%. This shows that both vectors add useful information when employed together thereby reaching the stated accuracy of 78.02%.

To compare the performance of TM with classical interpretable models such as Logistic Regression (LR), we use our preprocessed BOW as input to LR. We observed that the TM performs better than LR in terms of accuracy. LR obtains the accuracy of 75.38% as compared with TM's 78.02% on the Restaurant 14 dataset. Indeed, those two approaches operate based on different concepts. LR is trained by adjusting weights and bias. TMs, on the other hand, relates words using propositional logic to represent a class. Employing propositional logic for knowledge representation provides rules rather than a mathematical computation. This crucial difference between a rule-based approach and regression methods is explored in [39]. One can analyze why a LR model assigns a particular class to an input by inspecting the weights and bias. However, assigning them meanings requires understanding of the mathematical computation that LR carries out. Since TM creates a list of patterns for a particular class based on the interaction of aspect words and the sentiment words in the context, its conjunctive clauses hold information of words in a rule-based form. It is well-known that evaluating a conjunctive clause is particularly easy for humans, making them natively interpretable and easier to explain than LR.

B.5 Interpretability and Analysis

B.5.1 Characteristics of Clauses

In this section, we will explain one phenomenon of a TM after training with the datasets. When analyzing the clauses after training, we noticed that the TM employs more negated literals to form a clause. This is a bit counter intuitive as there should be, intuitively, more literals in their original forms than their negations in a clause. To explain this behavior, let us study two general sentences having positive sentiments for aspect word "laptop":

- This laptop is in excellent condition.
- Battery life of this laptop is better compared to other brands.

In this example, we assume a vocabulary containing both positive and negative sentiment words of size 6, V = [excellent, bad, condition, worst, costly, better, laptop]. When literals in their original forms are utilized to compose a clause, the two sentences require two clauses to follow the sentiment, i.e., $C_1 = [laptop \wedge excellent \wedge condition]$ and $C_2 = [battery \wedge better \wedge laptop \wedge others]$. On the contrary, when negated form of literals are employed, only one clause is sufficient to satisfy the sentiment in both sentences: $C_1 = [laptop \wedge \neg bad \wedge \neg worst \wedge \neg costly]$. As the negation is a more efficient way to represent a pattern in NLP, the trained TM employs naturally more negated literals to form a clause.

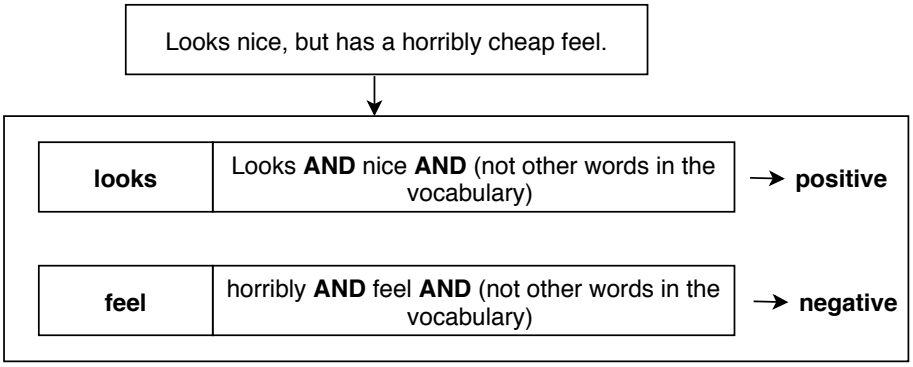

Figure B.11: Interpretation of a randomly selected sample from ABSA task.

B.5.2 A Case Study for Interpretability

In this case study, we demonstrate the interpretable result from trained model. We randomly select a sentence from the dataset as an example and demonstrate its literals that are responsible to form the clause. The selected sentence is "Looks nice, but has a horribly cheap feel." with a aspect word "looks" whose sentiment prediction of TM is positive. The sentence after pre-processing becomes "Looks positive, but has a negative negative feel." Among various clauses that are triggered by the given input, we randomly select a clause for interpretation. The clause is given by:

- $C_i^j = positive \wedge (2n+6) \wedge \neg(words\ not\ in\ the\ sentence\ and\ aspect)$.

The above clause can be interpreted as: the aspect word "looks" has positive sentiment because it has words "positive" and it lies in the second part of the sentence (indicated by $2n+6$, i.e., $LOC_{vec}^2 = [0, 0, 1]$) when split from aspect word "looks". Similarly, if the aspect word in the sentence is "feel" then its sentiment is predicted to be negative and a randomly selected clause is:

- $C_i^j = negative \wedge (2n+1) \wedge \neg(words\ not\ in\ the\ sentence\ and\ aspect)$.

This clause means that the sentiment is negative because it has words like "negative" and it lies in the first part of the sentence (indicated by $2n+1$, i.e $LOC_{vec}^1 = [1, 0, 0]$) when split from aspect word "feel". In both the cases, $\neg(words\ not\ in\ the\ sentence\ and\ aspect)$ represents the words in negated form that are presented in the input features. Finally, reversing back all the information and binarization to the original form of the words, we can obtain interpretation that shows the influence of words in the classification as in Figure B.11.

B.6 Conclusions

In this paper, we aim to reduce the gap between the interpretability and the accuracy of aspect based sentiment analysis (ABSA) by employing the recently introduced Tsetlin Machine (TM). Our proposed model embeds the aspect-based inputs into binary form for classifying the sentiment of a particular word in a sentence. Such binary representations are then fed to a TM architecture where the learning process is transparent, which gives a clear picture of what actually drives the TM to learn the particular sentiment for a given input. Additionally, we show the

involvement of words carrying the sentiment for the aspect words in the case study. In short, the proposed model successfully provides an human-interpretable learning approach on ABSA task with comparable accuracy.

B

Bibliography

[1] L. Zhang and B. Liu, *Sentiment Analysis and Opinion Mining*, pp. 1152–1161. Boston, MA: Springer US, 2017.

[2] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 27–35, ACL, 2014.

[3] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," in *IJCAI*, (Melbourne, Australia), pp. 4068–4074, 2017.

[4] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: A comparative review," *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019.

[5] W. Samek, G. Montavon, A. Vedaldi, L. Hansen, and K. Müller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, (California, USA), 2015.

[7] S. Serrano and N. A. Smith, "Is attention interpretable?," in *ACL*, (Florence, Italy), pp. 2931–2951, ACL, 2019.

[8] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 11–20, ACL, 2019.

[9] G. Brunner, Y. Liu, D. Pascual, O. Richter, M. Ciaramita, and R. Wattenhofer, "On identifiability in transformers," in *ICLR*, (Addis Ababa, Ethiopia), 2020.

[10] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across NLP tasks," *arXiv*, vol. 1909.11218, 2019.

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

[12] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *LREC*, (Genoa - Italy), 2006.

[13] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent Twitter sentiment classification," in *ACL*, (Portland, Oregon, USA), pp. 151–160, ACL, 2011.

[14] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, "NRC-Canada-2014: Detecting aspects and sentiment in customer reviews," in *International Workshop on Semantic Evaluation (SemEval 2014)*, (Dublin, Ireland), pp. 437–442, ACL, 2014.

[15] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *AAAI*, (New Orleans, USA), 2018.

[16] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *EMNLP*, (Austin, Texas), pp. 214–224, ACL, 2016.

[17] J. Liu and Y. Zhang, "Attention modeling for targeted sentiment," in *EACL*, (Valencia, Spain), pp. 572–577, ACL, 2017.

[18] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMs for target-dependent sentiment classification," in *COLING*, (Osaka, Japan), pp. 3298–3307, ACL, 2016.

[19] Y. Zheng, R. Zhang, S. Mensah, and Y. yi Mao, "Replicate, walk, and stop on syntax: An effective neural network model for aspect-level sentiment classification," in *AAAI*, (New York, USA), 2020.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, (Minneapolis, Minnesota), pp. 4171–4186, ACL, 2019.

[21] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL*, (New Orleans, Louisiana), pp. 2227–2237, ACL, 2018.

[22] C. Rudin, ""'stop explaining black box machine learning models for high stakes decisions and use interpretable models instead'"," *Nature Machine Intelligence*, vol. 1, pp. 206–215, 2018.

[23] Z. Lei, Y. Yang, M. Yang, W. Zhao, J. Guo, and Y. Liu, "A human-like semantic cognition network for aspect-level sentiment classification," in *AAAI*, (Hawaii, USA), 2019.

[24] X. Zhang, L. Jiao, O.-C. Granmo, and M. Goodwin, "On the convergence of tsetlin machines for the identity- and not operators," *arXiv*, vol. 2007.14268, 2020.

[25] O.-C. Granmo, S. Glimsdal, L. Jiao, M. Goodwin, C. W. Omlin, and G. T. Berge, "The convolutional tsetlin machine," *arXiv*, vol. 1905.09688, 2019.

[26] 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. 115134–115146, 2019.

[27] S. Gu, L. Zhang, Y. Hou, and Y. Song, "A position-aware bidirectional attention network for aspect-level sentiment analysis," in *COLING*, (Santa Fe, New Mexico, USA), pp. 774–784, ACL, 2018.

[28] J. Zhou, Q. Chen, X. Huang, Q. Hu, and L. He, "Position-aware hierarchical transfer model for aspect-level sentiment classification," *Information Sciences*, vol. 513, pp. 1–16, 2020.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, (Stateline, United States), Curran Associates, Inc., 2013.

[30] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *ACM SIGKDD*, (New York, NY, USA), p. 168–177, ACM, 2004.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[32] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *EMNLP*, (Austin, Texas), pp. 606–615, ACL, 2016.

[33] P. Chen, Z. Sun, L. Bing, and W. Yang, "Recurrent attention network on memory for aspect sentiment analysis," in *EMNLP*, (Copenhagen, Denmark), pp. 452–461, ACL, 2017.

[34] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Exploiting document knowledge for aspect-level sentiment classification," in *ACL*, (Melbourne, Australia), pp. 579–585, ACL, 2018.

[35] X. Li, L. Bing, W. Lam, and B. Shi, "Transformation networks for target-oriented sentiment classification," in *ACL*, (Melbourne, Australia), pp. 946–956, ACL, 2018.

[36] Y. Liang, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "A novel aspect-guided deep transition model for aspect based sentiment analysis," in *EMNLP-IJCNLP*, (Hong Kong, China), pp. 5569–5580, ACL, 2019.

[37] K. D. Abeyrathna, O.-C. Granmo, and M. Goodwin, "Extending the tsetlin machine with integer-weighted clauses for increased interpretability," *arXiv*, vol. 2005.05131, 2020.

[38] C. D. Manning and H. Schütze, "Foundations of statistical natural language processing," in *SGMD*, 2002.

[39] M. Haghighi, S. Johnson, X. Qian, K. Lynch, K. Vehik, and a. T. S. G. S. Huang, "A comparison of rule-based analysis with regression methods in understanding the risk factors for study withdrawal in a pediatric study," *Scientific Reports*, vol. 6, 2016.