Received May 17, 2020, accepted July 7, 2020, date of publication July 21, 2020, date of current version July 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010828

Thai Spelling Correction and Word Normalization on Social Text Using a Two-Stage Pipeline With Neural Contextual Attention

ANURUTH LERTPIYA[1], TAWUNRAT CHALOTHORN[2], AND EKAPOL CHUANGSUWANICH[3]

[1]Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
[2]Kasikorn Labs Co., Ltd., Kasikorn Business Technology Group, Nonthaburi 11120, Thailand
[3]Chula Intelligent and Complex Systems, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Corresponding author: Ekapol Chuangsuwanich (ekapolc@cp.eng.chula.ac.th)

ABSTRACT Text correction systems (e.g., spell checkers) have been used to improve the quality of computerized text by detecting and correcting errors. However, the task of performing spelling correction and word normalization (text correction) for Thai social media text has remained largely unexplored. In this paper, we investigated how current text correction systems perform on correcting errors and word variances in Thai social texts and propose a method designed for this task. We have found that currently available Thai text correction systems are insufficiently robust for correcting spelling errors and word variances, while the text correctors designed for English grammatical error correction suffer from overcorrections (text rewrites). Thus, we proposed a neural-based text corrector with a two-stage structure to alleviate issues of overcorrections while exploiting the benefits of a neural Seq2Seq corrector. Our method consists of a neural-based error detector and a Seq2Seq neural error corrector with contextual attention. This novel architecture allows the Seq2Seq network to produce corrections based on both the erroneous text and its context without the need for an end-to-end structure. Our method outperformed all the other evaluated text correction systems. When compared to the second-best result (copy-augmented transformer), our method further reduced the word error rate (WER) from 2.51% to 2.07%, improved the generalized language evaluation understanding (GLEU) score from 0.9409 to 0.9502 on the Thai text correction task, and improved the GLEU score from 0.7409 to 0.7539 on the English spelling correction task.

INDEX TERMS Natural language processing, machine learning, artificial neural networks, text generation, spelling correction, text normalization, Thai language.

I. INTRODUCTION

The fast and widespread adoption of social media as a means of communication has led to an explosive increase in user-generated text data on the Internet. Natural language processing (NLP) techniques are often used to keep up with the pace of rapidly growing data and introduce new and exciting applications such as real-time disease surveillance [1] and monitoring the public perceptions of brands, products, and services (social listening). However, social text also introduces challenges not previously found in traditional written media (e.g., news, published articles), such as a wide variety of language usage from users with varying levels of language proficiency, the diverse culture of Internet users, and a lack of formality and professionalism in the written texts [2], [3]. Natural language text correction systems (e.g., spell checkers and grammatical error correctors) are used to help improve writing quality by providing feedback on the correctness of written text and proposing corrections to the authors. The published literature related to Thai text correction has primarily focused on postprocessing results from optical character recognition (OCR) systems [4]–[8]. However, the large quantity of data on social media, which is input via other interfaces (e.g., physical and virtual keyboards), does not strictly exhibit the same types of errors as do data from OCR systems. Moreover, the text correction systems developed and employed in free open source software (FOSS) have yet to be evaluated on social texts.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

In this paper, we investigate how to perform spelling correction and word normalization tasks effectively on Thai communicational text collected from social media. Henceforth, we collectively refer to the tasks of spelling correction and word normalization as the text correction task (TC), refer to Thai communicational text collected from social media sites as Thai user-generated web content (Thai UGWC) and the spelling errors and correctable variances of words in the TC task as errors. The types of errors that naturally occur in Thai UGWC and the types of errors we aim to correct in Thai TC are covered in Section III-A. The contributions of this paper are as follows.

First, we evaluate the currently existing techniques for Thai text correction, as well as techniques borrowed from a similar task, English grammatical error correction (GEC). We examined a variety of text correction techniques, ranging from dictionary-based (i.e., Hunspell [9]) and statistically based methods (i.e., PyThaiNLP [10]) to modern systems featuring sequence-to-sequence neural networks employed in state-of-the-art English GEC systems (i.e., Bi-GRU Seq2Seq [11], Copy-Augmented Transformer [12]).

Second, we propose a text correction system designed for the TC task. Our proposed method features a two-stage structure containing a neural-based error detector and a neural sequence-to-sequence (Seq2Seq) error corrector with contextual attention. This novel neural architecture enables the Seq2Seq corrector to produce corrections based on both the detected errors and the text surrounding the error (context) without requiring an end-to-end (E2E) structure. As reported in Section VI-A, relying solely on the Seq2Seq corrector can lead to overcorrections (text is rewritten as opposed to simply corrected).

The remainder of this paper is structured as follows. Section II discusses works relating to Thai TC. Section III outlines our TC tasks (spelling correction and word normalization) task on Thai UGWC as well as the development of our Thai UGWC dataset. Section IV describes our proposed two-stage TC system for Thai UGWC. Section V details our experimental setups. Section VI discusses the results of other models we experimented with alongside those of our proposed method. Finally, Section VII reiterates our contributions and concludes the paper.

II. RELATED WORKS

In this section, we explore previous works related to our TC task on Thai UGWC. This section is split into four parts: an overview of text correction systems, a brief history of spell checkers, the works relating to Thai text correction, and the English grammatical error correction literature.

A. TEXT CORRECTION SYSTEMS FOR NATURAL LANGUAGE

In this section, we provide an overview of the two primary types of text correction systems: two-stage systems and end-to-end systems.

Two-stage systems separate the text correction task into two phases: error detection (detector) and error correction (corrector). For example, the detector in dictionary-based systems [9], [13] classifies whether a token is an error by searching its dictionary. The reliance on prebuilt dictionaries limits the detectable errors to nonword errors only (words not in the dictionary). More statically complex models have been proposed to achieve better error detection [3]–[5], [8], [10]. In the error correction stage, one or more tokens are chosen as a correction for each of the errors identified. The corrector in dictionary-based systems may suggest words based on spelling similarity and use some form of tie-breaking (e.g., the prior probability of a word derived from word frequency encoded in the dictionary). The accuracy of dictionary-based correctors suffers because context is often necessary to select the proper substitution. The use of language models (LMs) has been proposed to overcome this issue and produce context-dependent corrections [4], [5].

End-to-end systems (E2E) combine the detection and correction stages into a single step by reformulating the error correction task as a machine translation task (MT). Error correction is formulated as a translation from an ''erroneous/informal'' language into a ''correct/formal'' language. These systems are employed in modern correction systems designed for the English grammatical error correction task. Techniques from statistical machine translation [11], and subsequently, neural machine translation (NMT) [11], [14], [15] have been employed with great success compared to the traditional two-stage systems. More specialized architectures [12], [16] and techniques for data augmentation and training [12], [17] emerged later.

B. SPELL CHECKERS

Spell checkers are traditionally defined as systems for identifying nonword errors (words that do not exist in the dictionary). Given a dictionary, the task of building spell checkers is considered an engineering problem, where performance [18] or the performance-accuracy trade-off [19] is the primary concern. However, in languages where minor spelling errors often result in a valid dictionary word (e.g., Thai), spell checkers are also *expected* to detect real-word errors (errors that are valid words in the dictionary) [4], [5], [8]. A wide variety of methods have been proposed for non-English spelling correction: including dictionary-based, rule-based, statistically based, deep-learning-models and statistical machine translation models [20]. Hunspell [9] (dictionary-based) is the most widely adopted spell checker; it is used by LibreOffice, OpenOffice.org, Mozilla Firefox 3, Mozilla Thunderbird, and Google Chrome. However, Hunspell's popularity is likely due to the large number of languages it supports (56 languages).

C. THAI TEXT CORRECTORS

Publicly available works for Thai TC can be grouped into two categories: published literature and FOSS.

The published literature on Thai TC has focused heavily on correcting errors produced by optical characteristic recognition (OCR) systems [4]–[8]. In contrast, text correction

for text input via human-computer interfaces (HCIs), such as keyboards, is an underresearched area [20]. FOSS text correctors (e.g., Aspell [19], Hunspell [9], PyThaiNLP [10], [13]) are primarily meant for correcting text from HCIs.

Most Thai TC systems are two-stage systems. A variety of statistical models have been proposed for the detection stage: dictionary [9], character-gram [3], [8], WinNow [4], [5], and conditional random fields (CRF) [10]. However, the works on correction models include only dictionary-based [9] and statistical language models using part-of-speech (POS) trigrams [4], [5]. The current statistical methods used in error detectors cannot detect errors that require extended context [8] or require manual feature engineering to address out-of-vocabulary tokens [4], [5].

On the other hand, E2E systems for Thai TC based on token-passing algorithms rely exclusively on prebuilt dictionaries [6], [7]. Thus, these methods cannot address out-of-vocabulary tokens (i.e., names) at all.

D. ENGLISH GRAMMATICAL ERROR CORRECTION

The GEC task is an extension to the spelling correction task whose goal is to automatically produce a grammatically correct sentence when given an erroneous sentence—without changing the meaning. The most notable standard benchmark dataset for this task is the Conll-2014 shared task [21], which consists of essays written by English as a second language (ESL) learners and the corresponding corrections annotated by teachers (language owners).

Significant and recent advancements on the GEC task is the reformulation of GEC into MT. This reformulation has proved highly successful and has shifted the area of research from two-stage systems (refered to as ''classifier systems'' in the GEC literature) to end-to-end systems [22], [23].

III. THAI TEXT CORRECTION TASK

This section describes the Thai TC task and our dataset, which is built from user-generated web content (UGWC).

The goal of TC systems is to detect and correct errors that exist in the input text. In the scope of this research, we are interested only in correctable errors in UGWC. Details on the different types of errors that occur in UGWC (and which are correctable) are covered in Section III-A. In our task, errors are defined as words not in The Royal Institute Dictionary [24] or a word (or a sequence of words) that falsely represents the original intent of the author (e.g., ''sea'' in ''I sea the light.''). Such errors originate from two primary sources: the input method and nonstandard language usage by the authors.

Textual data input via different methods suffer from different types of errors. One type can be introduced from unreliable input methods. For example, artifacts from OCR systems (i.e., similar-looking characters being mistaken for another character), incorrect keyboard decoding (typos: striking improper keys), and even from false corrections by automatic correction systems (e.g., autocorrect on virtual keyboards on touch screen enabled devices). In this work, we primarily correct errors that originate from texts input by Internet users (i.e., keyboard decoding errors).

TABLE 1. Examples of different types of errors and their respective corrections.

Type of Error	Occurrence (%)	Error	Correction
Misspelling	61.38	ทุกคน	ทุกคน
		คว่ำบัตร	คว่ำบาตร
Morphed	24.00	ครัช	ครับ
		ตั้ลล้าคคคค	น่ารัก
Abbreviation	15.00	มค	ม.ค.
		พน	พรุ่งนี้
Spoonerism	0.14	พับกบ	พบกับ
Slang	0.08	ตีเนียน	No correction
		อ้อย	ทอดสะพาน
Other	5.63	โรบินสัน	No correction

TABLE 2. Size of the UGWC dataset and the training-testing split.

	Tokens	Err. Tok.	Err. Seg.	Lines
Train + Dev	7,211,994	247,921	179,803	108,597
Train	6,894,886	236,404	171,487	103,597
Dev	317,108	11,517	8,316	5,000
Test	635,822	22,665	16,537	10,000

The demographics of the authors also play a role in the types of errors in a text. Errors can be attributed to nonstandard language use by the authors: intentional use of nonstandard words or nonstandard word spellings (e.g., morphed words, spoonerisms, and slang) and unintentional spelling errors (e.g., misspellings). For example, the characteristics of errors that occur in a business letter differ from those in a social media post.

A. UGWC DATASET

Our UGWC dataset is an expanded version of our previous UGWC dataset [3] and is constructed from text data collected from users of online social media platforms. This data differs from data collected from other online outlets (e.g., news sites) where the content is typically created by professionals and is often curated. Due to privacy concerns, the UGWC dataset for spelling correction and word normalization will be only *partially* released by Chulalongkorn University for future research purposes. The dataset consists of both longer bodies of text (e.g., discussions on public forums) and shorter conversational dialogues (e.g., posts and comments on social media). The dataset items have a mean length of 66 words and a median length of 19 words. Details on the size of our data are shown in Table 2. Errors and the corresponding corrections were annotated by language-major students from the Faculty of Arts of Chulalongkorn University. Errors typically involve one or more of the six main types of errors: misspelled words, morphed words, slang, spoonerism, incorrect abbreviation, and others. Errors in non-Thai languages are ignored (annotated as correct), and lines of text consisting purely of other languages were filtered out prior to data annotation. Real-world examples of each type of error and the

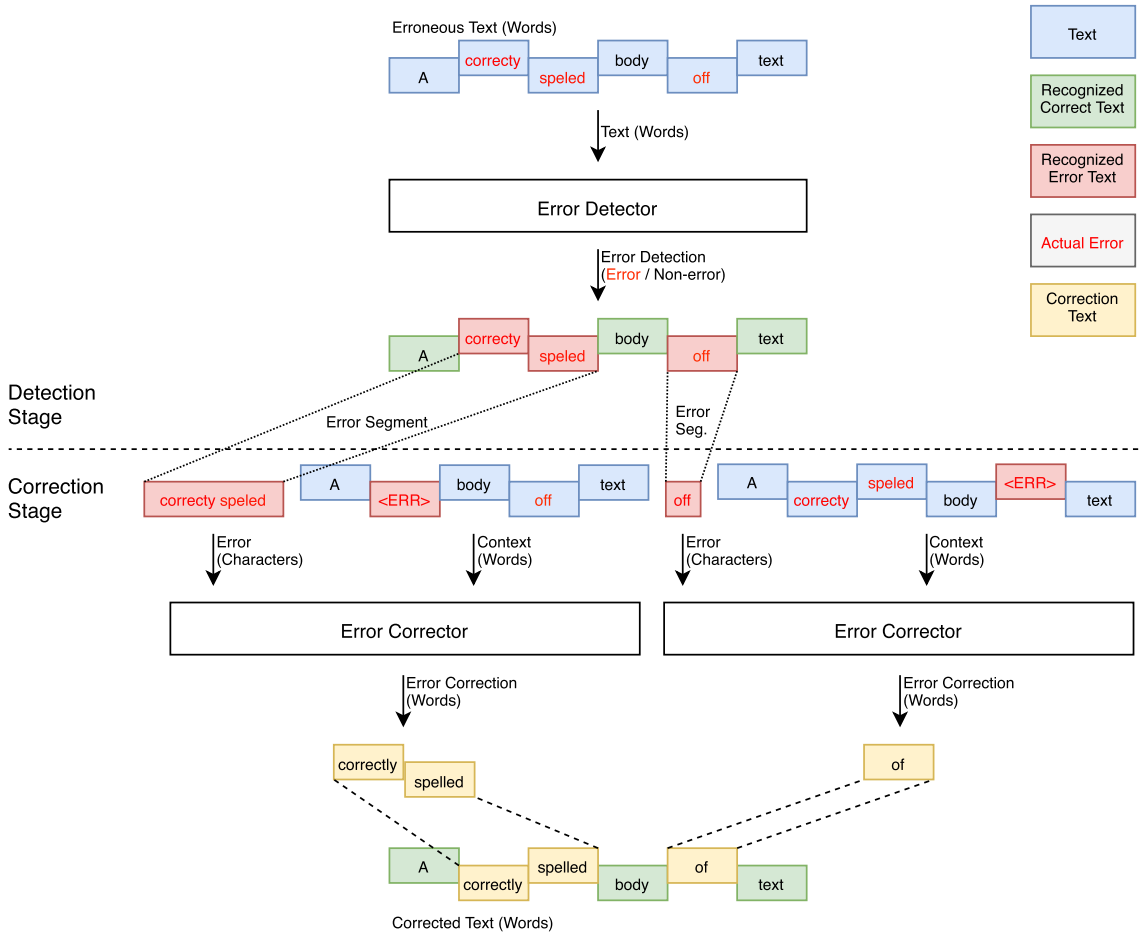

FIGURE 1. Overview of our text correction system. The " " (space) character between two words in the error segment is added for visual clarity. "<BEGIN>" and "<END>" tokens are omitted to reduce clutter.

respective corrections are shown in Table 1. Explanations of each type of error and their English equivalents are outlined below.

Misspelled words are words whose spelling deviates from the standard spelling (according to The Royal Institute Dictionary) of the intended word. In this study, misspelled words are not limited to words that do not appear in the dictionary. For example, the word "sea" in "I cannot sea in the dark" is a misspelling of the intended word "see", although the word "sea" is a valid word in the dictionary.

Morphed words are words intentionally morphed to emphasize emotions or replicate human speech. For example, by intentionally misspelling the phrase "sooo goooood" the author may intend to imitate vowel stresses as they might occur in a verbal conversation.

Incorrect abbreviation notations include abbreviated words that are misspelled (e.g., "USA." instead of "USA" or "U.S.A.") and words that are abbreviated despite not having an official abbreviation (e.g., "brb", which is an unofficial abbreviation of "be right back").

Spoonerisms are a form of wordplay on sound commonly found in informal Thai dialogue. An English example would be writing "beautiful world" as "weautiful borld".

Slang can consist of either new words (e.g., "Frenemy", which is a combination of "friend" and "enemy") or repurposed words that take new meanings (e.g., the verb "ride" is sometimes used as a noun to refer to a "car").

"Other" errors include words that do not exist in the dictionary, words that do not have an official spelling in Thai (i.e., named entities), and words that imitate sounds (e.g., "ahh", "eww", and "aww").

For our TC task, we are interested only in correctable errors. Thus, slang with no correction and "other" errors are not considered. The UGWC contains three separate sets of samples: a training set, a development set, and a test set, as shown in Table 2.

IV. METHOD

This section outlines our proposed two-stage TC method for Thai TC. This section is split into three subsections: model description, data augmentation, and training. The model section details the structure of our proposed text correction system. The data augmentation section describes the data augmentation techniques applied during training. Moreover, the training section outlines the techniques we found to be effective in improving model performance.

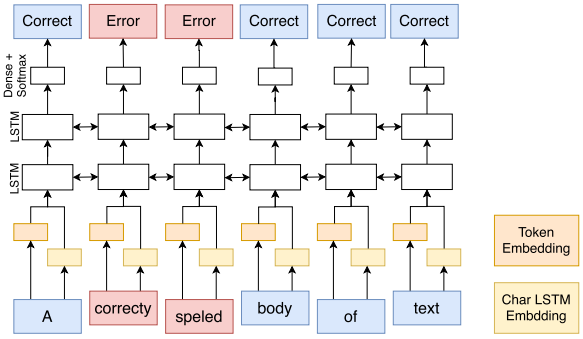

FIGURE 2. Error detector operating on a sequence.

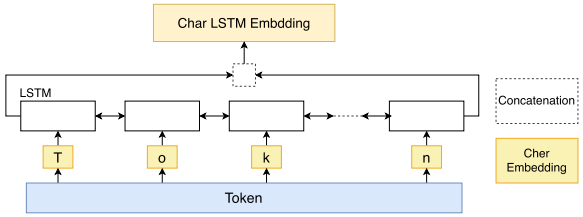

FIGURE 3. Character LSTM embedding layer encoding a word token.

A. MODEL

This section describes the two-stage corrector: the error detection stage and the error correction stage. A structural overview of the entire pipeline is shown in Fig 1. The inputs and outputs of each stage and the details of the models are outlined below.

Error Detection Stage: The input to the error detection stage is a sequence of words containing potentially erroneous input text $\vec{w} = \{w_1, w_2, \ldots, w_N\}$, where N is the total number of words. The detection stage uses the error detector to predict a sequence of labels of the same length $\vec{l} = \{l_1, l_2, \ldots, l_N\}$, where a prediction l_i denotes the prediction of the corresponding word w_i. A word is labeled either erroneous or correct $l_i \in \{error, correct\}$, where the erroneous label denotes correctable errors as defined in Section III.

Error Correction Stage: The error correction stage is given the same input sequence $\vec{w} = \{w_1, w_2, \ldots, w_N\}$ and error detection prediction $\vec{l}$. The error correction stage should produce the appropriate corrected sequence $\vec{w}^* = \{w_1^*, w_2^*, \ldots, w_M^*\}$ while leaving every correct input word unaltered. The error correction stage achieves this by extracting error segments from the error detection result. An error segment is a contiguous sequence marked as erroneous. The correction stage then uses the error corrector to produce a sequence of correction words to replace each error segment. The sequence-to-sequence structure of our error corrector allows the correction stage to produce a corrected sequence that may differ in length from the input sequence $N \neq M$. For example, given the input ป | ยัง | ใช้ | ไม่ | ได้ | อีก | หรอ | ครับ with the 1st, 2nd, and 7th words labeled as erroneous, the correction stage would extract two error segments: ป | ยัง and หรอ. Given the correction ยังไง and หรือ, the correction stage will produce the corrected sequence ยังไง | ใช้ | ไม่ | ได้ | อีก | หรือ | ครับ.

Error Detector: Our error detector is a bidirectional-LSTM (bi-LSTM) [25], [26] binary sequence tagger. An illustration of the detector is shown in Fig 2. The model consists of a word embedding layer (with a size of 64), a character embedding layer (with a size of 128), a character bi-LSTM encoder (32 nodes in each direction), a two-layer bi-LSTM (64 nodes in each layer and direction), and an output dense projection layer with a softmax activation function. The character-level embeddings are produced from the concatenation of the character encoder bi-LSTM last hidden state in both directions, as shown in Fig 3. The sequence tagger estimates the probability of each input word as either erroneous or correct. The detection threshold is selected based on the error detection F_1-score on the development set (detailed in Section VI-E). The vocabulary of the error detector is created by selecting the n most common words from the corrected text of our training data. This approach minimizes the number of erroneous words in our vocabulary because the presence of label noise causes a small number of words in the corrected text to be erroneous. We selected the 24,576 (3×2^{13}) most common words as our vocabulary. Words not in our vocabulary are replaced with a special out-of-vocabulary (OOV) token. We also explored using of sub-word units to handle OOV by evaluating our model with SentencePiece tokens [27] rather than word tokens. In the SentencePiece variant of our model, the vocabulary size is also 24,576 tokens. During training, the detection model is optimized using Adam [28] with a learning-rate of 0.002 on the cross-entropy loss. See Appendix B for a consolidated list of hyperparameters.

Error Corrector: Our proposed error corrector is an autoregressive sequence-to-sequence (Seq2Seq) neural network. An illustration of the overall structure is shown in Fig 4. For each error segment, the corrector is given the error segment in *characters* and the context of the error segment in *words*. The context is the input sequence with a portion of the error segment replaced with a special ERR token. The corrector then produces a sequence of *words* as a correction for the error segment. The difference between our model a typical Seq2Seq network is our context-aware encoder, which includes a contextual attention layer. Details of the encoder and the decoder of the corrector are provided later in this paper. The corrector shares the same vocabulary as the error detector. Corrections containing OOV tokens are discarded, and the error segment is left unaltered. The corrector is optimized using Adam [28] with a learning rate of 0.002 on the cross-entropy loss. The main hyperparameters are $m = 24$ and $n = 128$. See Appendix B for a consolidated list of hyperparameters.

1) ENCODER

The context-aware encoder is composed of two embedding layers, 3 bi-LSTM encoders, and a contextual attention layer as illustrated in Fig 5. The contextual attention layer allows the corrector to encode both the erroneous sequence and the

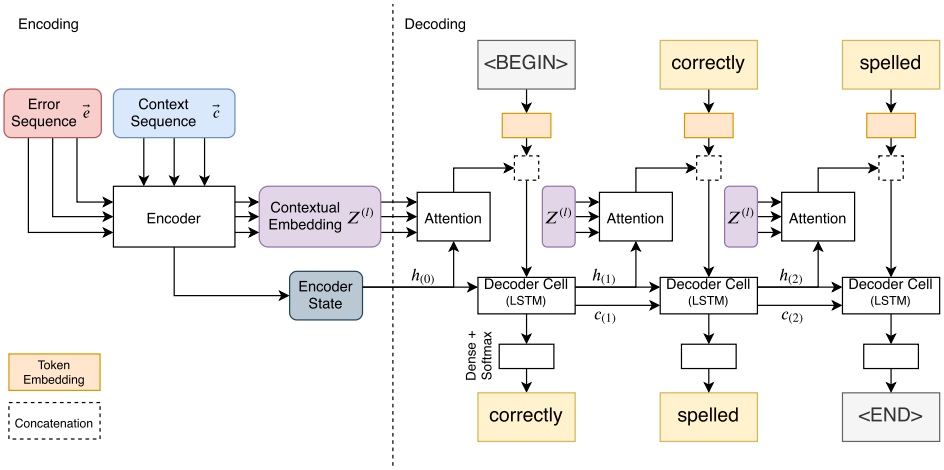

FIGURE 4. Structure of the Seq2Seq text corrector. The model is split into two parts: the encoder and decoder, separated by the dotted line. This figure highlights the decoder; a more detailed view of the encoder is shown in Fig 5 . Three parallel arrows represent passing a sequence of vectors (a matrix), while a single arrow represents passing a single vector.

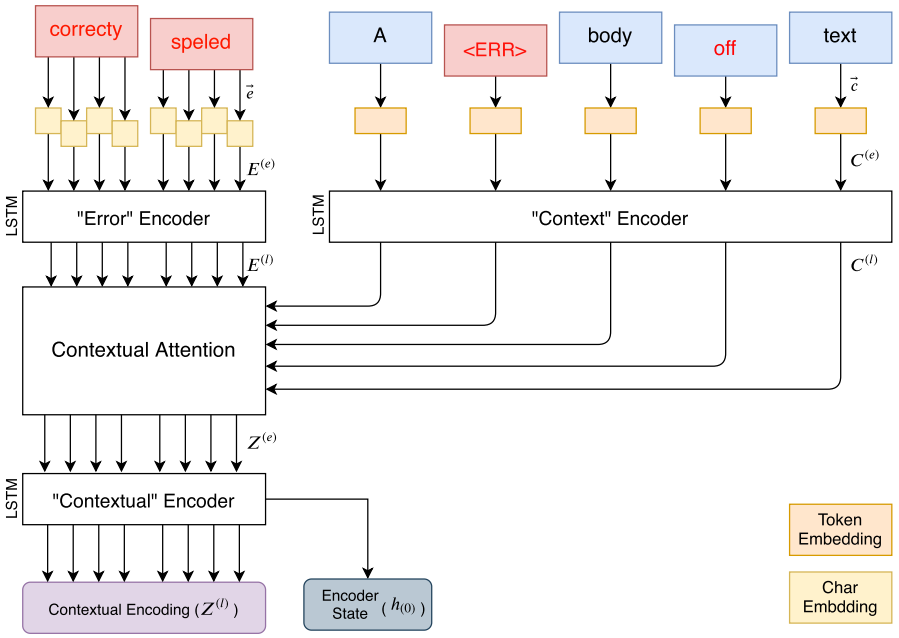

FIGURE 5. Structure of our context-aware encoder. The boxes denoted with LSTM represent bidirectional LSTM layers. The "<ERR>" token is a special token that denotes the position of the erroneous input within the context. Both the error input and context also feature special "<BEGIN>" and "<END>" tokens but those are omitted to reduce clutter. Each of the erroneous character embeddings is simplified in the drawing.

context sequence into the encoded sequence. The encoder was inspired by the query-to-context attention mechanism in BiDAF, which is a proven architecture originally proposed for the machine comprehension task [29]. BiDAF is used to model a sequence generation task for two input sequences of varying lengths. The encoder in BiDAF computes two attention matrices: query-to-context (Q2C) and context-to-query (C2Q), which are combined along with the encoded query into a single encoded sequence that represents both the query and the context. Our context-aware encoder encodes the context by performing dot-product attention from the erroneous sequence to the context sequence. This approach is similar to the Q2C attention in BiDAF. Our erroneous sequence is equivalent to the query sequence in BiDAF, and our context sequence is equivalent to the context sequence in BiDAF. The encoded context represents the information in the context relevant to decoding the erroneous characters. Our preliminary experiments showed that the corrector performed better when utilizing only BiDAF's Q2C encoder rather than the full array of encoders in BiDAF. Our experiments were developed using the AllenNLP framework [30] and the BiDAF implementation in AllenNLP [31] as a reference. The details of each layer of our context-aware encoder are described below.

a: ERROR ENCODING

The erroneous character tokens $\vec{e} = \{e_1, e_2, \ldots, e_J\}$ of the error segment are embedded with the character embedding layer. The embedding layer projects each character into a $2m$ vector space, which produces a matrix $E^{(e)} \in \mathbb{R}^{2m \times J}$. The character embeddings are then encoded by the ''error encoder'' (a bidirectional LSTM with m nodes in each direction) into an erroneous-encoding matrix $E^{(l)} \in \mathbb{R}^{2m \times J}$.

b: CONTEXT ENCODING

The contextual word tokens $\vec{c} = \{c_1, c_2, \ldots, c_K\}$ are embedded with the word embedding layer. The word embeddings project each word into a $2m$ vector space, which produces a matrix $C^{(e)} \in \mathbb{R}^{2m \times K}$. The word embeddings are then encoded by the ''context encoder'' (a bidirectional LSTM with m nodes in each direction) into a context-encoding matrix $C^{(l)} \in \mathbb{R}^{2m \times K}$.

c: CONTEXTUAL ENCODING

The erroneous-encoding matrix $E^{(l)}$ and the context-encoding matrix $C^{(l)}$ are then input to the contextual attention layer, which computes the contextual embeddings matrix $Z^{(e)} \in \mathbb{R}^{4m \times J}$. The contextual embedding $z_j^{(e)} \in \mathbb{R}^{2m}$ is a concatenation of the error encoding $e_j^{(l)} \in \mathbb{R}^{2m}$ and the error-to-context vector $x_j \in \mathbb{R}^{2m}$ as shown in Eq 2. The error-to-context matrix $X^{(e)} \in \mathbb{R}^{2m \times J}$ is the attention of error encoding on the context encoding computed from the similarity matrix $S \in \mathbb{R}^{K \times J}$ as shown in Eq 1.

$$\begin{aligned} S \in \mathbb{R}^{K \times J} \quad & s_{kj} = c_k^{(l)T} \cdot e_j^{(l)} \in \mathbb{R} \\ A \in \mathbb{R}^{K \times J} \quad & a_j = softmax(s_j) \in \mathbb{R}^K \\ X^{(e)} \in \mathbb{R}^{2m \times J} \quad & x_j = (C^{(l)T} \cdot a_j)^T \in \mathbb{R}^{2m} \end{aligned} \tag{1}$$

$$Z^{(e)} \in \mathbb{R}^{4m \times J} \quad z_j^{(e)} = [e_j^{(l)}; x_j] \in \mathbb{R}^{4m} \tag{2}$$

Subsequently, the contextual embeddings $Z^{(e)} \in \mathbb{R}^{4m \times J}$ are encoded by the ''contextual encoder'' (a bidirectional LSTM layer containing n nodes in each direction) into a contextual-encoding matrix $Z^{(l)} \in \mathbb{R}^{2n \times J}$.

2) DECODER

The decoder is a typical LSTM decoder (a unidirectional LSTM containing $2n$ nodes) with an attention mechanism [32] that observes contextual encoding, as shown in Fig 4. The decoder produces a correction for the error segment. Because the corrector is an autoregressive network, the decoder operates by predicting a token $w_t^{(c)}$ given the token predicted from the previous timestep $w_{t-1}^{(c)}$; therefore, the tokens before and after the actual correction words are special tokens, as shown in Eq 3, where L is the number of words in the correction.

$$\vec{w}^{(c)} = \{BEGIN, w_1^{(c)}, w_2^{(c)}, \ldots, w_L^{(c)}, END\} \tag{3}$$

The decoding process is repeated until the timestep following the end of the correction sequence $t = L + 1$, where the network is expected to output a special *END* token to indicate the end of the sequence. The hidden state $h_0 \in \mathbb{R}^{2n}$ of the decoder LSTM is initialized with the final hidden state of the ''contextual encoder''. The input to the LSTM decoder is a concatenation of the word embeddings and the context vector, as shown in Eq 4. The token produced from the previous timestep $w_{t-1}^{(c)}$ is embedded with the word embedding layer, which produces word embeddings $e_t^{(e)} \in \mathbb{R}^{2n}$. The context vector $e_t^{(c)} \in \mathbb{R}^{2n}$ is computed with dot-product attention from the previous hidden state h_{t-1} to the encoded sequence $Z^{(l)}$. The embedding e_t is a concatenation between the word embeddings and the context vector, as shown in Eq 4.

$$\begin{aligned} e_t &= [e_t^{(e)}; e_t^{(c)}] \in \mathbb{R}^{4n} \\ h_t, c_t &= LSTM_{decoder}(h_{t-1}, c_{t-1}, e_t) \end{aligned} \tag{4}$$

The decoder is trained with teacher forcing. Thus, during training, the decoder's input is derived from data instead of the output from the previous timestep.

B. DATA AUGMENTATION

We experimented with injecting noise into the dataset during model pretraining to help increase the number of erroneous examples in the training set. Our method was inspired by the data augmentation technique employed in the copy-augmented transformer [12]. However, we inject character errors rather than word errors. We inject three types of errors: a random character deletion, a random character substitution, and a random character insertion. Each type of error possesses a 3% probability of appearing for every position in the text. When a character is replaced or inserted, the replacement character is chosen at random based on the distribution of that character in the training set.

C. TRAINING

This section describes the training routine, which is shared by both the detection and the correction stages. Any details that differ between the two models are outlined in their corresponding subsection under Section IV-A.

The dataset is separated into three sets: a training set, a development set, and a test set. The test set is used only to report the model performance after training and to conduct the error analyses reported in this paper. The details of each set for UGWC are covered in Section III-A, while details on other tasks are listed in their corresponding experiments.

We evaluated three training configurations: training only on the training set, training only on the noise-injected training set, and models pretrained on the noise-injected training set and fine-tuned on the original training set. During fine-tuning, we reduced the learning rate to 0.0005 for both the detector and the corrector.

During training and pretraining, the models are validated (evaluated on the development set) to prevent overfitting between epochs. The corrector is evaluated after a fixed number of iterations (i.e., 25,000 error segments) instead of finishing the whole epoch. Early stopping patience is 20 epochs for the detector and 20 groups (of 25,000 error

segments) for the corrector. The models were evaluated using their respective loss functions. We found that neither using the F_1-score for the detector validation nor accuracy for the corrector validation resulted in improved performances.

V. EXPERIMENTS

In this section, we discuss the procedures used for each experiment in detail. The first subsection outlines the evaluation metrics used for each experiment. The second subsection outlines the experiments performed on our Thai TC task. The third subsection outlines the experiments performed on the publicly available Conll-2014 [21] dataset from the English GEC task and a cut-down version of Conll-2014 to create a spelling correction task for English.

A. EVALUATION CRITERIA

Word-error-rate (WER) and generalized language evaluation understanding (GLEU) [33], [34] were adopted as the metrics for the TC task. WER is the standard evaluation metric used in past literature on Thai TC [4], [5]. GLEU [33], [34] was developed as an evaluation metric for English GEC and has a high correlation with human preference by extending BLEU [35]. Because GLEU evaluates words based on n-grams instead of individual tokens, it tends to favor grouped errors over scattered ones, whereas WER treats all errors equally. For the English GEC and spelling correction tasks, we employed the standard M2 and GLEU for comparability with the existing literature [11], [12], [14]–[17], [21]. Our initial goal was to adopt both correction metrics from English GEC for our Thai TC task. However, we dropped M2 [36] due to a combination of M2's high computational complexity and the Thai language's lack of explicit sentence boundaries [37]. We found that a single paragraph of text can take upwards of an hour to evaluate.

B. THAI TEXT CORRECTION ON UGWC

We evaluated five methods on the Thai UGWC dataset: an industry-standard spell checker (i.e., Hunspell), a well-known Thai NLP toolchain (i.e., PyThaiNLP), two models from the English GEC task (i.e., Bi-GRU [11] and the copy-augmented transformer [38]), and our proposed method. We categorize the approaches into two groups: two-stage error correction (i.e., Hunspell, PyThaiNLP, and ours) and end-to-end (E2E) error correction (i.e., Bi-GRU and copy-augmented transformer). The configurations used for each method are outlined below, and the hyperparameter tuning is detailed in Appendix B.

Hunspell [9] was evaluated using both the provided prebuilt Thai dictionary and a dictionary constructed from the training data. The constructed dictionary built from the words in the corrected text of the UGWC training set. We experimented with multiple cut-off thresholds for a word to be added to the dictionary; however, we report using only the best performing threshold ($frequency \geq 1$).

PyThaiNLP is a popular toolchain in the Thai NLP community that employs techniques adapted from state-of-the-art research on other languages. PyThaiNLP has a ready-to-use text correction module that uses a two-stage approach. PyThaiNLP employs a detector that uses passive-aggressive CRF [39] and a Norvig corrector [38].

Two neural sequence-to-sequence models were evaluated: the bidirectional GRU (Bi-GRU) network [11] and the copy-augmented transformer [12]. Bi-GRU represents a baseline for a neural Seq2Seq model, because Bi-GRU is a strictly neural-based MT method that achieved relatively good performance at its time of publication. In contrast, the copy-augmented transformer represents the current state-of-the-art architecture from the English GEC task; it employs specifically designed techniques to perform text corrections (i.e., the copy substructure and pretraining on augmented data). For the Bi-GRU model, where the model is meant to operate on SentencePiece tokens (SP) [27], the SP tokens are encoded from tokenized Thai text and space tokens are used to denote word boundaries, the existing space characters are escaped (replaced with special characters).

C. ENGLISH GEC & SPELLING CORRECTION

Further experiments were performed to evaluate our method on two additional text correction tasks: the Conll-2014 shared-task for English Grammatical Error Correction [21] and an English spelling correction task we built from Conll-2014 [21]. The detailed annotation in Conll-2014 allowed us to create a TC that involved only spelling errors, which resembles our Thai TC task. All types of errors other than spelling errors (denoted with ''Mec'') were pre-corrected in both Conll-2013 and Conll-2014 to build the development and test sets. The training set is a combination of NUCLE [40] and Lang-8 [41] *without* any precorrections because there is no way to discern the type of error for each correction. Using NUCLE and Lang-8 as training sets, Conll-2013 as the development set, and Conll-2014 as the test set is a standard practice in the previous literature [11], [15], [23]. The hyperparameters and training configuration for our models remain the same as on the UGWC dataset, except for the detection threshold selection. We found that a threshold tuned for $F_{0.5}$ on the development set (instead of F_1) produced a better correction overall. The hyperparameters for other methods were based on their respective papers (i.e., Bi-GRU [11] and the copy-augmented transformer [38]).

VI. RESULTS AND DISCUSSION

In this section, we outline and discuss the results of our experiments. The TC approaches were evaluated on three tasks: our TC task on Thai UGWC and two TC tasks derived from the English Conll-2014 shared task [21].

A. THAI UGWC

In this section, we cover the results on the TC task on the Thai UGWC dataset. The results include the evaluation of existing techniques from Thai TC, English GEC, and our method.

The TC results on the Thai UGWC dataset are shown in Table 3. We categorized the results into two groups:

TABLE 3. Evaluation of end-to-end error correction on the Thai UGWC test set by various systems.

Model	Type	GLEU	WER (%)	ΔWER (%)
Do nothing (source text)	-	0.8845	3.77	0.00
Ideal correction (Oracle)	-	1.0000	0.00	-100.00
Off the shelf				
Hunspell	2-Stage	0.8267	8.11	+115.12
PyThaiNLP	2-Stage	0.8612	5.58	+48.01
Trained				
Hunspell	2-Stage	0.8598	5.57	+47.75
Bi-GRU (180 token limit)	E2E	0.4035	50.82	+1,247.92
Bi-GRU (20 token limit)	E2E	0.7404	15.97	+323.57
Bi-GRU (50 token limit)	E2E	0.7462	17.51	+364.38
Copy-Aug Transformer*	E2E	0.9374	2.58	-31.56
Copy-Aug Transformer**	E2E	0.9409	2.51	-33.42
Ours	2-Stage	0.9453	2.24	-40.66
Ours*	2-Stage	0.9361	2.83	-25.03
Ours**	2-Stage	**0.9502**	**2.07**	**-45.21**
Ours** (with Oracle Detection)	2-Stage	0.9774	1.08	-71.39

* Model is trained only on the noise-injected dataset.
** Model is pre-trained on the noise-injected dataset before being fine-tuned on the regular training-set.

off-the-shelf ready-to-use models and models trained on the UGWC training set. Two off-the-shelf models were evaluated: Hunspell with its prebuilt dictionary [9], PyThaiNLP [10]. Furthermore, we evaluated three trained models: Hunspell (dictionary-based) [9], Bi-GRU (Neural Seq2Seq) [11], and the copy-augmented transformer (Neural Seq2Seq with Augmentation) [12]. Samples of the corrections produced by the individual models are shown in Appendix A. The time required by each model to perform inference on the test set is shown in Table 5. Below, we discuss the shortcomings of the methods that struggled on the Thai TC task before reporting the overall results.

Correction systems with dictionary-based correctors (i.e., Hunspell [9], and PyThaiNLP [10]) often struggle to select a correct correction candidate. As a result, system with more conservative error detectors produce less incorrect corrections. Hunspell with its prebuilt dictionary performed the worst. Although we experimented with multiple cut-off thresholds for creating the custom dictionary for Hunspell, the best result is reported in Table 3.—the dictionary built from words in the corrected text from the training set (*frequency* ≥ 1). Although this model suffers from erroneous words in the dictionary, due to label noise in the corrected text, compared to the provided dictionary, the errors left uncorrected outweigh the potential errors introduced from corrections with false-positives.

Although the GEC literature may suggest that end-to-end (E2E) correction systems are the natural step forward for text correction systems, our results showed that the basic E2E text corrector is insufficient for correcting errors in Thai UGWC (see Bi-GRU [11] in Table 3). The error analysis showed that most of the errors made by Bi-GRU result from the model getting stuck in a loop, thus repeatedly producing the same groups of tokens. To combat this, we tried performing corrections on truncated inputs, which improves the score.

TABLE 4. Detailed WER evaluation of correction methods on the Thai UGWC test set.

	Sub. (%)	Del.* (%)	Ins.* (%)	WER (%)
Do nothing (source text)	2.84	0.64	0.27	3.77
Copy-augmented**	1.63	0.56	0.30	2.51
Ours**	**1.44**	**0.33**	**0.28**	**2.07**

* Due to the unpredictable nature of performing tokenization on erroneous text, the number of tokens before and after correction may not be consistent and thus result in deletion and insertion error when evaluated with WER.
** Model is pre-trained on the noise-injected dataset before being fine-tuned on the regular training-set.

However, the Bi-GRU with the best performing input size (20 tokens for WER and 50 tokens for GLEU) still suffered from looping and *overcorrections* (i.e., text being rewritten with a different meaning). As a result, Bi-GRU performed significantly worse than did the simpler two-stage correctors. In Table 3, Bi-GRU scores are reported from the model operating on SentencePiece tokens with the original space characters escaped. However, we also experimented with executing Bi-GRU on SentencePiece with untokenized Thai text, and that model still exhibits the issues mentioned above. Some samples produced by executing Bi-GRU on SentencePiece on untokenized Thai text are shown in Appendix A.

The copy-augmented transformer [12] showed massive improvement over Bi-GRU and even produced a positive word error rate reduction on the input text. However, the error analysis showed that the copy-augmented model still suffers from *overcorrections*, primarily randomly dropping words from the input and producing corrected text with a different meaning. Table 4 shows a breakdown of the WER score; the copy-augmented model shows substantially worse deletion and insertion error rates compared to our model, which confirms our analysis.

Our proposed method without data augmentation outperforms all the other models evaluated on the Thai TC task. The results also showed that pretraining on the augmented training set followed by fine-tuning further improves the correction performance. However, training on an augmented training set without fine-tuning significantly degrades the correction performance.

B. SentencePiece AS UNIT TOKENS

In this section, we evaluate how our model performs with two different token types: word tokens and SentencePiece tokens. SentencePiece (subword tokens) potentially allow a model to operate on text with an open vocabulary. Because SentencePiece-based models do not produce word boundaries, word tokenization is required to postprocess the results for evaluation. To ensure a fair comparison between the two models, we also retokenized the results from our word-based model. The results are shown in Table 6. In terms of GLEU, both models scored similarly. However, the word model substantially outperformed the SentencePiece model in terms of WER. The error analysis showed that when the word-based model is unable to produce a correction (outputs an OOV token), the SentencePiece-based model also produced

TABLE 5. Inference time on the Thai UGWC test set by various systems.

	Detection	Correction	End-to-end	Lines per Second
CPU				
Hunspell	0:00:02	0:11:15	0:11:17	14.79
Hunspell (Trained)	**0:00:01**	**0:04:02**	**0:04:03**	**41.06**
PyThaiNLP	0:04:18	4:16:37	4:20:55	0.65
Bi-GRU (20 token limit)	-	-	3:54:07	0.71
Copy-augmented	-	-	0:18:05	9.22
Ours	0:03:38	0:03:07	0:06:45	24.69
GPU				
Bi-GRU (20 token limit)	-	-	0:23:12	7.18
Copy-augmented	-	-	0:04:19	38.61
Ours	0:01:23	**0:01:23**	**0:02:46**	**60.20**

The CPU timing information is on an i7-7800X.
The GPU timing information is on a Geforce GTX 1080 Ti.
Inference is performed line-by-line (without line-level parallelism).

TABLE 6. Evaluation of retokenized output from our method with different unit token types on the Thai UGWC test set.

Model	GLEU	WER (%)	ΔWER (%)
Retokenized source text	0.8870	3.55	0.00
Ours with Word as unit token**	0.9565	**1.71**	**-51.72**
Ours with SentencePiece as unit token**	**0.9567**	1.94	-45.24

** Model is pre-trained on the noise-injected dataset before being fine-tuned on the regular training-set.

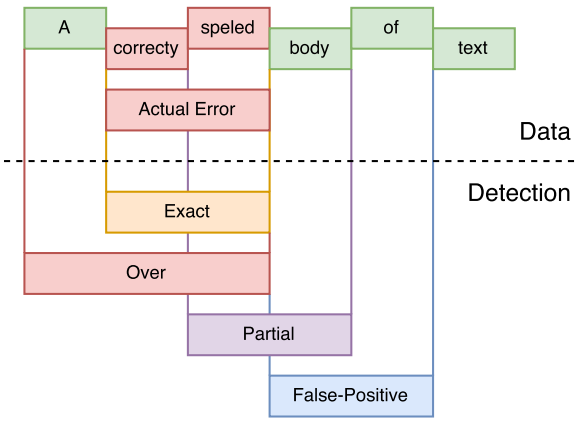

FIGURE 6. Four types of error segments.

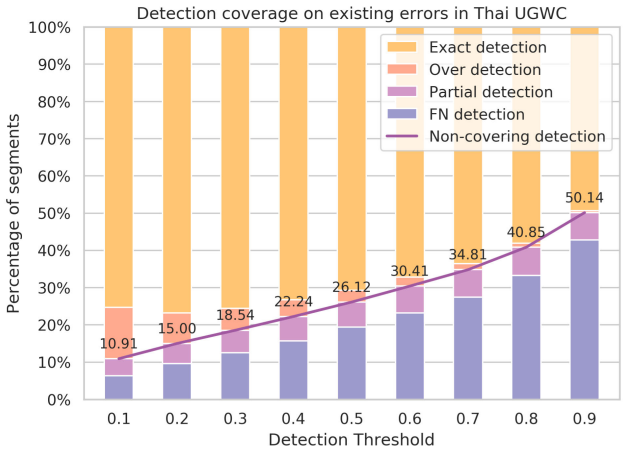

FIGURE 7. Detection coverage of our method on the Thai UGWC test set.

incorrect corrections. Due to false-positives in the detection stage, not correcting is the better option in such cases.

C. DETECTION STAGE

This section examines the detection stage from two aspects: detection coverage of the errors in the data and the error segments produced from the detection stage. An error segment can be classified into four types: exact detection, overdetection, partial detection, and false-positive detection. Fig 6 shows the four types of error segments. An exact detection occurs when the predicted boundaries of an error segment match the true boundaries of the error segment. Overdetection occurs when the predicted error segment covers an area larger than the actual error segment. A partial detection occurs when a predicted error segment only partially covers the actual error segment. Last, false-positive detection occurs when a predicted error segment does not overlap with any actual error segments. From the perspective of the actual errors in the data, reducing the detection threshold increases the detection coverage, as shown in Fig 7. However, there is a trade-off between the detection coverage and the number of false-positive detections, as shown in Fig 8 and Fig 9. Overdetection increases as the threshold decreases; but Interestingly, partial detection remains roughly the same across all detection thresholds. The evaluation broken down by error types is shown in Fig 10 and Fig 11. The performance is consistent for misspelled words, morphed words, and incorrect abbreviation notations but varies for spoonerisms and Slang, which have smaller numbers of samples.

D. CORRECTION STAGE

This section also examines the correction stage from two aspects: correction coverage of the errors in the data and the error segments that are corrected in the correction stage. Correction of a covering error segment (i.e., exact detection and overdetection) is corrected either accurately or incorrectly. For false-negative detection, the error is uncorrected.

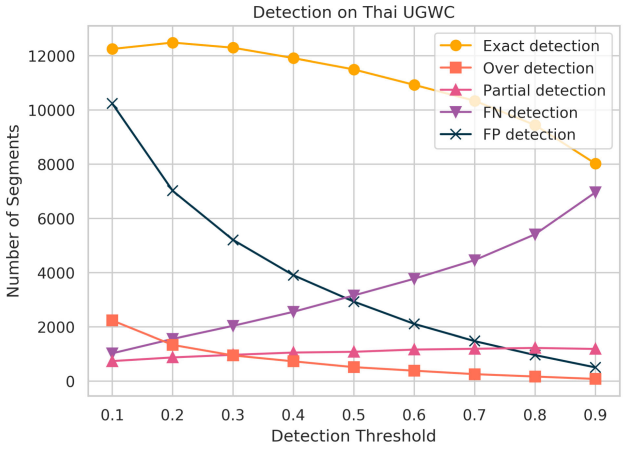

FIGURE 8. Types of error segments produced from the detection stage of our method on the Thai UGWC test set.

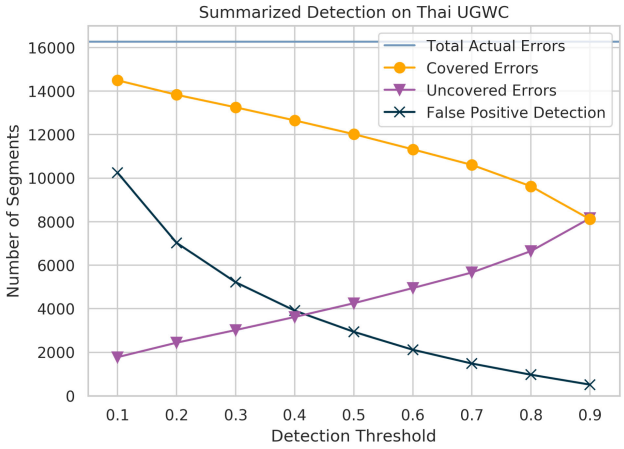

FIGURE 9. Coverage of the detection and false-positives produced from the detection stage of our method on the Thai UGWC test set.

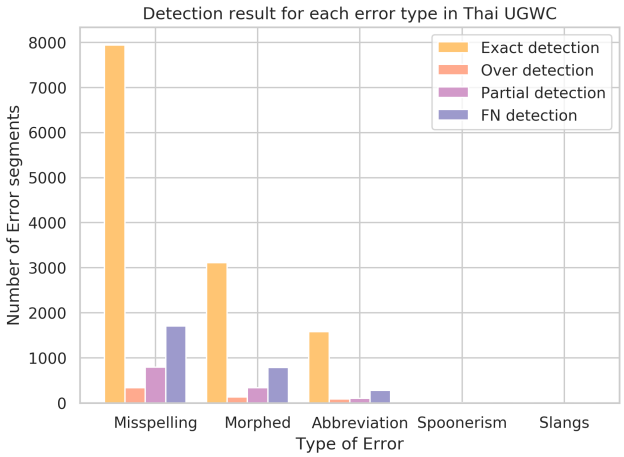

FIGURE 10. Number of error segments produced for different types of errors on the Thai UGWC test set.

For partial detection, any correction is considered incorrectly corrected because portions of the erroneous text lie outside

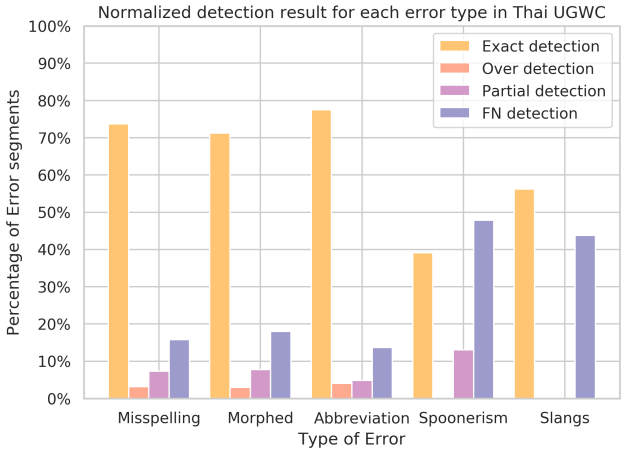

FIGURE 11. Normalized number of error segments produced for different types of errors on the Thai UGWC test set.

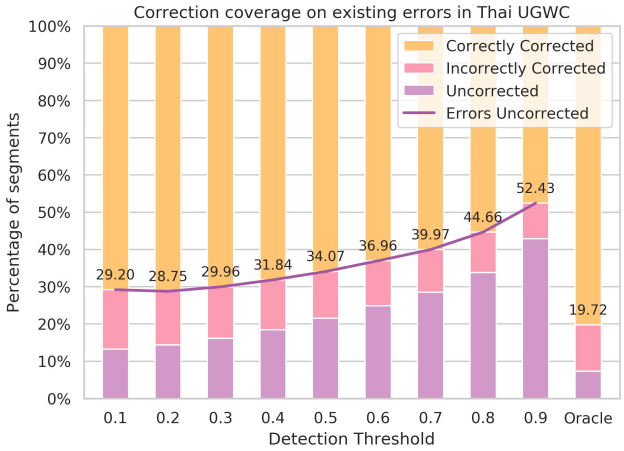

FIGURE 12. Correction coverage of our method on the Thai UGWC test set. "Oracle" is used to denote the results from using an ideal detection output. Uncorrectable Oracle detection is due to the corrector rejecting the detection by producing OOV tokens.

the error segments. Fig 12 shows the correction coverage of our method. From the perspective of the actual errors in the data, reducing the detection threshold tends to increase the correction coverage. However, the correction coverage flattens out at lower thresholds and even decrease slightly at a threshold of 0.1. Thus, a trade-off exists between the correction coverage and the number of remaining errors in the corrected text, as shown in Figs 13, 14, and 15. A breakdown of the evaluation by error types is shown in Figs 16 and 17. In line with error detection, performance is consistent for misspelled words, morphed words, and incorrect abbreviation notations but varies for spoonerisms and slang.

E. DETECTION SENSITIVITY

In this section, we evaluate how the detector performance correlates with the end-to-end correction performance at different detection sensitivities. The detection stage is evalu-

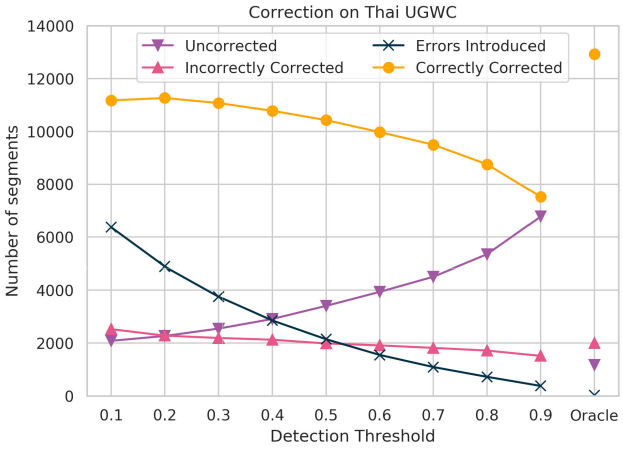

FIGURE 13. Types of corrections produced by our method on the Thai UGWC test set. "Oracle" is used to denote the results from using an ideal detection output. Uncorrectable Oracle detection is due to the corrector rejecting the detection by producing OOV tokens.

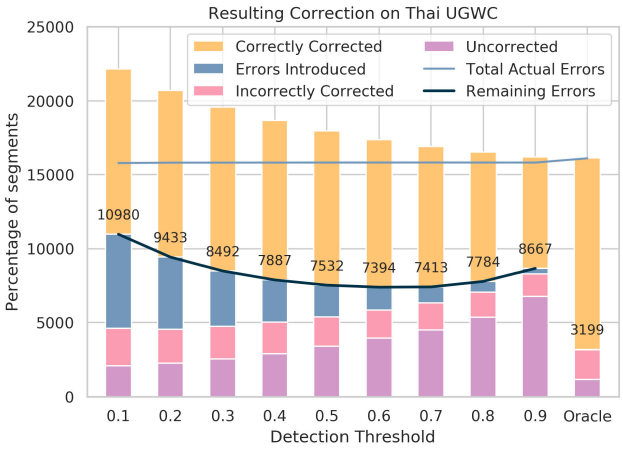

FIGURE 14. Breakdown of the remaining errors after executing our method on the Thai UGWC test set. "Oracle" is used to denote the results from using an ideal detection output. Uncorrectable Oracle detection is due to the corrector rejecting the detection by producing OOV tokens.

ated as a typical detection task using the F_1-score on the test set. The results are shown in Fig 18. We found that the trend of the GLEU score follows the detection F_1-score and that a threshold of 0.4 performs best on both metrics. For all the results reported outside of this section, we tune the detection threshold on the development set and set the threshold to 0.5.

F. ENGLISH GEC CONLL-2014

Our method was evaluated on two tasks: the full grammatical error correction (GEC) task and the spelling correction task built from Conll-2014. The results are shown in Tables 7 and 8. As expected, our method performs very poorly on the full GEC task because performing GEC effectively requires the ability to rewrite large portions of the text. The GLEU scored our model below doing nothing.

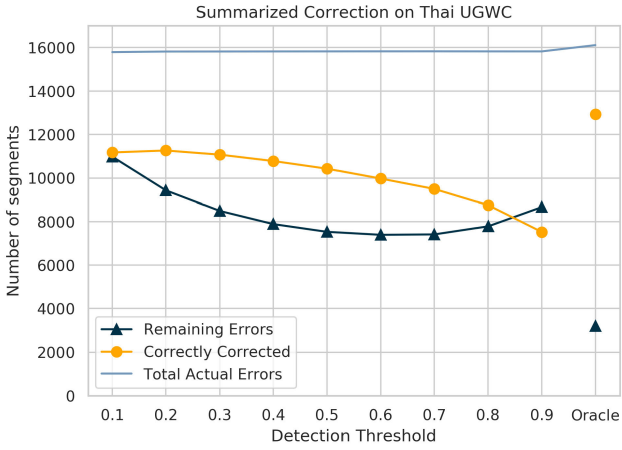

FIGURE 15. Coverage of the correction and the resulting number of errors from our method on the Thai UGWC test set. "Oracle" is used to denote the results from using an ideal detection output. The total number of segments is neither consistent with Fig 9 nor consistent across different detection thresholds, because nonideal detection can cause the true error segments to merge or split.

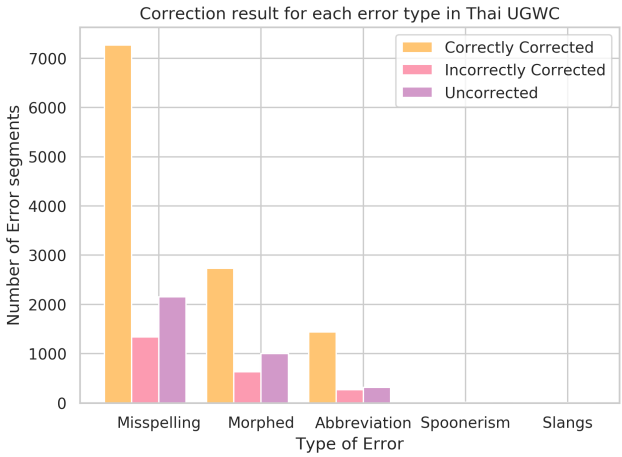

FIGURE 16. The number of error segments corrected for different types of errors on the Thai UGWC test set.

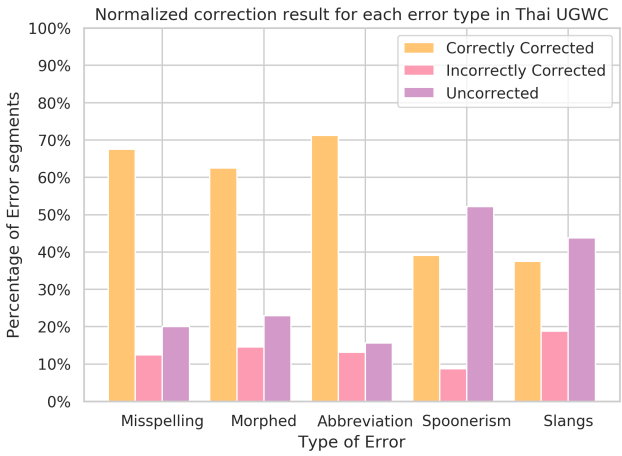

FIGURE 17. Normalized number of error segments corrected for different types of errors on the Thai UGWC test set.

To evaluate only the spelling correcting capabilities of our model, we built a spelling correction task using the Conll-

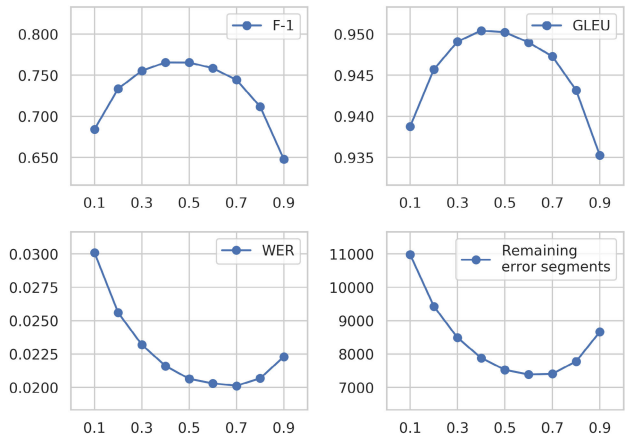

FIGURE 18. Detection and end-to-end correction performances at different detection sensitivities.

TABLE 7. Evaluation of end-to-end error correction on the GEC Conll-2014 dataset by various systems.

Model	M2	GLEU
Do nothing	0.0000	0.5663
Oracle	1.0000	0.8187
Literature		
Bi-GRU	0.4276	-
SMT + Bi-GRU	0.5625	-
Copy-Aug Transformer	0.5642	-
Copy-Aug Transformer + (A + F)	**0.6115**	-
Reproduced		
Bi-GRU without Lang 8	0.2158	0.5354
Bi-GRU	0.4288	**0.5931**
Ours	0.0195	0.5630

TABLE 8. Evaluation of end-to-end error correction on the misspelling subset of GEC Conll-2014 by various systems.

Model	Precision	Recall	M2	GLEU
Pre-corrected	**0.9913**	0.9048	**0.9727**	0.7520
Oracle	0.9917	0.9896	0.9900	0.7767
Bi-GRU	0.9266	0.8660	0.9138	0.7358
Bi-GRU (A**)	0.8926	0.8623	0.8863	0.7356
Copy-Aug (A*)	0.8066	0.8552	0.8159	0.7382
Copy-Aug (A* + F)	0.8351	0.8587	0.8397	0.7409
Ours	0.9184	0.8982	0.9143	0.7487
Ours (A**)	0.9073	0.9028	0.9064	0.7496
Ours (A** + F)	0.9567	**0.9082**	0.9466	**0.7539**
Ours (A** + F) + Oracle Detection	0.9436	0.9314	0.9411	0.7665

* Data augmentation from using the supplied pre-trained weights from [12].
** Data augmentation with injected character errors.

2013 and Conll-2014 datasets by precorrecting any grammatical errors and leaving only the spelling errors (corrections marked as ''Mec'' in the dataset). We tested both our model and Bi-GRU [11] under two training regimes: with and without data augmentation on the training set. For the copy-augmented transformer, we used the pretrained weights provided by the authors (which were trained on augmented data according to their paper [12]) and fine-tuned it on the training set. Our model outperformed both the Bi-GRU model and the copy-augmented transformer model with respect to both M2 and GLEU scores on the spelling correction task. However, due to the sparse nature of the misspelling errors in the test set (only 228 misspelled segments constituting only 9.54% of all 2,391 erroneous segments) spanning 1,312 sentences, the resulting corrected text from all the evaluated models received a lower M2 score than did the precorrected text used as the input. Only our model (with augmentation and fine-tuning) produced an increased GLEU score over the precorrected text.

VII. CONCLUSION

In this study, we investigated how various text correction systems and our proposed method performed on our Thai text correction (Thai TC) task.

Our investigation into the various techniques showed that most systems struggle when applied to Thai TC. Traditional two-stage Thai text correction systems, which rely on a dictionary-based corrector, suffer because they select improper candidates during the correction stage. As a result, these systems are unable to produce an output with error levels below those of the input. However, these results are in-line with the current use of these systems because spell checkers require human intervention to select the proper correction. On the other hand, the basic end-to-end correction systems (E2E) (i.e., Bi-GRU Seq2Seq [11]) suffer other issues when applied to Thai TC and perform much worse. However, moving to a more advanced E2E system (i.e., copy-augmented transformer [12]) showed that a Seq2Seq corrector with a substructure that encourages copying can enable a corrector to produce text corrections that are better than the original input text.

Our proposed model is a neural-based two-stage error correction system with a novel context-aware correction stage. We investigated how the detection stage affects the overall correction performance and how to tune the proposed text correction system for optimal correction performance. Our proposed system outperformed all the existing tested techniques on the TC task on the Thai UGWC dataset. Our proposed method also outperformed the state-of-the-art GEC model on an English spelling correction task constructed from the Conll2014 task [21].

In this work, we proposed a handcrafted procedure for performing data augmentation for model pretraining. In future work, a learning-based method for data augmentation (e.g., back-translation [17]) could further improve the correction performance.

APPENDIX A
ERROR ANALYSIS OF TEXT CORRECTION ON THAI UGWC

In this section, we perform error analyses on each of the results reported on our text correction task on the Thai

TABLE 9. Error analysis of each model on the first sample line on the TC task on the Thai UGWC dataset.

Line 1		
Source	“พอ|ผม|โหลด|แอพ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|คับ”	
Annotation	“พอ|ผม|โหลด|แอป| |K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|ครับ”	
	From 3 To 4 “แอป| ”	misspelling
	From 16 To 17 “ครับ”	misspelling
Annotation-2	“พอ|ผม|ดาวน์โหลด|แอป| |K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|ครับ”	
	From 2 To 4 “ดาวน์โหลด|แอป| ”	slang, misspelling
	From 16 To 17 “ครับ”	misspelling”
Ours	“พอ|ผม|โหลด|แอป| |K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อะ|ครับ”	
	From 3 To 4 “แอป| ”	
	From 15 To 17 “อะ|ครับ”	over corrected: “Other” sound correction
Hunspell	“พอ|ผม|โหลด|แอ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อะ|คับ”	
	From 3 To 4 “แอ”	incorrect
	From 15 To 16 “อะ”	over corrected: “Other” sound correction
Hunspell (Trained)	“พอ|ผม|โหลด|แอพ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|คับ”	
		no correction: same as “Source”
PyThaiNLP	“พอ|ผม|โหลด|แอพ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|คับ”	
		no correction: same as “Source”
Bi-GRU (20 Token)	“พอ|ผม|โหลด|แอป| |K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|ครับ”	
	From 3 To 4 “แอป| ”	
	From 16 To 17 “ครับ”	
Bi-GRU (50 Token)	“พอ|ผม|โหลด|แอพ|K| |Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อะ|ครับ”	
	From 3 To 5 “แอพ|K| |Bank”	incorrect, space token deleted
	From 15 To 17 “อะ|ครับ”	over correction: “Other” sound correction
Bi-GRU (untokenized)	“พอ|ผม|โหลด|แอพ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อะ|ครับ”	
	From 15 To 17 “อะ|ครับ”	over corrected: “Other” sound correction
Copy-Augmented	“พอ|ผม|โหลด|แอป|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อะ|ครับ”	
		same as “Ours”
	From 3 To 4 “แอป| ”	
	From 15 To 17 “อะ|ครับ”	over corrected: “Other” sound correction

user-generated web content (text data collected from social media) described in Table 3. We selected two lines (see Tables 9 and 10) to illustrate the types of issues each correction system struggled with. ‘‘Annotation’’ is used to denote the test set, while ‘‘Annotation-2’’ denotes another annotation by our linguist at KLabs. For the bidirectional GRU (Bi-GRU) model [11] on untokenized text, where the model operated on the SentencePiece tokens [27], the results are hand tokenized in favor of the model (leading to the lowest amount of errors). For models with multiple configurations, only the best configuration is analyzed. That is, our pretrained and fine-tuned model with words as the unit tokens, the Bi-GRU model with 50-token limits, the Bi-GRU model with 20-token limits, the Bi-GRU model with 40-token limits (untokenized), and the pretrained and fine-tuned copy-augmented transformer.

Error correction systems with dictionary-based correction (i.e., Hunspell, Hunspell (trained), PyThaiNLP), struggle with choosing the proper correction even when the target word is in the dictionary. Thus, the system with the most conservative detection stage (i.e., PyThaiNLP) performs the best by introducing the fewest number of corrections. While Hunspell with the provided dictionary corrects some error segments correctly, the introduced errors outweigh the effect of the corrections made.

End-to-end correction systems (i.e., Bi-GRU, copy-augmented) often produce correct Thai sentences but with a different meaning. An analysis of the output suggested that the model prefers to produce sentences or phrases that are common in the training dataset. While the copy-augmented transfer’s substructure should alleviate this issue, the model still suffers from this issue, but to a lesser extent when compared to Bi-GRU.

APPENDIX B
HYPERPARAMETERS

On the Thai text correction task, we obtained the hyperparameter values for both our method and the other neural-based methods evaluated in this paper (i.e., Bi-GRU [11] and copy-augmented transformer [38]) using grid-search for optimal performance. Table 11 shows a consolidated list of the hyperparameters for our method. The search range for each component of our error detector was 32-512 with multiple-of-2 increments. For our corrector, the search

TABLE 10. Error analysis of each model on the second sample line on the TC task on the Thai UGWC dataset.

Line 2			
Source	"เช่น|โอน|เข้า|เวบ|นี้|เพือ|เข้า|ใป|เล่น|ทัมอย่างใร|คับ"		
Annotation	"เช่น|โอน|เข้า|เว็บไซต์|นี้|เพื่อ|เข้า|ไป|เล่น|ทำ|อย่างไร|ครับ"		
	From 3 To 4	"เว็บไซต์"	misspelling (full word)
	From 5 To 6	"เพื่อ"	misspelling
	From 7 To 8	"ไป"	misspelling
	From 9 To 11	"ทำ|อย่างไร|ครับ"	misspelling
Annotation-2	"เช่น|โอน|เข้า|เว็บ|นี้|เพื่อ|เข้า|ไป|เล่น|ทำ|อย่างไร|ครับ"		
	From 3 To 4	"เว็บ"	misspelling
	From 5 To 6	"เพื่อ"	misspelling
	From 7 To 8	"ไป"	misspelling
	From 9 To 11	"ทำ|อย่างไร|ครับ"	misspelling
Ours	"เช่น|โอน|เข้า|เว็บไซต์|นี้|เพื่อ|เข้า|ไป|เล่น|ทำ|อย่างไร|ครับ"		
			same as "Annotation"
	From 3 To 4	"เว็บไซต์"	
	From 5 To 6	"เพื่อ"	
	From 7 To 8	"ไป"	
	From 9 To 11	"ทำ|อย่างไร|ครับ"	
Hunspell	"เช่น|โอน|เข้า|เว็บ|นี้|เพื่อ|เข้า|ใบปก|เล่น|อย่างทรมาน|คับ"		
	From 3 To 4	"เว็บ"	
	From 5 To 6	"เพื่อ"	
	From 7 To 8	"ใบปก"	incorrect
	From 9 To 10	"อย่างทรมาน"	incorrect
Hunspell (Trained)	"เช่น|โอน|เข้า|เวบ|นี้|เพอ|เข้า|ใป|เล่น|ทัพอย่าง|คับ"		
	From 5 To 6	"เพอ"	incorrect
	From 9 To 10	"ทัพอย่าง"	incorrect
PyThaiNLP	"เช่น|โอน|เข้า|เวบ|นี้|เพือ|เข้า|ใป|เล่น|ทัมอย่างใร|คับ"		
Bi-GRU (20 Token)	"เช่น|โอน|เข้า|เวบ|นี้|เพื่อ|เข้า|ไป|เล่น|ทำ|อย่างไร|ครับ"		
	From 5 To 6	"เพื่อ"	
	From 7 To 8	"ไป"	
	From 9 To 11	"ทำ|อย่างไร|ครับ"	
Bi-GRU (50 Token)	"เช่น|โอน|เข้า|เวบ|นี้|เพื่อ|เข้า|ไป|เล่น|ทำ|อย่างไร|ครับ"		
			same as "Bi-GRU (20 Token)"
	From 5 To 6	"เพื่อ"	
	From 7 To 8	"ไป"	
	From 9 To 11	"ทำ|อย่างไร|ครับ"	
Bi-GRU (untokenized)	"เช่น|โอน|เข้า|บัญชี|นี้|ครับ"		
	From 3 To 4	"บัญชี"	over corrected (meaning lost)
	From 5 To 11	"ครับ"	over corrected (meaning lost)
Copy-Augmented	"เช่น|โอน|เข้า|เว็บ|นี้|เพื่อ|เข้า|ไป|เล่น|ครับ"		
	From 3 To 4	"เว็บ"	
	From 5 To 6	"เพื่อ"	
	From 7 To 8	"ไป"	
	From 9 To 11	"ครับ"	over corrected (words dropped)

TABLE 11. Consolidated list of hyperparameters for our proposed method.

Component	Configuration
Error Detector	
Vocabulary	24576 tokens
Word embeddings layer	64 nodes
Character embeddings layer	128 nodes
Character bi-LSTM Encoder	32 nodes (in each direction)
Bi-LSTM Encoder	2 layers × 32 nodes (in each direction)
Bi-LSTM Encoder dropout	0.5
Batch size	32
Optimizer	Adam [28]
Learning-rate	0.002
Fine-tuning learning-rate	0.0005
Early stopping patience	20 epochs
Error Corrector	
Vocabulary	24576 tokens
m	24
n	128
Erroneous encoding matrix ($E^{(l)}$) dropout	0.5
Context encoding matrix ($C^{(l)}$) dropout	0.5
Batch size	32
Optimizer	Adam [28]
Learning-rate	0.002
Fine-tuning learning-rate	0.0005
Early stopping patience	20 groups*

* Error Corrector training procedure are detailed further in Section IV-C.

ranges for "m" and "n" are 16-40 with increments of 8 and 64-256 with multiples of 2 increments, respectively. On the English spelling correction task, our hyperparameters remain the same as on the Thai TC task, while the hyperparameters for other methods were set according to their original papers (i.e., Bi-GRU [11] and copy-augmented transformer [38]).

ACKNOWLEDGMENTS

This work was supported in part by the joint research of Kasikorn Business Technology Group (KBTG) and the Faculty of Engineering, Chulalongkorn University. The authors would like to thank the linguist at KLabs, Nutcha Tirasaroj, for her help with the error analysis.

REFERENCES

[1] K. Lee, A. Agrawal, and A. Choudhary, "Real-time disease surveillance using Twitter data: Demonstration on flu and cancer," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery data Mining*, 2013, pp. 1474–1477.
[2] A. Farzindar and D. Inkpen, "Natural language processing for social media," *Synth. Lectures Hum. Lang. Technol.*, vol. 8, no. 2, p. 9, 2015.
[3] A. Lertpiya, T. Chaiwachirasak, N. Maharattanamalai, T. Lapjaturapit, T. Chalothorn, N. Tirasaroj, and E. Chuangsuwanich, "A preliminary study on fundamental Thai NLP tasks for user-generated Web content," in *Proc. Int. Joint Symp. Artif. Intell. Natural Lang. Process. (iSAI-NLP)*, Nov. 2018, pp. 1–8.
[4] S. Meknavin, B. Kijsirikul, A. Chotimongkol, and C. Nuttee, "Progress of combining trigram and winnow in Thai OCR error correction," in *Proc. IEEE. APCCAS . IEEE Asia–Pacific Conf. Circuits Syst. Microelectron. Integrating Systems. Proc.*, 1998, pp. 555–558.
[5] S. Meknavin, B. Kijsirikul, A. Chotimongkol, and C. Nuttee, "Combining trigram and winnow in Thai OCR error correction," in *Proc. 17th Int. Conf. Comput. Linguistics*, 1998, pp. 836–842.
[6] M. Rodphon, K. Siriboon, and B. Kruatrachue, "Thai OCR error correction using token passing algorithm," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process.*, 2001, pp. 599–602.
[7] B. Kruatrachue, K. Somguntar, and K. Siriboon, "Thai OCR error correction using genetic algorithm," in *Proc. 1st Int. Symp. Cyber Worlds*, 2002, pp. 137–141.
[8] S. Watcharabutsarakham, "Spell checker for Thai document," in *Proc. IEEE Region Conf.*, Nov. 2005, pp. 1–4.
[9] Hunspell. (Nov. 2019). *Hunspell/Hunspell*. [Online]. Available: https://github.com/hunspell/hunspell
[10] PyThaiNLP. (May 2019). *PyThaiNLP/Spelling-Check*. [Online]. Available: https://github.com/PyThaiNLP/spelling-check
[11] R. Grundkiewicz and M. Junczys-Dowmunt, "Near human-level performance in grammatical error correction with hybrid machine translation," 2018, *arXiv:1804.05945*. [Online]. Available: http://arxiv.org/abs/1804.05945
[12] W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu, "Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data," 2019, *arXiv:1903.00138*. [Online]. Available: http://arxiv.org/abs/1903.00138
[13] PyThaiNLP. (Nov. 2019). *PyThaiNLP/pythainlp*. [Online]. Available: https://github.com/PyThaiNLP/pythainlp
[14] S. Chollampatt and H. T. Ng, "A multilayer convolutional encoder-decoder neural network for grammatical error correction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5755–5762.
[15] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, "Approaching neural grammatical error correction as a low-resource machine translation task," 2018, *arXiv:1804.05940*. [Online]. Available: http://arxiv.org/abs/1804.05940
[16] S. Chollampatt and H. T. Ng, "Neural quality estimation of grammatical error correction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2528–2539.
[17] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. Inui, "An empirical study of incorporating pseudo data into grammatical error correction," 2019, *arXiv:1909.00502*. [Online]. Available: http://arxiv.org/abs/1909.00502
[18] wolfgarbe. (Nov. 2019). *Wolfgarbe/SymSpell*. [Online]. Available: https://github.com/wolfgarbe/SymSpell
[19] K. Atkinson. (2019). *GNU Aspell*. [Online]. Available: http://aspell.net/
[20] N. Zukarnain, B. S. Abbas, S. Wayan, A. Trisetyarso, and C. H. Kang, "Spelling checker algorithm methods for many languages," in *Proc. Int. Conf. Inf. Manage. Technol. (ICIMTech)*, Aug. 2019, pp. 198–201.
[21] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," in *Proc. 18th Conf. Comput. Natural Lang. Learn., Shared Task*, 2014, pp. 1–14.
[22] A. Rozovskaya and D. Roth, "Grammatical error correction: Machine translation and classifiers," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistic*, Berlin, Germany, Aug. 2016, pp. 2205–2215. [Online]. Available: https://www.aclweb.org/anthology/P16-1208
[23] M. Junczys-Dowmunt and R. Grundkiewicz, "Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin, TX, USA, Nov. 2016, pp. 1546–1556. [Online]. Available: https://www.aclweb.org/anthology/D16-1161
[24] *The Royla Inst. Dictionary 2542 B.E.* Nanmeebooks, Bangkok, Thailand, 1999.
[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[26] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.
[27] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," 2018, *arXiv:1808.06226*. [Online]. Available: http://arxiv.org/abs/1808.06226
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[29] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2016, *arXiv:1611.01603*. [Online]. Available: http://arxiv.org/abs/1611.01603
[30] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer, "AllenNLP: A deep semantic natural language processing platform," 2017, *arXiv:1803.07640*. [Online]. Available: https://arxiv.org/abs/1803.07640
[31] AllenNLP. (Feb. 2019). *AllenNLP Bidirectional Attention Flow Implementation*. [Online]. Available: https://github.com/allenai/allennlp/blob/v0.8.2/allennlp/models/reading_comprehension
[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[33] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "Ground Truth for Grammatical Error Correction Metrics," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics*, Beijing, China, Jul. 2015, pp. 588–593. [Online]. Available: https://www.aclweb.org/anthology/P15-2097
[34] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "GLEU without tuning," 2016, *arXiv:1605.02592*. [Online]. Available: http://arxiv.org/abs/1605.02592
[35] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics*, 2001, pp. 311–318. [Online]. Available: https://www.aclweb.org/anthology/P02-1040
[36] R. Grundkiewicz, M. Junczys-Dowmunt, and E. Gillian, "Human evaluation of grammatical error correction systems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 568–572. [Online]. Available: http://dl.acm.org/citation.cfm?id=2382029.238211
[37] W. Aroonmanakun, "Thoughts on word and sentence segmentation in Thai," in *Proc. 7th Symp. Natural Lang. Process.*, Pattaya, Thailand, Dec. 2007, pp. 85–90.
[38] P. Norvig. (Feb. 2007). *How to Write a Spelling Corrector*. [Online]. Available: http://norvig.com/spell-correct.html
[39] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.
[40] D. Dahlmeier, H. T. Ng, and S. M. Wu, "Building a large annotated corpus of learner English: The NUS corpus of learner English," in *Proc. 8th Workshop Innov.*, Atlanta, Georgia, Jun. 2013, pp. 22–31. [Online]. Available: https://www.aclweb.org/anthology/W13-1703
[41] T. Mizumoto, Y. Hayashibe, M. Komachi, M. Nagata, and Y. Matsumoto, "The effect of learner corpus size in grammatical error correction of ESL writings," in *Proc. COLING*, 2012, pp. 863–872.

ANURUTH LERTPIYA received the B.Eng. degree in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 2018, where he is currently pursuing the M.Eng. degree in computer engineering. His research interests include natural language generation, information extraction, and named-entity recognition.

TAWUNRAT CHALOTHORN received the B.S., M.S., and Ph.D. degrees from the University of Northumbria, Newcastle, U.K., in 2010, 2011, and 2016, respectively. She then joined the Natural Language Processing Team, Kasikorn Labs (KLabs) Company Ltd., Kasikorn Business Technology Group (KBTG), which mostly research based on Thai language. Her research interests include chatbots, social listening, and text analytics.

EKAPOL CHUANGSUWANICH received the B.S. and M.S. degrees in electrical and computer engineering from Carnegie Mellon University, in 2008 and 2009, respectively, and the Ph.D. degree from the MIT, in 2016. He then joined the Spoken Language Systems Group, MIT Computer Science and Artificial Intelligence Laboratory. He is currently a Faculty Member of the Department of Computer Engineering, Chulalongkorn University. His research interests include speech processing, assistive technology, and health applications.