การเพิ่มข้อมูลสำหรับระบบประมวลภาษาธรรมชาติภาษาไทยโดยใช้การแบ่งเป็นโทเค็นที่แตกต่างกัน

นายปฐวี ปราการกมานันท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

DATA AUGMENTATION FOR THAI NATURAL LANGUAGE PROCESSING USING DIFFERENT TOKENIZATION

Mr. Patawee Prakrankamanant

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2021

Thesis Title	DATA AUGMENTATION FOR THAI NATURAL LANGUAGE PROCESSING USING DIFFERENT TOKENIZATION
By	Mr. Patawee Prakrankamanant
Field of Study	Computer Engineering
Thesis Advisor	Assc. Prof. Atiwong Suchato, Ph.D.
Thesis Co-advisor	Ekapol Chuangsuwanich, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

. . . . . Dean of the Faculty of Engineering

(Prof. Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

. . . . . Chairman

(Assc. Prof. Proadpran Punyabukkana, Ph.D.)

. . . . . Thesis Advisor

(Assc. Prof. Atiwong Suchato, Ph.D.)

. . . . . Thesis Co-advisor

(Ekapol Chuangsuwanich, Ph.D.)

. . . . . External Examiner

(Prachaya Boonkwan, Ph.D.)

ปฐวี ปราการกมานันท์ : การเพิ่มข้อมูลสำหรับระบบประมวลภาษาธรรมชาติภาษาไทยโดยใช้การแบ่งเป็นโทเค็นที่แตกต่างกัน. (DATA AUGMENTATION FOR THAI NATURAL LANGUAGE PROCESSING USING DIFFERENT TOKENIZATION) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร.อติวงศ์ สุชาโต, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : อ. ดร.เอกพล ช่วงสุวนิช 54 หน้า.

การทำให้เป็นโทเค็น (tokenization) เป็นหนึ่งในขั้นตอนการดำเนินการเบื้องต้น (pre-processing) ในระบบของแบบจำลองแบ่งประเภทข้อความ (text classification model) และเป็นส่วนหนึ่งที่ส่งผลต่อประสิทธิภาพของแบบจำลอง แต่อย่างไรก็ตามการทำให้เป็นโทเค็น ไมใช่ปัญหาทั่วไปสำหรับ noisy text หรือ ภาษาที่ไม่มีขอบเขตของคำ (word boundary) ที่ชัดเจนเช่น ภาษาไทย ในการศึกษานี้เราได้นำเสนอวิธีการเพิ่มข้อมูล (data augmentation) เพื่อเพิ่มความคงทน (robustness) และประสิทธิภาพโดยการใช้การทำให้เป็นโทเคนหลากหลายรูปแบบ (multi-tokenization) เราวัดผลบนแบบจำลองแบ่งประเภทข้อความภาษาไทย จากผลการศึกษาพบว่าแบบจำลองที่ถูกเรียนรู้ด้วยการเพิ่มข้อมูลที่เรานำเสนอนั้น สามารถคงทนต่อ การตัดคำที่ผิดพลาด และสามารถใช้ร่วมกับ การเพิ่มข้อมูลแบบอื่นด้วย

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่ออ.ที่ปรึกษาหลัก	.....
ปีการศึกษา	2564	ลายมือชื่ออ.ที่ปรึกษาร่วม	.....

6170204021: MAJOR COMPUTER ENGINEERING
KEYWORDS: MACHINE LEARNING / NATURAL LANGUAGE PROCESSING / TOKENIZATION

PATAWEE PRAKRANKAMANANT : DATA AUGMENTATION FOR THAI NATURAL LANGUAGE PROCESSING USING DIFFERENT TOKENIZATION. ADVISOR : ASSIST. PROF. ATIWONG SUCHATO, Ph.D., THESIS CO-ADVISOR : EKAPOL CHUANGSUWANICH, Ph.D., 54 pp.

Tokenization is one of the most important data pre-processing steps in the text classification task and also one of the main contributing factors in the model performance. However, getting good tokenizations is non-trivial when the input is noisy, and is especially problematic for languages without an explicit word delimiter such as Thai. Therefore, we proposed an alternative data augmentation method to improve the robustness of poor tokenization by using multiple tokenizations. We evaluated the performance of our algorithms on different Thai text classification datasets. The results suggested our augmentation scheme makes the model more robust to tokenization errors and can be combined well with other data augmentation schemes.

Department	: Computer Engineering	Student's Signature	.....
Field of Study	: Computer Engineering	Advisor's Signature	.....
Academic Year	: 2021	Co-advisor's signature	.....

CONTENTS

Page

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

Chapter I

INTRODUCTION

1.1 Motivation

Natural language processing (NLP) is one of the subfields of artificial intelligence (AI) and linguistics that study intercommunication between humans and computers and how computers analyze human language. The examples of NLP systems help to solve the problems such as text classification, part of speech tagging, named entity recognition, machine translation, etc. The core of NLP system usually uses a machine learning model which is why the NLP system has to preprocess data before feeding it to the machine learning model.

Tokenization is the first process of text data analysis and is a common problem in text classification. In the tokenization process, the algorithm tries to segment sentences into useful tokens sequentially such as characters, words, and subwords [Zhang et al. (2015); Joulin et al. (2017)]. In deep learning, tokens are usually fed to the embedding layers followed by other layers such as multilayer perceptron (MLP) [Iyyer et al. (2015)], convolutional neural network (CNN) [Chen (2015)], and long short-term memory (LSTM) network [Liu et al. (2016a)].

Tokenization in English is a trivial problem because it has white-space for separating words into sentences, while tokenization in Thai, Chinese, Japanese, and other languages with no distinct word boundaries is more challenging. The tokenization choice of a system has a significant impact on the performance of text classification. For Thai, different tokenization methods such as maximum matching, longest matching, and supervised machine learning have been widely used to tokenize and address these problems. For instance, in Thai language, there are ambiguous homographs such as “ตากลม”, which can be tokenized into two ways; “ตา-กลม” (round eyes) or “ตาก-ลม” (feel the wind). Tokenization in noisy text data

such as text in non-official websites and social text also pose a significant problem. We often see these internet variant informal text chatting in Thai such as “มาก กก” (lotssss, the final consonant duplicated for emphasis) of the official spelling of the same word “มาก”. This word can be wrongly tokenized as “มาก” (very) and “กก” (root) [Heigold et al. (2018)]. The word embedding that is trained from other datasets might not be satisfied due to out-of-vocabulary (OOV). There are works that have been demonstrated to solve this problem such as robust tokenizer [Remus et al. (2016)], tokenizer with POS tagging [Liu et al. (2020)], and robust word embeddings [Wang et al. (2020)].

To improve the robustness of underperforming tokenization on the natural language processing model, we applied various types of tokenization to the noisy text data, serving as a data augmentation, which helps increase the model’s robustness. Data augmentation is one of the famous techniques to increase the performance and generalization of machine learning models. Augmentation techniques are used to increase the amount of data by adding slightly modified copies or newly created data from existing data [Goodfellow et al. (2016)]. In computer vision, data augmentations are done almost everywhere to get the effect of a larger training data to make models generalize better. In the natural language processing (NLP) field, conversely, it is hard to augment text due to the high complexity of language. By changing a single word, the surrounding context will be totally different. However, these different tokenizations could be alternatively used to generate more data for the Thai NLP model.

For this reason, we proposed a novel data augmentation method for Thai language by using different and possibly errorful tokenizations to improve the Thai text classification. The data augmentation system in our study is shown in Fig. 1.1. Different tokenization methods were used to augment the data while keeping the vocabulary fixed. This can be used to train different kinds of models. To handle potential OOVs from different tokenization outputs, we also used character embeddings to augment the word representation. The results from our experiments showed

that augmenting the dataset in this way can improve text classification performance and robustness in several model settings and datasets.

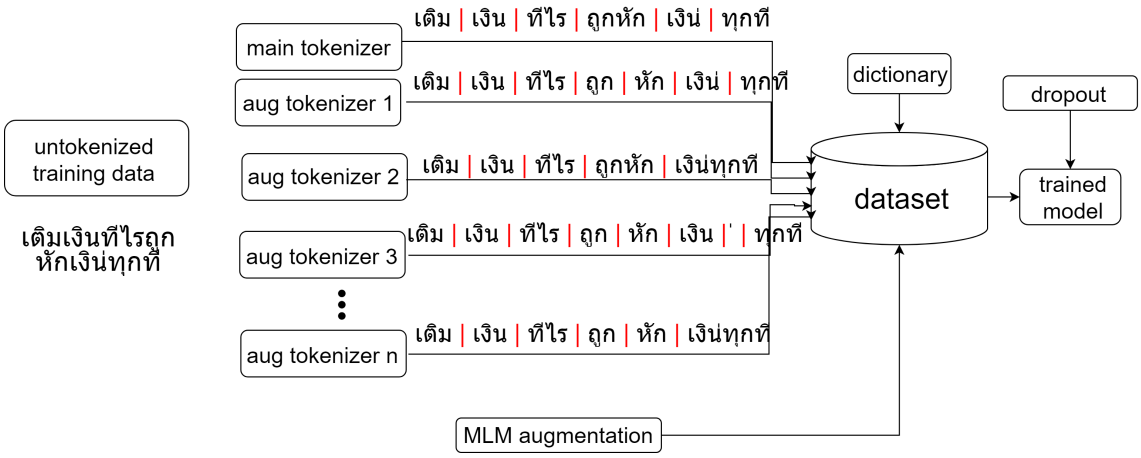

Figure 1.1: An overview of our data augmentation method. In this example "เติมเงินทีไรถูกหักเงิน่ทุกที", can be tokenized as "เติม | เงิน | ทีไร | ถูกหัก | เงิน่ | ทุกที" using the main tokenizer and the other tokenizers can be used to create various token sequences. We trained the model using the vocabulary from the main tokenizer on the combined dataset.

1.2 Research Questions

1. Can data augmentation from different tokenization improve NLP classification model's performance?

2. Can data augmentation from different tokenization improve robustness for error tokenization of NLP classification model?

1.3 Objectives

1. To improve NLP classification model performance by using data augmentation from different tokenization.

2. To improve robustness for error tokenization of NLP classification model by using data augmentation from different tokenization.

1.4 Scope Of Work

This experiment was only focus in Thai language datasets.

Chapter II

BACKGROUND KNOWLEDGE

Tokenization

Tokenization is one step of pre-processing data in NLP system before feeding it to a machine learning model. Tokenization process is breaking raw text to sequence of tokens. Tokens can be sentences, words or characters. Tokenization processes raw text data to be meaningful information.

Token Representation

After tokenization process, a sequence of tokens that represent raw text is obtained. However, tokens are still text information. They cannot be fed to machine learning models because only numerical features are allowed. Token is changed to a numerical feature by using token representation. Tokens are represented with sparse representation such as Term-document matrix, Co-occurrence matrix, Positive Pointwise Mutual Information (PPMI), and Term Frequency–Inverse Document Frequency (TF-IDF) [Jurafsky and Martin (2009)] or dense representation such as SVD-based method [Jurafsky and Martin (2009)] or Word2Vec (embedding) [Mikolov et al. (2013b,a)]. NLP models that use neural network architecture in the main structure of the model usually use embedding to represent tokens.

Embedding

Embedding is one of the famous token representations used in neural network models. Mechanism of embedding uses token's one-hot encoding multiplied with embedding matrix to create dense vector representation. Output vector of embedding must be significantly smaller than one-hot encoding vector. Embedding is usually used in word level tokens (word embedding). The recent of research that improved word embedding such as Skip-gram [Mikolov et al. (2013b)] and CBOW [Mikolov et al. (2013a)].

Classification Model

Classification model is the machine learning model that predicts the class of a given data point. In Fig 2.1, text classification model in NLP is demonstrated. For example, the text classification model tries to predict star score from review text in yelp (restaurant review website), that labels of this text classification model are shown in Fig 2.2. Classification model predicts the probability of each class.

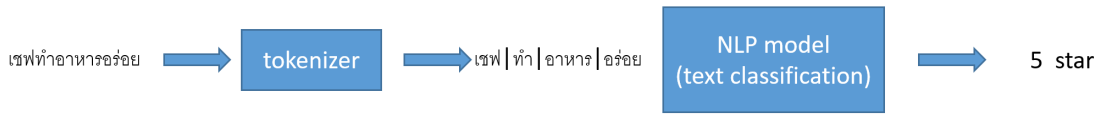

Figure 2.1: Overview of text classification system

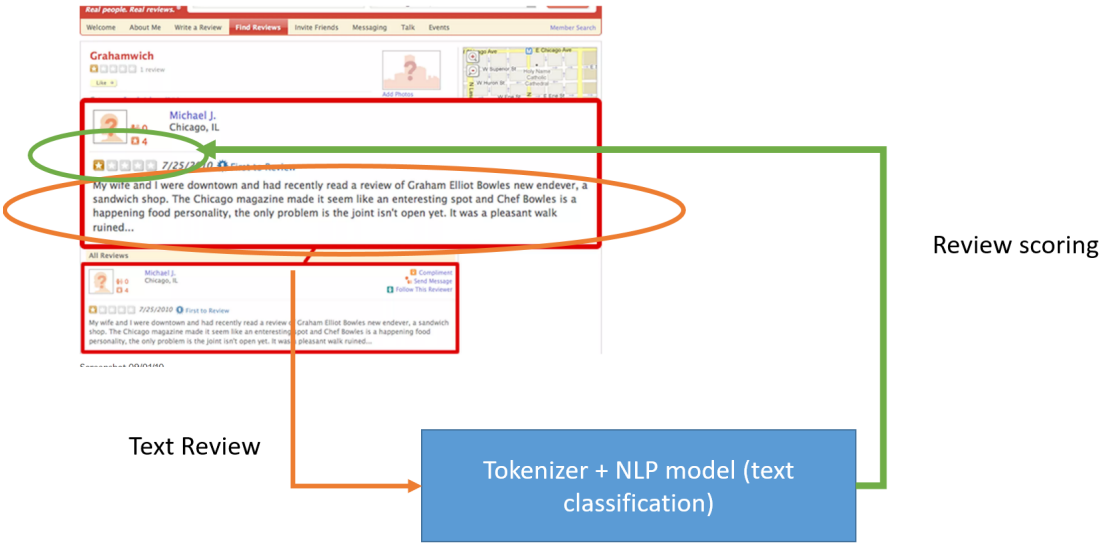

Figure 2.2: Example of text classification model that predict star score from yelp's dataset text review.

Language Model

Language model is the model that predicts words. Language model tries to find patterns in human language and predicts the probability of the next word from information of the previous word by using conditional probability. Language model generates sentences from the start token only (generative model). However, the model has limited capacity to use previous words, so language model use N-gram method in conditional probability, as shown in Equation 2.1.

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1}, w_{i-2}, \ldots, w_1) \tag{2.1}$$

When w_i is i^{th} token in token sequence, $P(w_1, w_2, \ldots, w_n)$ is probability to generate $w_1, w_2, \ldots, w_n$ token sequence.

2.1 Supervised Learning

Supervised learning is the most popular approach that used in various tasks. This learning method required the solution (always called labels or ground truth) to teach the model. When the data fed into the model, it will learn to solve the tasks based on the labels of each data. For this reason, the labels must be cleaned and corrected to make the model understand the correlation between data and labels. The examples of the most important supervised learning algorithms are following ; [Goodfellow et al. (2016)]

- Linear Regression
- Logistic Regression
- Decision Trees and Random Forests
- k-Nearest Neighbors
- Support Vector Machines (SVMs)
- Neural Networks

Unsupervised Learning

For unsupervised learning, this learning system does not require the labels of data (often called unlabeled data). Unsupervised learning tries to learn without a teacher. This approach provides useful clues for how to group examples in representation space.

- Clustering
  - K-Means
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

2.2 Neural Network Model

An artificial neural network learning algorithm, neural network, or deep neural network, is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form, as shown in Fig 2.3. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses. Neural networks are one of many tools and approaches used in machine learning algorithms. The neural network itself may be used as a piece in many different machine learning algorithms to process complex data inputs into a space that computers can understand.

Activation Function

Activation functions or transfer functions are designed to convert an input signal of a node in a neural network to an output signal. Activation function gives the neuron know the bounds of the value whether the neuron should activate or not. Moreover, these functions aim to map between the inputs and response variables.

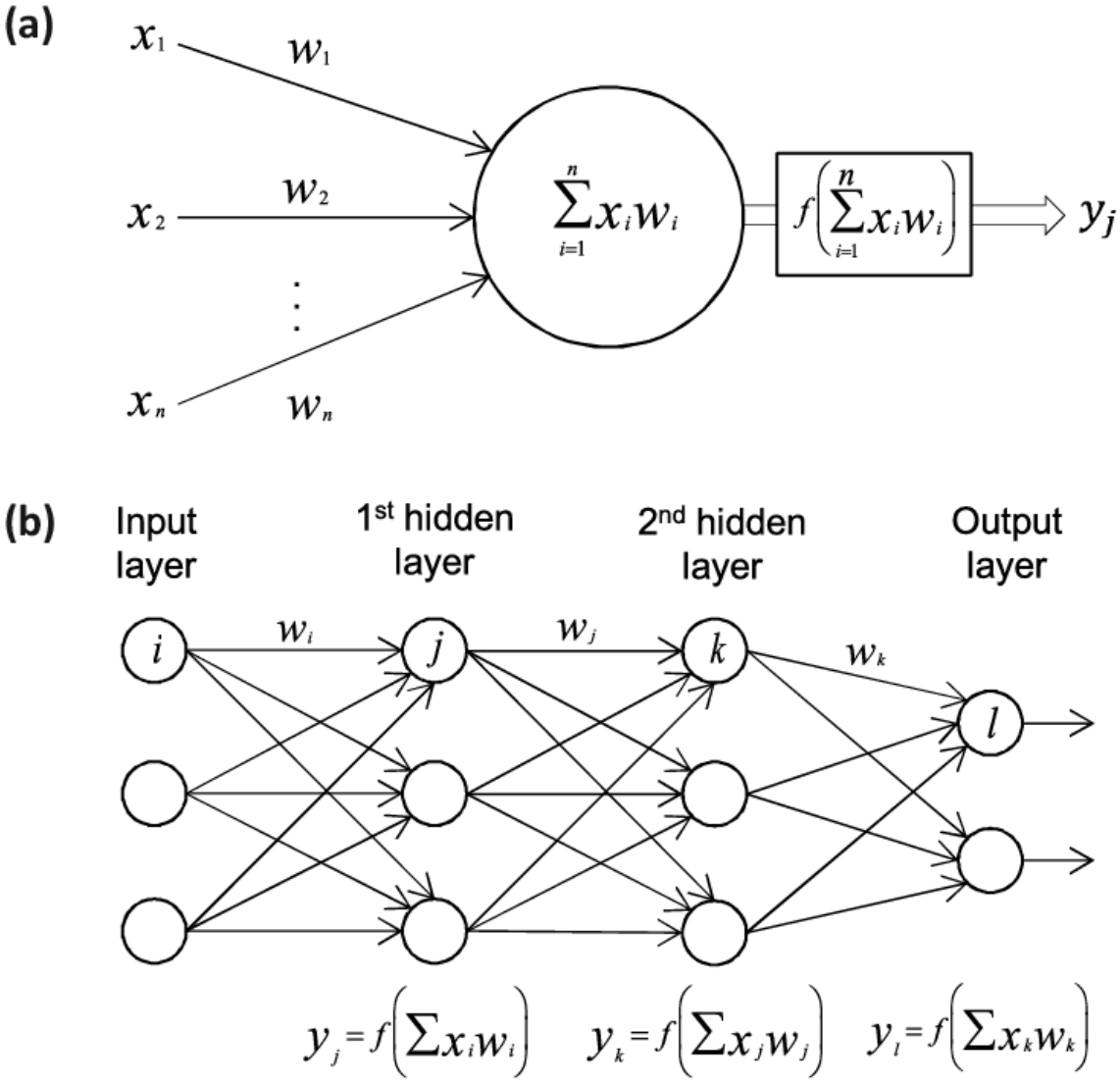

Figure 2.3: Neural network (a) operation in node (neuron) (b) Overview of neural network that has 2 hidden layers

The activation function can be divided into two types which are linear activation function and non-linear activation function.

Linear Activation Function

A linear activation function is a straight line function where activation is proportional to the input, as shown in Equation 2.2.

$$F(x) = x \tag{2.2}$$

when x is input, $F(\cdot)$ is a linear activation function.

The linear function is designed for a non-complex model because it has a polynomial of one degree. For this reason, this function is limited in their complexity and no ability to learn the complex neuron. The output of this function can be any value without boundary.

Non-linear Activation Function

The non-linear activation functions are the most used activation function. These functions make the model to generalize or to deal with complexity between the output. In this experiment, main non-linear activation functions uesd are sigmoid, tanh, ReLU, and softmax activation functions.

In Sigmoid or logistic activation function, in the case of choosing Sigmoid function as an activation function is because the output value after passing the Sigmoid function exists between 0 to 1. Thus, it is always used to predict the probability as an output. Due to this, the probability exists only between the range of 0 and 1. The Sigmoid function is shown in Equation 2.3.

$$F(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

when x is input, $F(\cdot)$ is a sigmoid activation function.

Tanh or hyperbolic tangent activation function is similar to Sigmoid function but Tanh function is more wide-ranging from -1 to 1. The advantage is this function supports negative input and it can also produce negative output. The Tanh function is shown in Equation 2.4.

$$F(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

when x is input, $F(\cdot)$ is the Tanh activation function.

Notice that the Sigmoid and Tanh activation function are used in a feed-forward neural network.

ReLU or Rectified Activation Function Linear Unit is the most used activation function. ReLU is a half rectified look like a linear function but it is a nonlinear function. The output value of this function will be zero if the input value is negative. For the positive input value, the output will be obtained from a linear function. The range of ReLU is from 0 to infinite, as shown in Equation 2.5

$$F(x) = \max(0, x) \tag{2.5}$$

when x is input, $F(\cdot)$ is ReLU activation function.

Softmax Activation Function always used in a classification task. Softmax converts the input logits into probabilities that sum to one. The output of this function is a vector that represents the probability distributions of a list of potential outcomes, as shown in Equation 2.6.

$$F(x)_k = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}} \tag{2.6}$$

when x is input, $F(\cdot)_k$ is k^{th} index of Softmax activation function.

Objective Function

The cost function, lost function, objective function, or criterion is the function that we want to minimize or maximize to measure the performance of the machine learning model. Cost function estimates the error between the prediction results and the actual values. The value of cost function is present in the form of a single real

number. There are many types of the cost function in machine learning but some details about the cost function used in this study is described here.

Cross-Entropy

Cross-Entropy is a cost function for a classification model. The cost function can be used for a binary classification task and a multi-class classification task. The cost computed for every output vector component is independent which does not affect the other values.

$$\mathcal{L}_{CE} = -\sum_{i=1}^{2} t_i \log\left(F(s)_i\right) \tag{2.7}$$

when $\mathcal{L}_{CE}$ is Cross-Entropy loss, t_i is the label of data point with i label ($t_i \in \{0, 1\}$), $F(\cdot)$ is sigmoid or softmax activation function (binary classification task uses sigmoid activation function and multi-class classification task uses softmax activation function).

2.3 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of neural network model that model's input have sequence, order or time-step information such as language model [Jozefowicz et al. (2016); Zoph et al. (2016)], text classification model [Liu et al. (2016b)]. The basic structure of RNN model is shown in Fig 2.4. RNN models use output of previous time steps to be input [Medsker and Jain (2001)]. The famous RNN model has 2 types that are Gate Recurrent Unit (GRU) [chung2014empirical] and Long short-term memory (LSTM) [Malhotra et al. (2015)], as shown in Fig 2.5.

2.4 Metrics

Our experiment used metrics for classification task. We evaluated in a different metric including precision, recall, f1-score, and accuracy. The details of each type of measurement are described separately in each section.

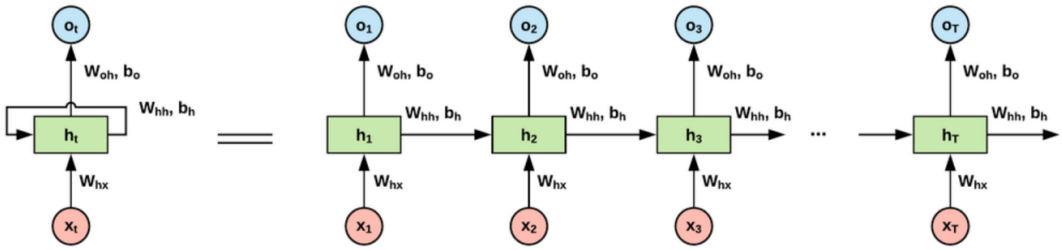

Figure 2.4: Structure of Recurrent Neural Network (RNN)

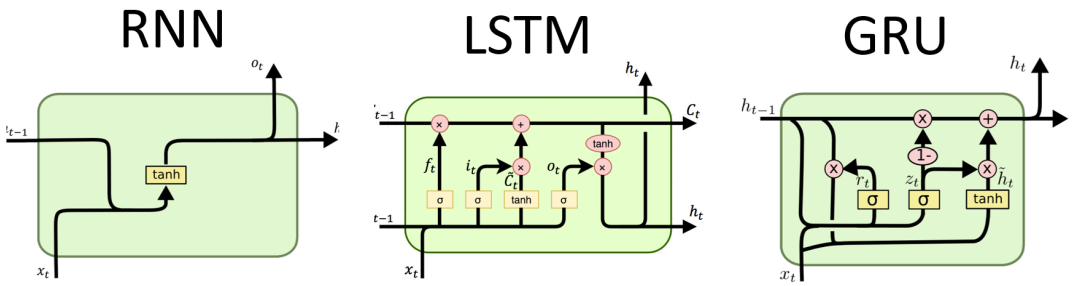

Figure 2.5: Comparison between Recurrent Neural Network (RNN), Long short-term memory (LSTM) and Gate Recurrent Unit (GRU)

	Predicted classes	
Actual classes	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion metric

Accuracy

Accuracy is the most intuitive performance measurement and it is simply a ratio of correctly predicted observations to the total observations. The definition of accuracy is shown in Equation 2.8.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.8}$$

Precision

Precision is the ratio of positive observations correctly predicted to the positive total predicted observations. The definition of precision is shown in Equation 2.9.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.9}$$

Recall

Recall or sensitivity is the ratio of positive observations correctly predicted to the all actual class - yes in observations, The definition of recall is shown in Equation 2.9.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.10}$$

F1-score

F1-score is the weighted average of precision and recall, as shown in Equation 2.11

$$\text{F1-score} = \frac{2}{\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)} = 2\left(\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{2.11}$$

Chapter III

RELATED WORKS

3.1 Dictionary-based Tokenization

Dictionary-based tokenization uses a pre-existing vocabulary. The vocabulary lists all tokens that will be used for tokenization. Different algorithms have been proposed to provide word boundaries based on the vocabulary such as maximum matching and longest matching [Haruechaiyasak et al. (2008)]. They rely on heuristics such as preferring the minimal amount of words or preferring the longest words, respectively. The vocabulary can be derived from real words [Fan et al. (2008); Mikolov et al. (2013a)], subwords (such as syllables) [Van Heerden et al. (2017)], or learned from data [Sennrich et al. (2015); Wu et al. (2016)]. Recent trends have been more focusing on learning subword tokenization without the need to rely on human experts. Vocabulary can be learned from the dataset by using Byte-Pair Encoding (BPE), which creates tokens from the frequency of sticky bytes (in Thai language equals one character). However, word and subword tokenizers can segment a sentence into many types of token sequences [Kudo (2018a)]. From this problem, we used a language model to sample a candidate of token sequence such as SentencePiece [Kudo and Richardson (2018b)] and Stochastic tokenization [Hiraoka et al. (2019)]. However, dictionary-based tokenizers usually have some out-of-vocabulary problems because they are unlikely to correctly anticipate all possible tokens in the actual usage.

Longest Matching

Longest matching is one of algorithms for word segmentation that is dictionary-based tokenization. Longest matching tries to segment the longest possible word with a greedy algorithm [Meknavin et al. (1997)]. The example of segmentation raw text, “ป้ายกลับรถ”, with longest matching is shown in Table 3.1.

Table 3.1: Example of Longest matching tokenize, “ป้ายกลับรถ”, to be (“ป้าย”,“กลับ”,“รถ”) with vocabulary {“ป้า”, “ป้าย”, “กลับ”, “รถ”}

character	is in dictionary ?		token
ป	F	้ายกลับรถ	
ป้	F	ายกลับรถ	
ป้า	T	ยกลับรถ	ป้า
ป้าย	T	กลับรถ	ป้าย
ป้ายก	F	ลับรถ	ป้าย
ก	F	ลับรถ	ป้าย
กล	F	ับรถ	ป้าย
กลั	F	บรถ	ป้าย
กลับ	T	รถ	ป้าย
กลับร	F	ถ	ป้าย, กลับ
ร	F	ถ	ป้าย, กลับ
รถ	T		ป้าย, กลับ, รถ

Maximum matching

Maximum matching is one of algorithms for word segmentation that is dictionary-based tokenization. Maximum matching tries to segment raw text to be the fewest tokens that are possible words with dynamic programming. Maximum matching uses distance function ($d(i,j)$) to calculate the number of tokens in Equation 3.1. The example of segmentation raw text, “ไปหามเหสี”, with maximum matching is shown in Table 3.2.

$$d(i,j) = \begin{cases} 1 & \text{if } i = 1 \,\&\, \text{text}[1:j] \in \mathcal{V} \\ 1 + min_{k=1,..,i-1}(d(k,i-1)) & \text{text}[i:j] \in \mathcal{V} \\ \infty & \text{otherwise} \end{cases} \tag{3.1}$$

when $\mathcal{V}$ is vocabulary, and text$[i:j]$ is characters between i^{th} to j^{th}.

Table 3.2: Maximum matching “ไปหามเหสี” to be (“ไป”,“หา”,“มเหสี”) with vocabulary {“ไ”, “ป”, “ห”, “า”, “ม”, “เ”, “ส”, “ี ”, “ไป”, “หา”, “หาม”, “เห”, “สี”, “มเหสี”}

$d(i,j)$	ไ	ป	ห	า	ม	เ	ห	ส	ี
index	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$
$i=1$	1	1	∞	∞	∞	∞	∞	∞	∞
$i=2$		2	∞	∞	∞	∞	∞	∞	∞
$i=3$			2	2	2	∞	∞	∞	∞
$i=4$				3	∞	∞	∞	∞	∞
$i=5$					3	∞	∞	∞	3
$i=6$						3	3	∞	∞
$i=7$							4	∞	4
$i=8$								4	∞
$i=9$									∞

Subword Tokenization

Subword tokenization is a famous tokenization of machine translation models that use Transformer-based architecture. Subword tokenization tries to split prefix, subfix, and meaning from the original word. For example, “usefulness” is segmented to be “useful” and “ness”. Subword tokenization creates vocabulary from Byte-Pair Encoding (BPE). Byte-Pair Encoding is a simple data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single or unused byte [Kudo and Richardson (2018b); Le et al. (2018)]. We showed how to create vocabulary in Table 3.3. However, subword tokenization can segment raw text to token sequence variously, as shown in Table 3.4. Kudo and team used the Unigram Language Model (ULM) to control the subword tokenizer. We collected token sequences from tokenizer by sampling. ULM creates a probabilistic model from Equation 3.2 and ULM is trained with loss function in Equation 3.3.

$$\begin{aligned} &P(x_1, x_2, \ldots, x_M) = \textstyle\prod_{i=1}^{M} P(x_i) \\ &\forall x_i \in \mathcal{V}, \textstyle\sum_{x \in \mathcal{V}} P(x) = 1 \end{aligned} \tag{3.2}$$

when $x_1, x_2, \ldots, x_M$ is token sequence, $\mathcal{V}$ is vocabulary, $P(x_i)$ is probability to have x_i token in token sequence.

Table 3.3: How to created vocabulary from raw text “Aaabdaaabac” with Byte-pair encoding

	Old text	new token	new text
1	Aaabdaaabac	Z→ aa	ZabdZabac
2	ZabdZabac	Y→ ab	ZYdZYac
3	ZYdZYac	X→ ZY	XdXac
all Token	Z→ aa, Y→ ab, X→ ZY or X→ aaab		

$$\mathcal{L} = -\sum_{s=1}^{|D|} \log(P(X^{(s)})) = -\sum_{s=1}^{|D|} \log\left(\sum_{x \in \mathcal{S}(X^{(s)})} P(x)\right) \tag{3.3}$$

when $\mathcal{L}$ is negative maximum likelihood (loss function), X^s is s^{th} token sequence in dataset D.

Table 3.4: “Hello world” is tokenized with a subword tokenization method that has various possible token sequences.

Subwords (_ means spaces)	Vocabulary id sequence
_Hell / o / _world	13586, 137, 255
_H / ello / _world	320, 7363, 255
_He / llo / _world	579, 10115, 255
_ / He / l / l / o / _world	7, 18085, 356, 356, 137, 255
_H / el / l / o / _ / world	320, 585, 356, 137, 7, 12295

3.2 Machine Learning-based Tokenization

To reduce the OOV problem of dictionary-based methods, a machine learning model can be used to segment tokens from a sentence with a regression model. Machine learning-based tokenizer is widely used in languages that do not have white-space to split words such as Thai [Kittinaradorn et al. (2018); Jousimo et al. (2017)], Japanese [Hanlon (2018)], and Chinese [Huang et al. (2007)]. For deep learning models, the model architecture is usually a neural network with characters as input and character level output [Kittinaradorn et al. (2018)]. The example of machine learning-based tokenizer is shown in Fig 3.1. Recent works used other information to improve the model, including syllable embedding [Chormai et al. (2019)] and POS tagging [Heigold et al. (2018); Remus et al. (2016); Liu et al. (2020)].

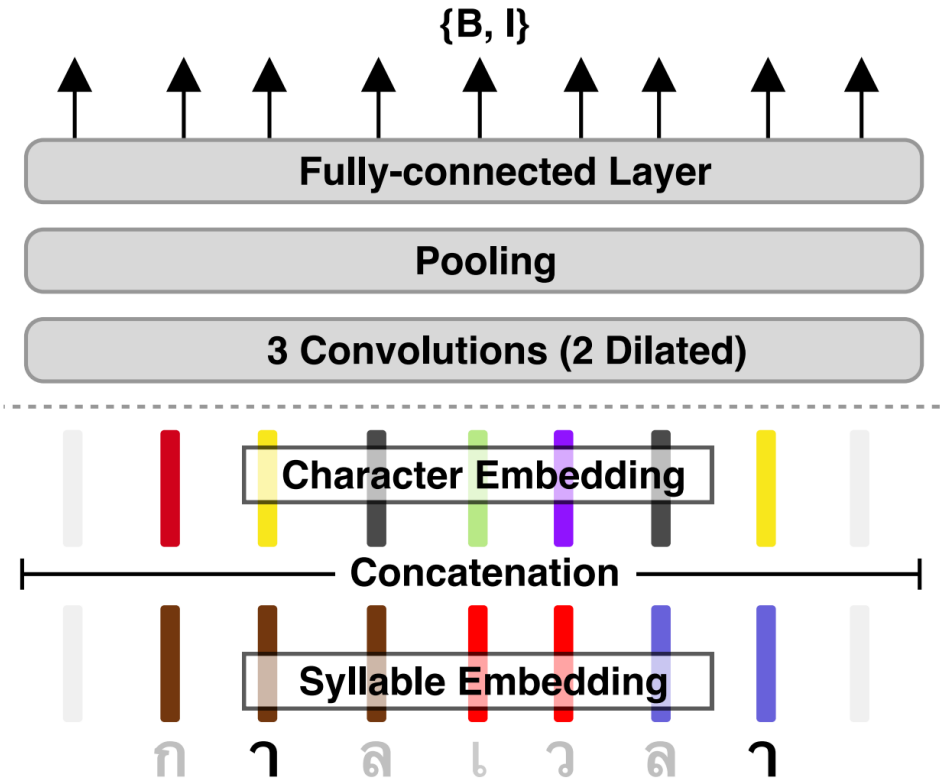

Figure 3.1: example of machine learning based tokenization (adapted from AttaCut [Chormai et al. (2019)])

3.3 Masked Language Model Augmentation

The masked language model (MLM) is a type of deep learning-based language model that is trained by predicting the masked word within a sentence [Devlin et al. (2018)]. The MLM has been used to perform data augmentation by masking certain words in a sentence and have the MLM predict a new word to provide a slightly modified copy of the sentence [Wu et al. (2019a); Shi et al. (2019); Park and Ahn (2019)].

For Thai language, the appropriate choice of the MLM model is WangchanBERTA [Lowphansirikul et al. (2021)]. WangchanBERTA is a RoBERTa-based architecture model [Liu et al. (2019)] which was trained on a 75.8 GB dataset of Thai and English text. WangchanBERTA is considered state-of-the-art in many

Thai NLP tasks due to the model and dataset used in the training process.

In this study, we also investigated how MLM interacts with our proposed tokenization-based augmentation. The pre-trained WangchanBERTA was used for augmentation by randomly masking tokens in the dataset. For text classification tasks, naive implementation of this kind of augmentation can potentially flip the label. For example, "This soup is very good" can be augmented into "This soup is not good," changing the underlying sentiment. To alleviate this problem, we followed previous works where the WangchanBERTA model was adapted to different domains [Gururangan et al. (2020)] to provide label-dependent mask prediction [Wu et al. (2019b)] for each dataset.

3.4 Word Embedding Dropout

Dropout is a popular regularization technique for deep neural networks [Hinton et al. (2012); Srivastava et al. (2014)] where network unit outputs are randomly masked (dropped) during training. It prevents overfitting and efficiently provides a way of approximately combining exponentially many different neural network architectures. The term "dropout" refers to dropping out units (hidden and visible) in a neural network, along with all its incoming and outgoing connections.

In a model with discrete inputs such as words, dropout can be used in the embedding layer by dropping the same words throughout the sequence, i.e., we drop word types at random rather than word tokens. For example, the sentence "the dog and the cat" might become "- dog and - cat" if "the" is chosen to be dropped [Stafylakis and Tzimiropoulos (2018)].

Just like the MLM augmentation, we also performed experiments on word embedding dropout as another technique that can be used with or without our proposed augmentation method.

Chapter IV

PROPOSED METHOD

The main problem in Thai tokenization is the lack of explicit word boundaries. Many researchers have studied extensively on improving the tokenization performance [Kittinaradorn et al. (2018); Jousimo et al. (2017); Chormai et al. (2019)]. In this study, we proposed an alternative method for augmenting data by using multiple standard tokenization algorithms (e.g., maximum matching and longest matching). In order to improve the performance of text classification tasks, the model would be more robust to segmentation errors in test data by using multiple segmentation methods. The issue of word segmentation on strange words is shown in Fig 4.1. Using different word segmentation techniques can yield vastly different results, some of which might be the correct segmentation. Thus, we hypothesized that augmenting the training data with different tokenization methods might make the model more robust by allowing the model to pick up the correct segmentation and also be more resistant to segmentation noise.

4.1 Noising Word Segmentation

To simulate an errorful tokenization process, we moved the segmented word boundary either forward or backward. The shifting boundary can be moved forward or backward by 1 character token from the original word boundary. The shifting probability is set to λ, with equal probability of shifting forward, p_{sf}, or backward, p_{sb}, ($p_{sf} = p_{sb} = \lambda/2$). The noising process is illustrated in Fig 4.2. This process should provide additional robustness for wrongly tokenized words. Noted that when the word boundary is shifted, it usually causes the affected tokens to become OOVs. This can be considered as token-based dropout (rather than the typical type-based embedding dropout). In order to get the best performance from this noisy tokenization-based augmentation method, the corresponding model should be able to provide meaningful OOV word embeddings, such as using character-based em-

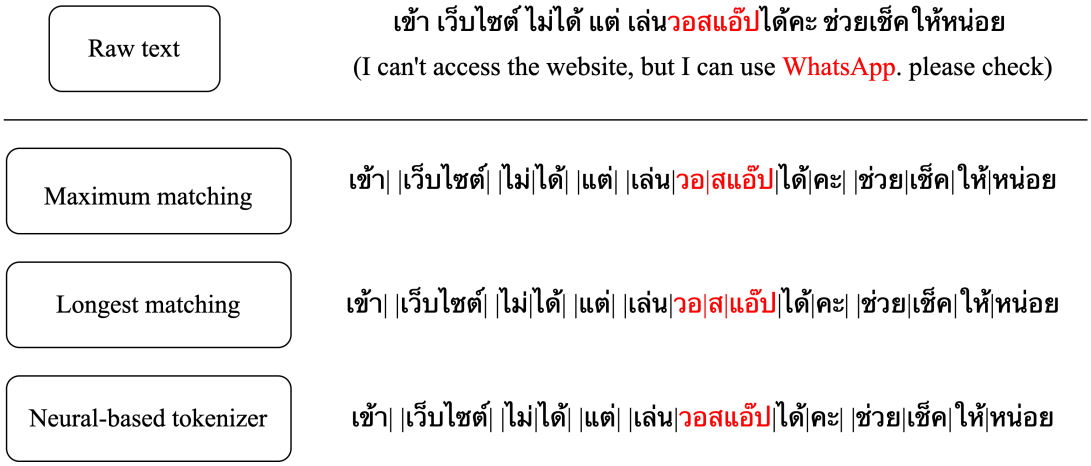

Figure 4.1: An example sentence “เข้า เว็บไซต์ ไม่ได้ แต่ เล่นวอสแอ๊ปได้คะ ช่วยเช็คให้หน่อย” (I can’t access the website, but I can use WhatsApp. please check) is tokenized with different tokenizers (Maximum Matching, Longest Matching, Neural-based tokenizer) which are word level tokenizers. The red highlight word “วอสแอ๊ป” should be single token from raw text. However, maximum matching and longest matching algorithms can not segment to single token.

beddings in conjunction with word embeddings.

4.2 Tokenization-based Augmentation

The algorithm for augmenting the training data using different tokenization is outlined in Algorithm 1. We started with a main tokenizer that is used to segment the data and create the vocabulary. Afterwards, other tokenizers were used to segment the data in order to augment the training set. Additional segmentation noise can be applied to this augmented data. Note that the augmentation tokenizers can also include the main tokenizer if additional noising is desired.

Figure 4.2: An example sentence “ข้าราชการได้รับการหมุนเวียนเป็นระยะ” is tokenized as “ข้าราชการ|ได้รับ|การ|หมุนเวียน|เป็นระยะ”. After adding noise, it becomes “ข้าราชการไ|ด้รับ|การ|หมุนเวียนเ|ป็นระยะ”.

Algorithm 1 Tokenization-based augmentation algorithm

Set: training set $D = \emptyset$, Vocabulary $V = \emptyset$
Input: add_noise, shifting probability λ, main tokenizer and set of augmentation tokenizers
Output: D, V
for each sentence s in dataset **do**
  tok_seq_main = $\text{tkz}_{main}(s)$
  for each token tok in tok_seq_main **do**
    $V = V \cup \{tok\}$ // generate the vocabulary
  end for
  for each tokenizer tkz_{aug} in augmentation tokenizers **do**
    tok_seq = tkz_{aug}(s) // create different token sequences
    if add_noise **then**
      tok_seq = shift_boundary(tok_seq, λ) // noise the segmentation
    end if
    $D = D \cup \{tok\_seq\} \cup \{tok\_seq\_main\}$
  end for
end for

Chapter V

EXPERIMENTAL SETUPS

In this section, we present the setup used in our experiments. All text classification tasks were evaluated on precision, recall, and F1-score.

Datasets

We used three classification benchmark datasets [cstorm125 and lukkiddd (2020)] which are all informal texts with different writing styles. The datasets included Truevoice (customer service chat log), Wisesight1000 (social media), and Wongnai (restaurant review). The detail of the datasets is shown in Table 5.1

Table 5.1: Thai language data used in this study

Dataset	Detail	Objective	Size	Label
Truevoice	Informal, Customer service	Intent	16k	7
Wongnai	Informal, Restaurant review	Sentiment	40k	5
Wisesight1000	Informal, Conversation/opinion	Sentiment	1k	4

Truevoice

Truevoice data is an informal text data which is provided by True Corporation. It contains logs from a call center service. This dataset is often used to benchmark Thai text classification. The goal is to predict the intent of the customer from the seven possible classes. The main tokenizer for this data was Deepcut [Kittinaradorn et al. (2018)] provided in PyThaiNLP [Wannaphong Phatthiyaphaibun (2016)].

Wisesight1000

Wisesight1000 dataset is a subset of the Wisesight dataset [Suriyawongkul et al. (2019)] which is text scraped from social media. The data has four labels (question, positive, negative, and neutral) with 250 samples each. These one thousand samples have manually segmented word boundaries which were considered as the main tokenizer segmentations.

Wongnai

Wongnai data is a collection of user generated restaurant reviews from www.wongnai.com [Wongnai (2018)]. The data contains textual reviews with a rating (a score 1 to 5). The goal is to predict the rating from the review. In consideration of the more standard nature of this text, maximal matching with the default dictionary from PyThaiNLP was used as the main tokenizer. Since the labels for the official test sets were not available, we split the training set into train-val-test with a ratio of 75:10:15.

Tokenizers

The PyThaiNLP library was used to implement all tokenization methods. We considered three different tokenizations: (1) the main tokenization method (MT), (2) maximum matching (MM), and (3) longest matching (LG). Tokenization noise can be applied to any of the three tokenization methods, denoted as "Method_λ", where λ is the noise rate. We set λ to 0.5 in almost experiments except Effect of the amount of randomness experiment.

Text classification model

The machine learning model used with our method can be any text classification model. However, since our augmentation method can potentially create OOV word tokens, our method suits models that have some potential to differentiate between different OOV words. To this end, we chose char-word embedding [Lertpiya et al. (2020)] to create the token representation (v^{tok}) from both the words (w) and the characters in each word ($w = \{c_1, c_2, \ldots, c_{N_w}\}$).

The characters and word tokens are converted into vectors $\{v_1^c, v_2^c, \ldots, v_{N_w}^c\}$ and v^w, via embedding layers respectively. The character embeddings in each word are fed into a Bi-directional Gated Recurrent Unit (BiGRU) which summarizes the character embeddings for the word. The word embeddings are concatenated with the first and the final output of the BiGRU as shown in Equation 5.1.

$$
\begin{aligned}
h^{c,F} &= GRU_{forward}(\{v_1^c, v_2^c, \ldots, v_{N_w}^c\}) \\
h^{c,B} &= GRU_{backward}(\{v_{N_w}^c, v_{N_w-1}^c, \ldots, v_1^c\}) \\
v^{tok} &= v^w \oplus h^{c,B} \oplus h^{c,F}
\end{aligned}
\tag{5.1}
$$

where $h^{c,F}$ is the final output of the forward direction GRU, $h^{c,B}$ is the final output of the backward direction GRU, and $\oplus$ denotes concatenation.

The v^{tok} for each token in the sentence are then fed to another BiGRU and then to a dense layer with a softmax activation function to predict class labels, so called the BiGRU model.

We also tried a simpler shallow model where all of the v^{tok} are averaged into a single vector which is then passed through a dense layer with a softmax activation function.

The BiGRU and shallow models used a word embedding size of 256, character embedding size of 256, character-level GRU hidden layer size of 64. The main BiGRU model used a GRU hidden layer size of 64. The models were trained using the Adam optimizer with a learning rate of 0.001. The vocabulary was selected from the top 80 % of the training tokens in terms of frequency.

In addition, ULMFiT [Howard and Ruder (2018)] and WangchanBERTa were chosen and used in the PyThaiNLP classification benchmark [Lowphansirikul et al. (2021)] to represent strong baselines for our train-test split (train:valid:test=75:10:15). We used the following training scripts for thai2fit [Polpanumas and Phatthiyaphaibun (2021)] [1] and WangchanBERTa [Lowphansirikul et al. (2021)][2]. For the WangchanBERTa baseline, there are many possible baselines. Some version of WangchanBERTa is trained with SentencePiece segmentation which is a sub-word segmentation method. The tokenization in the Sentecne-

[1]https://github.com/cstorm125/thai2fithttps://github.com/cstorm125/thai2fit.

[2]https://github.com/vistec-AI/thai2transformershttps://github.com/vistec-AI/thai2transformers.

Piece model is done using the ULM [Kudo (2018b); Kudo and Richardson (2018a)]. To add noise in the segmentation one can sample from the ULM for multiple segmentations. Another way to use WanghchanBERTa is used the model pre-trained on the word level segmentation (wangchanberta-wiki-newmm). Preliminary experiments showed that the word level model performed better with our augmentation scheme and thus chosen as the WangchanBERTa baseline. It also served as a more comparable baseline to other methods since they were work wll on the word level.

Chapter VI

RESULTS & DISCUSSIONS

This chapter presents our experimental results and ablation studies. We firstly discussed the effect of performing tokenization-based augmentation on different datasets and classification models. Then, we compared the effect of different augmentation strategies and studied how they interact with each other. We also showed that our augmentation method can provide additional robustness against errorful tokenizations and discussed why our method can be helpful. We performed studies on the choice of λ.

6.1 Performance with and without data augmentation

In this part, our proposed data augmentation (with MM_0.5) were compared on the three datasets (Truevoice, Wisesight1000, and Wongnai) using four different classification models (GRU, shallow model, ULMFit, and WangchanBERTa). As shown in Table 6.1, the F1-score for almost models showed the effectiveness of our method. As expected, the smaller models (BiGRU and the shallow model) performed competitively on short texts from Truevoice and Wisesight1000. WangchanBERTA performed better on the longer texts from Wongnai or in low resource settings of Wisesight1000. However, the performance of WangchanBERTA decreased after augmentation for the Wisesight1000 and Wongnai datasets. This is expected, since WangchanBERTA suffers from high OOV rate from the random augmentation, and the model itself lacks the mechanism to handle OOV token embeddings.

We then compared the performance of the models trained with different augmentation strategies using the shallow model. This experiment has 3 main settings: (1) only using tokenization-based augmentation, (2) tokenization-based augmentation and dropout (DO, word dropout rate = 0.5), and (3) tokenization-based augmentation and MLM (MLM). As shown in Table 6.2, the performance can be improved

Table 6.1: F1-scores of models trained with tokenization-based augmentation on Truevoice, Wisesight1000, and Wongnai dataset

Model	MM_0.5	Truevoice	Wisesight1000	Wongnai
BiGRU		73.4	50.2	**27.6**
	✓	**75.3**	**52.2**	24.8
Shallow model		76.8	52.1	26.4
	✓	**79.4**	**55.5**	**27.1**
ULMFiT		79.3	48.1	42.2
	✓	**80.9**	**56.1**	**46.3**
WangchanBERTA		77.9	**76.7**	**57.4**
	✓	**79.6**	75.7	56.0

by combining different augmentation strategies but it depends on the combination. After two augmentations, adding more augmented versions were not seem to improve the performance.

6.2 Effect of the amount of randomness

In this part, we analyzed the effect of the randomness parameter λ and the sensitivity of our method to hyperparameter. In this experiment, we trained models with MT_λ and MM_λ with $\lambda = 0.1, 0.2, \ldots, 1.0$ and showed the performance of the models on the test sets in terms of the F1-score. We chose to use the shallow model for simplicity and found λ around 0.8-1.0 gave the best performance for the Truevoice dataset and around 0.2-0.3 for Wisesight1000 dataset, as shown in Fig 6.1 and 6.2, respectively. The difference between the most suitable range of λ for Truevoice and Wisesight1000 could be explained by the text writing style. Truevoice data contains more typos and misspellings with extra characters which causes the tokenization to be harder compared to Wisesight1000. In general, we recommend 0.5 as a starting point, and can be additionally tuned if desired.

6.3 Robustness to tokenization errors or mismatch

In this experiment, we studied the effectiveness of the proposed augmentation methods for increasing the robustness of the models against unseen and errorful tokenization. To simulate the scenarios where there are multiple possible outcomes

Table 6.2: (Truevoice) The results of the text classification model using different augmentation techniques. **Bold** denotes the best performance in the dataset, while _underline denotes the best performance in the block.

Augmentation						Test		
DO	MLM	MM	LG	MT_0.5	MM_0.5	prec	recall	F1
Truevoice								
						80.2	74.4	76.8
With one tokenization-based augmentation								
		✓				81.4	77.6	79.2
			✓			80.8	76.7	78.4
				✓		80.8	78.5	79.5
					✓	81.5	77.7	79.4
With two tokenization-based augmentations								
		✓	✓			80.9	78.0	79.1
		✓		✓		81.4	**78.9**	79.9
		✓			✓	80.7	77.0	78.6
			✓	✓		81.4	78.8	79.8
			✓		✓	81.3	78.4	79.6
				✓	✓	81.7	78.8	**80.1**
With three tokenization-based augmentations								
		✓	✓	✓		80.4	78.2	79.0
		✓	✓		✓	80.7	77.6	78.9
		✓		✓	✓	79.5	78.1	78.6
			✓	✓	✓	81.1	78.4	79.5
With four tokenization-based augmentations								
		✓	✓	✓	✓	79.8	77.9	78.6
With Dropout and a tokenization-based augmentation								
✓						79.6	75.5	77.1
✓		✓				81.4	77.1	78.9
✓			✓			81.5	76.5	78.6
✓				✓		80.9	77.7	79.1
✓					✓	81.0	77.1	78.8
With MLM and a tokenization-based augmentation								
	✓					79.8	74.7	76.9
	✓	✓				81.2	77.5	79.1
	✓		✓			**82.0**	77.3	79.3
	✓			✓		81.1	78.8	79.7
	✓				✓	81.3	77.3	79.0

Table 6.3: (Wisesight1000) The results of the text classification model using different augmentation techniques. **Bold** denotes the best performance in the dataset, while _underline denotes the best performance in the block.

Augmentation						Test		
DO	MLM	MM	LG	MT_0.5	MM_0.5	prec	recall	F1
Wisesight1000								
						52.6	51.6	52.1
With one tokenization-based augmentation								
		✓				49.7	50.7	50.2
			✓			49.8	50.7	50.2
				✓		50.6	51.5	51.1
					✓	**57.0**	54.0	**55.5**
With two tokenization-based augmentations								
		✓	✓			54.6	**55.4**	54.5
		✓		✓		51.9	52.2	52.0
		✓			✓	51.6	51.6	51.5
			✓	✓		52.6	52.9	52.6
			✓		✓	52.4	53.4	52.7
				✓	✓	50.4	50.3	50.2
With three tokenization-based augmentations								
		✓	✓	✓		51.2	51.5	51.3
		✓	✓		✓	50.2	52.0	50.5
		✓		✓	✓	51.7	51.6	51.3
			✓	✓	✓	54.0	54.2	53.9
With four tokenization-based augmentations								
		✓	✓	✓	✓	51.5	52.2	51.5
With Dropout and a tokenization-based augmentation								
✓						53.0	52.9	52.8
✓		✓				52.2	52.7	52.4
✓			✓			51.0	52.1	51.3
✓				✓		52.2	52.9	52.2
✓					✓	51.2	51.5	51.2
With MLM and a tokenization-based augmentation								
	✓					52.3	50.9	51.2
	✓	✓				51.8	52.1	51.9
	✓		✓			50.6	50.9	50.7
	✓			✓		51.6	52.2	51.6
	✓				✓	50.3	50.8	50.4

Table 6.4: (Wongnai) Selected results of the text classification model using different augmentation techniques on the Wongnai dataset. **Bold** denotes the best performance in the dataset, while _underline denotes the best performance in the block. All underline results are statistically significant compared to their respective non-tokenization-based-augmentation baselines using the McNemar's test ($p > 0.05$).

Augmentation						Test		
DO	MLM	MM	LG	MT_0.5	MM_0.5	prec	recall	f1
wongnai								
baseline								
		✓				24.4	24.8	24.6
With one tokenization-based augmentation								
		✓	✓			24.9	**25.7**	**25.3**
		✓			✓	24.1	23.1	23.6
With two tokenization-based augmentation								
		✓	✓		✓	24.7	24.2	24.4
With Dropout and a tokenization-based augmentation								
✓		✓				24.7	24.7	24.7
✓		✓	✓			24.7	**25.7**	25.2
✓		✓			✓	**25.1**	24.5	24.8
With MLM and a tokenization-based augmentation								
	✓	✓				24.5	25	24.7
	✓	✓	✓			25.0	25.4	25.2

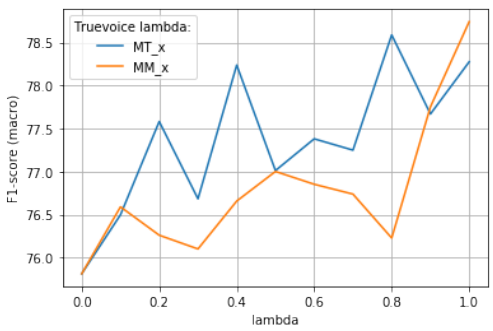

Figure 6.1: F1-score vs λ in the Truevoice dataset

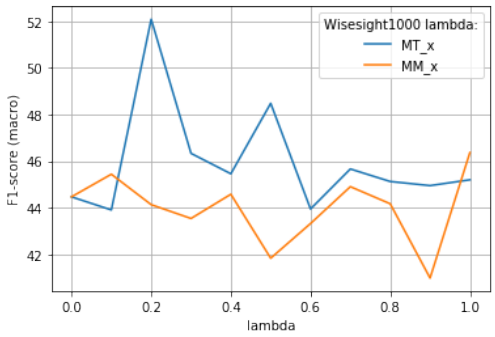

Figure 6.2: F1-score vs λ in the Wisesight1000 dataset

of tokenizing difficult-to-tokenize sentences, we used tokenizers of different nature (e.g. deep learning-based tokenizer, maximum matching, and longest matching) on the training and test data, which tokenized differently. As demonstrated in Table 6.6 and 6.7, the selected results were obtained from the test set that used longest matching with different tokenization noise ($\lambda = 0.1, 0.4,$ and 0.7). We found the models

trained with MT_0.5 and MM_0.5 augmentation are robust against randomized token sequences. Upon further qualitative error analysis, we have found that the models trained with tokenization-based augmentation performed comparatively better on sentences that were especially hard to tokenize. For example, “รบกวนสอบถามตัวเพกเกจหน่อยครับ” was tokenized as “รบกวน|สอบถาม|ตัว|เพก|เกจ|หน่อ|ย|ครับ”. The “เพก”, “เกจ”, “หน่อ”, and “ย” were bad tokens. They should be tokenized as “เพก เกจ” and “หน่อย”).

Table 6.5: Raw text and tokenized text from main tokenizer that model without augmentation predicted wrong. However, model trained with augmentation can predict the right label. The red is a token from main tokenizer that the different from human expert.

raw text	main tokenizer	MM	MM_0.5
Truevoice			
รบกวนสอบถามตัว เพกเกจหน่อยครับ	รบกวน|สอบถาม|ตัว| เพก|เกจหน่อ|ย|ครับ	F	T
เมื่อวานซิมมันหาย จะไปรับซิมใหม่ แถวบางนา-ตราดนี่ มีร้านอยู่ที่ไหนบ้างครับ	เมื่อ|วาน|ซิม|มัน|หาย |จะ|ไป|รับ|ซิม|ใหม่| แถว|บาง|นา|-|ตราด|นี่ |มี|ร้าน|อยู่|ที่|ไหน|บ้าง|ครับ	T	T
Wisesight1000			
กำลังสีไปๆนำบ่ ลางานเคิงมื้อมา	กำลัง|สี|ไป|ๆ|นำ|บ่| ลา|งาน|เคิง|มื้อ|มา	T	T
ใช้มาสคาร่าเดอะเฟสชอป แล้วจะสวยแบบนี้จริงไหมคะ	ใช้|มาสคาร่า|เดอะ|เฟส|ชอป| แล้ว|จะ|สวย|แบบ|นี้|จริง|ไหม|คะ	T	T
ยึดแค่กระเป๋าพอแล้วมั้ง .... เหมือนมีอะไรแอบแฟง 555	ยึด|แค่|กระเป๋า|พอ|แล้ว|มั้ง| .... |เหมือน|มี|อะไร|แอบ|แฟง|555	T	T

6.4 Conclusions

We proposed an alternative data augmentation method by using multiple tokenizations to improve text classification models for Thai language. With this data augmentation method, text classification models are more robust against poor tokenization and can be combined with other text augmentation methods such as dropout and MLM. Our augmentation method works especially well when the model can prop-

Table 6.6: (Truevoice) The performance of different augmentation strategies on mismatch/errorful tokenization settings.

Training set						Test set		
DO	MLM	MM	LG	MT_0.5	MM_0.5	LG_0.1	LG_0.4	LG_0.7
Truevoice								
With one tokenization-based augmentation								
						60.4	50.9	42.0
		✓				67.3	58.0	51.9
			✓			70.4	61.3	55.0
				✓		71.2	69.5	64.9
					✓	74.3	70.7	65.9
With Dropout and a tokenization-based augmentation								
✓						70.5	57.4	50.9
✓		✓				74.5	67.1	62.8
✓			✓			**76.6**	70.6	63.8
✓				✓		74.2	65.6	61.1
✓					✓	73.7	73.4	68.7
With MLM and a tokenization-based augmentation								
	✓					69.3	60.5	56.1
	✓	✓				71.2	60.8	55.1
	✓		✓			71.8	63.1	57.1
	✓			✓		75.7	74.2	70.8
	✓				✓	76.5	**74.6**	71.0

erly handle OOVs. To improve the performance on the BERT-based model, further improvement can be explored to improve the vocabulary coverage by adding new embeddings in the pre-trained model.

Table 6.7: (Wiseight1000) The performance of different augmentation strategies on mismatch/errorful tokenization settings.

Training set						Test set		
DO	MLM	MM	LG	MT_0.5	MM_0.5	LG_0.1	LG_0.4	LG_0.7
Wisesight1000								
With one tokenization-based augmentation								
						48.2	38.1	32.7
		✓				51.5	**47.3**	38.9
			✓			50.3	43.6	37.3
				✓		45.2	41.8	40.3
					✓	47.9	42.0	35.7
With Dropout and a tokenization-based augmentation								
✓						46.2	40.2	36.8
✓		✓				**52.5**	45.1	40.2
✓			✓			51.7	42.9	35.5
✓				✓		45.2	43.0	40.6
✓					✓	50.1	41.9	41.7
With MLM and a tokenization-based augmentation								
	✓					47.7	37.9	31.4
	✓	✓				50.0	46.3	41.4
	✓		✓			51.6	43.2	33.9
	✓			✓		46.6	43.3	40.4
	✓				✓	46.7	41.7	**42.1**

Table 6.8: (Truevoice) The performance of the text classification model (swallow model) when we changed tokenizer of test set

Augmentation						Test			
DO	MLM	MM	LG	MT_0.5	MM_0.5	MM	LG	MT_0.5	MM_0.5
Truevoice									
With one tokenization-based augmentation									
						75.5	75.3	63.5	61.2
		✓				79	78	64.3	65.3
			✓			78	77.8	62	64.4
				✓		77.9	73.9	71.7	71.6
					✓	80.1	79.3	71.5	71.4
With Dropout and a tokenization-based augmentation									
✓						76.5	76.2	62.7	60.9
✓		✓				78.9	77.9	64.6	65.1
✓			✓			78.2	77.7	62.9	64.8
✓				✓		78.5	78.1	71.9	71.5
✓					✓	79.3	78.5	58.9	71.8
With MLM and a tokenization-based augmentation									
	✓					75.1	75.5	58.9	57.1
	✓	✓				78.4	77.8	62.7	63.9
	✓		✓			78.5	78.2	62.9	65.6
	✓			✓		77.2	77.3	71.7	72.3
	✓				✓	79.2	78.4	69.7	73.4

Table 6.9: (Wisesight1000) The performance of the text classification model (swallow model) when we changed tokenizer of test set

Augmentation						Test			
DO	MLM	MM	LG	MT_0.5	MM_0.5	MM	LG	MT_0.5	MM_0.5
Wisesight1000									
With one tokenization-based augmentation									
						48.2	47.4	41.7	36
		✓				51.4	49.9	42.4	42.9
			✓			52.1	51.8	41.3	41.7
				✓		48.3	48.6	44.7	46.6
					✓	49.9	46.3	44	42.7
With Dropout and a tokenization-based augmentation									
✓						47.8	46.3	47.1	40
✓		✓				52.5	49.8	45.1	42.3
✓			✓			51.9	51.1	43.2	41.9
✓				✓		51.9	49.1	43.2	48.2
✓					✓	51.7	48.2	43.8	47.7
With MLM and a tokenization-based augmentation									
	✓					44.9	45.5	39.8	37.6
	✓	✓				51.4	47.7	43.4	42.3
	✓		✓			50.7	50.1	40.8	43.3
	✓			✓		49.9	49.5	44.8	47.5
	✓				✓	50.6	48.7	41.8	47.8

REFERENCES

Chen, Y. 2015. Convolutional neural network for sentence classification. Master's thesis, University of Waterloo.

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

cstorm125 and lukkiddd 2020. Pythainlp/classification-benchmarks: v0.1-alpha [Online]. Available from: https://doi.org/10.5281/zenodo.3852912 [2020,May].

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018):

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. 2008. Liblinear: A library for large linear classification. the Journal of machine Learning research 9 (2008): 1871–1874.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. 2016. Deep learning, volume 1. MIT press Cambridge.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8342–8360. Online: Association for Computational Linguistics.

Hanlon, C. 2018. Tokenization of japanese text: Using a morphological transducer. (2018):

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, volume 1, pp. 125–128. :

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 Track), pp. 68–80. Boston, MA: Association for Machine Translation in the Americas.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012):

Hiraoka, T., Shindo, H., and Matsumoto, Y. 2019. Stochastic tokenization with a language model for neural text classification. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1620–1629. :

Howard, J. and Ruder, S. 2018. Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (2018):

Huang, C.-R., Šimon, P., Hsieh, S.-K., and Prévot, L. 2007. Rethinking chinese word segmentation: tokenization, character classification, or wordbreak identification. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, pp. 69–72. :

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, 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), pp. 1681–1691. :

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. 2017. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics:

Volume 2, Short Papers, pp. 427–431. Valencia, Spain: Association for Computational Linguistics.

Jousimo, J., Laokulrat, N., Carr, B., Thongthanomkul, E., and Satayamas, V. 2017. Thai word segmentation with bi-directional RNN.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016):

Jurafsky, D. and Martin, J. H. 2009. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International. ISBN 9780135041963. Available from: https://www.worldcat.org/oclc/315913020 .

Kittinaradorn, R., Chaovavanich, K., Achakulvisut, T., and Kaewkasi, C. 2018. A Thai word tokenization library using deep neural network.

Kudo, T. 2018a. Subword regularization: Improving neural network translation models with multiple subword candidates. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 66–75. Melbourne, Australia: Association for Computational Linguistics.

Kudo, T. 2018b. Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959 (2018):

Kudo, T. and Richardson, J. 2018a. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 66–71. Brussels, Belgium: Association for Computational Linguistics.

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

Le, Q. V., Luong, M.-T., Sutskever, I., Vinyals, O., and Zaremba, W. 2018. Neural machine translation systems with rare word processing.

Lertpiya, A., Chalothorn, T., and Chuangsuwanich, E. 2020. Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention. IEEE Access 8 (2020): 133403–133419.

Liu, C., Wang, J., and Lei, K. 2016a. Detecting spam comments posted in microblogs using the self-extensible spam dictionary. In 2016 IEEE International Conference on Communications (ICC), pp. 1–7. :

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 Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 8384–8391. :

Liu, P., Qiu, X., and Huang, X. 2016b. Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101 (2016):

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019):

Lowphansirikul, L., Polpanumas, C., Jantrakulchai, N., and Nutanong, S. 2021. Wangchanberta: Pretraining transformer-based thai language models. arXiv preprint arXiv:2101.09635 (2021):

Malhotra, P., Vig, L., Shroff, G., Agarwal, P., et al. 2015. Long short term memory networks for anomaly detection in time series. In Proceedings, volume 89, pp. 89–94. :

Medsker, L. R. and Jain, L. 2001. Recurrent neural networks. Design and Applications 5 (2001): 64–67.

Meknavin, S., Charoenpornsawat, P., and Kijsirikul, B. 1997. Feature-based thai word segmentation. In Proceedings of Natural Language Processing Pacific Rim Symposium, volume 97, pp. 41–46. :

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. Advances in neural information processing systems 26 (2013):

Park, D. and Ahn, C. W. 2019. Self-supervised contextual data augmentation for natural language processing. Symmetry 11.11 (2019): 1393.

Polpanumas, C. and Phatthiyaphaibun, W. 2021. thai2fit: Thai language implementation of ulmfit [Online]. Available from: https://doi.org/10.5281/zenodo.4429691 [2021,January].

Remus, S., Hintz, G., Biemann, C., Meyer, C. M., Benikova, D., Eckle-Kohler, J., Mieskes, M., and Arnold, T. 2016. Empirist: Aiphes-robust tokenization and pos-tagging for different genres. In Proceedings of the 10th Web as Corpus Workshop, pp. 106–114. :

Sennrich, R., Haddow, B., and Birch, A. 2015. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909 (2015):

Shi, L., Liu, D., Liu, G., and Meng, K. 2019. Aug-bert: An efficient data augmentation algorithm for text classification. In International Conference in Communications, Signal Processing, and Systems, pp. 2191–2198. :

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15.1 (2014): 1929–1958.

Stafylakis, T. and Tzimiropoulos, G. 2018. Deep word embeddings for visual speech recognition. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4974–4978. :

Suriyawongkul, A., Chuangsuwanich, E., Chormai, P., and Polpanumas, C. 2019. Pythainlp/wisesight-sentiment: First release [Online]. Available from: https://doi.org/10.5281/zenodo.3457447 [2019,September].

Van Heerden, C., Karakos, D., Narasimhan, K., Davel, M., and Schwartz, R. 2017. Constructing sub-word units for spoken term detection. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5780–5784. :

Wang, H., Zhang, P., and Xing, E. P. 2020. Word shape matters: Robust machine translation with visual embedding. arXiv preprint arXiv:2010.09997 (2020):

Wannaphong Phatthiyaphaibun, C. P. A. S. L. L. P. C., Korakot Chaovavanich. 2016. PyThaiNLP: Thai Natural Language Processing in Python [Online]. Available from: http://doi.org/10.5281/zenodo.3519354 [2016,June].

Wongnai 2018. Wongnai/wongnai-corpus: Collection of wongnai's datasets [Online]. Available from: https://github.com/wongnai/wongnai-corpus [2018,].

Wu, X., Lv, S., Zang, L., Han, J., and Hu, S. 2019a. Conditional BERT contextual augmentation. In International Conference on Computational Science, pp. 84–95. :

Wu, X., Zhang, T., Zang, L., Han, J., and Hu, S. 2019b. Mask and infill: Applying masked language model for sentiment transfer. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 5271–5277. : International Joint Conferences on Artificial Intelligence Organization.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016):

Zhang, X., Zhao, J., and LeCun, Y. 2015. Character-level convolutional networks for text classification. Advances in neural information processing systems 28 (2015): 649–657.

Zoph, B., Vaswani, A., May, J., and Knight, K. 2016. Simple, fast noise-contrastive estimation for large rnn vocabularies. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1217–1222. :