A Comparative Study of Pretrained Language Models on Thai Social Text Categorization

Thanapapas Horsuwan[1], Kasidis Kanwatchara[1],
Peerapon Vateekul[1], and Boonserm Kijsirikul[1]

Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok, Thailand
{thanapapas.h,kanwatchara.k}@gmail.com
{peerapon.v,boonserm.k}@chula.ac.th

Abstract. The ever-growing volume of data of user-generated content on social media provides a nearly unlimited corpus of unlabeled data even in languages where resources are scarce. In this paper, we demonstrate that state-of-the-art results on two Thai social text categorization tasks can be realized by pretraining a language model on a large noisy Thai social media corpus of over 1.26 billion tokens and later fine-tuned on the downstream classification tasks. Due to the linguistically noisy and domain-specific nature of the content, our unique data preprocessing steps designed for Thai social media were utilized to ease the training comprehension of the model. We compared four modern language models: ULMFiT, ELMo with biLSTM, OpenAI GPT, and BERT. We systematically compared the models across different dimensions including speed of pretraining and fine-tuning, perplexity, downstream classification benchmarks, and performance in limited pretraining data.

Keywords: language model · pretraining · Thai social media · comparative study · data preprocessing

1 Introduction

Social networks are rich and active platforms that can quickly access public opinions and sentiment regarding different trending topics. The growth of the online lifestyle could be observed by the active communication on social media platforms. In the industry, opinion-oriented information gathering systems aim to extract people's opinion regarding different topics. The sentiment-aware systems have numerous application from business to social sciences. Nevertheless, not much effort has been spent on utilizing these expanding noisy user-generated content as a basis for transfer learning in Thai language.

First introduced in [6], pretrained language models are the recent trend and has proven to improve the state-of-the-art results on a diverse set of tasks in NLP. The language models (LM) that we chose to compare include ELMo [12], ULMFiT [8], GPT [14], and BERT [7]. The goal of unsupervised pretraining is to find a good initialization point to capture the various general meaningful

aspects of a language. With large amounts of unlabeled data from user-generated content, we will investigate whether these language models are able to capture the relevant features of a language. All of our language models are trained in a three-stage process as suggested in [8]: LM pretraining, LM fine-tuning, and classifier fine-tuning.

Pantip is the largest Thai-owned webboard site with a huge active community where a diverse range of topics are discussed. We expect that the variability of surplus examples from Pantip would cover the basic linguistic syntax of Thai language while maintaining the colloquial and noisy nature of online user-generated content.

The main contributions of this paper are the following:

- We developed data preprocessing techniques unique to the Thai social media corpus.
- We pretrained ULMFiT, ELMo, GPT, and BERT on a noisy Thai social media corpus much larger than the existing Thai Wikipedia Dump.
- We compared the language models across different dimensions including speed of pretraining and fine-tuning, perplexity, downstream classification benchmarks, and performance in limited pretraining data.
- Our pretrained models and code will be publicly released.

This paper is organized as follows. Our data preprocessing techniques are explained in Section 2 and the LMs used for pretraining are briefly described in Section 3. The datasets used in this paper are described in Section 4 and Section 5 explains our hyperparameters and evaluation metrics. The results are reported in Section 6 and finally concluded in Section 7.

2 Our Data Preprocessing for Thai Social Media

Data preprocessing is one of the most important phases in improving the learning comprehension of the language models. If much irrelevant and redundant information exists in the noisy and unreliable data, it is difficult for the models to discover knowledge during the training phase. This is especially true for unfiltered data from user-generated content on social media, where it requires specific methods of data preprocessing unique to the domain.

The Thai Webboard Pantip allows members to freely create threads as long as it conforms to a list of actively regulated etiquette. The colloquial nature of the data posted allows for huge amounts of noise to be introduced in the data, such as ASCII arts, language corruption (ภาษาวิบัติ), irregular spacing, misspelling, character repetition, and spams. To reduce the potentially unwanted noise, the data preprocessing approaches we employed are as follows:

1. **Length Filtering** To select threads that would be meaningful to the language model, the thread should have a title and the original post should contain more than 100 characters.

2. **Language Filtering** Threads should mainly consists of text belonging to Thai language, an n-gram-based text categorization library langdetect[1] was utilized to filter out the threads that are not labeled as Thai language.
3. **General Preprocessing before Tokenization** These techniques were inspired by [8,3], including fixing HTML tags, removing duplicate spaces and newlines, removing empty brackets, and adding spaces around '/' and '#'. In addition, character order in Thai language may be typed in a different sequence but visually rendered in the same way. This is due to the fact that vowels, diphthongs, tonal marks, and diacritics may sometimes be physically located above or below the base glyph–allowing different sequential orders to appear the same visually. Thus, normalizing the character order is required for the machine to understand the seemingly similar tokens.
4. **Customized Preprocessing before Tokenization** We also developed and customized some techniques to be more suitable with our social media corpus. Special tokens are utilized for character repetitions, since it is common for Thai people to repeat the last character. This is analogous to prolonging the vowel sounds of a word in spoken language to emphasize certain emotions. pyThaiNLP [13] adopted a similar technique but we implemented minor modifications to add a space following the token for better tokenization results. We adopted a special token for word repetitions similar to the fastai [8] preprocessing technique, which at the time this technique wasn't widely used in Thai language preprocessing. Since Thai is a language without word boundaries, our algorithm recognizes words as any character sequence of more than 2 characters with more than 2 repetitions of that sequence. We truncated all type of repetitions to only 5 times to limit the amount of vocabulary and due to the fact that it doesn't provide any more emotional semantics.

   In addition, we also propose 2 new preprocessing methods: a special token for any numeric strings and a special token for laughing expressions.

   We replaced all strings related to numbers with a special token; this includes general numbers, masked and unmasked phone numbers, Thai numbers, date and time, masked prices, and numbers of special forms. Although differentiating the numbers provide some semantic value, the sparsity of the information would most likely make these token the tail out of vocabulary (OOV) tokens. We believe that this preprocessing method would allow the language model to more generally understand about how numbers are used in text.

   In an online environment, Thai people often express laughter in written language with an onomatopoeia with numbers, utilizing the repetition of '5' followed by an optional '+'. This is due to the fact that the pronunciation of '5' is 'ha'. We replaced all tokens with more than 3 consecutive '5' and an optional '+' with a special laugh token. Although this may have a minor effect on actual numbers, this onomatopoeia is very commonly used in Thai online context and it is important for the model to learn this special token.

[1] github.com/fedelopez77/langdetect

5. **Tokenization** We used the pyThaiNLP [13] default tokenizer, which is a dictionary-based Maximum Matching with Thai Character Cluster. However, we created our own aggregated dictionary for tokenization to improve the tokenization accuracy for colloquial user-generated content. The dictionary[2] is compiled from various sources of data, including general words, abbreviations, transliterations, named entities, and self-annotated Thai slangs and commonly used corrupted language. This includes word variants like ฮะ ฮาฟ ฮับ ฮัฟ คร้าบ ค้าฟ ค้าบ คับ ครัช which are all word variants of the suffix to indicate formality ครับ. The vocabulary is built from the most common 80k tokens.
6. **General Preprocessing after Tokenization:** Following [8] and [3], some general preprocessing techniques after tokenization were used. This includes ungrouping the emoji's from text, and to lowercase all English words.
7. **Spelling Correction** In an effort to reduce the number of unnecessary tokens sprouting from incorrectly spelled words, we compiled a list of common misspelled word mappings aggregated from various sources. We corrected and standardized the vocabulary used. This is an important task due to the free and lax nature of the corpus, where a single word may be represented in different variants or misspelled and abbreviated into various tokens. Note that not all replacements can be made due to the collision of actual vocabularies and the limited comprehensiveness of the list.

3 Pretrained Language Models

3.1 Universal Language Model Fine-tuning (ULMFiT)

A single model architecture that is used for both LM pretraining and downstream fine-tuning was first introduced in ULMFiT [8]. This allows the weights learnt during pretraining to be reused instead of constructing a new task-specific model. Howard and Ruder suggested that LM overfits to small datasets and suffers catastrophic forgetting when directly fine-tuned to a classifier. Hence, the ULMFiT approach was proposed to attempt to effectively fine-tune the AWD-LSTM [9] model. ULMFiT is a 3-stage training method consisting of LM pretraining, LM fine-tuning, and classifier fine-tuning. They also proposed novel techniques such as discriminative fine-tuning, gradual unfreezing, and slanted triangular learning rates for stable fine-tuning.

3.2 Embeddings from Language Models (ELMo)

Traditional monolithic word embeddings such as word2vec [10] and GloVe [11] cannot model how the meaning of each word is varied across different contexts. Hence, ELMo [12] produces contextualized word embeddings by utilizing the pretrained LM as a fixed feature extractor and incorporate its embedding representation as features into another task-specific model for downstream tasks.

[2] The dictionary is referenced at our Gitub `https://github.com/Knight-H/thai-lm`

The authors suggested that combining the internal states of the task RNN layers allows for rich contextualized word representations on top of the original context-independent word embeddings. The bidirectional LM constructed by combining forward and backward LMs would then have its weights frozen and the new model would then be trained as a normal classifier for classification tasks.

3.3 Generative Pre-trained Transformer (GPT)

Sequential computation models used in sequence transduction problems [4,5,15] forbids parallelization in the training examples. The transformer [16] is the first transduction model based solely on self-attention to draw global dependencies between input and output, totally eliminating recurrence and convolutions. OpenAI introduced GPT [14] by extending the idea to use a multi-layer transformer decoder for language modeling. Additionally, LM fine-tuning and classifier fine-tuning is done simultaneously by using language modeling as an auxiliary objective. The authors suggested that this improves generalization of the supervised model and accelerates convergence.

3.4 Bidirectional Encoder Representations from Transformers (BERT)

ULMFiT [8] and GPT [14] uses a unidirectional forward architecture while ELMo [12] only uses shallow concatenation of independently trained forward and backward LMs. With criticism on the standard unidirectional LMs as suboptimal by severely restricting the power of pretrained representations, BERT [7] was proposed as a multi-layer transformer encoder designed to pretrain deep bidirectional representations by jointly conditioning on both left and right context in all layers. Since the standard autoregressive LM pretraining method is not suitable for bidirectional contexts, BERT is trained on masked language modeling (MLM) and next sentence prediction (NSP) tasks. MLM masks 15% of the input sequences at random and the task is to predict those masked tokens, requiring more pretraining steps for the model to converge. The output of the special first token is used to compute a standard softmax for classification tasks.

4 Dataset

4.1 Pretraining Dataset

To collect our Thai social media corpus data, we extracted non-sensitive information from all threads from the most popular Thai webboard *Pantip.com* since 1^{st} January 2013 up until 9^{th} February 2019 using our own implementation of Scrapy Framework [2]. A total of $8,150,965$ threads were extracted. As discussed in Section 2, data preprocessing techniques are applied to the corpus. Length filtering and language filtering filtered down the threads to $5,524,831$ and $5,487,568$ respectively. After preprocessing, tokenization, and postprocessing the data, we divided our pretraining dataset into 3 parts: $5,087,568$ threads

for train, 200,000 threads for validation, and 200,000 threads for test. The train dataset, validation dataset, and test dataset has a total of 1,262,302,083 tokens, 4,701,322 tokens, and 4,588,245 tokens respectively. By comparison, our pretrain dataset is more than 31 times larger than the Thai Wikipedia Dump in terms of number of tokens, which consists of only 40M, 200k, and 200k tokens for train, validation, and test respectively. This makes our proposed pretrain dataset the largest Thai corpus dataset with respect over all Thai public datasets to date.

4.2 Benchmarking Dataset

Two Thai social text classification tasks were chosen to benchmark the model for extrinsic model evaluation as shown in Table 1. Since both are originally Kaggle competitions, the Kaggle evaluation server will be used for benchmarking.

Wongnai Challenge: Rating Review Prediction First initiated as a Kaggle competition, the Wongnai Challenge is to create a multi-class classification sentiment prediction model from textual reviews. As an emerging online platform in Thailand, Wongnai holds a large user base of over 2 million registered users with a surplus of user-written reviews accompanied by a rating score ranging from 1 to 5 stars. This is challenging due to the varying user standards, and the varying weighted importance of each sentiment in mixed reviews.

Wisesight Sentiment Analysis The Wisesight Sentiment Analysis is a private Kaggle competition where it is a multi-class classification on 4 categories: positive (pos), negative (neg), neutral (neu), and question (q). Wisesight, a company doing social data analytics, provides data from various social media sources with various topics on current internet trends. It should be noted that the topics and the source of the data is much more irregular than that of Wongnai.

Table 1. Datasets, tasks, number of classes, train and test examples, and the average example length measured in tokens. The OOV rate is measured with respect to the original vocabulary of the pretraining corpus.

Dataset	Task	Classes	Train	Test	OOV	Average Length
Wongnai	Sentiment Classification	5	40k	6.2k	0.71%	126 ± 124
Wisesight	Sentiment Classification	4	26.7k	3.9k	2.69%	27 ± 44

5 Experimental Setup

5.1 Implementation Details

ULMFiT We mostly used the same model hyperparameters as the popular Thai github repository *thai2fit* [3]: the base model is a 4-layer AWD-LSTM with

1,550 hidden activation units per layer and an embedding size of 400. A BPTT batch size of 70 was used. We applied dropout of 0.25 to output layers, 0.1 to RNN layers, 0.2 to input embedding layers, 0.02 to embedding layers, and weight dropout of 0.15 to the RNN hidden-to-hidden matrix.

ELMo We used the same biLM architecture from the original implementation [12] with all default hyperparameters, where the LM is a 2-layer biLSTM with 4096 units and 512 dimension projections with another static character-based representations layer with convolutional filters. In the task-specific model, a 3-layer biLSTM was used with 256 hidden units.

GPT Default configurations of [14] were used. The resulting model has 12 layers of transformer each with 12 self-attention heads and 768 dimensional states. We used learnt position embeddings and a maximum sequence length of 256 tokens.

BERT We used the publicly available $BERT_{BASE}$ unnormalized multilingual cased model, which has a hidden size of 768, 12 self-attention heads, and 12 transformer blocks. Note that the $BERT_{BASE}$ was chosen to have identical hyperparameters as GPT for comparative purposes.

5.2 Evaluation Metrics

A total of 4 tasks were evaluated: the proposed data preprocessing technique in Section 2, LM pretraining, LM fine-tuning, and classifier fine-tuning. We chose to benchmark on the easiness to train each model (speed and number of epochs), the intrinsic evaluations (perplexity), and the extrinsic evaluations (downstream classification tasks). In addition, an ablation study of limited corpus data will be compared to see the performance of each model in smaller data scenarios.

Data Preprocessing To benchmark the quality of our own unique data preprocessing techniques for Thai social media corpus, we sampled a thread from our training data and request expert Thai native speakers to help tokenize the sample. Although this process takes a lot of effort, up until now there is no standard corpus for benchmarking the tokenization accuracy of non-standard colloquial Thai social media language. The final tokenization micro-f1 score with respect to the first character of a word would then be compared between our method and the default pyThaiNLP [13] Maximum Matching (newmm).

Language Model Pretraining Pretraining a language model is the most expensive process in the transfer learning workflow. However, this task is normally done only once before fine-tuning on a target task. With minimal hyperparameter tuning, we evaluated the pretraining process on: (1) the speed of training in each epoch and (2) the intrinsic perplexity value. Although with the ambiguity that comes with intrinsic metrics, perplexity is one of the traditional

methods in LM evaluation. It measures the confidence of the model on the observed sequence. Due to resource constraints, each model was pretrained for a fixed number of epochs. NVIDIA P6000 is used to pretrain each model, and the appropriate batch size was selected such that it maximizes the GPU VRAM of 24 GB. The models were trained for 3 epochs and the best performing model was selected. However, since BERT trains using MLM and is able to learn just 15% of the corpus during 1 epoch, we decided to train for the standard 1 million steps [7] instead (equivalent to around 6.5 epochs).

Language Model Fine-tuning Each model was benchmarked on the number of epochs used and the total time until convergence. This process aims to learn the slight differences in data distribution of the target corpus. The model overfits very easily due to the modest size of the corpus, thus each LM was fine-tuned until early stopping.

Classifier Fine-tuning In this paper, we reported each downstream task performance in accordance with the metric used in each Kaggle competition. Wongnai Rating Review Challenge uses micro-f1 score while Wisesight Sentiment Analysis uses categorization accuracy. Kaggle ranks the competitors' final standings with the private score, hence this will be used as the benchmark.

6 Results

In this section, we first report the results of our unique preprocessing methods, followed by the results of pretraining the data. We then compare the results of ULMFiT, ELMo, GPT, and BERT with the previous state-of-the-art models in the Thai NLP research community from the Kaggle competition benchmarks.

6.1 Data Preprocessing

Results are shown in Table 2, where our preprocessing method allows the default pyThaiNLP maximum matching (MM) tokenizer to more precisely segment noisy social media data. This is due to the lower false positive tokens segmented by the noisiness of the data, where most of the spams and repetitions are preprocessed correctly. Note that this does not account for the supposed increased comprehension of the model from standardizing the data.

Table 2. Tokenization Precision, Recall, and F1 of the sampled thread.

Tokenizer	Precision	Recall	Micro-F1
MM+Our Preprocessing	**95.92%**	88.11%	**91.85%**
MM	92.47%	**88.89%**	90.65%

6.2 Language Model Pretraining

From Table 3, AWD-LSTM with ULMFiT requires the least amount of time per epoch and the least total time, 100 hours and 33 hours respectively. Due to resource scheduling limitations, ELMo is trained with 2 P6000 GPUs, making the total time and the time per epoch much lower than the supposed value. With character-level convolutions and character-based operations, ELMo training time should be the longest amongst all the LMs. Transformer-based models requires time around more than 1.5x of ULMFiT.

Table 3. Model Pretraining Time. t_{epoch} is the time used per epoch.

Model	t_{epoch}
ULMFiT	**33 hr**
biLM(ELMo) (2 GPU)	52 hr
GPT $seq_{max} = 256$	55 hr
BERT $seq_{max} = 256$	49 hr

The model's final loss and perplexity is shown in Table 4. Perhaps unsurprisingly, the lowest word-level cross entropy loss is the BERT architecture with an MLM perplexity of 15.3857, due to the difference of the MLM prediction task that will always see both the beginning and ending context. This provides more complete contextual information to guess the masked word to the model than the traditional forward or backward models. For a traditional autoregressive models, GPT has lower perplexity than ULMFiT. Since ELMo is character-level, the perplexity is then reported as a character-based perplexity and is not compared to other models.

Table 4. Model Loss and Perplexity After Pretraining

Model	Loss	Perplexity
ULMFiT	3.5281	34.0603
GPT $seq_{max} = 256$	3.1735	23.8913
BERT MLM $seq_{max} = 256$	**2.7334**	**15.3857**
biLM(ELMo) (Character-Level)	1.7140	5.5512

6.3 Language Model Fine-tuning

All the language models are fine-tuned with the target corpus until it gives the best result with respect to the validation loss. An NVIDIA P6000 is used for each model and the time required is presented in Table 5. Transformer-based models are shown to overfit quicker than LSTM-based models.

Table 5. Language Model Fine-tuning Time. t_{total} is the total time used and t_{epoch} is the time used per epoch.

Model	Wisesight			Wongnai		
	#Epoch	t_{total}	t_{epoch}	#Epoch	t_{total}	t_{epoch}
ULMFiT	11	**11 min**	**1 min**	11	99 min	**9 min**
biLM(ELMo)	5	25 min	5 min	**2**	64 min	32 min
GPT $seq_{max} = 256$	**3**	57 min	19 min	3	90 min	30 min
BERT $seq_{max} = 256$	**3**	36 min	12 min	**2**	**38 min**	19 min

6.4 Classifier Fine-tuning

The results of the downstream classification tasks are shown in Table 6. It can be observed that BERT with our pretraining data achieves the state-of-the-art micro-f1 and accuracy on Wongnai and Wisesight respectively. Score improvements can be seen when comparing similar models between pretraining with the Thai Wiki Dump and pretraining with the our Thai Social Media data. Comparing the architecture ULMFiT in both tasks, there are improvements of around 2% after pretraining on our data. For BERT, after increasing the sequence length and pretraining on our data, improvements of around 5% can be seen in the Wongnai task. Although ELMo has relatively lower results compared to the other LM architectures, this is suspected to be due to the simplistic nature of the biLSTM task-specific model.

6.5 Limited Pretraining Corpus

We will also investigate the performance of the models in the scenario where the pretraining corpus is limited. This result reflects the learning ability of the model in low-resource languages or domains where training data is scarce. We randomly sampled a total of 40M tokens (equivalent to around 234K threads) from the whole dataset of 1.2B tokens used in our previous experiments. ULMFiT, ELMo, and GPT are trained with 3 epochs while BERT is trained for 30k steps (equivalent to approximately 6.5 epochs on this data). Table 7 shows that ULMFiT and GPT performs considerably well. On the other hand, adding ELMo to LSTM input shows little improvement. This means that ELMo requires larger corpus to be effective. Although BERT performs well in the Wisesight dataset, it has a drop of performance in Wongnai.

7 Conclusion

Our work shows that by using our unique data preprocessing methods and our pretraining social media data, we can improve the performance of the LMs in the downstream tasks. The improvement of all models from pretraining data of the same domain suggests that pretraining data has a significant impact on

Table 6. Classifier Fine-tuning Results. Wongnai evaluates with micro-f1 while Wisesight evaluates with accuracy. Our model is compared to other models: the baseline that predicts the most frequent label, the latest Kaggle competition winner, and public github repositories. The public leaderboard and private leaderboard is calculated with approximately 30% and 70% of the test data respectively.

Model	Wisesight (Acc.)		Wongnai (f1)	
	Private	Public	Private	Public
Baseline	0.5809	0.6044	0.4785	0.4785
Kaggle Best	0.7597	0.7532	0.5914	0.5814
fastText [3]	0.6131	0.6314	0.5145	0.5109
LinearSVC [3]	-	-	0.5022	0.4976
Logistic Regression [3]	0.7499	0.7278	-	-
Thai Wiki Dump Pretraining				
ULMFiT [3]	0.7419	0.7126	0.5931	0.6032
ULMFiT Semi-supervised [3]	0.7597	0.7337	-	-
BERT $seq_{max} = 128$ [1]	-	-	0.5661	0.5706
Ours (Thai Social Media Pretraining)				
ULMFiT	0.7586	0.7346	0.6203	**0.6409**
biLSTM	0.6366	0.6213	0.4773	0.4946
ELMo+biLSTM	0.6866	0.6450	0.5310	0.5226
GPT $seq_{max} = 256$	0.7669	**0.7540**	0.6088	0.6145
BERT $seq_{max} = 256$	**0.7691**	0.7439	**0.6251**	0.6231

LM performance. Moreover, the possibility for LM pretraining on a noisy corpus shows the ability of the models to learn in spite of the quality of the data.

Results-wise, BERT is the most suitable model for text classification with respect to accuracy. It can achieve state-of-the-art results on both of our benchmarking downstream tasks. However, it has unstable performance on pretraining on a small corpus and uses a lot of pretraining time. If speed and ease of training is the main consideration, we recommend using AWD-LSTM with ULMFiT due to its speed of pretraining and fine-tuning, while the results are on par with transformer-based models. Although OpenAI GPT shows promising results with acceptable pretraining speed, it is overshadowed by other models in both aspects. Finally, although ELMo shows significant improvements when compared with the baseline biLSTM, it places a dependency on designing a powerful task-specific model to achieve good performance.

Acknowledgements

In the making of the paper, the authors would like to acknowledge Mr. Can Udomcharoenchaikit for his continuous and insightful research suggestions until completion.

Table 7. Limited Pretraining Corpus Results. The public leaderboard and private leaderboard is calculated with approximately 30% and 70% of the test data respectively.

Model	Wisesight (Acc.)		Wongnai (f1)	
	Private	Public	Private	Public
ULMFiT	0.7358	0.7143	0.5984	**0.6290**
biLSTM	0.6366	0.6213	0.4773	0.4946
biLSTM + ELMo	0.6489	0.6095	0.4879	0.4753
GPT $seq_{max} = 256$	0.6931	0.7075	**0.6111**	0.6102
BERT $seq_{max} = 256$	**0.7467**	**0.7244**	0.5650	0.5516

References

1. Bert-th. `https://github.com/ThAIKeras/bert` (2019)
2. scrapy. `https://github.com/scrapy/scrapy` (2019)
3. thai2fit. `https://github.com/cstorm125/thai2fit` (2019)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
5. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv e-prints arXiv:1406.1078 (Jun 2014)
6. Dai, A.M., Le, Q.V.: Semi-supervised Sequence Learning. arXiv e-prints arXiv:1511.01432 (Nov 2015)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv e-prints arXiv:1810.04805 (Oct 2018)
8. Howard, J., Ruder, S.: Universal Language Model Fine-tuning for Text Classification. arXiv e-prints arXiv:1801.06146 (Jan 2018)
9. Merity, S., Shirish Keskar, N., Socher, R.: Regularizing and Optimizing LSTM Language Models. arXiv e-prints arXiv:1708.02182 (Aug 2017)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. arXiv e-prints arXiv:1301.3781 (Jan 2013)
11. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). p. 1532–1543 (2014)
12. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv e-prints arXiv:1802.05365 (Feb 2018)
13. Pythainlp 2.0. `https://github.com/PyThaiNLP/pythainlp` (2019)
14. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
15. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to Sequence Learning with Neural Networks. arXiv e-prints arXiv:1409.3215 (Sep 2014)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. arXiv e-prints arXiv:1706.03762 (Jun 2017)