MULTI-LABEL CLASSIFICATION FOR LOW-RESOURCE ISSUE TICKETS WITH BERT

BY

CHAWANEE SUNGTHONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE (COMPUTER SCIENCE)
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCE AND TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2023

THAMMASAT UNIVERSITY

FACULTY OF SCIENCE AND TECHNOLOGY

THESIS

BY

CHAWANEE SUNGTHONG

ENTITLED

MULTI-LABEL CLASSIFICATION FOR LOW-RESOURCE ISSUE TICKETS WITH BERT

was approved as partial fulfillment of the requirements for
the degree of Master of Science (Computer Science)

on June 5, 2024

Chairman [illegible]
(Assistant Professor Denduang Pradubsuwun, D.Eng.)

Member and Advisor [illegible]
(Assistant Professor Pokpong Songmuang, Ph.D.)

Member P. Leesutthipornchai
(Assistant Professor Pakorn Leesutthipornchai, Ph.D.)

Member [illegible]
(Prachya Boonkwan, Ph.D.)

Dean [illegible]
(Associate Professor Supet Jirakajohnkool, Ph.D.)

Thesis Title	MULTI-LABEL CLASSIFICATION FOR LOW-RESOURCE ISSUE TICKETS WITH BERT
Author	Chawanee Sungthong
Degree	Master of Science (Computer Science)
Department /Faculty/University	Computer Science Faculty of Science and Technology Thammasat University
Thesis Advisor	Assistant Professor Pokpong Songmuang, Ph.D.
Academic Year	2023

ABSTRACT

Nowadays, many companies, especially in IT, use project management tools to manage and track ongoing projects. Issue tickets within the tools can be assigned to any team member with details about the task. However, failing to label tickets with task categories affects progress monitoring. The research addresses the multi-label classification problem and the challenge of low-resource dataset. The dataset in this study is different from the large datasets usually used in current multi-label classification studies. Therefore, this research proposes embedding techniques to handle this situation. The methodology in this study applies several embedding techniques, including TF-IDF, Word2Vec, and BERT (WangchanBERTa and PhayaThaiBERT). The classification models, such as Logistic Regression, Support Vector Classifiers, Decision Trees, Neural Networks, and Deep Neural Networks, are experimented in this study to assess their performance. According to the experimental results, a combination of PhayaThaiBERT and Deep Neural Networks outperforms conventional methods. The combination yielded an F1 score of 0.769 on the test dataset.

Keywords: multi-label classification, issue tickets, low-resource dataset, text classification, text embedding, Transformer, BERT

ACKNOWLEDGEMENTS

Many obstacles and moments of doubt have been my companions on the path to finishing this thesis. ”You turn if you want to. The Lady’s not for turning.”—Margaret Thatcher’s words—gave me confidence in the face of doubt.

I would like to start by expressing gratitude to my adviser, Asst. Prof. Dr. Pokpong Songmuang. Your patience and effort helped shape this work, and I am grateful.

I am forever grateful to my late father, who is now in heaven, for giving me the strength to follow my heart and follow my ambitions. No matter how far away you are, your confidence in me gives me the will to make it happen.

I also want to express my deep thanks to my mother, who worked tirelessly to support the family after my father passed away. My motivation and strength have come from your sacrifice and hard work along this journey.

My partner, Harry, I am always grateful for your continuous support during all of the academic storms. Your belief in me has given me more confidence. Thank you so much.

Dear friends, I am grateful for your encouragement. I couldn’t have asked for a better.

Also, kudos to myself for all my dedication and hard work that made this possible. Despite everything that I went through, I never give up to become a better person. Cheers to resilience.

Finally, I would like to thank to the members of the commitee for their time as examiners for this thesis: Asst. Prof. Dr. Denduang Pradubsuwun, Asst. Prof. Dr. Pakorn Leesutthipornchai, and Dr. Prachya Boonkwan.

Last but not least, I want to express my gratitude to everyone who has contributed, no matter how big or small, to this work. This thesis would not have been possible without your faith in me.

Chawanee Sungthong

TABLE OF CONTENTS

	Page
ABSTRACT	(1)
ACKNOWLEDGEMENTS	(2)
LIST OF TABLES	(6)
LIST OF FIGURES	(7)
CHAPTER 1 INTRODUCTION	1
1.1 Background and Rationale	1
1.2 Purposes of this study	3
1.3 Limitations	3
1.4 Expected Benefits	4
CHAPTER 2 REVIEW OF LITERATURE	5
2.1 Challenges in Multi-label Classification with Low-Resource Datasets	5
2.2 Text Vectorization	6
2.2.1 Baseline: TF-IDF	6
2.2.2 RNN: Word2Vec	7
2.2.3 Transformer: BERT	8
2.3 Classification models	9
2.3.1 Logistic Regression (LR)	9
2.3.2 Support Vector Classifier (SVC)	9
2.3.3 Decision Tree (DT)	9
2.3.4 Neural Networks (NN)	10
2.3.5 Deep Neural Networks (DNN)	10

CHAPTER 3 RESEARCH METHODOLOGY	11
3.1 Dataset	11
3.2 The Workflow of the Research Methodology	12
3.2.1 Preprocessing	12
3.2.1.1 Noise Cleaning/ Special Characters	13
3.2.1.2 Word Tokenization and Lowercase Conversion	13
3.2.2 Modelling	13
3.2.3 Evaluation	14
3.2.3.1 3-Fold Cross-Validation	14
3.2.3.2 Evaluation Metrics	15
CHAPTER 4 RESULTS AND DISCUSSION	17
4.1 Experiment Settings	17
4.1.1 Configurations Parameters for Embedding Models	18
4.1.1.1 TF-IDF	18
4.1.1.2 WangchanBERTa	18
4.1.1.3 PhayaThaiBERT	19
4.1.2 Configurations Parameters for Classification Models	19
4.1.2.1 Logistic Regression	19
4.1.2.2 Decision Tree	20
4.1.2.3 Support Vector Classifier	20
4.1.2.4 Neural Networks	20
4.1.2.5 Deep Neural Networks	21
4.2 Experiment Results	22
CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS	25
5.1 Conclusion	25
5.2 Recommendations	28
REFERENCES	29

BIOGRAPHY 32

LIST OF TABLES

Tables	Page
3.1 Sample Dataset of Issue Tickets	11
4.1 F1 Score of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)	23
4.2 Precision of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)	23
4.3 Recall of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)	24
5.1 Training Time of Classification Models with Different Embedding Models	27

LIST OF FIGURES

Figures Page

3.1 Numbers of Ticket for Each Tag in the Dataset 12

3.2 The Workflow of the Research Methodology for Multi-label Classification of Issue Tickets 12

3.3 Example of the Multi-Label Classification with PhayaThaiBERT 14

5.1 Comparison of Machine Learning Models' Performances on a Multi-Label Classification Task, Using PhayaThaiBERT Embedding 26

CHAPTER 1
INTRODUCTION

1.1 Background and Rationale

In the present day, many companies, particularly those in the Information Technology industry, use project management tools to manage and monitor the progress of ongoing projects. These tools are important for task organization, milestones tracking, and project timeline management. A key component of the tools is the issue ticket, which is a task that a user can assign to any team member. Each issue ticket includes specific details about the work being allocated, allowing for clear communication and delegation of responsibilities.

One of the advantages of using issue tickets in project management is that they provide a comprehensive project overview that is accessible to all users, ranging from team members to stakeholders, through the use of dashboards. These dashboards summarize information such as tasks, statuses, assignees, and task categories, providing an overall picture of the project's progress. This overview allows companies to make suitable decisions based on real-time data about the project's status. Using issue tickets for project management can improve productivity, ensure timely project delivery, and facilitate collaboration among teams.

However, problems arise when users fail to label the ticket with the appropriate category of task. When tickets are not labelled, it can lead to considerable issues. Project managers and stakeholders are unable to keep track on the project's progress when labels are missing. These labels are used to create dashboards and reports that show the current status of the project. Lack of labels can result in a misleading report, which may present a wrong picture of the project's progress.

Furthermore, it is important to note that the issue tickets can have multiple labels, making the problem at this point a multi-label classification problem. Multi-label classification requires that each ticket be classified into more than one labels, as opposed to single-label classification that has only one label per ticket. Accurate labelling requires a thorough understanding of the specific details of each task.

Given the users' lack of expertise in all areas of work, a challenge arises when users in charge of issuing tickets have to assign tasks that can be classified into multiple categories. The process requires extra work and time, as users have to manually check and assign the labels to each ticket. Occasionally, users may experience difficulty in selecting appropriate labels and may accidentally skip labelling particular topics. This can lead to incomplete labelling, impacting the team's performance. As a consequence, the actual progress of the team may be underreported.

This study proposes a text classification approach that uses word embedding techniques to address the challenges of low-resource datasets and multi-label classification. Recent studies in multi-label classification mostly rely on large datasets containing over 50,000 samples (Polpanumas, C., & Suwansri, P., 2019; Venkatesan, R., & Er, M. J., 2014). However, the dataset used in this study is significantly limited in resources, containing under 1,000 samples. As a result, the use of approaches that have been successful with large datasets is not suitable for this study due to small amount of data. (Karl, F., & Scherp, A., 2023; Shleifer, S. , 2019).

In response to this challenge, this study uses embedding techniques to improve the performance of classification models under low-resource dataset. This study uses conventional embedding techniques such as TF-IDF and Word2Vec to transform textual data into numerical representations. In addition to these conventional methods, this study conducts experiments using the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT) variants, namely WangchanBERTa (Lowphansirikul, L.et al., 2021) and PhayaThaiBERT (Sriwirote, P. et al., 2023).

This study also experiments with different classification models. These include conventional machine learning models such as Logistic Regression and Support Vector Classifiers. Moreover, this study experiments Decision Trees as well as more complex models including Neural Networks, and Deep Neural Networks.

By comparing the combination of embedding methods and classification models, this study aims to find the optimal approach for multi-label classification in a low-resource environment. The purpose of this study is to determine the combinations that result in the highest accuracy when applied to dataset with limited number of samples.

1.2 Purposes of this study

1) To study on the topic of multi-label classification for issue tickets.
2) To study and compare different embedding techniques in a low-resource dataset between conventional embedding techniques such as TF-IDF and Word2Vec and the BERT variants, namely WangchanBERTa and PhayaThaiBERT.
3) To study and compare classification models, including Logistic Regression, Support Vector Classifiers, Decision Trees, Neural Networks, and Deep Neural Networks.
4) To determine the combinations that result in the highest accuracy when applied to a low-resource dataset.

1.3 Limitations

1) The issue ticket dataset used in this study is only available in the context of Applied Machine Learning team.
2) Each issue ticket can only have up to three labels.
3) The dataset contains only internal company data.

1.4 Expected Benefits

1) The approach in this study can demonstrate robust performance in predicting issue ticket labels in a multi-label classification problem.

2) The approach can reduce the time for issue ticket labelling by humans.

CHAPTER 2
REVIEW OF LITERATURE

This paper focuses on multi-label classification in the context of issue tickets. This chapter discusses multi-label classification and different techniques of text vectorization and classification model.

2.1 Challenges in Multi-label Classification with Low-Resource Datasets

Multi-label classification is a type of machine learning task in which each instance can be given more than one label at the same time from a list of possible labels. This task is for applications such as text classification (Zhang, M.-L., & Zhou, Z.-H., 2014). In traditional single-label classification, each instance is classified into only one class. Multi-label classification, on the other hand, can handle situations where instances have attributes that are complex, and can be represented by more than one label.

However, low-resource datasets usually have limited labelled instances, which causing the difficulty to train complex models. Because of this, models might overfit the existing data or produce models that have poor generalization when given new data. Conventional multi-label classification methods require a large quantity of data to train models that can capture multi-label relationships. When working with low-resource dataset, conventional methods of multi-label classification can underperform; therefore, robust model training depends on large, labelled datasets (Demsar, J., 2006).

To solve these problems, both zero-shot learning and few-shot learning are supervised approaches that are designed to deal with low-resource settings. Zero-shot learning is a machine learning method that allows pre-trained models to predict unseen classes from the training data by learning semantic embeddings or attributes from labelled data as information to predict unseen classes (Xian et al., 2018).The model can classify the unseen data by measuring the similarity between embeddings. Few-shot learning is also a method that only use a minimal number of labelled instances per each class (Snell et al., 2017).

Although zero-shot and few-shot learning provide novel approaches for multi-label classification in low-resource environments, their implementation in this study may be computationally expensive and impractical. Due to the lack of labelled instances in low-resource datasets, the substantial resources needed to train and implement such models may surpass their potential advantages. Thus, while zero-shot and few-shot learning have the potential to tackle the challenges of conventional multi-label classification, their high computational needs make them less appropriate for this research.

2.2 Text Vectorization

In order to interpret text, Natural Language Processing (NLP) must be able to distinguish words, syntax, and other linguistic characteristics. This task is difficult for computers because they can only process numbers. To fix this, text vectorization is developed, which convert words into numerical representations. text vectorization work by tokenizing each word in a sentence and transforming it to a vector space as real-valued numerical vectors. This technique attempts to capture the semantic meaning of words by assigning numerical numbers to words that have similar meanings. Text classification often relies on foundation approaches such as TF-IDF and Logistics Regression (Liu, Z. et al., 2023).

2.2.1 Baseline: TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF), a simple yet effective method for capturing the frequency of terms and inversely weighing the relevance of documents in order to generate word embeddings (Wandee, W., & Songmuang, P., 2022; Songmuang, P., et al. , 2021). Term Frequency (TF) measures the frequency of words in a document. It counts the frequency of each word and calculate the TF by dividing the number of frequencies of a certain word (*i*) by the total number of words (*N*) in the document (*j*).

$$\text{TF(i)} = \frac{\log(frequency(i,j))}{\log(N(j))} \tag{2.1}$$

Inverse Document Frequency (IDF) measures the infrequency of terms. TF gives more weight to frequently occurring words but words that appear less often in the document can still be important. IDF deals with this by figuring out the significance of rare words. IDF is calculated by dividing the total number of document (*d*) by the number of documents that contain the word (*t*)

$$\text{IDF(t)} = \log\left(\frac{N(d)}{frequency(d,t)}\right) \tag{2.2}$$

Logarithms are used to lessen the effect of high numbers for TF and IDF. The TF and IDF values are then multiplied to get the final TF-IDF value. Due to its simplicity and computational efficiency, this research uses this technique to work with low-resource dataset.

2.2.2 RNN: Word2Vec

Word2Vec (Mikolov, T. et al., 2013) is a widely used word embedding technique that represents words as dense vectors, capturing semantic relationships and enabling algorithms to understand contextual meanings in Natural Language Processing. It uses cosine similarity metric to find words that are related. A cosine angle of 1 means that two words are overlapping, whereas a cosine angle of 90 degrees means that two words are separate and do not belong in the same context. Word2Vec is effective in capturing semantic relationship between words. For example, it can capture that ”cat” has a similar meaning to ”kitten”. This technique achieves the best performance in prachathai-67k benchmark (Polpanumas, C., & Suwansri, P., 2019), proving it as a reliable candidate for text vectorization.

Polpanumas C. and Suwansri P. (2019), perform a benchmark on Thai-English text classification with datasets prachathai67k, specifically focusing on a 12-topic multi-label classification task using SVC as the classifier. Additionally, they deal with different embedding models, specifically, TF-IDF, fastText (Joulin, A. et al., 2017), ULMFit (Polpanumas, C., & Phatthiyaphaibun, W., 2021), and Universal Sentence Encoder (Cer, D., et al. , 2018).

Karl F. and Scherp A. (2023) indicates that state-of-the-art methods for multi-label classification primarily rely on Transformers-based architectures. This study also noted that the Transformer model achieved state-of-the-art accuracy on two newly introduced real-world datasets: NICE-2 and STOPS-2.

2.2.3 Transformer: BERT

Bidirectional Encoder Representations from Transformers or BERT (Devlin, J. et al., 2019) uses unlabelled large text to pretrain deep bidirectional representations. BERT revolutionized word embedding by using attention mechanism to create top-notch context-aware embeddings. During training, embeddings are refined over multiple iterations of BERT's encoder layers. The attention mechanism captures word associations based on surrounding words, both before and after each word. Positional encoding is also used in word embeddings to keep position of word in a sentence. The BERT model can be fine-tuned by adding a single output layer to create models for various applications. Based on their research results, increasing the size of the model significantly leads to significant improvements in extremely small-scale tasks, as long as the model has been given sufficient pre-training. This is particularly useful in situations with a lack of labelled data, when a pre-trained large model can still achieve promising results. However, BERT is not specifically trained for Thai language.

WangChanBERTa (Lowphansirikul, L.et al., 2021) is the first Thai pre-training BERT model based on a Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu, Y., et al. , 2019), which improves the early BERT by training on larger amount of data for a longer time with increased batch sizes. ROBERTa also removes the next sentence prediction task and modifies the static masking pattern to a dynamic one. WangChanBERTa still have difficulty in understanding loan words from foreign languages.

PhayaThaiBERT (Sriwirote, P. et al., 2023), also based on RoBERTa, is an improved pre-trained Thai language model by including unassimilated loanwords that appear often in the dataset in this study and a two-times larger dataset than WangChanBERTa. With this improvement, the model can represent modern Thai words, especially the use of foreign words that are still not completely assimilated.

2.3 Classification models

Classification is a technique in Natural Language Processing that classify texts into one or more categories.

2.3.1 Logistic Regression (LR)

A foundational and reliable baseline linear classification model (Wandee, W., & Songmuang, P., 2022; Shah, K. et al., 2020) that efficiently distinguishing categories by probabilities from a sigmoid function with simplicity and interpretability in various classification tasks. The sigmoid function, which is also referred as the logistic function, serves as the foundation of logistic regression. This function converts any integer input into a value that falls within the range of 0 to 1. With its simplicity, this research uses this model as a benchmark against more complex models.

2.3.2 Support Vector Classifier (SVC)

Known for its efficacy in high-dimensional spaces. A powerful and adaptable machine learning algorithm used for classification tasks especially non-linear input spaces. Its main goal is to find the optimal hyperplane in an N-dimensional space so that data points can be easily put into different classes within the feature space. This hyperplane tried to maximize the margin between the closest points of different classes. Although SVC is mainly used for binary classification, it can be modified for multi-label classification using One-vs-One and One-vs-Rest techniques. It is suitable for this research due to its robustness to overfitting problems that are likely to occur in a low-resource dataset in this study. Also, SVC is used as a comparison model for prachathai-67k benchmark (Polpanumas, C., & Suwansri, P., 2019)

2.3.3 Decision Tree (DT)

This model classifies the data by splitting it according to feature values, using a tree-like model with a root node, branches, internal nodes, and leaf nodes. The decision tree starts with a root node. Internal nodes, which are also called decision nodes, are reached by branches that go out from the root node. The purpose of decision nodes is to assess the features in order to divide the data into leaf nodes. Each leaf node contains all possible results within the dataset. Decision Trees can help reduce dimensions of complex data and can also handle non-linear data (Rahman, M. A., & Akter, Y. A. , 2019).

Tsoumakas G. and Katakis I. (2007) indicate that, multi-label classification can be done in two-ways: Firstly, one option is to use a classical model with a "one-vs-all" approach, which might result in excessive demands on computational resources. Secondly, the approach is to use multilayer perceptron (MLP) classifier as a unified, scalable model to handle the large number of labels in the dataset in this study.

2.3.4 Neural Networks (NN)

A neural network is a type of machine learning that attempts to imitate how the brain makes decisions. In a neural network, artificial neurons employ layered, interconnected nodes to assess and anticipate complex patterns in data. (LeCun, Y. et al., 2015). It usually has an input layer, one or more hidden layers, and an output layer. Every node is connected to every other node and has its own weight and threshold, which are changed iteratively through the process called backpropagation, to minimize the difference between predicted and actual outputs. This learning process enables the model to learn from an input data and predict the unseen data.

2.3.5 Deep Neural Networks (DNN)

Deep Neural Networks (DNN) are an improvement over conventional Neural Network because they have more hidden layers. This increased depth helps DNNs find complex patterns and relationships within the data, which leads to more accurate prediction and do not need the use of "one-vs-all" techniques (Tsoumakas, G., & Katakis, I. , 2007; Nam, J. et al., 2014).

Most of machine learning techniques, as well as the previously mentioned approaches, require a large dataset. Nevertheless, as previously stated, the challenge of this study is the low-resource dataset. BERT enables a model to achieve superior performance on such datasets. Therefore, this research experiments with combinations of text vectorization and classification models.

CHAPTER 3
RESEARCH METHODOLOGY

This chapter discusses the research methodology, starting from dataset, pre-processing, modelling, and evaluation. The details are as follows:

3.1 Dataset

The dataset consists of 940 records that include title, descriptions, and labels, as shown in Table 3.1. The title and description contain both Thai and English words, as well as unassimilated loanwords. The content is made up of short instructions. The dataset is split into 20% test set and remainder is for training with a 3-fold cross validation approach.

Table 3.1 Sample Dataset of Issue Tickets

Title	Description	Label
หา resource และ tools	ค้นหา resources ที่จำเป็น เช่น ที่จำเป็นสำหรับการพัฒนาและปรับปรุงโมเดล	- literature review
Create Image Transform pipeline	DOD: ได้ตาราง run details ใน datastore - ใช้ function location ในการสร้าง column id - partition_cols: [’year’, ’month’, ’day’], - mode = overwrite_partition	- data pipeline
ทดลอง data augmentation เพื่อเพิ่มประสิทธิภาพโมเดล ในการเทรนโมเดล	ประยุกต์ใช้เทคนิคต่างๆ เช่น flip หรือ crop เพื่อเพิ่มความหลากหลาย ของข้อมูล เพื่อเพิ่มประสิทธิภาพในการเทรนโมเดล	- data processing - model training - literature review

There are 30 labels that show a significant imbalance in class as shown in Figure 3.1. The highest number of ticket tags is “data processing” (179 tickets) followed by “model training” (144 tickets) and “model evaluation” (139 tickets). The lowest number of ticket tags is “requirement gathering” (28 tickets), followed by “design data architecture” (29 tickets) and “model pipeline” (30 tickets).

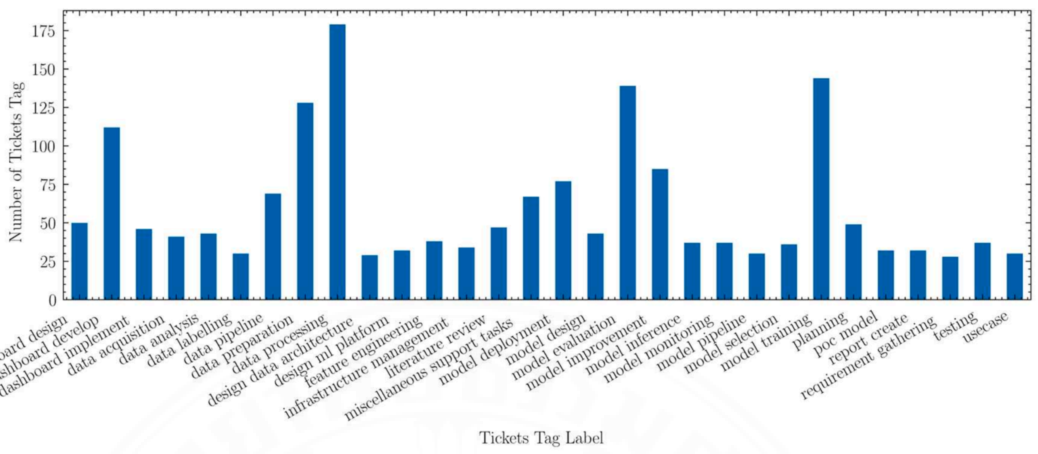

Figure 3.1 Numbers of Ticket for Each Tag in the Dataset

3.2 The Workflow of the Research Methodology

The process of the research methodology is divided into three phases: Pre-Processing, Modeling, and Evaluation. The workflow of research methodology for multi-label classification of issue tickets is shown in Figure 3.2.

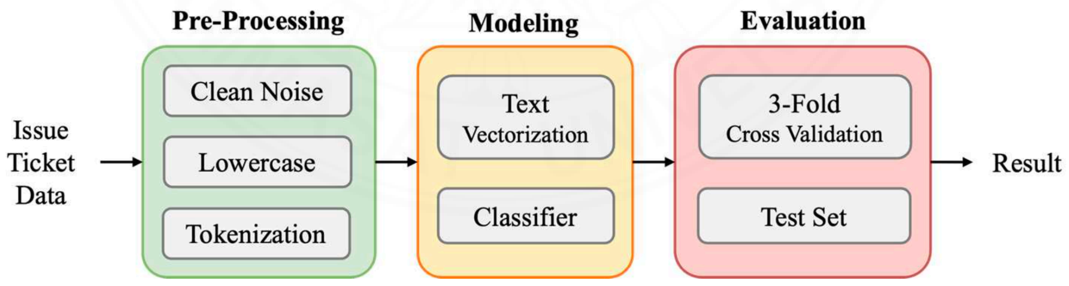

Figure 3.2 The Workflow of the Research Methodology for Multi-label Classification of Issue Tickets

3.2.1 Preprocessing

In the preprocessing stage, the following techniques are used to eliminate noisy and insignificant data from the dataset prior to feeding it into the model.

3.2.1.1 Noise Cleaning/ Special Characters

In the initial step of data preprocessing, several tasks are done to prepare the data. First, single characters are removed as they do not mean anything significant. Special characters and excessive spaces are also removed. In addition, the issue tickets' titles and descriptions are concatenated with newlines into a single column to order to simplify preprocessing step.

3.2.1.2 Word Tokenization and Lowercase Conversion

The second step in the preprocessing stage is word tokenization and normalization. In this step, all texts are split into single words, transforming each sentence or phrase into a list. Word tokenization allows text to be easier processed.

After tokenization is completed, all the characters in the word list are converted to lowercase. For example, words like "CAT", "cat", "Cat" will all be changed to "cat", to ensure that all words are treated the same.

Tokenization and lowercase conversion are performed to prepare the data for the next steps. This ensures that the dataset is in a format that is ready for word vectorization and classification processes.

3.2.2 Modelling

To tackle the issue of low resource datasets, the pre-trained BERT model named PhayaThaiBERT that pre-trained on unassimilated loanwords is used. The comprehensive diagram of this approach is depicted in Figure 3.3.

The input sequence as tokens is fed into the Transformer block, which functions as the underlying model for PhayaThaiBERT. The tokens in Figure 3.3. are "[CLS]", "[ทดลอง]", "[data]", "[augmentation]", and "[SEP]". The "[CLS]"token (stands for classification) implies that it is intended for use in classification task, whereas the separate token or "[SEP]" is used to indicate the separation of the input sequence.

Each Transformer block contains multi-head self-attention mechanism, allowing them to measure the importance of each word according to its context and to capture the relationships between words in the sentence. In this stage, each word is given a specific vector that contains different values to represent its information. After the last Transformer block, the sequence of vectors is transformed into a single vector representation and is fed into the fully connected layer for classification task using DNN.

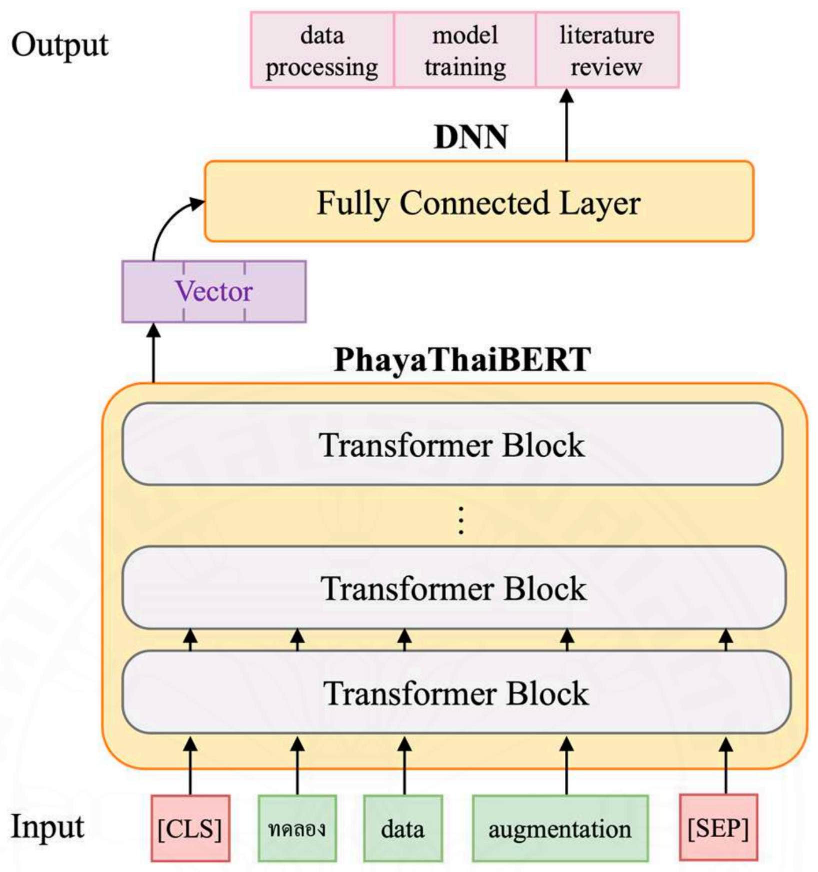

Figure 3.3 Example of the Multi-Label Classification with Pre-trained PhayaThaiBERT

DNNs are referred to as “deep” because there are many hidden fully connected layers between the input and output layers to capture complex relationships among data. These networks can predict the output with probability for each class, allowing the application of a single model without resorting to ”one-vs-all” methods. In other word, only one model can be used to predict multiple classes instead of using 30 models, which is the number of classes in this study. Finally, the output of this approach is a prediction of the class with the highest probability.

3.2.3 Evaluation

3.2.3.1 3-Fold Cross-Validation

To assess the performance of the classifier, a three-fold cross-validation approach is used. The number of folds is chosen in consideration of the dataset’s characteristics and in order to avoid computational complexity (Nti, I. K. et al., 2021). This approach involves partitioning the dataset into three equal folds and the classifier undergoes three separate trials. For each trial, two folds are combined to create the training set, while the remaining fold is used as the test set.

3.2.3.2 Evaluation Metrics

The results of the experiment are evaluated in this study using three metrics: precision, recall, and F1 score.

Precision is the proportion of correctly identified positives to all instances labeled as positive. An ideal classifier would achieve a precision of 1, calculated as:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{3.1}$$

Recall, also referred to as sensitivity, measures the proportion of actual positives correctly identified by the classifier. An ideal classifier would have recall values approaching 1, which recall is defined as:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3.2}$$

The F1 score is a harmonic mean of precision and recall, providing a unified measurement for cases where the data is unevenly distributed across several classes. It is defined as:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

To account for the unbalanced dataset and ensure the model performs effectively across all classes, rather than just the majority, the macro F1 score is used to evaluate overall performance in this study. The macro F1 score computes the F1 score for each class separately and then finds the average. It is computed as follows:

$$\text{Macro F1} = \frac{1}{N}\sum_{c=1}^{N} \text{F1}_c \tag{3.4}$$

where N is the number of classes, $F1_c$ is the F1 for class c.

CHAPTER 4
RESULTS AND DISCUSSION

After finishing the construction and testing of the model, the experiment settings and results of this research are presented in this chapter.

4.1 Experiment Settings

The implementation of this research is done using Python 3.9. The data is tokenized using the newMM tokenization method, which is the dictionary-based Maximum Matching with Thai Character Cluster. The experiments are conducted on servers with an Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, with 8 cores and 16 threads, together with 48 GB of RAM. The graphical unit consists of Nvidia RTX 3090 with a VRAM capacity of 24 GB.

This work involves the integration of machine learning and deep learning methodologies. This research uses Scikit-learn version 1.3.2 to implement a range of classification models, including TF-IDF, LR, DT, SVC, and NN. This adaptable library offers a robust framework for conventional machine learning approaches. PyTorch 1.12.1 is used to implement the DNN.

Additionally, this study uses advanced semantic embedding models to capture complex relationships within the data. PyThaiNLP version 4.0.2 is used for Word2Vec, particularly using the ”thai2fit_wv” model. This research implement BERT Transformer models, specifically WangchanBERTa and PhayaThaiBERT, using the HuggingFaces Transformer library version 4.17.0 and sentence-transformers version 2.2.2. The HuggingFaces repository provides two specific model names: ”airesearch/wangchanberta-base-att-spm-uncased” and ”clicknext/phayathaibert”.

4.1.1 Configurations Parameters for Embedding Models

4.1.1.1 TF-IDF

input='content'

encoding='utf-8'

decode_error='strict'

strip_accents=None

lowercase=True

preprocessor=None

tokenizer=None

analyzer='word'

stop_words=None

token_pattern='(?u)\b\w\w+\b'

ngram_range=(1, 1)

max_df=1.0

min_df=1

max_features=None

vocabulary=None

binary=False

dtype=<class 'numpy.float64'>

norm='l2'

use_idf=True

smooth_idf=True

sublinear_tf=False

4.1.1.2 WangchanBERTa

hidden_size=768

num_attention_heads=12

num_hidden_layers=12

intermediate_size=3072

hidden_act="gelu"

initializer_range=0.02

layer_norm_eps=1e-12

max_position_embeddings=514

vocab_size=25000

type_vocab_size=2

pad_token_id=1

position_embedding_type=”absolute”

4.1.1.3 PhayaThaiBERT

hidden_size=768

num_attention_heads=12

num_hidden_layers=12

intermediate_size=3072

hidden_act=”gelu”

initializer_range=0.02

layer_norm_eps=1e-12

max_position_embeddings=512

vocab_size=30000

type_vocab_size=2

pad_token_id=0

position_embedding_type=”absolute”

4.1.2 Configurations Parameters for Classification Models

4.1.2.1 Logistic Regression

penalty=’l2’

dual=False

tol=1e-4

C=1.0

fit_intercept=True

intercept_scaling=1

class_weight=None

random_state=None

solver=’lbfgs’

max_iter=100

multi_class=’auto’

verbose=0

warm_start=False

n_jobs=None

l1_ratio=None

4.1.2.2 Decision Tree

criterion=’gini’

splitter=’best’

max_depth=None

min_samples_split=2

min_samples_leaf=1

min_weight_fraction_leaf=0.0

max_features=None

random_state=None

max_leaf_nodes=None

min_impurity_decrease=0.0

class_weight=None

ccp_alpha=0.0

4.1.2.3 Support Vector Classifier

penalty=’l2’

loss=’squared_hinge’

dual=True

tol=1e-4

C=1.0

multi_class=’ovr’

fit_intercept=True

intercept_scaling=1

class_weight=None

verbose=0

random_state=None

max_iter=1000

4.1.2.4 Neural Networks

hidden_layer_sizes: (20,)

activation=’relu’

solver=’adam’

alpha=0.0001

batch_size=’auto’

learning_rate=’constant’

learning_rate_init=0.001

power_t=0.5

max_iter=200

shuffle=True

random_state=None

tol=1e-4

verbose=False

warm_start=False

momentum=0.9

nesterovs_momentum=True

early_stopping=False

validation_fraction=0.1

beta_1=0.9

beta_2=0.999

epsilon=1e-8

n_iter_no_change=10

max_fun=15000

4.1.2.5 Deep Neural Networks

hidden_layer_sizes: (100,)

activation=’relu’

solver=’adam’

alpha=0.0001

batch_size=’auto’

learning_rate=’constant’

learning_rate_init=0.001

power_t=0.5

max_iter=200

shuffle=True

```
random_state=None
tol=1e-4
verbose=False
warm_start=False
momentum=0.9
nesterovs_momentum=True
early_stopping=False
validation_fraction=0.1
beta_1=0.9
beta_2=0.999
epsilon=1e-8
n_iter_no_change=10
max_fun=15000
```

4.2 Experiment Results

The prediction results shows how well the model works in different classes. There is a class imbalance among the 30 labels that were looked at. The class ”dashboard develop” turned out to be the most correctly predicted. In short, this means that the models, especially WangchanBERTa with Logistic Regression and WangchanBERTa with Deep Neural Network, were effecting at predicting class that had to do with dashboard development.

On the other hand, the class ”requirement gathering” was the least correctly predicted. This means that the models had difficulty correctly identifying and classifying tasks that were related to gathering requirements. The class imbalance could explain this lower accuracy; the class for ”requirement gathering” had the fewest ticket tags (28 tickets).

Table 4.1 shows the combination for the F1 score. The best combination consists of using PhayaThaiBERT embeddings in conjunction with a DNN classifier. This combination achieves an F1 score of 0.769 for the test set and 0.522 in 3-fold cross-validation, with a standard deviation of 0.047.

Table 4.1 F1 Score of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)

	TF-IDF	Word2Vec	WangchanBERTa	PhayaThaiBERT
LR	0.557 / 0.088(0.003)	0.506 / 0.024(0.003)	0.726 / 0.437(0.006)	0.763 / 0.493(0.036)
DT	0.663 / 0.386(0.035)	0.565 / 0.173(0.016)	0.561 / 0.205(0.017)	0.593 / 0.219(0.009)
SVC	0.711 / 0.418(0.023)	0.623 / 0.216(0.041)	0.725 / 0.472(0.018)	0.758 / 0.520(0.038)
NN	0.689 / 0.375(0.044)	0.594 / 0.154(0.038)	0.677 / 0.376(0.024)	0.707 / 0.372(0.033)
DNN	0.702 / 0.429(0.028)	0.670 / 0.303(0.003)	0.742 / 0.470(0.011)	0.769 / 0.522(0.047)

It should be noted that the Macro-F1 score is used in this study because it provides a balanced assessment for every class, giving the majority and minority classes equal weight.

There are several reasons for the differences in F1 score between test set evaluation and 3-fold cross validation. The test set evaluates the model's performance using unseen data, while 3-fold cross validation randomly divides dataset into three folds. Because of this, both can have different distributions of data and can affect the model's performance evaluation. Also, due to the characteristic of the dataset, the changes in data used for training and validating in each fold can significantly affects the results.

In terms of precision as shown in Table 4.2, which measures the accuracy of the positive predictions, the pairing of PhayaThaiBERT with a DNN classifier once again shows superior performance, achieving a precision of 0.738 on the test set and 0.722 during 3-fold cross-validation with a standard deviation of 0.050.

Table 4.2 Precision of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)

	TF-IDF	Word2Vec	WangchanBERTa	PhayaThaiBERT
LR	0.55 / 0.309(0.021)	0.514 / 0.100(0.010)	0.695 / 0.636(0.023)	0.733 / 0.711(0.036)
DT	0.663 / 0.427(0.033)	0.575 / 0.175(0.021)	0.570 / 0.190(0.008)	0.607 / 0.209(0.007)
SVC	0.682 / 0.635(0.042)	0.596 / 0.550(0.099)	0.723 / 0.550(0.041)	0.748 / 0.628(0.030)
NN	0.663 / 0.611(0.106)	0.580 / 0.325(0.021)	0.653 / 0.535(0.104)	0.679 / 0.640(0.071)
DNN	0.681/0.610(0.061)	0.649/ 0.527(0.048)	0.710/ 0.694(0.013)	0.738/ 0.722(0.050)

When assessing recall in Table 4.3, the metric that determines the classifier's ability to correctly identify all relevant instances, the combination of PhayaThaiBERT embeddings with a DNN classifier stands out, with a recall of 0.846 on the test set and 0.435 as per 3-fold cross-validation with a standard deviation of 0.038.

Table 4.3 Recall of the Embedding Vector and Classifier. Test Set Evaluation/ 3-Fold Cross Validation with Standard-Deviation (In Bracket)

	TF-IDF	Word2Vec	WangchanBERTa	PhayaThaiBERT
LR	0.614 / 0.057(0.001)	0.528 / 0.015(0.002)	0.802 / 0.363(0.007)	0.836 / 0.409(0.035)
DT	0.686 / 0.392(0.038)	0.561 / 0.197(0.018)	0.561 / 0.243(0.016)	0.596 / 0.240(0.022)
SVC	0.785 / 0.338(0.024)	0.766 / 0.148(0.036)	0.753 / 0.435(0.005)	0.801 / 0.475(0.037)
NN	0.759 / 0.277(0.050)	0.672 / 0.100(0.038)	0.775 / 0.303(0.022)	0.807 / 0.323(0.003)
DNN	0.751 / 0.361(0.031)	0.729 / 0.240(0.037)	0.836 / 0.374(0.013)	0.846 / 0.435(0.038)

Similar to the F1 score results, there is a difference between recall of test set evaluation and 3-fold cross validation. The classes in the dataset used in this study is imbalanced, meaning that there is a difference in the number of instances between majority and minority classes. It is possible that the minority class is fairly represented in the test set compared to the three-fold cross validation, hence the model performs better on the test set.

The results of this study show the use of advanced word embedding techniques to improve classification performance, especially in low-resource environment, and the effectiveness of BERT-based embeddings when combined with other classification models.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

In this chapter, conclusions of the findings from this study and recommendations are provided.

5.1 Conclusion

In conclusion, the combination of the PhayaThaiBERT embedding vector and a Deep Neural Network (DNN) classifier consistently achieves the best results in terms of several evaluation metrics, such as F1 score, precision, and recall. This exceptional performance was confirmed by validating with both test set evaluation and 3-fold cross-validation methods.

The efficacy of the PhayaThaiBERT embedding vector can be attributed to its capacity to capture complex linguistic characteristics and contextual information, which allows the effective classification within the dataset. This feature greatly improves the model's capacity to make accurate predictions, which could potentially eliminate the need for more complicated models in some situations.

Moreover, the results show that by using robust embedding vectors like PhayaThaiBERT or other BERT models, even less complex classifier models like Logistic Regression can achieve good performance. This suggests that the complex representations obtained by BERT models can effectively support simpler classifiers.

Figure 5.1. illustrates a comparison of various machine learning models' performances on a multi-label classification task, using embeddings from PhayaThaiBERT. The bar chart visually represents the relative predictive behaviors of the models in relation to the actual label counts in the dataset. Note that the figure explain count of predicted labels and discrepencies but does not assess the correctness of predictions

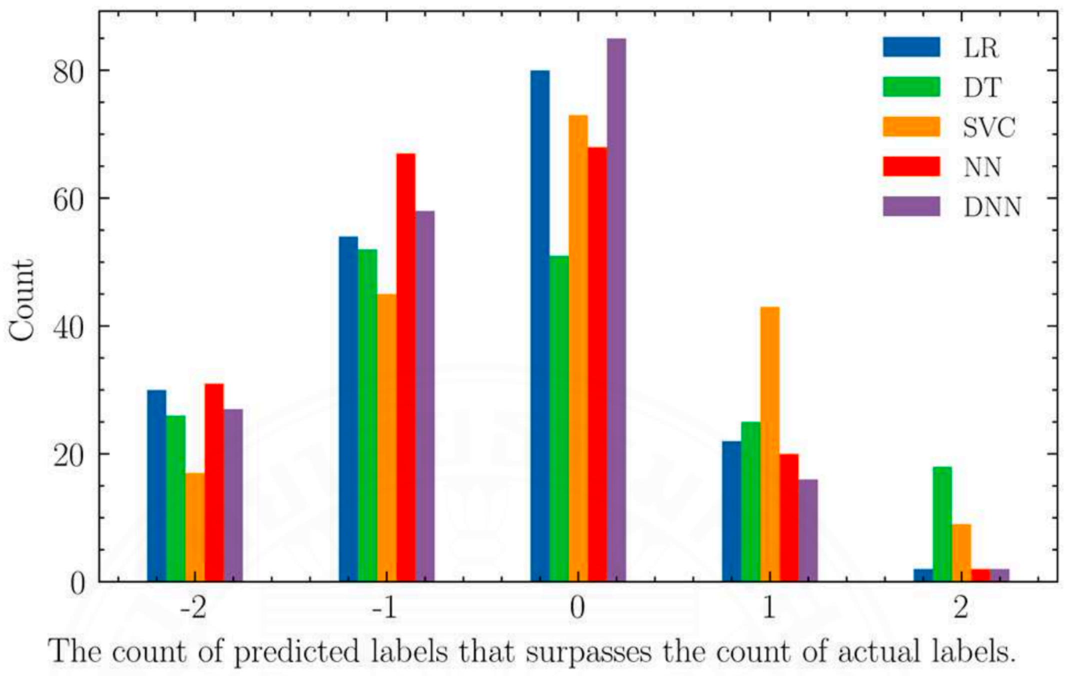

Figure 5.1 Comparison of Machine Learning Models' Performances on a Multi-Label Classification Task, Using PhayaThaiBERT Embedding

The y-axis shows how many times each model under-predicts or over-predicts. The x-axis represents the count of predicted labels that surpasses the count of actual labels, which divided in to -2, -1, 0, 1, and 2. A zero on the x-axis means there is no difference between the count of actual and predicted labels. A difference of 1 or 2 shows that the predicted label is one or two classes more than the actual label. On the other hand, the difference of -1 or -2 occurs when the predicted label is one or two classes below the actual label. Lastly, different colors of the bar graph correspond to different classifiers.

From the figure, it shows that the SVC model tends to over-predict the number of labels in comparison to other models. While DT and NN models tend to under-predict. The SVC has a tendency to over-predict multiple labels, some of which may not be correct. However, it shows high precision which means it is usually correct when it predicts a label. The DT model tends to under-predict the label. As a result, the model fails to correctly predict many of true labels, resulting in a low recall. Similar to the DT, the NN tends to predict fewer labels than are available but has a high recall, which means that when it can predict a label correctly.

However, both LR and DNN show a similar performance, with a higher count of correct label predictions and the least over or under prediction. These results show that LR and DNN models are better at predicting the right number of labels, which leads to better overall performance in multi-label classification tasks.

Table 5.1 shows a comparison of the training time for different combinations. The use of PhayaThaiBERT in with Logistic Regression results in a much reduced training duration of 6.541 seconds, as opposed to using PhayaThaiBERT with Deep Neural Networks, which takes 17.512 seconds. Using PhayaThaiBERT as the embedding model in combination with simpler classifiers such as Logistic Regression might result in faster training times, making it a more favorable option for applications that require rapid model training.

Table 5.1 Training Time of Classification Models with Different Embedding Models

Embedding Model	Classification Model	Training Time (seconds)
TF-IDF	LogisticRegression	1.839
TF-IDF	DecisionTreeClassifier	0.676
TF-IDF	LinearSVC	0.659
TF-IDF	NeuralNetwork	1.362
TF-IDF	DeepNeuralNetwork	2.921
Word2vec	LogisticRegression	4.176
Word2vec	DecisionTreeClassifier	4.339
Word2vec	LinearSVC	4.052
Word2vec	NeuralNetwork	5.819
Word2vec	DeepNeuralNetwork	10.830
WangchanBERTa	LogisticRegression	6.890
WangchanBERTa	DecisionTreeClassifier	7.393
WangchanBERTa	LinearSVC	6.987
WangchanBERTa	NeuralNetwork	11.001
WangchanBERTa	DeepNeuralNetwork	17.186
PhayaThaiBERT	LogisticRegression	6.541
PhayaThaiBERT	DecisionTreeClassifier	7.732
PhayaThaiBERT	LinearSVC	6.891
PhayaThaiBERT	NeuralNetwork	11.128
PhayaThaiBERT	DeepNeuralNetwork	17.512

5.2 Recommendations

Taking into consideration the results of the experiments conducted for this research, there is potential for improvement in the following areas.

1) In the future, researchers may consider using a larger dataset in order to improve the accuracy of the model.
2) The approach of pseudo labelling can be used to increase the quantity of training data and address the issue of imbalanced class, which might eventually result in improved performance of the model.
3) Integrating local language BERT models with multilingual datasets might enhance Thai language understanding while including variations in linguistic and contextual information for other languages.

REFERENCES

Articles

Cer, D., et al. (2018). Universal sentence encoder. arXiv. Retrieved from https://arxiv.org/abs/1803.11175v2

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv. http://arxiv.org/abs/1810.04805

Karl, F., & Scherp, A. (2023). Transformers are short text classifiers: A study of inductive short text classifiers on benchmarks and real-world datasets. arXiv. http://arxiv.org/abs/2211.16878

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

Liu, Y., et al. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. https://doi.org/10.48550/arXiv.1907.11692

Lowphansirikul, L., Polpanumas, C., Jantrakulchai, N., & Nutanong, S. (2021). WangchanBERTa: Pretraining transformer-based Thai language models. arXiv. https://doi.org/10.48550/arXiv.2101.09635

Liu, Z., Benge, C., & Jiang, S. (2023). Ticket-BERT: Labeling incident management tickets with language models. arXiv. http://arxiv.org/abs/2307.00108

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv. https://doi.org/10.48550/arXiv.1301.3781

Polpanumas, C., & Phatthiyaphaibun, W. (2021). thai2fit: Thai language implementation of ULMFit. Zenodo. https://doi.org/10.5281/zenodo.4429691

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. Augmented Human Research, 5(1), 12. https://doi.org/10.1007/s41133-020-00032-0

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning.

Advances in Neural Information Processing Systems.

Shleifer, S. (2019). Low resource text classification with ULMFit and backtranslation. arXiv. https://arxiv.org/abs/1903.09244

Sriwirote, P., Thapiang, J., Timtong, V., & Rutherford, A. T. (2023). PhayaThaiBERT: Enhancing a pretrained Thai language model with unassimilated loanwords. arXiv. https://doi.org/10.48550/arXiv.2311.12475

Songmuang, P., et al. (2021). Development of keyword suggestion for Thai document search. Journal of Technology Management Rajabhat Maha Sarakham University, 8(2), 63–76.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. International Journal of Data Warehousing and Mining, 3(3), 1–13. https://doi.org/10.4018/jdwm.2007070101

Xian, Y., Liu, X., Hu, H., & Sebe, N. (2018). Zero-shot learning: A comprehensive evaluation of the good, the bad and the ugly. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Zhang, M.-L., & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering.

Proceeding

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers (pp. 427–431). https://doi.org/10.18653/v1/E17-2068

Nam, J., Kim, J., Mencía, E. L., Gurevych, I., & Fürnkranz, J. (2014). Large-scale multi-label text classification - Revisiting neural networks. In Proceedings of the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II (pp. 437–452). https://doi.org/10.1007/978-3-662-44851-9_28

Nti, I. K., Nyarko-Boateng, O., & Aning, J. (2021). Performance of machine learning algorithms with different K values in K-fold cross-validation. International Journal of Information Technology and Computer Science (IJITCS), 13(6), 61–71.

https://doi.org/10.5815/ijitcs.2021.06.05

Rahman, M. A., & Akter, Y. A. (2019). Topic classification from text using decision tree, K-NN and multinomial naïve Bayes. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (pp. 1–4). https://doi.org/10.1109/ICASERT.2019.8934502

Venkatesan, R., & Er, M. J. (2014). Multi-label classification method based on extreme learning machines. In 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV) (pp. 619–624). https://doi.org/10.1109/ICARCV.2014.7064375

Wandee, W., & Songmuang, P. (2022). Hierarchical multi-label classification of library subject headings. In 2022 International Conference on Cybernetics and Innovations (ICCI) (pp. 1–5). https://doi.org/10.1109/ICCI54995.2022.9744189

Electronic Medias

Polpanumas, C., & Suwansri, P. (2019). prachathai67k [GitHub repository]. GitHub. https://github.com/PyThaiNLP/prachathai-67k

BIOGRAPHY

Name	Chawanee Sungthong
Date of Birth	August 24, 1993
Educational Attainment	2016: Bachelor of International Management (Accounting and Finance) Ritsumeikan Asia Pacific University
Scholarship	2012: 100% APU Tuition Reduction Scholarship

Publications

Sungthong, C., Ruangtanusak, S., & Songmuang, P. (2024). Multi-label Classification for Low-Resource Issue Tickets with BERT. Paper presented at the 2024 International Conference on Cybernetics and Innovations (ICCI).

Work Experiences	2016 - 2019: International Trade Service Specialist Kasikorn Bank PCL.
	2022 - Present: Data Scientist (BI & Analytics) Bedrock Analytics (PTTEP AI and Robotics Ventures)