Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention

Mr. Anuruth Lertpiya

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2019
Copyright of Chulalongkorn University

การแก้คำผิดและทำให้เป็นมาตราฐานบนข้อความโซเชียลมีเดียภาษาไทยโดยการทำงานสองขั้นตอน ด้วยโครงข่ายประสาทเทียมที่ใช้กลไกจุดสนใจบนบริบท

นายอนุรุธ เลิศปิยะ

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2562

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Thesis Title	Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention
By	Mr. Anuruth Lertpiya
Field of Study	Computer Engineering
Thesis Advisor	Ekapol Chuangsuwanich, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Engineering

.... Dean of the FACULTY OF ENGINEERING

(Professor SUPOT TEACHAVORASINSKUN, D.Eng.)

THESIS COMMITTEE

.... Chairman

(Assistant Professor PEERAPON VATEEKUL, Ph.D.)

.... Thesis Advisor

(Ekapol Chuangsuwanich, Ph.D.)

.... External Examiner

(Thadpong Pongthawornkamol, Ph.D.)

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

อนุรุธ เลิศปิยะ : การแก้คำผิดและทำให้เป็นมาตราฐานบนข้อความโซเชียลมีเดียภาษาไทยโดยการทำงานสองขั้นตอนด้วยโครงข่ายประสาทเทียมที่ใช้กลไกจุดสนใจบนบริบท. (Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention) อ.ที่ปรึกษาหลัก : ดร.เอกพล ช่วงสุวนิช

ระบบแก้ไขข้อความ (เช่นระบบแก้คำผิด) ถูกนำมาใช้เพื่อปรับปรุงคุณภาพของข้อมูลตัวอักษรบนระบบคอมพิวเตอร์โดยการตรวจจับและแก้ไขข้อผิดพลาด งานวิจัยก่อนหน้ายังไม่ได้รับการสำรวจโจทย์การแก้ไขคำผิดและการทำให้เป็นมาตรฐานของข้อความ (การแก้ไขข้อความ) สำหรับข้อความโซเชียลมีเดียภาษาไทย ในวิทยานิพนธ์ฉบับนี้เราได้ศึกษาความสามารถของระบบแก้ไขข้อความในปัจจุบันบนโจทย์การแก้ไขคำผิดและการทำให้เป็นมาตรฐานของข้อความ บนโซเชียลมีเดียภาษาไทย และ เสนอวิธีการที่ได้ถูกออกแบบมาสำหรับโจทย์นี้ เราพบว่าระบบแก้ไขข้อความภาษาไทยที่มีอยู่ในปัจจุบันมีประสิทธิภาพไม่เพียงพอสำหรับการแก้ไขคำผิดและความไม่เป็นมาตรฐานของข้อความ ในขณะที่ระบบแก้ไขข้อผิดพลาดทางไวยากรณ์ภาษาอังกฤษมีปัญหาการแก้ไขมากเกินไป (การเขียนข้อความใหม่) ดังนั้นเราจึงเสนอระบบแก้ไขข้อความ ซึ่งใช้ระบบประสาทเทียมที่งานสองขั้นตอนเพื่อบรรเทาปัญหาการแก้ไขมากเกินไปในขณะที่ได้ประโยชน์จากระบบประสาทเทียมแบบข้อความสู่ข้อความ ระบบของเราประกอบด้วยตัวตรวจจับข้อผิดพลาดที่ใช้ระบบประสาทเทียม และตัวแก้ไขข้อผิดพลาดทางประสาทแบบข้อความสู่ข้อความที่ใช้กลไกจุดสนใจบนบริบท สถาปัตยกรรมแบบใหม่นี้ช่วยให้ระบบประสาทเทียมแบบข้อความสู่ข้อความสร้างแก้ไขตามทั้งข้อความโดยคำนึงถึงบริบทโดยไม่จำเป็นต้องทำงานแบบหนึ่งขั้นตอนวิธีการของเรามีประสิทธิภาพดีกว่าระบบแก้ไขข้อความอื่นๆ ที่เราได้ประเมินทั้งหมด

สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต ....
ปีการศึกษา	2562	ลายมือชื่อ อ.ที่ปรึกษาหลัก ....

6170322321 : MAJOR COMPUTER ENGINEERING
KEYWORD: Spelling correction, Text generation, Text normalization, Thai language, Natural language processing, Artificial neural networks, Text processing, Machine learning

Anuruth Lertpiya : Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention. Advisor: Ekapol Chuangsuwanich, Ph.D.

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 thesis, 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.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Field of Study:	Computer Engineering	Student's Signature ....
Academic Year:	2019	Advisor's Signature ....

ACKNOWLEDGEMENTS

I want to thank the joint research venture between Kasikorn Business—Technology Group (KBTG) and the Faculty of Engineering, Chulalongkorn University. This study would not be possible without the funding, linguistic and technical capability from both parties. In addition, we would also like to thank the Faculty of Arts, Chulalongkorn University, for their linguistic expertise.

I would like to thank my supervisor, Dr. Ekapol Chuangsuwanich. Your invaluable advice, feedback, and encouragement are indispensable for the development of this thesis.

I am very honored to have Asst. Prof. Peerapon Vateekul and Dr. Thadpong Pongthawornkamol as the chairman and external examiner of my thesis committee. I also would like to thank them for spending their precious time to examine and critique my manuscript.

I would like to acknowledge the team of linguists from both the Faculty of Arts, Chulalongkorn University, and KBTG. First, I would like to thank Assoc. Prof. Wirote Aroonmanakun, Dr. Supawat Taerungruang, Ms. Sasiwimon Kalunsima, Dr. Tantong Champaiboon, Dr. Nutcha Tirasaroj, and language-major students for the development of the specification and annotation Thai User-generated web-content corpus.

I would also like my colleagues at KBTG for their guidance and support. Special thanks to my fellow friends at Chulalongkorn University, for their advice, aid, and encouragement.

Moreover, I would like to thank my parents for them for both their physical and mental support. Without them, my thesis would not be where it is at today.

Anuruth Lertpiya

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

TABLE OF CONTENTS

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

LIST OF TABLES

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

LIST OF FIGURES

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

1 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 (Lee, Agrawal, & Choudhary) 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 (Farzindar & Inkpen; Lertpiya et al.). 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 (Kruatrachue, Somguntar, & Siriboon; Meknavin, Kijsirikul, Chotimongkol, & Nuttee; Rodphon, Siriboon, & Kruatrachue; Watcharabutsarakham). 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.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

In this thesis, 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 Chapter 4.2. The contributions of this thesis 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

(Hunspell)) and statistically based methods (i.e., PyThaiNLP (PyThaiNLP)) to modern systems featuring sequence-to-sequence neural networks employed in state-of-the-art English GEC systems (i.e., Bi-GRU Seq2Seq (Grundkiewicz & Junczys-Dowmunt), Copy-Augmented Transformer (Zhao, Wang, Shen, Jia, & Liu)).

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 Chapter 4.2, relying solely on the Seq2Seq corrector can lead to overcorrections (text is rewritten as opposed to simply corrected).

1.1 Aim and Objectives

This study aims to learn how to effectively perform the spelling correction and word normalization task on Thai social media text (the Text Correction task). The objectives of this research are as follows:

- Evaluate how well publicly available tools and methods for text correction perform on the spelling correction and word normalization task on Thai social media text.
- Propose a novel method for performing the spelling correction and word normalization task on Thai social media text.
- Evaluate our method on the publicly available version of our Text Correction dataset.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

1.2 Contributions

This study focuses on the four contributions: evaluating the current literature on Thai text spelling correction and text normalization (text correction task), propose a method for the Thai text correction task, evaluate our proposed method on a multitude of datasets, and conduct error analysis to examine our model further.

- Study the currently existing tools and methods for text correction and adapt it to the spelling correction and word normalization task on Thai social media text.

    - The scope of text correction is limited to the five types of errors that naturally exist in Thai social media texts: misspelled words, morphed words, slangs, spoonerisms, and incorrect abbreviation notations.
- Design and implement a Text Correction method for the Thai social media text.
    - Explore the characteristic of our method for text correction and apply techniques to improve the performance.
- Compare our method with the currently existing tools and methods researched.
- Perform error analysis on our method as well as other text correction methods presented

1.3 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 overviews the background knowledge relating to this study. Chapter 3 discusses works relating to Thai spelling correction and text normalization (Thai TC). Chapter 4 outlines our TC tasks (spelling correction and word normalization) task on Thai UGWC as well as the development of our Thai UGWC dataset. Chapter 5 describes our proposed two-stage TC system for Thai UGWC. Chapter 6 discusses the results of other models we experimented with alongside those of our proposed method. Finally, Chapter 7 reiterates our contributions and concludes the thesis.

1.4 Publication

The methods and some of the results in this thesis have been previously published (Lertpiya, Chalothorn, & Chuangsuwanich).

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

2 Background knowledge

2.1 Introduction

This chapter covers the background knowledge relating to the Thai text correction task (Thai TC). The first section introduces the concept of spell checkers. The second section outlines the building blocks of deep learning models in natural language processing.

2.2 Spell Checkers & Misspellings

Spell checkers are defined as systems for identifying non-word errors (words that do not exist in the dictionary). In addition to identifying the errors, the spell checker often produces a sequence of correction candidates for the identified error. Where the candidates are ordered according to the probability of the candidate being the proper correction to the error, these spell checkers are sometimes referred to as spelling corrector.

Traditionally, spell checkers are relatively simple since user written text is matched against an internal dictionary. Thus, the task of building spell checkers is considered an engineering problem, where performance (wolfgarbe) or the performance-accuracy trade-off (Atkinson) 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) (Meknavin et al.; Watcharabutsarakham).

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 (Zukarnain, Abbas, Wayan, Trisetyarso, & Kang). Hunspell (Hunspell) (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).

2.3 Deep learning for Natural Language Processing

2.3.1 Deep learning

Deep learning (DL) is a sub-branch of machine learning (ML) where models are primarily based on deep neural networks (DNN), hence the name. The advantage of DNN stems primarily from representation learning. Whereas traditional ML methods use statistical methods to model the task on low-level features and handcrafted higher-level features, DL utilizes DNN to extract higher-level features automatically. At the cost of requiring more data and higher computation cost. For NLP, these two downsides are often offset by the availability of text data on the Internet and compute accelerators (e.g., graphic processing units).

2.3.2 Recurrent neural networks

Recurrent neural networks (RNNs) are a type of DNN where part of the input and outputs of the networks is connected to itself. RNNs are typically used for sequence modeling tasks, due to its ability to process a sequence of input of arbitrary lengths. RNNs have a variety of different structures, for example, Long-short-term-memory (LSTM) (Hochreiter & Schmidhuber), and Gated-Recurrent-Unit (GRU) (Cho et al.). These unique structures allow the RNN to handle sequence modeling tasks better. For example, the introduction of LSTM is to improve modeling tasks with long-term dependencies by reducing the issues of vanishing gradients, by not having non-linear functions across its long-term memory channel.

2.3.3 Sequence-to-Sequence neural networks

A sequence-to-sequence neural network (Seq2Seq) a neural network that takes a sequence of inputs and produces a sequence of outputs. Seq2Seq networks consisting of two main sub-structures: an encoder and a decoder. The encoder encodes the input sequence into some representation. Then that encoded representation is decoded by the decoder into the output sequence. Seq2Seq networks are employed tasks such as Machine Translation, Grammatical Error Correction.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

3 Related works

3.1 Introduction

In this chapter, we explore previous works related to our Text Correction (TC) task on Thai User-generated Web Content (UGWC). Similar to most Thai NLP task, works on Thai TC are often adaptations of existing works on other languages (i.e., English). Thus, it is quite useful is to understand the overall trends of text correction systems, as well as to know the current state of both Thai and English. This chapter is split into three parts: an overview of text correction systems, the Thai text correction literature, and the English grammatical error correction literature.

3.2 Text Correction Systems for Natural Language

In the context of this study, we will separate text correction systems in two types: two-stage correction systems, and end-to-end correction systems.

Two-stage systems separate the text correction task into two phases: error detection (detector) and error correction (corrector). The primary rationale of this specific structure is to reduce the computational cost of performing correction by first identifying potentially erroneous areas of the text (Meknavin et al.). For example, the detector in dictionary-based systems (Hunspell; PyThaiNLP) classifies whether a token is an error by searching its dictionary. The reliance on prebuilt dictionaries limits the detectable errors to non-word errors only (words not in the dictionary). More statically complex models have been proposed to achieve better error detection (Lertpiya et al.; Meknavin et al.; PyThaiNLP; Watcharabutsarakham). 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 since 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. However, Thai text correction systems only utilize traditional LM implemented using tri-grams (Meknavin et al.).

End-to-end systems (E2E) combine the detection and correction stages into a single step. As such, the corrector produces a correction for every word in the input. In the event where the word is already correct, the corrector is expected to output the same token. A popular approach

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

in modern text correction literature (i.e., English grammatical error correction task) is 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. Techniques from statistical machine translation (Grundkiewicz & Junczys-Dowmunt), and subsequently, neural machine translation (NMT) (Chollampatt & Ng; Grundkiewicz & Junczys-Dowmunt; Junczys-Dowmunt, Grundkiewicz, Guha, & Heafield) have been employed with great success compared to the traditional two-stage systems. Then more specialized architectures (Chollampatt & Ng; Zhao et al.) and techniques for data augmentation and training (Kiyono, Suzuki, Mita, Mizumoto, & Inui; Zhao et al.) emerged later.

3.3 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 character recognition (OCR) systems (Kruatrachue et al.; Meknavin et al.; Rodphon et al.; Watcharabutsarakham). In contrast, text correction for text input via human-computer interfaces (HCIs), such as keyboards, is an underresearched area (Zukarnain et al., 2019). FOSS text correctors (e.g., Aspell (Atkinson), Hunspell (Hunspell), PyThaiNLP (PyThaiNLP)) 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 (Hunspell), character-gram (Lertpiya et al.; Watcharabutsarakham), WinNow (Meknavin et al.), and conditional random fields (CRF) (PyThaiNLP). However, the works on correction models include only dictionary-based (Hunspell) and statistical language models using part-of-speech (POS) trigrams (Meknavin et al.). The current statistical methods used in error detectors cannot detect errors that require extended context (Watcharabutsarakham, 2005) or require manual feature engineering to address out-of-vocabulary tokens (Meknavin et al.).

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

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

3.4 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 (Ng et al.), 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 (referred to as "classifier systems" in the GEC literature) to end-to-end systems (Junczys-Dowmunt & Grundkiewicz; Rozovskaya & Roth).

3.5 Conclusion

In this chapter, we covered works relating to the Thai TC task. We overviewed the text correction systems in terms of architectures (i.e., two-stage correction systems, and end-to-end correction systems) and research communities (i.e., Thai Text Correction, and English Grammatical Error Correction).

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

4 Thai Text Correction task

4.1 Introduction

This chapter describes the Thai Text Correction (Thai TC) task, our user-generated web content (UGWC) dataset, as well as our experimental setup for evaluating various methods on the Thai TC task. The text task presented in this chapter was previously published (Lertpiya et al.).

The goal of TC systems is to correct errors in the input text without altering the meaning. In the scope of this research, we are only interested in correctable errors in UGWC. Details on the different types of errors that occur in UGWC (and which are correctable) are covered in Chapter 4.2. In our task, errors are defined as words not in The Royal Institute Dictionary (Society) 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 suffers 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).

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

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.

In this chapter, we will outline our UGWC dataset, and the experimental setup used to evaluate a variety of methods (including ours detailed in Chapter 5) on the Thai Text Correction (Thai TC) Task. The first section details our UGWC dataset. The second section outlines the evaluation metrics used for each experiment. The third section outlines the experiments performed on our Thai TC task. The results of each experiment are later detailed in Chapter 6.

For experimentations on the publicly released version of the Thai TC task as well as Conll-2014 (Ng et al.), see Chapter 6.8.

4.2 UGWC dataset

Our UGWC dataset is an expanded version of our previous UGWC dataset (Lertpiya et al.) 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 1. 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 before data annotation. Real-world examples of each type of error and the respective corrections are shown in Table 2. The approximate number of errors in the training set is shown in Table 3. The numbers are an approximation since the word tonkenizer may produce in accurate tokenization due to the errors present. Meanwhile, error segments merges multiple consegutive errors into one. Explanations of each type of error and their English equivalents are outlined below.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

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.

Table 1 Size of the UGWC dataset and the training-testing split

	Tokens	*Error tokens*	*Error Segments*	*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

Table 2 Examples of different types of errors and their respective corrections

Type of Error	*Error*	*Correction*	*Type of Error*	*Error*	*Correction*
Misspelling	ทุ๊กคน	ทุกคน	Morphed	ครัช	ครับ
	คว่ำบัตร	คว่ำบาตร		ตั๊ลล๊าคคคค	น่ารัก
Abbreviation	มค	ม.ค.	Spoonerism	พับกบ	พบกับ
	พน	พรุ่งนี้			
Slangs	ตีเนียน	No Correction	Other	โรบินสัน	No Correction
	อ่อย	ทอดสะพาน			

Table 3 Approximate number of errors in UGWC training set

	Error tokens	*Error Segments*	*Error Segments (%)*
Misspelling	133,045	102,048	59.5%
Morphed	58,857	44,911	26.2%
Abbreviation	33,580	20,022	11.7%
Spoonerism	318	200	0.1%
Slangs	122	91	0.1%
Other	10,482	4,216	2.5%

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

Slangs 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”).

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”.

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”).

“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 1.

4.3 Evaluation criteria

Word-error-rate (WER) and generalized language evaluation understanding (GLEU) (Napoles, Sakaguchi, Post, & Tetreault) were adopted as the metrics for the TC task. WER is the standard evaluation metric used in past literature on Thai TC (Meknavin et al., 1998a, 1998b). GLEU (Napoles et al., 2015, 2016) was developed as an evaluation metric for English GEC and has a high correlation with human preference by extending BLEU (Papineni, Roukos, Ward, & Zhu). 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 (Chollampatt & Ng; Grundkiewicz & Junczys-Dowmunt; Junczys-Dowmunt et al.; Kiyono et al.; Ng et al.; Zhao et al.). Our initial goal was to adopt both correction metrics from English GEC for our Thai TC task. However, we dropped M2 (Dahlmeier & Ng) due to a combination of M2's high computational complexity and the Thai language's lack of explicit sentence boundaries (Aroonmanakun). We found that a single paragraph of text can take upwards of an hour to evaluate.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

4.4 Experiment setup

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 (Grundkiewicz & Junczys-Dowmunt) and the copy-augmented transformer (Norvig)), 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 9.2.

Hunspell (Hunspell) was evaluated using both the provided prebuilt Thai dictionary and a dictionary constructed from the training data. The constructed dictionary was 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 (Crammer, Dekel, Keshet, Shalev-Shwartz, & Singer) and a Norvig corrector (Norvig).

Two neural sequence-to-sequence models were evaluated: the bidirectional GRU (Bi-GRU) network (Grundkiewicz & Junczys-Dowmunt) and the copy-augmented transformer (Zhao et al.). 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) (Kudo & Richardson), 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).

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

4.5 Chapter summary

In this chapter, we covered the Thai Text Correction (Thai TC) task, from the details of our dataset to the experimental setup used to evaluate various methods on the Thai TC task.

Our dataset is built from UGWC (social text) containing mainly five types of errors and variances (i.e., misspellings, morphed words, slangs, spoonerisms, and incorrection abbreviation annotations). We have chosen two evaluation metrics for the Thai TC task: word-error-rate (WER) and generalized language evaluation understanding (GLEU). Then, we detailed our experimental setup for evaluating models on Thai TC.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

5 Method

5.1 Introduction

This chapter outlines our proposed two-stage TC method for Thai TC. The method presented in this chapter was previously published in (Lertpiya et al.). This chapter 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.

5.2 Model

This chapter describes the two-stage corrector: the error detection stage and the error correction stage. A structural overview of the entire pipeline is shown in Figure 1. The “ ” (space) character between two words in the error segment is added for visual clarity. “<BEGIN>” and “<END>” tokens are omitted to reduce clutter. The inputs and outputs of each stage and the details of the models are outlined below.

The input to the error detection stage is a sequence of words containing potentially erroneous input text $\vec{w} = \{w_1, w_2, \dots, w_N\}$, 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 $\vec{l} = \{l_1, l_2, \dots, l_N\}$, 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 Chapter 4.2.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

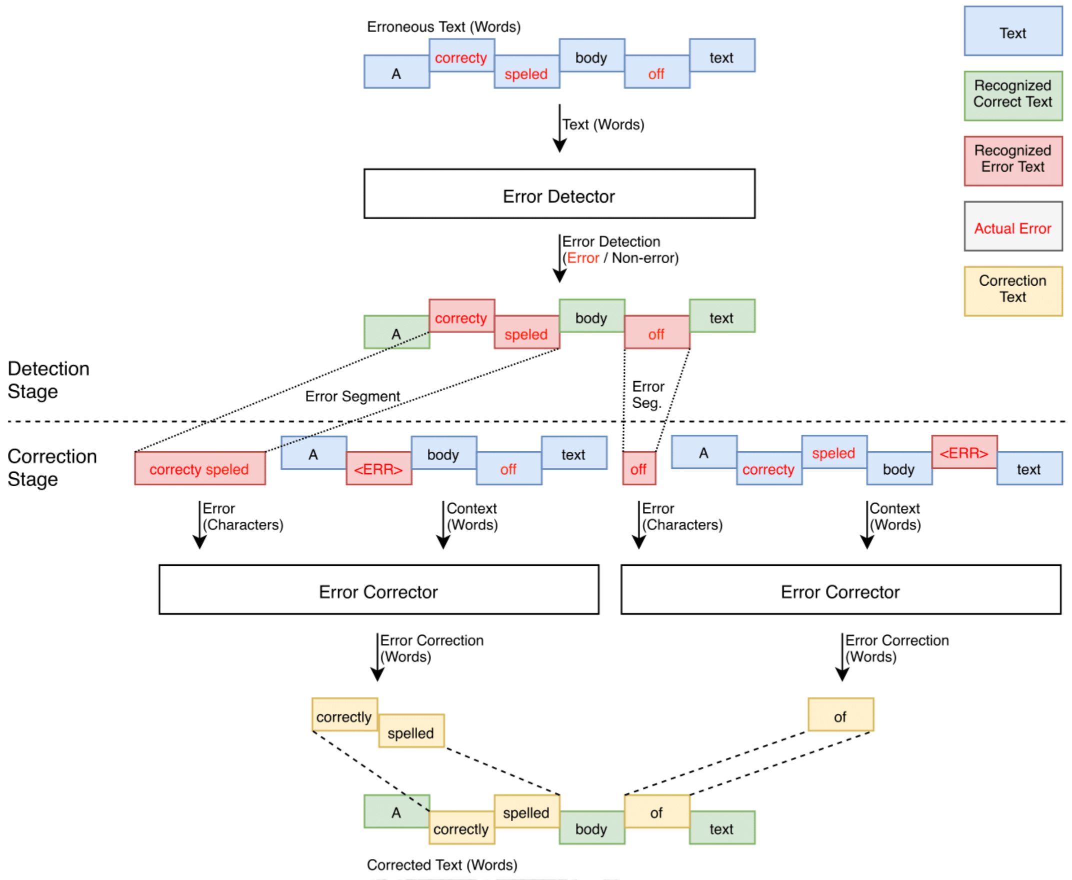


Figure 1 Overview of our text correction system

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 $\overrightarrow{w^*} = \{w_1^*, w_2^*, \ldots, w_M^*\}$ 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 “p | ั้ง | ใช้ | ไม่ | ได้ | อีก | หรอ | ครับ” where the 1st, 2nd, and 7th words labeled as erroneous, the correction stage would extract two error segments: “p | ั้ง” and “หรอ”. Given the correction “ยัง” and “หรือ”, the correction stage will produce the corrected sequence “ยัง | ใช้ | ไม่ | ได้ | อีก | หรือ | ครับ”

The error correction stage is given the same sequence of words containing erroneous text and the prediction from the detection stage. The error correction stage then produces a new sequence $\overrightarrow{w^*}$ where every word w_i marked as correct $l_i = correct$ is left unaltered. The error correction stage does this by extracting error segments from the detection data. An error segment is a contiguous sequence marked as errors by the detector. The text corrector produces a correction based on the erroneous text and the context text the error. For example, extracting from this sequence “p | ั ง | ใช้ | ไม่ | ได้ | อีก | หรอ | ครับ” where the 1st, 2nd, and 7th words are erroneous, would result in two error segments corresponding to “p | ั ง” and “หรอ” respectively. And the appropriate correction would be “ยัง” and “” which will produce the corrected sequence “ยัง | ใช้ | ไม่ | ได้ | อีก | หรือ | ครับ”. By grouping up the errors into segments and using a corrector that produces a sequence of words, our method avoids the issue where the erroneous word in the original text does not have a one-to-one mapping to the words in the correction.

5.2.1 Error Detector

Our error detector is a bidirectional-LSTM (bi-LSTM) (Hochreiter & Schmidhuber; Schuster & Paliwal) binary sequence tagger. An illustration of the detector is shown in Figure 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 Figure 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 Chapter 6.6). 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 subword units to handle OOV by evaluating our model with SentencePiece tokens (Kudo & Richardson) 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 (Kingma & Ba) with a learning-rate of 0.002 on the cross-entropy loss. See Appendix 9.2 for a consolidated list of hyperparameters.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

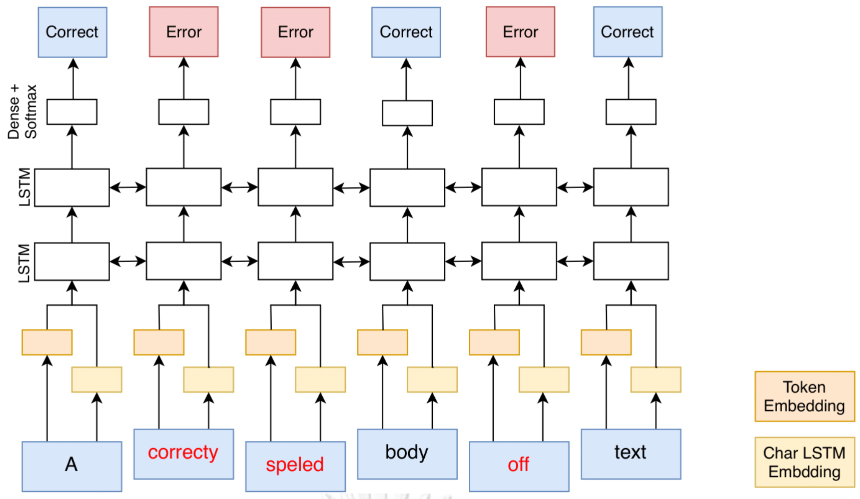


Figure 2 Error detector operating on a sequence

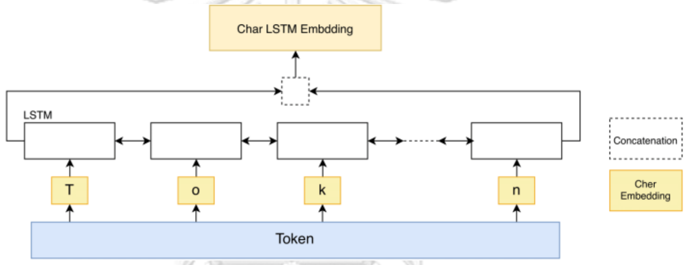


Figure 3 Character LSTM embedding layer encoding a word token

5.2.2 Error Corrector

Our proposed error corrector is an autoregressive sequence-to-sequence (Seq2Seq) neural network. An illustration of the overall structure is shown in Figure 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 thesis. 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 (Kingma & Ba) with a learning rate of 0.002 on the cross-entropy loss. The main hyperparameters are $m = 24$ and $n = 128$. See Appendix 9.2 for a consolidated list of hyperparameters.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

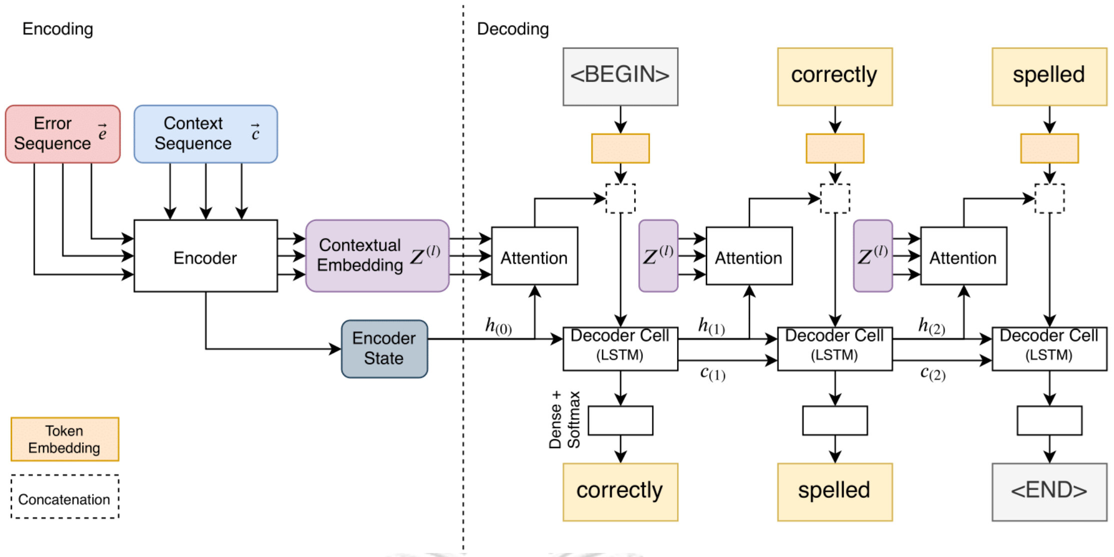


Figure 4 Structure of the Seq2Seq text corrector

5.2.2.1 Encoder

The context-aware encoder is composed of 2 embedding layers, 3 bi-LSTM encoders, and a contextual attention layer, as illustrated in Figure 5. The contextual attention layer allows the corrector to encode both the erroneous sequence and the 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 (Seo, Kembhavi, Farhadi, & Hajishirzi). 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 (Gardner et al.) and the BiDAF implementation in AllenNLP (AllenNLP) as a reference. The details of each layer of our context-aware encoder are described below.

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 $E^{(e)} \in R^{2m \times J}$. The character embeddings are then encoded by the “error encoder” (a bidirectional LSTM with m nodes in each direction) into an erroneous-encoding matrix $E^{(l)} \in R^{2m \times J}$.

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

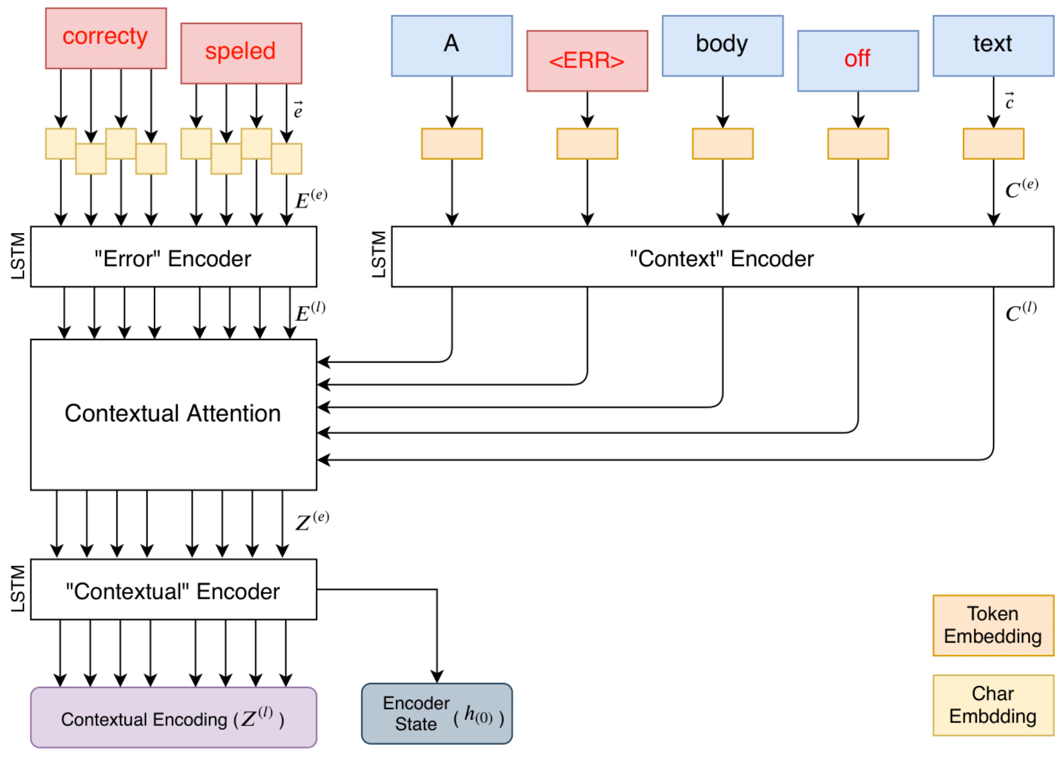


Figure 5 Structure of the encoder

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

$$
\begin{aligned}
&S \in R^{K\times J} && s_{kj} = c_k^{(l)T} \cdot e_j^{(l)} \in R \\
&A \in R^{K\times J} && a_j = softmax(s_j) \in R^K \\
&X^{(e)} \in R^{2m\times J} && x_j = \left(C^{(l)T} \cdot a_j\right)^T \in R^{2m} \qquad (1) \\
&Z^{(e)} \in R^{4m\times} && z_j^{(e)} = \left[e_j^{(l)}; x_j\right] \in R^{4m} \qquad (2)
\end{aligned}
$$

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

5.2.2.2 Decoder

The decoder is a typical LSTM decoder (a unidirectional LSTM containing 2n nodes) with an attention mechanism (Bahdanau, Cho, & Bengio) that observes contextual encoding, as shown in Figure 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.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

$$\overrightarrow{w^*} = \{BEGIN, w_1^*, w_2^*, \dots, w_L^*, END\} \qquad (3)$$

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 $h_0 \in R^{2n}$ 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 $e_t^{(e)} \in R^{2n}$. The context vector $e_t^{(c)} \in R^{2n}$ 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.

$$e_t = \left[e_t^{(e)}; e_t^{(c)}\right] \in R^{4n} \tag{4}$$

$$h_t, c_t = LSTM_{decoder}(h_{t-1}, c_{t-1}, e_t)$$

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.

5.3 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 (Zhao et al.). 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 at 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.

5.4 Training

This chapter 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 section under Chapter 5.2.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

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 thesis. The details of each set for UGWC are covered in Chapter 4.2, 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.

5.5 Chapter summary

In this chapter, we covered our proposed method of performing Thai TC on UGWC data. Our proposed method is a two-stage corrector, for additional details of the different types of text correctors see Chapter 3.2. A two-stage corrector is comprised of a detector and a corrector. The detector is responsible for identifying the erroneous portions of the text, which is then corrected by the correction stage. Unlike traditional two-stage correctors, our correction stage features the contextual attention layer, which allows the corrector to produce the correction based on both the erroneous portion of the text and the surrounding text.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

6 Results and Discussion

6.1 Introduction

In this chapter, we outline and discuss the results of our experiments, detailed in Chapter 4.4. First, Text Correction approaches were evaluated on the Thai UGWC. Second, we further explore how our method performs with SentencePiece tokens. Third, each stage of our proposed pipeline (i.e., detection stage, correction stage) is evaluated in isolation. Fourth, we investigate how detection sensitivity should be tuned for optimal performance. Fifth, we examine how iterative correction affects the results of our method. Lastly, we will evaluate our method on the publicly available version of our Thai UGWC dataset. The results presented from Chapter 6.2 to 6.6 were previously published in (Lertpiya et al.).

6.2 Thai UGWC

The TC results on the Thai UGWC dataset are shown in Table 4. We categorized the results into two groups: 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 (Hunspell), PyThaiNLP (PyThaiNLP). Furthermore, we evaluated three trained models: Hunspell (dictionary-based) (Hunspell), Bi-GRU (Neural Seq2Seq) (Grundkiewicz & Junczys-Dowmunt), and the copy-augmented transformer (Neural Seq2Seq with Augmentation) (Zhao et al.). Samples of the corrections produced by the individual models are shown in Appendix 9.1.1. The time required by each model to perform inference on the test set is shown in Table 6. Below, we discuss the shortcomings of the methods that struggled with the Thai TC task before reporting the overall results.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Correction systems with dictionary-based correctors (i.e., Hunspell (Hunspell), and PyThaiNLP (PyThaiNLP)) often struggle to select a correct correction candidate. As a result, a system with a more conservative error detector would 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 4.---the dictionary built from words in the corrected text from the training set ($frequency \geq 1$). 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 (Grundkiewicz & Junczys-Dowmunt) in Table 4). 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. 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 4, 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 9.1.1.

The copy-augmented transformer (Zhao et al.) 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 5 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.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

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.

6.3 SentencePiece as unit tokens

In this chapter, 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 7. 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 incorrect corrections. Due to false-positives in the detection stage, not correcting is the better option in such cases.

Table 4 Evaluation of end-to-end error correction on the Thai UGWC test set by various systems

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

* Model is only trained on noise injected dataset.

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

Table 5 Detailed WER evaluation of correction methods on the Thai UGWC test set

Model	*Substitution (%)*	*Deletion (%)*	*Insertion (%)*	*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***

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

Table 6 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 Intel i7-7800X. The GPU timing information is on an Nvidia Geforce GTX 1080 Ti. Inference is performed line-by-line (without line-level parallelism).				

Table 7 Evaluation of retokenized output from our method with different unit token types on the Thai UGWC test set

	GLEU	*WER (%)*	*∆WER (%)*
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 noise injected dataset before fine-tuned on the regular training			

6.4 Detection stage

This chapter 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. Figure 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 Figure 7. However, there is a trade-off between the detection coverage and the number of false-positive detections, as shown in Figure 8 and Figure 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 Figure 10 and Figure 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.

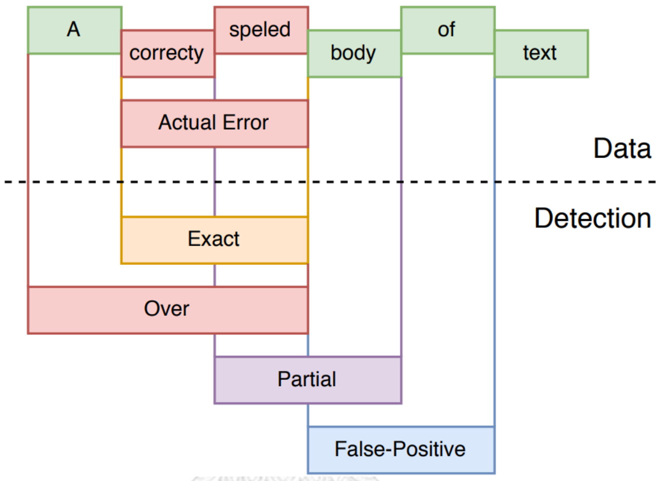


Figure 6 Four types of error segments

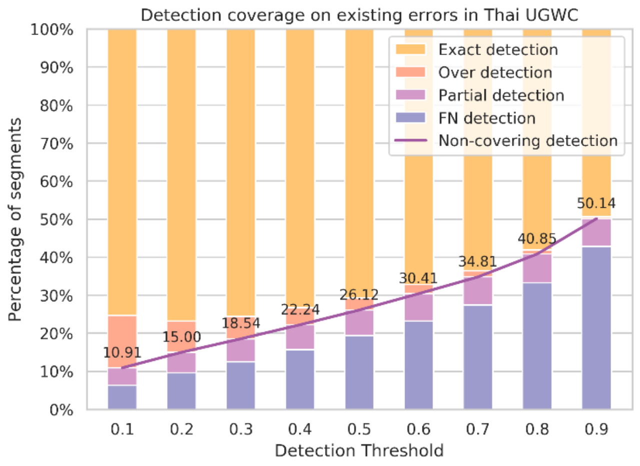


Figure 7 Detection coverage of our method on the Thai UGWC test set

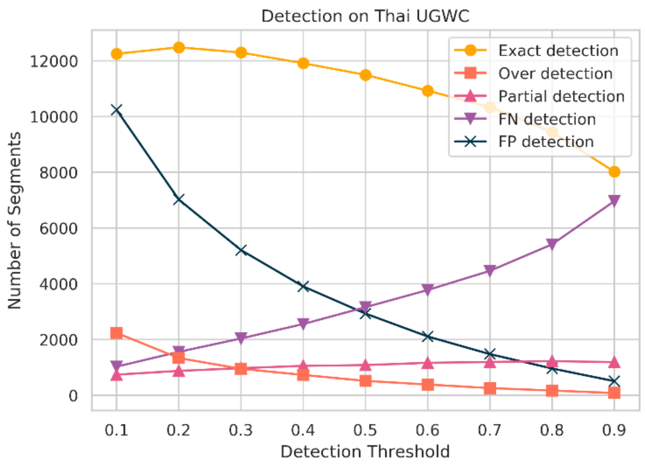


Figure 8 Types of error segments produced from the detection stage of our method on the Thai UGWC test set

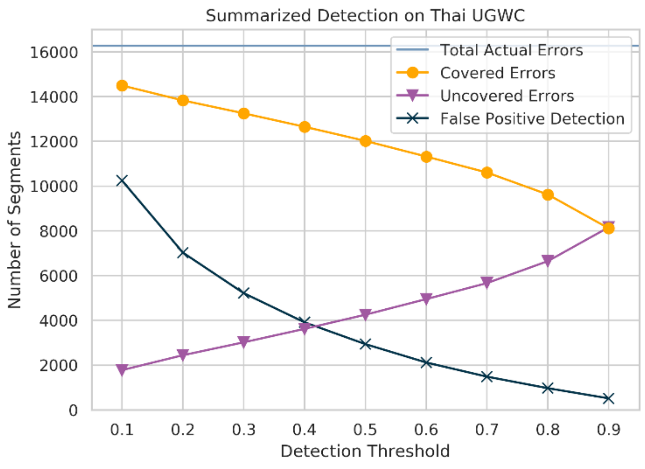


Figure 9 Coverage of the detection and false-positives produced from the detection stage to our method on the Thai UGWC test set

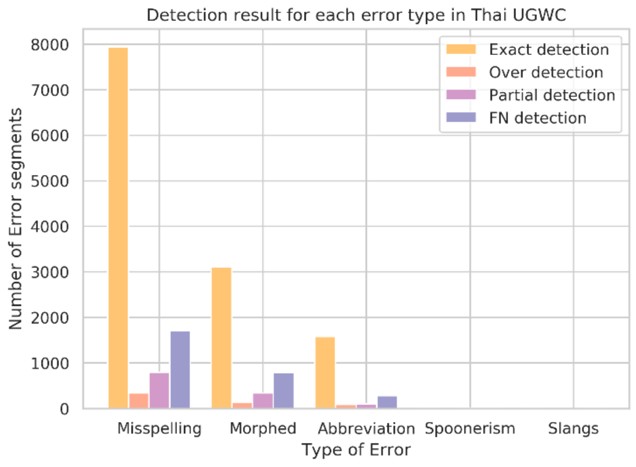


Figure 10 Number of error segments produced for different types of errors on the Thai UGWC test set

จุฬาลงกรณ์มหาวิทยาลัย

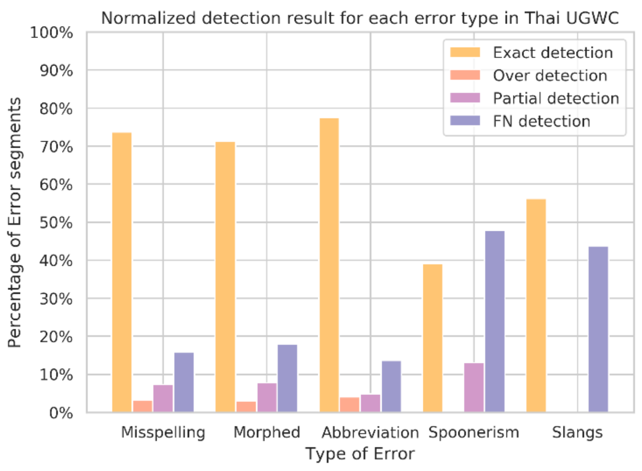


Figure 11 Normalized number of error segments produced for different types of errors on the Thai UGWC test set

6.5 Correction stage

This chapter 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. For partial detection, any correction is considered incorrectly corrected because portions of the erroneous text lie outside the error segments. Figure 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 Figures Figure 13, Figure 14, and Figure 15. A breakdown of the evaluation by error types is shown in Figures Figure 16 and Figure 17. In-line with error detection, performance is consistent for misspelled words, morphed words, and incorrect abbreviation notations but varies for spoonerisms and slang.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

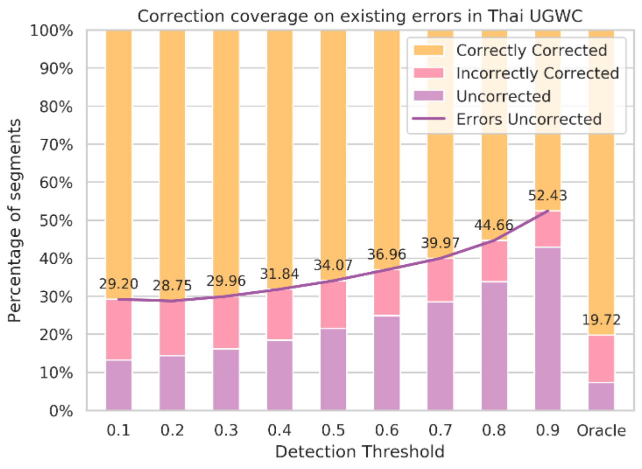


Figure 12 Correction coverage of our method on the Thai UGWC test set

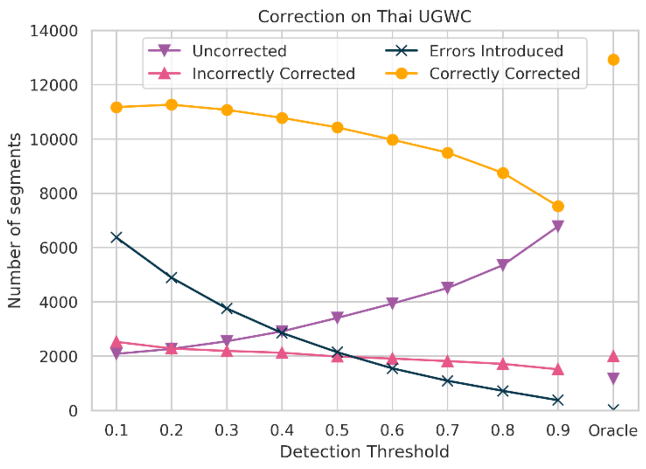


Figure 13 Types of corrections produced by our method on the Thai UGWC test set

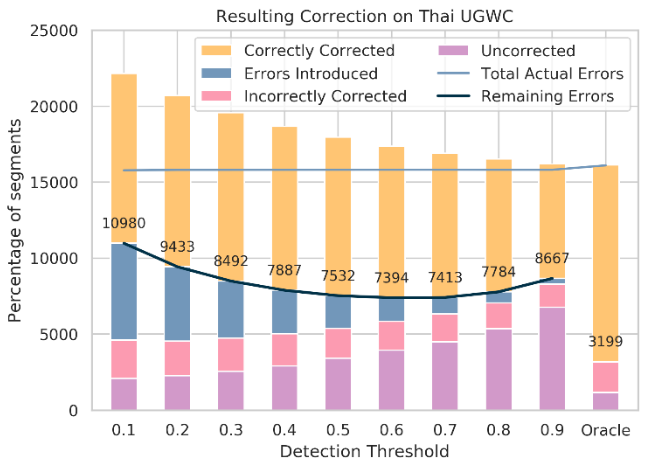


Figure 14 Breakdown of the remaining errors after executing our method on the Thai UGWC test set

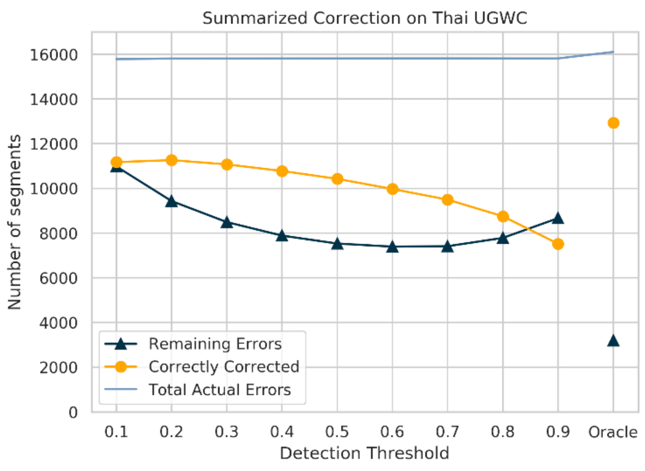


Figure 15 Coverage of the correction and the resulting number of errors from our method on the Thai UGWC test set

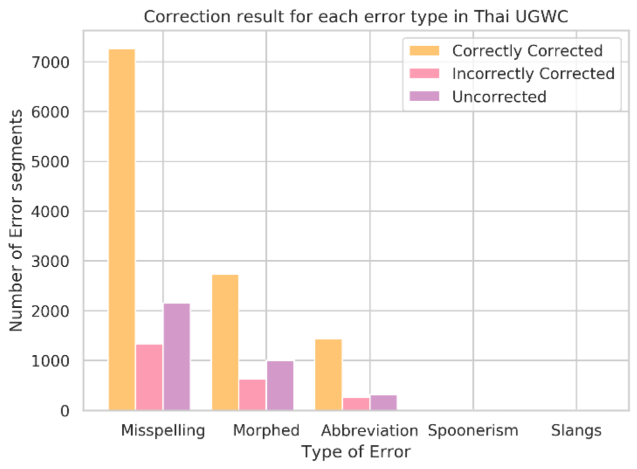


Figure 16 The number of error segments corrected for different types of errors on the Thai UGWC test set

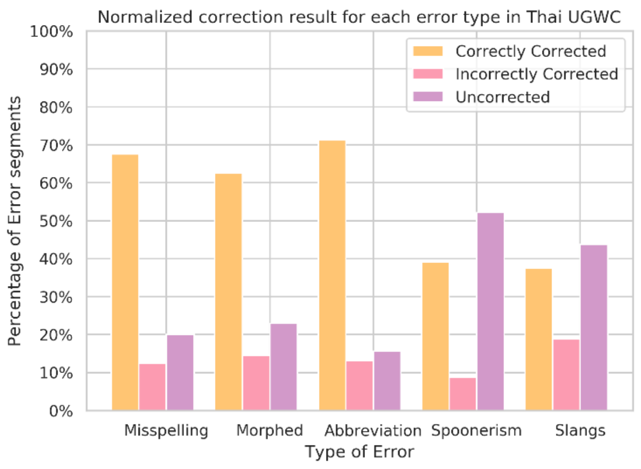


Figure 17 Normalized number of error segments corrected for different types of errors on the Thai UGWC test set

6.6 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 evaluated as a typical detection task using the F_1-score on the test set. The results are shown in Figure 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 chapter, we tuned the detection threshold on the development set, which sets the $\text{threshold} = 0.5$.

6.7 Multi-pass correction

In this section, we investigate how performing multiple correction pass with our method. Prior research on English Grammatical Error Correction has shown that an error corrector may not be able to correct all errors within a single correction pass (Ge, Wei, & Zhou). With the rationale being that the model may be confused by the errors within the context. Thus, by first reducing the error, the model should better correct the remaining errors.

The process for multi-pass correction is as follows. We initially perform correction as detailed in Chapter 5; however, the corrected output is then fed back to be re-corrected. For evaluation, this process is repeated for five iterations to ensure convergence of correction score.

Our method was evaluated on three groups multi-pass correction configurations: regular correction, under-correction, over-correction. As concluded from our last section (Chapter 6.6), our method produced the best overall correction when the detection stage's threshold is tuned to the using the F_1 score on the detection task, that is, $\text{threshold} = 0.5$ (when tuned on the development set). For regular correction, the same tuned threshold is chosen and fixed for each iteration. And, in theory, allow the corrector to correct the remaining errors in the text. For under-correction, the threshold is raised. For over-correction, the threshold is lowered. Based on our results in Section 6.5, the detection threshold dictates the trade-off between the number of existing errors corrected, and the number of errors introduced by the corrector.

The results for multi-pass correction are shown in Table 8. Multi-pass correction, when applied directly to our method, does not yield any improvements in performance for any of the detection thresholds. For over-correction and regular correction thresholds, the corrector produces the best correction on the first iteration, whereas the subsequent iterations lead to more errors being introduced. For under-correction thresholds, the corrector converges to some

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

correction score. However, the resulting correction is lower than that of a single pass correction at the tuned threshold value. The higher correction score at $threshold = 0.4$ is explained in Chapter 6.6.

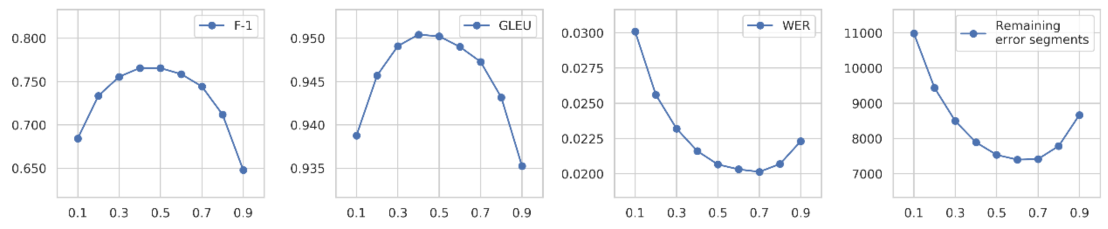


Figure 18 Detection and end-to-end correction performances at different detection sensitivities

Table 8 Evaluation of our method with multi-pass correction at different detection thresholds on the Thai UGWC test set

	Detection Threshold					
	Over-correction		Regular	Under-correction		
Iteration	0.3	0.4	0.5	0.6	0.7	0.8
1	**0.9491**	**0.9504**	**0.9502**	0.9490	0.9473	0.9432
2	0.9455	0.9484	0.9498	0.9496	0.9483	0.9451
3	0.9461	0.9486	0.9499	**0.9497**	**0.9484**	**0.9453**
4	0.9449	0.9480	0.9498	0.9496	0.9483	0.9453
5	0.9458	0.9484	0.9498	0.9496	0.9483	0.9453

6.8 Publicly released Thai UGWC dataset

In this section, we outline the results of our method on the two publicly released versions of our Text Correction on the Thai UGWC dataset. First is a smaller version of our dataset, containing 15,597 entries. Second is a text correction dataset with additional annotation for transliteration, which contains 17,307 entries. Both released datasets are pre-shuffled. For evaluation, **without reordering the data**, split the data into three sets (training set 75%, development set 5%, and test set 20%). For the first set (without transliteration), the 11,697 leading entries are the training set, then the next 780 entries are the development set, and the last 3120 entries are the test set. For the second set (with transliteration), the 12,980 leading entries are the training set, then the next 865 entries are the development set, and the last 3,462 entries are the test set. Our method is trained and evaluated on each dataset, as detailed in

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Chapter 5.4. Evaluation of the detection stage is shown in Table 9. And the end-to-end evaluation is shown in Table 10.

On the public text correction task, our method was able to produce output with lower word-error-rates and higher fluency scores (GLEU). With the augmented model without fine-tuning performing the worse, and the augmented + fine-tuned model performing the best. Results on the public task are in-line with the full-size dataset reported in Chapter 6.2. However, on the public text correction with transliteration, employing augmentation and fine-tuning did not result in better detection performance.

Table 9 Detection evaluation of our method on the test set of the publicly released Thai UGWC datasets

Model	*Precision*	*Recall*	*F_1*
Public Text Correction			
Detection	**0.910**	0.899	0.905
Detection*	0.899	0.892	0.896
Detection**	0.905	**0.908**	**0.907**
Public Text Correction (with Transliteration)			
Detection	**0.922**	0.900	**0.911**
Detection*	0.894	**0.911**	0.902
Detection**	0.901	0.907	0.904
* Model is only trained on noise injected dataset. ** Model is pre-trained on noise injected dataset before fine-tuned on the regular training			

CHULALONGKORN UNIVERSITY

Table 10 End-to-end evaluation of our method on the test set of the publicly released Thai UGWC datasets

Detector	*Corrector*	*GLEU*	*WER (%)*	*∆WER (%)*
Text Correction				
Do nothing		0.484	16.04	0.000%
Oracle		1.000	0.00	-100.000%
Ours	Ours	0.886	4.55	-71.600%
Oracle	Ours	0.956	2.14	-86.641%
Ours*	Ours*	0.883	4.72	-70.579%
Ours**	**Ours****	**0.896**	**4.10**	**-74.404%**
Oracle	Ours**	0.966	1.59	-90.106%
Text Correction (with Transliteration)				
Do nothing		0.484	16.04	0.000%
Oracle		1.000	0.00	-100.000%
Ours	Ours	0.865	6.50	-59.489%
Oracle	Ours	0.937	3.40	-78.814%
Ours*	Ours*	0.872	6.50	-59.473%
Ours**	Ours**	0.873	6.32	-60.586%
Ours	**Ours****	**0.875**	**5.97**	**-62.779%**
Oracle	Ours**	0.948	2.68	-83.267%

* Model is only trained on noise injected dataset.

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

CHULALONGKORN UNIVERSITY

7 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 (Grundkiewicz & Junczys-Dowmunt)) 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 (Zhao et al.)) showed that a Seq2Seq corrector with a substructure that encourages copying could 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.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

7.1 Limitations

When compared to traditional methods (e.g., Hunspell (Hunspell)), the vocabulary of deep-learning-based methods (e.g., our method, Bi-GRU (Grundkiewicz & Junczys-Dowmunt), copy-augmented transformer (Zhao et al.)) are not as easily extendable. Every new word must be introduced in the form of example sentences, as is opposed to a dictionary. Also, the number of examples required for the model to generalize is not apparent.

In terms of our experiment, our method is evaluated a specific type of data (i.e., Thai user-generated web-content). As such, further experimentation is required to determine the performance of a text corrector on other domains.

7.2 Contributions

In this section, we will cover the completed works during the development of this thesis, in addition to the findings of the study.

With cooperation with Faculty of Arts of Chulalongkorn University, Faculty of Engineering of Chulalongkorn University, and Kasikorn Labs, we developed the annotations guidelines for a variety of Thai natural language processing task (including the Thai Text Correction in this thesis) as well as annotated a corpus build on User-generated web content (social text). The overall details of the corpus were published in (Lertpiya et al., 2018). Additional details of the text correction task are covered in Chapter 4.2.

This study was conducted to develop fundamental natural language processing (NLP) services to enhance KBank's NLP capabilities, where the text correction task is one of six fundamental NLP tasks. Thus, in addition to the research of the text correction method (detailed in this thesis), we also developed a web API to increase model ease of use.

Partially of our Thai user-generated web-content is the first publicly available dataset on the text correction task for Thai text.

7.3 Future work

In this work, 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 (Kiyono et al.)) could further improve the correction performance.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

From our observation, we found that large portions of errors and variance in the Text Correction task (as defined in Chapter 4) are filler words. As such, further investigation is needed for adapting text correctors as preprocessing of another task.

In addition, we found during error analysis (outlined in Appendix 0) that reducing error propagation from word tokenization can also increase correction performance, thus further research into error resilient word tokenization could benefit the text correction task.

8 Appendix

8.1 Hyperparameters

In this section, we cover the process and results of hyperparameters tuning on our method on the Thai Text Correction (Thai TC) task.

On the Thai text correction task, we obtained the hyperparameter values for both our method and the other neural-based methods evaluated in this thesis (i.e., Bi-GRU (Grundkiewicz & Junczys-Dowmunt) and copy-augmented transformer (Norvig)) 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 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 (in Chapter 8.3), 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 (Grundkiewicz & Junczys-Dowmunt) and copy-augmented transformer (Norvig)).

8.2 Error Analysis of Text Correction on Thai UGWC

8.2.1 End-to-end correction

In this chapter, we perform end-to-end error analyses on each of the results reported on our text correction task on the Thai user-generated web content (text data collected from social media) described in Table 4. We selected two lines (see Table 12, Table 13, and Table 14) 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 (Grundkiewicz & Junczys-Dowmunt) on untokenized text, where the model operated on the SentencePiece tokens (Kudo & Richardson), 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.

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 (Kingma & Ba)
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 ($E^{(l)}$) dropout	0.5
Context encoding ($C^{(l)}$) dropout	0.5
Batch size	32
Optimizer	Adam (Kingma & Ba)
Learning-rate	0.002
Fine-tuning learning-rate	0.0005
Early stopping patience	20 groups*
* Error Corrector training procedures are detailed further in Chapter 5.4.	

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.

จุฬาลงกรณ์มหาวิทยาลัย

End-to-end correction systems (i.e., Bi-GRU, Copy-augmented Transformer) 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.

8.2.2 Correction stage

In this chapter, we perform error analyses of our method correction stage on the Thai user-generated web content (text data collected from social media). The incorrect output produced from our method is shown in Table 15, Table 16, and Table 17. We analyze the incorrect outputs produced from the correction stage, given an exact detection (as described in Chapter 6.4). We sampled 15 corrections for “misspelling” since this is the majority of the errors in the dataset. We sampled ten corrections for “morphed” and “abbreviation”. Due to a lack of “slangs” and “spoonerism” errors, we only sampled the correct outputs produced, only nine corrections were sampled for “slangs” and one correction for “spoonerism”.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Table 12 End-to-end 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|แอฟ|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" overcorrected: "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" overcorrected: "Other" sound correction*
Hunspell (Trained)	*"พอ|ผม|โหลด|แอฟ|K Bank| |มา|แล้ว|ผม| |อยาก|เข้า|บัญชี|อีก|อัน|อ่ะ|คับ"* *no correction: same as "Source"*
PyThaiNLP	*"พอ|ผม|โหลด|แอฟ|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" overcorrection: "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" overcorrected: "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" overcorrected: "Other" sound correction*

CHULALONGKORN UNIVERSITY

Table 13 End-to-end 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 (untokenized)	*“เช่น|โอน|เข้า|บัญชี|นี้|ครับ”*		
	From 3 To 4	*“บัญชี”*	*overcorrected (meaning lost)*
	From 5 To 11	*“ครับ”*	*overcorrected (meaning lost)*

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Table 14 End-to-end error analysis of each model on the second sample line on the TC task on the Thai UGWC dataset (continue)

Line 2	
Source	*“เช่น|โอน|เข้า|เวบ|นี้|เพือ|เข้า|ใป|เล่น|ทัมอย่างใร|คับ”*
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”*
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” overcorrected (words dropped)*

Table 15 Correction error analysis of abbreviation errors on the TC task on the Thai UGWC dataset

	Error Types	Issues with correction	Erroneous text	Labels	Correction produced
	Abbreviation				
1	Official abbreviation	Miscorrected	สว	สูง|วัย	สวย
2			พศ	พ|.|ศ|.	พี่
3	Non-official abbreviation + incorrect tokenization	Uncorrected	ปิดจนท	ปิด|เจ้าหน้าที่	<OOV>
4	Label noise	Label: non-official abbreviation	จขกท	จขกท|.	เจ้าของ|กระทู้
5		Label: overcorrection to English	คซต	call center	คอลเซ็นเตอร์
6		Label: overcorrection to full word	ธ	ธนาคาร	ธ.
7			ชม	ชั่วโมง	ชม.
8	Official abbreviation + incorrect tokenization	Word tokenization inconsistency	ที่กทม	ที่|กทม|.	ที่กทม|.
9	Non-official abbreviation		จขกท	เจ้า|ของ|กระทู้	เจ้าของ|กระทู้
10			คห	ความเห็น	ความ|เห็น

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Table 16 Correction error analysis of misspelling on the TC task on the Thai UGWC dataset

	Error Types	Issues with correction	Erroneous text	Labels	Correction produced
	Misspelling				
1	Single-word	Corrected as non-existent word	ล๊อค	ล็อก	ล็อค
2		Contextually incorrect (real word error)	อยาง	อยาก	อย่าง
3			สตาง	สตางค์	ส|.|ต่าง
4			ยอก|เลิด	ยกเลิก	ยอด
5			ไช่	ใช่	ใช้
6		Uncorrected	กาแลคซี่	กาแล็กซี	<OOV>
7	Single-word + incorrect tokenization		ฅนล้า	คนล้า	<OOV>
8			นุ๋พยายาม	หนูพยายาม	<OOV>
9		Dropping word(s)	ค่พอไป|ตืดต่อ	ค่ะ|พอ|ไป|ติดต่อ	พอไป|ติดต่อ
10	Single-word + punctuation + incorrect tokenization		สมัค|sm|s	สมัคร| |sms	สมัคร
11	Multi-word	Partially correct	ยุ|ตรุรกี	อยู่|ตุร|กี	อยู่|กรณี
12			ใด้|หรอ	ได้|หรือ	ได้|หรอ
13	Multi-word + Label noise	Label: Contextually incorrect (real word error) (proper correction is อย่างไร|คะ)	งัย|ค้ะ	ไง|ค่ะ	ไงคะ
14	Label noise	Label: Corrected as non-existent word	แอฟ	แอพ	แอป
15	Evaluation limitation	Multiple possible spelling: both the label and the correction are correct	เวบ	เว็บ	เว็บไซต์

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Table 17 Correction error analysis of morphed, spoonerism, and slang on the TC task on the Thai UGWC dataset

	Error Types	Issues with correction	Erroneous text	Labels	Correction produced
	Morphed				
1	Single-word	Contextually incorrect (real word error)	กิง	จริง	กิน
2			ปั้ง	ปัง	ทั้ง
3			คร้า	ค่ะ	คะ
4		Dropping word(s)	ปะ	หรือเปล่า	เปล่า
5	Single-word + incorrect tokenization	Uncorrected	เจ้ามือหลังม่าน|เมกา	เจ้ามือหลัง|ม่า|นอเมริกา	<OOV>
6			โอ้ยยยรำคาญ	โอ๊ย|รำคาญ	โอ๊ย|<OOV>
7	Label noise	Label: Corrected as non-existent word	เหยดดดด	เหยด	เย็ด
8			หรอ|เนี่ยยย	เหรอ|เนี่ย	เหรอ|นี่
9		Label: Contextually incorrect (real word error)	จ่ะ	จ๊ะ	จ้ะ
10	Evaluation limitation	Multiple possible spelling: both the label and the correction are correct	ใด้|เหรอ	ได้|หรือ	ได้|เหรอ
	Slang				
1	Single-word	Contextually incorrect (real word error)	ค้า|ป	ครับ	คะ
2			ปะ	ไหม	เปล่า
3		Corrected as non-existent word	ตรู	กู	ตู
4		None: Good correction	ปั๋ว	ผัว	ผัว
5			มะ	ไหม	ไหม
6			คับ	ครับ	ครับ
7			ขนาดด	ขนาด	ขนาด
8			จอดดด	จอด	จอด
9	Single-word + incorrect tokenization	Word tokenization inconsistency	เด๋วนี้	เดี๋ยว|นี้	เดี๋ยวนี้
	Spoonerism				
1	Single-word	None: Good correction	สวีดัด	สวัสดี	สวัสดี

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

8.2.3 Word tokenization on the correction stage

In this chapter, we investigate how removing tokenization issues will impact the correction stage. We hand tokenized input of problematic correction from Chapter 8.2.2. The results are shown in Table 18. Six out of nine the problematic correction have been resolved, two are left uncorrected, and the last missing a punctuation token.

Table 18 Correction error analysis on retokenized erroneous text on the TC task on the Thai UGWC dataset

Error Types		Issues with correction	Retokenized erroneous text	Labels	Correction produced
	Abbreviation				
3		False negative detection	ปิด|จนท	ปิด|เจ้าหน้าที่	ปิด|จนท
8		None: Good correction	ที่|กทม	ที่|กทม|.	ที่|**กทม|.**
	Misspelling				
7		None: Good correction	ฅน|ล้า	คน|ล้า	**คน**|ล้า
8			นู๋|พยายาม	หนู|พยายาม	**หนู**|พยายาม
9			ค่|พอไป|ตืดต่อ	ค่ะ|พอ|ไป|ติดต่อ	**ค่ะ**|พอไป|**ติดต่อ**
10		Dropping word(s)	สมัค|sms	สมัคร| |sms	**สมัคร**|sms
	Morphed				
5		Uncorrected	เจ้ามือ|หลัง|ม่าน|เมกา	เจ้ามือ|หลัง|ม่าน|อเมริกา	เจ้ามือ|หลัง|**<OOV>**
6		None: Good correction	โอ้ยยยรำคาญ	โอ๊ย|รำคาญ	**โอ๊ย**|รำคาญ
	Slang				
9		None: Good correction	เด๋วนี้	เดี๋ยว|นี้	เดี๋ยว|นี้

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

8.3 Evaluation on Conll-2014

In this chapter, we evaluate our method on the original as well as a modified version of the Conll-2014 Grammatical Error Correction shared task (Ng et al., 2014). The results are shown in Table 19 and Table 20. 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 method below doing nothing.

To evaluate only the spelling correcting capabilities of our model, we built a spelling correction task using the Conll-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 (Grundkiewicz & Junczys-Dowmunt) 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 thesis (Zhao et al., 2019)) 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.

Table 19 Evaluation of end-to-end error correction on the GEC Conll-2014 dataset by various systems

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

* Model is only trained on noise injected dataset.

** Model is pre-trained on character-level noise injected dataset before fine-tuned on the regular training

**** Model is pre-trained on word-level noise injected dataset before fine-tuned on the regular training according to (Zhao et al., 2019)

จุฬาลงกรณ์มหาวิทยาลัย

Table 20 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
Ideal correction (Oracle)	0.9917	0.9896	0.9900	0.7767
Bi-GRU	0.9266	0.8660	0.9138	0.7358
Bi-GRU*	0.8926	0.8623	0.8863	0.7356
Copy-Aug Transformer***	0.8066	0.8552	0.8159	0.7382
Copy-Aug Transformer****	0.8351	0.8587	0.8397	0.7409
Ours	0.9184	0.8982	0.9143	0.7487
Ours*	0.9073	0.9028	0.9064	0.7496
Ours**	**0.9567**	**0.9082**	**0.9466**	**0.7539**
Ours** + (Oracle Detection)	0.9436	0.9314	0.9411	0.7665

* Model is only trained on noise injected dataset.

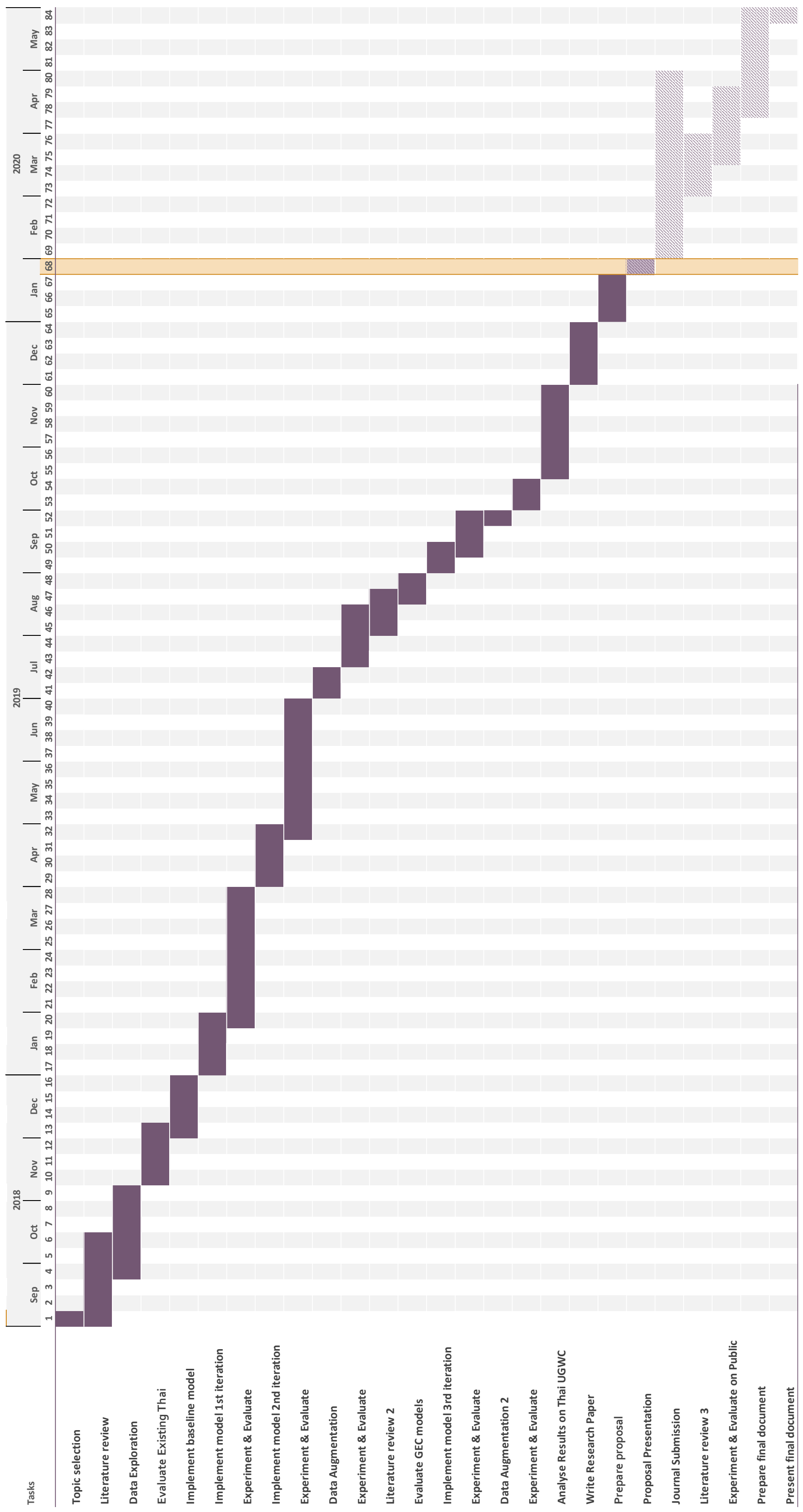
** Model is pre-trained on character-level noise injected dataset before fine-tuned on the regular training

*** Model is pre-trained on word-level noise injected dataset according to (Zhao et al., 2019)

**** Model is pre-trained on word-level noise injected dataset before fine-tuned on the regular training according to (Zhao et al., 2019)

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

9 Research plan



REFERENCES

AllenNLP. (2019). AllenNLP Bidirectional Attention Flow Implementation. Retrieved from https://github.com/allenai/allennlp/tree/v0.8.2/allennlp/models/reading_comprehension

Aroonmanakun, W. (2007). *Thoughts on word and sentence segmentation in Thai.* Paper presented at the Proceedings of the Seventh Symposium on Natural language Processing, Pattaya, Thailand, December 13–15.

Atkinson, K. (2018). GNU Aspell. Retrieved from http://aspell.net/

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. J. a. p. a. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation.

Chollampatt, S., & Ng, H. T. (2018a). *A multilayer convolutional encoder-decoder neural network for grammatical error correction.* Paper presented at the Thirty-Second AAAI Conference on Artificial Intelligence.

Chollampatt, S., & Ng, H. T. (2018b). *Neural quality estimation of grammatical error correction.* Paper presented at the Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research, 7*(Mar), 551-585.

Dahlmeier, D., & Ng, H. T. (2012). *Better evaluation for grammatical error correction.* Paper presented at the Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Farzindar, A., & Inkpen, D. (2015). Natural language processing for social media. *Synthesis Lectures on Human Language Technologies, 8*(2), 1-166.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., . . . Zettlemoyer, L. (2018). Allennlp: A deep semantic natural language processing platform. *arXiv*

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

preprint arXiv:1803.07640.

Ge, T., Wei, F., & Zhou, M. (2018). *Fluency boost learning and inference for neural grammatical error correction.* Paper presented at the Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Grundkiewicz, R., & Junczys-Dowmunt, M. (2018). Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv preprint arXiv:1804.05945*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

Hunspell. (2019). Hunspell. Retrieved from https://github.com/hunspell/hunspell

Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S., & Heafield, K. (2018). Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv preprint arXiv:1804.05940*.

Junczys-Dowmunt, M., & Grundkiewicz, R. J. a. p. a. (2016). Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiyono, S., Suzuki, J., Mita, M., Mizumoto, T., & Inui, K. (2019). An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction. *arXiv preprint arXiv:1909.00502*.

Kruatrachue, B., Somguntar, K., & Siriboon, K. (2002, 6-8 Nov. 2002). *Thai OCR error correction using genetic algorithm.* Paper presented at the First International Symposium on Cyber Worlds, 2002. Proceedings.

Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lee, K., Agrawal, A., & Choudhary, A. (2013). *Real-time disease surveillance using Twitter*

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

data: demonstration on flu and cancer. Paper presented at the Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, Chicago, Illinois, USA.

Lertpiya, A., Chaiwachirasak, T., Maharattanamalai, N., Lapjaturapit, T., Chalothorn, T., Tirasaroj, N., & Chuangsuwanich, E. (2018, 15-17 Nov. 2018). *A Preliminary Study on Fundamental Thai NLP Tasks for User-generated Web Content.* Paper presented at the 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP).

Lertpiya, A., Chalothorn, T., & Chuangsuwanich, E. (2020). Thai Spelling Correction and Word Normalization on Social Text Using a Two-Stage Pipeline With Neural Contextual Attention. *IEEE Access, 8*, 133403-133419. doi:10.1109/ACCESS.2020.3010828

Meknavin, S., Kijsirikul, B., Chotimongkol, A., & Nuttee, C. (1998a). *Combining trigram and Winnow in thai OCR error correction.* Paper presented at the Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2, Montreal, Quebec, Canada.

Meknavin, S., Kijsirikul, B., Chotimongkol, A., & Nuttee, C. (1998b, 24-27 Nov. 1998). *Progress of combining trigram and Winnow in Thai OCR error correction.* Paper presented at the IEEE. APCCAS 1998. 1998 IEEE Asia-Pacific Conference on Circuits and Systems. Microelectronics and Integrating Systems. Proceedings (Cat. No.98EX242).

Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2015). *Ground truth for grammatical error correction metrics.* Paper presented at the Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers).

Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2016). GLEU without tuning. *arXiv preprint arXiv:1605.02592*.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014). *The CoNLL-2014 shared task on grammatical error correction.* Paper presented at the Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task.

Norvig, P. (2007). How to Write a Spelling Corrector. Retrieved from https://norvig.com/spell-correct.html

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). *BLEU: a method for automatic evaluation of machine translation.* Paper presented at the Proceedings of the 40th annual meeting of the Association for Computational Linguistics.

PyThaiNLP. (2019a). PyThaiNLP. Retrieved from https://github.com/PyThaiNLP/pythainlp

PyThaiNLP. (2019b). PyThaiNLP/spelling-check. Retrieved from https://github.com/PyThaiNLP/spelling-check

Rodphon, M., Siriboon, K., & Kruatrachue, B. (2001, 26-28 Aug. 2001). *Thai OCR error correction using token passing algorithm.* Paper presented at the 2001 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (IEEE Cat. No.01CH37233).

Rozovskaya, A., & Roth, D. (2016). *Grammatical error correction: Machine translation and classifiers.* Paper presented at the Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*(11), 2673-2681.

Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Society, O. o. t. R. (1999). *The Royal Institute Dictionary 2542 B.E.*: Nanmeebooks.

Watcharabutsarakham, S. (2005, 21-24 Nov. 2005). *Spell Checker for Thai Document.* Paper presented at the TENCON 2005 - 2005 IEEE Region 10 Conference.

wolfgarbe. (2019). SymSpell. Retrieved from https://github.com/wolfgarbe/SymSpell

Zhao, W., Wang, L., Shen, K., Jia, R., & Liu, J. (2019). Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

arXiv preprint arXiv:1903.00138.

Zukarnain, N., Abbas, B. S., Wayan, S., Trisetyarso, A., & Kang, C. H. (2019). *Spelling Checker Algorithm Methods for Many Languages.* Paper presented at the 2019 International Conference on Information Management and Technology (ICIMTech).

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

VITA

NAME	Anuruth Lertpiya
DATE OF BIRTH	11 September 1995
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University
HOME ADDRESS	52/17 Nuanchan Rd. Nuanchan, Bungkum, Bangkok 10230
PUBLICATION	Lertpiya, A., Chaiwachirasak, T., Maharattanamalai, N., Lapjaturapit, T., Chalothorn, T., Tirasaroj, N., & Chuangsuwanich, E. (2018, 15-17 Nov. 2018). A Preliminary Study on Fundamental Thai NLP Tasks for User-generated Web Content. Paper presented at the 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP). Lertpiya, A., Chalothorn, T., & Chuangsuwanich, E. (2020). Thai Spelling Correction and Word Normalization on Social Text Using a Two-Stage Pipeline With Neural Contextual Attention. IEEE Access, 8, 133403-133419. doi:10.1109/ACCESS.2020.3010828

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University