Thai Comments Sentiment Analysis on Social Networks with Deep Learning Approach

Chayapol Piyaphakdeesakun, Nuttanart Facundes, Jumpol Polvichai
Department of Computer Engineering
King Mongkut's University of Technology Thonburi, Bangkok, Thailand
chayapol.lxze@outlook.com, nuttanart@cpe.kmutt.ac.th, jumpol@gmail.com

Abstract

Sentiment analysis is a kind of a Natural Language Processing (NLP) tasks in text classification used to determine the sentiment polarity from any given document. In this research, two problems were solved. The first problem is the word ambiguous manipulation. Another task is an automatic sentiment classification. This work shows our process of Thai online document cleaning to attack the uncertainty in Thai text. This work also includes attempts to find the appropriate deep learning algorithm to classify the sentiment of Thai online document by comparing the performance among several approaches. The results show that a model using the bi-directional gated recurrent unit (GRU) with attention mechanism yields the best performance in sentiment classification.

Keywords: Sentiment Analysis/ Thai Language/ Natural Language Processing/ Deep Learning

1. Introduction

Sentiment analysis is a process of identifies the text data that contains emotion or opinion, understand and determine the positive or negative aspect of comments. There are plenty of data available in social media and several online platforms, which allow us to perform a sentiment analysis for the favour of business profit or academic interest. Otherwise, dealing with text data from social media, which contain tons of uncertainty, that somehow difficult to handle. Especially a text from social media, which mostly contains a slang, cultural context, dialect word and typo requires some works on data cleaning and text normalisation.

In this research, the deep learning model has acquired to perform a classification task. However, training a deep learning model until a model yields the result in satisfied accuracy legitimately relies on both quality and quantity of data; therefore, paying attention to data collecting and processing is needed to assure that there is sufficient amount of data to make a model work well. Consequently, to deal with the uncertainty in documents, our customised data preparation process has developed according to the pattern that frequently occurred in user-generated Thai text. Even though several works already applied the deep learning algorithm to working with Thai text classification, they only perform with the regular recurrent neural network model. Then, the attention mechanism has been introduced to improve the performance of neural network model; therefore, there might be an opportunity to apply the attention mechanism and observe the result in Thai text sentiment prediction. At last, the model with the most satisfied F1 score indicates as the best algorithm that performs decently in the sentiment classification.

This work contains topics as follows. Section 2 states the related works. Next, section 3 describes the involved algorithms and techniques, and section 4 describes the experiment and result. Finally, section 5 is a conclusion and some discussion on this work.

2. Related Works

NLP on Thai text has issued and recently solved with many techniques. There is an improvement in performance in many NLP tasks such as word segmentation, text classification and sentiment analysis by using a machine learning model. Although, most of the works in Thai NLP researches rely on the public dataset in which error within document already manipulated or the data crawled from the trusted sources such as news site; therefore, using a model that training on the well-organised dataset with common user-generated data might yield an unexpected result due to a flaw within data. therefore, to deal with uncertainty and error that mostly occurred in an online document, the development in more specific cleaning method is significant for creating the training data more reliable. In 2016, Peerapon V. et al. experimented with online Thai text and mentioned about the text cleaning method that design specifically [1]. Hence, adopting the existed cleaning method from the previous work and added a more specific procedure is more practical.

About sentiment analysis on Thai text, there are some recent researches based on online documents that conducted with the deep learning algorithm. Peerapon V. et al. [1] studied on Dynamic Convolutional Neural Network (DCNN) and Long Short-term Memory (LSTM) and experimented its performance on Thai twitter data as mentioned. Also, in 2018, Chawisa P. [2] studied on several kinds of deep learning architecture and proposed a Unified

CNN Bidirectional GRU (UCBGRU) which perform at its best on sentiment classification for Thai financial news specifically. There is an intention to experiment on CNN, LSTM and GRU with the bi-directional scheme with our crawled data. Furthermore, the attention mechanism is also a topic worth mentioning at the moment. Attention mechanism has been invented initially to help a model decide to focus or filter the particular area of information before passing to the classification layer. This mechanism was acquired to several works in NLP and show improvement on the increasing performance of document classification [3]. Also, the attention mechanism has not experimented yet in Thai text classification; therefore, applying and experimenting with this mechanism including other deep learning models might be beneficial.

3. Methodology

This section contains the methodology of cleaning method described in detail, the structure of involved deep learning algorithms which are LSTM, GRU, bi-directional recurrent neural network (RNN), CNN and attention mechanism.

A. Data Preparation

In this work, as the users generate these documents, the data might contain lots of human error; of course, the raw data requires some process to clean and fix before using the data for the next step of data processing. In order to clean up the user-generated Thai document, our cleaning method has developed for this task specifically. Here are our steps of the cleaning process.

1) Remove the username pattern (@abc), Emoji, URL, hashtag, unique character (ex: "|", "\", "•") duplicated whitespace and line break.
2) Filter only Thai (U+0E00 to U+0E7F), English character ("a" to "z" and "A" to "Z") and some mathematical characters (0 to 9, ฿, %)
3) Manipulate some pattern which usually occurred in Thai
   - Laughing Thai texts: substitute the phrase "555", "555+", "ถถถ" "๕๕๕" with "_lol_" token instead.
   - Fix vowel character typos: Replacing "เเ" (double "เ") with "แ".
   - Reduce the repeated characters: if the repeated character is a vowel, then reduce to one character. If there are more than three consonant characters repeated, reduce to two characters.

After the cleaning process, the maximum matching algorithm is used to tokenise each document into a list of words. Word embedding technique [4], which is a process that converts each word in a sentence into a vector form, is also necessary in order to represent word semantic with numerical data in several dimensions and be able to use as an input of deep learning model. For ease of development, pre-trained word vector from the repository [5] of public Thai ULMFiT (Universal Language Model with Fine-Tuning) is acquired to use as an initialised value.

B. Long Short-Term Memory

LSTM [6] is one of the RNN architectures. LSTM contains the gated mechanism and internal cell memory that help the model decide how much information pass and capture more information between large timestep. Fig. 1 illustrates the LSTM architecture. There are three gates in every recurrent node, input gate, forget gate and output gate, which indicated with i_t, f_t, o_t respectively. Each gate let the model consider the data should be passed or not, by the result of sigmoid which is in the range of zero to one. Each gate's equation is:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3}$$

Then, the gate limits the information value by applying with the previous cell state, the current cell state, and calculated output. The equation to calculate cell state and output is:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$
$$C_t = (f_t * C_{t-1}) + \left(i_t * \tilde{C}_t\right) \tag{5}$$
$$h_t = o_t * \tanh(C_t) \tag{6}$$

Finally, to use the information that has been encoded by LSTM layer, the classification layer is applied to the output of the last step only.

C. Gated Recurrent Unit

GRU [7] is also another kind of RNN architecture. GRU uses update gate z_t and reset gate r_t. Each GRU node computes the new candidate output from the current given input and previous output, multiplied with the z_t and its inverted value respectively, which means that a node computes the output according to the fraction of update gate's value. If there is more update value, a GRU node is likely to use the new candidate output more; on the other hand, it uses the old result. Fig. 2 shows the architecture of GRU. Update gate and output gate are calculated as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{7}$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{8}$$
$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t]) \tag{9}$$
$$h_t = \left((1 - z_t) * h_{t-1}\right) + \left(z_t * \tilde{h}_t\right) \tag{10}$$

D. Bi-directional RNN

Bi-Directional RNN [8] is an RNN structure that working in bi-directional instead of unidirectional. This structure works by adding one more hidden cell state in each node a recurrent unit in the same layer, but the additional cell transferring output in a backward direction. This structure let model takes information not only from the past but also from the future, so each hidden state inside a node capture both historical and futuristic information, then combine these two values to final output. This implementation also applies to LSTM or GRU as a recurrent node for a better result. Fig. 3 represents the bi-directional structure using LSTM.

E. Attention Mechanism

In order to find the attention vector, the signal of each time step of output from the previous layer h_t is applied with weight matrix and bias before passed into the Tanh function, then, transpose the result and perform dot product with another weight vector u_w and apply a Softmax function to find the position that should pay attention. Finally, multiply the attention vector a_t with the output from the previous layer. The formulae to calculate the resulted vector is:

$$u_t = \tanh(W_w \cdot h_t + b_w) \tag{11}$$

$$a_t = \frac{\exp(u_t^{\mathrm{T}} \cdot u_w)}{\sum_t \exp(u_t^{\mathrm{T}} \cdot u_w)} \tag{12}$$

$$v = \sum_t a_t h_t \tag{13}$$

F. Convolutional Neural Network

In this work, using a one-dimension convolution method in CNN [9], which also called temporal convolution, is suitable for the text classification tasks. Each filter convolutes through the matrix of word vector and filter size is vector dimension multiply by the number of a word according to the filter width. Eventually, the activation function calculates the output of this convolutional step.

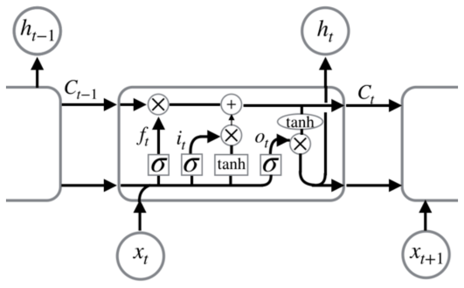

Fig. 1. Long Short-Term Memory

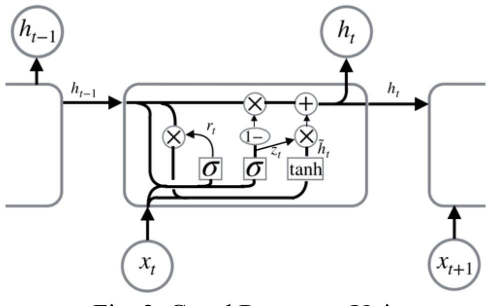

Fig. 2. Gated Recurrent Unit

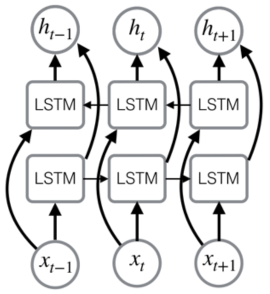

Fig. 3. Bi-directional LSTM

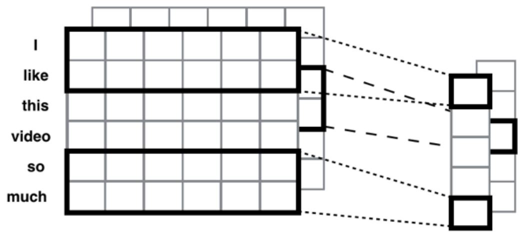

Fig. 4. One-dimension Convolution

Next, Pooling process, which is the necessary step that performs after convolution, reduces the dimension of the output from previous convolution step by combining them into a single output by depends on method, for example, max pooling which selects the maximum value from each of output clusters according to pooling size. Fig. 4 illustrates the one-dimension convolution method.

4. Experiment Results

The total amount of data collection is 41,073 documents, divided into 21,490 positive documents and 19,583 negative documents. The ratio of splitting the train and test dataset is 85:15 respectively. For the hyperparameter configuration, there is a setting as follows. Using Adam optimiser [10] with learning rate 0.001, in RNN, dropout probability = 0.2, recurrent dropout = 0.1, in Multi-layer perceptron (MLP), dropout probability = 0.2. Each word contains 300 dimensions of vector; each document contains 600 words. The batch size is 50 documents per batch. For CNN, there are two layers, filter size = 3, pooling size = 2 in both layers. The first layer contains 300 kernels, and the next layer contains 256 kernels. For Bi-directional LSTM (BLSTM), there are two layers. The first layer contains 150 cells within the node in each direction. The next layer contains 128 cells. For Bi-directional GRU (BGRU), there are two layers with the same configuration with BLSTM. For MLP, there are two layers; the first one contains 128 nodes which use Sigmoid as an activation function; the next layer use Softmax to classify the data into two classes. Each model is trained with a maximum of 15 epochs with early stopping technique that prevents the model from becoming overfitting with the training data. These number of kernels and cells of CNN and RNN in hyperparameter configurations were determined by the empirical experiment with concerning the size of the word vector's dimension, outputs size and the similarity of the dimension of data among each architecture.

The F-measure (F1 score) is acquired to use as a criterion to define the performance of the model. It is a combination of precision and recall value. Precision is a measure of how much our model predict sentiment correctly, and Recall is a measure of how many related phrase rates as sentimental.

All of the combinations of deep neural network algorithm, which are CNN, BLSTM and BGRU, with and without attention mechanism (ATTN), are experimented and concatenated into MLP. In this work, the baseline algorithm is Naïve Bayes (NB) algorithm. The results of our experiment are shown in table 1.

Table 1: F1 score & Training Time on Various Type of Deep Learning Model. Boldface is The Winner.

Model	F1 Score	Training Time (second)
NB	71.50	-
CNN	91.71	991
CNN+ATTN	91.31	978
BLSTM	91.36	61,884
BLSTM+ATTN	91.56	50,415
CNN+BLSTM	91.27	16,564
CNN+BLSTM+ATTN	90.95	13,775
BGRU	91.69	42,885
BGRU+ATTN	**91.85**	41,119
CNN+BGRU	91.66	13,506
CNN+BGRU+ATTN	91.37	13,122

Table 1 shows that BGRU with attention mechanism yields the best performance in sentiment classification compared with other deep learning models. Also, the model that contains BGRU performs slightly better than the model that contains BLSTM; plus, applying the attention mechanism after CNN in the model declines the performance in classification. Surprisingly, using only CNN resulted in more F1 score than the stacked CNN and RNN model. As shown, every deep learning model outperforms the Naïve Bayes algorithm.

Considering the results of an experiment, when each model consists only RNN which are BLSTM and BGRU, it resulted in more F1 score when used the attention mechanism. On the other hand, the model that has CNN as a part of a model and using the attention mechanism reduces the F1 score. Regarding the model training time, even though using CNN before RNN causes the model uses less training time, it slightly decreases the performance of the model in this circumstance. Also, LSTM requires more time to train than GRU, and simple CNN model performs with satisfying performance with clearly seen that it requires noticeably less training time than other models.

Besides, in the prediction results, compare between BGRU-ATTN and CNN-BLSTM-ATTN, CNN-BLSTM-ATTN cannot predict correctly where some inputs do not contain strong sentiment words; thus, with a properly designed deep learning model, it results in a model has more ability to recognise sentiment in a document within the same training epochs even without strong emotional words contained.

5. Conclusions

Several deep learning models were experimented to classify Thai comments on social networks. The performance of each model was evaluated based on F1 score. The results show that most of the algorithms except the Naïve Bayes algorithm perform satisfied performance. The bi-directional GRU with attention mechanism applied yields the best F1 score in this circumstance. Also, the attention mechanism makes RNN perform better in text classification. Surprisingly, the model using only simple CNN achieves second place in the model evaluation with rapid training time.

The Thai text cleaning process was also developed to manipulate unusual patterns in documents. In future works, the additional input features such as part-of-speech will be applied for practical usage.

References

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

[2] P. Chawisa, "Integration of Fine-Grained Sentiment Analysis and Deep Neural Network to Analyze Financial News," 2018.

[3] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[4] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research,* vol. 3, pp. 1137-1155, 2003.

[5] C. Polpanumas, "cstorm125/thai2vec," 23 June 2018. [Online]. Available: https://github.com/cstorm125 /thai2vec. [Accessed 24 October 2018].

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, pp. 1735-1780, 1997.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing,* vol. 45, pp. 2673-2681, 1997.

[9] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882,* 2014.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.