ระบบแบบทนทานสำหรับเทคโนโลยีหลักในการประมวลผลภาษาธรรมชาติภาษาไทย

นายแคน อุดมเจริญชัยกิจ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

A ROBUST SYSTEM FOR CORE THAI NATURAL LANGUAGE PROCESSING TECHNOLOGIES

Mr. Can Udomcharoenchaikit

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2020

Thesis Title	A ROBUST SYSTEM FOR CORE THAI NATURAL LANGUAGE PROCESSING TECHNOLOGIES
By	Mr. Can Udomcharoenchaikit
Field of Study	Computer Engineering
Thesis Advisor	Asst. Prof. Peerapon Vateekul, Ph.D.
Thesis Co-advisor	Prachya Boonkwan, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

Dean of the Faculty of Engineering

. . . .

(Prof. Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

. . . . Chairman

(Prof. Boonserm Kijsirikul, Ph.D.)

. . . . Thesis Advisor

(Asst. Prof. Peerapon Vateekul, Ph.D.)

. . . . Thesis Co-advisor

(Prachya Boonkwan, Ph.D.)

. . . . Examiner

(Assoc. Prof. Wirote Aroonmanakun, Ph.D.)

. . . . Examiner

(Ekapol Chuangsuwanich, Ph.D.)

. . . . External Examiner

(Prof. Thanaruk Theeramunkong, Ph.D.)

แคน อุดมเจริญชัยกิจ: ระบบแบบทนทานสำหรับเทคโนโลยีหลักในการประมวลผลภาษาธรรมชาติภาษาไทย. (A ROBUST SYSTEM FOR CORE THAI NATURAL LANGUAGE PROCESSING TECHNOLOGIES) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร.พีรพล เวทีกูล, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : ดร.ปรัชญา บุญขวัญ 139 หน้า.

เมื่อข้อมูลที่เป็นข้อความภาษามีจำนวนมากขึ้นการสร้างระบบอัจฉริยะที่สามารถประมวลผลภาษามนุษย์ได้จึงมีความสำคัญมากขึ้น ระบบประมวลผลภาษาธรรมชาติเป็นเทคโนโลยีที่ช่วยให้คอมพิวเตอร์ใช้ประโยชน์จากภาษาของมนุษย์เพื่อทำงานต่าง ๆ จึงมีความจำเป็นมากขึ้น โมเดลการเรียนรู้เชิงลึกได้แสดงผลลัพธ์ที่ยอดเยี่ยมในงานพื้นฐานในการประมวลผลภาษาธรรมชาติ เช่น การตัดคำ การจำแนกชนิดของคำ และการรู้จำชื่อเฉพาะ อย่างไรก็ตามในบาง สถานการณ์วิธีการที่เสนอเหล่านี้ไม่สามารถทำงานได้ดีเท่าที่ควร เพื่อให้ระบบประมวลผลภาษาธรรมชาติมีเสถียรภาพมากขึ้น เราควรแก้ไขปัญหาที่ปรากฏขึ้นบ่อยครั้ง และมีอิทธิพลต่อประสิทธิภาพของระบบ ได้แก่ ปัญหาการรับมือกับคำศัพท์ที่ไม่เคยพบและคำสะกดผิด เป้าหมายการวิจัยของวิทยานิพนธ์นี้คือการพัฒนาแบบระบบประมวลผลภาษาธรรมชาติที่สามารถจัดการกับข้อความที่สะกดผิดเพื่อปรับปรุงโมเดลให้ใช้งานได้ดีขึ้นเมื่อนำไปใช้จริง วิทยานิพนธ์นี้เสนอโมเดลการเรียนรู้ของเครื่องและการประเมินผลแบบใหม่ที่มุ่งเน้นไปที่การเพิ่มความทนทานต่อข้อความที่มีการสะกดผิดรูปแบบ

วิทยานิพนธ์ฉบับนี้เสนอกลยุทธ์และระบบประมวลผลภาษาธรรมชาติใหม่ เพื่อปรับปรุงความทนทานต่อคำสะกดผิด วิทยานิพนธ์นี้สำรวจกลยุทธ์การจัดการข้อมูลอินพุตที่ทำให้ข้อมูลอินพุตมีความหลากหลายมากขึ้น เช่นการใส่หน้ากากคำที่ไม่เคยพบ (UNK Masking) และการฝึกปรปักษ์ (Adversarial Training) วิทยานิพนธ์ฉบับนี้สำรวจว่าหน่วยของภาษาที่เล็กกว่าคำสามารถปรับปรุงความแข็งแกร่งของการฝังคำได้อย่างไร นอกจากนี้ยังตรวจสอบเทคนิคการ จำกัดความคล้ายคลึงกันระหว่างข้อความเช่นการใช้ฟังก์ชันการสูญเสียแบบชุดสาม (Triplet Loss) เพื่อจำกัดความคล้ายคลึงกันระหว่างข้อความต้นฉบับกับข้อความที่สะกดผิด

นอกจากนี้ยังเสนอรูปแบบการประเมินแบบใหม่ที่เปิดเผยจุดอ่อนของระบบประมวลผลภาษาธรรมชาติ โดยการใส่ตัวอย่างปรปักษ์ (Adversarial Examples) จากการพิมพ์ผิดลงไปในชุดข้อมูลสำหรับทดสอบ แผนการประเมินแบบปรปักษ์ (Adversarial Evaluation) ที่ได้เสนอในวิทยานิพนธ์ฉบับนี้แสดงให้เห็นว่าแบบจำลองการเรียนรู้เชิงลึกในปัจจุบันไม่ทนทานเมื่อเจอข้อมูลที่สะกดผิดและยังแสดงให้เห็นว่ากลยุทธ์และสถาปัตยกรรมระบบประมวลผลภาษาธรรมชาติของเราสามารถปรับปรุงประสิทธิภาพได้เมื่อเจอข้อความที่มีการสะกดผิด

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต	....
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อ อ.ที่ปรึกษาหลัก	....
ปีการศึกษา	2563	ลายมือชื่อ อ.ที่ปรึกษาร่วม	....

5971402521: MAJOR COMPUTER ENGINEERING

KEYWORDS: NATURAL LANGUAGE PROCESSING / ADVERSARIAL EXAMPLES / DEEP LEARNING

CAN UDOMCHAROENCHAIKIT : A ROBUST SYSTEM FOR CORE THAI NATURAL LANGUAGE PROCESSING TECHNOLOGIES. ADVISOR : ASST. PROF. PEERAPON VATEEKUL, Ph.D., THESIS CO-ADVISOR : PRACHYA BOONKWAN, Ph.D., 139 pp.

As the amount of unstructured textual data grows, it becomes increasingly important to build an intelligent system that can process it. Natural Language Processing (NLP) is a technology that allows a computer to exploit human languages to perform tasks. Deep learning models have shown excellent results across fundamental tasks in NLP, such as word segmentation, part-of-speech tagging, and named-entity recognition. However, in many situations, these proposed methods fail to perform well. For an NLP system to be robust, it must address issues such as out-of-vocabulary and spelling-mistakes. This thesis's research goal is to develop NLP models that can handle malformed texts to improve their real-world setting usability. In this thesis, I propose novel models and evaluations that focus on robustness against malformed texts.

This dissertation proposes multiple novel training strategies and architectures to improve the robustness against malformed texts. This thesis explores input data manipulation strategies that diversify training data, such as UNK masking and adversarial training. It explores how sub-lexical information can improve the robustness of word embeddings. Furthermore, it examines similarity constraint techniques, such as triplet loss, which constraint the similarity between the original texts and the parallel perturbed texts.

I also propose alternative evaluation schemes that reveal the weaknesses of NLP systems by introducing typographical adversarial examples to the test sets. Our adversarial evaluation schemes show that current deep learning models are not robust against misspelled inputs, and they also show that our proposed training strategies and architectures can improve the performance over malformed texts.

Department: Computer Engineering	Student's Signature ....
Field of Study: Computer Engineering	Advisor's Signature ....
Academic Year: 2020	Co-advisor's Signature ....

Acknowledgements

This dissertation is dedicated to those special humans who believe in me. I am grateful for my family and friends, who are an important part of my life. Those who lift me up when I am down. Those who never let me give up. Those who reminded me of the importance of being resilient.

I acknowledge the support of the Thailand Graduate Institute of Science and Technology (TGIST) for the generous funding throughout the years. I would like to thank my two advisors Peerapon Vateekul and Prachya Boonkwan, as well as the thesis committee, Boonserm Kijsirikul, Wirote Aroonmanakun, Ekapol Chuangsuwanich, and Thanaruk Theeramunkong, for their insightful suggestions.

CONTENTS

Page

LIST OF TABLES

Page

Page

LIST OF FIGURES

Page

Chapter I

INTRODUCTION

1.1 Overview

Natural Language Processing (NLP) is an important technology that bridges the gap between computers and humans. Humans communicate through natural languages; therefore, we need technology to process humans' complex language utterances. Many computer applications, from web search to language translation, rely on NLP as their core technology. This thesis aims to develop NLP systems that can robustly process the Thai language with out-of-distribution inputs. In particular, inputs with malformed or misspelled texts in order to process multiple fundamental language tasks for the Thai language.

Thai NLP is a challenging research problem due to the characteristic of the Thai language. Thai is an isolating language with a very low morpheme per word ratio with no inflection or derivation; therefore, Thai compensates for the lack of inflection by allowing words with higher complexity to be built from multiple morphemes. However, this relaxation causes ambiguity from the morphological level to the pragmatical level. For example, at the morphological level, the Thai writing system does not have explicit word boundaries, which makes word segmentation–a task that seems to be trivial in many languages–a challenging task for the Thai language. The lack of explicit word boundaries contributes to ambiguity in word segmentation task; for instance, there are two ways to segment “ตากลม”: “ตา|กลม-round eyes” and “ตาก|ลม-to be exposed to the wind”.

In modern NLP research, researchers have employed deep learning techniques, and they have obtained very high performance across various NLP tasks without relying on handcrafted features. Since deep learning can learn multiple levels of features or representations from raw input automatically, we no longer have to rely on a tedious feature engineering process to obtain state-of-the-art performance. There are many NLP literature and applications with promising results on standard test sets. However, in many situations, their proposed methods fail to perform well. For an NLP system to be robust, it must be able to address issues such as out-of-vocabulary, spelling-mistake, etc. The majority of language representations in previous work are word-level representations. Although word-level representation can provide a strong semantic representation, it is too rigid to handle scenarios where out-of-vocabulary and malformed words are present. This thesis seeks to improve two common shortcomings in current NLP literature:

1. **Model/Training Strategy:** since the standard evaluation paradigm often does not include non-standard text samples, researchers often train models without thinking consciously about non-standard text samples that may occur in the real-world. Here we develop training methods that aim to enhance the accuracy of the model on non-standard textual inputs.

2. **Evaluation:** the standard paradigm for evaluation estimates the performance by using train-validation splits that often do not contain enough out-of-distribution data in the test set. It overlooks common cases in real-world settings, such as spelling errors, which causes the accuracy to be overestimated. An alternative evaluation framework is needed to capture a wider range of phenomena. This thesis draws inspiration from challenge set evaluation and adversarial evaluation. It includes test sets that test the behavior of models on non-standard textual inputs.

This thesis proposes robust deep learning approaches to Thai NLP as well as evaluation frameworks to benchmark them. In this research, we develop learning models to improve the robustness against misspelled texts. Our evaluation frameworks are based on spelling errors, which also cause out-of-vocabulary (OOV) words and rare words. This thesis focuses on the core sequential labeling tasks for Thai NLP, such as tokenization, part-of-speech tagging, and named-entity recognition because they are the building blocks for other downstream NLP applications.

1.2 Aims and Objectives

This research's main objective is to build and evaluate NLP models that are robust against spelling errors. This dissertation aims to provide methods that can improve the robustness of the three core Thai NLP applications (word segmentation, part-of-speech (POS) tagging, and named-entity recognition (NER)). It explores alternative evaluation approaches that can reveal the weaknesses of the NLP systems and then uses this knowledge to design learning methods that enhance the robustness of the NLP models.

1.3 Scope of Work

Following tasks will be undertaken as a part of the proposed research:

1. Develop neural network architectures for three core Thai NLP applications: word segmentation, POS tagging, and NER
2. Develop neural network architectures for sequential tagging systems that are robust against spelling mistakes
3. Evaluate the performance of the proposed neural network architectures on misspelled words.

1.4 Contributions

Fig. 1.1 shows a visual overview of this thesis. The main contributions of this thesis can be found in chapter 4 and chapter 5. Both chapters evaluate the robustness of sequential tagging systems using typographical adversarial examples. Chapter 4 introduces black-box adversarial examples for Thai language. The results from this chapter show that NLP models are not robust against small spelling perturbations. Chapter 5 extends the results from chapter 4 to build stronger adversarial examples; it emphasizes on white-box adversarial examples. Both chapters also introduce different techniques that can be used to improve the robustness of the models over typographical adversarial examples. Chapter 5 applies black-box adversarial examples introduced in chapter 4 for adversarial training. The list of main contributions are as follows:

- **Black-box Adversarial Evaluation Scheme** for Thai NLP tasks based on known Thai spelling errors (Chapter 4)
- **UNK Masking**: A training data perturbation technique that improves the robustness of the sequential tagging models (Chapter 4)
- **Hidden State Initialization with Affixation Enbeddings**: A hidden state initialization method with linguistic knowledge (Chapter 4)
- **Untied-directional Self-Attention**: A self-attention mechanism that increases the interpretability of the neural networks (Chapter 4)
- **White-box Adversarial Evaluation Scheme** for Sequential tagging tasks (Chapter 5)
- A training framework for robust sequential taggers that combines **adversarial training** with **triplet loss**. (Chapter 5)

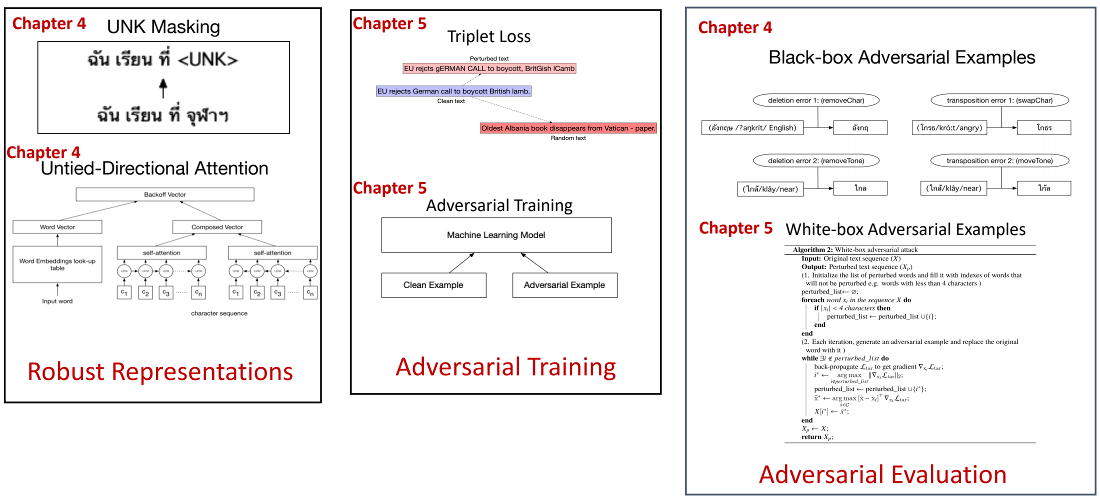

Chapter 4
UNK Masking
ฉัน เรียน ที่ <UNK>
ฉัน เรียน ที่ จุฬาฯ
Chapter 4
Untied-Directional Attention
Backoff Vector
Word Vector
Composed Vector
Word Embeddings look-up table
self-attention
self-attention
Input word
character sequence
Robust Representations
Chapter 5
Triplet Loss
Perturbed text
EU rejcts gERMAN CALL to boycott, BritGish lCamb
EU rejects German call to boycott British lamb.
Clean text
Oldest Albania book disappears from Vatican - paper.
Random text
Chapter 5
Adversarial Training
Machine Learning Model
Clean Example
Adversarial Example
Adversarial Training
Chapter 4
Black-box Adversarial Examples
deletion error 1: (removeChar)
transposition error 1: (swapChar)
deletion error 2: (removeTone)
transposition error 2: (moveTone)
Chapter 5
White-box Adversarial Examples
Algorithm 2: White-box adversarial attack
Adversarial Evaluation

Figure1.1: Thesis Overview

1.5 Publications

1. **Udomcharoenchaikit, C.**, Boonkwan, P. and Vateekul, P., 2020. Towards Improving the Robustness of Sequential Labeling Models Against Typographical Adversarial Examples Using Triplet Loss. Natural Language Engineering (NLE)[Under Review]

2. **Udomcharoenchaikit, C.**, Boonkwan, P. and Vateekul, P., 2020. Adversarial Evaluation of Robust Neural Sequential Tagging Methods for Thai Language. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 19(4), pp.1-25.

3. Jettakul, A., Thamjarat, C., Liaowongphuthorn, K., **Udomcharoenchaikit, C.**, Vateekul, P. and Boonkwan, P., 2018, July. A comparative study on various deep learning techniques for Thai NLP lexical and syntactic tasks on noisy data. In 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 1-6). IEEE.

4. **Udomcharoenchaikit, C.**, Vateekul, P. and Boonkwan, P., 2017, August. Thai named-entity recognition using variational long short-term memory with conditional random field. In The Joint International Symposium on Artificial Intelligence and Natural Language Processing (pp. 82-92). Springer, Cham

Chapter II

BACKGROUND

2.1 Characteristics of Thai language

Thai is an isolating language, and its word order is SVO (Minegishi, 2011). There is no inflectional morphology in the Thai language to encode grammatical information such as number and tense. Thai Part-of-Speech does not inflect; instead, it uses additional words or morphemes to convey such information. Orthographically, Thai has no word boundary. Also, Thai has no inflectional morphology; therefore, Thai sentences can be ambiguous both temporally and aspectually (Koenig and Muansuwan, 2005).

Thai has derivational morphology. Thai can form a complex word through affixation (Iwasaki etal., 2005). Through affixation, a complex word can be formed using a root and an affix. Usually, a root is a free morpheme; and an affix is a bound morpheme. In this research, we use this linguistics knowledge to initialize the hidden state of the character-to-word (C2W) model. The rest of this section discusses Thai affixation and examples of affixes, as mentioned in Iwasaki etal. (2005).

2.1.1 Prefixes

Prefix is a type of affix that precedes a root of a complex word. Prefixes can be categorized into modifying, classifying, noun-forming, adjective-forming, and adverbial-forming prefixes (Iwasaki etal., 2005):

- มหา- /mahaǎ/ 'great' - a modifying prefix

มหาวิทยาลัย /mahaǎ-wìttayalai/ ‘university’ < วิทยาลัย /wìttayalai/ ‘college’

- นัก- /nák/ - a classifying prefix that indicates that the noun concept belongs to a person class.

  นักเรียน /nák-rian/ ‘student’ < เรียน /rian/ ‘to study’

- การ- /kaan-/ - a noun-forming prefix which forms an activity noun with a verbal root.

  สอนการ /kaan-sˇn/ ‘teaching’ < สอน /sˇn/ ‘to teach’

- ช่าง- ‘having a characteristic of’ /chǎa/ - an adjective-forming prefixes

  คุยช่าง /chǎa-khuy/ ‘chatty’ < คุย /khuy/ ‘to chat’

- อย่าง- ‘in a manner that’ /chǎa/ - an adverbial-forming prefix which is placed before an adjective to produce an adverbial word. An English affix equivalent of this affix would be ‘-ly’.

2.1.2 Suffixes

Suffix is a type of affix that appears after a root of a complex word. In Thai, suffixes are used to form abstract or sophisticated words and they are less common than prefixes (Iwasaki etal., 2005):

- -กร /kn/ ‘agent’

  อาชญากร /áatchayaa-kn/ ‘criminal’ < อาชญา /’aatchayaa/ ‘crime’

2.1.3 What Is a Word?

Haspelmath (2011) suggests that there are four word-defining definitions: orthographic, phonological, semantic, and morphosyntactic.

- **Orthographic definition:** There are many languages in which their orthographies use spaces between words, especially languages based on the Greek, Latin, and Cyrillic alphabets. However, there are many languages without word spacing, such as Chinese, Japanese, Sanskrit, and Thai.
- **Phonological definition:** There is phonological criteria for delimiting a word; for example, words can have one main stress.
- **Semantic definition:** A word is a unit that has a reference to a meaning or a semantic concept. However, a single semantic idea can span across multiple words.
- **Morphosyntactic definition:** Words are building blocks for sentences; hence, we can apply morphosyntactic criteria to identify words.

In recent research, the Thai word segmentation standard is based on the idea of minimal integrity unit. Minimal integrity unit is a word segmentation standard proposed by Aroonmanakun (2007). Previously, the disagreement on what is considered a word makes it difficult to compare the results from multiple word segmentation systems. Aroonmanakun (2007) argues that Thai word segmentation systems should give us two types of word:

1. Simple word: A word with one morpheme (e.g.ดี-‘good’, สะพาน-‘bridge’, etc.)
2. True compound word: A word with two or more morphemes in which its meaning is immensely different to the sum of its morphemes. (e.g.หายใจ-‘to breathe’ $\neq$

หาย-‘to be lost’+ใจ-‘heart’)

The major source of the disagreement comes from long compound words; this minimalist approach can ease the disagreement on Thai word segmentation by segmenting each compound word that is not firmly bound into multiple words or minimal integrity units. In addition, these minimal integrity units can be then combined into a larger linguistic unit later on for applications that rely on larger linguistic units such as machine translation. In modern Thai corpora, such as BEST (Kosawat etal., 2009), words are segmented based on this idea of minimal integrity unit.

2.1.4 Errors in Thai Texual Data

Haruechaiyasak and Kongthon (2013) and Kriengket etal. (2017) extensively studied errors that occurred in Thai textual data. Kriengket etal. (2017) found that the errors during the word segmentation process can be divided into three categories: cognitive, unintentional, and intentional. Cognitive errors are resulted from past cognition, misunderstanding, and illiteracy of users. Cognitive errors can be divided into two categories: illiteracy and various transliteration (Kriengket etal., 2017). Unintentional errors are caused by careless typing, it is a well-studied topic in the area of word editing and optical character recognition (OCR). There are three types of unintentional typographical errors (Haruechaiyasak and Kongthon, 2013):

1. Insertion: e.g. ข้าสว (ข้าว = rice)
2. Deletion: e.g. หน้ต่าง (หน้าต่าง = window)
3. Transposition: e.g. ทำงนา (ทำงาน = to work)

Kriengket etal. (2017) found that there are three types of intentional errors:

1. Transformation: errors from intentionally changing some elements of the entry words
2. Insertion: errors from repeating the last character or the word to emphasize the feeling
3. Onomatopoeia: words that are created to imitate sounds and noises.

2.2 Sequence Processing with Statistical Sequential Models

Human languages are sequential. Every utterance depends on previous utterances to build up a context. Therefore, many NLP models are based upon sequential information. In this section, we will discuss sequential models used in NLP.

2.2.1 Temporal Probablilistic Graphical Models

Probabilistic graphical models are a graph-based framework where a graph encodes the conditional dependency between variables. One type of probabilistic graphical models is temporal model which is suitable for a system that evolves over time. This section introduces 2 well-known probabilistic temporal models in NLP applications: Hidden Markov Model (HMM) and Conditional Random Fields (CRFs).

HMM is widely used in a wide range of applications for sequential modeling. HMM is composed of two probabilistic components: the transition model that represents the transition from one state to the next state over time, and the observation model that represents the likelihood of different observations. Given a sequence of inputs, HMM computes a joint probability distribution over possible sequences of target labels and choose the label sequence with the highest probability (Rabiner, 1989).

Conditional Random Fields (CRFs) are a framework for creating probabilistic graphical models that can be used to predict sequences of target labels from sequences of input samples (Lafferty etal., 2001a). CRFs learn the context from neighboring samples to form a predictive model. The strength of probabilistic graphical models, including CRFs, lies in their ability to infer information from multiple interdependent variables. In linear-chain CRFs, the outputs are linked together to form a linear chain. Linear-chain CRFs, for a task such as NER, are composed of two main factors: one factor represents the dependency between output labels, and another factor that represents the dependency between an output and its input features. The probability distribution, which can be represented by CRFs, has the form (Koller and Friedman, 2009):

$$P\left(\mathrm{Y}|\mathrm{X}\right) = \frac{1}{Z\left(\mathrm{X}\right)}\tilde{P}\left(\mathrm{Y},\mathrm{X}\right)$$

where $\tilde{P}\left(\mathrm{Y},\mathrm{X}\right) = \prod_{i=1}^{k-1}\phi\left(Y_i,Y_{i+1}\right)\prod_{i=1}^{k}\phi\left(Y_i,X_i\right)$ and $Z\left(\mathrm{X}\right) = \sum_{\mathrm{Y}}\tilde{P}\left(\mathrm{Y},\mathrm{X}\right)$

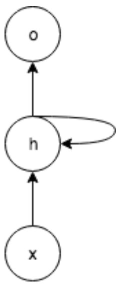
The main difference between HMM and CRFs is that CRFs are discriminative models that are trained to maximize the conditional probability instead of the joint probability of observation and state sequences (Ponomareva etal., 2007).

2.2.2 Recurrent Neural Networks

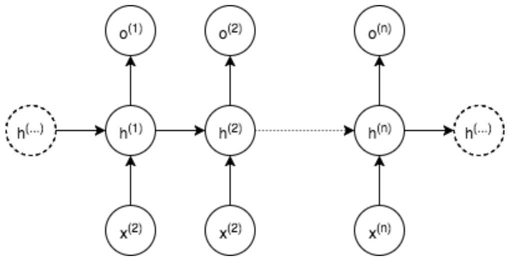
Recurrent Neural Networks (RNNs) are a type of neural networks that are designed to process sequential data (Williams and Hinton, 1986), they are very common in deep learning based NLP applications. Goodfellow etal. (2016) shows that RNNs can be expressed as follows:

$$\boldsymbol{h}^t = f\left(\boldsymbol{h}^{t-1},\boldsymbol{x}^t;\boldsymbol{\theta}\right) \tag{2.1}$$

Where $\boldsymbol{h}$ represents the state; $\boldsymbol{x}$ represents the input; t represents the timestep, and $\boldsymbol{\theta}$ represents the weight parameters for the activation function "f". RNN considers all



(a) RNN



(b) Unfolded RNN

Figure2.1: A many-to-many recurrent network. (a) a cyclic circuit RNN diagram (b) The same RNN network can be unfolded to reveal a computational graph that passes information forward through time.

the previous history by compressing them into a vector. It compresses all the history by maintaining a state vector $\boldsymbol{h}^t$ which is a function of the previous state vector $\boldsymbol{h}^{t-1}$ and the input $\boldsymbol{x}^t$. Recurrent Neural Networks can be constructed in various ways, but the core recurrence component which passes information forward through time must be included. Fig. 2.1 shows a many-to-many RNN diagram–for n timesteps input data–which is very common for word and character tagging applications. This figure shows the unrolled computational graph which we can use the backpropagation algorithm to calculate the gradient. Parameters of the RNNs are shared across time steps and the derivatives are accumulated across time. This variation of backpropagation is called backpropagation through time (BPTT).

It is important to mention that RNNs have a tendency to suffer from vanishing and exploding gradient, because the gradient is passed back through many time steps. Exploding gradient is trivial to solve by using a gradient clipping technique which clips the gradient once it exceeds a certain threshold. Two common gradient clipping techniques are norm clipping (Pascanu etal., 2013) and element-wise clipping (Mikolov etal., 2012).

Norm clipping:

$$\tilde{\nabla} \leftarrow \begin{cases} \frac{c}{\|\nabla\|}\nabla & \text{if } \|\nabla\| \geq c \\ \nabla & \text{otherwise} \end{cases} \tag{2.2}$$

Element-wise clipping:

$$\tilde{\nabla}_i \leftarrow \min\left\{c, |\nabla_i|\right\} \times \operatorname{sign}\left(\nabla_i\right), \forall i \tag{2.3}$$

LSTM is one of Recurrent Neural Network (RNN) variants. As the temporal span between dependencies grows, traditional RNNs become increasingly inefficient in term of representing such dependencies. This problem is also known as the problem of long-term dependencies (Bengio etal., 1994). LSTM architecture is designed to overcome this issue by incorporating a new structure called memory cell. LSTM adds or removes information to a memory cell by using gates to control changes to a memory cell. Therefore, LSTM is able to allow the constant error to flow through memory cells bridging long-term dependencies together (Hochreiter and Schmidhuber, 1997). The mathematical equations for LSTM are as follows:

$$\begin{aligned} i_t &= \sigma\left(w_i\left[h_{t-1}, x_t\right] + b_i\right) \\ f_t &= \sigma\left(w_f\left[h_{t-1}, x_t\right] + b_f\right) \\ o_t &= \sigma\left(w_o\left[h_{t-1}, x_t\right] + b_o\right) \\ \tilde{c_t} &= \tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right) \\ c_t &= f_t * c_{t-1} + i_t * \tilde{c_t} \\ h_t &= o_t * \tanh\left(c^t\right) \end{aligned} \tag{2.4}$$

There are three gates in the LSTM architecture: input gate i_t, forget gate f_t, and output gate o_t. These gates along with the candidate cell state $\tilde{c_t}$ are used to calculate the cell state c_t and the hidden state vector h_t. LSTM also alleviates the gradient vanishing problem by using memory cell to avoid the multiplicative effect from the BPTT algorithm, by allowing information to flow through memory cells additively. Recent research in NLP technologies such as NER has shown that LSTM can be the main component for a neural NLP system (Chiu and Nichols, 2016; Lample etal., 2016a).

2.3 Vector Embeddings

In this research, our representational techniques are based on distributional methods in which the word representation is calculated from the distribution of words around it. This idea is dated back to 1957 when J.R. Firth (1957) argued that words that occur near each other are likely to have similar meanings through his famous statement "You shall know a word by the company it keeps". The technique of this sort often represents a word as a vector of numbers. In the era of deep learning, a word vector is often dense (as opposed to sparse, where most of the elements are zero.) and often has smaller dimension comparing to the traditional representation methods. In this section, we will focus on word representation techniques used in deep learning. We also examine the use of such techniques at subword level.

2.3.1 Word Representation

It is very common in deep learning for NLP to have a word embedding layer as its core component. A word embedding layer encodes the discrete word indexes into a high-dimensional continuous vector space. Word Embedding is a group of natural language processing techniques, which is capable of learning high-quality word representations in a geometric space. Techniques such as Continuous Bag-of-Words and Skip-Gram have shown to provide the state-of-the-art performance in representing syntactic and semantic meaning (Mikolov etal., 2013a). The parameters of these word representations can also be used as pretrained weights for other deep learning-based NLP applications. The pretrained weights such as GloVe (Pennington etal., 2014) and

FasText (Joulin etal., 2016) are freely available in multiple languages, and they have been employed in many NLP applications.

2.3.2 Robust Representation for Out-Of-Vocabulary Words

Most of word-based NLP systems can recognize only words that exist in some predefined finite word dictionaries or corpora. Word-based NLP models often use “UNK” as a token that represents an unknown word. Hence, the expressivity of these models is limited to words in their training corpora. Another approach to this problem is to use subwords instead of words to represent an input text, this approach provides more flexibility and allows us to deal with a wider range of input text. The core idea of this approach is that word embeddings can be composed from character embeddings. This allows us to generate embedding for unknown words. Compositionality is the key property of embedding technique which allows us to create a larger linguistic unit form smaller linguistic units. In Mikolov et al. (2013b), the authors illustrate this property by using simple element-wise addition to combine two word vectors to form a phrase vector. For example, a phrase vector is formed by composing two word vectors, “Vietnam” and “capital”, in this experiment the authors get the phrase vector that is close to the word vector for “Hanoi”. This concept of composing smaller linguistic units to form a larger unit can also be applied to sublexical units as well.

Ling etal. (2015) proposed the character to word (C2W) compositional model based on bidirectional LSTM, this character-based model can be used to form a word representation from just characters. Ling etal. (2015) found that the C2W model can be used in NLP tasks such as language modeling and POS tagging, while still maintain a comparable performance to other state-of-the-art models.

Luong etal. (2013) observed that, in deep learning for NLP, words are often regarded as independent entities without any connection to their morphological structures. To consider morphological information into a deep learning model, Luong etal. (2013) proposed that the vector representations for morphologically complex words can be built from their morphemes. Luong etal. (2013) found that this technique is more robust to rare and out-of-vocabulary (OOV) words.

Unfortunately, Thai is not a morphologically rich language. Instead of using a morpheme-based representation, It is also possible to use compositional technique to combine syllable vectors or other subword vectors together to form a word vector representation. Since there is neither inflection nor derivation in Thai, Chunwijitra etal. (2016) proposed a syllable-based unit called pseudo-morpheme to be used as a subword unit. Furthermore, Chormai etal. (2019) uses syllable embeddings to improve the performance of a word segmentation model.

2.3.3 Contextual Representation

Instead of using static word embeddings, we can also allow word embeddings to change dynamically based on their context. Contextual word representations, such as ELMo (Peters etal., 2018) and BERT (Devlin etal., 2019), are internal states of neural networks trained with a variant of language model objective on a large text corpus. These internal states change according to both linguistic context and syntax. Using contextual representations has been shown to improve the performances of NLP systems on almost every NLP task. Contextual word representations address the problem of polysemous words and the context-dependent nature of human language.

2.4 Adversarial Examples

The accuracy of a well-trained model can be degraded with a small perturbation to an input. Inputs that are intentionally designed to reduce a well-trained model's performance are called "adversarial examples". Adversarial examples can reveal how weak our current machine learning models are. Small perturbation to an input can cause a large drop in performance (Szegedy etal., 2014; Goodfellow etal., 2015). If x' is an adversarial example of an input x for a model $f_\theta(x)$. The goal of an adversarial example can be defined with the following equations:

$$\begin{aligned} f_\theta(x) &\neq f_\theta(x') \\ f_{\text{human}}(x) &= f_{\text{human}}(x') \end{aligned} \tag{2.5}$$

The key idea is that a machine learning model, $f_\theta(x)$, should classify different labels for x and x'. While a human, $f_{\text{human}}(x)$, should classify x and x' with the same label.

Here we show an example of how an adversarial example is created. A classifier is usually trained to optimize the parameters θ to minimize loss over a training set of size N:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell\left(f_\theta(x_i), y_i\right) \tag{2.6}$$

In deep learning, this can be solved by using a gradient descent optimization algorithm. We can compute the gradient of the loss function with respect to the parameters θ; then we can update the parameters θ by taking a step proportional to the gradient:

$$\theta := \theta - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_\theta \ell\left(f_\theta(x_i), y_i\right) \tag{2.7}$$

The gradient $\nabla_\theta \ell\left(f_\theta(x_i), y_i\right)$ can be computed efficiently via back-propagation. It also tells us how the adjustment in each θ will affect the loss score. It is important to note that we can also compute the gradient of the loss function with respect to the input

x_i: $\nabla_{x_i}\ell\left(f_\theta\left(x_i\right), y_i\right)$. This shows how changes to the input x_i will affect the loss score. Here we can formulate an adversarial example generation problem as an optimization problem:

$$\max_{x'} \ell\left(f_\theta(x'), y\right) \tag{2.8}$$

Where x' is an adversarial example. We cannot optimize over x' directly without defining a constraint because this will allow us to have x', which is completely different from x, and this will defeat the purpose of creating an adversarial example. We need to ensure that an adversarial example x' is similar to the original example x. In computer vision, these adversarial examples are often indistinguishable from the original examples that they appear to be identical (Szegedy etal., 2014; Goodfellow etal., 2015). In NLP, inputs are not continuous; textual data are discrete. Any perturbation to discrete data is noticeable. The mainstream methods of adversarial examples are based on L^P-space (e.g. L_∞, L_2, L_1, etc.):

$$\left\{x' : \|x' - x\|_p \leq \epsilon\right\} \tag{2.9}$$

Here we provide an example of how we generate an adversarial example. Fast Gradient Sign Method (FGSM) is one of the first white-box adversarial attack methods introduced to the machine learning community (Szegedy etal., 2014; Goodfellow etal., 2015). Firstly, we can formulate an optimization problem to find a perturbation δ that would maximize the loss function:

$$\max_{\|\delta\|_\infty \leq \epsilon} \ell\left(h_\theta(x + \delta), y\right) \tag{2.10}$$

We define an adversarial example as follows:

$$x' = x + \delta \tag{2.11}$$

In order to maximize the loss function, we update δ in the same direction as its gradient:

$$\delta := \delta + \alpha \nabla_{\delta} \ell \left(h_{\theta}(x + \delta), y \right) \tag{2.12}$$

We also seek to constraint the magnitude of perturbation to keep perturbation imperceptible by using the L_{∞} norm; therefore, the perturbation is defined by the following set:

$$\{\delta : \|\delta\|_{\infty} \leq \epsilon\} \tag{2.13}$$

For L_{∞} norm, we can easily find the projection of δ to the norm ball by clipping the values of δ to be within the range $[-\epsilon, \epsilon]$:

$$\delta := \mathrm{clip}(\delta, [-\epsilon, \epsilon]) \tag{2.14}$$

If the step-size α is large enough, then each element of δ can either be $-\epsilon$ or ϵ depending on the sign of the corresponding element in the gradient. Hence, we can express it as:

$$\delta := \epsilon \cdot \mathrm{sign} \left(\nabla_{\delta} \ell \left(h_{\theta}(x + \delta), y \right) \right) \tag{2.15}$$

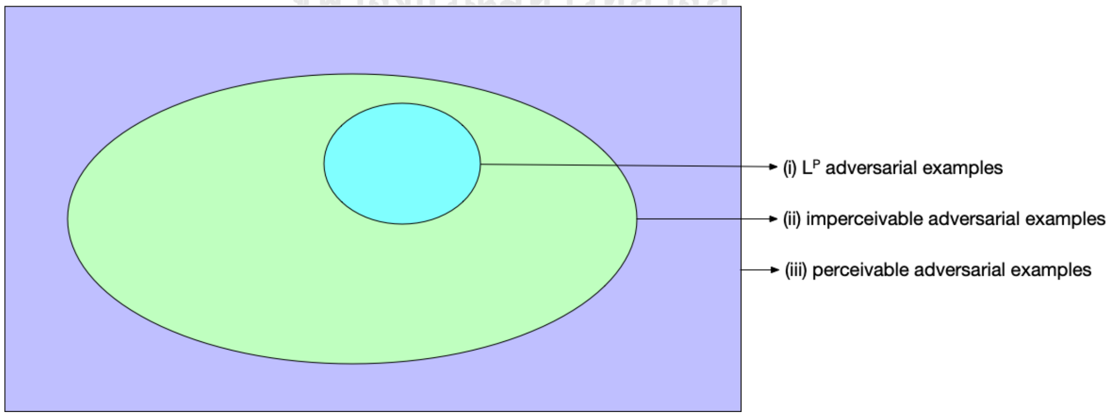

(i) L^P adversarial examples
(ii) imperceivable adversarial examples
(iii) perceivable adversarial examples

Figure2.2: $\{x' : f_{\text{human}}(x) = f_{\text{human}}(x')\}$ a set of all possible adversarial examples such that humans would give the same label to both original and perturbed examples

Fig. 2.2 shows a set of all adversarial examples where humans would give the same label. It shows that here are non-L^P methods as well. It categorizes adversarial examples into three categories: (i) L^P adversarial examples, (ii) imperceptible adversarial examples, (iii) perceptible adversarial examples. L^P adversarial examples are subsets of imperceptible adversarial examples.

Non-L^P imperceptible adversarial examples include Wasserstein adversarial examples (Wong etal., 2019), which cover standard image manipulations (e.g., translation, rotation, and scaling). If we were to shift an image one pixel to the right, the distance between the original and the perturbed images is much smaller with Wasserstein distance than L^P distance.

There are threat models where changes are perceptible to human eyes; for example, we can apply an adversarial patch to an image where a part of an image is completely masked with the patch (Brown etal., 2017). Hence, we can also apply this idea to NLP, where we allow perceptible adversarial examples.

Adversarial examples can also be categorized based on how they are generated. The equation 2.8 shows that we need access to the models' parameters to generate an adversarial example. We call this a **white-box** adversarial example because we have full knowledge of the models' parameters. The other paradigm is to generate an adversarial example without any knowledge of the model's parameters. An adversarial example generated using this paradigm is called a **black-box** adversarial example.

2.4.1 Adversarial Training

One of the simplest ways of training a model that is robust to adversarial examples is adversarial training. Adversarial training implicitly augments training data with adversarial examples. The adversarial training procedure minimizes error when the training examples are perturbed by an adversary. Also, adversarial training gives a regularization effect (Goodfellow etal., 2015).

Chapter III

LITERATURE REVIEW

The recent NLP research trend has shown that deep learning approaches have attained high performance across various NLP tasks. In In contrast to traditional machine learning methods, deep learning approaches are not dependent on domain-specific handcrafted features and external resources. These handcrafted features and resources are expensive and laborious to create; deep learning reduces the need for such features and resources by automatically learning features from its network. Despite its success, neural networks are brittle. Even small changes to an input can induce misclassification. One of the key issues is that most NLP systems are trained and evaluated on clean formal texts. They ignore the possibility of unknown words and misspelled words. This chapter focuses on sequential taggers. We discuss sequential tagging techniques for NLP applications and also review the past development of these applications in the context of Thai NLP. Then we discuss why current training and evaluation procedures do not lead to robust NLP systems. Finally, we explore the current progress in building robust NLP systems.

3.1 Current State of Sequential Tagging Models

Before the deep learning era, most sequential tagging models are linear statistical models such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Lafferty etal., 2001b; Nguyen and Guo, 2007; Ratinov and Roth, 2009). These sequential labeling models are heavily dependent on task-specific resources and handcrafted features to improve their performances. It is important to highlight that these task-specific resources and features are costly to develop and often do not apply to other tasks (Ma and Xia, 2014). In recent years, deep learning techniques have been successfully applied to linguistic sequence labeling tasks without having to rely on hand-crafted features. Current top-performing approaches often use BiLSTM-CRF as a core component in their architectures (Akbik etal., 2018; Peters etal., 2018; Straková etal., 2019; Xin etal., 2018). Transformer-based models such as Bidirectional Encoder Representations from Transformers (BERT) also achieve high performances score across multiple sequential labeling datasets (Devlin etal., 2019; Heinzerling and Strube, 2019). Other top-performing approaches use differentiable neural architecture search (NAS) to find an optimal architecture for a sequential tagging task (Jiang etal., 2019).

These deep learning architectures all have one thing in common. They were designed and benchmarked using clean test sets. Therefore, we do not know whether they are robust when presented with malformed texts.

3.2 Thai Natural Language Processing

Building a Thai NLP application is not a simple task; one must handle the Thai language's ambiguity and its unique characteristics. This section examines the past development of Thai NLP research on the three core technologies: word segmentation, POS tagging, and Named-Entity Recognition (NER). In addition, this section also discusses the implementation of deep learning models on Thai NLP tasks.

3.2.1 Word Segmentation

Word segmentation or word tokenization is one of the most fundamental tasks in Natural Language Processing. It is the first language processing step in many NLP pipelines. In many languages, word segmentation is seen as a trivial task. However, the

Table3.1: The accuracy of two dictionary-based systems vs. percentage of unknown words (Theeramunkong and Usanavasin, 2001)

Unknown word (%)	Accuracy (%)	
	Maximum Matching	Longest Matching
0	97.24	97.03
5	95.92	95.63
10	93.12	92.23
15	89.99	87.97
20	86.21	82.60
25	78.40	74.41
30	68.07	64.52
35	69.23	62.21
40	61.53	57.21
45	57.33	54.84
50	54.01	48.67

lack of explicit word boundary in the Thai language causes word segmentation to be considered challenging.

There are two main approaches for Thai word segmentation: dictionary-based and statistical-based. The early works on Thai word segmentation are based on dictionary-based approach. Poowarawan (1986) proposed a dictionary-based approach to Thai word segmentation using the longest matching algorithm. Sornlertlamvanich (1993) introduced another dictionary-based approach using the maximum matching algorithm; this approach generates all possible word segmentation results for a sentence, then it selects the results with the fewest words. However, these two dictionary-based approaches have been proved to be inferior to statistical-based algorithms such as conditional random fields (CRFs) (Haruechaiyasak etal., 2008). In addition, Theeramunkong and Usanavasin (2001) found that these dictionary-based approaches perform very poorly on unknown words as shown in table 3.1.

There are many statistical-based approaches to Thai word segmentation such as trigram Markov model (Asanee and Chalathip, 1997), decision trees (Theeramunkong and Usanavasin, 2001), syllable collocation (Aroonmanakun, 2002), conditional random fields (Kruengkrai etal., 2006), etc. Theeramunkong and Usanavasin (2001) found that dictionary-based approaches do not perform well if the dictionary is not adequate enough because it will lead to a large number of unknown words. Aroonmanakun (2002) argues that dictionary is still a crucial component for defining a word. Aroonmanakun (2002) found that word segmentation can also be done by first segmenting a text into syllables and then merging syllables into words based on their collocation strength. This approach relies on a dictionary to determine whether a sequence of syllables could be a word, using a dictionary can greatly reduce the number of possible sequences of syllables. However, Aroonmanakun (2002) also found that the performance of this approach drops drastically when there are many unknown words.

3.2.2 Part-Of-Speech Tagging

Part-Of-Speech (POS) is a category in which a word is classified to according to its grammatical function in a sentence. Part-Of-Speech Tagging is one of the fundamental tasks in NLP. It is a building block for NLP applications such as homonym disambiguation, information extraction, etc.

One of the most important corpora for building Thai POS tagger is ORCHID (Sornlertlamvanich etal., 1998). ORCHID is the first large Thai Part-Of-Speech tagged corpus which is also freely available for researchers. In addition to the ORCHID corpus, Sornlertlamvanich et al. (1998) also proposed a unified probabilistic trigram model for simultaneous word segmenting and POS tagging. As mentioned in section 3.2.1, the writing system for Thai language has no explicit word boundary indicators. Therefore, there are Thai POS tagging systems that are designed as morphological analyzers that can detect both word boundary and POS for each word. Kruengkrai et al. (2006)

proposed a Conditional Random Field framework for Thai morphological analysis. This framework allows word segmentation and POS tagging to be done simultaneously.

Boonkwan et al. (2013) found that Support Vector Machines (SVMs) and Conditional Random Fields (CRFs) models can be improved by statistically retagging ambiguous POS tags using locally trained tagger which only focuses on POS tags with high ambiguity. Note that in this experiment, Boonkwan et al. (2013) assumed that the words in the corpus are perfectly segmented.

3.2.3 Named-Entity Recognition

Named-Entity Recognition (NER), a task of locating and identifying named entities into pre-defined categories, is essential to many downstream NLP applications. In previous research, NER systems often rely on handcrafted resources such as gazetteers, etc. These resources are expensive to create. They require a large amount of work from linguists and domain experts. Many difficulties in Thai NER are caused by the characteristics of Thai language. Thai orthography does not have any special character that segregates named-entity from other word types. For example, in English orthography, the first letter of an named-entity is an uppercase letter. It is harder to define word boundary in Thai due to the lack of delimiter between words in Thai orthography. In addition, a named-entity in Thai can be very long and can be composed of multiple morphemes. e.g.“คณะกรรมการกิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคมแห่งชาติ” (National Broadcasting and Telecommunications Commission). Despite many challenges, Thai language also contains clue words for named-entity such as “องค์กร” (organization), “สำนักงาน” (office), “นาย” (mister), etc. Therefore, Thai NER is more dependent on external resources provided by linguists and specialists.

During 2000’s, there were multiple works focusing on Thai named-entity recognition using machine learning approach such as Maximum Entropy (ME) (Chanlekha and Kawtrakul, 2004), Support Vector Machine (SVM) (Suwanno etal., 2007), Conditional Random Fields (CRFs) (Tirasaroj and Aroonmanakun, 2009) (Tirasaroj and Aroonmanakun, 2011), etc. They rely on specialized knowledge resources to perform NER task. Tirasaroj and Aroonmanakun (2009) proposed a Thai NER system using CRFs, in their work they employed domain-specific resources such as dictionaries with entity names and keyword list for each type of named entities. In section 3.2.4, we will introduce you to deep learning literature for Thai NLP; The NLP systems in these works can perform relatively well without domain-specific resources.

3.2.4 Deep Learning in Thai Natural Language Processing

The rise of deep learning has led to breakthroughs in many areas. In Thai NLP, open-source projects, such as CutKum (Treeratpituk, 2017) and DeepCut (Kittinaradorn etal., 2017), have brought attention to deep learning for Thai NLP. The DeepCut word segmentation library by Kittinaradorn et al. (2017) has obtained f1-score of 98.1 by using convolution neural networks (CNN) without any domain-specific resource such as a dictionary. Chormai etal. (2019) introduce AttaCut–a dilated CNN model with syllable embeddings. It has shown to be 5.6 times faster than DeepCut. Limkonchotiwat etal. (2020) further enhances DeepCut and AttaCut for domain adaptation by using a stacked ensemble learning method.

In recent research, Boonkwan and Supnithi (2017) proposed a bidirectional gated recurrent neural networks model for joint word segmentation and POS tagging. This model does not separate word segmentation and POS tagging into two steps but it is a unified model that learn these two tasks simultaneously. Boonkwan and Supnithi (2017) argue that information for POS level also helps constrain word segmentation. Without using any domain-specific features other than words, characters, and character

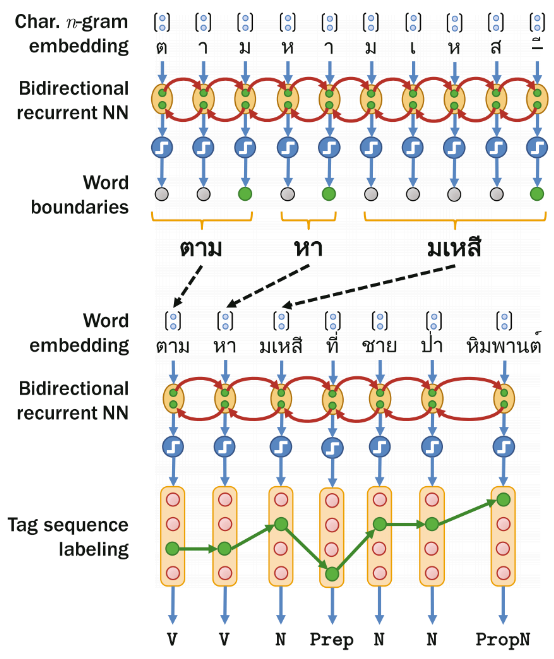


Figure3.1: The dynamic deep network for joint word segmentation and POS tagging tasks (Boonkwan and Supnithi, 2017)

n-grams; this model has obtained high f1-score on both word segmentation and POS tagging tasks on the ORCHID corpus. In addition, they also found that their model can also cope with the out-of-vocabulary (OOV) issue.

In the domain of Thai NER, Udomcharoenchaikit etal. (2017) proposed a deep learning model that combines LSTM with a CRF output layer and a variational inference-based dropout as shown in figure 3.2. This model can be used to train a high-quality NER system without resorting on domain-specific resources.

3.3 Behavioral Analyses: The Rise of Challenge Sets

Most test sets in the standard evaluation procedure aim to benchmark system performance in the average case. While the average-case evaluation is popular, it rewards models that perform well on frequent cases and overlooks cases that may not occur frequently. These cases represent weaknesses that the current NLP systems are facing.

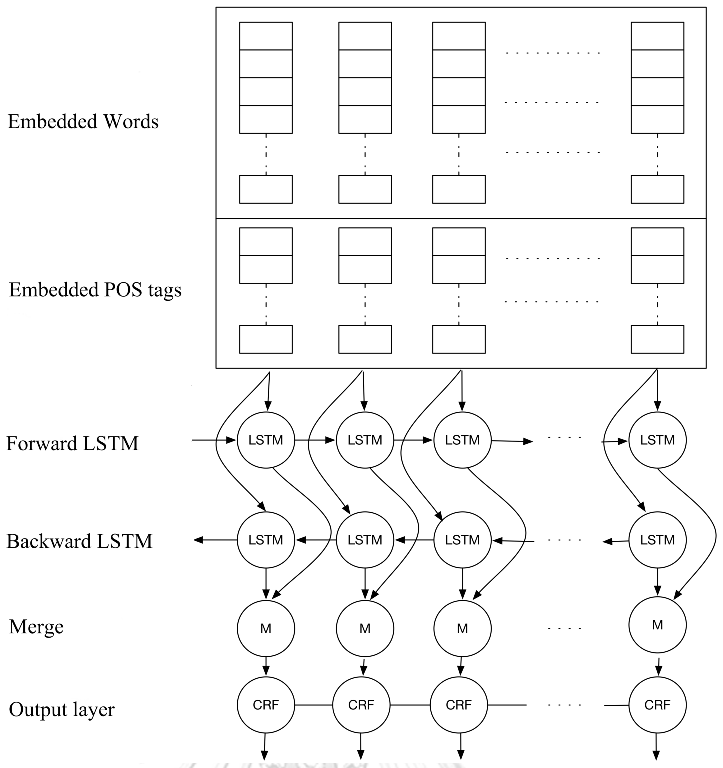


Figure3.2: Variational LSTM-CRF model for Thai Named-Entity Recognition

We can design test sets that allow us to measure the performance on specific cases systematically. An evaluation framework with test sets that focus on specific linguistic phenomena or "challenge sets" have been introduced to the NLP community for a long time (King and Falkedal, 1990; Lehmann etal., 1996). Lehmann etal. (1996) show the four main key properties of this alternative evaluation framework: (i) *systematicity*, (ii) *control over data*, (iii) *inclusion of negative data*, and (iv) *exhaustivity*. A recent revival of "challenge sets" is due to the fact that current evaluation schemes start to lose their effectiveness as the NLP systems continue to improve. The standard average-case evaluation scheme does not reveal the models' performance on specific phenomena, making it harder to diagnose and improve performance.

Recent research uses challenge sets to benchmark the performance of NLP systems on specific linguistics phenomena. Sennrich (2017) introduce a challenge set with contrastive translation pair. Each pair includes a correct reference translation and its parallel contrastive version, which is perturbed to induce translation error. They benchmark the NMT systems by measuring how many times they assign a higher score to an original translation than a contrastive translation. They select linguistic phenomena that are known to be hard for English-German NMT systems: (i) noun phrase agreement, (ii) subject-verb agreement, (iii) separable verb particle, (iv) polarity, and (v) transliteration. Linzen etal. (2016) focus only on subject-verb agreement to test whether LSTMs can learn syntax-sensitive dependencies. Burlot and Yvon (2017) introduce a machine translation challenge set to test morphological competence of NMT systems. Burlot and Yvon (2017) benchmark the robustness of NMT systems when presented with morphological alterations. Robustness to typos is another capability that researchers design challenge sets to benchmark (Belinkov and Bisk, 2018; Rychalska etal., 2019). They show that high-performance models are not robust against spelling errors. Character perturbations such as swap, substitution, deletion, and insertion are used to generate test samples for their challenge sets.

There are also works that cover a large number of linguistic phenomena. Burchardt etal. (2017) introduce a test suite with 120 linguistic phenomena to analyze the strengths and weaknesses of NMT systems. Ribeiro etal. (2020) introduce “CheckList” –a task-agnostic evaluation framework for NLP systems. CheckList can generate a large number of diverse test cases, including negation, synonyms, antonyms, typos, ability to handle logic, etc.

We can also construct challenge sets automatically. One of the frameworks for automatically constructing a challenge set is adversarial examples. In the next section, we discuss recent research in adversarial examples for NLP and techniques to improve the robustness on adversarial examples.

3.4 Adversarial Robustness for NLP

Previous literature on adversarial examples has exposed brittleness in machine learning systems by showing that subtle changes to the input can lead to failures in prediction outcomes (Szegedy etal., 2014; Goodfellow etal., 2015). While adversarial examples are more common in computer vision, there is a growing literature on adversarial examples in the NLP domain. For instance, Miyato etal. (2017) construct adversarial examples by using perturbation on continuous word embeddings instead of discrete textual inputs. A problem with this approach is that it ignores the discrete nature of textual data.

In order to apply perturbation to the textual input, Jia and Liang (2017) evaluate the robustness of question-answering systems by adding a generated sentence to distract systems without changing the correct answer. Gao etal. (2018) introduce a black-box attack for text classifiers by injecting small misspelling errors into a text sequence. Belinkov and Bisk (2018) reveal weaknesses of character-based neural machine translation (NMT) models by evaluating them on malformed texts. Belinkov and Bisk (2018) use rule-based synthetic spelling errors and natural spelling errors to evaluate the model. Ebrahimi etal. (2018) investigate white-box adversarial examples; they trick a character-level model by generating adversarial examples by choosing character-edit operations, such as deletion, insertion, and substitution ("flip"), based on the gradients of the target model. Michel etal. (2019) enforce constraints on adversarial examples so that they are meaning-preserving on the source side. Wallace etal. (2019) propose input-agnostic universal adversarial triggers with white-box access to a model's parameters. It studies adversarial attacks to help analyze and diagnose NLP systems. Furthermore, Gardner etal. (2020) propose a new evaluation paradigm by creating contrast sets. A contrast set consists of test instances that are manually perturbed by domain experts to change their gold labels. This contrasts with adversarial examples, where inputs are perturbed to change the model's decisions, but their gold labels do not change.

To overcome these weaknesses, previous literature shows that NLP systems with subword-level or character-level embeddings can generalize to out-of-vocabulary words (Ling etal., 2015; Sennrich etal., 2016). Moreover, subword embeddings and character-level embeddings can help avoid filling up word-level embeddings with a large number of vocabulary. However, NLP systems with subword-level or character-level embeddings are often trained on clean data, and their performance drops drastically when they encounter misspelled words (Belinkov and Bisk, 2018). Piktus etal. (2019) alleviate this issue by training subword embeddings to be robust against typographical errors by adding the spell correction loss function to the FastText (Bojanowski etal., 2017) loss function, which allows misspelled data to be incorporated. They benchmark the robustness of their models on malformed texts by perturbing words in the test set. Their perturbations are based on spelling errors collected from a search engine's query logs, which are only publicly available in English. This is not applicable to other languages without such resources. Instead, Belinkov and Bisk (2018) artificially inject a training set with misspelled variants of each word. This technique of increasing robustness by training on noisy data is called adversarial training.

Adversarial training (Goodfellow etal., 2015) is a standard method for improving robustness against adversarial examples. It improves the robustness of a model by training on both unperturbed original examples and perturbed examples. Liu etal. (2020) improve robustness by including noisy sentences in the training process and using loss function to constraint the similarity between clean and perturbed representations. For sequential tagging tasks, Yasunaga etal. (2018) improve the BiLSTM-CRF model's robustness on infrequent and unseen words using adversarial training method. They generate adversarial examples by adding small perturbation to continuous word embeddings. Pruthi etal. (2019) overcome adversarial misspellings on text classification tasks by using a word recognition model. Liu etal. (2020) use character embeddings, the adversarial training method, and similarity constraint to improve robustness against character-level adversarial examples on text classification tasks.

Previous literature on robust sequential taggers focuses on improving the performance of sequential taggers on out-of-distribution texts. They rely on adversarial training methods to improve their models' robustness by perturbing the embeddings in continuous space instead of perturbing the input texts (Yasunaga etal., 2018; Zhou etal., 2019). Yasunaga etal. (2018) conduct part-of-speech tagging experiments on multiple languages on the Pen Treebank WSJ corpus (English) and the Universal Dependencies dataset (27 languages). Yasunaga etal. (2018) find that adversarial training improves overall accuracy. Also, adversarial training alleviates over-fitting in low resource languages, and it also increases part-of-speech tagging accuracy for infrequent and unseen words. Yasunaga etal. (2018) benchmark the robustness of the models on rare and unseen words. Zhou etal. (2019) focus on NER task. They address the data imbalance issue by incorporating label statistics to the CRF loss and combine it with focal loss. Zhou etal. (2019) benchmark the robustness of the models on user-generated data. Bodapati etal. (2019) improve robustness to capitalization errors in NER by using data augmentation. Their data augmentation technique augments the training set with sentences with all upper-cased characters and sentences with all lower-case characters. This allows a model to learn to exploit or ignore orthographic information depending on the contextual information.

3.5 Concluding Remarks

The review of literature in this chapter has mainly concentrated on statistical methods in NLP. This chapter illustrates the development of Thai NLP research and the limitations of traditional machine learning methods in Thai NLP, such as the reliance on handcrafted features. Deep learning approaches have been shown to overcome the limitation of traditional machine learning methods and have obtained state-of-the-art results. It also identifies the lack of focus on robustness in the current training and testing paradigm. It then reviews the current research on behavioral analyses that proposes alternative evaluation frameworks for benchmarking various linguistics capabilities of the NLP systems. Finally, it discusses the current progress to improve and benchmark the robustness of sequential tagging tasks. Thus this chapter provides a basis for developing robust deep learning models for NLP tasks for this dissertation.

Chapter IV

ROBUST WORD REPRESENTATIONS AND BLACK-BOX ADVERSARIAL EVALUATION FOR THAI

4.1 Introduction

This chapter is a slightly modified version of “Adversarial Evaluation of Robust Neural Sequential Tagging Methods for Thai Language” published in the ACM Transactions on Asian and Low-Resource Language Information Processing.

Tagging sequences of labels to sequences of inputs is a ubiquitous task in NLP. Tasks such as part-of-speech (POS) tagging, named-entity recognition (NER), and shallow parsing are examples of sequence labeling tasks. In low-resource settings, these tasks can provide important syntactic clues for downstream applications.

In recent years, recurrent neural models such as long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) have been the core component of many sequential tagging applications. These neural models have reached state-of-the-art performance and outperformed the conventional models without relying on external domain knowledge (Huang etal., 2015). Despite promising results on clean formal texts annotated by domain experts, most literature does not evaluate their models’ robustness on texts that are informally written, which contain a more substantial amount of unknown words and misspelled words.

Standard sequential taggers fail to perform well on non-standard text such as microblogs with an error rate up to ten times higher than on a standard text such as newswire (Derczynski etal., 2013). Microblogs (e.g., Twitter) are known to contain a higher number of unknown and misspelled words (Ritter etal., 2011). For an NLP system to be robust, it must be able to address issues such as out-of-vocabulary (OOV) and spelling mistakes.

One way to curtail the performance drop is to train the model on a new training set from a new domain, but it is very laborious to create a new corpus. Another approach is lexical normalization – an additional preprocessing step to produce a more standard text input. Lexical normalization often requires language-specific resources.

We introduce a new adversarial evaluation scheme for the Thai language, in which we create adversarial samples based on unintentional mistakes. We propose a training strategy and a neural representation technique that are robust against OOV words. The proposed method does not require a large amount of external language-specific resources other than a list of affixations.

The contributions of this chapter are as follows:

1. **Adversarial Evaluation:** We propose an adversarial evaluation scheme for Thai to evaluate models' robustness against unintentional typological errors (section 4.2). We compare our proposed methods against two baseline models: Bidirectional LSTM (BiLSTM) and BiLSTM with Embeddings from Language Models (BiLSTM-ELMo).

2. **UNK Masking:** We introduce a training data perturbation technique which significantly improves the robustness of the neural networks (section 4.3.2).

3. **Condition Initialization with Affixation Embeddings:** We investigate an initial hidden state initialization strategy with linguistic knowledge (section 4.3.1).

4. **Untied-directional Self-Attention:** We propose an untied-directional self-

attention mechanism that increases the interpretability of the neural networks (section 4.3.1).

4.2 Adversarial Evaluation

There are many NLP literature and applications with promising results on standard test sets. However, in many situations, their proposed methods fail to perform well. For an NLP system to be robust, it must be able to address issues such as Out-Of-Vocabulary (OOV) and spelling-mistake. In this chapter, we propose an adversarial evaluation scheme for Thai. To determine whether existing models are robust against OOV words and spelling-mistake, we introduce adversaries by altering the test samples' characters.

4.2.1 Thai Spelling Error

Kriengket etal. (2017) investigate spelling mistakes in Thai by analyzing frequently unmatched search queries on LEXiTRON, a popular online English-Thai dictionary. They have classified the errors found on LEXiTRON into three categories: cognitive errors, intentional errors, and unintentional errors.

- Cognitive errors are caused by ignorance of correct spelling knowledge.
- Intentional errors are deliberate misspellings or non-standard spellings of words with an intent to intensify meaning, emotion, or to express informal speech patterns.
- Unintentional errors are caused by carelessness. They can be categorized into two subtypes: typing mistakes and typographical errors.

  - typing mistakes: this type of error occurs when pressing wrong nearby keys on a keyboard or misusing a keyboard (e.g., using a wrong input language, pressing the ‘shift’ key accidentally).
  - typographical errors: this type of error is caused by carelessness when typing. The three typographical errors subtypes are insertion, deletion, and transposition.
    - insertion error: additional characters are added to the word.
    - deletion error: characters are missing from the word.
    - transposition error: characters are typed in the wrong order.

Kriengket etal. (2017) found that cognitive errors (59.7%) are the most common, followed by unintentional errors (33.5%), and intentional errors (6.8%).

4.2.2 Thai Adversarial Sample Generation

Character-level adversarial examples from previous work in English (Ebrahimi etal., 2018; Belinkov and Bisk, 2018) do not have language-specific constraints. In this work, we introduce a rule-based adversarial example generator based on known Thai spelling errors. We generate unintentional errors mentioned in work done by Kriengket etal. (2017) to produce adversarial samples. Unintentional error is the second most frequent incorrect spelling error; it consists of 33.5% of an overall error on LEXiTRON, according to Kriengket etal. (2017). Carelessness, rather than users’ illiteracy, causes unintentional errors. An unintentional typographical error is easily simulated using character edit operations, such as deletion and transposition. Typographical errors consist of 47.1% of all unintentional errors. In our adversarial evaluation scheme, Algorithm 1 generates four types of unintentional typographical errors as follows:

deletion error 1: (removeChar) remove the last character of the word if the word has more than two characters.

deletion error 2: (removeTone) remove all tonal characters.

transposition error 1: (swapChar) swap the last two characters of the word if the word has more than two characters and one of the last two characters is a consonant.

transposition error 2: (moveTone) move the first tonal character one position toward the left.

Algorithm1: Thai adversarial sample generator

```
Data: input word
Result: adversarial word
initialization;
r ⟵ sample a number between 0 and 1 from a uniform distribution;
if r<stressRate then
    if No tonal character then
        word ⟵ randomly select between removeChar and swapChar;
    else
        word ⟵ randomly select between all adversarial choices;
```

Fig. 4.1 shows possible adversarial examples generated for the adversarial evaluation scheme. Character-level adversarial examples often do not alter a text's meaning, while word-level alterations are more likely to change the meaning of a text (Ebrahimi etal., 2018). Human readers are likely to correctly inferred the meaning of a text after a small number of character alterations (Rawlinson, 1976). To further test whether humans can correctly infer textual samples from our adversarial evaluation scheme, we sampled 100 text samples from one of the perturbed BEST 2010 test sets, and we asked 3 participants to correct them. Some of the perturbed text samples can be found in Appendix A.1. The scores of 98, 96, and 100 were scored respectively by each participant. Therefore, this evaluation scheme can be used to evaluate the robustness of our models against adversarial attacks.

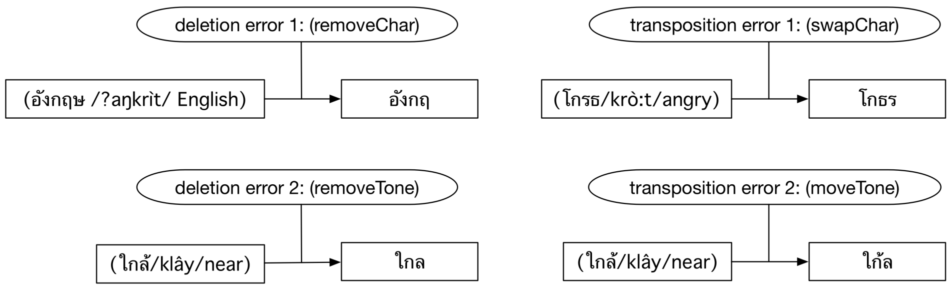

deletion error 1: (removeChar)
(อังกฤษ /ʔaŋkrìt/ English)
อังกฤ
transposition error 1: (swapChar)
(โกรธ/krò:t/angry)
โกธร
deletion error 2: (removeTone)
(ใกล้/klây/near)
ใกล
transposition error 2: (moveTone)
(ใกล้/klây/near)
ใก้ล

Figure4.1: Thai adversarial examples based on 4 types of typographical errors

4.3 Model Architecture

We introduce a model that exploits character-level information to improve its robustness against OOV words, and we also include a common linguistic characteristic, derivational affixation, which is found across multiple isolating languages to experiment whether it can improve the performance of our model. We introduce BiLSTM-U-BCAD and its implementation details in this section. We first discuss the input representations, then we cover each component of the model architecture.

4.3.1 Input Representation

This research experiment with a standard word representation that does not include character-level information and several variations of backoff representation to enhance input representation with character-level information.

Word Representation

All deep learning models for NLP tasks have a word embeddings layer as its core component. A word embedding layer maps each discrete word index to its corresponding word vector. Each word is represented by a vector $w_i \in IR^{d_w}$. Word representations are connected to the loss function, which is used to adjust the parameters of the model;

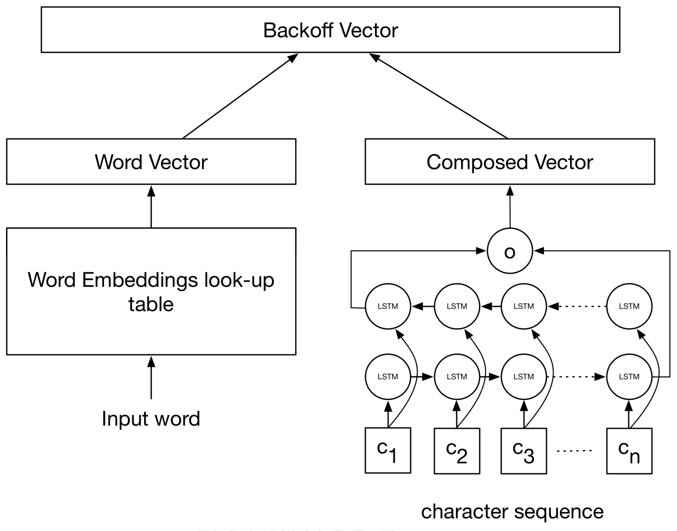


Figure4.2: Backoff Representation

these parameters are optimized through back-propagation

We assign a special dummy token (UNK) to represent words that do not occur in the training corpus and words that appear less than or equal to two times in the training corpus. We can lose valuable information by mapping these words to the UNK token. In the following section, we discuss a representation method to enhance rare and unknown words' representativeness.

Backoff Representation

Fig. 4.2 shows the backoff input representation of this model. The backoff representation is a concatenation between word representation and character-to-word representation. This is the input representation that is fed into the core Bi-LSTM component; we can define the input representation as:

$$\hat{w} = w_i \oplus w_{c2w,i}$$

We use the character-to-word (C2W) compositional model based on bidirectional LSTM proposed by Ling etal. (2015) to create the character-to-word representation.

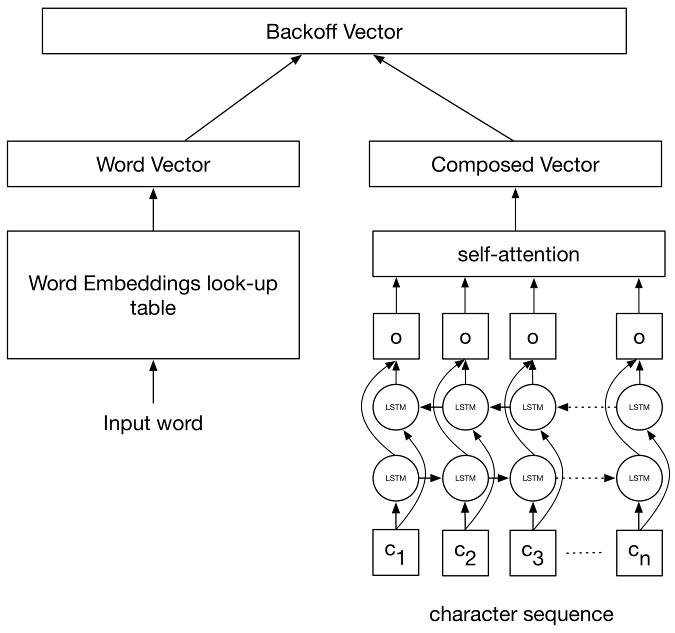


Figure4.3: Backoff Representation with self-attention mechanism

When the model encounters an unknown word, it still has the information from the character sequence to represent a word.

Backoff Representation with self-attention mechanism

We also experiment with a self-attention mechanism to focus on a specific sequence of characters to create character-to-word representations. In order to create a character-to-word compositional component, a character sequence of a word is fed into a separate Bi-LSTM component. Each character is represented by a vector $c_i \in IR^{d_c}$. The hidden dimension of the Bi-LSTM component is d_{c2w} for each direction. After all the characters are processed by the Bi-LSTM component, the last LSTM cells output of each direction are concatenated together to obtain a character-to-word vector representation $w_{c2w,i} \in IR^{2d_{c2w}}$.

Bi-directional LSTM is the core component of many state-of-the-art NLP models;

however, it is known that sequential recency is not the right inductive bias for NLP tasks; therefore additional mechanisms such as attention mechanism are included to create a direct connection back in time. Lin etal. (2017) propose a self-attention mechanism for extracting an interpretable sentence embedding from a variable length sentence. The self-attention mechanism is placed on top of the bidirectional LSTM (Lin etal., 2017):

$$\overrightarrow{h_t} = \overrightarrow{LSTM}\left(w_t, \overrightarrow{h_{t-1}}\right) \tag{4.1}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}\left(w_t, \overleftarrow{h_{t-1}}\right) \tag{4.2}$$

where each $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are concatenated together to obtain a hidden state:

$$H = (\mathrm{h}_1, \mathrm{h}_2, \cdots \mathrm{h}_\mathrm{n}) \tag{4.3}$$

Then the self-attention mechanism takes the whole LSTM states H as input, and outputs a vector of attention scores a:

$$\mathrm{a} = \mathrm{softmax}\left(\mathrm{w}_{\mathrm{s}2} \tanh\left(W_{s1} H^T\right)\right) \tag{4.4}$$

Where the shape of a weight matrix W_{s1} and a weight vector $\mathrm{w}_{\mathrm{s}2}$ are not dependent on the length of the input. Therefore, this self-attention mechanism is able to encode a variable length input into a fixed size embedding.

Backoff Representation with untied bidirectional self-attention mechanism

In this research, we apply self-attention mechanism to attend on a part of the character sequence instead of sentence. In addition, we also propose an untied bidirectional self-attention mechanism to focus on a specific sequence of characters to create character-to-word representations.

An untied bidirectional self-attention mechanism has two separated self-attention mechanisms for the forward LSTM hidden states and the backward LSTM hidden

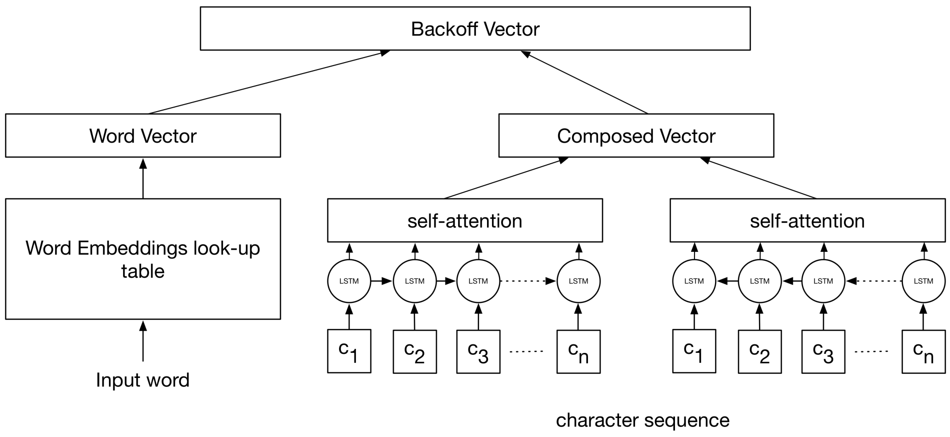


Figure4.4: Backoff Representation with untied bidirectional self-attention mechanism

states:

$$\overrightarrow{H} = \left(\overrightarrow{h_1}, \overrightarrow{h_2}, \cdots \overrightarrow{h_n}\right) \tag{4.5}$$

$$\overleftarrow{H} = \left(\overleftarrow{h_1}, \overleftarrow{h_2}, \cdots \overleftarrow{h_n}\right) \tag{4.6}$$

$$\overleftarrow{\mathrm{a}} = \mathrm{softmax}\left(\overleftarrow{\mathrm{w}_{s2}} \tanh\left(\overleftarrow{W_{s1}} \overleftarrow{H^T}\right)\right) \tag{4.7}$$

$$\overrightarrow{\mathrm{a}} = \mathrm{softmax}\left(\overrightarrow{\mathrm{w}_{s2}} \tanh\left(\overrightarrow{W_{s1}} \overrightarrow{H^T}\right)\right) \tag{4.8}$$

Where $\overrightarrow{\mathrm{a}}$ and $\overleftarrow{\mathrm{a}}$ represent a vector of attention scores for the the forward LSTM hidden states and a vector of attention scores for the backward LSTM hidden state, respectively.

The intuition behind this attention mechanism is that the forward component can capture the LSTM output after it processed a certain length of characters to form a representation for the first few characters to mimic a representation of prefix. The backward component relies on the same intuition, but for attending on a suffix. Fig. 4.4 shows an untied bidirectional self-attention mechanism as a C2W component of a backoff representation.

Affixation Representation and Conditional Initialization Weng etal. (2017) propose that instead of initializing the initial hidden state of RNN to be a zero vector, they replace the zero vector with word prediction mechanism to control the values of the initial state. In this research, we replace the initial zero vector with affixation embeddings to include linguistics knowledge into our neural networks.

Affixation Embeddings only contains two vectors one for words with an affix, and another vector for words without affix. We only use these vector to replace the initial hidden state of LSTM for character-to-word (C2W) component. The following is the list of Thai affixes used in this research:

- การ- /kaan-/ - a noun-forming prefix which forms an activity noun with a verbal root.
- ความ- /khwaam-/ - a noun-forming prefix which forms an abstract from an adjectival or verbal root.
- นัก- /nák/ - a classifying prefix that indicates that the noun concept belongs to a person class.
- ผู้- /phū-/ - a classifying prefix that indicates that the noun concept belongs to a person class.
- ไอ้- /ai/ - a classifying prefix that indicates that the noun concept belongs to a male class.
- -กร ‘agent’ /kn/ - a suffix
- ที่- /thi-/- a noun-forming prefix
- อย่าง- ‘in a manner that’ /chǎa/ - an adverbial-forming prefix
- แบบ- /baaep/ - an adverbial forming prefix

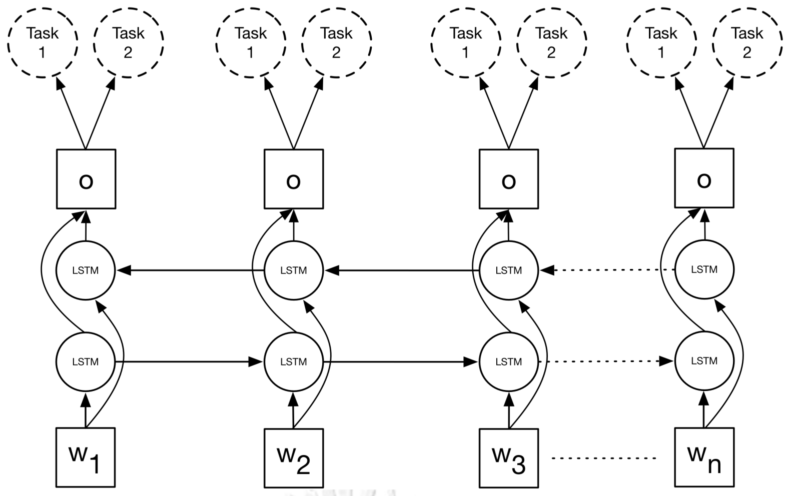


Figure4.5: Multi-task Bidirectional LSTM Model Architecture

- โดย- /dooy-/ - an adverbial forming prefix
- น่า- ‘inducing to’ /nǎa-/ - an adjective forming prefix

These affixes often appear next to the root of a complex noun word. They are a strong indication of a noun part-of-speech class. This list of affixes can also be augmented with more affixes.

4.3.2 Model Architecture

Long Short Term Memory (LSTM): LSTM (Hochreiter and Schmidhuber, 1997) is a deep learning model that is capable of learning long sequential data. Bidirectional architecture, which connects forward and backward layers increases, the amount of input information for the output layer. When two hidden layers of opposite directions are connected together, the output layer can adopt information from both previous and future time-steps. The forward LSTM hidden output $\overrightarrow{h}$ and the backward LSTM hidden output $\overleftarrow{h}$ are merged together by concatenation. The forward and backward hidden outputs, $\overrightarrow{h}$ and $\overleftarrow{h}$, are both of size d_h dimensions. The concatenated hidden output has $2d_h$ dimensions.

UNK Masking: We propose a novel technique to improve the robustness of our models. Unlike Word Dropout (Iyyer etal., 2015), UNK Masking does not exclude the selected words entirely. Usually, during the training phase, rare words are dealt with by assigning a special “UNK" token to represent all words that appear less than N times in the training set. Hence, for models with C2W representation, the models need to learn to infer from the input character sequence for rare words. In order to cover more cases than just rare words, we randomly mask 10% of all input words with UNK tokens to enrich the representativeness of UNK and augment the number of unique inputs during the training phase.

Multi-task: Our model is capable of generating outputs for multiple tasks without losing significant accuracy. We combine losses from n tasks in the following fashion:

$$\mathcal{L}\left(total\_loss\right) = \sum_{i=1}^{n} \mathcal{L}\left(task\left(i\right)\right) \tag{4.9}$$

4.4 Experiments

In order to study the effectiveness of our methods, series of experiments are conducted by adding one more component incrementally to the model at a time. List of models experimented in this research are shown in Table 4.3. Experiments were conducted on 2 Thai corpora. In this section, we discuss the training method and the datasets used in this research. Then we discuss the results of the experiments.

4.4.1 Training

For all cases, we set the dimension for each component as follows: 100 for character embeddings, 128 for word embeddings, 128 for affixation embeddings (64 for each direction), and 64 hidden dimensions for each direction of the LSTM. We update the learnable parameters by calculating the gradients from the negative log likelihood loss score using the back propagation algorithm, we use Adam (Kingma and Ba, 2015) with the learning rate of 0.001 to optimize the parameters. For each model, we train for 40 epochs and select the set of parameters of an epoch that gives the best performance on the validation set. When gradient clipping is applied, the maximum norm threshold of the gradients is set to 50.

The implementation details of BiLSTM-ELMo, the strong baseline, is as follows: we use a modified version of small ELMo model[1] where the LSTM hidden size is reduced from 1024 to 512 dimensions, and the output size is reduced from 128 to 64 dimensions (for each direction) to make this experiment feasible on our machine and make the embedding size comparable to other models. Two sets of parameters are pretrained separately on each corpus for 40 epochs. We do not finetune the parameters of ELMo during the training phase.

4.4.2 Datasets

We demonstrate the effectiveness of our proposed methods by experimenting on two Thai corpora. We use a fully annotated version of BEST2010, and ORCHID. Table 5.2 shows the details of corpora used in this chapter.

Table 4.2 shows number of tokens for each type of errors in the perturbed test sets. Stress level refers to the percentage of "stressRate" as shown in algorithm 1. Note that there are tokens that contain no perturbation when the stress level is at 100, because we do not perturb tokens that are numbers or non-Thai characters (e.g. white

[1]The original configuration for the small ELMo model can be found in the following link: `https://allennlp.org/elmo`. The configuration of the modified version can be found in the following link: `https://github.com/c4n/elmo_th`.

Table4.1: Data Statistics. † - number of unique words after pruning low-frequency words. We only included words that appear more than two times in the vocabulary; we replaced low-frequency words with “UNK" tokens.

	# of text files	# of words	# of vocab.	# of vocab. (>2)†	# of char.	avg. char/word
BEST2010						
train	4976	2,814,970	54,351	18,928	11,685,975	4.15
val	264	132,612	-	-	545,685	4.11
test	249	227,472	-	-	929,064	4.08
ORCHID						
train	132	269,707	14,459	4,801	1,167,359	4.33
val	15	27,733	-	-	119,848	4.32
test	17	45,193	-	-	198,420	4.39

Table4.2: Perturbed Test Sets

stress level	no perturbation	removeChar	swapChar	removeTone	moveTone
BEST2010					
level=20	233,688	18,896	16,344	427	426
level=40	197,545	37,538	32,978	874	846
level=60	161,358	56,405	49,534	1,227	1,257
level=80	125,422	75,087	65,996	1,646	1,630
level=100	89,592	93,659	82,471	2,033	2,026
ORCHID					
level=20	39,320	3,066	2,615	91	101
level=40	33,550	5,986	5,266	202	189
level=60	27,690	9,036	7,891	299	277
level=80	21,703	12,100	10,629	394	367
level=100	15,997	15,139	13,087	484	486

space, punctuation mark). Furthermore, adversarial sample generation functions for “removeChar" and “swapChar" do not perturb a token unless it has more than two characters. “swapChar" does not perturb a token unless one of the last two characters is a consonant.

BEST2010

Benchmark for Enhancing the Standard of Thai language processing, also known as BEST2010, was a competition held by National Electronics and Computer Technology Center (NECTEC) in 2010 to find the best Thai word segmentation algorithm. NECTEC also released a dataset for the competitors which contains word segmentation boundaries and location of each named-entity. The full version of this dataset also contains a predefined category of each named entity as well as a POS tag for each word. The full version of BEST 2010 dataset can be obtained from NECTEC for research purposes. No sentence boundary is provided in this corpus.

- Task 1: Part-of-Speech Tagging
- Task 2: Named Entity Recognition

ORCHID

ORCHID is a Thai part-of-speech tagged corpus developed by researchers from National Electronics and Computer Technology Center (NECTEC) in Thailand and Communications Research Laboratory (CRL) in Japan (Sornlertlamvanich etal., 1998). Unlike BEST2010, text in ORCHID is separated into sentences manually guided by our their own standard of sentence structuring. Unlike BEST2010, the texts in ORCHID only contain technical papers appeared in the the proceedings of the National Electronics and Computer Technology Center (NECTEC) annual conferences. Since ORCHID is only annotated with POS tags; in order to make multi-task model architecture possible to train on ORCHID, we add an auxiliary task –content and function words classification.

Table4.3: List of models experimented in this research

Abbreviation	**Description**
BiLSTM	Bidirectional Long Short-Term Memory (Weak Baseline)
+ELMo	BiLSTM with ELMo (Strong Baseline)
+U	BiLSTM with UNK masking
+U-B	BiLSTM-U with backoff representation
+U-BC	BiLSTM-U-B with conditional initialization
+U-BA	BiLSTM-U-B with self-attention mechanism
+U-BCA	BiLSTM-U-BC with self-attention mechanism
+U-BCAD	BiLSTM-U-BC with untied directional self-attention mechanism
+clp	Gradient clipping

- Task 1: Part-of-Speech Tagging
- Task 2: content vs function word classification (Auxiliary task). Content words are words that contain semantic meaning, while function words are words that contain structural meaning (Fries, 1952). Annotations for these two classes can be easily obtained by converting Part-of-Speech labels to content word and function word labels.

It is important to note that the ORCHID project is built upon a multi-lingual machine translation project. Therefore, it has a different word segmentation standard, while BEST2010's word segmentation standard is based on a minimalist approach (Aroonmanakun etal., 2007).

For example, ชาติแห่งคอมพิวเตอร์และอิเล็กทรอนิกส์เทคโนโลยีศูนย์ (National Electronics and Computer Technology Center) is a single word on ORCHID corpus, while it is segmented into 7 words on BEST2010 corpus: ศูนย์ (center) เทคโนโลยี (technology) อิเล็กทรอนิกส์ (electronics) และ (and) คอมพิวเตอร์ (computer) แห่ง (of) ชาติ (nation).

Table4.4: Micro F-1 scores between BiLSTM-U and the weak baseline model BiLSTM reveal the effect of UNK masking technique. Value in bracket denotes standard deviation across three runs with different random seeds. $*$ - weak baseline

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 Task: POS						
BiLSTM$*$	94.83 (0.02)	85.39 (0.11)	75.19 (0.25)	63.64 (1.25)	50.71 (3.23)	36.25 (5.48)
+U	**95.01 (0.03)**	**87.55 (0.08)**	**79.01 (0.1)**	**68.97 (0.14)**	**57.39 (0.21)**	**43.08 (0.55)**
Corpus: BEST2010 Task: NER						
BiLSTM$*$	81.04 (0.3)	70.54 (0.52)	56.63 (1.98)	40.0 (4.46)	24.68 (5.35)	13.72 (4.07)
+U	**82.05 (0.19)**	**75.54 (0.31)**	**67.03 (0.44)**	**56.47 (0.3)**	**42.74 (0.55)**	**26.38 (1.28)**
Corpus: ORCHID Task: POS						
BiLSTM$*$	**94.01 (0.18)**	86.88 (0.19)	79.48 (0.07)	71.82 (0.22)	62.79 (0.77)	**52.4 (2.1)**
+U	93.68 (0.23)	**87.54 (0.2)**	**80.47 (0.39)**	**72.36 (0.42)**	**62.83 (0.64)**	52.12 (0.7)

4.4.3 Results

This section presents the experimental results of several techniques performed on multiple noisy test sets to measure the impact of adversarial noise on the sequential taggers. The experiments are conducted on 2 corpora mentioned in the previous section. In this chapter, we use BiLSTM architecture shown in figure 4.5 with normal word embeddings layer with no enhancement as a weak baseline. For a strong baseline, we enhance the BiLSTM architecture with ELMo representation. Table 4.10 summarizes the experimental results for this chapter. Micro F-1 scores are averaged across three runs with different random seed.

UNK Masking

To investigate the effects of UNK masking (BiLSTM-U), we compared it with the weak baseline (BiLSTM). We found that, on the BEST2010 corpus, UNK masking contributes to significant improvement in robustness against adversarial samples when compared to the weak baseline –BiLSTM. Not only that, UNK Masking can improve the performance of the model when there are adversarial samples; the results in Table 4.4 have shown that this technique can improve overall performance on BEST2010 corpus. On BEST2010 corpus, BiLSTM-U outperformed the weak baseline BiLSTM even when there were no adversarial samples as well. At the maximum stress level, BiLSTM-U improved the F1-score from 36.25 to 43.08 on the POS tagging task and from 13.72 to 26.38 on the NER task on BEST2010 corpus. This shows that, on the BEST2010 corpus, our UNK Masking technique can also improve robustness without relying on subword level information. Furthermore, BiLSTM-U yielded smaller standard deviations when there are adversarial examples.

On the ORCHID corpus, BiLSTM-U slightly improved the F1-scores of the POS tagging task when the stress levels were 20, 40, 60, and 80. When the stress level was 100, UNK masking did not improve the average F-1 score. However, its standard deviation was much lower.

One of the advantages of including more UNK tokens during the training phase is that the word vector of UNK token is trained to handle different types of words instead of just rare words that appear less than two times in the training set. Another advantage is that the C2W component is trained to handle more variety of “UNK" inputs.

In addition, UNK masking also improves the robustness of Thai word segmentation task. Table 4.5 shows that UNK masking can improve the robustness of AttaCut-SC (Chormai etal., 2019). AttaCut-SC is a CNN-based word segmentation model that takes the concatenation of character and syllable embeddings as input. We use UNK masking technique to mask out syllable inputs with UNK tokens.

Table4.5: F-1 scores between attacut-sc-u and the baseline word segmentation model attacut-sc reveal the effect of UNK masking on word-segmentation task.

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 (Free) Task: Word Segmentation						
AttaCut-SC	**93.26**	**85.19**	75.13	63.95	52.06	38.70
AttaCut-SC-U	91.80	84.87	**76.88**	**68.73**	**59.89**	**50.84**

Table4.6: Micro F-1 scores between BiLSTM-U and BiLSTM-U-B. Value in bracket denotes standard deviation across three runs with different random seeds.

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 Task: POS						
+U	95.01 (0.03)	**87.55 (0.08)**	**79.01 (0.1)**	68.97 (0.14)	57.39 (0.21)	43.08 (0.55)
+U-B	**95.19 (0.03)**	87.4 (0.02)	78.99 (0.17)	**69.72 (0.34)**	**59.67 (0.65)**	**48.0 (1.1)**
Corpus: BEST2010 Task: NER						
+U	82.05 (0.19)	75.54 (0.31)	67.03 (0.44)	56.47 (0.3)	42.74 (0.55)	26.38 (1.28)
+U-B	**82.29 (0.24)**	**75.86 (0.11)**	**67.87 (0.48)**	**56.96 (1.33)**	**44.25 (2.65)**	**29.91 (3.18)**
Corpus: ORCHID Task: POS						
+U	93.68 (0.23)	87.54 (0.2)	80.47 (0.39)	72.36 (0.42)	62.83 (0.64)	52.12 (0.7)
+U-B	**94.89 (0.1)**	**90.35 (0.14)**	**85.6 (0.15)**	**80.66 (0.18)**	**75.17 (0.3)**	**69.61 (0.3)**

Backoff Representation

To investigate the effects of backoff representation, we added the backoff component to the BiLSTM-U model. By comparing BiLSTM-U-B to BiLSTM-U, we found that backoff representation can give us better overall performance as well as better robustness against adversarial samples. As shown in Table 4.6, BiLSTM-U-B improved overall performance in most cases except the POS tagging task on BEST2010 corpus when the stress level was at 20 and 40, where BiLSTM-U-B yielded very close results to BiLSTM. We found that on both corpora, BiLSTM-U-B outperformed BiLSTM-U when there were no adversarial samples at all. When there were no adversarial examples, BiLSTM-U-B improved the F1-score from 95.01 to 95.19 on the POS tagging

Table4.7: Micro F-1 scores between BiLSTM-U-B and BiLSTM-U-BC. Value in bracket denotes standard deviation across three runs with different random seeds.

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 Task: POS						
+U-B	**95.19 (0.03)**	**87.4 (0.02)**	**78.99 (0.17)**	**69.72 (0.34)**	**59.67 (0.65)**	**48.0 (1.1)**
+U-BC	95.15 (0.09)	87.02 (0.04)	78.23 (0.1)	68.74 (0.27)	58.59 (0.39)	47.18 (0.5)
Corpus: BEST2010 Task: NER						
+U-B	82.29 (0.24)	**75.86 (0.11)**	**67.87 (0.48)**	56.96 (1.33)	44.25 (2.65)	29.91 (3.18)
+U-BC	**82.37 (0.17)**	75.72 (0.33)	67.56 (0.71)	**57.22 (0.95)**	**45.41 (0.74)**	**31.98 (0.81)**
Corpus: ORCHID Task: POS						
+U-B	94.89 (0.1)	**90.35 (0.14)**	**85.6 (0.15)**	**80.66 (0.18)**	**75.17 (0.3)**	**69.61 (0.3)**
+U-BC	**95.03 (0.12)**	89.94 (0.13)	84.58 (0.24)	79.0 (0.43)	72.34 (0.82)	65.78 (1.32)

task and from 82.05 to 82.29 on NER task on BEST2010 corpus. On the ORCHID corpus, BiLSTM-U-B improved the F1-score of the POS tagging task from 93.68 to 94.89. When the adversarial stress level was at the maximum level, BiLSTM-U-B outperformed BiLSTM-U on all corpora. On BEST2010 corpus, BiLSTM-U-B improved the F1-score from 45.13 to 47.09 on the POS tagging task and from 26.38 to 29.91 on the NER task. On the ORCHID corpus, BiLSTM-U-B improved the F1-score of the POS tagging task from 52.12 to 69.61. The results suggest that subword level information can make sequential taggers more robust.

Conditional Initialization with Affixation Embeddings

In this section, we investigate the effects of the condition initialization component on the backoff representation component (BiLSTM-U-B vs. BiLSTM-U-BC). The initial hidden state of LSTM can be a placeholder for a set of learnable parameters; we can also use it as a placeholder for affixation embeddings vector to include more linguistics knowledge into our model. Table 4.7 shows that hidden state condition initialization can only improve the performance of the NER task on BEST2010 corpus when the stress

Table4.8: Micro F-1 scores between BiLSTM-U-B, BiLSTM-U-BA, and BiLSTM-U-BA-clp. Value in bracket denotes standard deviation across three runs with different random seeds.

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 Task: POS						
+U-B	95.19 (0.03)	**87.4 (0.02)**	**78.99 (0.17)**	**69.72 (0.34)**	**59.67 (0.65)**	**48.0 (1.1)**
+U-BA	95.15 (0.04)	87.06 (0.06)	78.45 (0.04)	69.15 (0.11)	59.06 (0.26)	47.72 (0.79)
+U-BA-clp	**95.24 (0.03)**	87.06 (0.03)	78.18 (0.09)	68.42 (0.22)	57.64 (0.51)	45.17 (1.1)
Corpus: BEST2010 Task: NER						
+U-B	82.29 (0.24)	**75.86 (0.11)**	**67.87 (0.48)**	**56.96 (1.33)**	44.25 (2.65)	29.91 (3.18)
+U-BA	82.27 (0.16)	75.54 (0.19)	67.08 (0.36)	56.67 (1.04)	**44.39 (1.77)**	**30.27 (2.6)**
+U-BA-clp	**82.34 (0.22)**	75.31 (0.03)	66.4 (0.25)	54.89 (0.25)	41.28 (0.44)	26.19 (0.75)
Corpus: ORCHID Task: POS						
+U-B	**94.89 (0.1)**	**90.35 (0.14)**	**85.6 (0.15)**	**80.66 (0.18)**	**75.17 (0.3)**	**69.61 (0.3)**
+U-BA	94.69 (0.1)	88.9 (0.07)	82.91 (0.3)	76.7 (0.53)	69.77 (0.82)	62.89 (0.97)
+U-BA-clp	94.89 (0.12)	90.3 (0.17)	85.35 (0.33)	80.11 (0.21)	74.0 (0.53)	67.69 (0.87)

level is 60 or higher. When the stress level was 0, BiLSTM-U-BC slightly improved the F-1 scores of the NER task on BEST 2010 from 82.29 to 82.37 and on ORCHID corpus from 94.89 to 95.03. In other cases, BiLSTM-U-BC has shown no improvement.

Self-Attention Mechanisms

In this section, we observe the effects of various self-attention mechanisms on the backoff component. First, we compare BiLSTM-U-B with BiLSTM-U-BA and BiLSTM-U-BA-clp to see the effects of self-attention mechanism and gradient clipping on the backoff component. Then we compare all models with self-attention mechanisms against each other.

Table 4.8 compares BiLSTM-U-B with BiLSTM-U-BA and BiLSTM-U-BA-clp by adding the combination of attention mechanism along with gradient clipping on the BiLSTM-U-B model. BiLSTM-U-BA and BiLSTM-U-BA-clp did not yield significant improvement over Thai adversarial test sets. Across all adversarial test sets, BiLSTM-U-BA only yielded a small increase in performance on the NER task on BEST2010 corpus when the stress levels were 80 and 100. When stress level was 0, BiLSTM-U-BA-clp slightly improved the F-1 scores on BEST 2010 corpus from 95.19 to 95.24 on the POS tagging task, and 82.29 to 82.34 on the NER task.

Table 4.9 shows the experimental results of models with a self-attention mechanism. We compared BiLSTM-U-BA with other models with a self-attention mechanism to test whether their additional components contribute to improvement in robustness. The combination of self-attention mechanism and condition initialization (BiLSTM-U-BCA-clp) did not give any improvement on BEST2010 adversarial test sets when stress level was 40 or higher, and they did not yield any improvement at all on ORCHID adversarial test sets. On the ORCHID corpus, gradient clipping was the only technique that yielded an improvement of the performance on adversarial examples, while other self-attention based models with condition initialization component (BiLSTM-U-BCA-clp and BiLSTM-U-BCAD-clp) did not further improve the performance on adversarial examples. Additional components contributed to the small improvements on clean test sets on both corpora. On BEST2010, BiLSTM-U-BA-clp improved the F-1 score of the POS tagging task from 95.15 to 95.24, and BiLSTM-U-BCA-clp improved the F-1 score of NER task from 82.27 to 82.58. On ORCHID, BiLSTM-U-BCAD-clp improved the F-1 score from 94.69 to 94.98.

While various self-attention mechanisms presented in this chapter do not show a significant overall improvement on the performance of the model, attention mechanism can provide us an interface to probe into our model and hence increases the interpretability of our model. The plot on the top-left corner of Fig. 4.6 shows the attention score over the word เจ็บไข้ภัยโรค (sickness), the backward mechanism focuses ภัยโรค part of the word which means disease. While the forward part focuses on ย,

Table4.9: Micro F-1 scores between models with self-attention mechanism. Value in bracket denotes standard deviation across three runs with different random seeds.

	Stress Level					
Model	0	20	40	60	80	100
Corpus: BEST2010 Task: POS						
+U-BA	95.15 (0.04)	87.06 (0.06)	**78.45 (0.04)**	**69.15 (0.11)**	**59.06 (0.26)**	**47.72 (0.79)**
+U-BA-clp	**95.24 (0.03)**	87.06 (0.03)	78.18 (0.09)	68.42 (0.22)	57.64 (0.51)	45.17 (1.1)
+U-BCA-clp	95.2 (0.01)	**87.09 (0.1)**	78.28 (0.23)	68.8 (0.41)	58.67 (0.63)	47.24 (0.84)
+U-BCAD-clp	95.18 (0.07)	86.9 (0.04)	77.96 (0.01)	68.32 (0.02)	57.94 (0.08)	46.51 (0.62)
Corpus: BEST2010 Task: NER						
+U-BA	82.27 (0.16)	75.54 (0.19)	**67.08 (0.36)**	**56.67 (1.04)**	**44.39 (1.77)**	**30.27 (2.6)**
+U-BA-clp	82.34 (0.22)	75.31 (0.03)	66.4 (0.25)	54.89 (0.25)	41.28 (0.44)	26.19 (0.75)
+U-BCA-clp	**82.58 (0.22)**	**75.56 (0.1)**	67.0 (0.39)	56.44 (0.61)	43.66 (1.8)	28.65 (4.23)
+U-BCAD-clp	82.21 (0.08)	75.12 (0.46)	66.04 (0.5)	55.02 (0.55)	42.2 (0.45)	27.32 (0.68)
Corpus: ORCHID Task: POS						
+U-BA	94.69 (0.1)	88.9 (0.07)	82.91 (0.3)	76.7 (0.53)	69.77 (0.82)	62.89 (0.97)
+U-BA-clp	94.89 (0.12)	**90.3 (0.17)**	**85.35 (0.33)**	**80.11 (0.21)**	**74.0 (0.53)**	**67.69 (0.87)**
+U-BCA-clp	94.92 (0.03)	89.45 (0.18)	83.71 (0.61)	77.57 (0.76)	70.75 (1.16)	63.61 (1.68)
+U-BCAD-clp	**94.98 (0.06)**	89.76 (0.34)	84.52 (0.48)	78.85 (0.74)	72.42 (0.73)	65.99 (0.86)

which is the sixth character of the word, this shows that the forward part of the model builds a context vector using the output from the sixth LSTM cell. Both backward and forward components can focus on the informative part of the word. From the visualization, one might imply that the forward component only focuses on the sixth character of the word, but the forward component focuses on the output of the sixth LSTM cell, which contains information from previous timesteps in it and contains no information from the future timesteps. A context vector produced by a standard self-attention mechanism is a product of a concatenated vector of forward and backward LSTM outputs. Therefore, it contains information from both directions. When interpreting whether a self-attention mechanism is attending on the prefix or suffix of a word, it is possible for an untied directional self-attention mechanism to produce a context vector that only contains information from the previous timesteps.

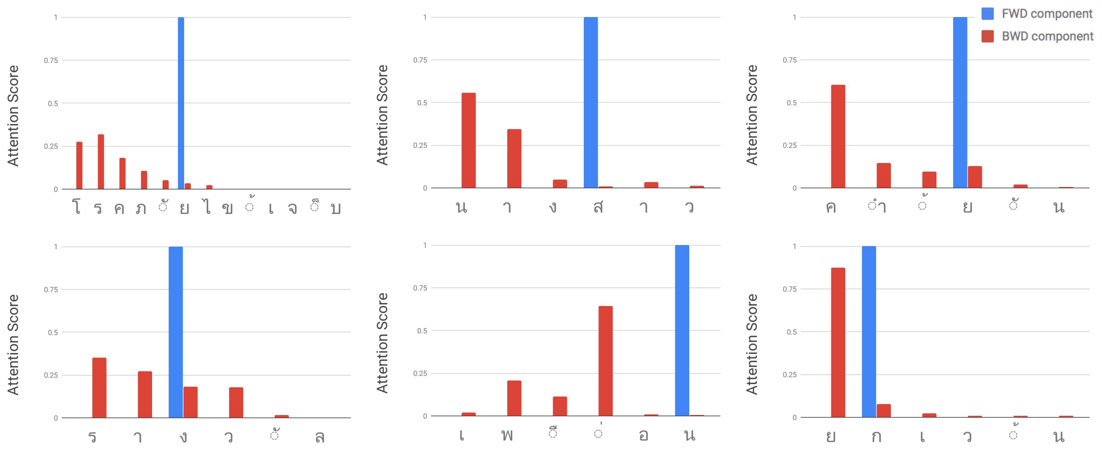


Figure4.6: The visualization of an untied-directional self-attention mechanism

Strong baseline: BiLSTM-ELMo

ELMo (Peters etal., 2018) is known to be robust against unknown words by leveraging contextual and morphological clues. To investigate the robustness of our proposed methods, we used ELMo to create a strong baseline (BiLSTM-ELMo) and compared it against our proposed methods. Table 4.10 shows the results of all experiments in this chapter, including BiLSTM-ELMo. For the POS tagging task on BEST2010 corpus, BiLSTM-ELMo showed to be the most robust against adversarial examples when the stress level was at 40 to 100, while maintaining competitive results when there were no adversarial examples and when the stress level was 20. For the NER task on BEST2010, the BiLSTM-ELMo model only obtained the best performance when the stress level was at 100. On ORCHID corpus, BiLSTM-ELMo showed to be the most robust against adversarial examples when the stress level was at 80 to 100. In this chapter, BiLSTM-ELMo showed to be the most robust on all corpora when the stress level was at 100. However, the speed of BiLSTM-ELMo was much slower than our proposed architectures, as shown in figure 4.7. Speed is a substantial issue for an NLP system that processes terabytes of text (e.g., online content analysis) (Al-Rfou and Skiena, 2012). We can conclude that our proposed models are faster than BiLSTM-ELMo and can produce a competitive result in cases where the stress level is less than or equal to 40.

Table4.10: Micro F-1 scores of all models on 2 Thai Corpora: BEST2010 and ORCHID. Value in bracket denotes standard deviation across three runs with different random seeds. ∗ - weak baseline, † - strong baseline

	Stress Level					
Model	0	20	40	60	80	100
			Corpus: BEST2010 Task: POS			
BiLSTM∗	94.83 (0.02)	85.39 (0.11)	75.19 (0.25)	63.64 (1.25)	50.71 (3.23)	36.25 (5.48)
+ELMo†	94.27 (0.04)	86.98 (0.1)	**79.42 (0.17)**	**71.78 (0.15)**	**64.21 (0.22)**	**56.27 (0.32)**
+U	95.01 (0.03)	**87.55 (0.08)**	79.01 (0.1)	68.97 (0.14)	57.39 (0.21)	43.08 (0.55)
+U-B	95.19 (0.03)	87.4 (0.02)	78.99 (0.17)	69.72 (0.34)	59.67 (0.65)	48.0 (1.1)
+U-BC	95.15 (0.09)	87.02 (0.04)	78.23 (0.1)	68.74 (0.27)	58.59 (0.39)	47.18 (0.5)
+U-BA	95.15 (0.04)	87.06 (0.06)	78.45 (0.04)	69.15 (0.11)	59.06 (0.26)	47.72 (0.79)
+U-BA-clp	**95.24 (0.03)**	87.06 (0.03)	78.18 (0.09)	68.42 (0.22)	57.64 (0.51)	45.17 (1.1)
+U-BCA-clp	95.2 (0.01)	87.09 (0.1)	78.28 (0.23)	68.8 (0.41)	58.67 (0.63)	47.24 (0.84)
+U-BCAD-clp	95.18 (0.07)	86.9 (0.04)	77.96 (0.01)	68.32 (0.02)	57.94 (0.08)	46.51 (0.62)
			Corpus: BEST2010 Task: NER			
BiLSTM∗	81.04 (0.3)	70.54 (0.52)	56.63 (1.98)	40.0 (4.46)	24.68 (5.35)	13.72 (4.07)
+ELMo †	79.92 (0.22)	72.12 (0.26)	63.14 (0.34)	53.79 (0.38)	44.04 (0.3)	**34.32 (0.27)**
+U	82.05 (0.19)	75.54 (0.31)	67.03 (0.44)	56.47 (0.3)	42.74 (0.55)	26.38 (1.28)
+U-B	82.29 (0.24)	**75.86 (0.11)**	**67.87 (0.48)**	56.96 (1.33)	44.25 (2.65)	29.91 (3.18)
+U-BC	82.37 (0.17)	75.72 (0.33)	67.56 (0.71)	**57.22 (0.95)**	**45.41 (0.74)**	31.98 (0.81)
+U-BA	82.27 (0.16)	75.54 (0.19)	67.08 (0.36)	56.67 (1.04)	44.39 (1.77)	30.27 (2.6)
+U-BA-clp	82.34 (0.22)	75.31 (0.03)	66.4 (0.25)	54.89 (0.25)	41.28 (0.44)	26.19 (0.75)
+U-BCA-clp	**82.58 (0.22)**	75.56 (0.1)	67.0 (0.39)	56.44 (0.61)	43.66 (1.8)	28.65 (4.23)
+U-BCAD-clp	82.21 (0.08)	75.12 (0.46)	66.04 (0.5)	55.02 (0.55)	42.2 (0.45)	27.32 (0.68)
			Corpus: ORCHID Task: POS			
BiLSTM∗	94.01 (0.18)	86.88 (0.19)	79.48 (0.07)	71.82 (0.22)	62.79 (0.77)	52.4 (2.1)
+ELMo †	93.37 (0.12)	89.02 (0.06)	84.82 (0.09)	80.39 (0.13)	**75.95 (0.15)**	**71.3 (0.27)**
+U	93.68 (0.23)	87.54 (0.2)	80.47 (0.39)	72.36 (0.42)	62.83 (0.64)	52.12 (0.7)
+U-B	94.89 (0.1)	**90.35 (0.14)**	85.6 (0.15)	**80.66 (0.18)**	75.17 (0.3)	69.61 (0.3)
+U-BC	**95.03 (0.12)**	89.94 (0.13)	84.58 (0.24)	79.0 (0.43)	72.34 (0.82)	65.78 (1.32)
+U-BA	94.69 (0.1)	88.9 (0.07)	82.91 (0.3)	76.7 (0.53)	69.77 (0.82)	62.89 (0.97)
+U-BA-clp	94.89 (0.12)	90.3 (0.17)	**85.35 (0.33)**	80.11 (0.21)	74.0 (0.53)	67.69 (0.87)
+U-BCA-clp	94.92 (0.03)	89.45 (0.18)	83.71 (0.61)	77.57 (0.76)	70.75 (1.16)	63.61 (1.68)
+U-BCAD-clp	94.98 (0.06)	89.76 (0.34)	84.52 (0.48)	78.85 (0.74)	72.42 (0.73)	65.99 (0.86)

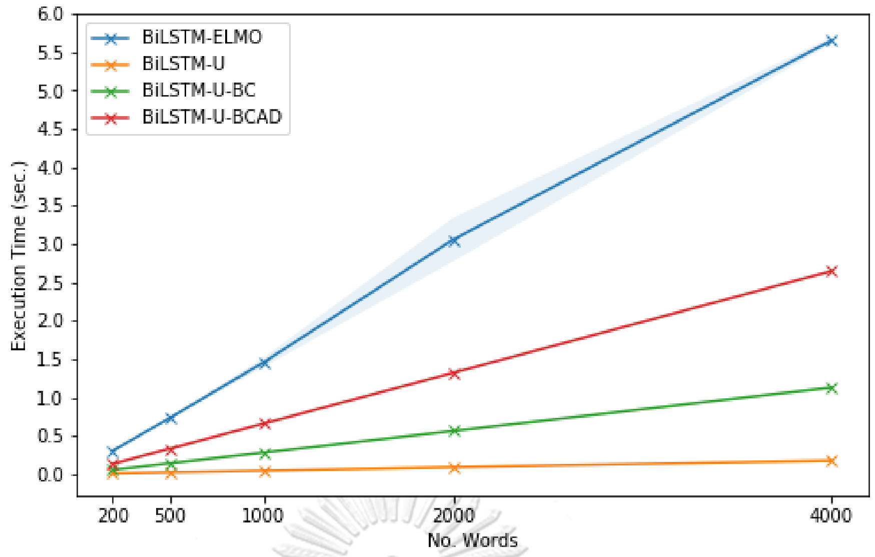


Figure4.7: Average inference speed of BiLSTM-ELMo and our proposed models measured as execution time (in seconds with an NVIDIA's GeForce GTX 1080, we repeated each execution for 5 times to get its average inference speed) using *timeit*, the standard python library for measuring execution time.

4.4.4 Error Analysis

In order to understand the problems caused by adversarial spelling perturbations, we analyze the error rate of each perturbation type. Table 4.11 shows the error rate of each model on each perturbation type across the two corpora. Error rates are averaged across three runs with a different random seed. We only focus on the POS tagging task in this section to make a comparison between the two corpora.

During an adversarial evaluation, there are four scenarios depending on which word we are observing and its context. Fig. 4.8 illustrates a possible example for each scenario.

The four scenarios are as follows:

1. **observed words without perturbation in a low-noise context**: at stress level 20, "no perturbation" column in Table 4.11 shows results of observed words with-

วัน(day)|นี้(this)|ฉัน(I)|รู้สึก(to feel)|เมื่อย(be exhausted)|มาก(very)

removeTone(รู้สึก)—>รูสึก

1) an observed word without perturbation in a low-noise context
วัน|นี้|ฉัน|รู้สึก|เมือย|มาก
2) an observed word with perturbation in a low-noise context
วัน|นี้|ฉัน|รูสึก|เมือย|มาก
3) an observed word without perturbation in a high-noise context
วั|นี|ฉั|รู้สึก|เมือย|มา
4) an observed word with perturbation in a high-noise context
วั|นี|ฉั|รูสึก|เมือย|มา

Figure4.8: Examples of the Thai sentence in various scenarios. The observed word is "to feel". The blue color denotes the observed word without perturbation. The pink color denotes the observed word with perturbation. The red color denotes perturbed contextual words.

out perturbation in a low-noise context. The differences in the error rates between the models are within 2 percentage points across the two corpora.

2. **observed words with perturbation in a low-noise context**: at stress level 20, "lastChar", "swapChar", "moveTone", and "removeTone" columns in Table 4.11 shows results of observed words with perturbation in a low-noise context. All models not only degrade substantially when evaluated on all 4 types of perturbations, but it also appears that the differences in error rates between the models are much greater than scenario 1. On BEST2010 corpus, BiLSTM-U performed surprisingly well for a model without character-level representation. It outperformed BiLSTM-ELMo, the strong baseline, on "swapChar" and "removeTone" perturbations. BiLSTM-U-B outperformed BiLSTM-ELMo on "moveTone". On ORCHID test set, BiLSTM-U-B outperformed BiLSTM-ELMo on on "swapChar", "moveTone", and "removeTone" perturbations. However, BiLSTM-ELMo obtained the lowest error rate on "lastChar".

3. **observed words without perturbation in a high-noise context:** at stress level 100, "no perturbation" column in Table 4.11 shows the results of observed words

without perturbation in a high-noise context. The differences in the error rates between the models are within four percentage points across the two corpora. It appears that error rates do not always increase with increases in stress level for words without perturbation.

4. **observed words with perturbation in a high-noise context**: at stress level 100, “lastChar", “swapChar", “moveTone", and “removeTone" columns in Table 4.11 shows results of observed words with perturbation in a high-noise context. On the BEST2010 test set, BiLSTM-ELMo has demonstrated its robustness across all perturbation types when the stress level is at 100. However, it only outperformed on one perturbation type (lastChar) on the ORCHID test set when the stress level is at 100. Note that the ORCHID corpus is much smaller than the BEST2010 corpus (2,814,970 tokens vs. 269,707 tokens). BiLSTM-ELMo is better when training data is large. When training data is small, our proposed methods are competitive to ELMo. On the ORCHID corpus, BiLSTM-U-B outperformed BiLSTM-ELMo on “swapChar” and “moveTone” perturbations. BiLSTM-ELMo obtained the lowest error rate on “lastChar” in this scenario.

By comparing scenario 1 to scenario 2, and comparing scenario 3 to scenario 4, we can observe that error rates of observed words with perturbation are much greater than error rates of observed words without perturbation in both low and high-noise scenarios.

By comparing scenario 1 to scenario 3, we can observe that a higher amount of perturbations in surrounding texts do not substantially increase the error rates of words without perturbation. In contrast, we can compare scenario 2 to scenario 4 and observe that the error rates increase substantially for perturbed words. We can conclude in this adversarial evaluation scheme, the performance of all models on perturbed words degraded when the surrounding words are also perturbed. Appendix A.2 shows examples of part-of-speech predictions of the perturbed inputs shown in Fig. 4.8.

4.5 Conclusions

In this chapter, multiple techniques are introduced to improve the overall performance and the robustness of the weak baseline BiLSTM. Each technique can also give different output for each corpus due to its differences and amount of training data. Our techniques do not perform as well as the strong baseline BiLSTM-ELMo when the stress level is at 100, but they give competitive results when the stress level is less or equal to 40 while having significantly less inference time.

Our adversarial evaluation scheme has shown that the baseline BiLSTM model's performance drops rapidly when it is tested against adversarial samples. Our adversarial evaluation scheme can give more insight into the robustness of the model against spelling mistakes. Looking at the evaluation result without any adversarial sample, one can assume that these techniques offer no significant improvement. E.g., the difference between the micro F1 scores of BiLSTM-U and the baseline BiLSTM is less than 1 in all experimental setups where there are no adversarial examples. However, this gap grows as there are more adversarial examples in the test set. Through our adversarial evaluation scheme, we can uncover the incompetency of the weak baseline BiLSTM model. The robustness of NLP systems can only be measured and improved with evaluation methods that challenge them with noisy and non-standard text examples.

It is important to note that this chapter only explores black-box adversarial examples. This chapter does not experiment with white-box adversarial examples. We can use the knowledge of the model's parameters to build adversarial examples designed to maximize a loss score. In the next chapter, I will introduce a method to create white-box adversarial examples for sequential tagging tasks. In addition, I will introduce an

Table4.11: Error rate % of each perturbation of each model tested on different noise levels. Value in bracket denotes standard deviation across three runs with different random seeds. Bolded text denotes the lowest error rate among each type of perturbation. Red text denotes the highest error rate among each type of perturbation.

model	no perturbation	lastChar	swapChar	moveTone	removeTone
	Corpus: BEST2010	Task: POS	Stress Level : 20		
BiLSTM∗	4.94 (0.05)	61.84 (0.41)	57.82 (0.52)	63.22 (1.6)	63.23 (2.04)
+ELMo†	5.22 (0.04)	**43.28 (0.46)**	53.84 (0.62)	46.71 (1.24)	54.10 (1.54)
+U	4.60 (0.02)	54.95 (0.85)	**47.07 (0.78)**	46.48 (2.54)	**49.10 (0.54)**
+U-B	4.42 (0.04)	53.09 (0.2)	48.41 (0.6)	**44.44 (2.78)**	49.41 (1.02)
+U-BC	4.47 (0.07)	55.37 (0.22)	50.27 (0.29)	46.64 (3.71)	49.88 (1.69)
+U-BA	4.47 (0.04)	55.10 (0.11)	49.75 (0.24)	47.97 (1.56)	52.22 (3.04)
+U-BA-clp	**4.40 (0.02)**	55.32 (0.22)	50.50 (0.25)	50.31 (2.6)	52.62 (0.82)
+U-BCA-clp	4.41 (0.04)	55.07 (0.85)	50.14 (0.41)	52.66 (0.36)	52.93 (0.47)
+U-BCAD-clp	4.46 (0.05)	56.27 (0.45)	50.74 (0.29)	51.72 (0.27)	53.47 (2.79)
	Corpus: BEST2010	Task: POS	Stress Level : 100		
BiLSTM∗	5.71 (0.5)	77.61 (5.58)	77.41 (7.89)	83.17 (7.16)	82.85 (6.59)
+ELMo†	4.22 (0.16)	**47.67 (0.79)**	**59.07 (0.04)**	**54.36 (0.46)**	**60.30 (1.0)**
+U	4.69 (0.12)	73.8 (0.95)	73.35 (1.14)	77.57 (0.99)	76.75 (1.22)
+U-B	**3.84 (0.07)**	64.50(1.21)	62.60 (1.7)	66.54 (2.93)	72.13 (2.6)
+U-BC	4.19 (0.34)	66.02 (0.95)	62.77 (0.34)	66.93 (4.1)	70.55 (4.53)
+U-BA	4.02 (0.43)	65.45 (0.75)	62.08 (0.89)	68.72 (1.1)	72.34 (0.84)
+U-BA-clp	4.25 (0.28)	67.83 (1.39)	65.98 (1.22)	72.05 (2.64)	75.57 (2.23)
+U-BCA-clp	4.05 (0.06)	65.73 (1.08)	62.96 (1.59)	71.08 (3.01)	72.26 (1.51)
+U-BCAD-clp	4.12 (0.14)	66.95 (0.65)	63.42 (0.93)	72.89 (1.1)	74.09 (2.15)
	Corpus: ORCHID	Task: POS	Stress Level : 20		
BiLSTM∗	6.34 (0.23)	57.98 (0.3)	59.96 (0.45)	48.18 (0.57)	47.25 (2.2)
+ELMo†	6.76 (0.11)	**30.28 (0.21)**	49.89 (0.62)	38.61 (2.62)	32.97 (2.2)
+U	6.51 (0.17)	52.07 (1.25)	52.79 (0.06)	55.12 (2.06)	45.79 (1.27)
+U-B	5.15 (0.11)	40.11 (0.69)	**39.97 (2.47)**	**29.70 (2.62)**	**30.40 (1.27)**
+U-BC	**5.13 (0.1)**	43.07 (0.96)	43.53 (1.98)	33.99 (6.29)	36.63 (4.58)
+U-BA	5.48 (0.15)	49.29 (1.85)	49.02 (1.08)	34.65 (5.24)	39.56 (2.91)
+U-BA-clp	5.16 (0.1)	39.53 (1.05)	41.39 (2.39)	30.03 (6.74)	35.16 (1.1)
+U-BCA-clp	5.19 (0.05)	46.38 (1.31)	47.1 (1.24)	36.30 (1.51)	39.93 (2.77)
+U-BCAD-clp	5.17 (0.07)	44.39 (2.65)	44.56 (2.7)	34.65 (3.96)	38.46 (2.91)
	Corpus: ORCHID	Task: POS	Stress Level : 100		
BiLSTM∗	6.49 (1.22)	71.3 (2.64)	69.32 (2.74)	62.89 (1.21)	62.6 (3.31)
+ELMo†	4.16 (0.02)	**34.67 (0.51)**	51.07 (0.6)	41.43 (2.06)	**35.67 (1.34)**
+U	4.78 (0.14)	72.23 (1.0)	70.96 (1.3)	65.71 (1.17)	69.42 (2.03)
+U-B	**3.05 (0.14)**	46.9 (0.8)	**44.14 (1.4)**	**35.32 (2.06)**	40.56 (2.52)
+U-BC	3.33 (0.28)	52.21 (1.8)	50.55 (2.33)	40.6 (6.51)	44.08 (2.42)
+U-BA	3.52 (0.11)	57.02 (1.68)	54.48 (1.28)	44.51 (4.24)	47.11 (0.36
+U-BA-clp	3.18 (0.2)	49.64 (0.87)	47.17 (1.67)	39.37 (9.98)	44.42 (4.47)
+U-BCA-clp	3.27 (0.26)	55.4 (2.77)	54.26 (2.2)	41.02 (2.34)	48.42 (0.63)
+U-BCAD-clp	3.2 (0.18)	52.16 (1.38)	49.97 (1.35)	39.99 (0.95)	47.59 (2.03)

adversarial training technique to improve the robustness of sequential tagging models.

Chapter V

ADVERSARIAL TRAINING AND WHITE-BOX ADVERSARIAL EVALUATION

5.1 Introduction

This chapter is a slightly modified version of "Towards Improving the Robustness of Sequential Labeling Models Against Typographical Adversarial Examples Using Triplet Loss" submitted to Natural Language Engineering. In contrast to the previous chapter, this chapter proposes white-box adversarial examples instead of black-box adversarial examples. White-box adversarial examples have an advantage over black-box adversarial examples because they can exploit the knowledge of the target model's parameters to approximate the worst-case perturbations.

This chapter focuses on improving the robustness of sequential labeling models using adversarial training and metric learning techniques. In NLP, there are many sequence labeling tasks. Many NLP tasks such as part-of-speech (PoS) tagging, named-entity recognition (NER), and chunking can be formulated as sequence labeling problems. The goal of any sequence labeling problem is to assign a categorical label to each word or token (or even character) in a natural language sequence. Recent advancements in NLP have shown that deep learning models can be very effective sequence taggers. BiLSTM-CRF is one of the well-known techniques which has become a standard method for building a sequential tagger (Huang etal., 2015; Lample etal., 2016b). Recent works improve the accuracy of BiLSTM-CRF models by using contextualized word embeddings (Peters etal., 2018; Akbik etal., 2018, 2019). These methods have high evaluation scores across many sequence labeling tasks. However, it is common to only evaluate sequential taggers on clean datasets with regular textual inputs. The models' performances drop substantially when evaluated against malformed texts (Udomcharoenchaikit etal., 2020). In the real-world, misspelling errors are common, and detrimentally affect the performance of sequential labeling models. It is important to note that most NLP models are not trained to be robust against spelling errors.

It is known that machine learning systems, state-of-the-art systems, are sensitive to very small input perturbations. These small perturbations can cause poorer performance. In the field of computer vision, it has been exhibited that machine learning models often misclassify perturbed examples with differences that are indistinguishable to humans.(Goodfellow etal., 2015). These perturbed examples are also called *adversarial examples*. Adversarial examples are more common in computer vision than in NLP. Due to textual inputs' discrete nature, it is almost impossible to create imperceptible perturbation on textual inputs. Adversarial examples in NLP are discrete perturbations, such as typographical errors or token substitutions. NLP models are also sensitive to these adversarial examples. Textual adversarial examples have shown to cause detrimental effects on NLP systems such as, text classification (Ebrahimi etal., 2018), machine translation (Belinkov and Bisk, 2018), and reading comprehension systems (Jia and Liang, 2017).

Previous works on adversarial examples for NLP mostly targets text classification or machine translation tasks. Even though sequential tagging tasks are common in NLP, research on adversarial examples for sequential tagging tasks is not common. Sequential tagging models are the building blocks for many NLP applications, especially applications without enough data to be trained end-to-end. Error analysis on sequential labeling tasks can provide an insight to robustness against spelling errors because every word in sequential labeling tasks has its annotation. Therefore we can point out where a sequential tagger has failed.

Previous literature improve the robustness of sequential taggers on out-of-distribution texts. They apply adversarial training methods to enhance the robustness of their models; they perturb the embeddings in continuous space and do not perturb the input texts. (Yasunaga etal., 2018; Zhou etal., 2019). They define the robustness of sequential taggers by testing them on twitter data (Zhou etal., 2019), on rare words (Yasunaga etal., 2018) or on input texts with capitalization errors (Bodapati etal., 2019). It is not common to include typographical errors as a benchmark for robustness. Since spelling errors are prevalent, we should consider it for evaluating robustness.

In this chapter, we introduce a training strategy for sequential labeling models designed to be robust when there are misspelled inputs. We apply triplet loss (Schroff etal., 2015) to enforce similarity between clean texts' encodings and their parallel perturbed texts. A novel training strategy that combines adversarial training with triplet loss (AT-T) is proposed. Instead of perturbing the embeddings, we perturb the discrete textual inputs for both adversarial training and evaluation. We benchmark the robustness of the models against typographical errors using black-box and white-box adversarial examples. We create these adversarial examples by generating spelling errors within an input sequence. We do not assume any access to the model parameters when generating black-box adversarial examples; while, white-box adversarial examples are generated with full-access to the model parameters to find the set of perturbations that maximize loss for each input sequence. Hence, we can gain an insight towards the worst-case perturbations to the textual inputs from the white-box adversarial examples. Here we show that our AT-T method is more robust than the baseline training method when tested on adversarial test sets.

In this chapter, we present two main contributions:

1. A novel training framework for sequential taggers that incorporates adversarial training with triplet loss (AT-T) which improves the robustness over spelling er-

Table5.1: The typographical errors for creating adversarial examples. * Note that capitalization error is not applicable to writing systems without capitalization.

Typographical error	Example	
Insertion	Thailand → Thailaand	arrow → arrrow
Transposition	Thailand → Thailadn	arrow → arorw
Deletion	Thailand → Thailnd	arrow → arow
Substitution	Thailand → Thailxnd	arrow → arroe
Capitalization*	Thailand → tHAILAND	arrow → ARROW

rors by constraining the similarity between unperturbed textual inputs and perturbed textual inputs (Section 5.2.3).

2. An adversarial evaluation scheme for sequential labeling tasks in NLP that benchmarks with both white-box and black-box adversarial examples (Section 5.2.4).

5.2 Methodology

This chapter's main idea is that we want to build a robust model against misspelled texts. Ideally, a robust model should perform well on both clean test sets and adversarial test sets. This section discusses our methodology for generating adversarial examples, our training strategies, and an adversarial evaluation scheme that evaluates our models with multiple parallel test sets.

5.2.1 Adversarial Examples

For discrete textual inputs, one of the perturbations that we can generate is perturbation at the character-level. This is more likely to maintain the meaning of the original input in contrast to word-level perturbation. Previous research found that NLP models are not robust to character changes, while humans are much more robust to these changes (Ebrahimi etal., 2018).

In this chapter, we generate adversarial examples to simulate a scenario where a model encounters misspelled textual inputs. Table 5.1 reveals five typographical errors used to create typographical adversarial examples. Previous research experimented with insertion, transposition, substitution, and deletion errors (Heigold etal., 2018; Belinkov and Bisk, 2018; Karpukhin etal., 2019). Since capitalization is an important feature to sequential tagging tasks such as NER (Nebhi etal., 2015; Bodapati etal., 2019), we also include capitalization error in our experiment. To generate a capitalization error, we swap uppercase characters with lowercase characters and vice-versa. By synthetically perturbing spellings of words within the existing datasets, we can generate new test sets without having to collect and annotate new test sets.

We use the following constraints to generate adversarial examples are more likely to be comprehensible to humans: we only perturb words with at least four characters. We do not perturb the first or last letters except for transposition and capitalization errors. For transposition error, we do not perturb the first letter. For capitalization error, almost all perturbations do not stop humans from inferring the original text.

Black-box adversarial examples

Black-box adversarial examples are created without any knowledge about the model's parameters. For each word with the length of at least four characters, we sample a perturbation type from a discrete uniform distribution. Then we generate all the possible variations within the perturbation type and constraints mentioned in the previous section, then we sample one of perturbations from a discrete uniform distribution.

White-box adversarial examples

White-box adversarial examples have a full-access to the model's parameters, it uses this knowledge to create worst-case adversarial examples. In this work, we extend previous literature (Ebrahimi etal., 2018; Michel etal., 2019; Wallace etal., 2019) to generate adversarial examples with typographical errors. Given a textual input sequence of n words $X = \{x_1, \ldots, x_n\}$, we represent each word with its vector representation x_i. Then We start by selecting the position i^* in the input sequence with the largest gradient magnitude. If the word x_i has at least four characters, then it satisfies our constraint then we find an adversarial example $\hat{x}$ for the word x_i based on the following optimization problem:

$$\underset{\hat{x} \in C}{\arg\max} \mathcal{L}_{\text{tar}} \left(x_0, \ldots, x_{i-1}, \hat{x}, x_{i+1}, \ldots, x_n\right) \tag{5.1}$$

Where C is all possible misspelled variations for the word x_i, and $\hat{x}$ is one of the possible perturbations. $\mathcal{L}_{\text{tar}}$ is the target loss function. For all white-box adversarial attacks in our experiments, we use a standard loss function $\mathcal{L}_{\text{tar}}$ without any auxiliary losses. The equation 5.1 can be approximated using the first-order approximation to save the computational cost (Ebrahimi etal., 2018; Michel etal., 2019; Wallace etal., 2019):

$$\underset{\hat{x} \in C}{\arg\max} \left[\hat{\mathrm{x}} - \mathrm{x}_i\right]^\top \nabla_{\mathrm{x}_i} \mathcal{L}_{\text{tar}} \tag{5.2}$$

Algorithm 2 shows that we do not apply any perturbation to words with the length of less than four characters. At each iteration, we select a position in the sequence with the largest gradient magnitude. Then we alter the word at the selected position with the perturbation that causes the highest loss. Then we replace the word at the selected position in the input sequence with its adversarial version. We iterate the same process without perturbing the already altered word until every word in the input sequence is perturbed. The final result is a perturbed text sequence (X_p).

Algorithm2: White-box adversarial attack

```
Input: Original text sequence ($X$)
Output: Perturbed text sequence ($X_p$)
(1. Initialize the list of perturbed words and fill it with indexes of words that
will not be perturbed e.g. words with less than 4 characters )
perturbed_list← $\varnothing$;
foreach word $x_i$ in the sequence $X$ do
    if $|x_i|$ < 4 characters then
        perturbed_list ← perturbed_list $\cup\{i\}$;
    end
end
(2. Each iteration, generate an adversarial example and replace the original
word with it )
while $\exists i \notin perturbed\_list$ do
    back-propagate $\mathcal{L}_{\mathrm{tar}}$ to get gradient $\nabla_{\mathrm{x}_i}\mathcal{L}_{\mathrm{tar}}$;
    $i^* \leftarrow \underset{i \notin perturbed\_list}{\arg\max} \|\nabla_{\mathrm{x}_i}\mathcal{L}_{\mathrm{tar}}\|_2$;
    perturbed_list ← perturbed_list $\cup\{i^*\}$;
    $\widehat{\mathrm{x}}^* \leftarrow \underset{\hat{x} \in C}{\arg\max} \, [\hat{\mathrm{x}} - \mathrm{x}_i]^\top \nabla_{\mathrm{x}_i}\mathcal{L}_{\mathrm{tar}}$;
    $X[i^*] \leftarrow \hat{x}^*$;
end
$X_p \leftarrow X$;
return $X_p$;
```

Figure5.1: Real samples collected from the CoNLL2003 test set with predictions from the baseline model (ELMo enhanced BiLSTM-CRF): (a) unperturbed original text, (b) black-box perturbation, and (c) gradient-based white-box perturbation. Blue texts and boxes refer to correct predictions. Red texts and boxes refer to incorrect predictions.

Fig. 5.1 illustrates a text sequence collected from the CoNLL2003 test set along with its black-box and white-box perturbed versions. Fig. 5.1 also includes predictions from the baseline model (ELMo enhanced BiLSTM-CRF (Peters etal., 2018)) for the NER task.

5.2.2 Adversarial Training

Adversarial Training (AT) is a common technique for enhancing the robustness of deep learning models against adversarial examples. It improves the robustness against adversarial attacks by including adversarial examples into the training set. We employ a black-box adversarial training procedure, where we augment input sequences with their parallel perturbed text sequences without using any access to the target neural networks' parameters. Adversarial examples are generated using the method discussed in Section 5.2.1.

At the beginning of every iteration, black-box adversarial examples are generated dynamically for all the training inputs in order to train the model using both clean examples and typographical adversarial examples. Therefore, data augmentation would vary every epoch. This enhances the robustness of the model against an array of perturbed input texts. At each training step, the loss function for adversarial training is defined as:

$$\mathcal{L}_{AT} = \mathcal{L}_{clean}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \gamma \mathcal{L}_{perturbed}\,(\boldsymbol{\theta}; \boldsymbol{X}_{\mathrm{p}}, \boldsymbol{y}) \tag{5.3}$$

where $\mathcal{L}_{clean}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y})$ and $\mathcal{L}_{perturbed}\,(\boldsymbol{\theta}; \boldsymbol{X}_{\mathrm{p}}, \boldsymbol{y})$ represent the loss for an unperturbed text sequence (clean) and the loss for the perturbed text sequence, respectively. y represents an output sequence of n predictions, where $\mathrm{y} = (y_1, y_2, \ldots, y_n)$. $\boldsymbol{\theta}$ represents the learnable parameters. $\gamma \in (0,1)$ is a constant weight (hyperparameter) for the loss score of the perturbed text input. In our experiments, we used $\gamma = 0.2$.

5.2.3 Auxiliary Losses

In our experiments, we used two auxiliary losses: pair similarity loss and triplet loss. The central idea of these losses is that we want to constraint the distance between an encoded representation of an original input sequence and an encoded representation of its perturbed pair.

A contextual encoder takes vectors of an input sequence $(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$ and returns vectors of an output sequence $(\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_n)$, which represents encoded information about the input sequence at each time-step.

We yield a fixed dimensional embedding representation of an entire text sequence by averaging the sequence of output vectors. Hence, we can calculate the distance between input sequences of different lengths.

$$\boldsymbol{s} = \frac{\sum_i \boldsymbol{h}_i}{|X|} \tag{5.4}$$

Where $\boldsymbol{s}$ is a fixed length encoded vector of a sequence, and $|X|$ is a number of words in the sequence.

$\boldsymbol{s}$ is normalized using L2 normalization. When two vectors are normalized to unit length, the squared Euclidean distance between two vectors is proportional to their cosine similarity score.

$$\boldsymbol{z} = \frac{\boldsymbol{s}}{\max\left(\|\boldsymbol{s}\|_2, \epsilon\right)} \tag{5.5}$$

Where ϵ is a small positive value that helps us avoid division by zero. We used $\epsilon = 10^{-12}$ in all of our experiments.

Pair Similarity Loss

Pair similarity loss constraints the distance between the encoded representation of a clean input text sequence and a perturbed input text sequence.

$$\mathcal{L}_{pair} = \|\boldsymbol{z}^c - \boldsymbol{z}^p\|_2^2 \tag{5.6}$$

Figure5.2: An example of a triplet from the CoNLL2003 corpus. With triplet loss as a part of training objective, we want the distance between a clean text sequence and its perturbed pair to be closer than the distance between a clean text sequence and a random text sequence as illustrated in this figure.

where z^c is an encoded representation of an original text sequence (clean) and z^p an encoded representation of a perturbed text sequence. We dynamically generate a perturbed text sequence from the clean text sequence as mentioned in section 5.2.1; therefore, perturbations for each text sequence are vary in each epoch.

Triplet Loss

The triplet loss from (Schroff etal., 2015) is applied as an auxiliary loss to constraint the distance between text sequences, so that an encoded representation of a clean text sequence is more similar to encoded representations of its perturbed versions than a random text sequence:

$$\mathcal{L}_{triplet} = \left[\|z^c - z^p\|_2^2 - \|z^c - z^r\|_2^2 + \alpha \right]_+ \tag{5.7}$$

where z^r represents an encoded representation of a random text sequence and α represents a margin between perturbed and random text sequences. We used $\alpha = 0.2$ in all of our experiments.

In order to mine each triplet, we sample a random sequence by randomly select a sequence from a training set, excluding the original text sequence (clean). As shown in Fig. 5.2, we provide an example of a triplet used for training one of our models; the distance between the clean text sequence and the perturbed text sequence are smaller than the distance between the clean text sequence and the random text sequence.

In summary, there are four training objectives that we experimented with:

1. **baseline:**
   $\tilde{L} = \mathcal{L}_{clean}$
2. **adversarial training (AT):**
   $\tilde{L} = \mathcal{L}_{AT}$
3. **adversarial training + pair similarity loss (AT-P):**
   $\tilde{L} = \mathcal{L}_{AT} + \mathcal{L}_{pair}$
4. **adversarial training + triplet loss (AT-T):**
   $\tilde{L} = \mathcal{L}_{AT} + \mathcal{L}_{triplet}$

The baseline method is a standard loss value of the original corpus (clean). We use it to train the baseline model to test whether we can improve the robustness with other training objectives.

AT, AT-P, and AT-T are training objectives for training models that can obtain high accuracy on clean sequential inputs and also robust against malformed texts at the same time. The AT method includes adversarial text sequences to the model at training time, making it more robust to adversarial examples. As shown in Fig. 5.3, the AT-P method constraints similarity between original and adversarial inputs at the encoder layer instead of the embedding layer. This allows us to freeze pre-trained embeddings and save on computation time since the expensive forward pass on the frozen parts will only be computed once for the clean inputs. Fig. 5.4 shows our proposed AT-T framework, which combines the adversarial training technique with triplet loss.

Figure5.3: The AT-P framework for training robust neural networks against typographical adversarial examples. It combines adversarial training with the pair similarity loss.

Figure5.4: Our AT-T framework for training robust neural networks against typographical adversarial examples. It incorporates adversarial training with the triplet loss.

5.2.4 Adversarial Evaluation Scheme

Despite strong performances on standard evaluation schemes, current sequential tagging systems perform poorly under adversarial evaluation. To determine whether sequential tagging systems can handle inputs with spelling errors, we use an adversarial evaluation scheme that contains parallel test sets with altered input texts.

Since we cannot directly evaluate the robustness of the models from the standard test sets. Therefore, it is necessary to simulate an environment where there are typographical errors. We do this by injecting typographical errors into our test sets. We generate typographical errors by using black-box and white-box methods discussed in section 5.2.1 and section 5.2.1. Our adversarial evaluation scheme tests whether NLP systems can tag text sequences that contain perturbed parallel text sequences without changing the original labels, while it also evaluates whether our models can maintain their performance on unperturbed texts. We propose that an evaluation scheme should evaluate models on three parallel test sets:

1. **original test set:** We evaluate the models on an original test set without any perturbation to ensure that our models can also perform well on unperturbed texts.
2. **black-box adversarial test set:** We alter and replace the original test set with

black-box adversarial examples. A test set with black-box adversarial examples can show us the robustness of the models against typographical errors in general. A perturbation for each word is selected without access to the model's parameters.

3. **white-box adversarial test set:** We alter and replace the original test set with gradient-based white-box adversarial examples. Instead of sampling a perturbation from a discrete uniform distribution, the white-box adversarial attack method selects an adversarial example using knowledge of the model's parameters to maximize the loss. Since there are multiple ways to generate an adversarial example for each word, the white-box adversarial attack method allows us to approximate the worst perturbation.

5.3 Experiments

In this section, we discuss the experimental setup, evaluation metric, and corpora used for training our sequence labeling models.

5.3.1 Experimental Setup

Our experiments rely on ELMo enhanced BiLSTM-CRF sequential tagger (Piktus etal., 2019) as a baseline for comparison since it can exploit character-level information as well as contextual information, making it suitable for dealing with malformed texts.

We follow the AllenNLP (Gardner etal., 2017) implementation of this model. The ELMo enhanced BiLSTM-CRF sequential tagger concatenates 512-dimensional pre-trained ELMo embeddings with 50-dimensional pre-trained GloVe embeddings and 16 dimensional CNN character embeddings to create a token representation. The CNN character representation uses 128 convolutional filters of size three with ReLU activation and max-pooling. Since there is no pre-trained ELMo or GloVe for Thai language, we trained ELMo from scratch on the ORCHID corpus. We also replaced GloVe with randomly initialized Thai word embeddings of the same dimensional size. The token representation is forwarded to two stacked BiLSTM layers with 200 hidden units in each layer. The second BiLSTM layer's output is passed to a CRF model to predict a label for each token in the input sequence. The dropout rate between each layer is 0.5 (including dropout between the first and the second BiLSTM layer). During training, the parameters are optimized using Adam optimizer with a constant learning rate of 0.001. The gradient norms are rescaled to have a maximum value of 5.0. The pre-trained ELMo embeddings are not fine-tuned. We train all the models in this research for 40 epochs and use early stopping to stop training after 10 epochs with no improvement on the clean validation set. We report the averaged score across three runs with different random seeds.

5.3.2 Evaluation Metric

For each corpus, we evaluate our models in three different test sets: clean, black-box, white-box as discussed in section 5.2.4. We benchmark our models using F_1 metric:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision } + \text{recall})} \tag{5.8}$$

For span labeling tasks, such as chunking and NER, a prediction is considered correct only if it is an exact match of a corresponding span in the corpus. A span is a unit that consists of one or more words. Part-of-Speech tagging task is not a span-based task therefore we can calculate F_1 score at word-level.

5.3.3 Corpora

In this chapter, we present the results obtained from four benchmarks across two corpora: CoNLL2003 (NER, English), CoNLL2003 (chunking, English), CoNLL2003 (PoS, English), and ORCHID (PoS, Thai).

CoNLL2003 is a language-independent named entity recognition corpus (Tjong KimSang and DeMeulder, 2003). It contains data from two European languages: English and German. Only English data is freely available for research purposes. Therefore, we only use the English version in this study. Apart from NER annotations, CoNLL2003 also contains part-of-speech and syntactic chunk annotations.

In addition, we also use a Thai ORCHID corpus as mentioned in section 4.4.2 because it has a similar size to CoNLL2003, and it also has part-of-speech annotation. Since Thai has no explicit word boundary, we use gold segmentation provided by the Thai ORCHID corpus to define our word inputs. We split 10 % of the ORCHID corpus for testing. Then we split the remaining corpus into a training set (90 %) and a validation set (10 %).

Table 5.2 shows the detailed statistics of the two corpora. For the CoNLL2003 corpus, the number of classes does not include the class ‘O’ for non-named entities or non-syntactic chunks.

For each corpus, we use a separated set of characters for creating insertion and substitution errors. For the English CoNLL2003 corpus, we use all English alphabets, both lowercase, and uppercase. For the Thai ORCHID, we use all the Thai consonants, the vowels, and the tonal marks.

Table5.2: Data statistics of the English CoNLL2003 corpus and the Thai ORCHID corpus. (training set/validation set/test set)

	CoNLL2003	**ORCHID**
Language	English	Thai
Task	PoS/NER/Chunking	PoS
# of classes	44/4/9	47
# of words	203,621/51,263/46,425	274,006/25,401/43,226
# of characters	890,042/224,197/200,112	1,187,721/107,512/190,394
avg. char / word	4.4/4.4/4.3	4.3/4.2/4.4

5.4 Experimental Results

We performed experiments over three sequence labeling tasks across two languages to better understand the relative importance of our training objectives. We investigated the robustness of sequential taggers through a series of black-box and white-box attacks. As shown in Table 5.3, the baseline method performs much worse on white-box adversarial examples than black-box adversarial examples, indicating that white-box attacks can generate stronger adversarial examples. This suggests that we can select a misspelling variant that is more damaging than the others. It is important to point out that the NER task is more sensitive to typographical errors than the chunking and PoS tasks. For the CoNLL2003 corpus, the differences between F_1 scores on black-box and white-box test sets are much higher for the NER task. The degradation of the F_1 scores–when evaluated on noisy test sets–seems to be less severe for the chunking and PoS tasks. For the Thai ORCHID corpus, the gap between F_1 scores on the black-box and the white-box test sets are much closer than the CoNLL2003 corpus.

Comparing to the baseline, AT, AT-P, and AT-T can improve the robustness against malformed text without degrading its performance on the clean texts. AT, AT-P, and AT-T can all maintain competitive results on the original test sets compared to the baseline method. The AT-T method outperforms all other methods on both black-box and white-box test sets for span-based tasks (NER and chunking).

Table5.3: F_1 scores and theirs standard deviations on the English CONLL2003 and the Thai ORCHID corpora. We averaged the scores across three runs with different random seeds, and calculated their standard deviations. A bolded number refers to the best F-1 score for each task in each test set.

	original test set	black-box test set	white-box test set
	CoNLL2003 (English,NER)		
baseline	91.72 ± 0.24	55.87 ± 1.61	39.53 ± 1.69
AT	91.85 ± 0.00	83.42 ± 0.28	69.06 ± 0.51
AT-P	91.78 ± 0.07	83.74 ± 0.20	69.17 ± 3.07
AT-T	**92.04** ± 0.00	**83.87** ± 1.30	**71.55** ± 2.42
	CoNLL2003 (English,Chunking)		
baseline	**91.92** ± 0.11	76.48 ± 0.90	66.03 ± 0.60
AT	91.87 ± 0.07	81.62 ± 0.41	77.90 ± 0.99
AT-P	**91.92** ± 0.04	86.97 ± 0.13	79.65 ± 0.57
AT-T	91.68 ± 0.08	**87.37** ± 0.27	**80.79** ± 1.16
	CoNLL2003 (English, PoS)		
baseline	**95.74** ± 0.12	84.01 ± 0.42	77.31 ± 0.29
AT	95.73 ± 0.06	**93.03** ± 0.04	88.79 ± 0.14
AT-P	95.65 ± 0.03	92.93 ± 0.05	**88.86** ± 0.05
AT-T	95.56 ± 0.00	92.83 ± 0.09	88.64 ± 0.33
	ORCHID (Thai, PoS)		
baseline	93.89 ± 0.08	77.36 ± 0.31	75.55 ± 1.13
AT	**94.13** ± 0.02	**90.37** ± 0.26	**85.44** ± 0.52
AT-P	93.89 ± 0.03	89.65 ± 0.44	83.21 ± 1.10
AT-T	93.91 ± 0.07	88.76 ± 0.65	83.19 ± 1.11

Furthermore, we also provide average cosine similarity scores between clean-perturbed pairs (z^c, z^p) and clean-random pairs (z^c, z^p) as shown in Table 5.4. This reveals the effectiveness of our similarity constraint techniques: pair similarity loss and triplet loss. Ideally, we want the cosine similarity score of a clean-perturbed pair to be high, and the cosine similarity score of a clean-random pair to be low.

5.4.1 Effects of Adversarial Training

We examined the performance of the models trained by the adversarial training (AT) method compared to the baseline method. Table 5.3 indicates that adversarial training is a very effective method for improving the robustness of sequential labeling models against typographical adversarial examples across all tasks on both black-box and white-box test sets. The adversarial training method also maintains competitive results on the original test sets.

For the Thai ORCHID corpus, the AT method obtained the best scores on all test sets. The differences between the F_1 scores of black-box and white-box test sets are lower for the PoS tagging task on the ORCHID test set than the CoNLL2003 corpus. For the English CoNLL2003 corpus, adversarial training obtains the best result on the black-box test set for the PoS tagging task.

Table 5.4 shows that adversarial training slightly improves the similarity between the encoded sequence representations of clean-perturbed pairs.

5.4.2 Effects of Pair Similarity Loss

We investigated the performance of the models trained by the AT-P method compared to the AT method. The AT-P method improves the performance by constraining the similarity between original texts and perturbed texts. Table 5.3 shows a marginal improvement on black-box and white-box test sets for the NER task; the improvement is more noticeable on the chunking task. The AT-P method can also maintain competitive results on the original test sets. The results suggest that, for span-based tasks (NER and chunking), we can improve the robustness against typographical errors by constraining the similarity between original and malformed text sequences. For the PoS tagging task, the AT-P method does not improve the F_1 score on the ORCHID adversarial test sets, and it marginally improves the F-1 score by only 0.07 on CoNLL2003 white-box test set from 88.79 (AT) to 88.86 (AT-P).

Table 5.4 shows that the pair similarity loss improves the similarity between the encoded sequence representations of clean-perturbed pairs substantially. However, it also increases the similarity between clean-random pairs for which we want to keep low.

5.4.3 Effects of Triplet Loss

We observed the performance of the models trained by the AT-T method compared to the AT and AT-P methods. Triplet loss minimizes the distance between clean text sequence and its parallel perturbed text sequence and maximizes the distance between clean text sequence and a random perturbed text sequence. Table 5.3 indicates that the AT-T method outperforms the AT-P method on both black-box and white-box test sets for span-based tasks (NER and chunking) on CoNLL2003. The AT-T method maintains competitive results on the original test sets and achieves the highest score on the original NER dataset. However, the AT-T method does not improve the PoS task performances on the CoNLL2003 and the ORCHID corpora on both black-box and white-box test sets.

Regarding to the fact that both AT-P and AT-T methods did not provide a clear improvement over the AT method, we can conclude that our sequence-level similarity constraints can improve the performances over adversarial test sets only for the span-based tasks. It does not improve the robustness over adversarial examples for the PoS tagging task. Note that a span can be a large portion of the sequence; therefore, it can benefit more from sequence-level similarity constraints.

Table 5.4 shows that triplet loss lowers the similarity between clean-random pairs. However, it does not improve the similarity between the encoded sequence representations of clean-perturbed pairs. Nevertheless, the gaps between the similarity scores between clean-perturbed pairs and clean-random pairs are larger with triplet loss.

Table5.4: Average cosine similarity scores between clean-perturbed pairs and clean-random pairs. We averaged the cosine similarity scores and their standard deviations across three runs with different random seeds. We want the similarity between each clean-random pair to be high, and the similarity between each clean-random pair to be low. † - This shows a difference between the average cosine similarity scores of clean-perturbed pairs and clean-random pairs for each model.

	clean-perturbed pair	clean-random pair	Δ avg sim. †
	CoNLL2003 (English,NER)		
baseline	0.72	0.57	0.15
AT	0.74	0.57	0.17
AT-P	**0.85**	0.81	0.04
AT-T	0.70	**0.29**	**0.41**
	CoNLL2003 (English, Chunking)		
baseline	0.76	0.66.	0.10
AT	0.77	0.66	0.11
AT-P	**0.85**	0.81	0.04
AT-T	0.70	**0.38**	**0.32**
	CoNLL2003 (English, PoS)		
baseline	0.64	0.36	0.28
AT	0.65	0.36	0.29
AT-P	**0.82**	0.69	0.13
AT-T	0.64	**0.25**	**0.39**
	ORCHID (Thai, PoS)		
baseline	0.65	0.36	0.29
AT	0.69	0.34	0.35
AT-P	**0.79**	0.53	0.26
AT-T	0.72	**0.32**	**0.40**

5.5 Error Analysis

In this section, we systematically examine what types of typographical errors are harder to predict. We analyzed the error rate of each typographical error type. Studying typographical errors on sequential labeling task has an advantage because we can analyze word-level errors by comparing them to their ground truths and their typographical error types. Therefore, we can study the impact of each error type in a test set with multiple error types. We evaluate errors with the black-box test sets since we can use the same black-box test set across multiple runs and multiple tasks within the same corpus. In addition, the error types are sampled from a discrete uniform distribution. For the CoNLL2003 black-box test set, the proportion of each error type is as follows: 8.9% insertion, 9.2% deletion, 9.2% transposition, 9.1% substitution, 9.1% capitalization, and 54.5% clean. For the ORCHID black-box test set, the proportion of each error type is as follows: 11.9% insertion, 11.7% deletion, 11.7% transposition, 11.4% substitution, and 53.3% clean. All the clean samples are words with less than four characters which we do not consider for perturbation.

Table 5.5 shows the error rates of the models on five typographical error type. The baseline model is trained on vanilla texts only. As shown in Table 5.5, the baseline model is the most sensitive to all types of typographical errors on all tasks and corpora. The baseline model is also sensitive to words without perturbation (Clean) for span-based tasks (NER and chunking), this suggests that they were also affected by their surrounding perturbed words. The error rate of the baseline model on words without perturbation is 25.04 percent for the NER task and 6.31 percent on the chunking task, while the error rate remains low at 2.53 and 3.30 for the part-of-speech tagging task on the ORCHID corpus and the CoNLL2003 corpus, respectively. For the English CoNLL2003 corpus, insertion error has the highest error rates for all tasks for the baseline model. Interestingly, substitution error has lower error rates than transposition error, which contains all the original letters for NER and chunking tasks. For the chunking task, transposition error rates are higher than deletion error, which also alters the length of each word. Surprisingly, capitalization error also has a high error rate even we normalized it for word embeddings. The only difference is that the character-level part of ELMo and the CNN character-to-word embeddings are sensitive to capital letters. Therefore capitalization is still an important feature for these sequential tagging tasks. For the English CoNLL2003 corpus, models trained with triplet loss are the most robust for span-based tasks (NER and chunking).

The AT method is the most effective on the black-box test set for the Thai ORCHID corpus. Substitution error causes the highest error rates for all methods. This error analysis reveals that the error rates on all typographical errors on the ORCHID corpus are high, while our models obtained high F_1 scores on the ORCHID test sets. This is because we did not use a span-based F_1 score to evaluate the ORCHID corpus since part-of-speech tagging is not a span-based problem—a span-based evaluation is more strict. The span-based evaluation is per span, not per token. Therefore, if there is an incorrect prediction within a span, then that span is considered incorrect. For the PoS tagging task, methods with additional similarity constraints (AT-P and AT-T) do not improve or only slightly improve the robustness against typographical errors. For span-based tasks, methods with similarity constraints (AT-P and AT-T) can enhance the performance over typographical adversarial examples.

5.6 Real-World Data

In this section, we set up an extra experiment using a real-world dataset to show the models' performances on real out-of-distribution data. We collected 41 Thai-language samples with spelling errors from Twitter (Twitter-41). The Twitter-41 dataset[1] contains 965 words. 291 out of these 965 words are out-of-vocabulary words. The number of unique words is 438. We annotated this dataset with the Universal Dependency (UD) POS tags because they can be mapped directly from ORCHID POS tags[2] Hence, we can test our models trained on the ORCHID corpus on this new dataset. Although the dataset is very small, it can reveal the NLP systems' weaknesses and illustrate the importance of building robust models. Also, it shows what the lack of diversity in data collection can lead to.

[1]The dataset can be downloaded from this link: `https://github.com/c4n/thai-political-tweets`

[2]PyThaiNLP documentation contains the mapping between UD tags and ORCHID tags: `https://www.thainlp.org/pythainlp/docs/2.0/api/tag.html`

Table5.5: Error rates and their standard deviations of models on different typographical errors. We calculate the error rates at word-level only. We excluded words with “O” (non-named entities, or non-syntactic chunks) as their ground truth from the calculation. ‡ - “Clean” refers to unperturbed words, these are words that are shorter than 4 characters. We do not alter words that are shorter than 4 characters to ensure that the perturbed words are intelligible to humans. † - Thai has no capitalization.

	Insertion	Deletion	Transposition	Substitution	Capitalization	Clean‡
	CoNLL2003 (English,NER)					
baseline	42.56 ± 1.13	41.61 ± 0.63	40.68 ± 0.38	37.98 ± 0.94	38.96 ± 3.74	25.04 ± 0.07
AT	18.00 ± 0.99	18.53 ± 0.54	18.21 ± 0.54	15.81 ± 0.15	15.00 ± 1.57	**11.38** ± 0.97
AT-P	**17.69** ± 1.80	17.71 ± 3.53	17.82 ± 2.26	15.98 ± 1.76	13.81 ± 5.76	12.02 ± 2.59
AT-T	18.21 ± 0.57	**17.56** ± 1.70	**16.88** ± 0.97	**15.09** ± 1.14	**12.93** ± 0.83	11.68 ± 1.00
	CoNLL2003 (English, Chunking)					
baseline	17.21 ± 0.94	14.39 ± 0.69	16.13 ± 1.09	15.32 ± 1.32	11.62 ± 0.62	6.31 ± 0.64
AT	7.64 ± 0.34	7.29 ± 0.16	7.64 ± 0.21	6.95 ± 0.14	5.52 ± 0.08	4.21 ± 0.08
AT-P	7.39 ± 0.29	6.57 ± 0.07	7.12 ± 0.19	6.59 ± 0.07	**5.08** ± 0.07	4.16 ± 0.15
AT-T	**7.18** ± 0.58	**6.44** ± 0.40	**7.07** ± 0.61	**6.54** ± 0.17	5.18 ± 0.37	**4.13** ± 0.05
	CoNLL2003 (English, PoS)					
baseline	34.63 ± 0.23	25.64 ± 2.06	30.76 ± 1.41	30.79 ± 1.11	39.18 ± 1.20	2.53 ± 0.08
AT	**12.92** ± 0.20	12.13 ± 0.10	14.62 ± 0.28	13.25 ± 0.08	10.62 ± 0.37	**2.48** ± 0.08
AT-P	12.97 ± 0.37	**12.04** ± 0.19	**14.10** ± 0.16	**13.15** ± 0.18	**10.53** ± 0.13	2.53 ± 0.00
AT-T	13.27 ± 0.53	12.17 ± 0.25	14.77 ± 0.53	13.29 ± 0.18	10.62 ± 0.59	2.61 ± 0.03
	ORCHID (Thai†, PoS)					
baseline	44.31 ± 0.80	42.72 ± 0.56	44.73 ± 0.84	46.91 ± 0.68	N/A	3.30 ± 0.16
AT	**16.61** ± 0.81	**16.89** ± 0.47	**16.90** ± 0.87	**20.28** ± 0.77	N/A	**2.57** ± 0.03
AT-P	19.35 ± 1.80	19.75 ± 1.82	19.92 ± 1.58	23.46 ± 2.12	N/A	2.70 ± 0.02
AT-T	20.20 ± 1.81	20.47 ± 1.59	20.11 ± 1.14	24.05 ± 1.72	N/A	2.79 ± 0.06

Although it is easier to annotate with UD tags because the number of unique tags is much lower than the ORCHID tags, the Twitter-41 dataset is very different from the ORCHID dataset used to train the model. The ORCHID dataset contains texts from academic conference proceedings, while the Twitter-41 dataset contains mostly informal political discussions in Thai. Therefore, the performances of the models on the Twitter dataset are worse than the ORCHID dataset. Table 5.6 shows that our proposed methods can improve the performance on the Twitter-41 dataset over the baseline model. Even though the performances on the Twitter-41 dataset are much lower, our results highlight the need to address out-of-distribution data.

Table5.6: The F1-scores of the models trained on the ORCHID dataset and evaluated on the Twitter-41 dataset

	Baseline	AT	AT-P	AT-T
Twitter-41 (PoS)	67.56	68.08	68.39	69.12

5.7 Conclusions

In this chapter, we propose a novel training method (AT-T) for a sequential labeling model comprising adversarial training and triplet loss. We also propose an adversarial evaluation scheme to conduct experiments on both original and adversarial test sets.

We have shown that our proposed AT-T training objective, incorporating adversarial training method and triplet loss, can withstand typographical adversarial examples while maintaining high F_1 scores on the original test sets. For span-based tasks (NER and chunking), the AT-T method yields higher F_1 scores than all the other methods on both black-box and white-box adversarial test sets.

Adversarial training can strengthen the robustness of all sequential taggers against spelling errors. Moreover, it slightly increases the similarity between the encoded sequence representations of clean-perturbed pairs.

Using similarity constraints, such as pair loss and triplet loss, at the sentence level improves the performance for span-based tasks (NER and Chunking). For PoS tagging, no substantial improvement is observed. Pair loss also increases the similarity between clean-perturb sequence pairs substantially. However, it also increases the similarity between clean-random pairs; and the differences between the average cosine similarity scores of clean-perturbed pairs and clean-random pairs become lower. Triplet loss widens these differences.

Our adversarial evaluation scheme shows the weaknesses of deep learning models on typographical errors. It approximates the worst-case scenario with white-box attacks and approximates a general performance on malformed texts with black-box attacks. The adversarial evaluation scheme also includes the performance scores on the unperturbed test sets to ensure that we can enhance adversarial examples without degrading our performance in a standard-setting. The experimental results have shown that typographical adversarial attacks can cause deterioration in the performance of sequential taggers. White-box adversarial examples can deteriorate the performance of the taggers much stronger than the black-box adversarial examples. This suggests for each sentence there is a perturbation variant that is much more effective than the others.

This work only addresses typographical adversarial examples. However, there are many linguistics capabilities that we should consider for improving the robustness of the model. The results on the Twitter-41 dataset have shown that there is a big gap in performance between the artificially perturbed dataset and the real-world dataset. Nevertheless, adversarial training and similarity constraint through metric learning are promising research directions for improving current NLP systems' overall robustness.

Chapter VI

CONCLUSIONS

In this dissertation, I introduced new adversarial evaluation schemes for benchmarking the robustness of NLP systems. I showed that adversarial examples could reveal weaknesses, which are not shown in a standard evaluation scheme (train-validation split) for both Thai and English. In addition, I also introduced new training methods to improve robustness over typographical errors. I have shown improvement over three core NLP tasks (word segmentation, part-of-speech tagging, and named-entity recognition), which are sequential tagging tasks. The benefit of experimenting on these tasks is that we can easily analyze errors at word-level since each word is annotated (or even character-level in a word segmentation task). Table 6.1 summarizes the adversarial attacks used in this dissertation. Table 6.2 compares multiple proposed strategies for enhancing the robustness against malformed texts.

Chapter 4 shows an adversarial evaluation scheme explicitly designed for Thai based on known errors; furthermore, it shows various strategies to enhance the robustness without augmenting adversarial examples to the training data. UNK masking has shown to be very effective in improving the performance of sequential taggers on adversarial examples. Backoff representation can further improve the robustness using character-level information. In some cases, conditional initialization with affixation embeddings can also improve the performance. This shows that a linguistically motivated design can help enhance the performance of the model. Furthermore, the untied bidirectional self-attention mechanism can help improve the interpretability of the model.

Chapter 5 shows an adversarial evaluation scheme with black-box and white-box adversarial examples; and, it also shows an adversarial training strategy along with similarity constraint objective to improve the performance over the adversarial test sets. In contrast to techniques used in chapter 4, adversarial training includes adversarial examples during the training time; it provides a large improvement for robustness against typographical adversarial examples. Furthermore, metric learning objectives, such as pair loss and triplet loss, help limit the distance between original and perturbed samples, further enhancing the model's robustness.

One can argue that evaluating with malformed text alone may be insufficient to claim robustness. However, malformed text can reveal a large performance gap between models designed to handle it and models that are not. Ribeiro etal. (2020) shows important progress in comprehensively testing different NLP systems' capabilities by including a large and diverse number of test cases to their test sets. They also include robustness to typos as one of the capabilities that they suggest to consider. This shows that malformed text is still a necessary aspect of robustness.

In future work, I hope to experiment with more aspects of robustness beyond malformed texts. I hope to include linguistic knowledge to create a behavioral testing framework that covers more aspects of robustness. I also would like to build models that are robust under this framework. One of the directions towards robustness is metric learning, where one can train a model that learns a distance metric that keeps original and perturbed samples close together in an embedding space. I hope that adversarial frameworks and models in this thesis can be extended to create one coherent framework for training and testing robust models.

Table6.1: Comparison of all adversarial attacks in this thesis: the black-box attack does not have access to the model's parameters, while the white-box attack has a full-access to the model's parameters.

Chapter	Black-Box/White-Box	Description
4	Black-Box	Adversarial examples are generated using a rule-based model that is inspired by unintentional errors in Thai. The generated typographical errors are transposition and deletion errors along with their variants that only focus on tonal characters.
5	Black-Box	Adversarial examples are generated using five spelling errors: insertion, transposition, deletion, substitution, and capitalization
5	White-Box	Adversarial examples are generated with the knowledge of the model's parameters: 1. Select a word position with the largest gradient $i^* \leftarrow \underset{i \notin perturbed\_list}{\arg\max} \|\nabla_{\mathrm{x}_i} \mathcal{L}_{\mathrm{tar}}\|_2$ 2. Pick a perturbation that would give the largest loss $\widehat{\mathrm{x}}^* \leftarrow \underset{\hat{x} \in C}{\arg\max} \, [\hat{\mathrm{x}} - \mathrm{x}_i]^\top \nabla_{\mathrm{x}_i} \mathcal{L}_{\mathrm{tar}}$ $X[i^*] \leftarrow \hat{x}^*$ 3. Repeat until there are no word left to perturb.

Table6.2: Comparison of models in chapter 4 and 5

Chapter	Model	Summary
4	BiLSTM	A weak baseline model
4	BiLSTM-ELMo	A strong baseline model with embeddings that is robust to spelling mistakes
4	BiLSTM-U	Mask training data with UNK tokens
4	BiLSTM-U-B	The additional backoff (B) component gives the model access to character-level information.
4	BiLSTM-U-BC	The conditional initialization (C) component initializes the LSTM hidden state for composing character-level information based on affixation.
4	BiLSTM-U-BA	The attention (A) component combines character-level information into a word representation using self-attention mechanism.
4	BiLSTM-U-BCA	A combination of all previous techniques
4	BiLSTM-U-BCAD	The untied bidirectional self-attention mechanism (D) has two separated self-attention mechanisms for forward and backward LSTM hidden states. This enhances the interpretability of the model.
5	ELMo-BiLSTM-CRF	A strong and robust baseline model for sequential tagging tasks
5	ELMo-BiLSTM-CRF (AT)	Adversarial Training (AT) presents the model with adversarial examples during training to improve its robustness.
5	ELMo-BiLSTM-CRF (AT-P)	Pair loss (P) minimizes the similarity between the original and perturbed sequences.
5	ELMo-BiLSTM-CRF (AT-T)	Triplet loss (T) minimizes the similarity between the original and perturbed sequences and maximizes the difference between unique textual sequences.

REFERENCES

Akbik, A., Blythe, D., and Vollgraf, R. 2018. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics, pp. 1638–1649. Santa Fe, New Mexico, USA: Association for Computational Linguistics.

Akbik, A., Bergmann, T., and Vollgraf, R. 2019. Pooled contextualized embeddings for named entity recognition. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 724–728.

Al-Rfou, R. and Skiena, S. 2012. Speedread: A fast named entity recognition pipeline. In Proceedings of COLING 2012, pp. 51–66.

Aroonmanakun, W. 2002. Collocation and thai word segmentation. In Proceedings of the 5th SNLP & 5th Oriental COCOSDA Workshop, pp. 68–75.

Aroonmanakun, W. etal. 2007. Thoughts on word and sentence segmentation in thai. In Proceedings of the Seventh Symposium on Natural language Processing, Pattaya, Thailand, December 13–15, pp. 85–90.

Asanee, K. and Chalathip, T. 1997. A statistical approach to thai morphological analyzer. In Fifth Workshop on Very Large Corpora.

Belinkov, Y. and Bisk, Y. 2018. Synthetic and natural noise both break neural machine translation. In International Conference on Learning Representations.

Bengio, Y., Simard, P., and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5.2 (1994): 157–166.

Bodapati, S., Yun, H., and Al-Onaizan, Y. 2019. Robustness to capitalization errors in named entity recognition. In Proceedings of the 5th Workshop on Noisy

User-generated Text (W-NUT 2019), pp. 237–242. Hong Kong, China: Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5 (2017): 135–146.

Boonkwan, P. and Supnithi, T. 2017. Bidirectional deep learning of context representation for joint word segmentation and pos tagging. In International Conference on Computer Science, Applied Mathematics and Applications, pp. 184–196.

Boonkwan, P., Supnithi, T., Pailai, J., and Kongkachandra, R. 2013. Gradient-descent error correction of pos tagging. Proceedings of SNLP (2013)

Brown, T., Mane, D., Roy, A., Abadi, M., and Gilmer, J. 2017. Adversarial patch.

Burchardt, A., Macketanz, V., Dehdari, J., Heigold, G., Peter, J.-T., and Williams, P. 2017. A linguistic evaluation of rule-based, phrase-based, and neural mt engines. The Prague Bulletin of Mathematical Linguistics 108.1 (2017): 159–170.

Burlot, F. and Yvon, F. 2017. Evaluating the morphological competence of machine translation systems. In Proceedings of the Second Conference on Machine Translation, pp. 43–55. Copenhagen, Denmark: Association for Computational Linguistics.

Chanlekha, H. and Kawtrakul, A. 2004. Thai named entity extraction by incorporating maximum entropy model with simple heuristic information. In Proceedings of the IJCNLP.

Chiu, J.P. and Nichols, E. 2016. Named entity recognition with bidirectional LSTM-CNNs. Transactions of the Association for Computational Linguistics 4 (2016): 357–370.

Chormai, P., Prasertsom, P., and Rutherford, A. 2019. Attacut: A fast and accurate neural thai word segmenter. arXiv preprint arXiv:1911.07056 (2019)

Chunwijitra, V., Chotimongkol, A., and Wutiwiwatchai, C. 2016. A hybrid input-type recurrent neural network for lvcsr language modeling. EURASIP Journal on Audio, Speech, and Music Processing 2016.1 (Aug 2016): 15.

Derczynski, L., Ritter, A., Clark, S., and Bontcheva, K. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013, pp. 198–206.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. 2018. Hotflip: White-box adversarial examples for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 31–36.

Firth, J.R. 1957. A synopsis of linguistic theory, 1930-1955. Studies in linguistic analysis (1957)

Fries, C.C. 1952. The structure of english. (1952)

Gao, J., Lanchantin, J., Soffa, M.L., and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops (SPW), pp. 50–56.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., and Zettlemoyer, L.S. 2017. Allennlp: A deep semantic natural language processing platform.

Gardner, M., Artzi, Y., Basmova, V., Berant, J., Bogin, B., Chen, S., Dasigi, P., Dua, D., Elazar, Y., Gottumukkala, A., etal. 2020. Evaluating nlp models via contrast sets. arXiv preprint arXiv:2004.02709 (2020)

Goodfellow, I., Bengio, Y., and Courville, A. 2016. Deep Learning. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I.J., Shlens, J., and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representations (ICLR).

Haruechaiyasak, C., Kongyoung, S., and Dailey, M. 2008. A comparative study on thai word segmentation approaches. In 2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, volume1, pp. 125–128.

Haruechaiyasak, C. and Kongthon, A. 2013. Lextoplus: A thai lexeme tokenization and normalization tool. WSSANLP-2013 (2013): 9.

Haspelmath, M. 2011. The indeterminacy of word segmentation and the nature of morphology and syntax. Folia linguistica 45.1 (2011): 31–80.

Heigold, G., Varanasi, S., Neumann, G., and van Genabith, J. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers), pp. 68–80. Boston, MA: Association for Machine Translation in the Americas.

Heinzerling, B. and Strube, M. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 273–291. Florence, Italy: Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. 1997. Long short-term memory. Neural computation 9.8 (1997): 1735–1780.

Huang, Z., Xu, W., and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)

Iwasaki, S., Ingkaphirom, P., and Horie, I.P. 2005. A reference grammar of Thai. Cambridge University Press.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and DauméIII, H. 2015. Deep unordered composition rivals syntactic methods for text classification. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), volume1, pp. 1681–1691.

Jia, R. and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2021–2031. Copenhagen, Denmark: Association for Computational Linguistics.

Jiang, Y., Hu, C., Xiao, T., Zhang, C., and Zhu, J. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3585–3590. Hong Kong, China: Association for Computational Linguistics.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)

Karpukhin, V., Levy, O., Eisenstein, J., and Ghazvininejad, M. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019), pp. 42–47. Hong Kong, China: Association for Computational Linguistics.

King, M. and Falkedal, K. 1990. Using test suites in evaluation of machine translation systems. In COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics.

Kingma, D.P. and Ba, J. 2015. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015).

Kittinaradorn, R., Chaovavanich, K., Achakulvisut, T., and Kaewkasi, C. 2017. Deepcut - A Thai word tokenization library using Deep Neural Network. urlhttps://github.com/rkcosmos/deepcut.

Koenig, J.-P. and Muansuwan, N. 2005. The syntax of aspect in thai. Natural Language and Linguistic Theory 23.2 (2005): 335–380.

Koller, D. and Friedman, N. 2009. Probabilistic graphical models: principles and techniques. MIT press.

Kosawat, K., Boriboon, M., Chootrakool, P., Chotimongkol, A., Klaithin, S., Kongyoung, S., Kriengket, K., Phaholphinyo, S., Purodakananda, S., Thanakulwarapas, T., and Wutiwiwatchai, C. 2009. Best 2009 : Thai word segmentation software contest. In 2009 Eighth International Symposium on Natural Language Processing, pp. 83–88.

Kriengket, K., Jumpathong, S., Boonkwan, P., and Supnithi, T. 2017. A cognitive and linguistic analysis of search queries of an online dictionary: A case study of lexitron. Artificial Intelligence and Natural Language Processing (iSAI-NLP 2017) (2017): 1.

Kruengkrai, C., Sornlertlamvanich, V., and Isahara, H. 2006. A conditional random field framework for thai morphological analysis. In Proceedings of LREC, pp. 2419–2424.

Lafferty, J., McCallum, A., Pereira, F., etal. 2001a. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the eighteenth international conference on machine learning, ICML, volume1, pp. 282–289.

Lafferty, J.D., McCallum, A., and Pereira, F. C.N. 2001b. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, p. 282289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. 2016a. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. 2016b. Neural architectures for named entity recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270. San Diego, California: Association for Computational Linguistics.

Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkedal, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L., and Arnold, D. 1996. TSNLP - test suites for natural language processing. In COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics.

Limkonchotiwat, P., Phatthiyaphaibun, W., Sarwar, R., Chuangsuwanich, E., and Nutanong, S. 2020. Domain adaptation of Thai word segmentation models using stacked ensemble. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 3841–3847. Online: Association for Computational Linguistics.

Lin, Z., Feng, M., Santos, C. N.d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. 2017. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)

Ling, W., Dyer, C., Black, A.W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and Luís, T. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1520–1530. Lisbon, Portugal: Association for Computational Linguistics.

Linzen, T., Dupoux, E., and Goldberg, Y. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. Transactions of the Association for Computational Linguistics 4 (2016): 521–535.

Liu, H., Zhang, Y., Wang, Y., Lin, Z., and Chen, Y. 2020. Joint character-level word embedding and adversarial stability training to defend adversarial text. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pp. 8384–8391. : AAAI Press.

Luong, T., Socher, R., and Manning, C.D. 2013. Better word representations with recursive neural networks for morphology. In CoNLL, pp. 104–113.

Ma, X. and Xia, F. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1337–1348. Baltimore, Maryland: Association for Computational Linguistics.

Michel, P., Li, X., Neubig, G., and Pino, J. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 3103–3114. Minneapolis, Minnesota: Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In Burges, C. J.C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K.Q. (ed.), Advances in Neural Information Processing Systems 26, pp. 3111–3119. : Curran Associates, Inc.

Mikolov, T. etal. 2012. Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April 80 (2012): 26.

Minegishi, M. 2011. Description of thai as an isolating language. Social Science Information 50.1 (2011): 62–80.

Miyato, T., Dai, A.M., and Goodfellow, I.J. 2017. Adversarial training methods for semi-supervised text classification. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. : OpenReview.net.

Nebhi, K., Bontcheva, K., and Gorrell, G. 2015. Restoring capitalization in #tweets. In Proceedings of the 24th International Conference on World Wide Web, WWW 15 Companion, p. 11111115. New York, NY, USA: Association for Computing Machinery.

Nguyen, N. and Guo, Y. 2007. Comparisons of sequence labeling algorithms and extensions. In Proceedings of the 24th International Conference on Machine Learning, ICML '07, p. 681688. New York, NY, USA: Association for Computing Machinery.

Pascanu, R., Mikolov, T., and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, p. III1310III1318. : JMLR.org.

Pennington, J., Socher, R., and Manning, C. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. 2018. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Piktus, A., Edizel, N.B., Bojanowski, P., Grave, E., Ferreira, R., and Silvestri, F. 2019. Misspelling oblivious word embeddings. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 3226–3234. Minneapolis, Minnesota: Association for Computational Linguistics.

Ponomareva, N., Rosso, P., Pla, F., and Molina, A. 2007. Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP, pp. 479–483.

Poowarawan, Y. 1986. Dictionary-based thai syllable separation. In Proc. Ninth Electronics Engineering Conference (EECON-86), Thailand, pp. 409–418.

Pruthi, D., Dhingra, B., and Lipton, Z.C. 2019. Combating adversarial misspellings with robust word recognition. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5582–5591. Florence, Italy: Association for Computational Linguistics.

Rabiner, L.R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77.2 (1989): 257–286.

Ratinov, L. and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), pp. 147–155. Boulder, Colorado: Association for Computational Linguistics.

Rawlinson, G.E. 1976. The significance of letter position in word recognition. PhD thesis, University of Nottingham.

Ribeiro, M.T., Wu, T., Guestrin, C., and Singh, S. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4902–4912. Online: Association for Computational Linguistics.

Ritter, A., Clark, S., Etzioni, O., etal. 2011. Named entity recognition in tweets: an experimental study. In Proceedings of the conference on empirical methods in natural language processing, pp. 1524–1534.

Rychalska, B., Basaj, D., Gosiewska, A., and Biecek, P. 2019. Models in the wild: On corruption robustness of neural nlp systems. In Gedeon, T., Wong, K.W., and Lee, M. (ed.), Neural Information Processing, pp. 235–247. Cham: Springer International Publishing.

Schroff, F., Kalenichenko, D., and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823.

Sennrich, R. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 376–382. Valencia, Spain: Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Sornlertlamvanich, V. 1993. Word segmentation for thai in machine translation system (in thai). (01 1993)

Sornlertlamvanich, V., Takahashi, N., and Isahara, H. 1998. Thai part-of-speech tagged corpus: Orchid. In Proceedings of the Oriental COCOSDA Workshop, pp. 131–138.

Straková, J., Straka, M., and Hajic, J. 2019. Neural architectures for nested NER through linearization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5326–5331. Florence, Italy: Association for Computational Linguistics.

Suwanno, N., Suzuki, Y., and Yamazaki, H. 2007. Selecting the optimal feature sets for Thai named entity extraction. Proceedings of ICEE-2007 & PEC 5 (2007)

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. 2014. Intriguing properties of neural networks. In International Conference on Learning Representations.

Theeramunkong, T. and Usanavasin, S. 2001. Non-dictionary-based thai word segmentation using decision trees. In Proceedings of the first international conference on Human language technology research, pp. 1–5.

Tirasaroj, N. and Aroonmanakun, W. 2009. Thai named entity recognition based on conditional random fields. In Natural Language Processing, 2009. SNLP'09. Eighth International Symposium on, pp. 216–220.

Tirasaroj, N. and Aroonmanakun, W. 2011. The effect of answer patterns for supervised named entity recognition in Thai. In PACLIC, pp. 392–399.

Tjong KimSang, E.F. and DeMeulder, F. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp. 142–147.

Treeratpituk, P. 2017. Thai Word-Segmentation with Deep Learning in Tensorflow. urlhttps://github.com/pucktada/cutkum.

Udomcharoenchaikit, C., Vateekul, P., and Boonkwan, P. 2017. Thai named-entity recognition using variational long short-term memory with conditional random field. The Joint International Symposium on Artificial Intelligence and Natural Language Processing (2017): 82–92.

Udomcharoenchaikit, C., Boonkwan, P., and Vateekul, P. 2020. Adversarial evaluation of robust neural sequential tagging methods for thai language. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 19.4 (May 2020)

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. 2019. Universal adversarial triggers for attacking and analyzing NLP. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2153–2162. Hong Kong, China: Association for Computational Linguistics.

Weng, R., Huang, S., Zheng, Z., DAI, X.-Y., and Jiajun, C. 2017. Neural machine translation with word predictions. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 136–145.

Williams, D. and Hinton, G. 1986. Learning representations by back-propagating errors. Nature 323.6088 (1986): 533–538.

Wong, E., Schmidt, F., and Kolter, Z. 2019. Wasserstein adversarial examples via projected sinkhorn iterations. In International Conference on Machine Learning, pp. 6808–6817.

Xin, Y., Hart, E., Mahajan, V., and Ruvini, J.-D. 2018. Learning better internal structure of words for sequence labeling. In Proceedings of the 2018 Conference on

Empirical Methods in Natural Language Processing, pp. 2584–2593. Brussels, Belgium: Association for Computational Linguistics.

Yasunaga, M., Kasai, J., and Radev, D. 2018. Robust multilingual part-of-speech tagging via adversarial training. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 976–986. New Orleans, Louisiana: Association for Computational Linguistics.

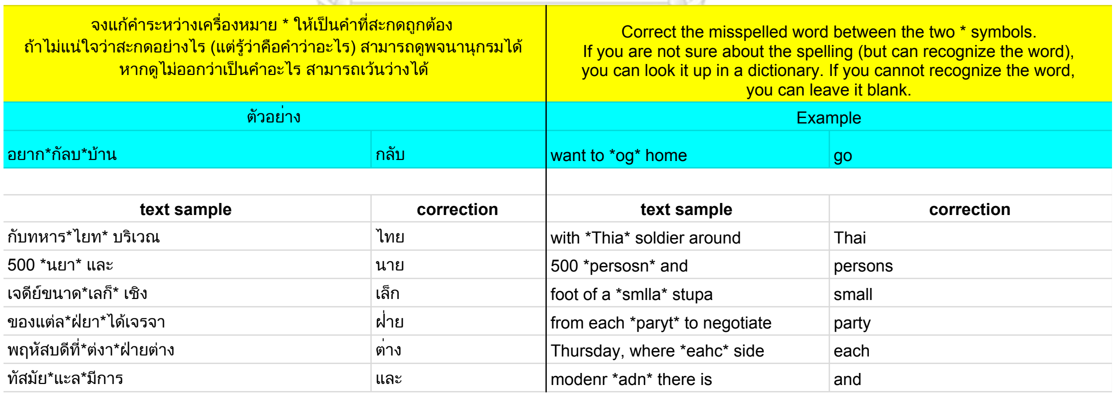
Zhou, J.T., Zhang, H., Jin, D., Peng, X., Xiao, Y., and Cao, Z. 2019. Roseq: Robust sequence labeling. IEEE transactions on neural networks and learning systems (2019)

Appendix I

SUPPLEMENTARY MATERIAL OF CHAPTER 4

A.1 Text samples from the perturbed BEST2010 test set (stress level = 10)

In order to confirm that humans can understand textual samples from our adversarial evaluation scheme, we sampled 100 text samples from the stress level 10 perturbed BEST2010 test set to create a word correction exercise and we asked 3 participants to correct them. Full details including the exercise solution, the result of each participant, can be found on `https://github.com/c4n/Thai-Adversarial-Evaluation/blob/master/survey/word_correction_exercise_results.ods`



จงแก้คำระหว่างเครื่องหมาย * ให้เป็นคำที่สะกดถูกต้อง ถ้าไม่แน่ใจว่าสะกดอย่างไร (แต่รู้ว่าคือคำว่าอะไร) สามารถดูพจนานุกรมได้ หากดูไม่ออกว่าเป็นคำอะไร สามารถเว้นว่างได้		Correct the misspelled word between the two * symbols. If you are not sure about the spelling (but can recognize the word), you can look it up in a dictionary. If you cannot recognize the word, you can leave it blank.	
ตัวอย่าง		Example	
อยาก*กัลบ*บ้าน	กลับ	want to *og* home	go
text sample	**correction**	**text sample**	**correction**
กับทหาร*ไยท* บริเวณ	ไทย	with *Thia* soldier around	Thai
500 *นยา* และ	นาย	500 *persosn* and	persons
เจดีย์ขนาด*เลก็* เชิง	เล็ก	foot of a *smlla* stupa	small
ของแต่ล*ฝ่ยา*ได้เจรจา	ฝ่าย	from each *paryt* to negotiate	party
พฤหัสบดีที่*ต่งา*ฝ่ายต่าง	ต่าง	Thursday, where *eahc* side	each
ทัสมัย*แะล*มีการ	และ	modenr *adn* there is	and

FigureA.1: A snapshot from the word correction exercise with 6 text samples with their solutions along with its English translation

A.2 Examples of predictions on perturbed inputs

Table A.1 compares the outputs from four different models. Models that incorporate character-level information, BiLSTM-ELMo and BiLSTM-BCAD-clp, were more robust even when all the surrounding words were misspelled. The four models predicted the part-of-speech of the observed word correctly in the first three scenarios. However, in the last scenario where the observed word was perturbed and was surrounded by misspelled words, BiLSTM-ELMo and BiLSTM-BCAD-clp could predict correctly. At the same time, models without character-level information, BiLSTM and BiLSTM-U, failed to predict the part-of-speech of the observed word correctly. The definitions of part-of-speech tags shown in Table A.1 are as follows:

- ADV - adverb
- DDEM - demonstrative determiner
- FWN - non-proper noun written in non-Thai script
- FXAV - adverbial prefix
- JJA - noun-modifying adjective
- NN - non-proper noun
- NR - proper noun
- P - preposition
- PPER - personal pronoun
- PU - punctuation
- VA - predicate adjective

TableA.1: Examples of part-of-speech predictions of a Thai sentence in each scenario, as discussed in section 4.4.4. This table focuses on the observed word รู้สึก, but it also shows the prediction of surrounding words. Red text denotes incorrect prediction.

original sentence without perturbation						
	วัน	นี้	ฉัน	รู้สึก	เมื่อย	มาก
Ground Truth	**NN**	**DDEM**	**PPER**	**VV**	**VV**	**ADV**
1) an observed word without perturbation in a low-noise context						
	วัน	นี้	ฉัน	รู้สึก	ยเมือ	มาก
BiLSTM	NN	DDEM	PPER	VV	VA	ADV
+ELMo	NN	DDEM	PPER	VV	VV	ADV
+U	NN	DDEM	PPER	VV	FXAV	JJA
+BCAD-clp	NN	DDEM	PPER	VV	VV	ADV
2) an observed word with perturbation in a low-noise context						
	วัน	นี้	ฉัน	สึกรู	ยเมือ	มาก
BiLSTM	NN	DDEM	P	VV	ADV	ADV
+ELMo	NN	DDEM	PPER	VV	ADV	ADV
+U	NN	DDEM	PPER	VV	FXAV	JJA
+BCAD-clp	NN	DDEM	PPER	VV	VV	ADV
3) an observed word without perturbation in a high-noise context						
	วั	นี	ฉั	รู้สึก	ยเมือ	มา
BiLSTM	NR	NN	NR	VV	VV	VV
+ELMo	NN	DDEM	PPER	VV	ADV	ADV
+U	NN	NR	PU	VV	VV	VV
+BCAD-clp	NN	DDEM	PPER	VV	VV	ADV
4) an observed word with perturbation in a high-noise context						
	วั	นี	ฉั	สึกรู	ยเมือ	มา
BiLSTM	FWN	NN	NR	NR	VV	VV
+ELMo	NN	DDEM	PPER	VV	VV	ADV
+U	NN	NR	PU	NR	VV	VV
+BCAD-clp	NN	DDEM	PPER	VV	VV	ADV

- VV - verb