BiLSTM-CRF and BiGRU-CRF for Thai Segmentation

Nick Tantivasadakarn, Armando Bañuelos
Stanford University
nantanic@stanford.edu, abanuelo@stanford.edu

Abstract

Thai is one of the languages that does not have explicit segmentation, and cannot be used with most word based models. In this paper we will be tackling this problem by implementing BiLSTM-CRF and BiGRU-CRF based segmentation algorithms to parse Thai text. Our model achieves an F1 score of 94.78 (micro) and 96.26 (macro). Our model outperforms the micro averaged F1 score from previous models and has comparable macro F1 score. The model also works well on small data, but struggles with named entities.

1 Introduction

Thai is one of the languages that does not have explicit segmentation similar to languages such as Chinese, Japanese, and Arabic. This means that Thai cannot benefit from the use of word embeddings and most word based models without the use of a segmentation algorithm. The main difficulty lies in the fact that Thai is a phonetic script, which means that words tend to contain multiple characters and can have multiple valid segmentations depending on the context. For example, the phrase “ตากลม" has two valid segmentations. The first is “ตา-กลม" which means “round eye", and the second is “ตาก-ลม" which means “catching the wind". In this project, we aim to create a BiLSTM-CRF and BiGRU-CRF to learn Thai segmentation. Note that in this paper we will be using the words tokenization and segmentation interchangeably.

2 Related Work

Among the officially published work, the task of Thai segmentation has been dictionary-based (DCB) and pre-neural machine learning (MLB) [1]. The accuracy of DCB methods such as longest matching and maximal matching depends on a set of parsed and segmented input texts. Whereas the accuracy of MLB methods such as naive bayes (NB), decision trees, support vector machines (SVM), and conditional random fields (CRF) depends on the training corpora. In comparing both methodologies, DCB approaches yielded better performances than NB, decision tree, or SVM algorithms. CRFs produced the best performance with an F1 of 95.38 [1]. To our knowledge, there does not exists published work on neural network (NN) based Thai segmentation algorithms. Nevertheless, there exists unpublished work that make use of Convolutional Neural Networks (CNN) [2] and Recurrent Neural Networks (RNN) [3].

Inspiration for our BiLSTM-CRF and BiGRU-CRF Thai segmentation model stems from related work in segmentation of other languages that have historically proven difficult to tokenize such as Arabic and Chinese. Yao and Huang introduced a BiLSTM recurrent neural network for Chinese word segmentation proving highly effective in tokenization tasks (F1 of 97.5%) [4]. In further work, Samih et. al. applied a character-based BiLSTM-CRF Arabic tokenizer that achieves an F1 of 92.65 % [5]. Interest in experimenting with the BiGRU-CRF for Thai segmentation stems from the faster convergence rates in comparison to BiLSTM layers.

3 Approach

3.1 BiLSTM

Most pre-neural MLB Thai segmentation approaches are constrained by small contextual windows for tokenization, inhibiting the learning ability of the models. Interest in incorporating a BiLSTM layer into the architecture of our segmentation model stems from their ability to learn long-term dependencies and contextual features from previous and future states. The BiLSTM calculates two parallel layers, a forward hidden layer and backward hidden layer, to generate an output sequence y as illustrated:

$$h_{f_t} = \sigma(W_{xh_f}x_t + W_{h_fh_f}h_{f_{t-1}} + b_{h_f}) \tag{1}$$

$$h_{b_t} = \sigma(W_{xh_b}x_t + W_{h_bh_b}h_{b_{t-1}} + b_{h_b}) \tag{2}$$

$$y_t = W_{h_{f_y}}h_{f_t} + W_{h_{b_y}}h_{b_t} + b_y \tag{3}$$

Here $x_t \in \mathbb{R}^d$ is a d-dimensional input vector at time step t, W are the weight matrices, b are bias vectors, and $h_f \in \mathbb{R}^d$, $h_b \in \mathbb{R}^d$ are the output of the LSTM forward and backward layers respectively.

3.2 BiGRU

As an extension to the BiLSTM layer, interest in incorporating a GRU layer stems from the fact that they combine hidden state and cell state into one, resulting in faster training, particularly when training on large corpora. More formally, a GRU network reads input tokens t_i and previous hidden state h_{i-1} to generate an output sequence c_i and hidden unit h_i, as illustrated:

$$z_i = \sigma(W_zt_i + V_zh_{i-1} + b_z) \tag{4}$$

$$r_i = \sigma(W_rt_i + V_rh_{i-1} + b_r) \tag{5}$$

$$c_i = tanh(Wt_i + V(r_i \odot h_{i-1}) + b) \tag{6}$$

$$h_i = z_i \odot h_{i-1} + (1 - z_i) \odot c_i \tag{7}$$

Here $\mathbf{z} \in \mathrm{R}^d$ and $\mathbf{r} \in \mathrm{R}^d$ represent the input and reset gates for a d-dimensional input, $\{W_z, W_r, W, V_z, v_r, V\}$ are the weight matrices, $\{b_z, b_r, b\}$ are bias vectors, and $\odot$ indicates element-wise matrix multiplication.

3.3 CRF

When training BiLSTMs or BiGRUs, the output probability distribution y_t assumes that each time steps are independent. Incorporating CRFs allows us to overcome these independence assumptions by labeling the entire sequence. CRFs can take entire contexts into account to predict sequences of labels determining where input should be segmented as illustrated:

$$P(\hat{y}|\hat{x};w) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \hat{x}, i))}{\sum_{y_z \in y} exp(\sum_i \sum_j w_j f_j(y_{z_{i-1}}, y_{z_i}, \hat{x}, i))} \tag{8}$$

Here $\hat{y}$ represents the sequence of labels, w is the weight vector for weighing the output feature vector f from the BiLSTM or BiGRU. Training and decoding will be performed by the Viterbi algorithm.

3.4 Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm used primarily within the context finding the most likely sequence of hidden states. Our model incorporates the Viterbi algorithm to decode outputs from the CRF layer into binary output, denoting the existence or nonexistence of a parse after a character. Below is the pseudocode for the Viterbi algorithm.

Algorithm 1: Viterbi(Emission, Trans, Start, End):

```
$s \leftarrow 0$
$y \leftarrow []$
$Trellis \leftarrow$ empty NxL matrix
$Backpointers$ empty (N-1)xL matrix
$Trellis[0,:] \leftarrow Start + Emission[0,:]$
for $i \leftarrow 1 : N-1$ do
    for $j \leftarrow 0 : L-1$ do
        $Trellis[i,j] \leftarrow Emission[i,j] + max_k(Trans[k,j] + Trellis[i-1,k])$
        $Backpointers[i-1,j] \leftarrow argmax_k(Trans[k,j] + Trellis[i-1,k]$
    end
end
$s \leftarrow max_k(End[k] + Trellis[-1,k])$
$b_{next} \leftarrow argmax_k(End[k] + Trellis[N-1,k])$
$Y[N-1] \leftarrow b_{next}$
for $i \leftarrow N-2$ to 0 do
    $b_{next} \leftarrow Backpointers[i, b_n ext]$
    $Y[i] \leftarrow b_n ext$
end
return (s,Y)
```

Here, N is the number of characters in the input string and L is the number of labels.

3.5 BiLSTM-CRF and BiGRU-CRF for Thai Segmentation

Our approach utilizes a BiLSTM architecture, BiGRU architecture and CRF for the learning of contextual segmentation for Thai as shown in Figure 1. More specifically, our model

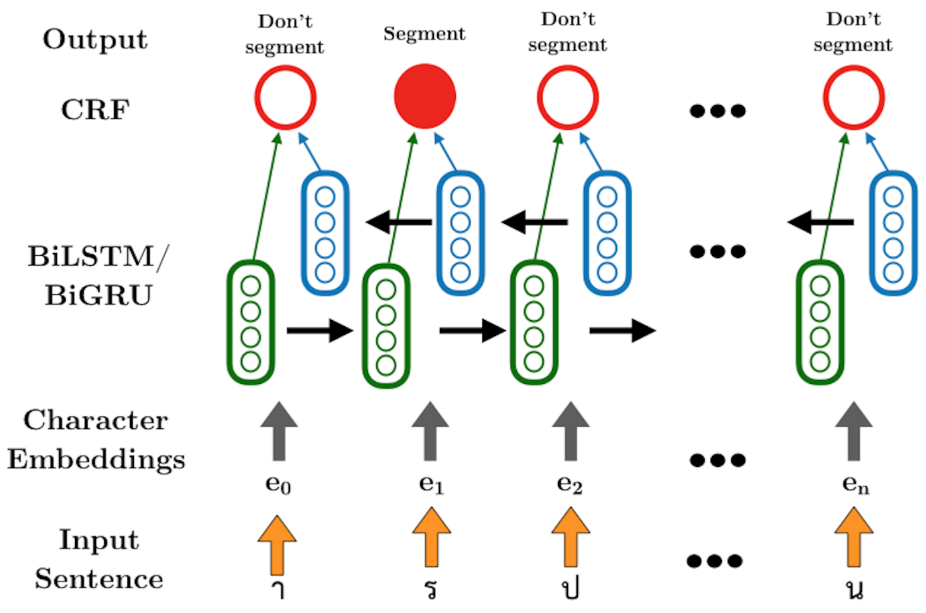

Figure 1: Thai Segmentation Model using BiLSTM-CRF architecture or BiGRU-CRF architecture. [6].

comprises three different layers: an input layer containing character embeddings, a hidden layer where the BiLSTM or BiGRU maps embeddings to hidden sequences, and an output layer which takes in hidden sequences to compute probability of tokenization labels. The CRF tokenization labels are binary to denote the existence or nonexistence of a parse after a character.

4 Experiments

4.1 Dataset

Our data set is the BEST2010 [1] corpus, compiled by Thailand's National Electronics and Computer Technology Center. This corpus contains 5 million segmented Thai words from articles, news publications, encyclopedia entries, and novels. The corpus also include named entity and abbreviation tagging.

4.2 Evaluation Method

We use precision and recall to calculate character-level micro F1-score and macro F1-score as the evaluation metric for our model.

4.3 Experimental Details

Our data will first be split into 70% training data, 20% development data, and 10% testing data. There are four experimental conditions for the BiLSTM-CRF. The first trains on all of the training data with the named entities and abbreviations collapsed into new tokens. The second is training with the named entities and abbreviations included. The third and fourth

[1]https://lst-nlp.openservice.in.th/data/Best.rar

are similar to the first two, but with a small subset (roughly 10%) of the training examples. The BiGRU-CRF incorporates the first, third, and fourth experimental conditions but was unable do the second condition due to time constraints.

All models and conditions were run on a Microsoft Azure NV6 virtual machine. The models use 64 embedding dimensions, 128 hidden dimensions, and learning rate of 0.001. Models that run on 10% of the training data have a maximum of 30 training epochs, while models that trained on the full data set were run for 4 days on a GPU due to time limitations.

4.4 Baseline

We compare our results to two pre-trained neural network models that are not officially published. The first is the CutKum model, which uses a recurrent neural network (RNN)[2]. The second is the Deepcut model which uses a convolutional neural network (CNN). These two models are chosen due their ease of access.

4.5 Results

Model	Without named entities		With named entities	
	F1-Micro	F1-Macro	F1-Micro	F1-Macro
CutKum	88.96	96.95	88.18	96.27
DeepCut	89.95	**97.99**	89.61	97.80
BiLSTM-CRF	94.39	95.72	92.54	93.74
BiLSTM-CRF (10%)	93.47	94.40	91.11	92.24
BiGRU-CRF	**94.78**	**96.26**	-	-
BiGRU-CRF (10%)	93.88	95.01	91.36	92.70

Table 1: F1 Metrics gathered from baselines, BiLSTM-CRF and BiGRU-CRF models.

Table 1 lists the performances of our models as well as previous research. Out of the three sets of tests conducted, the BiGRU-CRF performed best in both the F1-Micro and F1-Macro metrics. However, the Deepcut model performs best in the F1-Macro metric. Both Deepcut and CutKum express lower F1-Micro scores in comparison to the three evaluative tests conducted. The high F1-Micro scores for the BiLSTM-CRF and BiGRU-CRF models provides reason to believe that these models handle segmentation of individual Thai sentences more consistently.

When comparing F1 scores between models with named entities and without named entities, the CutKum and DeepCut models have less than a 0.5% decreased score whereas the BiLSTM-CRF and BiGRU-CRF models have approximately a 2% decrease. Another insight from the testing results is that despite training on a larger dataset, the BiLSTM-CRF and BiGRU-CRF models that were trained on 10% of data reported comparable F1 metrics. This suggests that this model can work relatively well on small datasets.

When comparing results between the BiGRU-CRF and BiLSTM-CRF with name-entities, the BiGRU-CRF outperforms the model by 2-3%. Without the inclusion of name-entities, however, the BiGRU-CRF performs marginally better than the BiLSTM-CRF. Speculation as to why the BiGRU-CRF performs better regardless of the inclusion of name entities in training data stems from GRU's faster convergence speeds.

[2]It is not specified whether the model was a normal RNN or LSTM

4.6 Analysis

Our model occasionally outputs padding tokens instead of the 0s and 1s that denote having or not having a space. We suspect that this is due to how the CRF portion of the model is implemented and decoded. In our model, sentences are decoded from the end to the start using the Viterbi algorithm. We set the weights of the CRF such that the padding token cannot transition to any other state. If the algorithm mistakenly starts with a padding token, it is unable to recover. This problem might also be due to the fact that some of our training sentences are blank lines, which will appear as a series of padding tokens to the model.

The model tends to fail on compounds words[3]. (Note that | is a word delimiter.)
Input sentence: ...เท้าให้คล่องแคล่วและว่องไว...
Correct segmentation: ...| เท้า | ให้ | คล่องแคล่ว | และ | ว่องไว | ...
Model segmentation: ...| เท้า | ให้ | คล่อง | แคล่ว | และ | ว่อง | ไว |...
This is probably because each component in a compound word appears to be the same as a complete word.

The model also struggle with names of Thai royalty and nobles. theses names tend to be long, and have multiple components including ranks.
Input sentence: ...พระองค์เจ้ารพีพัฒนศักดิ์...กรมพระยาดำรงราชานุภาพ...
Correct segmentation: ... | พระองค์เจ้ารพีพัฒนศักดิ์ | ... | กรมพระยาดำรงราชานุภาพ | ...
Model segmentation: ...| พระองค์ | เจ้าร | พี | พัฒนศักดิ์ | ... | กรมพระยาดำรง | ราชานุภาพ | ...
(The two names refer to Prince Raphi Phatthanasak, Prince of Ratchaburi and Prince Tisavarakumarn, the Prince Damrong Rajanubhab.)

Another interresting behavior can be seen in web urls:
Input sentence http://www.bloggang.com/viewdiary.php?id=sabanngamonth=06-2008date=15group=2gblog=6
Correct segmentation http://www.bloggang.com/viewdiary.php?id=sabanngamonth=06-2008date=15group=2gblog=6|
Model segmentation http://www.bloggang.|com/viewdiary.php?id=sabanngamonth=06|-|2008date=15group=2gblog=6|
Despite the incorrect parsing, notice that the model has learned to parse English characters using punctuation such as full stop and hyphen.

5 Conclusion

Using the BiLSTM-CRF and BiGRU-CRF shows that an effective Thai segmentation model can be constructed despite the size of segmented Thai data. The results produced from the BiGRU-CRF without the inclusion of name entities is comparable to existing neural Thai segmentation baselines. Admittedly, time constraints did not facilitate exploration with dropout and hyper-parameter tuning to improve the model. The same could be said for the padding token problem mentioned in the previous section.

We suspect that improvements can be achieved by pre-training the model on unlabeled such as the Thai Wikipedia data. Another possible approach is to train a multitask model that

[3]A compound word is a word comprised of two separate words. There are three main types of compound words in Thai. The the type in the example is a combination of two semantically or phonetically similar words to emphasize or de-emphasize the meaning.

does both named entity recognition as well as segmentation to help improve the performance on segmenting names.

6 Additional Information

Mentor: Sahil Chopra

References

[1] Haruechaiyasak, Choochart, Sarawoot, Kongyoung. & Matthew, Dailey. (2008) A Comparative Study on Thai Word Segmentation Approaches. *Proceedings of ECTI-CON 2008*, pp.25-128.

[2] Pucktada Treeratpituk (2017). CutKum: Thai Word-Segmentation with LSTM in Tensorflow. May 5, 2017. See https://github.com/pucktada/CutKum

[3] Kittinaradorn, Rakpong, Chaovavanich, Korakot, Achakulvisut, Titipat & Kaewkasi, Chanwit. Deepcut: A Thai word tokenization library using Deep Neural Network. See https://github.com/rkcosmos/deepcut.

[4] Yao, Yushi & Huang, Zheng (2016) Bi-directional LSTM Recurrent Neural Network for Chinese Word Segmentation. A. Hirose et al. (eds.):, *ICONIP 2016, Part IV, LNCS 9950*, pp.45—353, 2016.

[5] Samih, Younes & Attia, Mohammed (2017) A Neural Architecture for Dialectal Arabic Segmentation *Proceedings of The Third Arabic Natural Language Processing Workshop (WANLP).* Valencia, Spain, pp.6-54, 2017.

[6] Deletion-based sentence compression using Bi-enc-dec LSTM - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/BI-LSTM-CRF-model_fig1_319186302 [accessed 7 Mar, 2019]

[7] Sahu, Sunil Kumar & Anand, Ashish. (2016) Recurrent neural network models for disease name recognition using domain invariant features. *Department of Computer Science at Cornell University.*