Engineering Journal issn: 0125-8281

Article

Semi-supervised Thai Sentence Segmentation Using Local and Distant Word Representations

Chanatip Saetia[1,a], Ekapol Chuangsuwanich[2,b], Tawunrat Chalothorn[3,c], and Peerapon Vateekul[1,d,*]

1 Chulalongkorn University Big Data Analytics and IoT Center (CUBIC), Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
2 Chula Intelligent and Complex Systems, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
3 Kasikorn Labs Co., Ltd., Kasikorn Business Technology Group, Nonthaburi 11120, Thailand
E-mail: [a]6170135321@student.chula.ac.th, [b]ekapol.c@chula.ac.th, [c]tawunrat.c@kbtg.tech, [d]peerapon.v@chula.ac.th (Corresponding author)

Abstract. A sentence is typically treated as the minimal syntactic unit used to extract valuable information from long text. However, in written Thai, there are no explicit sentence markers. Some prior works use machine learning; however, a deep learning approach has never been employed. We propose a deep learning model for sentence segmentation that includes three main contributions. First, we integrate n-gram embedding as a local representation to capture word groups near sentence boundaries. Second, to focus on the keywords of dependent clauses, we combine the model with a distant representation obtained from self-attention modules. Finally, due to the scarcity of labeled data, for which annotation is difficult and time-consuming, we also investigate two techniques that allow us to utilize unlabeled data: Cross-View Training (CVT) as a semi-supervised learning technique, and a pre-trained language model (ELMo) to improve word representation. In the experiments, our model reduced the relative error by 7.4% and 18.5% compared with the baseline models on the Orchid and UGWC datasets, respectively. Ablation studies revealed that the main contributing factor was adopting n-gram features, which were further analyzed using the interpretation technique and indicated that the model utilizes the features in the same way that humans do.

Keywords: Natural language processing, machine learning, artificial neural networks, sequence tagging model, Thai sentence segmentation, Thai language.

ENGINEERING JOURNAL Volume 25 Issue 6
Received 15 December 2020
Accepted 5 June 2021
Published 30 June 2021
Online at http://engj.org/
DOI:10.4186/ej.2021.25.6.15

DOI:10.4186/ej.2021.25.6.15

1. Introduction

Machine translation, automatic text summarization, dependency parsing, and semantic parsing are useful for processing, analyzing, and extracting meaningful information from text. These tasks require a basic unit that has a simple grammatical structure to reduce the tasks' complexity. For example, dependency parsing [1], which extracts a syntactic structure for language understanding, needs to consider every word pair in the text to assign a relation. Thus, the complexity of dependency parsing depends on the input text's sequence length. If the text is segmented into smaller parts, the task will be easier to perform. However, the basic unit should be not only as small as possible but also required to be complete in itself. For instance, if the basic unit is too short and does not contain sufficient information, many meaningful relations in dependency parsing will not be extracted inside the basic unit. Thus, the basic unit should be small yet contain complete meaning to make the mentioned tasks more efficient.

A sentence is raised as a basic unit because a sentence is always complete in itself. Moreover, a sentence can also be easily extracted from raw text because the sentence boundaries in English are easily identified by a period (".") [2]. Many prior works require a sentence to perform their tasks. For example, in machine translation, a sentence pair is required for supervised training [3, 4, 5]. Meanwhile, many automatic text summarization works treat a sentence as one item of information and select the important ones to be summarized [6, 7, 8]. Dependency parsing also requires a sentence as an input text to extract its syntactic structure that the machine understands [1, 9, 10].

However, there is no explicit end-of-sentence marker for identifying the sentence boundary in some written languages, such as Thai, Arabic, Khmer, and Lao [11]. Therefore, extracting sentences from raw text in these languages is not trivial. For example, "He wishes to buy 4 ingredients for cooking an omelet. Therefore, he goes shopping and buys an egg, milk, salt, and pepper." The text can be segmented with a period "." into two sentences "He wishes to buy 4 ingredients for cooking a fried egg." and "Therefore, he goes shopping and buys an egg, milk, salt, and pepper." Meanwhile, in Thai, the same text is "เขาต้องการที่จะซื้อส่วนประกอบ 4 อย่างสำหรับทำไข่เจียว ดังนั้นเขาจึงไปซื้อไข่ นม เกลือ และพริกไทย." Note that there is no punctuation or even a word to indicate where the text should be segmented. Although most Thai people usually use a space character as a sentence boundary, the illustrated text shows that only one out of six space characters is a sentence boundary. Therefore, there is no explicit marker for identifying sentence boundaries to segment the text, especially for the exemplified case.

Prior works on Thai sentence segmentation have adopted traditional machine learning models to predict where a sentence boundary is in the text. The authors of [11, 12, 13] proposed traditional models to determine whether a considered space is a sentence boundary based on the words and their part of speech (POS) near the space. Although a space is usually considered essential as a sentence boundary marker in Thai, approximately 23% of the end of sentences is not a space character in a news domain corpus [14]. Thus, Zhou N. et al. [14] proposed a conditional random field (CRF)-based model with n-gram features to predict which word is the sentence boundary. This work considered Thai sentence segmentation as a sequence tagging problem similar to named entities recognition and part-of-speech tagging. Each word in the text will be classified whether it is the end of a sentence or not, as shown in Fig. 1. With a CRF module [15], the model extracts sentence-level tag information, where each prediction in a sequence considers the previous predicted tags instead of only the input words. Meanwhile, the n-gram [16], which is an input feature, is constructed from a combination of words around the considered position. This method achieves the state-of-the-art result for Thai sentence segmentation and achieves greater accuracy than other models by approximately 10% on the Orchid dataset [17].

Several deep learning approaches have been applied in various tasks of natural language processing (NLP), including the long short-term memory (LSTM) [18], self-attention [19], and other models. To tackle the sequence tagging problem, Huang Z. et al. [20] proposed a deep learning model called Bi-LSTM-CRF, which integrates a CRF module to gain the benefit of both deep learning and traditional machine learning approaches. In their experiments, the Bi-LSTM-CRF model achieved an improved level of accuracy in many NLP sequence tagging tasks, such as named entity recognition, POS tagging and chunking. This model is also used as a base (backbone) model for many works that achieve promising accuracy in sequence tagging tasks. [21, 22, 23, 24, 25]

In this work, two models are chosen as baseline models. First, the Bi-LSTM-CRF model is adopted as our baseline and backbone model because many sequence tagging works also apply the model as a backbone model and yield respectable performance. Second, the CRF model, which achieved the best result on the Thai sentence segmentation task [14], is also treated as another baseline model to compare with prior works.

This work makes three contributions to improve Bi-LSTM-CRF for Thai sentence segmentation. These contributions apply the suitable deep learning modules carefully to tackle various problems of this task. Each contribution is described as follows.

First, we propose adding n-gram embedding to Bi-LSTM-CRF due to its success in [14]. To integrate n-gram features in Bi-LSTM-CRF, the feature is embedded into a dense embedding vector trained along with the model. With the n-gram embedding addition, it can extract a local repre-

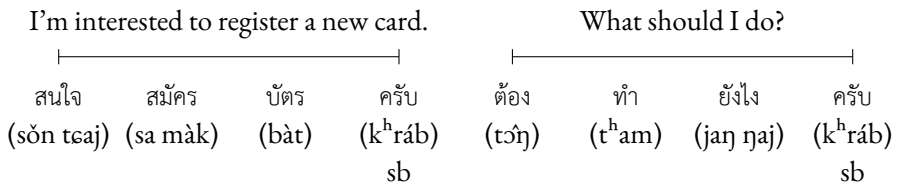

Fig. 1. An example of a labeled paragraph. Here, *sb* represents a sentence boundary.

sentation from n-gram embedding, which helps in capturing word groups that exist near a sentence boundary. Although Jacovi A. et al. [26] reported that a convolutional neural network (CNN) can be used as an n-gram detector to capture local features, we chose n-gram embedding over a CNN due to its better accuracy, as will be shown in Section B.

Second, we propose adding distant representation to the model via a self-attention mechanism[19], which can focus on the keywords of dependent clauses that are far from the considered word. Self-attention has been used in many recent state-of-the-art models, most notably the transformer [19] and Bidirectional Encoder Representations from Transformers (BERT) [27]. BERT has outperformed Bi-LSTM on numerous tasks, including question answering and language inference. Therefore, we choose to use self-attention modules to extract distant representations along with local representations to improve model accuracy.

Third, we also apply two techniques to utilize unlabeled data: semi-supervised learning and a pre-training method, which are essential for low-resource languages such as Thai, for which annotation is costly and time-consuming. The first technique is semi-supervised learning [28]. Many semi-supervised learning approaches have been proposed in the computer vision [29, 30] and natural language processing [31, 32, 33] fields. Our choice for semi-supervised learning to enhance model representation is Cross-View Training (CVT) [33]. Clark K. et al. [33] claim that CVT can improve the representation layers of the model, which is our goal. However, CVT was not designed to be integrated with self-attention and CRF modules; consequently, we provide a modified version of CVT in this work.

Instead of using only CVT to improve the representation with unlabeled data, the pre-training method is also adopted in our model. There are many proposed pre-trained language models in the field of NLP [27, 34, 35]. BERT [27] or other variant models of BERT are usually a part of the state-of-the-art methods for many tasks [21, 36, 37]. However, BERT models require a large amount of GPU memory in the training process. Therefore, ELMo, which needs less GPU memory than BERT [38], is chosen in this work due to our resource limitations. Note that some variants of BERT models are also optimized for less memory usage but require additional techniques, such as knowledge distillation [38] and factorized embedding parameterization [39], which requires further investigation on the Thai corpus.

In conclusion, we propose three contributions, which are summarized as follows:

- **Local representation** is extracted from n-gram embedding to capture word groups around a sentence boundary and improve the accuracy of sentence segmentation.
- A self-attention mechanism is adopted as a **distant representation** to capture the keywords of dependent clauses that are far from the considered word.
- To utilize unlabeled data, **semi-supervised learning** and a pre-training method are adopted to improve the model generalization.

Based on the three contributions above, the experiment was conducted on two Thai datasets, Orchid [17] and UGWC [40], to evaluate our Thai sentence segmentation model. In this case, our model achieves F1 scores of 92.5% and 89.9% on Orchid and UGWC, respectively, and it outperforms all the baseline models. Moreover, we also perform ablation studies, which add each proposed contribution sequentially to observe their individual effect on the performance. The ablation studies found that local representation (n-grams) yields the largest improvement on Thai sentence segmentation; thus, its effect is further analyzed using interpretation techniques.

There are five sections in the remainder of this paper. Section 2 reviews the related works on Thai sentence segmentation, English punctuation restoration and introduces the original CVT. Section 3 describes the proposed model architecture and the integration of cross-view training. The datasets, implementation process and evaluation metrics are explained in Section 4. The results of the experiments are discussed in Section 5. Finally, Section 6 concludes the paper.

2. Related Works

This section includes three subsections. The first subsection concerns Thai sentence segmentation, which is the main focus of this work. The task of English punctuation restoration, which is similar to our main task, is described in the second subsection. The last subsection describes the original Cross-View Training initially proposed in [33].

2.1. Thai Sentence Segmentation

In most languages, a sentence boundary can be determined via a specific character. For example, a period is used as a

marker to identify a sentence boundary in English. However, In Thai and other languages (Arabic, Khmer, and Lao), texts do not contain markers that definitively identify sentence boundaries. Therefore, segmenting text in those languages into sentences cannot be performed simply by splitting a text at some specific characters.

Although there is no marker for definitively identifying the sentence boundary in Thai, Thai people write the text using a space as a vital element that separates text into sentences. Thus, a space is an important signal for splitting the text into sentences. However, there are three ways that spaces are used in this context [41]: before and after an interjection, before conjunctions, and before and after numeric expressions. Therefore, segmenting the text into sentences using a space in Thai is not trivial and cannot be solved by only a rule-based strategy.

Previous works from [12, 13, 11] have focused on disambiguating whether a space functions as the sentence boundary. These works extract contextual features from words and POS around the space. Then, the obtained features around the corresponding space are input into traditional machine learning models to predict whether space is a sentence boundary. Moreover, to improve the model accuracy, Thai grammar rules [42, 43] are also integrated with the statistical models.

Although a space is usually considered essential as a sentence boundary marker, approximately 23% of the end of sentences is not a space character in a news domain corpus [14]. Hence, there are prior works that consider this task as a sequence tagging problem [14, 44]. In other words, a space is only considered as one of the possible candidates for a sentence boundary. Zhou N. et al. [14] proposed a word sequence tagging CRF-based model in which all words can be considered as candidates for the sentence boundary. The CRF-based model [15], which is extracted from n-grams around the considered word, achieves a F1 score of 91.9%, which is approximately 10% higher than the F1 scores achieved by other models [11, 12, 13] on the Orchid dataset, as mentioned in [14]. Nararatwong R. et al. [45] extend this model using a POS-based word-splitting algorithm to increase identifiable POS tags, resulting in better model accuracy. Because the focus of this work is adjusting the POS as a postprocessing method, which is an input of the model instead of proposing a new sentence segmentation model, this work will not be considered in this paper.

In this work, we apply the concept of word sequence tagging and compare it with two baselines: the CRF-based model with n-gram features, which is currently the state-of-the-art for Thai sentence segmentation, and the Bi-LSTM-CRF model, a common deep learning model for sequence tagging tasks.

2.2. English Punctuation Restoration

Most languages use a symbol that functions as a sentence boundary; however, a few do not use sentence markers including Thai, Arabic, Lao and Khmer. Thus, few studies have investigated sentence segmentation in raw text. However, studies on sentence segmentation, which is sometimes called sentence boundary detection, are still found in a speech recognition field [46]. The typical input to speech recognition model is simply a stream of words. If two sentences are spoken back to back, by default, a recognition engine will treat it as one sentence. Thus, sentence boundary detection is also considered a punctuation restoration task in speech recognition because when the model attempts to restore the period in the text, the sentence boundary position will also be defined.

Punctuation restoration not only provides a minimal syntactic structure for natural language processing similar to sentence boundary detection but also dramatically improves the readability of transcripts. Therefore, punctuation restoration has been extensively studied. Many approaches have been proposed for punctuation restoration that use different features, such as audio and textual features. Moreover, punctuation restoration is also considered to be a different machine learning problem, such as word sequence tagging and machine translation.

Focusing only on textual features, there are two main approaches, namely, word sequence tagging and machine translation. For the machine translation approach, punctuation is treated as just another type of token that needs to be recovered and included in the output. The methods in [47, 48, 49] restore punctuation by translating from unpunctuated text to punctuated text. However, our main task, sentence segmentation, is an upstream task in text processing, unlike punctuation restoration, which is considered a downstream task. Therefore, the task needs to operate rapidly; consequently, we focus only on the sequence tagging model, which is less complex than the machine translation model.

In addition to those machine translation tasks, both traditional approaches and deep learning approaches must solve a word sequence tagging problem. Of the traditional approaches, contextual features around the considered word were used to predict the following punctuation in the n-gram [50] and CRF model approaches [51, 52]. Meanwhile, in the deep learning approaches, a deep convolutional neural network [53], a T-LSTM (Textual-LSTM) [54] and a bidirectional LSTM model with an attention mechanism, called T-BRNN [55], have been adopted to predict a punctuation sequence from the word sequence. Meanwhile, Kim S. [56] proposed integrating layerwise multi-head attention in a Bi-LSTM to improve the model's accuracy and outperform the previous one. Yi J. et al. [21] adopt the pre-training method to improve the accuracy of the model. This work selects the

Bi-LSTM-CRF as a backbone model. Meanwhile, the input words are embedded by a pre-trained BERT before feeding to the backbone model. Because POS tags are helpful for this task, POS tags are always fed to the model in this task. However, POS tags are generated from the machine learning model which is usually error-prone. Thus, Yi J. et al. [21] also adopted adversarial transfer learning to imitate the effect of an error from predicted POS tags. As a result, their proposed model gains a 9.2 % F1 score improvement compared to prior works.

In our work, a Bi-LSTM and an attention module are also added to the model. To enhance the sentence segmentation model, we also applied a pre-training method, which achieves a significant improvement in English punctuation restoration. However, instead of using BERT similar to [21], we apply contextual text representations (ELMo) to leverage information from the unlabeled data due to our resource limitations.

2.3. Cross-View Training

CVT [33] is a semi-supervised learning technique whose goal is to improve the model representation using a combination of labeled and unlabeled data. During training, the model is trained alternately with one mini-batch of labeled data and B mini-batches of unlabeled data.

Labeled data are input into the model to calculate the standard supervised loss for each mini-batch and the model weights are updated regularly. Meanwhile, each mini-batch of unlabeled data is selected randomly from the pool of all unlabeled data; the model computes the loss for CVT from the mini-batch of unlabeled data. This CVT loss is used to train auxiliary prediction modules, which see restricted views of the input, to match the output of the primary prediction module, which is the full model that sees all the input. Meanwhile, the auxiliary prediction modules share the same intermediate representation with the primary prediction module. Hence, the intermediate representation of the model is improved through this process.

Similar to the previous work, we also apply CVT to a sequence tagging task. However, our model is composed of self-attention and CRF modules, which were not included in the sequence tagging model in [33]. The previous CVT was conducted on an LSTM using the concepts of forward and backward paths, which are not intuitively acquired by the self-attention model [19] attending all words of an input at the same time.

Moreover, the output used to calculate CVT loss was generated by the softmax function, which does not operate with CRF. Thus, in our study, it is necessary for both the primary and auxiliary prediction modules to be constructed differently from the original modules.

3. Proposed Method

In this section, we describe our proposed method in two subsections. The first subsection specifies the model architecture and the details of each module. Our first and second contributions, which are local and distant representations, are mainly described in this subsection. Meanwhile, the second subsection expounds on how the model is trained with unlabeled data through the modified CVT and using ELMo, which is our third contribution.

3.1. Model Architecture

In this work, the model predicts the tags $\vec{y} = [y_1, y_2, \ldots, y_N]$, $\forall y \in Y$ for the tokens in a word sequence $\vec{x} = [x_1, x_2, \ldots, x_N]$ where N is the sequence size and x_t, y_t denote the token and its tag at timestep t, respectively. The word sequence is fed into the model at the same time to provide sequential information of the input text to the model. Each token x_t consists of a word, its POS and its type. There are five defined word types: English, Thai, punctuation, digits, and spaces.

The tag set Y is populated based on the considered task. In Thai sentence segmentation, the assigned tags are sb and nsb; sb denotes that the corresponding word is a sentence boundary considered as the end of a sentence, while and nsb denotes that the word is not a sentence boundary.

Our model architecture is based on Bi-LSTM-CRF, as shown in Fig. 2. The model is divided into three modules. The first, low-level module, consists of two separate structures: local (n-gram features) and distant (self-attention) structures. This module is designed to utilize an unlabeled dataset via cross-view training, which will be described in Section 3.2. The module gives the two separate outputs from two structures, which are only concatenated to feed to the next module. Meanwhile, the second module, which is the high-level module, will combine these two outputs to produce the higher-level representation from a stack of Bi-LSTM and self-attention, which helps the model learn the context from the whole word sequence. The final module, the prediction module, is responsible for predicting the tags $\vec{y}$. Each module is described more completely in the next three subsections.

3.1.1. Low-level Module

A sequence of word tokens is input into the low-level module. The input tokens pass through two structures. The first structure generates sequence of local representation vectors $\mathbf{R}_{local} = [\vec{r}_{1,local}, \vec{r}_{2,local}, \ldots, \vec{r}_{N,local}]$ which is embedding vectors of n-gram features. After obtaining a sequence of representation vectors, the local representation vectors are fed to the Bi-LSTM to obtain the recurrent representation vectors $\mathbf{R}_{recurrent} = [\vec{r}_{1,recurrent}, \vec{r}_{2,recurrent}, \ldots, \vec{r}_{N,recurrent}]$, as shown in

DOI:10.4186/ej.2021.25.6.15

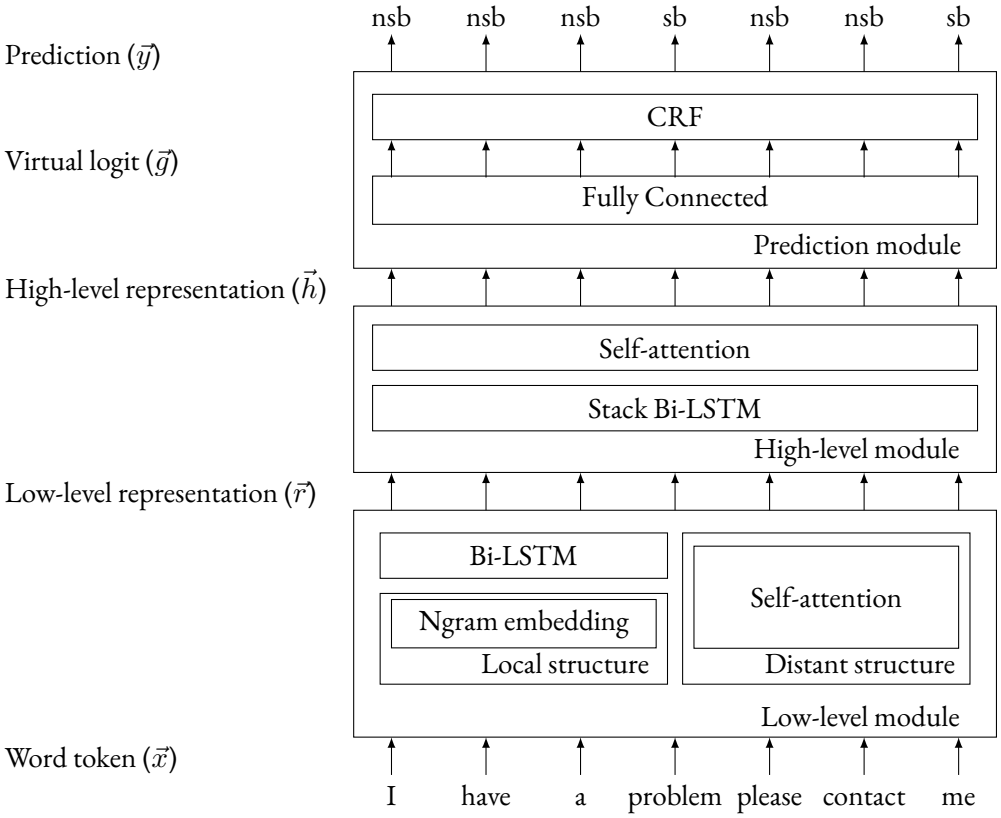

Fig. 2. Model architecture that integrates local and distant representation. This model is composed of three main modules: a low-level module, a high-level module and a prediction module. In the low-level module, two structures (local and distant) are responsible for extracting different features.

(1):

$$\mathbf{R}_{recurrent} = \text{BiLSTM}(\mathbf{R}_{local}) \tag{1}$$

The second structure generates low-level distant representation vectors $\mathbf{R}_{distant} = [\vec{r}_{1,distant}, \vec{r}_{2,distant}, \ldots, \vec{r}_{N,distant}]$ which produced by Self-attention. Then, the recurrent and distant representation vectors are concatenated to form the low-level representation vector $\mathbf{R} = [\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N]$, as shown in (2), where $\oplus$ represents a concatenation between two vectors:

$$\vec{r}_t = \vec{r}_{t,recurrent} \oplus \vec{r}_{t,distant} \tag{2}$$

Local structure This structure is shown as the left submodule of the low-level module in Fig. 2. It extracts the local representation vectors $\mathbf{R}_{local}$. Its input tokens are used to create n-gram tokens, which are unigrams x_a, bigrams (x_a, x_b), and trigrams (x_a, x_b, x_c). Each n-gram token is represented as an embedding vector, which is classified as a unigram embedding vector $\vec{e}_{uni}$, a bigram embedding vector $\vec{e}_{bi}$ or a trigram embedding vector $\vec{e}_{tri}$. Each vector $\vec{e}_{gram}$ is mapped from a token by gram embedding $\text{Embedding}_{gram}(x)$, which is a concatenated vector of the word embedding $\text{Word}_{\text{gram}}(x)$, POS embedding $\text{POS}_{\text{gram}}(x)$ and type embedding $\text{Type}_{\text{gram}}(x)$, as shown in (3):

$$\begin{aligned}\text{Embedding}_{\text{gram}}(x) = \text{W}_{\text{gram}}(x) \oplus \text{POS}_{\text{gram}}(x) \\ \oplus \text{Type}_{\text{gram}}(x)\end{aligned} \tag{3}$$

Unigram embedding vector $\text{Embedding}_{uni}(x)$ is included along with contextual pre-trained vector $ELMo_{uni}(x)$ from ELMo, as shown in (4):

$$\begin{aligned}\text{Embedding}_{\text{uni}}(x) = \text{W}_{\text{uni}}(x) \oplus \text{POS}_{\text{uni}}(x) \\ \oplus \text{Type}_{\text{uni}}(x) \oplus \text{ELMo}_{\text{uni}}(x)\end{aligned} \tag{4}$$

Each n-gram token at timestep t is generated by the previous, present and next token (x_{t-1}, x_t, x_{t+1}) and embedded into vectors as shown in (5), (6), and (7) for unigram, bigram, and trigram consecutively.

$$\vec{e}_{t,uni} = \text{Embedding}_{\text{uni}}(x_t) \tag{5}$$

$$\vec{e}_{t,bi} = \text{Embedding}_{\text{bi}}(x_{t-1}, x_t) \tag{6}$$

$$\vec{e}_{t,tri} = \text{Embedding}_{\text{tri}}(x_{t-1}, x_t, x_{t+1}) \tag{7}$$

At each timestep t, a local representation vector $\vec{r}_{t,local}$ is combined from the n-gram embedding vectors generated from the context around x_t. A combination of embedding vectors, which is used to construct a local representation vector, is shown in (8). A combination consists of the unigram,

bigram, and trigram embedding vectors at timesteps $t-1$, t and $t+1$ and it is a concatenation of all the embedding vectors:

$$\begin{aligned}\vec{r}_{t,local} =& \vec{e}_{t-1,uni} \oplus \vec{e}_{t,uni} \oplus \vec{e}_{t+1,uni} \\ & \oplus \vec{e}_{t-1,bi} \oplus \vec{e}_{t,bi} \oplus \vec{e}_{t+1,bi} \\ & \oplus \vec{e}_{t-1,tri} \oplus \vec{e}_{t,tri} \oplus \vec{e}_{t+1,tri}\end{aligned} \tag{8}$$

Distant structure The distant structure, which is a self-attention module SelfAttention, is shown in Fig. 2 on the right side of the low-level module. The structure extracts low-level distant representation vectors $\mathbf{R}_{distant}$ from a sequence of unigram embedding vectors $\mathbf{E}_{uni}$, as shown in (9). In this work, the self-attention mechanism is similar to the one used in Transformer [19], which is widely used in NLP tasks [36, 57, 58]. The self-attention module is a scaled dot-product attention [19], where key, query, and value vectors are the linear projections of the unigram embedding vectors shown in Fig. 3. The linear transformations for key, query, and value are learned separately and updated in the model through backpropagation. The output vector, which is the scaled dot-product attention at each timestep, is concatenated with the input vector $\vec{e}_{t,uni}$ and projected by a linear transformation. This projected vector is the output vector of a self-attention module, which is a low-level distant representation vector.

$$\mathbf{R}_{distant} = \text{SelfAttention}(\mathbf{E}_{uni}) \tag{9}$$

3.1.2. High-level Module

The low-level representation vectors **R** are used as the input for this module, which outputs the high-level representation vectors **H** whose calculation is shown in (10). The high-level module, as shown in Fig. 2, is composed of a stacked bidirectional LSTM (StackBiLSTM) and a self-attention modules. A stacked bidirectional LSTM contains K layers of bidirectional LSTMs in which the output from the previous bidirectional LSTM layer is the input of the next bidirectional LSTM layer. The self-attention part of this structure is the same as that in the low-level distant structure. The self-attention module helps to generate the high-level distant representation vectors that are output by the high-level module.

$$\mathbf{H} = \text{SelfAttention}(\text{StackBiLSTM}(\mathbf{R})) \tag{10}$$

3.1.3. Prediction Module

The prediction module is the last module. It includes two layers: a fully connected layer (NN) and a CRF layer (CRF). In the fully connected layer, the output vectors from the high-level module are projected by a linear transformation as shown in (11). The purpose of this layer is to create the virtual logit vectors $\mathbf{G} = [\vec{g}_1, \vec{g}_2, ..., \vec{g}_N]$, which represent the probability distribution for CVT, as discussed in Section 3.2. Therefore, the number of dimensions of logits equals the number of possible tags in each task:

$$\vec{g}_t = \text{NN}(\vec{h}_t) \tag{11}$$

The CRF layer is responsible for predicting the tag y_t of a token at each timestep, as shown in (12). The layer receives a sequence of virtual logit vectors (**G**) as input and then decodes them to a sequence of tags $\vec{y}$ using the Viterbi algorithm [59].

$$\vec{y} = \text{CRF}(\mathbf{G}) \tag{12}$$

3.2. Training Process

To train the sentence segmentation model, the process is split into two steps. First, the language model (ELMo) is trained with unlabeled data. The pre-trained language model is treated as a part of input vectors for the model. We pre-train ELMo on our unlabeled dataset. The second step is to train the model with CVT for the sentence segmentation problem, which described as follows.

As discussed in Section 2.3, CVT requires primary and auxiliary prediction modules for training with unlabeled data to improve the representation. Thus, we construct both types of prediction modules for our model. The flow of unlabeled data, which is processed to obtain a prediction by each module, is shown in Fig. 4. The output of each prediction module is transformed into the probability distribution of each class by the softmax function and then used to calculate $Loss_{CVT}$, as shown in (13).

$$\begin{aligned}Loss_{CVT} = \frac{1}{|D|} \sum_{t \in D} & D_{\text{KL}}(\vec{p}_{d,primary}, \vec{p}_{t,local}) \\ & + D_{\text{KL}}(\vec{p}_{t,primary}, \vec{p}_{t,distant})\end{aligned} \tag{13}$$

The $Loss_{CVT}$ value is based on the Kullback–Leibler divergence (KL divergence, D_{KL}) between the probability distribution of the primary $\vec{p}_{t,primary}$ output and those of two auxiliary modules, $\vec{p}_{t,local}$ and $\vec{p}_{t,distant}$, where $t \in [1, ..., N]$. To calculate the loss, the KL divergence at each timestep is averaged when the timesteps are dropped timesteps D, which is described in Section 3.2.2. The calculation of the primary output ($\vec{p}_{t,primary}$) and both auxiliary outputs ($\vec{p}_{t,local}$ and $\vec{p}_{t,distant}$), which are used in the $Loss_{CVT}$ calculation, are described in the following subsections.

3.2.1. Primary Prediction Module

In [33], the output of the primary prediction module is acquired from the last layer and used to predict tags. However, our model uses a CRF layer to decode the tags instead

DOI:10.4186/ej.2021.25.6.15

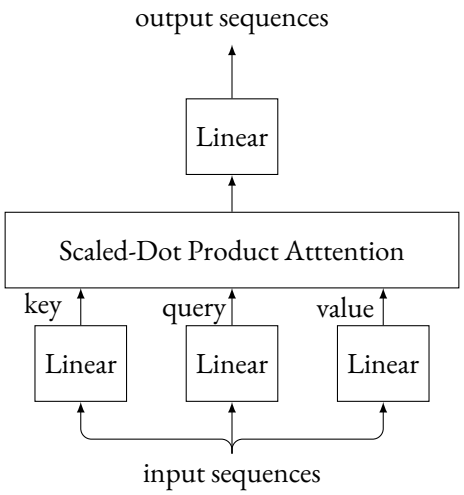

Fig. 3. The architecture of a self-attention module. This module mainly contains Scaled-Dot Product Attention, which requires three inputs: Key, Query and Value. Those inputs are generated from the same input sequence but projected by different linear transformations.

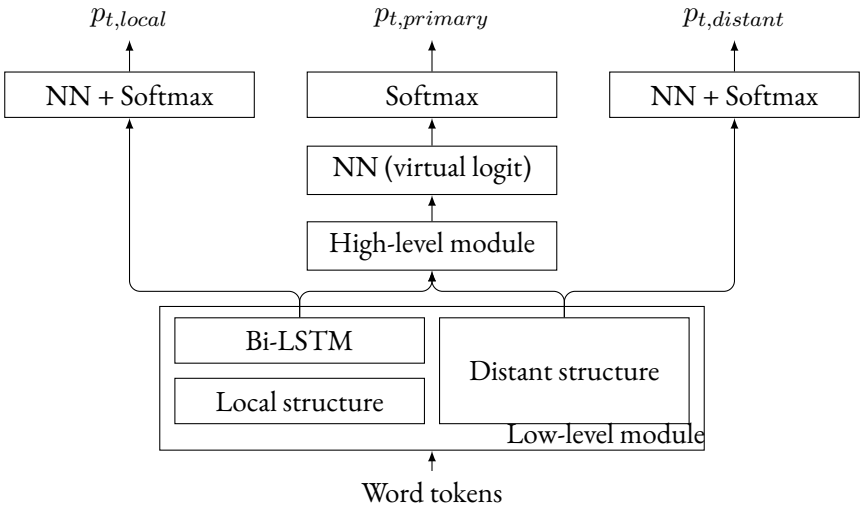

Fig. 4. Two auxiliary predictions ($\vec{p}_{t,local}$ and $\vec{p}_{t,distant}$) are obtained from the local and distant structures in the low-level module. The primary prediction $\vec{p}_{t,primary}$ is obtained from the virtual logit vector $\vec{g}_t$

of the softmax function. Thus, in semi-supervised learning, the probability distribution of the primary prediction module should be acquired from the CRF layer. However, the Viterbi algorithm, which is used for decoding, gives only the best combination for the prediction, but does not provide the probability distribution. Normally, the distribution from the CRF is calculated by a forward-backward algorithm [60] which is time consuming. To reduce the training time, the probability distribution of the primary prediction module $\vec{p}_{t,primary}$ is obtained from the output of the softmax function (Softmax), whose input is a virtual logit vector $\vec{g}_t$, as shown in (14).

$$\vec{p}_{t,primary} = \text{Softmax}(\vec{g}_t) \tag{14}$$

3.2.2. Auxiliary Prediction Module

Two auxiliary views are included to improve the model. The first view is generated from a recurrent representation vector $r_{t,recurrent}$ to acquire the local probability distribution $\vec{p}_{t,local}$, where $t \in [1, ..., N]$. The second view is generated from the low-level distant representation vectors $r_{t,distant}$ to acquire the probability distribution of a distant structure in the low-level module $\vec{p}_{t,distant}$, where $t \in [1, ..., N]$. By generating the views from these representation vectors separately, the local and distant structures in the low-level module can improve equally.

Although both representation vectors are used separately to create auxiliary views, the input of each structure is still not restricted, unlike [33], where the input is restricted to only previous or future tokens. Because BERT, which is trained by the masked language model, outperforms OpenAI GPT-2 [61], which uses an autoregressive approach

for training as reported in [27], we adopt the concept of the masked language model [62] to obtain both auxiliary views. This approach allows the representation to fuse the left and the right context, which results in a better representation. By using the masked language model, some tokens at each timestep are randomly dropped and denoted as removed tokens $\langle REMOVED \rangle$; then, the remaining tokens are used to obtain auxiliary predictions in the dropped timesteps $D = \{d \in N | d \text{ is a dropped timestep}\}$, as shown in Fig. 5. The details of both auxiliary prediction modules are described below.

Local auxiliary module For recurrent representation vectors, if one of the tokens is dropped, the related n-gram tokens that include the dropped tokens will also be dropped. For example, if (x_t) is dropped, (x_{t-1}, x_t) and (x_t, x_{t+1}) will also be dropped as removed tokens in the case of a bigram. The remaining n-gram tokens are then used to obtain the recurrent representation vectors at the dropped timesteps. Then, the vectors are provided as an input to the softmax function to obtain the probability distribution of the first auxiliary prediction module, as shown in (15).

$$\vec{p}_{d,local} = \text{Softmax}(\text{NN}(\vec{r}_{d,recurrent})) \tag{15}$$

Distant auxiliary module In the other auxiliary prediction module, a sequence of the low-level distant representation vectors is generated and some tokens are dropped. This sequence of vectors is also input into the Softmax function, just as in the first auxiliary prediction module, and the output is another probability distribution, which is the second auxiliary prediction, as shown in (16).

$$\vec{p}_{d,distant} = \text{Softmax}(\text{NN}(\vec{r}_{d,distant})) \tag{16}$$

3.3. Interpretation Process

In this section, we discuss our method to interpret our model decision process by identifying important n-gram features. These selected features can be compared to the human annotations or the label of the dataset in order to further analyze the model.

The process of interpretation is inspired by a gradient technique [63]. The intuition of this method is to identify how the change in each feature affects the model. Simonyan K. et al. [63] shows that the score, which indicates the effect of the change in each feature, can be derived from a gradient of the feature. Therefore, the gradient of the feature can indicate the importance of the feature in the model. This method is widely used in many NLP tasks [64, 65]. In this work, the model interpretation is discovered via the gradient of n-gram features.

Based on (8), nine types of features, which are unigram, bigram, and trigram at timesteps $t-1$, t and $t+1$, are used as input features for our model. Thus, the gradient of these nine types of n-gram features are calculated. The gradient $\nabla \vec{e}_{t+pos,gram}$ is the derivative of the output of the model y with respect to $\vec{e}_{t+pos,gram}$ where $gram \in \{uni, bi, tri\}$ is the token type and $pos \in \{-1, 0, +1\}$ is the relative position from the current timestep, as shown in (17).

$$\nabla \vec{e}_{t+pos,gram} = \frac{\partial y}{\partial \vec{e}_{t+pos,gram}} \tag{17}$$

After that, the score of each feature $u_{pos,gram}$ is calculated, as shown in (18). The score is the summation of all the gradients of tokens $x_{t+pos,gram}$ that are $u_{pos,gram}$ over all timesteps $t \in \{1, ..., N\}$ in all documents D. This score is used to measure the importance of each feature. When the summed gradient of the feature is high, that feature is important in contributing to the model's predictions. On the other hand, if the summed gradient of a feature is low, that feature is not necessary for the model's prediction.

$$\text{Score}\,(u_{pos,gram}) = \sum_{t=1}^{D} \sum^{N} \nabla \vec{e}_{t+pos,gram} \quad \text{where } x_{t+pos,gram} = u_{pos,gram} \tag{18}$$

To compare the model to humans, two lists of tokens are created from the computed score and labels. Then, the intersection of both lists is used to compare the similarity. The first list $\mathbf{T}^{(model)}_{pos,gram}$ includes the top 500 features which have the highest computed score $Score(u_{pos,gram})$ from the gradient. The second list $\mathbf{T}^{(label)}_{pos,gram}$ contains of top 500 features with the highest frequency as a sentence boundary in the training set. After that, the features in the intersection between two lists are counted, as shown in (19), where $\|\mathbf{A}\|$ is a number of instances in a set $\mathbf{A}$. The counted number Count $(pos, gram)$ will be used for further analysis in Section 5.1.1.

$$\text{Count}\,(pos, gram) = \|\mathbf{T}^{(model)}_{pos,gram} \cap \mathbf{T}^{(label)}_{pos,gram}\| \tag{19}$$

4. Experimental Setup

4.1. Datasets

Two datasets are used in the experiments as described in the following subsections. The structure of the datasets is the same as the one illustrated in Fig. 1. Each passage of the datasets is segmented into a word sequence. In this case, the method for word segmentation in each dataset is different and will be described in the following subsections. Then, each word is tagged manually by a human if the word is the end of a sentence or not.

The statistics of the preprocessed data are shown in Table 1, including the number of sequences, the number of words, and the number of vocabulary words in each dataset.

DOI:10.4186/ej.2021.25.6.15

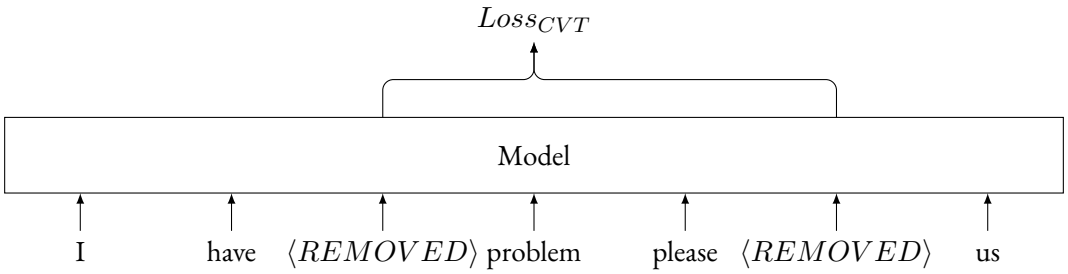

Fig. 5. Words are dropped (denoted as $\langle REMOVED \rangle$) randomly. Those positions are used to calculate $Loss_{CVT}$ and update the auxiliary prediction modules to improve the model.

Table 1. The number of passages and vocabulary words in each dataset. The labeled and unlabeled data are separately counted and shown in the rows.

Dataset	Orchid (Thai)			UGWC (Thai)		
	# passages	# words	# vocab	# passages	# words	# vocab
Labeled data	3,427	685,319	17,047	48,374	1,242,118	46,463
Unlabeled data	-	-	-	96,777	40,431,319	81,932
Labeled + Unlabeled data	-	-	-	145,151	41,673,437	109,415

Note: There are no unlabeled data in the Orchid dataset due to the lack of the same word segmentation and POS tag set.

4.1.1. Orchid [17]

This dataset is a Thai part-of-speech-tagged dataset containing 10,864 sentences. In the corpus, text was separated into paragraphs, sentences, and words hierarchically by linguists. Each word was also manually assigned a POS by linguists. A sample paragraph in this dataset is shown in Fig. 6 These data include no unlabeled data with the same word segmentation and POS tag set. Hence, we do not execute CVT on this dataset.

Our data preprocessing on the ORCHID corpus was similar to that in [14]: all the comments are removed, and the data are partitioned into 10 parts containing equal numbers of sentences to support 10-fold cross-validation. Each training set is split into one part used for validation and the rest is used for model training. Subsequently, all the words in each dataset are concatenated and then separated into sequences with 200 words per instance. Each sequence always begins with the first word of a sentence. If a sequence ends with an unfinished sentence, the next sequence starts with that complete sentence.

4.1.2. UGWC (User-Generated Web Content) [40]

This Thai dataset includes many types of labeled data useful in sentence segmentation tasks. The raw text was generated by users having conversations in the financial domain and were acquired mainly by crawling social sites. The labeled data for sentence segmentation were manually annotated by linguists using the definitions in [40]. A sample paragraph in this dataset is shown in Fig. 7

At the time of this study, the dataset was extended from that in [40]; the data were collected from January 2017 to December 2017. The labeled dataset includes 48,374 passages. To support semi-supervised learning, the first 3 months of data (96,777 passages) are unlabeled.

Because the data stem from social media, some existing text cannot be considered a part of any sentence, such as product links, symbols unrelated to sentences, and space between sentences. These portions were not originally annotated as sentences by the linguists. However, in this work, we treat these portions as individual sentences and tag the last word of each fraction as the sentence boundary.

For evaluation purposes, the collection of passages in this dataset is based on 5-fold cross-validation, similar to the previous work [40]. The passages are treated as input sequences for the model. For each passage, word segmentation and POS tagging are processed by the custom models from this dataset.

4.2. Implementation Details

Before mapping each token included in the unigram, bigram, and trigram to the embedding vector, we limit the minimum frequency of occurring words that are not marked as an unknown token. There are 2 parameters set for the unigram C_{word} and the remaining C_{ngram}, respectively. We found that model accuracy is highly sensitive to these parameters. Therefore, we use a grid search technique to find the best value for both parameters for the model.

We apply two optimizers used in this work: AdaGrad [66] and Adam [67], whose learning rates are set to 0.02. To generalize the model, we also integrate L2 regularization with an alpha of 0.01 to the loss function for model updat-

ออกแบบ	ระบบ	บทคัดย่อ	งานวิจัย	นี้	_	มี	จุดประสงค์	จะ	พัฒนา	ซอฟท์แวร์	ต้นแบบ	_	สำหรับ	การ	จำลอง	อิง	กรรมวิธี
(ɔ̀:k bè:p)	(ra bòp)	(bòt kʰát yɔ̌)	(ŋa:n wí tɕaj)	(ní:)		(mi:)	(jùt pra soŋ)	(tɕa)	(pʰát tʰa na:)	(sɒftweɔ)	(tôn bè:p)		(sam ráp)	(ka:n)	(jam lɔ:ŋ)	(iŋ)	(kam má wí tʰi:)
	sb	**sb**															**sb**

Translated: System design. Abstract. This research work is intended to develop a prototype of software for a process-oriented simulation.

ใน	การ	ออกแบบ	โทรศัพท์	จะ	มี	ขั้นตอน	ดัง	โฟล์วชาร์ต	ข้างล่าง	นี้	ศึกษา	รายละเอียด	ของ	โทรศัพท์	ที่	มี	ใช้	อยู่
(na:j)	(ka:n)	(ɔ̀:k bè:p)	(tʰo: ra sáp)	(tɕa)	(mi:)	(kʰân tɔ:n)	(daŋ)	(floo tɛart)	(kʰâŋ lâ:ŋ)	(ní:)	(suk sa:)	(ra:j la iat)	(kʰɔ:ŋ)	(tʰo: ra sáp)	(tʰi)	(mi:)	(tɕʰá:j)	(yù:)
										sb								**sb**

Translated: In a telephone design, the procedure is shown in the following flowchart. The study on the detail of the existing telephone.

สภาพการณ์	ที่	จำลอง	จะ	เป็น	สภาพการณ์	การ	ดำเนินงาน	เทียม	โดยที่	มี	องค์ประกอบทำงาน	พื้นฐาน	คือ	_	กรรมวิธี
(sa pʰâ:p ka:n)	(tʰi:)	(tɕam lɔ:ŋ)	(tɕa)	(pen)	(sa pʰâ:p ka:n)	(ka:n)	(dam nɣ:n ŋa:n)	(tʰiam)	(do:j tʰi:)	(mi:)	(oŋ pra kɔ̀ŋ)	(tʰam ŋa:n)	(puú:n tʰa:n)	(kʰɯ)	(kam má wí tʰi:)
								sb							**sb**

Translated: An environment that is simulated will be the artificial process environment. Its basic element is a process.

Fig. 6. Labeled paragraphs in Orchid dataset. Here, *sb* represents a sentence boundary.

เมื่อ	กี้	สาย	ตัด	ไป	_	อยาก	ทราบ	ว่า	ได้	รับ	เรื่อง	ไว้	แล้ว	ยัง	ค่ะ
(muâ)	(kî:)	(sa:j)	(tàt)	(paj)		(jà:k)	(sâ:p)	(wâ:)	(dâj)	(ráb)	(ruâŋ)	(wá:j)	(lɛ́:w)	(jaŋ)	(kʰâ)
				sb	**sb**										**sb**

Translated: The connection is lost. I want to know if you get the request or not.

เปิดบัญชี	ครั้ง	แรก	_	ใช้	เงิน	เท่าไหร่	ค่ะ	_	ถ้า	เป็น	บัตร	_	debit	_	ธรรมดา
(pɤ̀:t ban tɕʰi:)(kʰtáŋ)		(rêːk)		(tɕʰáj)	(ŋɤn)	(tɕʰāw rày)	(kʰâ)		(tʰâ:)	(pen)	(bàt)				(tʰam ma da:)
		sb	**sb**												**sb**

Translated: Open an account for the first time. How much fee for the registration if I want a normal debit card?

ต้องการ	จะ	เปลี่ยน	เบอร์	ต้อง	ทำ	ยังไง	บ้าง	ครับ
(tɔ̂ŋ ka:n)	(tɕà)	(plìan)	(bɛ:)	(tɔ̂ŋ)	(tʰam)	(jaŋ ŋa:j)	(bâ:ŋ)	(kʰráp)
			sb					**sb**

Translated: I want to change the number. What should I do?

Fig. 7. Labeled paragraphs in UGWC dataset. Here, *sb* represents a sentence boundary.

ing. Moreover, to mitigate the overfitting problem, dropout [68] is applied to the local representation vectors, recurrent representation vectors, between all bidirectional LSTMs and enclosed by the self-attention mechanism in the high-level module. The dropout ratio of each part is shown in Section A.

During training, both the supervised and semi-supervised models are trained until the validation metrics stop improving; the metric is the sentence boundary F1 score.

CVT has three main parameters that impact model accuracy. The first is the drop rate of the masked language model, which determines the number of tokens that are dropped and used for learning auxiliary prediction modules as described in Section 3.2. The second is the number of unlabeled mini-batches B used for training between supervised mini-batches. Third, rather than using the same dropout rate for the local representation vectors, a new dropout rate is assigned.

For other hyperparameters, the details are given in Section A, such as the hidden size of each layer and dropout rate.

4.3. Evaluation

During the evaluation, each task is assessed using different metrics based on previous works. For Thai sentence segmentation, three metrics are used in the evaluation: sentence boundary F1 score, non-sentence boundary F1 score, and space correct [14]. In this work, we mainly focus on the performance of sentence boundary prediction and not non-sentence boundary prediction or space prediction. Therefore, we make comparisons with other models regarding only their sentence boundary F1 scores. In calculating the F1 score, the positive class is defined as the sentence boundary, and the negative class is defined as the non-sentence boundary. The equation for the sentence boundary F1 score metric is shown in (20), where $\#(A)$ is the number of A and tp, fp, and fn are true positive, false positive, and false negative, respectively.

$$tp_{sb} = \#(\text{Correctly predicted sentence boundaries})$$

$$fp_{sb} = \#(\text{Incorrectly predicted sentence boundaries})$$

$$fn_{sb} = \#(\text{Incorrectly predicted non-sentence boundaries})$$

$$precision_{sb} = \frac{tp_{sb}}{tp_{sb} + fp_{sb}}$$

$$recall_{sb} = \frac{tp_{sb}}{tp_{sb} + fn_{sb}}$$

$$F_{1,sb} = \frac{2 \times precision_{sb} \times recall_{sb}}{precision_{sb} + recall_{sb}} \quad (20)$$

5. Results and Discussions

We report and discuss the results of our two tasks in four subsections. The first and second subsections include the effect of local representation and distant representation, respectively. The impact of CVT and ELMo is explained in the third subsection. The last subsection presents a comparison of our model and all the baselines. Moreover, we also conduct paired t-tests to investigate the significance of the improvement from each contribution, as shown in Section C.

5.1. Effect of Local Representation

To find the effect of local representation, we compare a standard Bi-LSTM-CRF model using our full implementation on the model that includes n-gram embedding to extract local representations. In (2), the standard Bi-LSTM-CRF model is represented as Bi-LSTM-CRF (row (e)), while the models with local features are represented as $+local$ (row (f)).

The results in Table 2 show that using n-grams to obtain the local representation improves the F1 score of the model from 90.9% (row (e)) to 92.4% (row (f)) on the Orchid dataset and from 87.6% (row (e)) to 88.7% (row (f)) on the UGWC dataset. These results occur because there are many word groups that can be used to signal the beginning and end of a sentence in Thai. For instance, "แล้ว|ครับ" (lɛ́:w | k$^{\text{h}}$ráp) is a word group that is usually located at a sentence boundary in Thai. The model with local representation can detect a sentence boundary at "ครับ" (k$^{\text{h}}$ráp) that is followed by "แล้ว" (lɛ́:w), as shown in Fig. 8, while the model without local representation cannot detect the word as a sentence boundary. To elaborate the details of these word groups, the result of an interpretation process will be further analyzed in the next section.

5.1.1. Interpretation of Local Representation

In this section, our model is interpreted and analyzed via the method presented in Section 3.3. Count $(pos, gram)$, which refer how similarity between the model and human, is calculated over the training set of each corpus, as shown in Table 3.

In UGWC section of Table 3, the result indicates that the model mostly focuses at the end of the sentence more than the beginning of the sentence due to the lower number of focused features at the relative position $pos = +1$. In the intersection list from UGWC at the relative position $pos = -1, 0$, the features found near sentence boundaries are usually final particles, e.g., "นะ|คะ" (ná | k$^{\text{h}}$á), "นะ|ครับ" (ná | k$^{\text{h}}$ráp), "เลย|ครับ" (lə:y | k$^{\text{h}}$ráp), "แล้ว|ครับ" (lɛ́:w | k$^{\text{h}}$ráp), and others. These features are usually used at the ends of sentences to indicate the formality level. Meanwhile, the intersection list at the relative position $pos = +1$ are composed of greeting words and "Thank you" phrases that are always at the beginning of sentence, e.g., "สวัสดี (Hello)", "ขอบคุณ (Thank you)" and others.

In contrast, in Orchid, the model focuses more at the beginning of sentence. The features can be categorized into two groups. First, the content words, such as topics ("บทคัดย่อ (abstract)", "บทนำ (introduction)"), are focused due to their high frequency caused by the similar structure between documents. The second group is the function words, such as "เป็นดังนี้ (as follows)", "ข้างล่างนี้ (as shown below)" and "ได้ดังนี้ (as follows)", which are simply classified as the end of paragraphs or sentences.

As a result, the models of both datasets focus on different types of features due to the different writing styles. Despite these findings, our model is able to learn and utilize these n-grams features, which are also used by humans to decide where the sentence boundary is. Therefore, using local representation to capture n-grams features attains more improvement over other contributions.

5.2. Effect of Distant Representation

The effect of this contribution can be found by comparing the model that integrates the distant representation and the model that does not. The model with distant features integrated is represented as $+local + distant$ (row (g)) in both tables. In this case, the distant representation is composed of the self-attention modules in both the low- and high-level modules, as shown in Fig. 2.

From the combination of local and distant representations, the results in 2 show that the distant feature improves the accuracy of the model on all datasets compared to the model with no distant representation. The F1 scores of the sentence segmentation models improved slightly, from 92.4% and 88.7% (row (f)) to 92.5% and 88.8% (row (g)) on the Orchid and UGWC datasets, respectively.

This also illustrates that the model can be improved by adding the self-attention modules to Bi-LSTM layers. In conclusion, the results have shown that each of the proposed modules have a positive effect on the overall performance.

5.3. Effect of Cross-View Training (CVT) and ELMo

To identify the improvement from CVT, we compared the models that use different training processes: standard supervised training ($+ local + distant$), CVT ($+ local + distant + CVT$), and CVT + ELMo ($+ local + distant + CVT + ELMo$). The model trained with CVT improves the accuracy in terms of the F1 score, as shown in

Table 2. The result of Thai sentence segmentation for each model. For the Orchid dataset, we report the average of each metric on 10-fold cross-validation. Meanwhile, average metrics from 5-fold cross-validation are shown for the UGWC dataset.

Model	Orchid			UGWC		
	precision (%)	recall (%)	F1 (%)	precision (%)	recall (%)	F1 (%)
(a) POS-trigram [12]	74.4	79.8	77.0	-	-	-
(b) Winnow [13]	92.7	77.3	84.3	-	-	-
(c) ME [11]	86.2	83.5	84.8	-	-	-
(d) CRF (Thai baseline) [14]	94.7	89.3	91.9	87.4	82.7	85.0
ine (e) Bi-LSTM-CRF [20]	92.1	89.7	90.9	87.8	87.4	87.6
(f) + local	93.1	91.7	92.4	88.4	89.0	88.7
(g) + local + distant	**93.5**	**91.5**	**92.5**	88.8	88.8	88.8
(h) + local + distant + CVT	-	-	-	**88.9**	89.0	88.9
(i) + local + distant + CVT + ELMo	-	-	-	88.8	**91.0**	**89.9**

Note: The CVT model is not tested on the Orchid dataset because of the lack of unlabeled data.

	มี	นาน	แล้ว	ครับ	เริ่มต้น	ที่	1	ล้าน
	(mi:)	(na:n)	(lɛ́w)	(kʰráp)	(rə̂m tôn)	(tʰî)	(nuŋ̀)	(lá:n)
Bi-LSTM-CRF								sb
+ local				sb				sb

Translated: I got it for a long time. It's start from one million.

Fig. 8. An example of sentence boundary prediction by a normal Bi-LSTM-CRF and by the model with local representation ($+local$). Here, *sb* indicates that the word is predicted as the sentence boundary.

Table 3. The number of features in the intersection of two lists that is created from the interpretation score and labels.

Type of n-gram ($gram$)	Relative Position (pos)					
	Orchid			UGWC		
	-1	0	+1	-1	0	+1
Unigram	51	64	117	203	171	152
Bigram	6	22	54	162	165	3
Trigram	45	98	56	206	75	28

2 (row (g) vs row (h)). Additionally, adding ELMo provides more improvement on the F1 score (row (h) vs row (i)).

This experiment was conducted only on the UGWC dataset because no unlabeled data are available in the Orchid dataset, as mentioned in Section 4.1.1. The model trained with CVT improves the F1 score slightly, from 88.8% (row (g)) to 88.9% (row (h)) on the UGWC dataset. Meanwhile, adding ELMo, we also gain the improvement from 88.9%(row (h)) to 89.9%(row (i)). As a result, utilizing unlabeled data from both methods enhances the model performance significantly.

5.4. Comparison with Baseline Models

Our model is superior to all the baselines on both Thai sentence segmentation datasets, as shown in Table 2. On the Orchid dataset, the supervised model that includes both local and distant representation was adopted for comparison with the baseline model. Our model improves the F1 score achieved by CRF-ngram, which is the state-of-the-art model for Thai sentence segmentation in Orchid, from 91.9% (row (d)) to 92.5% (row (g)). Meanwhile, in the UGWC dataset, our CVT model with ELMo (row (h)) achieves an F1 score of 89.9%, which is higher than the F1 score of both the baselines (CRF-ngram and Bi-LSTM-CRF (rows d and e, respectively)). Thus, our model is now the state-of-the-art model for Thai sentence segmentation on both the Orchid and UGWC datasets.

5.5. Discussion

We have shown that incorporating local and global information with CVT can be used to improve the Thai sentence segmentation task. However, we would like to note that our proposed method assumes no idiosyncrasies specific to the Thai language, might be able to improve other languages or tasks as well. For example, one might consider the tasks of Elementary Discourse Unit (EDU) and clause segmentation which can help downstream tasks such as text summarization and machine translation by providing the minimal syntactic units.

Moreover, to overcome the scarcity of labeled data in Thai, all available sentence segmentation datasets, such as ORCHID, UGWC, and the recently released LST20 [69], should be integrated. However, the annotation criteria of the datasets are different making the task non-trivial. One possible venue for exploration is to use multi-criteria learning to utilize the shared information in all datasets. This method was successfully applied to Chinese word segmentation, which has a similar problem [70, 71].

6. Conclusions

In this paper, we propose a novel deep learning model for Thai sentence segmentation. This study makes three main contributions. The first contribution is to integrate a local representation based on n-gram embedding into our deep model. This approach helps to capture word groups near sentence boundaries, allowing the model to identify boundaries more accurately. Second, we integrate a distant representation obtained from self-attention modules to capture sentence contextual information. This approach allows the model to focus on the initial words of dependent clauses (i.e., "Before", "If", and "Although"). The last contribution is an adaptation of CVT, which allows the model to utilize unlabeled data to produce effective local and distant representations.

The experiment was conducted on two Thai datasets, Orchid and UGWC. Our model achieves F1 scores of 92.5% and 89.9% on the Orchid and UGWC datasets, constituting a relative error reduction of 7.4% and 18.5%, respectively. Based on our contributions, the local representation has the highest impact on the Thai corpus.From the interpretation process, the local representation revealed that it captures phrases that frequently occurred near the sentence boundary, which is usually the approach used by humans to recognize the boundary.

References

[1] S. Kübler, R. McDonald, and J. Nivre, "Dependency parsing," *Synthesis lectures on human language technologies*, vol. 1, no. 1, pp. 1–127, 2009.

[2] D. Gillick, "Sentence boundary detection and the problem with the us," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 2009, pp. 241–244.

[3] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[5] R. Aharoni, M. Johnson, and O. Firat, "Massively multilingual neural machine translation," *arXiv preprint arXiv:1903.00089*, 2019.

[6] R. Mihalcea, "Graph-based ranking algorithms for sentence extraction, applied to text summarization," in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, 2004.

[7] M. Afsharizadeh, H. Ebrahimpour-Komleh, and A. Bagheri, "Query-oriented text summarization using sentence extraction technique," in *2018 4th International Conference on Web Research (ICWR)*. IEEE, 2018, pp. 128–132.

[8] A. Joshi, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "Summcoder: An unsupervised framework for extractive text summarization based on deep auto-encoders," *Expert Systems with Applications*, vol. 129, pp. 200–215, 2019.

[9] Z. Li, X. Peng, M. Zhang, R. Wang, and L. Si, "Semi-supervised domain adaptation for dependency parsing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2386–2395.

[10] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," *arXiv preprint arXiv:1901.10457*, 2019.

[11] G. Slayden, M.-Y. Hwang, and L. Schwartz, "Thai sentence-breaking for large-scale smt," in *Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing*, 2010, pp. 8–16.

[12] P. Mittrapiyanuruk and V. Sornlertlamvanich, "The automatic thai sentence extraction," in *Proceedings of the fourth symposium on Natural Language Processing*, 2000, pp. 23–28.

[13] P. Charoenpornsawat and V. Sornlertlamvanich, "Automatic sentence break disambiguation for thai," in *International Conference on Computer Processing of Oriental Languages (ICCPOL)*, 2001, pp. 231–235.

[14] N. Zhou, A. Aw, N. Lertcheva, and X. Wang, "A word labeling approach to Thai sentence boundary detection and POS tagging," in *Proceedings of*

COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 319–327. [Online]. Available: https://www.aclweb.org/anthology/C16-1031

[15] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[16] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009.

[17] V. Sornlertlamvanich, T. Charoenporn, and H. Isahara, "Orchid: Thai part-of-speech tagged corpus," *National Electronics and Computer Technology Center Technical Report*, pp. 5–19, 1997.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[20] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[21] J. Yi, J. Tao, Y. Bai, Z. Tian, and C. Fan, "Adversarial transfer learning for punctuation restoration," *arXiv preprint arXiv:2004.00248*, 2020.

[22] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 724–728.

[23] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1638–1649. [Online]. Available: https://www.aclweb.org/anthology/C18-1139

[24] J. Straková, M. Straka, and J. Hajic, "Neural architectures for nested NER through linearization," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5326–5331. [Online]. Available: https://www.aclweb.org/anthology/P19-1527

[25] Z. Wang, J. Shang, L. Liu, L. Lu, J. Liu, and J. Han, "CrossWeigh: Training named entity tagger from imperfect annotations," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5154–5163. [Online]. Available: https://www.aclweb.org/anthology/D19-1519

[26] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," *arXiv preprint arXiv:1809.08037*, 2018.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[28] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[29] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in neural information processing systems*, 2015, pp. 3546–3554.

[30] T. Miyato, S.-i. Maeda, S. Ishii, and M. Koyama, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[31] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.

[32] S. Ruder and B. Plank, "Strong baselines for neural semi-supervised learning under domain shift," *arXiv preprint arXiv:1804.09530*, 2018.

[33] K. Clark, M.-T. Luong, C. D. Manning, and Q. Le, "Semi-supervised sequence modeling with cross-view training," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1914–1925.

[34] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[35] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[36] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[37] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[38] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[39] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[40] A. Lertpiya, T. Chaiwachirasak, N. Maharattanamalai, T. Lapjaturapit, T. Chalothorn, N. Tirasaroj, and E. Chuangsuwanich, "A preliminary study on fundamental thai nlp tasks for user-generated web content," in *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. IEEE, 2019, pp. 1–8.

[41] S. Wathabunditkul, "Spacing in the thai language," 2003.

[42] H. WANG, J. WANG, Q. SHEN, Y. XIAN, and Y. ZHANG, "Maximum entropy thai sentence segmentation combined with thai grammar rules correction."

[43] N. Tangsirirat, A. Suchato, P. Punyabukkana, and C. Wutiwiwatchai, "Contextual behaviour features and grammar rules for thai sentence-breaking," in *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. IEEE, 2013, pp. 1–4.

[44] H. Wang, Z. Zhang, Q. Shen, Y. Xian, Y. Zhang, and C. Mao, "Thai language sentence segmentation based on n-gram context model," in *2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*. IEEE, 2019, pp. 813–817.

[45] R. Nararatwong, N. Kertkeidkachorn, N. Cooharojananone, and H. Okada, "Improving thai word and sentence segmentation using linguistic knowledge," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 12, pp. 3218–3225, 2018.

[46] Y. Gotoh and S. Renals, "Sentence boundary detection in broadcast speech transcripts," in *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

[47] S. Peitz, M. Freitag, A. Mauser, and H. Ney, "Modeling punctuation prediction as machine translation," in *International Workshop on Spoken Language Translation (IWSLT) 2011*, 2011.

[48] E. Cho, J. Niehues, K. Kilgour, and A. Waibel, "Punctuation insertion for real-time spoken language translation," in *Proceedings of the Eleventh International Workshop on Spoken Language Translation*, 2015.

[49] F. Wang, W. Chen, Z. Yang, and B. Xu, "Self-attention based network for punctuation restoration," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018.

[50] A. Gravano, M. Jansche, and M. Bacchiani, "Restoring punctuation and capitalization in transcribed speech," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 4741–4744.

[51] W. Lu and H. T. Ng, "Better punctuation prediction with dynamic conditional random fields," in *Proceedings of the 2010 conference on empirical methods in natural language processing*, 2010, pp. 177–186.

[52] N. Ueffing, M. Bisani, and P. Vozila, "Improved models for automatic punctuation prediction for spoken and written text." in *Interspeech*, 2013, pp. 3097–3101.

[53] X. Che, C. Wang, H. Yang, and C. Meinel, "Punctuation prediction for unsegmented transcript based on word vector," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016, pp. 654–658.

[54] O. Tilk and T. Alumäe, "Lstm for punctuation restoration in speech transcripts," in *Sixteenth annual conference of the international speech communication association*, 2015.

[55] ——, "Bidirectional recurrent neural network with attention mechanism for punctuation restoration." 2016.

[56] S. Kim, "Deep recurrent neural networks with layer-wise multi-head attentions for punctuation restoration," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7280–7284.

[57] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[58] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, É. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8440–8451.

[59] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[60] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[61] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners."

[62] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism Bulletin*, vol. 30, no. 4, pp. 415–433, 1953.

[63] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[64] E. Wallace, J. Tuyls, J. Wang, S. Subramanian, M. Gardner, and S. Singh, "Allennlp interpret: A framework for explaining predictions of nlp models," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, 2019, pp. 7–12.

[65] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 31–36.

[66] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

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

[68] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[69] P. Boonkwan, V. Luantangsrisuk, S. Phaholphinyo, K. Kriengket, D. Leenoi, C. Phrombut, M. Boriboon, K. Kosawat, and T. Supnithi, "The annotation guideline of lst20 corpus," *arXiv preprint arXiv:2008.05055*, 2020.

[70] W. Huang, X. Cheng, K. Chen, T. Wang, and W. Chu, "Towards fast and accurate neural chinese word segmentation with multi-criteria learning," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 2062–2072.

[71] X. Chen, Z. Shi, X. Qiu, and X.-J. Huang, "Adversarial multi-criteria learning for chinese word segmentation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1193–1203.

Appendix A Hyperparameters

The hyperparameter values were determined through a grid search to find their optimal values on the different datasets. All the hyperparameters for each dataset are shown in Table 4. The optimal values from the grid search depend on the task. For Thai sentence segmentation, the hyperparameters are tuned to obtain the highest sentence boundary F1 score.

Appendix B Comparison of CNN and n-gram models for local representation

Jacovi A. et al. [26] proposed that a CNN can be used as an n-gram detector to capture local text features. Therefore, we also performed an experiment to compare a CNN and n-gram embedded as local structures. The results in Table 5 show that the model using the embedded n-gram yields greater improvement than the one using an embedded CNN on the Orchid and UGWC datasets.

Appendix C Statistical Tests for Thai sentence segmentation

To prove the significance of the model improvements, we compared the cross-validation results using paired t-tests to obtain the p-values, which are shown in Table 6 for the Orchid dataset and Table 7 for the UGWC dataset.

Table 4. Model hyperparameters for each dataset.

Model	Orchid	UGWC
C_{word}	2	2
C_{ngram}	2	2
Optimizer	AdaGrad	AdaGrad
Learning rate	0.02	0.02
Batch size	16	16
Early stopping patience	5	5
Unigram embedding size (Text)	64	64
Unigram embedding size (POS and Type)	32	32
Bigram & Trigram embedding size (Text)	16	16
Bigram & Trigram embedding size (POS and Type)	8	8
LSTM hidden size	25	25
Number of LSTM layers in high-level module (K)	2	2
Self-attention output size	50	50
Number of Low-level self-attention layers	1	1
Number of High-level self-attention layers	1	1
Low-level self-attention projection size	64	64
High-level self-attention projection size	25	25
Local embedding dropout	0.30	0.30
Dropout between layers	0.15	0.15
Dropped rate of masked language model	-	0.30
Number of unlabeled mini-batch B	-	1
Dropout of the unlabeled input	-	0.50
Hidden size of LSTM in ELMo	-	4096
Number of layers of LSTM in ELMo	-	2

Table 5. Comparison between a CNN and n-gram embedding for local representation extraction.

Model	ORCHID	UGWC
Bi-LSTM-CRF	90.9%	87.6%
Bi-LSTM-CRF + n-gram	92.4%	88.7%
Bi-LSTM-CRF + CNN	91.2%	87.9%

Table 6. The improvement of each contribution on the Orchid dataset results shown as p-values from paired t-tests.

Model	+ local	+ local/distant
CRF	+0.47% ± 0.42% (0.009)	+0.54% ± 0.36% (0.002)
Bi-LSTM-CRF	+1.53% ± 0.39% (<0.001)	+1.60% ± 0.39% (<0.001)
+ local	-	+0.07% ± 0.22% (0.370)

Note: The number in the table reflects the percentage of improvement from the rows compared with the columns. The number shows the average and its standard deviation of the improvement. Moreover, the number in parentheses is the p-value computed from a paired t-test.

Chanatip Saetia received a B.Eng. degree in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 2018, where he is currently pursuing a M.Eng. degree in computer engineering. His research interests include natural language processing, information extraction, and text categorization.

Table 7. The improvement of each contribution on the UGWC dataset results shown as p-values from paired t-tests

Model	+local	+local/distant	+local/distant/CVT	+local/distant/CVT/ELMo
CRF	+3.66% ± 0.16%	+3.77% ± 0.20%	+3.92% ± 0.20%	+4.87% ± 0.25%
	(<0.001)	(<0.001)	(<0.001)	(<0.001)
Bi-LSTM-CRF	+1.09% ± 0.14%	+1.20% ± 0.20%	+1.34% ± 0.14%	+2.29% ± 0.16%
	(<0.001)	(<0.001)	(<0.001)	(<0.001)
+local	-	+0.11% ± 0.14%	+0.26% ± 0.06%	+1.21% ± 0.10%
	-	(0.182)	(0.001)	(<0.001)
+local/distant	-	-	+0.15% ± 0.12%	+1.10% ± 0.14%
	-	-	(0.065)	(<0.001)
+local/distant/CVT	-	-	-	+0.95% ± 0.05%
	-	-	-	(<0.001)

Note: The number in the table reflects the percentage of improvement from the rows compared with the columns. The number shows the average and its standard deviation of the improvement. Moreover, the number in parentheses is the p-value computed from a paired t-test.

Ekapol Chuangsuwanich received both B.S. and M.S. degrees in electrical and computer engineering from Carnegie Mellon University, in 2008 and 2009, respectively, and a Ph.D. degree from MIT, in 2016. He then joined the Spoken Language Systems Group, MIT Computer Science and Artificial Intelligence Laboratory. He is currently a Faculty Member in the Department of Computer Engineering, Chulalongkorn University. His research interests include speech processing, assistive technology, and health applications.

Tawunrat Chalothorn received B.S., M.S., and Ph.D. degrees from the University of Northumbria, Newcastle, U.K., in 2010, 2011, and 2016, respectively. She then joined the Natural Language Processing Team, Kasikorn Labs (KLabs) Company Ltd., Kasikorn Business Technology Group (KBTG), which mostly conducts based on Thai language. Her research interests include chatbots, social listening, and text analytics.

Peerapon Vateekul received his Ph.D. degree from the Department of Electrical and Computer Engineering, University of Miami (UM), Coral Gables, FL, U.S.A. in 2012. Currently, he is an assistant professor at the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand. His research is in the domain of machine learning, data mining, deep learning, text mining, and big data analytics. In particular, his works include variants of classification (hierarchical multi-label classification), natural language processing, data quality management, and applied deep learning techniques in various domains, such as medical images and videos, satellite images, meteorological data, and text.