Classification of Abusive Thai Messages in Social Networks Using Deep Learning

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Mr. Ruangsung Wanasukapunt

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science and Information Technology
Department of Mathematics and Computer Science
FACULTY OF SCIENCE
Chulalongkorn University
Academic Year 2021

Copyright of Chulalongkorn University

การจำแนกข้อความไทยที่ใช้ไม่เหมาะสมในเครือข่ายสังคมโดยใช้การเรียนรู้เชิงลึก

นายเรืองสรรค์ วนาสุขพันธ์

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการ
คอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Thesis Title	Classification of Abusive Thai Messages in Social Networks Using Deep Learning
By	Mr. Ruangsung Wanasukapunt
Field of Study	Computer Science and Information Technology
Thesis Advisor	Associate Professor SUPHAKANT PHIMOLTARES, Ph.D.

Accepted by the FACULTY OF SCIENCE, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

-- Dean of the FACULTY OF SCIENCE

(Professor POLKIT SANGVANICH, Ph.D.)

THESIS COMMITTEE

-- Chairman

(Professor CHIDCHANOK LURSINSAP, Ph.D.)

-- Thesis Advisor

(Associate Professor SUPHAKANT PHIMOLTARES, Ph.D.)

-- Examiner

(Prem Junsawang, Ph.D.)

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

เรืองสรรค์ วนาสุขพันธ์ : การจำแนกข้อความไทยที่ใช้ไม่เหมาะสมในเครือข่ายสังคมโดยใช้การเรียนรู้เชิงลึก. (Classification of Abusive Thai Messages in Social Networks Using Deep Learning) อ.ที่ปรึกษาหลัก : รศ. ดร.ศุภกานต์ พิมลธเรศ

สื่อสังคมมีการปรับปรุงแหล่งข่าวแบบดั้งเดิมโดยอนุญาตให้มีการเข้าถึงข่าวสารเพิ่มขึ้น อย่างไรก็ตามการยอมไม่ให้เปิดเผยชื่อในสื่อสังคมก่อให้เกิดข้อความที่ใช้ไม่เหมาะสมและมีเจตนาร้ายโดยปราศจากการตรวจหาหรือผลที่ตามมาจากบุคคลด้วยความตั้งใจมุ่งร้าย งานวิจัยนี้พัฒนาตัวแบบการจำแนกแบบทวินามและอเนกนามสำหรับจำแนกข้อความบนสื่อสังคมไทยออกเป็นห้าประเภทสำหรับการตรวจหาเนื้อหาที่ไม่เหมาะสมในสื่อสังคม อันได้แก่ข้อความหยาบคาย ข้อความอุปมาอุปไมย ข้อความลามก ข้อความก้าวร้าว และข้อความที่ใช้ได้เหมาะสม การทดลองได้แสดงให้เห็นว่าดิสทิลเบิร์ทได้ให้คะแนนเอฟวันสูงสุดที่ 0.8510 สำหรับตัวแบบทวินามและ 0.9067 สำหรับตัวแบบอเนกนาม แอลเอสทีเอ็มแบบสองทิศทางได้ให้ผลดีที่สุดเป็นอันดับสองด้วยคะแนนเอฟวัน 0.8403 และ 0.8969 สำหรับตัวแบบทวินามและอเนกนามตามลำดับ ตัวแบบการเรียนรู้เชิงลึกทั้งสองได้ผลที่ดีกว่าตัวแบบการเรียนรู้ของเครื่องแบบดั้งเดิมที่มีคะแนนเอฟวันสูงสุดอยู่ที่ 0.7452 และ 0.8090 สำหรับตัวแบบทวินามและอเนกนามตามลำดับ สถาปัตยกรรมการเรียนรู้เชิงลึกได้ยอมให้การแทนเชิงบริบทของกลุ่มคำดีขึ้น โดยดิสทิลเบิร์ทได้ทำให้การสร้างตัวแบบของความเกี่ยวข้องกันระหว่างกลุ่มคำในช่วงที่ยาวดีขึ้น

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อนิสิต ....
ปีการศึกษา	2564	ลายมือชื่อ อ.ที่ปรึกษาหลัก ....

6172627123 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORD: Abusive language detection / Thai natural language processing / Large scale social networks

Ruangsung Wanasukapunt : Classification of Abusive Thai Messages in Social Networks Using Deep Learning. Advisor: Assoc. Prof. SUPHAKANT PHIMOLTARES, Ph.D.

Social media has improved on traditional news sources by allowing increased access to information. However, the anonymity social media provides can lead to abusive and hateful speech without detection or repercussion from individuals with malicious intentions. This research develops a binomial and a multinomial classification model for classifying Thai social media text for five categories of abusive content detection in social media that include Rude, Figurative, Dirty, Offensive and Non-Abusive. The experiments demonstrated that DistilBERT achieved the highest F1 score with 0.8510 for the binomial model and 0.9067 for the multinomial model. BiLSTM performed second best with an F1 score of 0.8403 and 0.8969 for the binomial and multinomial models, respectively. Both deep learning models outperformed the traditional machine learning classifiers' highest F1 score of 0.7452 and 0.8090 for the binomial and multinomial models, respectively. The deep learning architectures allow for better contextual representations of the words with the DistilBERT, enabling better modeling of long-range dependencies between words.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Field of Study:	Computer Science and Information Technology	Student's Signature ....
Academic Year:	2021	Advisor's Signature ....

ACKNOWLEDGEMENTS

I would like to express my thanks to several individuals for their invaluable assistance and support guiding me to the successful completion of this thesis.

First, I am sincerely indebted to my research advisor, Associate Professor Dr. Suphakant Phimoltares, for his constant guidance and support thoughout this process. His deep knowledge and insights lead me since the initial step in developing the knowledge to properly research and approach the subject matter. The conversations and meetings were crucial in the successful design and production of the thesis.

Also, I would like to thank program chair, Professor Dr. Chidchanok Lursinsap, whose course on artificial neural networks provided inspiration in addressing the topic of the thesis. I am also grateful for his role as an examiner along with external examiner, Dr. Prem Junsawang, both of whom provided sage and thoughtful recommendations and comments for the thesis.

Last but not least, my parents deserve eternal gratitude for their encouragement and support during the production of this study.

Ruangsung Wanasukapunt

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

TABLE OF CONTENTS

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Chapter 1. Introduction

Social media has improved on traditional news sources by allowing increased access to information. Breaking news can be published instantly and spread quickly. Niche content can be customized for unique interests. However, the anonymity social media provides can lead to abusive and hateful speech without detection or repercussion from individuals with malicious intentions. In [1] the percentage of anonymous users posting random tweets on Twitter was 40% but for hate speech the number ranged from 46%-55%. In order to control this type of behavior, web platforms are developing and implementing algorithms to detect such content. Governments are also mandating controls as well. In a U.S. Congressional hearing in March 2021, CEO of Facebook Mark Zuckerberg suggested to Congress that it implements a law to require that platforms have systems that can detect illegal content and remove it [2]. The number of users in Thailand on Facebook numbered 56.4 million in 2021 [3] and thus could be affected as well. Also, the Royal Thai Police has stated that cyberbullying is a crime, citing a case involving the daughter of the Prime Minister Prayuth Chan-ocha as a victim.

Facebook currently has an AI system dedicated to detecting abusive content, which includes categories such as Hate Speech and Bullying and Harassment. This system detected hate speech before any human at a rate that rose from 24% in late-2017 to reach 97% in the fourth quarter of 2020 [4].

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

In light of these trends, abusive content detection should continue gaining in importance. Improvements in detection are enabled with state-of-the-art machine learning. While past research has focused on detecting Thai language abusive speech on social media using traditional machine learning techniques, this thesis improves upon this by using modern deep learning. This thesis will use variants of the Transformers [5] and Recurrent Neural Network (RNN) architectures.

1.1. Statement of the problem

Several problems are explored in this research:

1. How to improve Thai abusive speech detection using deep learning rather than traditional machine learning techniques such as SVM and random forest.

2. How to distinguish between different classes of abusive speech, such as rude, figurative, offensive, and dirty.
3. How to design the proposed method to be compatible with a text belonging to multiple classes.

1.2. Objective

In order to improve Thai abusive speech detection, there are three goals:

1. To develop a deep learning binomial model that can detect Thai abusive speech with high accuracy
2. To evaluate the deep learning models as compared to previous research that used traditional (shallow) machine learning models
3. To develop a multinomial model that can classify Thai abusive speech into five categories

1.3. Scope of thesis and constraints

There are two issues in this research as follows:

1. The abusive language dataset was collected from Facebook manually and labeled by linguistic major students from Chulalongkorn University, consisting of 6,770 texts.
2. The traditional machine learning models tested are the Support Vector Machine (SVM), Multinomial Naïve Bayes (MNB), Bernoulli Naïve Bayes (BNB), k-Nearest Neighbor (kNN), Random Forest (RF), and Decision Tree (DT) classifiers. The deep learning models tested are Bidirectional Long-Short Term Model (BiLSTM) and DistilBERT [6].

1.4. Expected outcome

This research aims to develop a binomial and a multinomial classification model for classifying Thai social media text. The binomial model will have two classes: Abusive and Non-Abusive. The multinomial model will have five classes: Rude, Figurative, Dirty, Offensive, and Non-Abusive.

Chapter 2. Related Works

Abusive text detection in Asian languages has historically used traditional machine learning or classic deep learning techniques such as RNN. However, with the advent of the Transformers architecture in 2017, more recent research has focused on the BERT variant of this architecture.

2.1. Thai Abusive Facebook Text using Traditional Machine Learning

Tuarob and Mitrpanont [7] developed a binomial model to detect Thai language abusive content within Facebook comments. The texts were labeled as Figurative, Dirty, Offensive, Rude, and Non-Abusive, though no multinomial model was created. The research used nine traditional machine learning classifiers including Discriminative Multinomial Naïve Bayes (DMNB) classifier, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Binomial Naïve Bayes (BNB) classifier, Support Vector Machine (SVM), Random Forest (RF), Maximum Entropy, K-Nearest Neighbor (kNN) classifier, C4.5 Decision Tree, and Decision Table/Naive Bayes Hybrid (DTNB). DMNB performed the best, achieving an 86% F-measure, 88.74% precision, and 83.53% recall. Notably, no deep learning techniques were used.

2.2. Hate, Offensive, and Clean Speech Detection in Vietnamese

The Vietnamese Hate Speech Detection campaign is a dataset offered by the Vietnamese Language and Speech Processing 2019 workshop for a conference challenge.[1] It is labeled with the Hate, Offensive, and Clean classes. Pham et al. adapted the Robustly Optimized BERT Pretraining Approach (RoBERTa) by re-training and fine-tuning the PhoBERT [8] model for the classification task. It achieved an F1 score of 0.7221, a new state-of-the-art result [9].

2.3. Nepali YouTube

Singh et al. created a dataset for targeted aspect-based sentiment analysis using comments from Nepali YouTube videos [10]. The target entities were tagged in the comments and include *Organization, Person*, *Location*, and *Miscellaneous*. These entities were annotated as being in the categories *Violence*, *General*, *Profanity*, *Sarcasm Feedback*, and *Out-of-scope*. The first task was to classify each entity as

[1] https://vlsp.org.vn/vlsp2019/eval/hsd

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

one of the first four categories. For this task, BERT performed best with an F1 of 57.98 compared to BiLSTM with an F1 of 57.07.

The second task was to detect the sentiment polarity of each aspect category with a Boolean output. The models tested were SVM, CNN, BiLSTM, and BERT. BiLSTM performed best with an F1 of 0.816 compared to BERT, CNN, and SVM, with F1 scores of 0.799, 0.811, and 0.712, respectively.

While BiLSTM outperformed BERT in the Nepali study, contrary to this study's result, this may have been due to the simpler nature of the task of binomial classification for easier to detect categories such as Profanity. The study noted more false positives in ambiguous categories such as *General*. Hence BERT's better contextual representations may only provide a meaningfully significant advantage when dealing with difficult to detect categories such as *Figurative*.

2.4. Indonesian Twitter

Hendrawan, Adiwijaya, and Al Faraby used the Twitter dataset from [11] and added 5,227 new tweets with the majority having some Indonesian language [12]. The labels divided hate speech into categories such as individual, group, religion, race, gender, other, weak, moderate, and strong. An individual tweet may have multiple labels. Accuracy is calculated as:

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

$$Accuracy = \left(\frac{1}{N}\right)\sum_{i=1}^{N}\left(\left|\frac{\hat{y}_i \cap y_i}{\hat{y}_i \cup y_i}\right|\right) \times 100\% \tag{1}$$

where N is the number of tweets, $\hat{y}$ is a prediction label set, and y is the actual label set. While [11] only used traditional machine learning, [12] included some deep learning models and tested RFDT, BiLSTM and BiLSTM with a pre-trained BERT model. The RFDT was found to perform the best with an accuracy level of 76.12% compared to BiLSTM at 68.49% and BiLSTM+BERT at 64.81%. No precision, recall or F1 statistics were provided.

2.5. Facebook

Facebook does not publicly disclose its abusive content detection algorithms. However, its research can reveal clues as to which models it prefers. Facebook AI compared the NN, Extreme Learning Machine (ELM), and LSTM models in predicting the total interaction with a post and the models had R^2 scores of 0.139, 0.053, and 0.174, respectively, with LSTM scoring the highest. The count of interactions with a post includes the number of likes, comments, and shares of a post. LSTM also outperformed other models in creating an efficient neural language model in regards to perplexity [13].

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Chapter 3. Theoretical Background

3.1. Thai abusive speech detection

Sentiment analysis is one of the major applications in natural language processing using machine learning. It is the analysis of text to systematically identify and extract emotional and subjective information from the source material.

Thai language social media sentiment analysis presents several challenges. First, the Thai language does not use any punctuation to denote separation of sentences. Second, the number of publicly available datasets is small as compared to English. Third, social media uses new slang that may not appear in standard texts such as Wikipedia. Fourth, social media may use poor grammar and contain misspellings.

Abusive speech detection, a specialized task within social media sentiment analysis, presents even more unique challenges. Thai language social media sentiment analysis has typically focused on straightforward, mutually exclusive classes such as positive, negative, and neutral ([14], [15], [16]) or variations of negative versus non-negative [17]. However, abusive speech detection research contains multiple classes which are not mutually exclusive and thus the same text may have more than a single label. Second, the use of one class in particular, the Figurative class, includes words that may have different meanings depending on the context, or the surrounding words.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

The scope of this thesis is limited to multiple label abusive speech detection as it is a more difficult task than simple sentiment polarity and therefore a better test when comparing the effectiveness of various models. Only [7] has published such research thus far for Thai language social media and those methods used only traditional machine learning models. The models used for testing will be described next.

3.2. Traditional machine learning models

Traditional machine learning models encompass a varied set of methodologies. They can be differentiated from deep learning in that the output from one layer is not used as input to another. Hence, they may also be referred to as shallow learning models.

This thesis aims to propose the use of deep learning-based models along with their comparison to shallow learning models similar to the ones used in [7]. The traditional machine learning models to be tested are SVM, MNB, BNB, kNN, RF, and DT.

3.2.1. One-hot encoding

In machine learning, the first step is to convert the words into vectors – in particular, one-hot vectors. In Figure 1, two similar words are encoded using one-hot vectors, where each vector has a single value of one representing the unique index and the rest of the cells contain 0's. The size of the word vector dimension is equal to the number of unique words in the corpus. While the two words in the figure, “dirty” and “unclean”, are similar in meaning, when doing a dot product with the word vectors to check for similarity, the score is 0 since the one-hot vectors are orthogonal to each other.

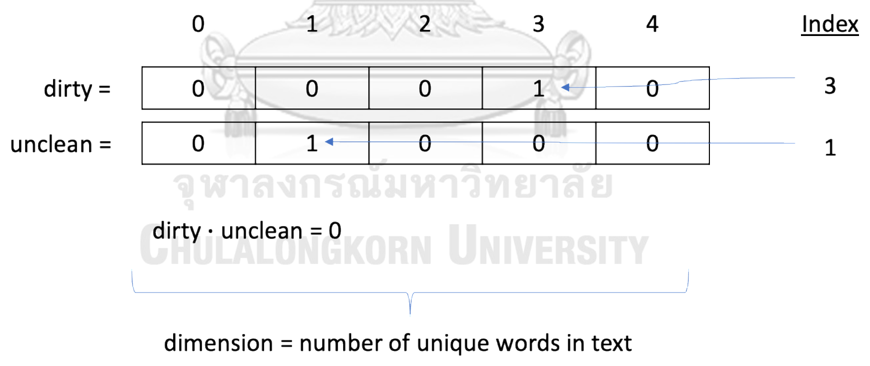


Figure 1: One-hot vectors

To encode a sentence, or sequence of words, several methods exist to convert the set of word vectors into a sentence vector. The sequence may also be several sentences which together combine to make a document. These methods are known as vectorizers.

A count vectorizer creates a sentence vector equal in dimension to the number of words in the corpus, and it will count the number of times a particular

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

word appears in a sentence and add it to that vector at the word's respective index. For example, the word "dirty" in Figure 1 has an index of three. Thus, if the word appeared in the sentence twice, there would be a count of two at index three in the sentence vector. The binary vectorizer simply replaces the word counts with a Boolean.

A term frequency inverse document frequency (tf-idf) vectorizer may also be used. The tf-idf formula is given below:

$$tfidf(t,d) = tf(t,d) \times idf(t) \tag{2}$$

$$idf(t) = \log\left(\frac{1+n}{1+df(t)}\right) + 1 \tag{3}$$

where tf is the raw count or occurrences of term (word) t, d is the document (set of words), n is the total number of documents, and $df(t)$ is the number of documents that contain the term t. The tf-idf vectors are then divided by the Euclidean norm. The effect is to give less weight to terms that are frequently used in many texts since their high frequencies implies less importance. However, when [7] tested the various methods, the difference in performance between different vectorizers was minimal. Sentence / document vectors may also be referred to as feature vectors.

3.2.2. Support Vector Machine

An SVM creates an optimal separating hyperplane or set of hyperplanes in high-dimensional space to separate the data points in different classes from one another. Figure 2 shows a simple linear binomial classifier. The green circles represent data points in the "Figurative" class and the orange circles represent data points in the "Non-Figurative" class. In this simple case, the data points from different classes are clearly separated. However, there exist several lines that can be chosen to separate the two classes. The question is which line should be used. The equation for the separating hyperplane used in SVM is:

$$h_{w,b}(x) = g(w^T x + b) \tag{4}$$

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

where h is the separating hyperplane, w is the weight vector, x is the input vector, b is the bias, and g is a function where $g = 1$ if $w^T x + b \geq 0$ and $g = -1$ otherwise.

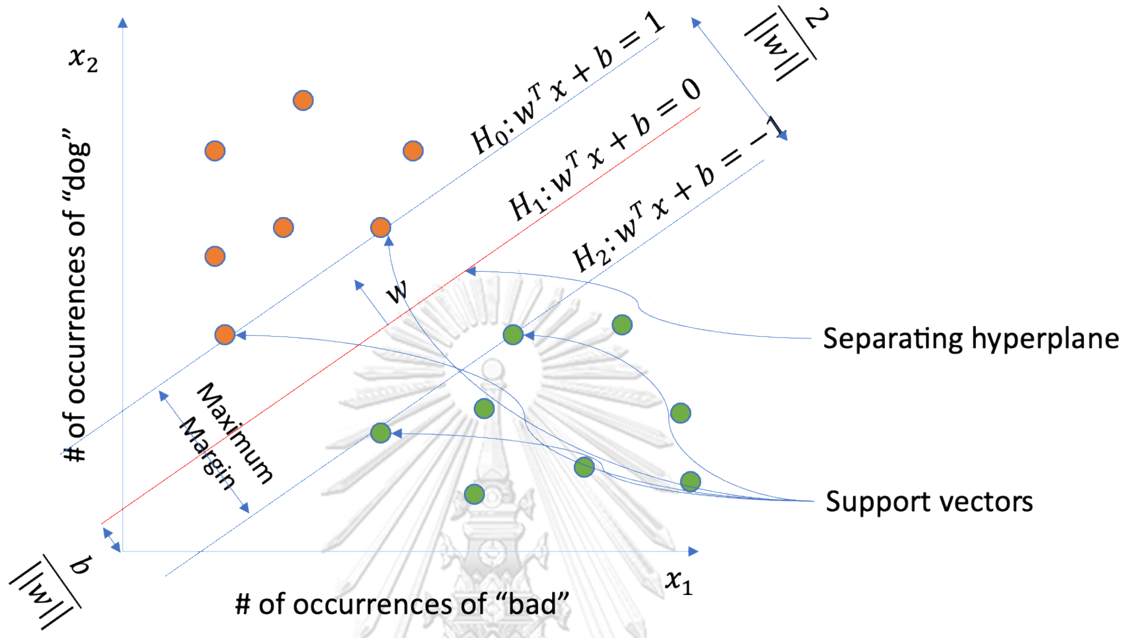


Figure 2: Linear binomial classifier using SVM

To reduce the generalization error, the hyperplane with the largest distance to the closest training points is chosen. The nearby points, which are the ones most likely to be misclassified, are known as support vectors. In this case, the separating hyperplane is H_1, a straight line in two dimensions. H_1 is assigned the equation $w^T x + b = 0$ where w is a vector orthogonal to H_1 and $\| w \| = 1$. Parallel lines H_0 and H_2 run through the support vectors on opposite sides of H_1. The margin is the distance between H_0 and H_2 and the hyperplanes chosen for $H_0, H_1,$ and H_2 create the maximum margin.

In the case of more than two classes, multiple hyperplanes are used. In the case the separation of classes in not linear, an SVM can also use a method known as the kernel trick for non-linear decision boundaries. If there is no clear separating hyperplane, a version of SVM known as soft margin can be used.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

3.2.3. Naïve Bayes

The Bayes theorem states that the conditional probability of an event is a function of prior knowledge of the probabilities of related conditions. Its formulation is seen in (5) where the probability of a document consisting of word vectors x_i for a vocabulary of size n being in one of K classes C_k is:

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k)P(x_1, \dots, x_n|C_k)}{P(x_1, \dots, x_n)} \tag{5}$$

To simplify the use of Bayes theorem, it is assumed that the features, the word vectors, are conditionally independent. This assumption, known as the naïve assumption, is given by the expression:

$$P(x_i|C_k, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|C_k) \tag{6}$$

although the co-occurrence of the words in a document may not be actually independent.
Combining (5) and (6) results in:

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k)\Pi_{i=1}^{n} P(x_i|C_k)}{P(x_1, \dots, x_n)} \tag{7}$$

Since the denominator $P(x_1, \dots, x_n)$ is constant, the equation can also be restated as a proportionality since only the relative probabilities are relevant:

$$P(C_k|x_1, \dots x_n)\ \alpha\ P(C_k)\Pi_{i=1}^{n} P(x_i|C_k) \tag{8}$$

This leads to the classification rule:

$$\hat{C} = \underset{k}{argmax}\, P(C_k)\Pi_{i=1}^{n} P(x_i|C_k) \tag{9}$$

where the class with the highest probability is assigned to the text.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

3.2.3.1. Multinomial Naïve Bayes

MNB is a variant of NB that works for multinomially distributed data which means a feature can have more than a success or failure outcome, or a Boolean value. For example, this is true when word counts are used as each word feature. From the NB formula, $P(x_i|C_k)$ is replaced with a smoothed version of maximum likelihood, also known as relative frequency counting, or $\hat{P}$, to arrive at:

$$\hat{P}(x_i|C_k) = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{10}$$

where $\hat{P}$ estimates $P(x_i|C_k)$ which is the probability of observing a word x_i given class C_k, $N_{yi} = \sum x_i$ is the number of time word x_i appears in class y (or C_k) in the training set, $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all words for class y, and α is a constant used for smoothing. MNB was found to outperform BNB in [18] where the authors believe it has an advantage when the document length has high variance.

3.2.3.2. Bernoulli Naïve Bayes

BNB is similar to MNB but instead of using word counts, each word is given a Boolean (Bernoulli) value. In other words, word count vectors are replaced by word occurrence vectors. The decision rule is based on:

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

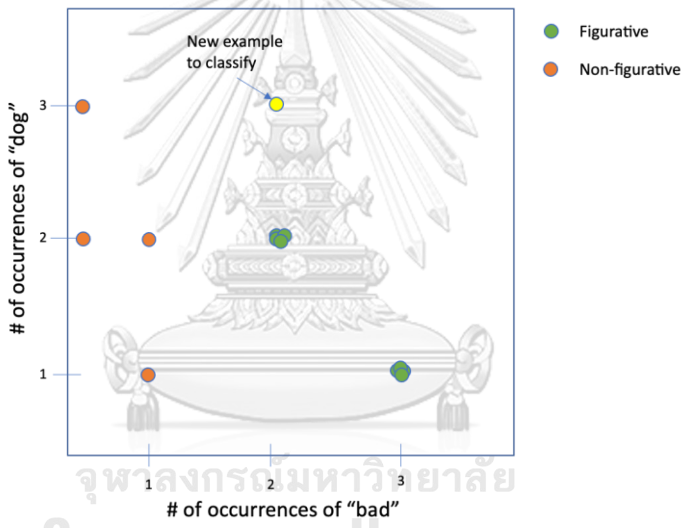
$$P(x_i|C_k) = P(x_i|C_k)x_i + (1 - P(x_i|C_k))(1 - x_i) \tag{11}$$

where $P(x_i|C_k)$ is the probability of class C_k generating the term x_i. One advantage is the ability to penalize the absence of a term. BNB was found to outperform MNB in [19] where it did better in four out of six datasets although the improved performance was not always statistically significant.

3.2.4. K Nearest Neighbors

In the kNN algorithm, during the training phase, the feature vectors for the training set and their class labels are stored in memory. During the test phase, for each sample, the k nearest points are selected. The label representing the majority of those k points is then assigned to that test data point. The parameter k is selected by the user. The distance metric used is typically the Euclidian distance.

Figure 3 shows an example of kNN. The green circles represent texts labeled "Figurative" and the orange circles "Non-Figurative". Only two dimensions appear here but the number of dimensions can be equal to the number of features. The green dots represent data points in the Figurative class and the orange dots for the Non-Figurative class. The yellow dot represents a new sample to classify. If $k = 5$, the algorithm finds the labels of the nearest five points. In this case, these are four Figurative and one Non-Figurative data points. Since the majority is Figurative, the new sample will be classified as Figurative text.



จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Figure 3: kNN

It is a simple algorithm that is easy to interpret. However, it requires high memory since it stores all of the training data in memory.

3.2.5. Decision Tree

A decision tree is a graph tree where internal nodes are a value query and leaf nodes are the classes. Figure 4 shows an example of a DT. In this case, the occurrence of a word in the document is the query node. If the word "bad" and "dog" both appear in a sentence, it is considered Figurative, but if only the word "dog" appears, it is Non-Figurative.

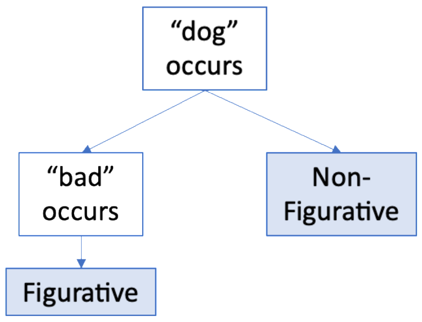


Figure 4: Decision Tree

The algorithm for the decision tree starts at the root node and then grows the tree by recursively splitting the features one by one. In choosing which feature to split, several criteria exist depending on the type of tree used.

For the CART (classification and regression tree) algorithm [20], the split is based on Gini impurity. The Gini impurity equation appears in (12) where p_i is the probability of a text with label i being chosen and $1 - p_i$ is the probability of an error in classifying that item with J total number of distinct events or classes. It measures the probability a text would be incorrectly labeled if it were chosen randomly from the distribution of labels in the training set. It results in a binary split.

$$Gini = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2 \qquad (12)$$

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Information gain is the other major splitting criteria used in algorithms such as C4.5 [21]. It is calculated as the expected reduction in entropy from branching on an attribute. Entropy represents the average level of "information" or "surprise" of a random variable, and the equation is seen in (13) where p is the vector $[p_1, \dots, p_J]$. The equation for information gain is seen in (14) where IG is the information gain, T is the training set, a is the attribute, $H(T)$ is the a priori entropy of the training set, and $H(T|a)$ is the conditional entropy.

$$H(p) = -\sum_{i=1}^{J} p(x_i) \log_2 p(x_i) \qquad (13)$$

$$IG(T,a) = H(T) - H(T|a) \tag{14}$$

3.2.6. Random Forest

Bagging averages a method's results over many samples to reduce the variance. Let $C(S,x)$ be a classifier given a training set S for data point x. The bagging procedure will draw samples $S^{*1}, \ldots, S^{*B}$ of size N each with replacement from the training set. The the output of the bagging (or bootstrap aggregation) of classifier C will follow:

$$\hat{C}_{bag}(x) = Majority\ Vote\{C(S^{*b},x)\}_{b=1}^{B} \tag{15}$$

Random forest is an improved version of bagged trees. Instead of considering all the features when splitting, a random sample of k features is taken where k is typically the square root or the log base 2 of the number of features. This reduces the correlation of the trees. Otherwise, if a small set of features strongly predicts the output, the trees may be correlated. The number of trees may range from a few hundred to several thousand depending on the training set.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Chapter 4. Proposed Methods

While Thai language social media sentiment analysis has been researched in the past using traditional machine learning methods such as in [7], [17], [22], and [16], more modern research techniques use deep learning. This includes RNN variants such as LSTM and GRU, as in [23] , [15] and [14]. Also, in 2020-1 a small number of sentiment analysis research includes variants of the more modern Transformer architecture, although they are limited to simple polarity sentiment as in [14] and [24].

Our proposed method will test two deep learning architectures – the RNN and the Transformer. In particular, this study will test the BiLSTM and DistilBERT models, respectively. These models are expected to outperform the traditional, shallow learning machine models due to better contextual representations of the words in the input. We will compare these methods to traditional machine learning methods similar to the ones used in Tuarob and Mitrpanont [7]. Both a binomial and a multinomial model will be tested. The proposed method process can be seen in Figure 5.

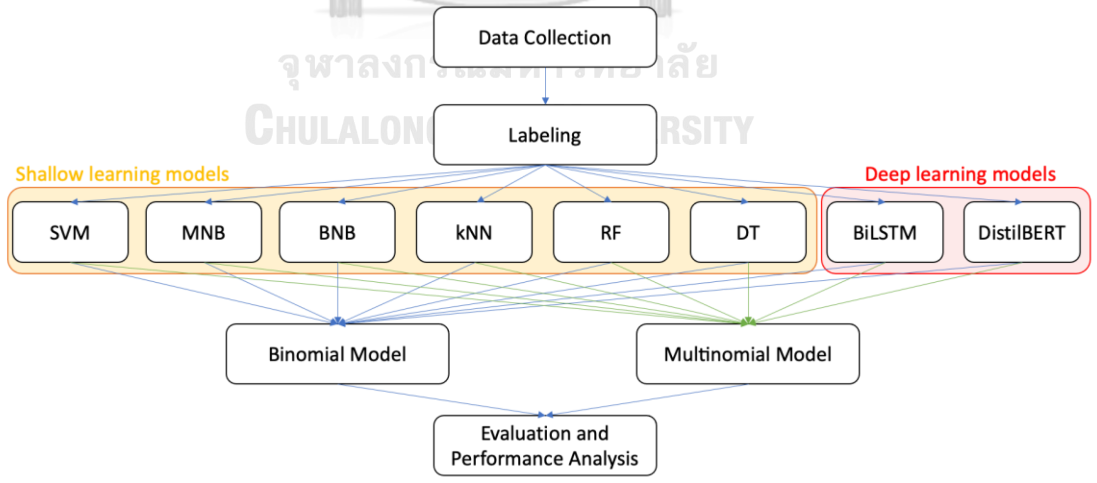


Figure 5: Proposed Method Process

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

4.1. Deep Representation Learning

The machine learning models have simple representations of the input. These models can also be referred to “shallow” learning or feature-based learning. They take the input vector x, extract features using some function $\phi(x)$, and then use these features to create a decision rule. For example, $\phi(x)$ could be an algorithm function to convert words to one-hot vectors and sum the vectors for each document. The decision rule could be $\phi(x)^T\theta \leq 0$ where θ is a vector of learned parameters. In shallow learning, the features must be manually designed and programmed by humans, which is a difficult task for complex inputs.

Deep learning will be defined as methods having two characteristics. First, they take the output from one layer and use it as input to another layer. Second, they use some form of gradient descent learning.

Deep learning, as compared to shallow learning, treats the features as learned, free parameters rather than as hard-coded, fixed inputs. Allowing the model to discover the representations needed for feature detection on its own is also known as representation learning [25]. Since the features are learned automatically, they can be represented as layers of features.

Figure 6 shows an example where a cat image is the input that is trained on for a convolutional neural network. The first transformation, pictured as the bottom-most arrow, transforms the image into a series of edges and curves as feature representations. The second transformation uses the edges and curves features from the first layer and converts them into eyes and ears feature representations in the second layer. The third transformation converts the eyes and ears feature representations into a face feature representation. These layers of transformations allow the model to learn its own features as a hierarchical representation where the lower layers are building blocks for the higher ones.

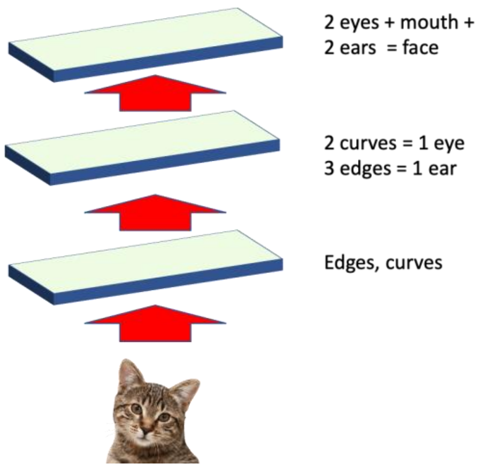


Figure 6: Layers of learned representations in deep learning

The function used to transform one layer to another layer is typically a neural network (NN). It has been shown that a single layer NN, the perceptron, has limitations, such as the ability to learn invariants, according to the Group Invariance Theorem for Perceptrons [26]. Hence the need for deep learning, or multiple layers.

Figure 7 illustrates an example of an NN diagram where x is the input, W_l and b_l are the weight matrix and bias for layer l, and y is the output.

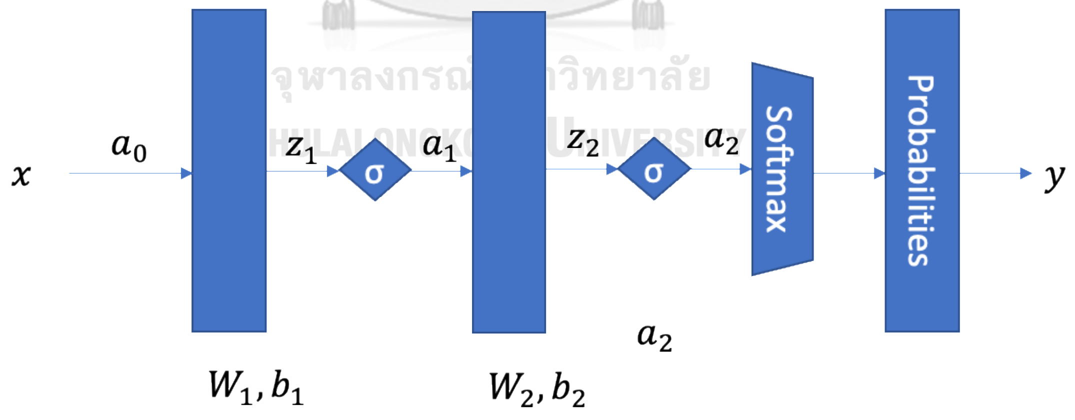


Figure 7: Deep Neural Network

Each layer calculates the formulas in (16), (17), and (18), where σ is some non-linear function known as the activation function. The presence of as little as two

จุฬาลงกรณ์มหาวิทยาลัย CHULALONGKORN UNIVERSITY

layers can approximate any continuous function within a reasonable accuracy, a property known as the Universal Approximation theorem [27] [28].

$$z_{i+1} = W_i a_i + b_i \tag{16}$$
$$a_{i+1} = \sigma(z_{i+1}) \tag{17}$$
$$softmax(z_i) = \frac{\exp(\mathrm{z_i})}{\sum_{j=1}^{K} \exp(\mathrm{z_j})} \tag{18}$$

The input is transformed at each layer from left to right. This process is known as forward propagation. At the last step of the network, a loss function is calculated. The loss functions measure the difference between predicted output and actual output. The algorithm is run multiple times. With each epoch, the parameters are adjusted in the direction of the negative gradient using backpropagation until the loss function appears to reach a minimum. In a sentiment classification task, the argmax of the set of probabilities output by the softmax function is the prediction.

NNs have several general advantages over traditional machine learning techniques. Non-linear relationships can be modeled due in part to the activation functions. Various techniques exist to improve generalization to new data such as randomized node dropout and regularization in the loss function that controls the usage of too many parameters to prevent overfitting.

One of the largest advantages NNs have is the better input representations. [7] used one-hot vectors that have no similarity measure. NNs, on the other hand, encode words as dense vectors and include a similarity measure as one of their operations. These dense vectors are referred to as word vectors, word embeddings, or word representations.

4.1.1. Word Embeddings

An ideal word embedding should include a measure of similarity. One instance where this is useful is when a particular word appears in the training set but not in the test set. One possible way to do this is to maintain a list of synonyms. However, this approach has several weaknesses. It could miss new words such as slang. The synonym chosen is subjective and different people may disagree on what is a proper synonym. It also cannot compute a similarity measure that could have a range rather than a Boolean.

One possible measure of similarity in meaning between two words is whether they are used in similar sentences. The distributional hypothesis states that words used in the same contexts are likely to have similar meanings [29]. Context refers to the nearby surrounding words in the document. The number of words is a hyperparameter.

For example, assume that some hypothetical training data has the following sentences:

- The doctor is treating patients.
- The nurse is treating patients.
- The doctor is going to the hospital.

Since "doctor" and "nurse" appear in identical contexts, it hints at the fact that these words have similar meanings, or semantic similarity. Assume that the test set may contain the partial sentence:

- The nurse is going to the ____

Based on the semantic similarity of "doctor" and "nurse", and the fact that "doctor" has appeared in a similar context to the training set, it can be predicted that there is a high probability that missing word is likely to be "hospital".

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Table 1 shows an example of a possible simple word embedding. Each column represents a different word vector with identical dimensions, with the word at top. The rows represent an aspect of its meaning. The word "Bangkok" may be rated high on the "Hot" dimension due to the weather but zero on the "Female" dimension since it does not have a gender. In an actual word embedding, however, the meaning of each dimension is learned and typically cannot be interpreted.

Table 1: Word embedding

	Bangkok	**Antarctica**	**Prince**	**Princess**
Hot	0.8	-0.9	0.0	0.0
Continent	0.2	1.0	0.0	0.0
Female	0.0	0.0	-1.0	1.0

In an NN, these embeddings must be learned. To do this, the task the NN will accomplish is to predict the nearby words of a word from its embedding value. Figure 8 illustrates the task where *w* is the word and *t* is the time step.

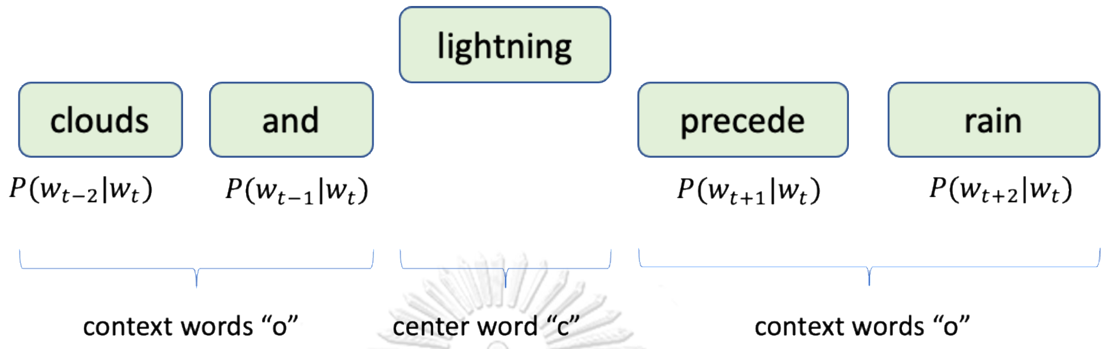


Figure 8: Context word prediction

The task can be described by first calculating:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} \tag{19}$$

where o is the context word, c is the center word, u is the learned vector representation of the word, w is the index of the word in the vocabulary, v is the learned vector representation of c, and V is the set of all possible words in the vocabulary. The goal is to optimize:

$$argmax_{u_1,\dots u_n, v_1,\dots v_n} \sum_{c,o} \log P(o|c) \tag{20}$$

for vectors u and v as optimization variables and sum for all possible c and o combinations and maximize the log likelihood which is given by the softmax expression in (19). However, this implementation is computationally expensive due to the denominator in (19).

CHULALONGKORN UNIVERSITY

The word2vec skip-gram model [30] uses a similar algorithm but is more computationally efficient. It replaces the softmax operation with a binary classification by replacing (19) and (20) with:

$$P(o\ is\ correct\ word|c) = \sigma(\mathrm{u}_{\mathrm{o}}^{\mathrm{T}} v_c) = \frac{1}{\exp(-u_w^T v_c)} \tag{21}$$

$$P(o\ is\ incorrect\ word|c) = \sigma(-\mathrm{u}_{\mathrm{o}}^{\mathrm{T}} v_c) = \frac{1}{\exp(-u_w^T v_c)} \tag{22}$$

$$argmax_{u_1,\dots u_n, v_1,\dots v_n} \sum_{c,o} (\log p(o\ is\ correct|c) + \sum_{w} \log p(w\ is\ incorrect|c)) \tag{23}$$

where σ is the sigmoid function so the denominator is no longer required and the output is a number between 0 and 1. In (23), w is chosen randomly and (22) is needed to provide contrast to words that are similar in meaning.

While these embeddings contain more information than the one-hot vectors in machine learning, they are local representations that do not take into account the context in which they are used within the actual document. For example, the word "chair" has different meanings in the phrases "office chair" and "committee chair".

Word embeddings are non-contextual word representations. The embedding for each word remains fixed. However, going forward the model architectures to be discussed will be limited to ones that use contextual representations where the same word may have more than one vector representation that changes depending on the context. The first of these is the RNN.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

4.2. Recurrent Neural Networks

An RNN is a class of variable-size NNs. This makes it naturally appropriate for natural language processing tasks since the input, sentences and documents, are also variable in size. The variable size architecture is illustrated in Figure 9. The first sentence has three words where each word is $x_{1,i}$ where i is the index of the word in the sentence 1. However, the second sentence only has two words so it skips the first layer and instead replaces it with a zero.

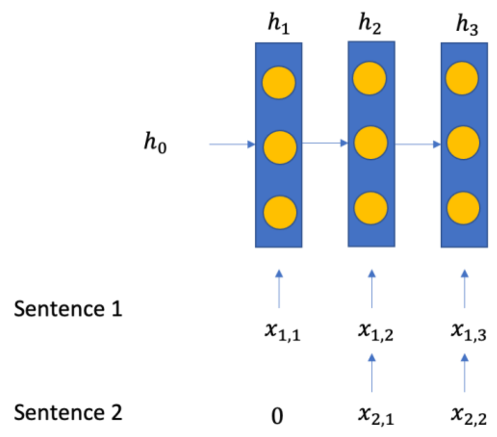


Figure 9: Variable layer count in RNN

Each layer calculates:

$$\bar{h}^{l-1} = [h^{l-1}, x_{i,t}] \tag{24}$$
$$z^l = W^l \bar{h}^{l-1} + b^l \tag{25}$$
$$h^l = \sigma(z^l) \tag{26}$$

where h is the hidden state layer, l is the layer, x is the input, i is the index of the input, t is the time step or word position, $\bar{h}$ is the concatenation of h and x, W is the weight matrix, b is the bias, and σ is the activation function.

If a different weight matrix is used at each layer, as in the DNN in Figure 7, then the later layers will end up being trained more often than others. To prevent this, the same W and b_1 is used at each layer, as shown in Figure 10.

The input x may start off as a one-hot vector before being transformed into a dense word embedding. The final output of the sequence is $\hat{y}$ which is equal to $softmax(Uh^{(t)} + b_2)$ with weight matrix U and constant b_2 applied to the final layer. In a sentiment classification task, $\hat{y}$ is the sentiment. t and l are similar variables but at $t = 0$ the sentence may not start processing at $l = 0$ if it is not a maximum length sequence.

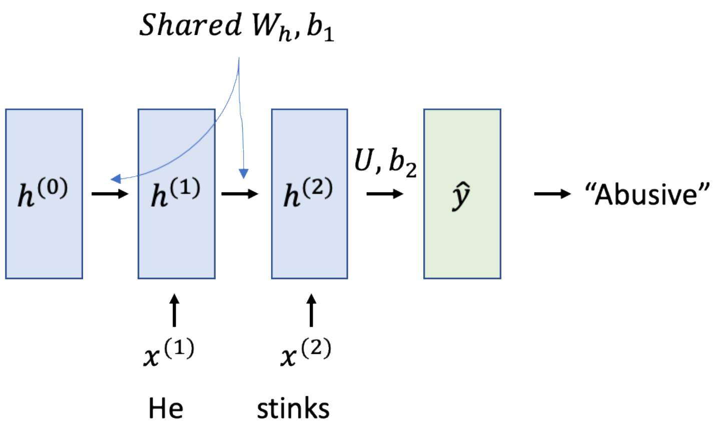


Figure 10: Simplified RNN

RNNs are very deep networks which causes problems during backpropagation where the gradient from each layer is multiplied by one another. Multiplying many gradients together will either lead to a very large number if most numbers are greater than 1 and 0 if most numbers are less than 1. This is known as the exploding gradient and vanishing gradient problems, respectively. In a vanishing gradient, the gradient signal from later layers diminishes rapidly as it approaches earlier ones [31].

4.3. Long Short-Term Memory

LSTM, a variant of RNN, preserves memory by using a cell state $c^{(t)}$to store long-term information, thereby remedying the gradient problem. Three gates are used to select and discard certain information as follows at each time step t:

CHULALONGKORN UNIVERSITY

$$f^{(t)} = \sigma\left(W_f h^{(t-1)} + U_f x^{(t)} + b_f\right) \quad (27)$$
$$i^{(t)} = \sigma\left(W_i h^{(t-1)} + U_i x^{(t)} + b_i\right) \quad (28)$$
$$o^{(t)} = \sigma\left(W_o h^{(t-1)} + U_o x^{(t)} + b_o\right) \quad (29)$$
$$\tilde{c}^{(t)} = tanh\left(W_c h^{(t-1)} + U_c x^{(t)} + b_c\right) \quad (30)$$
$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \quad (31)$$
$$h^{(t)} = o^{(t)} \circ tanh\left(c^{(t)}\right) \quad (32)$$

where $x^{(t)}$ is the input vector, $h^{(t)}$ is the hidden state vector, $c^{(t)}$ is the cell state vector, $i^{(t)}$ is the input gate vector, $f^{(t)}$is the forget gate vector, $\tilde{c}^{(t)}$ is the new cell content vector, $o^{(t)}$ is the output gate vector, b is a constant, W and U are weight matrices, $tanh$ is the hyperbolic tangent function, and σ is the sigmoid function.

Figure 11 shows a diagram of an LSTM cell where $c^{(t)}$ is known as the "long term" memory and $h^{(t)}$ is known as the "short term" memory since the former has small changes between steps while the latter changes frequently. The forget gate $f^{(t)}$ has a range between 0 and 1 where 0 would represent forgetting the previous value and the new value would only use the current input. The circle marked "W, U, b" includes the various weights.

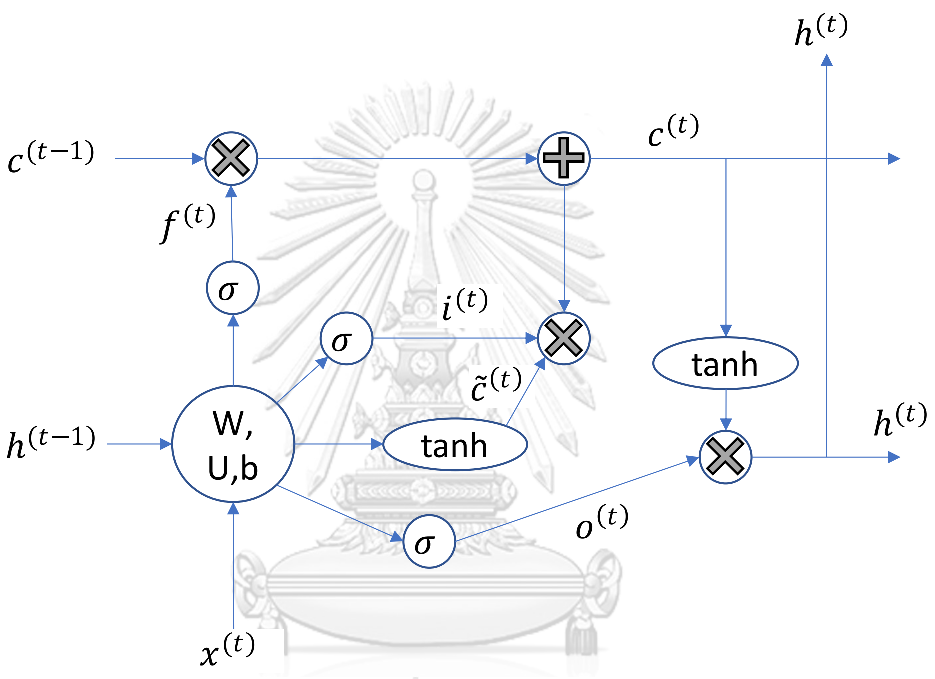


Figure 11: LSTM Cell

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

4.3.1. Bidirectional LSTM RNN

The BiLSTM has two LSTMs. The forward LSTM processes left-to-right while the backward LSTM processes right-to-left. The two opposite directions create different, complementary contextual representations that can lead to better predictions. Figure 12 shows an example. The final representation vector or hidden state concatenates the output of the forward and backward LSTMs.

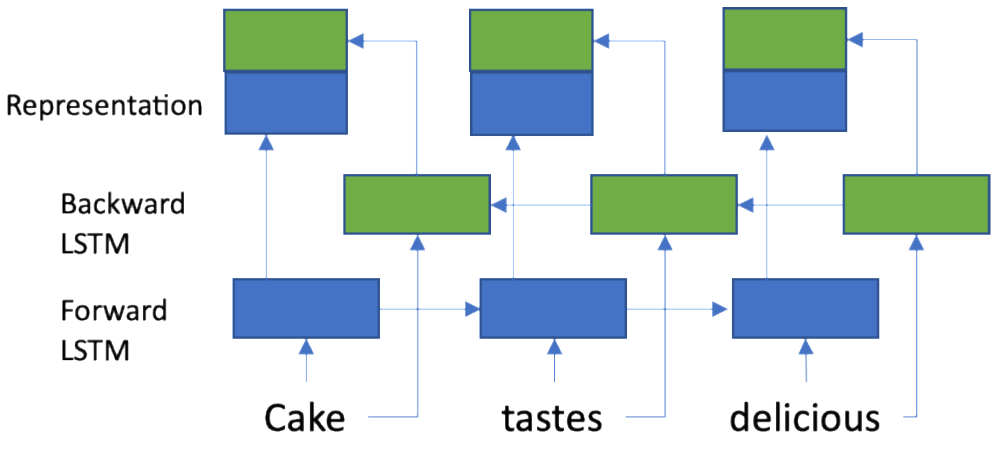


Figure 12: Bidirectional LSTM

The formulas for the BiLSTM are:

$$\overrightarrow{h}^{(t)} = LSTM_{FW}\left(\overrightarrow{h}^{(t-1)}, x^{(t)}\right) \quad (33)$$

$$\overleftarrow{h}^{(t)} = LSTM_{BW}\left(\overleftarrow{h}^{(t+1)}, x^{(t)}\right) \quad (34)$$

$$h^{(t)} = \left[\overrightarrow{h}^{(t)};\ \overleftarrow{h}^{(t)}\right] \quad (35)$$

where (33) is the forward LSTM, (34) is the backward LSTM, (35) is the concatenated hidden states, and $LSTM$ represents a computation of one step of the LSTM.

4.3.2. Stacked LSTM RNN

RNNs can have many layers, as in Figure 13. The output from one layer is used as input for another layer. These are known as stacked layer RNNs (or LSTMs). This allows for greater model complexity. Sutskever et al [32] found that four layers significantly outperformed single layer LSTMs in their sequence-to-sequence learning models.

In the experiments a four-layer stacked BiLSTM is used.

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

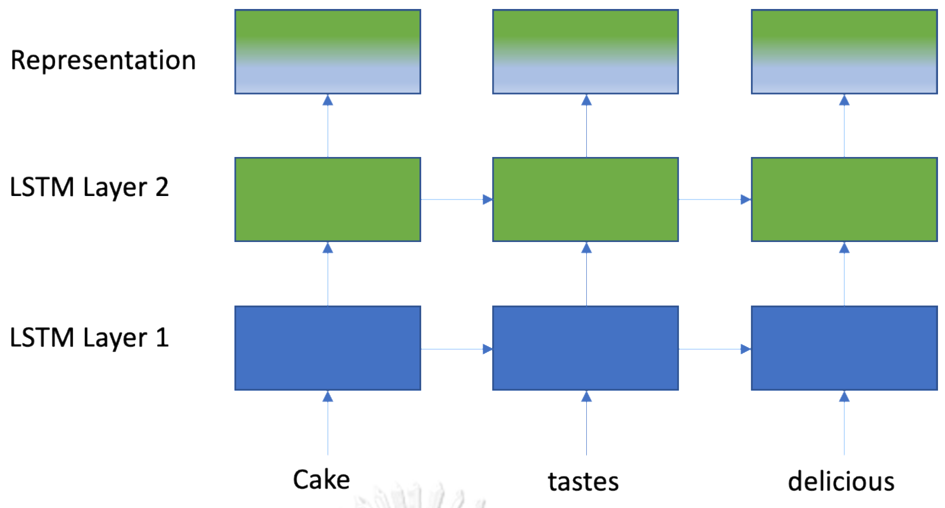


Figure 13: RNNs with multiple layers

4.4. Transformers

The recurrence mechanism in the LSTM makes parallelization difficult. Transformers fixed this problem using attention mechanisms. One example is in sequence-to-sequence (seq2seq) [32] machine translation models, where the first sequence is a sentence in one language to be translated to a different language in the second sequence. The source language is used as input for the first RNN where the RNN is referred to as the encoder. The output of the encoder is used as input to a second RNN that is referred to as the decoder which translates the input into a second language. Figure 14 shows an example:

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

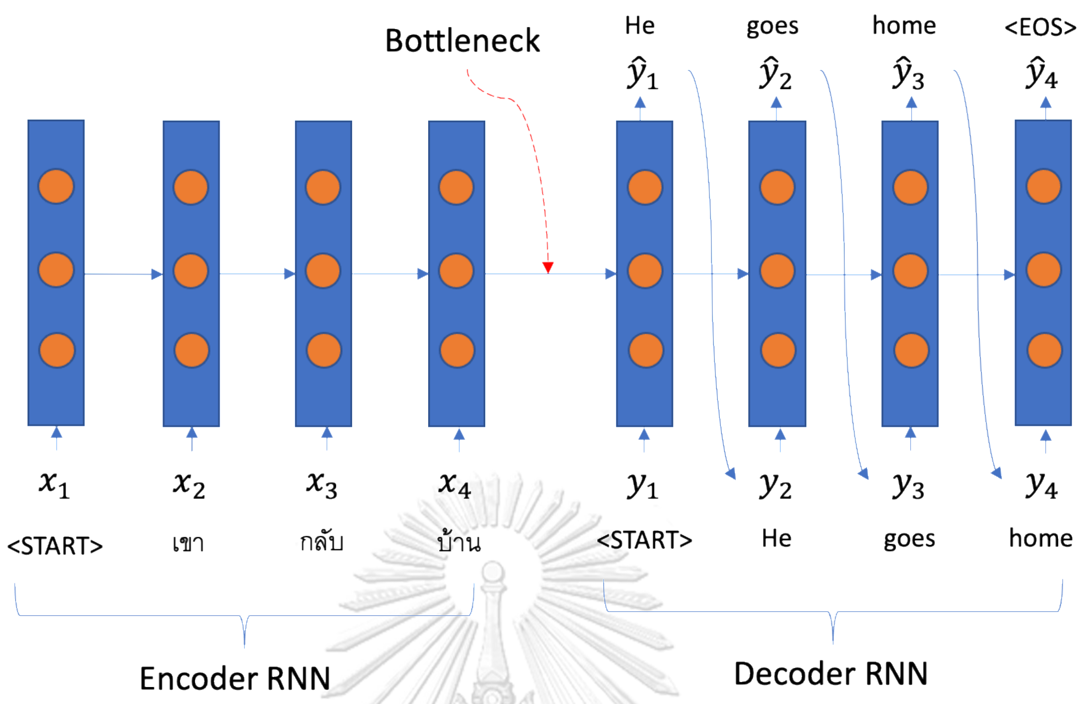


Figure 14: Sequence to sequence model

where x is the source sentence, x_t is the input word vector for x at time step t, y is the target sentence, <START> is a token indicating the start of a sentence, <EOS> is a token indicating the end of a sentence, and $\hat{y}_t$ is the output of each layer in the decoder RNN. The neural machine translation (NMT) model calculates:

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

$$P(y|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_1, y_2, x) \dots P(y_T|y_1, \dots, y_{T-1}, x) \qquad (36)$$

The task of the NMT is to predict the next word given the previous words, which is known as the neural language model. One problem with this architecture is that the decoder only has access to the information from the final layer and cannot access previous layer outputs of the encoder. This problem is also known as the bottleneck problem.

4.4.1. Model architecture

The Transformer takes the encoder and decoder RNNs and replaces it with a series of encoder and decoder modules. Attention mechanisms replace the recurrent mechanisms, allowing for greater parallelization.

Attention as a mechanism simply takes a weighted sum of the encoder hidden states of the input word vectors to use when decoding. The weights depend on the query. In other words, the decoder has access to every hidden state in the original input sequence, rather than simply the final hidden state.

A simple seq2seq model on the other hand cannot choose selectively which word vectors to weight since it cannot go back to the refer to the original input as it is processing. Thus, the attention mechanism solves the bottleneck problem. Several variations of attention exist. The Transformer uses self-attention, also more descriptively referred to as scaled dot product attention, which will be described later.

4.4.2. Encoder Stack

Each encoder layer is made up of two sub-layers which are the self-attention layer followed by a feed-forward network. A residual connection [33] is also used for each of the two sub-layers, and then goes to a layer normalization [34]. The outputs of each of these sub-layers as well as the embedding layer at the start have the same dimension output of $d_{model} = 512$. One encoder layer can be seen in Figure 15. Encoder layers are repeated six times.

จุฬาลงกรณ์มหาวิทยาลัย CHULALONGKORN UNIVERSITY

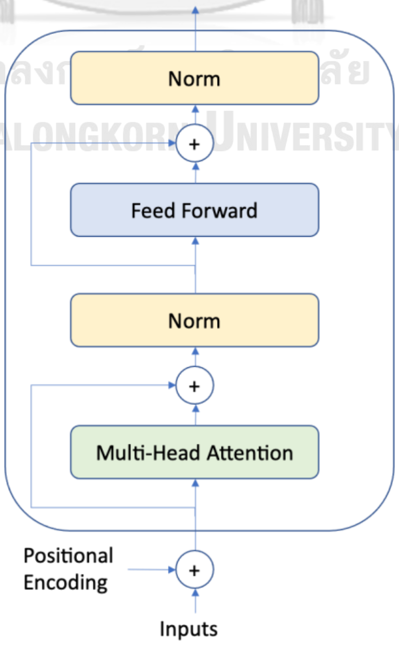


Figure 15: Encoder architecture

4.4.2.1 Residual connection

In a typical NN, the input x gets transformed into $F(x)$ in one layer, whose output is then used as an input for the next layer. A residual connection is shown in Figure 16. Here the output to the feed-forward network layer is $F(x)$ which is then added to the original input x to get $H(x) = F(x) + x$. $H(x)$ is used as input for the next layer. Intuitively, the reason for this is to not lose any information. Rather than using an entirely new input, the layer uses the previous input as well as a new input.

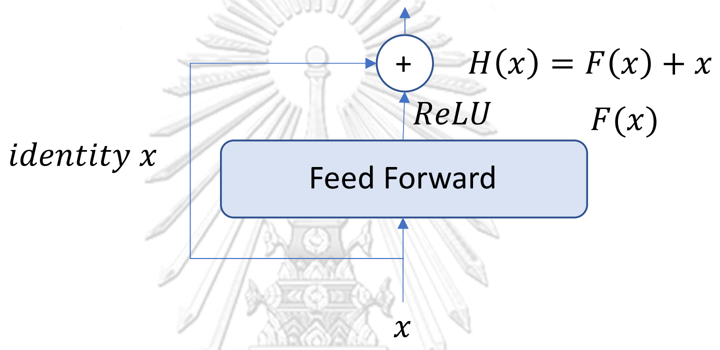


Figure 16: Residual connection

A residual connection, also known as a skip connection, can make deeper networks easier to train [33]. In a typical network, in order to back propagate, the derivative of many functions must be calculated and multiplied together due to the chain rule. This leads to either 0 if all the derivatives are small or infinity if they are all large. However, if they are close to 1, then they are less likely to converge to these two extremes. In a typical network, the Jacobian of a layer is $\frac{dH}{dx}$. In a residual connection, the Jacobian is $\frac{dH}{dx} = \frac{dF}{dx} + I$ where I is the identity matrix. Thus, if the value of $\frac{dF}{dx}$ is small, the Jacobian will be close to 1, leading to more usable value when applying the chain rule to these values.

4.4.2.2. Layer Normalization

Typical normalization standardizes the inputs by computing a new standardized mean and standard deviation for each dimension of the input vectors. Batch normalization does this for the layer inputs before activations (the summed

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

inputs to each hidden unit) on each training mini-batch [35]. However, in an RNN the sequences have different lengths so batches can vary in size depending on the length of the sentences.

Rather than calculating based on a mini-batch, layer normalization calculates the new standardized numbers by including all of the hidden layers on a single training data point, thereby making it more applicable for RNN models [34]. The calculations are:

$$\mu = \frac{1}{d}\sum_{j=1}^{d} a_j \tag{37}$$

$$\sigma = \sqrt{\frac{1}{d}\sum_{j=1}^{d}(a_j - \mu)^2} \tag{38}$$

$$\bar{a} = \frac{a - \mu}{\sigma}\gamma + \beta \tag{39}$$

where μ is the mean, j is the index for each dimension of a, σ is the standard deviation, a is the activation vector, d is the dimension, $\bar{a}$ is the transformed activation, γ is a learned scale, and β is the bias. Using these formulas, there are no dependencies between training cases, so information does not have to be shared across the batch.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Layer normalization is applied in a Transformer as follows:

$$LayerNorm\big(x + Sublayer(x)\big) \tag{40}$$

where $LayerNorm$ is the function, x is the input, and $Sublayer$ is either the attention layer or the feed-forward network layer.

4.4.3. Scaled dot-product attention

The self-attention layer operation is also known as scaled dot-product attention. An illustration can be seen in Figure 17.

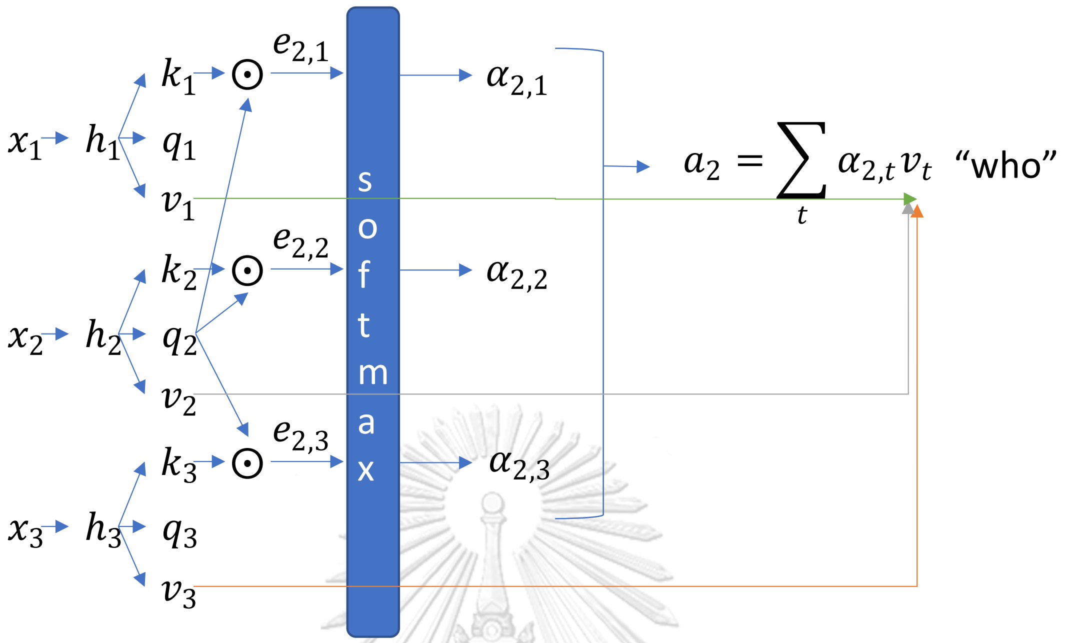


Figure 17: Simplified self-attention operation

Attention represents the weight assigned to each word in the sequence. The formula for it is:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{41}$$

where multiple queries are combined in a single matrix *Q*. The *K* and *V* matrices also represent multiple keys and values. The queries and keys have dimension d_k and the values have dimension d_v.

The variables in Figure 17 are calculated as follows:

$$h_t = \sigma(Wx_t + b) \tag{42}$$
$$q_t = q(h_t) \tag{43}$$
$$k_t = k(h_t) \tag{44}$$
$$v_t = \mathrm{v}(h_t) \tag{45}$$
$$e_{l,t} = \mathrm{q_l} \cdot k_t \tag{46}$$
$$\alpha_{l,t} = exp(e_{l,t})/(\sum_{t'} \exp(e_{l,t'}) \tag{47}$$
$$a_l = \sum_t \alpha_{l,t}\, v_t \tag{48}$$

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

where h_t is the hidden state for word x_t which uses weight matrix W, activation function σ, and bias b, to output a key vector k_t, a query vector q_t, and a value vector v_t at time step (position) t in the sentence.

k_t, q_t, and v_t are linear functions of h_t while h_t is a non-linear function of x_t. When using a seq2seq model, the queries are derived from the decoder while the keys are derived from the encoder, but in Self-Attention, k, q and v are generated from the same input. In Figure 17, word x_2 conducts a query q_2 to find the "who", or subject, of a sentence. q_2 finds a key k_t that is most similar to itself by executing a dot product operation with each k_t. The sums of these dot product operations are converted to a probability measure through the softmax operation and multiplied by the respective values v_t to arrive at an attention output a_2. The softmax operation is used in (47) rather than $argmax$ since it is differentiable which is needed during backpropagation.

The multiple keys, queries, and value vectors from different time steps are combined into single matrices. By combining multiple calculations into a single matrix, there is a single multiplication per layer which leads to efficiency on the Tensor Processing Unit (TPU). The batch size is based on number of words and not limited to the number of sequences.

In the Transformer, since the operation is a single multiplication per layer, it can process 3 sentences of 4 words each in a single batch size of 12, where each row represents a different sentence and each column a different word. In the LSTM, only one word at a time can be processed. Thus, for 3 sentences, the batch size is limited to 3. TPUs can process large matrix multiplications which the Transformer can take advantage of.

The attention output a_2 in Figure 17 is an example of a contextualized representation of input x_2. In a word2vec embedding, each word has only one representation vector. However, in many instances a word can have more than one meaning. To give an instance, the word "bank" may be used to either represent a place to store money or a geographic entity. A contextualized representation outputs a different word vector depending on the words surrounding it, so the single word "bank" can have multiple word vectors which allows for more accurate interpretations of the meanings of the sentences in which it is used. More accurate interpretations allow for better sentiment classifications.

4.4.4. Multi-head attention

The equation for multi-head attention is:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \tag{49}$$
$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{50}$$

where W_i^Q is the query weight matrix for attention head i, W_i^K is the key weight matrix, W_i^V is the value weight matrix, and W^O is the output weight matrix. Q, K and V represent the query, key, and value matrices.

An illustration of multi-head attention appears in Figure 18. Each row is made up of a different attention head. The queries, keys, and values are not the result of functions provided by the model but rather learned functions. Each attention head focuses on a different representation subspace. In the illustration, the subspaces focus on the "who", "did what", and "to whom" aspects of the sentence. The output of each layer is concatenated in a final a_2 attention output. The utilization of multiple dedicated attention head outputs should create a better representation than the output of a single attention head that averages the different aspects.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

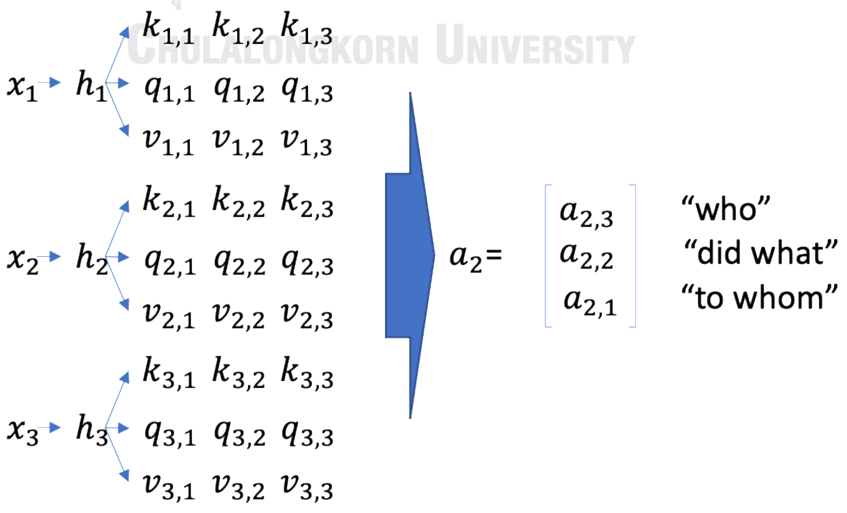

x_1
h_1
$k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$q_{1,1}$ $q_{1,2}$ $q_{1,3}$
$v_{1,1}$ $v_{1,2}$ $v_{1,3}$
x_2
h_2
$k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$q_{2,1}$ $q_{2,2}$ $q_{2,3}$
$v_{2,1}$ $v_{2,2}$ $v_{2,3}$
x_3
h_3
$k_{3,1}$ $k_{3,2}$ $k_{3,3}$
$q_{3,1}$ $q_{3,2}$ $q_{3,3}$
$v_{3,1}$ $v_{3,2}$ $v_{3,3}$
$a_2 =$
$a_{2,3}$
$a_{2,2}$
$a_{2,1}$
"who"
"did what"
"to whom"

Figure 18: Multi-head attention

The functions for computing multi-head attention are:

$$e_{l,t,i} = q_{l,i} \cdot k_{l,i} \tag{51}$$

$$\alpha_{l,t,i} = \frac{\exp(e_{l,t,i})}{\sum_{t'} \exp(e_{l,t',i})} \tag{52}$$

$$a_{l,i} = \sum_{t} \alpha_{l,t,i} v_{t,i} \tag{53}$$

where e is the attention score, l is the position where the position is being computed, t is the time step (the position for the values), i is the attention head index, α is the attention weight, v is the value vector, q is the query vector, and a is the attention.

4.4.5. Feed-forward network

The key, query, values, and attentions are linear functions. This can be seen in:

$$a_l = \sum_{t} \alpha_{l,t} v_t = \sum_{t} \alpha_{l,t} W_v h_t = W_v \sum_{t} \alpha_{l,t} h_t \tag{54}$$

where each subsequent self-attention layer is a linear transformation of the previous self-attention layer (not including the non-linear weights). The purely linear transformations limit the types of functions that can be approximated.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

The other sub-layer in the encoder in addition to the attention layer is a fully connected feed-forward network for each position in the sequence. It is a simple network with the function given in (55) where the $ReLU(x) := \max(0, x)$, x is the input, W_i are the weight matrices, and b_i are the biases.

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \tag{55}$$

Within the layer, the same parameters are used for each position of the sentence but are different from layer to layer. The feed-forward network adds more expressiveness to the representations.

4.4.6. Positional encoding

As the Transformer does not use recurrence to process information as the LSTM does, positional information is not included in the encoding, which can be important to the meaning of a sentence. For example, “The condo is horribly good” has a positive sentiment while “The ‘good’ condo is horrible” has a negative sentiment, despite using nearly the same words, due to the relative position of the words. The position embedding is added to the input word embedding early in the Transformer before the encoder stack.

A naïve position embedding could simply concatenate the absolute position with the word embedding. However, relative position is better. For example, in the sentences “I went to campus every Monday” and “Every Monday I went to campus”, the position of “to campus” coming after “I went” is more important than the absolute position of “to campus” in the sentence.

Therefore, a relative position encoding is added to the original word embedding. The positional encodings have a dimension size of d_{model} so they can be summed with the word embeddings. Sine and cosine functions with varying frequencies are used:

$$p_{(t,2i)} = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right) \tag{56}$$

$$p_{(t,2i+1)} = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right) \tag{57}$$

where t is the position of the word, i the index of the dimension of the position vector p_t, d the size of the word vector, and 10,000 an arbitrary constant which can be made smaller if d is smaller than the 512 used in the original paper. The position encoding vector p_t alternates between using a sine and cosine function.

Figure 19 illustrates the construction of a position embedding. The curves are sine waves for different values of i using $d = 128$. The y axis is the value of the sine curve and the x axis is the word index in the sequence. The sine wave is only used

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

for the even-numbered dimensions of the position embedding while a cosine one is used for the odd-numbered ones.

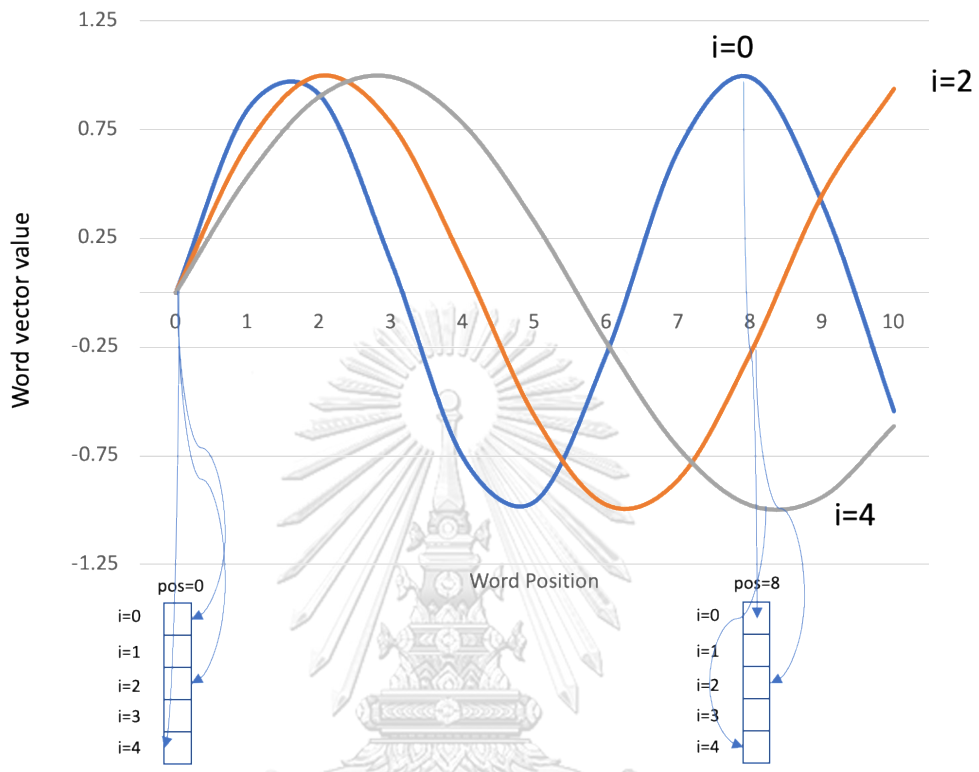


Figure 19: Position embeddings

Let $p_{t,i}$ represent the position embedding vector dimension index where t is the word index and i is the dimension of the vector. For the word at position 0, $p_{0,0}$, $p_{0,2}$ and $p_{0,4}$ all have values of 0. For the word at position 8, $p_{8,0}$, $p_{8,2}$ and $p_{8,4}$ have values of 0.99, -0.28, and -0.98.

4.4.7. Masked Attention in Decoder Stack

Once the encoder layers process their inputs, the outputs are used as input for the decoder stack. This thesis will use a pre-trained model for machine translation to compute the initial word embeddings so it can be helpful to understand the basic mechanics. Additionally, it will serve as relevant background to the development of the BERT model.

For the decoder, self-attention is modified to turn it into masked attention. Since the decoder must predict the next word in a sentence using only the previous

Chulalongkorn University

words, the words following the target word must be omitted from the calculation. Thus, the attention score $\boldsymbol{e}$ for layer $\boldsymbol{l}$ at time step $\boldsymbol{t}$ is calculated from query vector $\boldsymbol{q}$ and key vector $\boldsymbol{k}$ as follows:

$$e_{l,t} = \begin{cases} q_l \cdot k_t \; if \; l \geq t \\ 0 \; otherwise \end{cases} \tag{58}$$

However, since the primary task is sentiment classification, a decoder is not required. Thus, a variant of the Transformer was used, the BERT, whose architecture allows it to be used in a wider range of tasks.

4.5. Bidirectional Encoder Representations from Transformers

BERT extends the Transformer architecture to tasks other than language modeling [36]. The language modeling task is used to create a pre-trained model whose word representations can then be used for further processing for tasks such as sentiment classification.

4.5.1. Pre-trained language model

Natural language modeling is a task to predict the next word in a sentence, given the previous words. Mathematically, this probability is:

$$P(y_t | y_1, \dots, y_{t-1}) \tag{59}$$

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

where $\boldsymbol{y}$ is a word and $\boldsymbol{t}$ is the position in the sentence. The language model is a system to generate the probability distribution in (59).

BERT uses a pre-trained language model, which means it was trained on a corpus different from the one that will actually be tested. Using a pre-trained model is also known as transfer learning, where the learned representations from one model can be transferred, or used as input, to another.

Transfer learning is especially useful for a small corpus that has a model which has relatively many parameters. For example, our corpus has a few thousand texts as compared to Wikipedia which for English has around 2.5B words. For smaller models, in terms of number of parameters, such as LSTM, it may not be necessary to use a pre-trained model.

4.5.2. Masked language modeling

In typical language modeling, only the previous words in a sentence are used to predict the next word, as in (59). This is also the mechanism used for masked attention in the Transformer. However, BERT does something different it refers to as masked language modeling (MLM).

An illustration of BERT appears in Figure 20. The input starts at the left. An extra token [CLS] is added to the beginning, known as the classification token. One or more words is replaced with a [MASK] token to identify which words will be predicted during the language modeling task. The [SEP] token separates sentences.

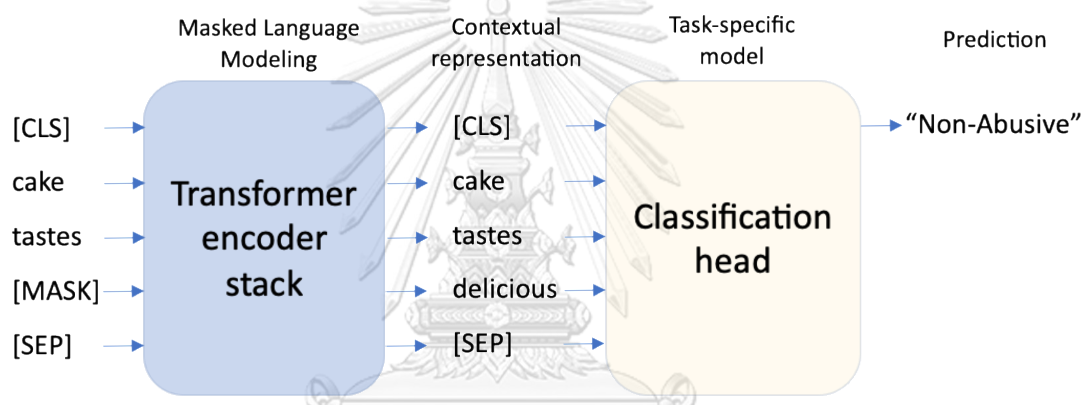


Figure 20: BERT model

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

The Transformer encoder stack executes the language modeling task. The Transformer's masked attention model predicts the following word to the right, given the previous words on the left, or uni-directionally. However, BERT's approach is to randomly mask a certain percentage of the words, typically 15%, and predicts these words. BERT refers to this as a bidirectional approach, although it could also be called non-directional. The output is a set of contextual representations of the words.

4.5.3. Task specific module

Several task-specific modules exist. This flexibility is one of the reasons why BERT is popular. BERT is pre-trained as a language model first in the encoder stack, as seen in Figure 20. Each encoder module outputs a different representation of the sequence. Each of these layer outputs can be used as an input into the task specific module in different ways including first layer only (the embedding), last layer only, sum all of the layers, sum certain layers, or concatenate certain layers.

The output from the language model is then used as input for the task-specific module. For the sentiment classification task, it can be a simple feedforward network that predicts the [CLS] token.

4.5.4. Fine tuning

Rather than simply tuning the task-specific module parameters, the parameters of the entire model are adjusted. Thus, if the pre-trained language model had 200M parameters and the classification head had 200k parameters, the fine-tuning could adjust 200.2M parameters, although the actual fine tuning would only impact the relevant parameters and thus possibly number only slightly above 200k.

4.5.5. Segment embeddings

In addition to position embeddings, BERT also has segment embeddings as seen in Figure 21. A segment (or type) embedding can detect the "type" of word. For example, in a web search, one segment could be the URL and another could be the query. Segment embeddings are learned. In the example in Figure 21, the segment embeddings represent two different sentences.

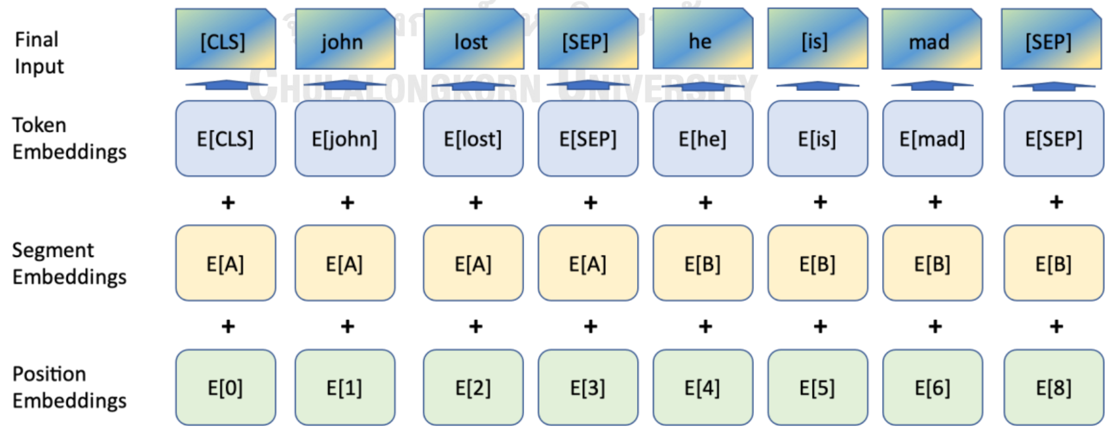


Figure 21: Embeddings

BERT is expected to outperform both the machine learning models and BiLSTM due to its better contextual representations due to the various architecture

CHULALONGKORN UNIVERSITY

improvements including the multi-head self-attention mechanism and masked language modeling.

Sentences in the figurative language classification class in particular can be considered a good test of the contextual representations. In this category the Thai word for certain animals or animals in general can be either treated literally or figuratively depending on the surrounding contextual words. For example, the appearance of the word “ไอ้” in a sentence connotes the animal words as being figurative and abusive.

4.6. DistilBERT

While a larger BERT model such as RoBERTa [37] could have been used in order to obtain higher accuracy levels, it would require the use of large computational resources and thus may process information with large latencies. However, in highly regulated countries where website owners can be liable for the abusive speech of users, real time detection with low latency is a priority. Additionally with social networks serving many users simultaneously, the amount of computation that can be used is a constraint.

Hence DistilBERT, a compressed version of BERT, was chosen over other variations of BERT. On a General Language Understanding Evaluation (GLUE) sentiment analysis task, DistilBERT finished 60% faster than BERT while scoring 97% of BERT’s performance on the GLUE benchmark despite using 40% less parameters [6]. DistilBERT reduced the number of layers used by half. It also removed the token-type embeddings and the pooler.

4.6.1. Teacher-Student Training

Distillation is one method of model compression [38] [39]. The idea can be seen in Figure 22. The Teacher is the full model trained to optimize for maximum accuracy, training on the actual dataset and labels. The Student is a smaller model, possibly 50x smaller, that instead of training on the original dataset, trains on the labels predicted by the Teacher, also known as pseudo-label data.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

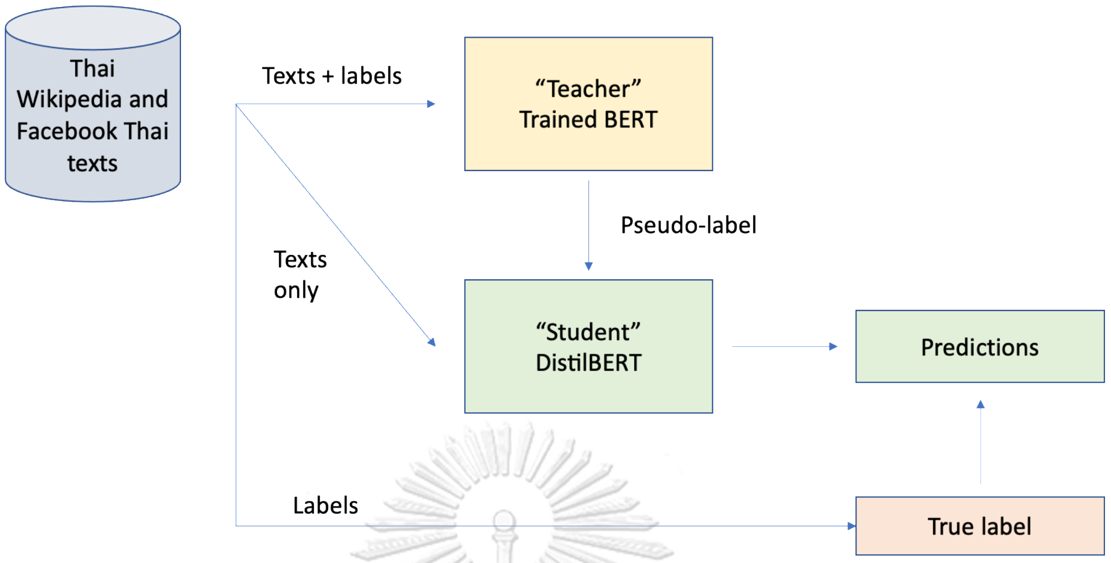


Figure 22: Distillation

In theory the Teacher builds a simpler function approximation of the actual generating function, and thus can be represented using a simpler, more compact model. Language modeling may be considered the "ultimate" NLP task since it can be used in other tasks. In the process of language modeling it learns many latent features. The specific task then only emphasizes the latent features relevant for its own objective, so it is only learning a subset of the original features. The alternative would be to simply skip the Teacher step and just use a smaller model but this was shown to have worse results [40].

4.6.2. Distillation

One naïve solution is to simply use the labels of the Teacher model as the true labels and have the Student maximize the log probability of predicting these labels. However, this leaves out the important information of the relative probabilities of the wrong answers. For example, the word "dog" is more similar to "cow" than it is to "car". This information could be useful in the classification of sentences.

One solution to retaining the information contained in the probability distribution while compressing the model was introduced in 2006 [38]. In the softmax formula, the output of the final layer is $z_i(x)$, also known as the logit or model

CHULALONGKORN UNIVERSITY

score, where $\boldsymbol{i}$ is the class from all classes $\boldsymbol{j}$ of input $\boldsymbol{x}$. The logit is then converted into a probability $\boldsymbol{p}$.

$$p_i(x) = \frac{\exp\big(z_i(x)\big)}{\sum_j \exp\big(z_i(x)\big)} \tag{60}$$

Rather than pass parameter $\boldsymbol{p}$ to the Student model, the predecessor logits $\boldsymbol{z}$ are transferred instead. The objective function is then set to minimize the squared difference between the logits of the Teacher and the Student.

Distillation is a refinement of the preceding model compression method that was introduced by Hinton in 2015 [39]. One weakness of the softmax function is that the exponential function produces extreme probabilities where small probabilities are transformed into being close to zero, making similarities more difficult to detect. Distillation remedies this by modifying the softmax with an additional hyperparameter referred to as "temperature" $\boldsymbol{T}$ in the formula below to create a smoother probability distribution. This function is referred to as the softmax-temperature.

$$p_i(x) = \frac{\exp\left(\frac{z_i(x)}{T}\right)}{\sum_j \exp\left(\frac{z_i(x)}{T}\right)} \tag{61}$$

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

The same softmax temperature parameter value is used in both the Teacher and Student models for training. During inference the temperature is reset to 1.

Cross entropy is often used as a measure of the difference between two probability distributions. In distillation, the cross-entropy training loss used by the Student is:

$$L_{ce} = \sum_i t_i \cdot \log(s_i) \tag{62}$$

where t_i is the Teacher's probability estimation for the target label and s_i is the Student's probability estimation of sample i.

In the equation below, L_{mlm} is the typical cross entropy loss function used as an objective function in the masked language modeling task where the identity of the masked words is predicted given the other words in the sentence. The variable p refers to the actual probability distribution and q the estimated distribution of sample i.

$$L_{mlm} = -\sum_i p_\mathrm{i} \log_2 q_i \tag{63}$$

These two loss functions, L_{ce} and L_{mlm}, are combined linearly to be used as the final objective function. To ensure correct alignment between the Teacher and Student hidden state vectors, a cosine embedding loss is also used, calculated as follows:

$$L_{cos}\,(x,y) = \begin{cases} 1 - \cos\,(x_t, x_s), & y = 1 \\ \max\,(0, \cos(x_t, x_s) - \mathrm{margin}), & y = -1 \end{cases} \tag{64}$$

where x is the hidden state vector of the Student s or Teacher t, y is a tensor label containing 1 or -1, and **margin** is a number from -1 to 1.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Chapter 5. Experiments and Results

Regulations for social media content differ from country to country. To account for this, two scenarios are envisioned. In a strict scenario, all abusive content is regulated and for this a binomial model is developed. In a less strict scenario, only certain types of abusive content are regulated and for this a multinomial model is developed.

5.1. Dataset

In order to replicate the methodology of [7], abusive comments were collected from Facebook public pages. The data were manually retrieved as automated scraping is no longer permitted. The same five classes as [7] were used: Figurative, Rude, Dirty, Offensive, and Non-Abusive. Rude texts will contain at least one rude word including มึง or กู. No hostile intent is required. Figurative text is defined as metaphorical and departing in meaning from the literal sense. Examples of figurative words include ควาย and สัตว์ which can have abusive meanings depending on how they are used. Context can be used to discern the intention of the sentence. Offensive texts are intended to attack, annoy, or harass an individual or group of people. Sarcasm is included in this category. Harsh language is not required. Dirty texts have sexual undertones and typically have explicit sexual wording. If none of these categories apply, the text is labeled Non-Abusive.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

A new dataset was created since no public dataset of this type exists. Since the categories can be difficult to interpret, undergraduate linguistic major students from Chulalongkorn university were hired. Offensive public Facebook pages were chosen as the data source and comments from 2018 to 2021 were collected. The initial labels were verified by a second student and if there was a disagreement, the researcher would decide the final labels. Neutral sentiment texts from the Wisesight Sentiment dataset [41], which also has Facebook comments as its sources, were used for the Non-Abusive texts. Tokenization, the process of dividing the sentences into individual words, was accomplished using PyThaiNLP [42]

A total of 6,770 texts in the Abusive category were gathered and 14,561 Non-Abusive texts were used. Within the Abusive category there were 5,617 Rude; 2,661

Figurative; 1,565 Dirty; and 1,224 Offensive comments, and each of these comments can have multiple labels. The range of the length of the texts was wide, ranging from one to 556 words. The dataset was unbalanced but oversampling and undersampling methods did not improve results.

5.2. Performance Measurement

Several performance measurements were available. The most obvious one to use may be accuracy. The formula is:

$$Accuracy = \frac{TP}{TP + FP + TN + FN} \tag{65}$$

where TP is True Positive, FP is False Positive, TN is True Negative, and FN is False Negative. The categories can be more easily seen in the table below:

Table 2: Confusion Matrix

		Actual values	
		Positive	**Negative**
Predicted Values	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative

A Positive result is a text classified as Abusive or in one of the Abusive sub-categories. However, in the case of unbalanced data, it could be a misleading statistic. For example, if 99.9% of the text on social media is not abusive and the model simply guessed all Negative, it would have a high accuracy statistic but would not detect any actual abusive text. To get better granularity and to account for unbalanced data, Precision and Recall are better statistics. Precision, also known as positive predictive value, measures the proportion of correct positive predictions to total positive predictions, as seen in the equation:

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

$$Precision = \frac{TP}{TP + FP} \tag{66}$$

Recall, also known as sensitivity, measures the correct positive predictions as a proportion of the number of samples that should have been marked positive, calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{67}$$

Since both Precision and Recall are both desirable yet measure different aspects, the harmonic mean of these two statistics, known as F1, is often used:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{68}$$

Precision is more desired than recall when the cost of false positives is high, such as in the detection of cancer which would cause unnecessary stress to a misdiagnosed patient. Recall is more appropriate when the cost of false negatives is high, such as in the detection of illegal content on a social media site. Hence greater importance is placed on Recall.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

5.3. Binomial model

Scikit-learn [43] software libraries were used for the traditional machine learning models. Feature extraction was done using the count vectorizer which transforms each sentence into a vector of word counts. MNB and BNB are used in both the multinomial and binomial models since the difference in the two methods is that MNB uses frequency counts of each feature while BNB uses a Boolean for each feature. The deep learning models used PyTorch [44] for implementation.

The experiments were run on Google Colab using an Nvidia Tesla P100 16GB GPU, Intel 2-core Xeon CPU at 2.00GHz, and 27GB of RAM.

The major parameter settings are listed in Table 3. A Gaussian Error Linear Unit (GELU) [45] was used for the activation for DistilBERT. The dataset is divided into training, validation, and test sets to prevent overfitting in the deep learning models.

Table 3: Binomial model parameter settings

Classifier	Parameter setting
SVM	Loss: Hinge (linear SVM)
MNB	Smoothing prior alpha: 1.0 (Laplace smoothing)
BNB	Smoothing prior alpha: 1.0
kNN	Number of neighbors: 5 Weight function: Uniform
RF	Number of trees: 100 Splitting criterion: Gini impurity
DT	Algorithm: CART Splitting criterion: Gini impurity
BiLSTM	Train Batch size: 64 Test Batch size: 64 Learning rate: 5e-4 Embedding dimension: 50 Hidden state dimension: 64 Number of stacked recurrent layers: 4 Epochs: 5 Dropout: 30% Maximum length of input: 80 Optimizer: Adam

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Table 3 (continued)

Classifier	**Parameter setting**
DistilBERT	Train Batch size: 32 Test Batch size: 64 Learning rate: 5e-5 Epochs: 1 Maximum length of input: 80 Positional embeddings: False Number of hidden layers in Transformer encoder: 6 Number of attention heads for each attention layer: 12 Dimension of encoder layers and pooler layer: 768 Hidden layer dimension: 3072 Dropout in fully connected layers in the encoder, embeddings, and pooler: 10% Dropout for attention probabilities: 10% Dropout for sequential classification model: 20% Activation: GeLU Pretrained Model: "distilbert-base-multilingual-cased" Optimizer: Adam

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

On F1 score, the DistilBERT performed best at 0.8510 while the BiLSTM was next at 0.8403, as seen in Table 4. The two deep learning models beat all of the machine learning models as predicted, with the best one being SVM at 0.7452.

Table 4: Binomial Model Results

Classifier	F1	Precision	Recall	Accuracy
BiLSTM	0.8403	0.8761	0.8074	0.9040
BNB	0.6232	0.4690	0.9282	0.6488
DistilBERT	0.8510	0.8336	0.8690	0.9048
DT	0.6477	0.6483	0.6471	0.7798
kNN	0.4901	0.7271	0.3696	0.7594
MNB	0.7029	0.6022	0.8439	0.7768
RF	0.7236	0.8344	0.6388	0.8473
SVM	0.7452	0.8098	0.6902	0.8524

For precision BiLSTM performed the best while in recall DistilBERT did best. Both had similar accuracy scores. Since abusive language detection could be regulated by the government, the cost of an abusive false negative is likely greater than that of a false positive, thereby making recall the most important statistic for this task. BNB performed the best of all the models in recall at 0.9282 but at the cost of being the worst in precision at 0.4690 since it made the most positive predictions.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

5.4. Multinomial model

Five categories will be classified by the multinomial model. While [7] also had five labels, only a binomial classifier was developed. Since a single text can have multiple labels, a prediction was considered correct if it was in the set of correct labels.

The major parameter setting changes from the binomial model are listed in Table 5.

Table 5: Multinomial model parameter changes

Classifier	Parameter setting
BiLSTM	Epochs: 2
DistilBERT	Epochs: 2 Learning rate: 5e-5 Dropout in fully connected layers in the encoder, embeddings, and pooler: 20% Dropout for attention probabilities: 20% Dropout for sequential classification model: 40%

Testing the same models as in the binomial classification task, the DistilBERT again performed best on F1 score at 0.9067 and BiLSTM was second with 0.8969, as seen in Table 6. The deep learning models outperformed the machine learning models with the closest one being SVM at 0.8090.

Table 6: Multinomial Model Results

Classifier	F1	Precision	Recall	Accuracy
BiLSTM	0.8969	0.9039	0.8900	0.8862
BNB	0.7092	0.7151	0.7034	0.6628
DistilBERT	0.9067	0.9128	0.9006	0.8965
DT	0.7383	0.7606	0.7174	0.7283
kNN	0.7163	0.7529	0.6830	0.7304
MNB	0.7786	0.7897	0.7679	0.7539
RF	0.7953	0.8240	0.7685	0.8026
SVM	0.8090	0.8339	0.7855	0.8112

Since the figurative class is the one most likely to contain texts with words that have more than a single meaning, it can better highlight the difference in the

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

effectiveness of different contextual representations between models. DistilBERT performed significantly better at 81.83% recall as seen in Table 7.

Table 7: Figurative Recall in Multinomial Model

Classifier	BiLSTM	BNB	DistilBERT	DT
Recall %	67.95	70.01	81.83	64.11

Classifier	kNN	MNB	RF	SVM
Recall %	42.39	70.75	56.13	61.45

The maximum recall rates for Rude, Figurative, Offensive, and Dirty was 91%, 82%, 72% and 87%, respectively. The Offensive category was the most difficult to predict potentially due to comprising only 18% of the total Abusive category while having a large set of possibly affiliated words in its category.

The BiLSTM did not use a pre-trained model as in [12] as the results did not improve in that study and is not considered fundamental to that particular architecture as the pre-trained model cannot be fine-tuned. DistilBERT used significantly more parameters with 135M compared to BiLSTM only using 1.7M. When positional encoding was tested, in some cases it improved performance and in others made it worse and therefore was not used.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

5.5. Discussion

For both the binomial and multinomial models, DistilBERT performed best. While parallelization is often cited as the reason to use the Transformer architecture, it is unlikely to be the reason for the consistent outperformance. Rather the multi-head self-attention mechanism appeared to be the biggest contributor.

One weakness with traditional learning models is the absence of contextual representations since one-hot vectors are used. BiLSTM is an improvement in that it does offer contextual representations. However, it has a locality bias. The nearby words will be weighted heavier than the ones that are further away.

Due to the complete reliance on self-attention mechanisms and foregoing any recurrence mechanism, DistilBERT can assign any weight to any other word in the sentence without any distance restriction.

In addition to being limited by distance, the LSTM contextual representations are also limited by the position of the words. For example, in a uni-directional LSTM, only the context words left of the target word are used for encoding. By adding a backwards LSTM and stacking multiple LSTMs, the BiLSTM can overcome this limitation somewhat. However, when encoding, the target word can “see itself” [36], as seen in Figure 23. When predicting and encoding the word “tastes” using a simple backward LSTM, the only input is the word “delicious”. When using the stacked BiLSTM, the input also includes the forward LSTM representation of “tastes” and thus the target word can “see itself”.

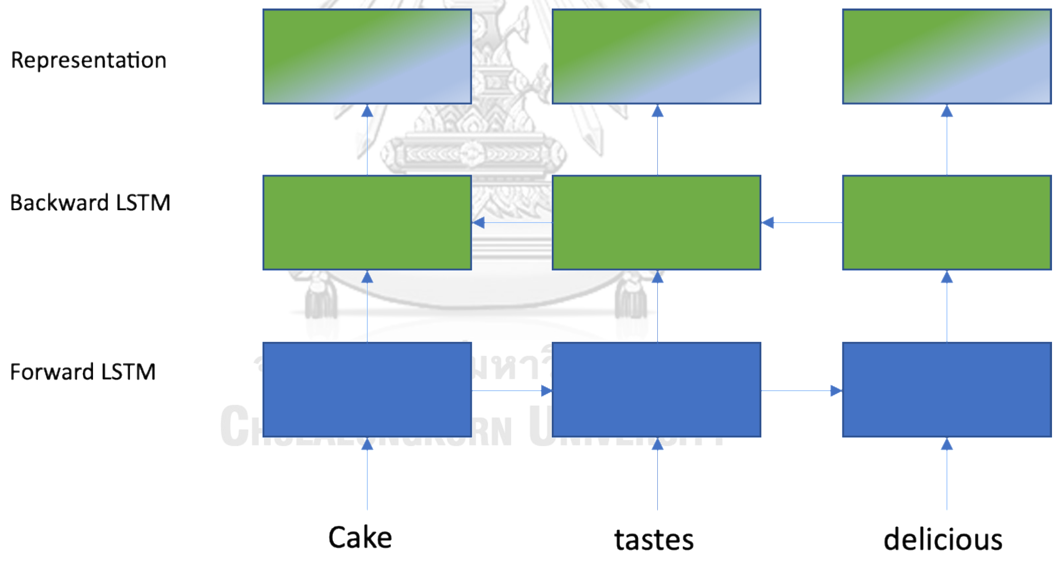


Figure 23: Stacked BiLSTM

Rather than the uni-directional approach, DistilBERT uses the MLM to randomly select words to predict. This non-directional approach has more degrees of freedom.

To highlight the effectiveness of the different models’ word representations, the Figurative class was highlighted since the type of words appearing within this category have a higher probability of having more than a single meaning. The intention and meaning of the figurative words could be inferred through better contextual representations. For example, if other abusive words appeared in the sentence, the figurative word is likely being used as slang. The multiple attention-head approach could lead to one of the attention heads specializing in figurative texts as well.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Chapter 6. Conclusion

In this thesis, we compared six traditional machine learning classifiers to two deep learning models. The machine learning classifiers include the SVM, MNB, BNB, kNN, RF, and DT and the deep learning models include variants of the RNN and Transformer architectures, the BiLSTM and DistilBERT. Both a binomial and multinomial model were developed for five categories of abusive content detection in social media that include Rude, Figurative, Dirty, Offensive and Non-Abusive. The experiments demonstrated that DistilBERT achieved the highest F1 score with 0.8510 for the binomial model and 0.9067 for the multinomial model. BiLSTM performed second best with an F1 score of 0.8403 and 0.8969 for the binomial and multinomial models, respectively. Both deep learning models outperformed the traditional machine learning classifiers' highest F1 score of 0.7452 and 0.8090 for the binomial and multinomial models, respectively. The deep learning architectures allow for better contextual representations of the words with the DistilBERT enabling better modeling of long-range dependencies between words.

For future work, a larger variant of BERT such as RoBERTa can be used. However, we believe a compressed model such as DistilBERT remains a better choice for real-time applications with many simultaneous users such as a social network due to its better speed and lower computational requirements. Also, more abusive classes can be included such as Bullying and Harassment, Violent Content, and Regulated Goods. Content from other social media platforms such as Twitter or YouTube can also be tested.

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

APPENDIX

Table 8: SVM Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	92.6%	7.4%
	Abusive	31.0%	69.0%

Table 9: MNB Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	74.6%	25.4%
	Abusive	15.6%	84.4%

Table 10: BNB Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	52.2%	47.8%
	Abusive	7.2%	92.8%

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

Table 11: kNN Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	93.7%	6.3%
	Abusive	63.0%	37.0%

Table 12: RF Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	93.9%	6.1%
	Abusive	36.1%	63.9%

Table 13: DT Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	84.0%	16.0%
	Abusive	35.3%	64.7%

Table 14: BiLSTM Binomial Confusion Matrix

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	94.8%	5.2%
	Abusive	19.3%	80.7%

Table 15: DistilBERT Binomial Confusion Matrix

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

		Predicted class	
		Non-Abusive	Abusive
Actual class	Non-Abusive	92.1%	7.9%
	Abusive	13.1%	86.9%

Table 16: SVM Multinomial Confusion Matrix

		Predicted class				
		Rude	Figurative	Offensive	Dirty	Non-Abusive
Actual class	Rude	57.1%	1.1%	2.2%	1.1%	38.5%
	Figurative	4.9%	61.5%	3.3%	1.0%	29.4%
	Offensive	9.6%	1.4%	63.9%	0.4%	24.6%
	Dirty	2.5%	0.5%	2.2%	45.8%	49.0%
	Non-Abusive	4.6%	0.6%	0.2%	0.1%	94.6%

Table 17: MNB Multinomial Confusion Matrix

		Predicted class				
		Rude	Figurative	Offensive	Dirty	Non-Abusive
Actual class	Rude	80.8%	0.5%	0.1%	0.2%	18.4%
	Figurative	18.8%	70.8%	0.0%	0.3%	10.2%
	Offensive	21.8%	0.4%	70.4%	0.0%	7.5%
	Dirty	8.2%	0.0%	0.0%	64.7%	27.1%
	Non-Abusive	20.9%	0.4%	0.1%	0.4%	78.2%

Table 18: BNB Multinomial Confusion Matrix

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

		Predicted class				
		Rude	Figurative	Offensive	Dirty	Non-Abusive
Actual class	Rude	88.8%	0.0%	0.6%	0.1%	10.6%
	Figurative	23.8%	70.0%	0.6%	0.0%	5.6%
	Offensive	25.7%	0.0%	69.6%	0.0%	4.6%
	Dirty	10.2%	0.0%	0.0%	78.6%	11.2%
	Non-Abusive	36.5%	0.1%	0.8%	0.1%	62.5%

Table 19: kNN Multinomial Confusion Matrix

Actual class \ Predicted class	Rude	Figurative	Offensive	Dirty	Non-Abusive
Rude	30.3%	2.6%	0.6%	2.2%	64.3%
Figurative	2.4%	42.4%	0.6%	1.9%	52.7%
Offensive	3.9%	2.5%	39.6%	2.9%	51.1%
Dirty	0.5%	1.5%	0.0%	37.1%	61.0%
Non-Abusive	3.0%	1.6%	0.2%	2.1%	93.1%

Table 20: RF Multinomial Confusion Matrix

Actual class \ Predicted class	Rude	Figurative	Offensive	Dirty	Non-Abusive
Rude	48.4%	1.1%	0.1%	0.1%	50.2%
Figurative	5.3%	56.1%	0.2%	0.0%	38.4%
Offensive	8.9%	1.8%	50.7%	0.0%	38.6%
Dirty	2.2%	0.3%	0.0%	48.0%	49.5%
Non-Abusive	2.3%	0.8%	0.0%	0.2%	96.7%

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Table 21: DT Multinomial Confusion Matrix

Actual class \ Predicted class	Rude	Figurative	Offensive	Dirty	Non-Abusive
Rude	49.9%	6.9%	2.8%	6.2%	34.3%
Figurative	3.4%	64.1%	2.1%	5.2%	25.3%
Offensive	4.3%	5.4%	59.3%	6.1%	25.0%
Dirty	1.5%	4.5%	1.2%	61.4%	31.3%
Non-Abusive	6.5%	3.6%	1.0%	5.5%	83.6%

Table 22: BiLSTM Multinomial Confusion Matrix

		Predicted class				
		Rude	Figurative	Offensive	Dirty	Non-Abusive
Actual class	Rude	90.5%	0.0%	0.0%	0.0%	9.5%
	Figurative	19.2%	68.0%	0.0%	0.0%	12.9%
	Offensive	24.3%	0.0%	65.7%	0.0%	10.0%
	Dirty	7.0%	0.0%	0.0%	85.1%	8.0%
	Non-Abusive	5.5%	0.0%	0.0%	0.0%	94.5%

Table 23: DistilBERT Multinomial Confusion Matrix

		Predicted class				
		Rude	Figurative	Offensive	Dirty	Non-Abusive
Actual class	Rude	85.5%	2.8%	0.1%	0.1%	11.5%
	Figurative	4.0%	81.8%	0.0%	0.0%	14.2%
	Offensive	9.3%	3.2%	71.8%	0.0%	15.7%
	Dirty	4.5%	0.8%	0.0%	87.1%	7.7%
	Non-Abusive	4.3%	0.4%	0.0%	0.3%	95.1%

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

REFERENCES

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

[1] M. Mondal, L. A. Silva, and F. Benevenuto, “A Measurement Study of Hate Speech in Social Media,” presented at the Proceedings of the 28th ACM Conference on Hypertext and Social Media, Prague, Czech Republic, 2017. [Online]. Available: https://doi.org/10.1145/3078714.3078723.

[2] U.S. House Committee on Energy & Commerce. 117th Congress, (2021). Hearing on “Disinformation nation: social media’s role in promoting extremism and misinformation”. [Online] Available: https://energycommerce.house.gov/committee-activity/hearings/hearing-on-disinformation-nation-social-medias-role-in-promoting

[3] “Facebook users in Thailand.” NapoleonCat. https://napoleoncat.com/stats/facebook-users-in-thailand/2021/02 (accessed April 19, 2021).

[4] M. Schroepfer. “Update on our progress on AI and hate speech detection.” https://about.fb.com/news/2021/02/update-on-our-progress-on-ai-and-hate-speech-detection/ (accessed April 14, 2021).

[5] A. Vaswani et al., “Attention is all you need,” presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA, 2017.

[6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” ArXiv, vol. abs/1910.01108, 2019.

[7] S. Tuarob and J. L. Mitrpanont, “Automatic discovery of abusive Thai language usages in social networks,” in Proceedings of the 19th International Conference on Asia-Pacific Digital Libraries (ICADL), Bangkok, Thailand, in Lecture Notes in Computer Science, vol. 10647, S. Choemprayong, F. Crestani, and S. J. Cunningham, Eds., Springer, 2017, pp. 267-278.

[8] D. Q. Nguyen and A. Tuan Nguyen, “PhoBERT: Pre-trained language models for Vietnamese,” in Proceedings of the, Online, in Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, nov 2020, pp. 1037-1042.

[9] Q. H. Pham, V. Anh Nguyen, L. B. Doan, N. N. Tran, and T. M. Thanh, “From universal language model to downstream task: improving RoBERTa-based

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Vietnamese hate speech detection," in Proceedings of the 2020 12th International Conference on Knowledge and Systems Engineering (KSE), Can Tho, Vietnam, 2020, pp. 37-42.

[10] O. M. Singh, S. Timilsina, B. K. Bal, and A. Joshi, "Aspect based abusive sentiment detection in Nepali social media texts," in Proceedings of the 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), The Hague, Netherlands, 2020, pp. 301-308.

[11] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter," in Proceedings of the, Florence, Italy, in Proceedings of the Third Workshop on Abusive Language Online, Association for Computational Linguistics, Aug 2019, pp. 46-57.

[12] R. Hendrawan, Adiwijaya, and S. Al Faraby, "Multilabel classification of hate speech and abusive words on Indonesian Twitter social media," in Proceedings of the 2020 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, August 2020, pp. 1-7.

[13] J. Chiu, E. Grave, and A. Joulin. "Building an efficient neural language model over a billion words." https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/ (accessed April 15, 2021).

[14] K. Pasupa and T. Seneewong Na Ayutthaya, "Hybrid Deep Learning Models for Thai Sentiment Analysis," Cognitive Computation, 2021/03/04 2021, doi: 10.1007/s12559-020-09770-0.

[15] P. Vateekul and T. Koomsubha, "A study of sentiment analysis using deep learning techniques on Thai Twitter data," in Proceedings of the 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), 13-15 July 2016 2016, pp. 1-6.

[16] S. Sangsavate, S. Tanthanongsakkun, and S. Sinthupinyo, "Stock Market Sentiment Classification from FinTech News," in Proceedings of the 2019 17th International Conference on ICT and Knowledge Engineering (ICT&KE), 20-22 Nov. 2019 2019, pp. 1-4.

[17] R. Arreerard and T. Senivongse, "Thai Defamatory Text Classification on Social Media," in Proceedings of the 2018 IEEE International Conference on Big Data,

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

Cloud Computing, Data Science & Engineering (BCD), 12-13 July 2018 2018, pp. 73-78.

[18] V. Metsis, I. Androutsopoulos, and G. Paliouras, “Spam Filtering with Naive Bayes - Which Naive Bayes?,” in Proceedings of the CEAS, 2008,

[19] K.-M. Schneider, “A comparison of event models for Naive Bayes anti-spam e-mail filtering,” presented at the Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1, Budapest, Hungary, 2003. [Online]. Available: https://doi.org/10.3115/1067807.1067848.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Wadsworth, 1984.

[21] J. R. Quinlan, C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., 1993.

[22] C. Apichai, K. Chan, and Y. Suzuki, “Classification of Thai Tweets: Mining Treasures from Tweet Heap,” in Proceedings of the 2018 5th International Conference on Systems and Informatics (ICSAI), 10-12 Nov. 2018 2018, pp. 311-315.

[23] C. Piyaphakdeesakun, N. Facundes, and J. Polvichai, “Thai Comments Sentiment Analysis on Social Networks with Deep Learning Approach,” in Proceedings of the 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), 23-26 June 2019 2019, pp. 1-4.

[24] P. Thiengburanathum and P. Charoenkwan, “A Performance Comparison of Supervised Classifiers and Deep-learning Approaches for Predicting Toxicity in Thai Tweets,” in Proceedings of the 2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, 3-6 March 2021 2021, pp. 238-242.

[25] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2012arXiv1206.5538B

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

[26] M. Minsky and S. A. Papert, Perceptrons: An Introduction to Computational Geometry. The MIT Press, 2017.

[27] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303-314, 1989/12/01 1989, doi: 10.1007/BF02551274.

[28] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” Neural Netw., vol. 4, no. 2, pp. 251–257, 1991, doi: 10.1016/0893-6080(91)90009-t.

[29] Z. S. Harris, “Distributional Structure,” WORD, vol. 10, no. 2-3, pp. 146-162, 1954/08/01 1954, doi: 10.1080/00437956.1954.11659520.

[30] T. Mikolov, W.-t. Yih, and G. Zweig, “Linguistic Regularities in Continuous Space Word Representations,” in Proceedings of the, Atlanta, Georgia, in Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, jun 2013, pp. 746-751.

[31] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, March 1994.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2014arXiv1409.3215S

[33] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv151203385H

[34] J. Lei Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2016arXiv160706450L

[35] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150203167I

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” CoRR, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805.

[37] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190711692L

[38] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, “Model compression,” presented at the Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, Philadelphia, PA, USA, 2006. [Online]. Available: https://doi.org/10.1145/1150402.1150464.

[39] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150302531H

[40] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, “Well-Read Students Learn Better: On the Importance of Pre-training Compact Models.” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190808962T

[41] A. Suriyawongkul, E. Chuangsuwanich, P. Chormai, and C. Polpanumas. PyThaiNLP/wisesight-sentiment: First release, September 22, 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3457447

[42] PyThaiNLP/pythainlp: PyThaiNLP 2.2.3. (2020). [Online]. Available: https://doi.org/10.5281/zenodo.3969544

[43] F. Pedregosa et al., “Scikit-learn: machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[44] A. Paszke et al., “PyTorch: an imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett Eds., Curran Associates, Inc., 2019, pp. 8024-8035.

[45] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs).” [Online]. Available: https://ui.adsabs.harvard.edu/abs/2016arXiv160608415H

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

VITA

NAME	Ruangsung Wanasukapunt
PLACE OF BIRTH	Illinois, USA
INSTITUTIONS ATTENDED	- Chulalongkorn University, Thailand
	- University of Chicago, USA
	- University of California at Berkeley, USA
HOME ADDRESS	Sky Villas at The Ascott Sathorn, Unit 2102
	187 South Sathorn Road
	ช่องนนทรี ยานนาวา กรุงเทพมหานคร 10120

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University

จุฬาลงกรณ์มหาวิทยาลัย
Chulalongkorn University