Myanmar Spelling Error Classification: An Empirical Study of Tsetlin Machine Techniques

Ei Thandar Phyu, Ye Kyaw Thu, Thazin Myint Oo, Hutchatai Chanlekha, and Thepchai Supnithi

***Abstract*— Accurate spelling and grammar checking is fundamental to the development of language tools for Myanmar language. Classifying spelling error types is crucial in spell checkers and other language processing tools because it enables more accurate and context-aware error corrections. This process categorizes spelling errors in written text into distinct types or categories. To address the lack of such resources for Myanmar language, we have developed a spelling corpus containing misspelled words alongside their corrected forms in a parallel structure, paired with a corpus categorizing types of spelling errors. This paper focuses on an observational study of Tsetlin Machine for Myanmar spelling error type classification, involving comprehensive parameter tuning and a performance comparison with fastText, a state-of-the-art natural language processing model. Our studies indicate that while Tsetlin Machine achieves comparable results to fastText specifically in the domain of phonetic error classification, it demonstrates lower efficacy in other error classes.**

***Index Terms*—Spelling error type classification, Spell checker, Tsetlin Machine, fastText, Natural Language Processing, Myanmar Language.**

I. Introduction

THE number of users on Facebook, X, Instagram, and other social media sites has significantly increased recently. As of January 2023, around 15 million people in Myanmar were active on social media, making up about 27.6% of the total population. Valuable knowledge and content, including users' opinions and reviews, are typically found in social media text. However, when developing natural language processing (NLP) models for analyzing or mining such knowledge, we commonly encounter challenges arising from misspelling. Social text and conversations often include many spelling errors, informal language, and lack of professionalism which can cause miscommunication and reduced credibility in the case of academic or business writing. They also introduce the challenges to NLP techniques in terms of accuracy and processing time. Spell checkers have been used to ensure the quality of written content and improve the quality of user-generated text.

Through the analysis of Myanmar spelling errors that are typically found on social media platforms, we defined the error patterns into ten categories. We found out that most common error types are phonetic errors (occurrence of 38.73%) and typographic errors (occurrence of 35.14%) followed by the combination of phonetic and typographic error (occurrence of 7.61%) and consonant error (occurrence of 6.12%). Other types are sequence error, slang error, stack word error, sensitive word error, short-form error, encoding error and dialect error. To handle these error patterns which can involve both real-word or non-word errors, it is important to understand the type of spelling errors to improve the NLP applications such as spell checker, automated proofreading and so on.

Spelling error type classification is the process of categorizing the misspelled words into specific types, each of which indicate a different nature of mistake. By detecting and categorizing the misspelled words into distinct types, specific spelling correction algorithms can be employed for each error category to suggest or automatically correct the misspelled word. This study on spelling error classification allows us to systematically identify and understand the patterns and variations in the spelling mistakes. This research aims to distinguish between common errors, and gain insights into the specific types of spelling errors present in our developing error-correction parallel corpus.

Tsetlin Machine (TM) [1] represents a pattern recognition methodology that prioritizes interpretability in the field of machine learning (ML). The transparent decision-making process inherent in TM facilitates a clear understanding of the rationale behind specific classifications, rendering them a promising avenue for the development of interpretable AI. This quality aligns with the growing demand for transparency and explainability in artificial intelligence systems. In this paper, we explored the properties of TM in the field of NLP, specifically focusing on their efficacy in the context of Myanmar spelling error type classification.

The Myanmar spelling error-correction parallel corpus, collected by Ei Phyu Phyu Mon et al. [2] [3], is extended for the purpose of developing a large-scale parallel Myanmar spelling error-correction corpus. The spelling error type corpus is created through the extraction of

Ei Thandar Phyu is with the Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand. email: eithandar.p@ku.th

Ye Kyaw Thu is with the National Electronics and Computer Technology Center, Pathum Thani, Thailand, and also with the Language Understanding Lab, Myanmar. email: yekyaw.thu@nectec.or.th

Thanzin Myint Oo is with the Language Understanding Lab., Myanmar. email: queenofthazin@gmail.com

Hutchatai Chanlekha is with the Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Bangkok, Thailand. email: fenghtc@ku.ac.th

Thepchai Supnithi is with the National Electronics and Computer Technology Center, Pathum Thani, Thailand. email: thepchai.supnithi@nectec.or.th

the error types from developing spelling error-correction parallel corpus. We presented the use of TM for spelling error type classification and compared its performance to fastText [4] library, which is widely recognized for its effectiveness in various language-related tasks. fastText is a reliable reference model, facilitating a meaningful evaluation of TM's performance in the context of spelling error type classification. By contrasting the performance of these two methods, this research aims to analyze the respective strengths and weaknesses of TM with the goal of identifying areas for improvement and advancing the understanding of its applicability in Myanmar spelling error type classification tasks.

II. Related work

Advancements in NLP research for the Myanmar language have been notable. Numerous scholarly papers and projects have surfaced, playing a crucial role in shaping new datasets and language processing tools specifically designed for Myanmar. These efforts have concentrated on diverse aspects, including Myanmar XNLI [5], machine translation [6], and tokenization [7]. Additionally, hate speech detection has gained prominence in Myanmar, prompting researchers to create annotated datasets and explore ML algorithms [8]. Despite these strides, challenges remain—chiefly limited data resources and computational costs—as NLP continues to evolve for the Myanmar language.

Transformer-based models and Large Language Models (LLM) represent the cutting edge in NLP techniques, finding widespread application across various tasks—including classification. These models, such as BERT [9], GPT-3 [10], and RoBERTa [11], leverage attention mechanisms to capture complex relationships in data, making them highly effective for tasks like text classification, sentiment analysis, and more. fastText [4] is a popular classifier that uses shallow neural networks to provide fast and efficient text classification by representing words as vectors and capturing their context in a given text. In recent years, TM [1] has gained significant interest in the field of ML, particularly for its simplicity, interpretability, and efficiency in classification tasks. Transformer-based models excel in capturing complex patterns where large amounts of data and computational resources are available. On the other hand, TMs offer a lightweight alternative with clear interpretability and lower resource requirements, making them suitable for environments where computational efficiency and model transparency are prioritized. There has been research done in terms of exploring TM capabilities. Rupsa Saha [12] proposed using TMs for Sentiment Analysis and Semantic Relation Categorization, showing that the patterns TMs find match expert-verified rules or lexicons while keeping the results easy to understand without losing accuracy compared to other ML techniques. Xuanyu Zhang [13] introduced a TM-based method for Chinese sentiment analysis and spam review detection that ensures a transparent and easily understandable learning process. The results indicate that this method outperformed complex, non-transparent deep-learning models like BERT in terms of accuracy and F1 scores.

The task of spelling checking for Myanmar language remains particularly challenging, with development still in its nascent stages. Existing published works on Myanmar spelling correction typically involve two-stage systems that rely on dictionary or rule-based approaches. Aye Myat Mon et al. [14] introduced a Myanmar spell checker utilizing a dictionary-based approach, which included a compound misused word detection algorithm and a bigram model designed to identify phonetic errors. Bayesian Classifier was employed for contextual errors, while typographical errors were addressed using a corpus lookup approach with a syllable dictionary.

Ei Phyu Phyu Mon et al. [2] explored an automatic rule extraction approach for Myanmar spelling checking. The tool 'wdiff' was utilized to compare two texts, and the resulting patterns were converted into five error-correction patterns. A total of 38,124 spelling correction rules were extracted to address the errors. Nevertheless, this system exhibited limitations in correcting phonetic and typographic errors, and out-of-vocabulary issues remained a concern.

In further research, Ei Phyu Phyu Mon et al. [3] assessed the efficacy of spelling checking by applying the Symmetric Delete Spelling Correction Algorithm (SymSpell). SymSpell [15] [16] supports `Lookup()` for single word spelling correction, `LookupCompound()` for compound aware multi-word correction and `WordSegmentation()` that integrates both word segmentation and spelling correction. `LookupCompound()` method enables compound-aware automatic spelling correction for multi-word input strings, supporting compound splitting and decompounding by addressing various cases of space-related errors within correct words. The findings suggested that the `LookupCompound()` method was not only faster but also provided better correction quality than the `WordSegmentation()` method. While this spell checker effectively addressed the combination of phonetic and typographic errors as well as typographical errors alone, it struggled with dialect, encoding, short-form, and slang word errors.

III. Methodology

A. Tsetlin Machine

Tsetlin Machine is a general-purpose, interpretable, and energy-efficient ML approach designed for classification and decision-making processes. Introduced by Ole-Christofer Granmo in 2018 [1], TM is notable for its reliable performance coupled with human-level interpretability, striking a balance between effectiveness and interpretability. It addresses complex pattern recognition challenges by employing propositional logic formulas, which are constructed by a collective of Tsetlin Automata. A Tsetlin Automaton (TA) is a finite-state automaton [17], where the current state determines the action taken, and rewards or penalties drive state transitions to reinforce successful actions.

Boolean feature inputs are used to take advantages such as efficient storage, programming language compatibility and ease of understanding for humans. TM breaks down problems into self-contained patterns, each of which is represented as conjunctive clauses in propositional logic. A conjunctive clause can be considered as "if-then" rules and is formed by using and-operator to combine the literals denoted as l. These rules specify patterns in the binary input data that are indicative of different types of spelling errors. Input feature vector $X = (x_1, \dots, x_i)$ of Boolean features, where i is the dimension of features, is fed to n multiple clauses. The feature vector X and their negated counterparts form a literal set $L = \{x_1, \dots, x_i, \bar{x}_1, \dots, \bar{x}_i\}$. Half of the clauses in TM are given a positive polarity, indicated by the upper index 1: C_j^1 and the remaining half is given a negative polarity with the upper index 0: C_j^0 where $j \in 1, 2, \dots, \frac{n}{2}$. The clauses learned sub-patterns in X as in Equation 1

$$C_j(X) = \bigwedge_{l_k \in L_j} l_k \tag{1}$$

The final output is computed through summation and unit step thresholding, $u(v)$ is 1 when $v \geq 0$ otherwise $u(v)$ is assigned to 0.

$$\hat{y} = u(\sum_{j=1}^{n/2} C_j^1(X) - \sum_{j=1}^{n/2} C_j^0(X)) \tag{2}$$

For example, considering XOR relation, $\hat{y}$ will be $u(x_1\bar{x_2} + \bar{x_1}x_2 - x_1x_2 - \bar{x_1}\bar{x_2})$.

In the case of multiclass TM, the threshold function of each output $y, i \in 1, 2, \dots, m$, is replaced by the argmax operator as in Equation 3

$$y = \text{argmax}_{i=1,2,\dots,m}(\sum_{j=1}^{n/2} C_j^{1,i}(X) - \sum_{j=1}^{n/2} C_j^{0,i}(X)) \tag{3}$$

TM inference structure is depicted in Figure 1. A clause consists of a team of Tsetlin Automata, where each Tsetlin Automaton determines whether to Include or Exclude a specific literal in the clause based on reinforcement: Type I feedback for frequent pattern generation and Type II feedback for enhancing discrimination power within the patterns. Type I feedback emphasizes the reinforcement of the Include actions over Exclude actions. When TM processes an input X and output y, the Type I feedback is provided to the clauses C_j^w when $y = w$. In the Type I feedback, the likelihood of obtaining a reward is $\frac{s-1}{s}$. The likelihood of taking no action is $\frac{1}{s}$ and the probability of incurring a penalty is 0. The hyper-parameter s (where $s \geq 1$) controls how strongly we favour Include actions. A higher s value stimulates Tsetlin Automata to incline towards including literals more strongly in their clauses.

Type II feedback is applied to the clauses C_j^w when $y \neq w$. Type II feedback only triggers when $y = 0$ for positive polarity clauses and $y = 1$ for negative polarity clauses. This feedback is strong and generates candidate literals that contribute to distinguishing between $y = 0$ and $y = 1$. Both Type I Feedback and Type II Feedback interact to minimize the expected output error, thereby converging towards a global optimum. The probability of reinforcing a clause gradually decreases to zero as the sum of clause output, $\sum_{j=1}^{n/2} C_j^1(X) - \sum_{j=1}^{n/2} C_j^0(X)$, approaches a user-defined target T. When the voting sum equals or surpasses the value of T indicating successful recognition of the input by TM, no further reinforcement of clauses occurs.

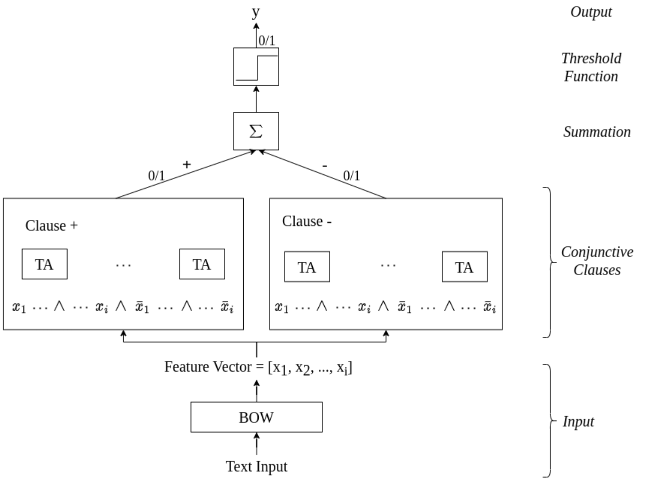

y
0/1
Output
Threshold Function
Σ
Summation
0/1
+
-
0/1
Clause +
TA
...
TA
$x_1 \cdots \wedge \cdots x_i \wedge \bar{x}_1 \cdots \wedge \cdots \bar{x}_i$
Clause -
TA
...
TA
$x_1 \cdots \wedge \cdots x_i \wedge \bar{x}_1 \cdots \wedge \cdots \bar{x}_i$
Conjunctive Clauses
Feature Vector = [x1, x2, ..., xi]
BOW
Text Input
Input

Fig. 1: TM Inference Structure

B. fastText

fastText, developed by Facebook's AI Research (FAIR) team, is an efficient NLP library designed for text classification and representation. fastText embeddings [18] utilize character n-grams to build word representations, enabling words to be expressed as the sum of their n-gram vectors. This approach extends the word2vec model by incorporating subword information and employs a skip-gram model during training to learn these embeddings. It also allows for the computation of word representations for out-of-vocabulary (OOV) words, due to its ability to train rapidly on large-scale corpora. A word is represented as a bag of character n-gram by adding special boundary symbols ⟨ and ⟩ at the beginning and end of the word. For example; if the word is "မြို့" ("City" in English) and $n = 3$, the character trigrams will be <မိ, မြိ, ြို, ို့, ု့> and the special sequence will be <မြို့>.

The foundational framework of fastText depends on the implementation of the continuous bag of words (CBOW) model, and a hierarchical classifier to optimize training efficiency. fastText uses a hierarchical classifier, organizing a large number of classes into a tree structure [4]. The n-gram features are embedded and averaged to get the hidden variable. To enhance the processing speed, a hierarchical softmax layer [19] is constructed based on the Huffman coding tree. The softmax activation function is used to calculate the probability distribution over the predefined classes. The probability of a given text with a particular class is examined through a depth-first search along the nodes. Therefore, classes assigned by low probability can be eliminated.

IV. Spelling Errors of Myanmar Language

The Myanmar language is tonal and comprises 33 consonants, 12 basic vowels, 4 medials, and 7 independent vowels. Each sound is represented as a syllable, which is a combination of consonants, vowels, and medials. For example, the word “စာ” /sà/ (“Letter” in English) is formed by combining the consonant ‘စ’ and the vowel ‘ာ’.

In the Myanmar language, a consonant shares a similar phonetic sound with another consonant, such as ‘စ’ /sa/ and ‘ဆ’ /s^{h}a/. Homophone errors can also occur when words share a similar or identical pronunciation but have different meanings and spellings. These errors can lead to confusion in both spoken and written communication. As an example, these four syllable ‘စီး’ /sí/ (“Ride, Flow” in English) , ‘ဆီး’ /s^{h}í/ (“Urine” in English), ‘စည်း’ /sí/ (“Line” in English), ‘ဆည်း’ /s^{h}í/ (“Save” in English) have similar pronunciation but their meaning and usage within word diverge.

Another common spelling error is caused by virama ‘်’, which removes the inherent vowel, resulting in a syllable-final consonant. This typically occurs with the consonants ‘က’, ‘င’, ‘စ’, ‘ည’, ‘ဉ’, ‘တ’, ‘န’, ‘ပ’, ‘မ’, ‘ယ’ and ‘ဝ’. The users usually confuse and make a mistake since some combinations possess the similar phonemic characteristics. For instance, လန်း and လမ်း are pronounced identically as /láɴ/.

In the context of the Myanmar language, the pronunciation of words doesn’t always correspond directly to their spelling. For instance, the word “အရုဏ်တက်” (“Dawn” in English) can be pronounced as /ʔə jòʊɴ dɛʔ/ based on its spelling. However, the word is pronounced as /ʔà jòʊɴ dɛʔ/ by adding the vowel sound ‘အာ’ /à/. The Myanmar language has a large number of homonyms or pairs of words that have the same phonetic or pronunciation which can lead to the phonetic errors.

In Myanmar language, there are some consonants and vowels that have a similar appearance such as ‘ဉ’ (consonant) and ‘ဥ’ (independent vowels), ‘ဝ’ (consonant) and ‘၀’ (number zero), ‘ရ’ (consonant) and ‘၇’ (number seven). The users also confuse between these consonants and independent vowels: ‘ဤ’, ‘ဣ’, ‘ဍ’, ‘ဥ’, ‘ဎ’ and ‘ဌ’. Additionally, users sometimes make errors by combining letters, vowels, and symbols to create a shape that looks like a certain letter. For example, typing (သ + ြ → သြ) instead of ‘ဩ’ where the users are required to type only U+1029 rather than typing U+101E U+103C. The users may also write a single letter instead of typing the correct combination. For instance, the users sometimes write ‘ဍ’ in place of (င + ှ → ငှ) and vice versa.

Myanmar language also has the double stack words that use conjunct consonants which are typed with two consonants with the symbol ‘္’ in the middle such as (က+္+က → က္က). Some double-stacked words is created by placing a superscript ‘င်္’, called Kinzi, above the consonant (e.g., ‘တနင်္ဂနွေ’ which means “Sunday” in English). The stack word errors commonly stem from incorrect arrangements of consonants, such as placing two consonants vertically or omitting a necessary lower consonant. For example, the user types ‘ပ္မ’ instead of typing ‘မ္ပ’ and writes ‘ပစည်း’ instead of the correct word ‘ပစ္စည်း’. The Myanmar Sign Virama (‘္’) is essential for accurate rendering of stacked words; its omission can result stack word errors (e.g., writing ‘ဒုကခ’ instead of ‘ဒုက္ခ’). Another reason of the stack word error is incorrect usage of ‘်’ instead of ‘င်္’ (e.g., typing ‘အဂ်လိပ်’ instead of ‘အင်္ဂလိပ်’).

Some encoding mistakes can be encountered during the process of converting Zawgyi to Unicode. Changing the keyboard and altering the order of typing can also impact the user, leading to errors in sequence or typography. The invisible Unicode characters such as ‘zero width space’, ‘zero width non-joiner’ or ‘zero width joiner’ can be contained causing misspelled errors (e.g, “ညံ့”). Users often utilize abbreviations, and specialized slang on social media, which may deviate from the spelling rules and result in instances of misspelling.

A. Myanmar Spelling Error Types

By studying and analyzing the Myanmar spelling errors, ten different error types are proposed [2] [3]. In our study, we categorize spelling errors into a single level of categories, intentionally avoiding hierarchical relationships or additional subdivisions based on aspects such as orthography level, dictionary level, semantic level, or social context. Moreover, the analysis emphasizes errors that impact entire words, assessing whether the chosen word is contextually appropriate or if another word would be more suitable.

1) Consonant error (con)

Consonant error is an error when writing or typing the wrong consonants which have the similar shape. It may occur when the user does not know the correct spelling or the user cannot find some consonant on the keyboard layout leading to typing the similar character.

- E.g., ပြဿနာ (“Problem” in English) → ပြသနာ, ပြသာနာ.

2) Dialect error (dialect)

Dialect error can occur from the influence of particularly spoken varieties specific to certain administrative divisions or ethnic groups of Myanmar. Often, these varieties exhibit small discrepancies in pronunciation and vocabulary.

- E.g., ပိုက်ဆံအိတ် (“Wallet” in English) → ပတ်တာအိတ်.

3) Encoding error (encode)

Encoding error is a flaw that appears during the encoding process of text data, resulting in the generation of characters that are not Myanmar consonants or medial. This type of error is an unintended mistake and leads to the generation of words that do not exist in the Myanmar dictionary.

- E.g., သင်္ချာ (“Math” in English) → သချာ.

4) Phonetic error (pho)

Phonetic errors occur when words are written based on their pronunciation rather than following the conventional spelling rules. These errors have the potential to generate real word mistakes, introducing ambiguity to the overall content or meaning of the sentence.

- E.g., ပတ်သက် (“Concerned with” in English) → ပတ်သတ်, ပက်သက်, ပက်သတ်.

5) Sensitive error (sensitive)

Sensitive errors are deliberate mistakes that happen when certain Myanmar characters are substituted with English or special characters to avoid explicitly using words that violate social community standards, such as curse words and hate speech.

- E.g., စောက် (abusive word) → $, S, s, သောက်, ဆောက်.

6) Sequence error (seq)

Sequence error may come out when the writer types the wrong typing sequence of Myanmar syllables, which are composed of consonants, medials, and vowels. For the word “ပြုံး” (“Smile” in English) with the correct order (ပ + ◌ြ + ◌ံ + ◌ု + ◌း), misspelled word “◌ြပုံး” can be occurred if the user types (◌ြ + ပ + ◌ံ + ◌ု + ◌း).

- E.g., အံ့သြ (“Amazing” in English) → အ့◌ံသြ.

7) Short-form error (short)

Short-form errors occur when people use unofficial abbreviations or write English characters which have the similar pronunciation of Myanmar syllable. As an example, the users write “zကား” rather than “ဇာတ်ကား” /zaʔ ka̰/ (“Movie” in English). When the word consist of two same syllables consecutively, social media users omit the second syllable and write “၂” (“2” in English), “2”, or “double quotation mark” in place of second syllable.

- E.g., ပြီးပြီ (“Finish” in English) → pp, p◌ီးp.

8) Slang word error (slang)

Slang word errors consist of neologisms that differ from the standard vocabulary. The widespread use of neologisms and slang words on social media can lead to language standardization degradation and can result in potential misunderstandings and communication problems.

- E.g., ကိုကြီး (“Brother” in English) → ကွီး.

9) Stack word error (stack)

Stack word error occurs when the writer misuses the double stack words. The causes of the stack word error are writing two consonants up and down wrongly, misusing one of the consonants, omitting the lower consonants, forgetting to type the symbol ‘◌္’ (Myanmar Sign Virama) and using ‘◌်’ instead of ‘◌င်္’.

- E.g., မင်္ဂလာပါ (“Hello” in English) → မဂ်လာပါ.

10) Typographic error (typo)

Typographical errors are unintentional and arise from simple mistakes such as pressing the wrong key, skipping a character or inserting an unintended character. When the user is writing in multiple languages, accidentally switching the keyboards can cause the chance of typo errors.

- E.g., ကျေဇူတင်ပါတယ် (“Thank you” in English)

V. Experiments

A. Myanmar Spelling Error Type Corpus

A well-designed and large-scale corpus is crucial for the Myanmar NLP community to develop innovative solutions and applications. We developed Myanmar spelling error-correction parallel corpus which is an expansion of the existing parallel corpus [2] [3]. The spelling error text data are collected manually from news, blogs, posts and comments on social media platforms (e.g., Facebook, Telegram) and short conversational dialogues. The corpus comprises erroneous sentences sourced from news outlets (e.g., Myanmar Now, BBC Burmese, Popular News Journal), entertainment news platforms (e.g., Myanmar Celebrity TV, Sunday Journal), as well as blogs related to various categories including beauty, health, food, travel, lifestyle, education and online business pages.

In our error-correction parallel corpus preparation, the words identified as errors are annotated to their corresponding correction following the rules and regulations of the Myanmar orthography book published by *Ministry of Education, Department of Myanmar Nationalities' Languages Myanmar Language Commission* [20]. The misspelled words are manually annotated according to corresponding error types.

- Example: <ဝဋ္ဌ|con>, <-ီး|sensitive>, <တိုင်း|typo>, <ပျောကော်စ|seq>, and <လှဲ့|pho>.

When a word falls into two error categories, it is tagged as belonging to both with the symbol “-” after each class.

- Example: <ပိ|pho-typo>, < ေ◌ညး|slang-seq> and <ေေတွ၇◌ာက်|con-seq>.

The dataset comprises 57,880 sentence pairs. The frequency of each error type is detailed in Figure 2. Sentences containing errors are tagged with their respective error types and are distinguished from the corrected sentences by three pipes (|||), as illustrated below:
<အကျီ|stack> အ<ေ◌ရာင်|seq> အ<စင်|pho>လေးကလှလိုက်
တာ|||<အင်္ကျီ> အ<ရောင်> အ<ဆင်>လေးကလှလိုက် တာ
(“The color and the design of the shirt is beautiful.” in English)

We have built Myanmar spelling error type corpus from the developing Myanmar spelling error-correction parallel corpus with the intention of training classification systems for spelling error types. We extracted the misspelled words that are annotated with their corresponding error types (eg., <ေ◌_သ|sensitive>) along with the adjacent syllables. The syllable segmentation process was performed using

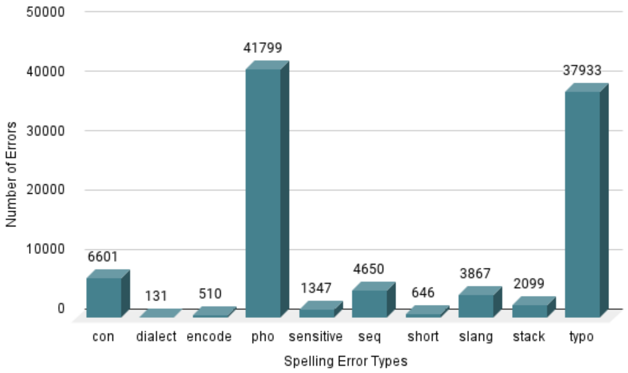


Fig. 2: Statistic of Spelling Error Corpus

Myanmar syllable segmentation tool: sylbreak4all [21]. This corpus contains misspelled syllable sequences ranging from 2-grams to 5-grams, accompanied by their corresponding error types. The corpus size is 107,935 n-grams syllable sequences and it is split into 90% for training the model and 10% for testing. Some examples from the corpus are shown below.

- မ ‌ေ- _ သ ရုံ ||| sensitive
- မြောက် ကွီး ယား ခေါင်း ||| slang
- တော် ◌² များ ||| short

B. Experimental Settings

TM requires binary vectors or numerical matrices as input representation to effectively learn patterns from data. To meet the input requirements of TM, we adopted the bag-of-words (BoW) approach. Our method involves converting text data into vectors based on the word n-grams. BoW model is straightforward and computationally efficient, making it suitable for large-scale text processing tasks. In this work, the task of converting text into vectors is accomplished using CountVectorizer [22]. We used character n-grams for vectorization, with a default range of one to three characters and an alternative range of one to four characters.

We tuned the parameters of TM such as `clauses`, `T`, `s` and `epochs`. `Clauses` represent the number of clauses TM will use to address the text learning process, comparable to the number of hidden nodes in a neural network layer. The `T` value sets the threshold, controlling how readily the clauses are allocated to represent each specific sub-pattern. Moreover, the `s` value is a crucial hyperparameter that affects the learning dynamics of the model, particularly influencing the probabilities of Reward, and Penalty during the learning process, as described in Section III. We employed a conventional TM model [23] for each experiment. The model is configured with different clauses, and the training proceeds for 100 complete iterations through the training dataset. Within each of these iterations, we experimented with different values for the 'epochs' parameter, each value determined the number of internal parameter updates for the model within a single pass through the dataset during the training of TM. The parameters `T` and `s` play a crucial role in shaping the behavior of TM for multiclass classification tasks. We conducted experiments to assess the impact of varying `T` values and `s` values on the model training process.

The spelling error type classification task using TM is compared with the fastText library, developed by Facebook's AI Research team. The Myanmar spelling error type corpus was transformed into a specified format suitable for input into the fastText classifier, as fastText requires the labeled dataset to be in a specific format `__label__<label_name> <text>`. The model was trained with the supervised classification technique within the fastText framework, with a learning rate of 0.1 and softmax function. We studied the performance of the model by tuning the training epoch, starting at epoch 10 and increasing the epoch number by 10 until it reached 100.

VI. Results and Discussion

A. Evaluation Metrics

To evaluate the performance of a multiclass classification model, three measures of assessment: precision, recall and F1-measure. These metrics are essential for a comprehensive understanding of how well a model accomplishes its intended task. Precision measures the accuracy of the positive prediction for each class. Precision is computed by adding up the total true positive predictions for all classes and dividing it by the sum of true positives and false positives across all classes. The formula of precision is as follow:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4}$$

Recall measures the count of positive class predictions relative to all positive instances present in the dataset. Recall is calculated as:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{5}$$

F1 score provides a balanced evaluation of a model's performance and it is useful when there is an imbalance between positive and negative instances in multiclass classification. The value of the metric has an interval of 0 to 1 and a value of 1 represents the optimal performance, resulting in no false positive or false negative predictions. The F1 score is calculated as in Equation 6.

$$\text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \tag{6}$$

B. Result of Tsetlin Machine

We studied the performance of Myanmar spelling error type classification using TM with the BoW model. We experimented with a range of TM hyperparameters, including the threshold for the summarization of the clauses in TM (`T` value) and the parameter for the reward and penalty strength during learning (`s` value).

For BoW-based character n-grams vector representation, experiments were performed using two different n-gram ranges, specified by the minimum and maximum values: (1, 3) and (1, 4). TM parameters—number of clauses, T value, s value, and the number of training epochs for each invocation of the fit method in TM model [23]—were set to 20, 15, 3.9, and 1, respectively. We trained the model for 100 complete iterations through the dataset. The experimental results for these configurations are summarized in Table I, with the precision, recall, and F1 score as evaluation metrics.

TABLE I: Performance Metrics for Spelling Error Type Classification using TM with BoW model over Diverse N-Gram Range Settings

ngram_range	Precision	Recall	F1 score
(1, 3)	0.58	0.55	0.54
(1, 4)	0.53	0.55	0.53

Next, we evaluate the impact of T to the performance of the model trained with TM. In this experiment, we considered three different T values: 10, 15, and 50. We examined the impact of incrementing from 10 to 15, with only a slight change in scores from the 5-unit increment. Therefore, we opted to evaluate the model's performance using the value of 50. We used 20 clauses, the s value of 3.9, CountVectorizer's ngram_range value of (1, 3), the epoch number of internal parameter updates in the fit method of 1 and trained the model for 100 iterations. Precision, recall and F1 score are presented in Table II. By analyzing the scores, it was observed that T values significantly influence the performance of our models.

TABLE II: Performance Metrics for Spelling Error Type Classification using TM with BoW model throughout Various 'T' Parameter Settings

T	s	Precision	Recall	F1 score
10	3.9	0.56	0.56	0.54
15	3.9	0.58	0.55	0.54
50	3.9	0.54	0.52	0.52

In Table III, we present the precision, recall, and F1 score for spelling error type classification, considering varying s parameter values. During this experiment, we configured the number of clauses to 20, established a T value of 15, initialized the epoch parameter in the fit() method of TM model to 1 and used ngram_range value of (1, 3). We referenced the s value of 3.9 which is used in NosiyXOR and Sentiment Analysis experiments, 5.0 in Breast Cancer Demo, and IMDB Text Categorization Demo and 8 in text classification for 20 Newsgroup Dataset. When adjusting the parameter s in our model, we found that lower s values, such as 3.9 and 5, achieved higher scores than higher s value of 8. Through analysis, s value had an impact on the performance of the model and lower s value can give the best results indicating a harmonious trade-off between precision and recall.

TABLE III: Performance Metrics for Spelling Error Type Classification using TM with BoW model across Varied Parameter 's'

T	s	Precision	Recall	F1 score
15	3.9	0.58	0.55	0.54
15	5	0.56	0.53	0.52
15	8	0.55	0.44	0.46

Through the selection of hyperparameters, including T value of 15, s value of 3.9, and the number of clauses of 20, we conducted a training experiment with distinct inner training epochs (fit_epochs). The model undergoes three values of internal parameter updates within a single pass through the dataset during the training of TM, including 1, 3, and 5. The results presented in Table IV demonstrate that the number of inner training epochs for the fit method has a significant impact on the performance of our model. Specifically, the results indicate an improvement in the F1 score, which increases from 0.54 (with 1 epoch) to 0.58 (with 5 epochs), marking the highest F1 score attained across all experiments.

TABLE IV: Performance Metrics for Spelling Error Type Classification using TM with BoW model throughout Different Inner Training Epochs

T	s	Epochs	Precision	Recall	F1 score
15	3.9	1	0.58	0.55	0.54
15	3.9	3	0.58	0.56	0.56
15	3.9	5	0.63	0.57	0.58

We examined the model's performance using varying numbers of clauses, specifically 20, 100, 200, and 300. Our approach involved utilizing a s value of 3.9, setting the fit-epoch parameter to 5, and experimenting with T values of 15 and 50. Due to time limitations and constraints on computing resources, the experiments were limited to 30 iterations for training the model. Table V presents the performance metrics for the top three optimal parameter configurations involving the number of clauses and the T values. The results suggest that increasing the number of clauses is beneficial since there is more information to be learned. Additionally, adjusting the T value is crucial for optimizing model performance, likely due to the increased complexity of the task.

TABLE V: Performance Metrics for Spelling Error Type Classification using TM with BoW model varying clauses and T values

Clauses	T	s	Precision	Recall	F1 score
100	15	3.9	0.72	0.71	0.69
200	50	3.9	0.73	0.70	0.69
300	50	3.9	0.75	0.73	0.72

C. Result of fastText

We investigated the performance of the fastText multiclass classifier for spelling error type classification and we found that the precision, recall, and F1 score had a consistent pattern over a range of training epochs from 10 to 100. Table VI shows precision, recall and F1 score for best three epoch numbers: 20, 40, and 60. The similarities in scores between the epochs indicate how early in the training process the model was able to converge to a stable state. The slight variation in the evaluation metrics emphasize the model's capacity to consistently provide accurate and reliable classification of spelling error types.

TABLE VI: Performance Metrics for Spelling Error Type Classification using fastText

Epochs	**Precision**	**Recall**	**F1 score**
20	0.842	0.781	0.810
40	0.841	0.780	0.809
60	0.840	0.779	0.808

D. Decision Information or Interpretability of Tsetlin Machine

TM can learn different clauses for each class in a multiclass classification task. Each clause represents the conditions or patterns that the model has learned to associate with a specific class. When a test word is input into TM, the model evaluates the word against all the clauses of all the classes to determine the most appropriate class for that word. Within our experiment, each of the ten classes, spanning from Class 0 to Class 9, is represented by 300 clauses. To provide insight into decision information of TM, we will explore a few examples of clauses, as outlined below:
Example:

1) Class 0 (___label___con) Clause 3: ု ∧ ဝ မ ∧¬ ဆာ ∧¬ ဂိ ∧¬ ဆွ ∧¬uk ∧¬က – ∧¬ဆက် ∧¬ဆု ∧¬နံ ∧¬ါ ∧¬ု လ ∧¬် ဆ
2) Class 9 (___label___typo) Clause 11: အ့

Example 1 represents the clause for Class 0, a class for consonant spelling error type, which recognizes the misuse of 'ဝ' (consonant) as '၀' (number zero) as explained in Section IV. This caluse has been learned from some sentences such as လုံး<ဝ|con> မသုံးပါနဲ့, လုံး<ဝ|con> မဖြစ်မနေ, လုံး<ဝ|con> မစား, which extracts training example: "*___label___con* လုံး ဝ မ". Therefore, the rule is defined as when if the testing word contains 'ု' and 'ဝ မ', indicative with consonant errors and does not contain other literals related to other classes such as 'ဆာ' for phonetic error and 'ါ' for encoding error, it will classify as Class 0.

Example 2 designed to detect a frequent typographic error 'အ့', which is often mistyped as the word 'အဲ့'. TM has described the rule such that if the test input contains 'အ့', it will categorize as Class 9, typographic error.

E. Comparative Analysis of Tsetlin Machine and fastText

We conducted a comprehensive analysis of our optimal model, utilizing classification with TM, text feature extraction with BoW model, 300 clauses, an inner training epoch value of 5, T of 50 and s of 3.9. The performance of this model is compared with the one using fastText classifier across various spelling error labels. According to the results in Figure 3, TM has obtained higher or comparable precision to fastText in certain classes. On the other hand, recall is comparable for 'pho' and 'typo' which are the top two largest classes, while recall scores are extremely low across most classes, as shown in Figure 4.

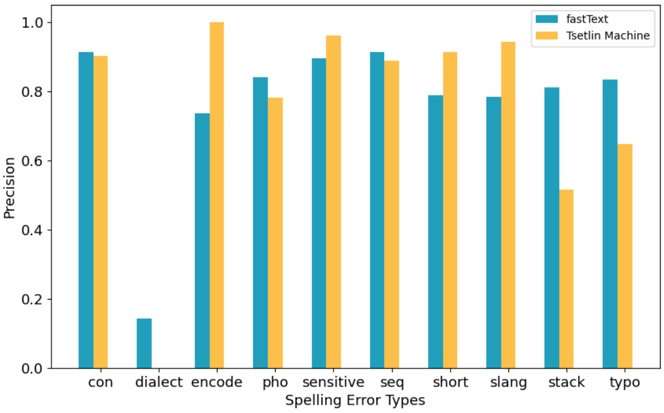


Fig. 3: Precision for TM's Classification with BoW model and fastText's Classification across Each Spelling Error Type

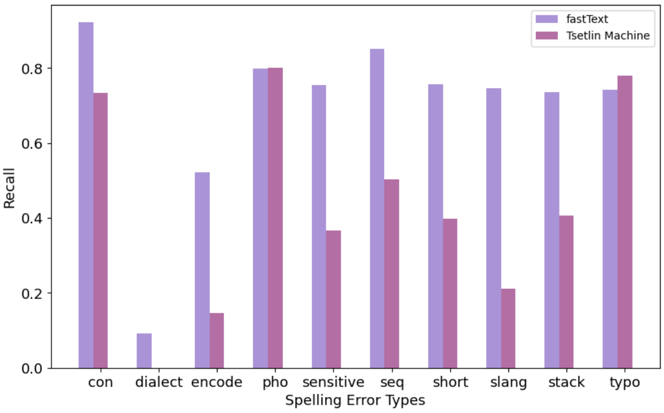


Fig. 4: Recall for TM's Classification with BoW model and fastText's Classification across Each Spelling Error Type

The results, depicted in Figure 5, underscore that TM has achieved comparable performance to fastText in classifying phonetic errors. TM has the proficiency in identifying consonant error with the F1 score of 0.812 and typographic error with the F1 score of 0.707. However, the model encounters challenges in identifying specific error labels such as 'slang' and 'encode'. While achieving remarkably high precision scores of 0.943 and 1 for the categories 'slang' and 'encode', respectively, the corresponding recall values are extremely low, resulting in lower F1 scores of 0.345 and 0.255. TM and fastText exhibit higher classification results for phonetic, typographic, and consonant errors; although, both models struggle to identify dialect errors. This may be attributed to the limited number of error sequences for the dialect category, with only a small number of instances, making it relatively small compared to other categories. In Figure 6, we can aslo observe that most of the error patterns are recognized as ‘typo’ and ‘pho’ errors. The difference in performance can be attributed to the disparate sizes of the training datasets. Specifically, the number of instances for phonetic errors and typographic errors significantly surpasses the training data for other error categories.

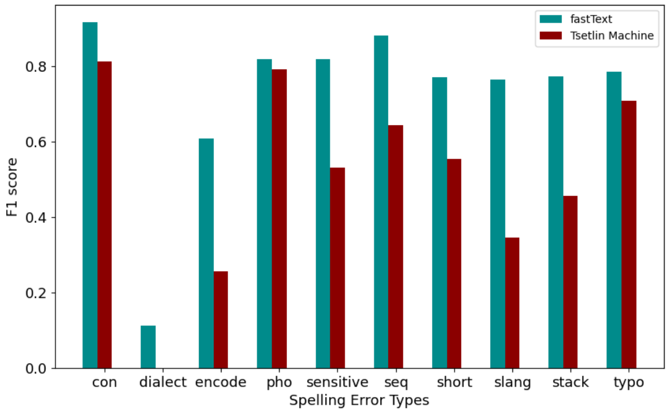
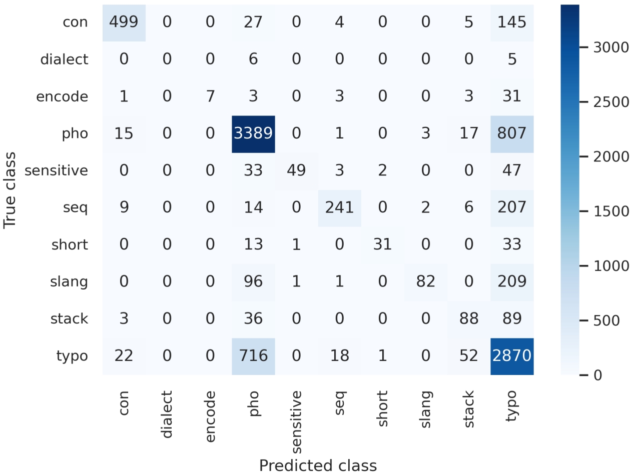


Fig. 5: F1 Scores for TM’s Classification with BoW model and fastText’s Classification across Each Spelling Error Type



	con	dialect	encode	pho	sensitive	seq	short	slang	stack	typo
con	499	0	0	27	0	4	0	0	5	145
dialect	0	0	0	6	0	0	0	0	0	5
encode	1	0	7	3	0	3	0	0	3	31
pho	15	0	0	3389	0	1	0	3	17	807
sensitive	0	0	0	33	49	3	2	0	0	47
seq	9	0	0	14	0	241	0	2	6	207
short	0	0	0	13	1	0	31	0	0	33
slang	0	0	0	96	1	1	0	82	0	209
stack	3	0	0	36	0	0	0	0	88	89
typo	22	0	0	716	0	18	1	0	52	2870

Fig. 6: Confusion Matrix for Spelling Error Type Classification Using TM with optimal parameter settings

F. Discussion

Our experimental results showed that the choice of T and s affected the performance of the spelling error type classification model. The T parameter determines the number of clauses required for a decision in the voting mechanism; a larger T means more clauses participate in the voting, thereby influencing the feedback to the Tsetlin Automata states. According to the results by setting the s value to 3 and number of clauses to 20, as presented in Table II, lower T values lead to better performance because they allow TM to make predictions with fewer clauses agreeing, thereby increasing the model’s flexibility and responsiveness to the data.

Moreover, the s parameter controls the specificity of the clauses in TM by influencing how strongly we favor Include actions. A higher s value stimulates the Tsetlin Automata to more strongly include literals in their clauses, which can capture finer details but may lead to overfitting. Conversely, a lower s value results in more general clauses, which can prevent overfitting but might miss important details. According to the results as shown in Table III, lower s values lead to better performance metrics. This is because they allow TM to create more general clauses, increasing the model’s ability to generalize from the training data and improving its precision, recall, and F1 score.

To better understand the capabilities of TM and to examine the effect of error type imbalance in our dataset, we conducted binary classification experiments using a balanced dataset. This approach allows us to evaluate whether TM can effectively distinguish between error types when given an equal representation of each error type. We selected samples from the ‘con’, ‘pho’ and ‘typo’ classes, each of which has more than 5,000 samples. The samples from each selected class are defined as the positive class, while randomly selected samples from remaining classes serve as the negative class. We trained the model for ‘con’ class with 6,000 syllable sequences for the positive class and 6,000 for the negative class. In these experiments, TM with the parameters (T – 50, s – 3, number of clauses – 300, fit-epoch - 5) achieved a comparable score to fastText as illustrated in Table VII.

TABLE VII: Performance Metrics of Binary Classification on the ‘con’ Class with Balanced Dataset

	Precision	**Recall**	**F1 score**
fastText	0.97	0.97	0.97
TM	0.97	0.97	0.97

We also conducted the experiments for the ‘pho’ class and ‘typo’ class and the experimental results are shown in Table VIII and Table IX. According to the analysis of the conducted experiments, we can improve the quality of the system by using a balanced dataset. Imbalances can lead the model to be biased towards the majority classes, resulting in relatively higher performance for those categories. Incorporating these insights into the development process can lead to a more effective and accurate spelling error correction system.

TABLE VIII: Performance Metrics of Binary Classification on the ‘pho’ Class with Balanced Dataset

	Precision	**Recall**	**F1 score**
fastText	0.86	0.86	0.86
TM	0.86	0.85	0.85

TABLE IX: Performance Metrics of Binary Classification on the ‘typo’ Class with Balanced Dataset

	Precision	**Recall**	**F1 score**
fastText	0.84	0.84	0.84
TM	0.82	0.82	0.82

VII. Conclusion and Future Work

The classification of Myanmar spelling errors using TM and fastText represents a significant advancement in developing an effective Myanmar spelling checker. This study introduces the first Myanmar spelling error type classification using TM with BoW word representation method and classification using fastText. We have constructed a Myanmar spelling error type corpus containing 100k syllable sequences across ten distinct classes for training and testing the models. We analyzed the effects of the number of clauses, `T` and `s` parameters on TM's performance and evaluated fastText's classification capabilities over various training epochs. We aim to publish our error type corpus to further support the Myanmar NLP community. Furthermore, our ongoing efforts are directed towards creating a Myanmar spell checker that incorporates state-of-the-art technologies, with the objective of enhancing its precision and accuracy.

Acknowledgment

This research is financially supported by National Research Council of Thailand (NRCT), Thailand Advanced Institute of Science and Technology (TAIST), National Science and Technology Development Agency (NSTDA), Tokyo Institute of Technology, and Faculty of Engineering, Kasetsart University (KU) under the TAIST Tokyo Tech Program. We would also like to express our sincere gratitude to Thura Aung from Language Understanding Lab., Myanmar, for his assistance with the data collection process.

References

[1] Ole-Christoffer Granmo, “TM - A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic”, August, 2018.

[2] Ei Phyu Phyu Mon, Ye Kyaw Thu, Thida San, Zun Hlaing Moe, Hnin Aye Thant, “Automatic Rule Extraction for Detecting and Correcting Burmese Spelling Errors”, The 4th ONA Conference, 17-18 December, Ministry of Posts and Telecommunications, Phnom Penh, Cambodia, 2021.

[3] Ei Phyu Phyu Mon, Ye Kyaw Thu, Than Than Yu, Aye Wai Oo, “SymSpell4Burmese: Symmetric Delete Spelling Correction Algorithm (SymSpell) for Burmese Spelling Checking”, 2021 16th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Ayutthaya, Thailand, 2021, pp. 1-6.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, “Bag of Tricks for Efficient Text Classification”, August, 2016.

[5] Aung Kyaw Htet, Mark Dras, Myanmar XNLI: Building a Dataset and Exploring Low-resource Approaches to Natural Language Inference with Myanmar, 02 May 2024.

[6] Mya Ei San, Sasiporn Usanavasin, Ye Kyaw Thu, Manabu Okumura, “A Study for Enhancing Low-resource Thai-Myanmar-English Neural Machine Translation”, ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), Volume 23, Issue 4, Apr 2024, Article No.54, pp. 1:24.

[7] Thura Aung, Ye Kyaw Thu, Zar Zar Hlaing, “mySentence: Sentence Segmentation for Myanmar Language using Neural Machine Translation Approach”, Journal of Intelligent Informatics and Smart Technology, Oct 2nd Issue, 2023, pp. 1 to 9.

[8] Nang Aeindray Kyaw, Ye Kyaw Thu, Hutchatai Chanlekha, Thazin Myint Oo, Okumura Manabu, Thepchai Supnithi, “Enhancing Hate Speech Classification in Myanmar Language Through Lexicon-Based Filtering”, the 21st International Joint Conference on Computer Science and Software Engineering (JCSSE 2024), June 19-22, Phuket, Thailand.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, June, 2019.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, “Language Models are Few-Shot Learners”, May 2020.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach”, 2019.

[12] Rupsa Saha, Ole-Christoffer Granmo, Morten Goodwin, “Mining interpretable rules for sentiment and semantic relation analysis using TMs.” International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer International Publishing, 2020.

[13] Xuanyu Zhang, Hao Zhou, Ke Yu, Xiaofei Wu, Anis Yazidi, 2023, “TM for Sentiment Analysis and Spam Review Detection in Chinese”, Algorithms 16, no. 2: 93.

[14] Aye Myat Mon, Thandar Thein, “Myanmar Spell Checker”, International Journal of Science and Research (IJSR), India, Online ISSN: 2319-7064, Volume 2 Issue 1, January, 2013.

[15] “Symmetric Delete spelling correction algorithm (SymSpell)”, https://github.com/wolfgarbe/SymSpell

[16] “1000x Faster Spelling Correction algorithm”, https://seekstorm.com/blog/1000x-spelling-correction

[17] John Carroll, Darrell Long, “Theory of Finite Automata With an Introduction to Formal Languages”, Prentice Hall, January, 1989.

[18] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolo, “Enriching Word Vectors with Subword Information”, Jun, 2016.

[19] Joshua Goodman, “Classes for fast maximum entropy training”, 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221), Salt Lake City, UT, USA, 2001, pp. 561-564 vol.1.

[20] “မြန်မာစာလုံးပေါင်းသတ်ပုံကျမ်း”, Department of Myanmar Language Commission, Ministry of Education, Union of Myanmar, September 2019.

[21] Ye Kyaw Thu, Hlaing Myat Nwe, Hnin Aye Thant, Hay Man Htun, Htay Mon, May Myat Myat Khine, Mi Hsu Pan Oo, Mi Pale Phyu, Nang Aeindray Kyaw, Thazin Myint Oo, Thazin Oo, Thet Thet Zin, Thida Oo, ”sylbreak4all: Regular Expression based Syllable Breaking Tool for Nine Major Ethnic Languages of Myanmar”, In Proceedings of the 16th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP 2021), virtual conference, Dec 21 to Dec 23, 2021, Thailand, pp. 1-6.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesna, “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[23] “Python implementation of TM based classifier”, https://github.com/cair/pyTsetlinMachine

Ei Thandar Phyu is currently pursuing a M.Eng. in Artificial Intelligence and Internet of Things at the Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Thailand. She received her B.Eng. in Information Science and Technology from University of Technology (Yadanarpon Cyber City), Myanmar. She is also a member of Language Understanding Lab., Myanmar. Her research interests include Natural Language Processing, and Data Analysis.

Ye Kyaw Thu is a Visiting Professor at NECTEC, Thailand, where he has been working since January 2019, and Founder of the Language Understanding Lab, Myanmar. He holds a Doctor of Science (2011) and a Master of Science (2006) from Waseda University, Japan, and a Bachelor of Science in Physics from Dagon University, Myanmar (2000). He also earned diplomas in Computer Studies (UK). His research focuses on Artificial Intelligence (AI), Natural Language Processing (NLP), and Human–Computer Interaction (HCI). He supervises students at various institutions, including Assumption University (AU), Kasetsart University, the King Mongkut's Institute of Technology Ladkrabang (KMITL), and the Sirindhorn International Institute of Technology (SIIT).

Thazin Myint Oo received her Bachelor of Computer Science (B.C.Sc. Hons.) and Master of Computer Science (M.C.Sc.) degrees from the University of Computer Studies, Yangon, in 2005 and 2008, respectively. From 2008 to 2014, she served as an Assistant Lecturer at the University of Computer Studies, Yangon, and from 2014 to 2019, as a Lecturer at Computer University, Kyaing Tong, Myanmar. She was an Associate Professor at the University of Computer Studies, Yangon, from 2019 to 2021. In 2024, she earned her Ph.D. in Computer Science from Assumption University of Thailand. She is currently working as a Senior Researcher at the Language Understanding Lab (LU Lab) in Myanmar.

Hutchatai Chanlekha is an assistant professor at the Department of Computer Engineering, Faculty of Engineering, Kasetsart University, Thailand. She earned her B.Eng and M.Eng degrees in Computer Engineering from Kasetsart University and holds a Ph.D. in Informatics from the Graduate University for Advanced Studies in Japan. Her research interests include Natural Language Processing, applied Machine Learning, and Knowledge Engineering.

Thepchai Supnithi received the B.S. degree in mathematics from Chulalongkorn University, in 1992, and the M.S. and Ph.D. degrees in engineering from Osaka University, in 1997 and 2001, respectively. He is currently a Director of Artificial Intelligence Research Group, National Electronics and Computer Technology Center (NECTEC), Thailand. He is currently also a Vice President of Artificial Intelligence Association of Thailand.