# Syllable-based Neural Thai Word Segmentation

**Pattarawat Chormai**[*,⋆,†], **Ponrawee Prasertsom**[‡], **Jin Cheevaprawatdomrong**[‡]
and **Attapol T. Rutherford**[‡]
[‡]Department of Linguistics, Chulalongkorn University, Bangkok, Thailand
[†]Computer Science and Electrical Engineering, Technische Universität, Berlin, Germany
[⋆]Max Planck School of Cognition, MPI for Human Cognitive and Brain Sciences,
Leipzig, Germany
p.chormai@tu-berlin.de, {ponrawee.pra, jin.ch}@gmail.com,
attapol.t@chula.ac.th

## Abstract

Word segmentation is a challenging pre-processing step for Thai Natural Language Processing due to the lack of explicit word boundaries. The previous systems rely on powerful neural network architecture alone and ignore linguistic substructures of Thai words. We utilize the linguistic observation that Thai strings can be segmented into syllables, which should narrow down the search space for the word boundaries and provide helpful features. Here, we propose a neural Thai Word Segmenter that uses syllable embeddings to capture linguistic constraints and uses dilated CNN filters to capture the environment of each character. Within this goal, we develop the first ML-based Thai orthographical syllable segmenter, which yields syllable embeddings to be used as features by the word segmenter. Our word segmentation system outperforms the previous state-of-the-art system in both speed and accuracy on both in-domain and out-domain datasets.

## 1 Introduction

Word segmentation presents a fundamental challenge for Thai language processing. Many of the downstream natural language processing tasks require that texts be broken into a sequence of words before applying any models. For example, bag-of-word models or RNN models usually require that the text is represented as a sequence of words. Notable examples of writing systems without clear word boundaries are Japanese, Chinese, and Khmer, which often require an automatic word segmentation. However, the Thai script differs from Chinese and Japanese in that a Thai character represents just a consonant or a vowel, but a Chinese character represents a whole syllable. The Thai script uses 44 consonant symbols, 15 vowel symbols, and 4 tonal symbols. A word is composed of one or more syllables, and each syllable is formed by a set of intricate orthographical rules for valid sequences of Thai alphabet symbols. Hence, it is not straightforward to extend the techniques from Chinese or Japanese to Thai word segmentation problem.

Thai Word segmentation is complicated by linguistic ambiguity and out-of-vocabulary cases. A word can be formed by juxtaposing two "words" e.g. เห็นชอบ (/hěn tɕʰɔ̂ːp/, approve) = เห็น (/hěn/, see) + ชอบ (/tɕʰɔ̂ːp/, like). This kind of word formation can be detected with a simple dictionary lookup, but harder cases that require context abound in the language. For example, กอดอกไม้ can be segmented into กอ|ดอก|ไม้ (/kɔː dɔ̀ːk máj/, flower bush) or กอด|อก|ไม้ (/kɔ̀ːt ʔók máj/, hugging a wooden chest), but the latter is rather nonsensical and statistically unlikely. The local context is needed to select the right segmentation; therefore, dictionaries only provide partial solutions to this problem. Further, a constant stream of new words and loanwords complicates the task of word segmentation because they never appear in the dictionaries.

In recent years, a few open-sourced Thai word segmenters have been introduced and used widely in the industry. Notable examples of open-sourced Thai word segmenters include PyThaiNLP (Phatthiyaphaibun et al., 2016), Sertis (Sertis Co., Ltd., 2017), and DeepCut (Kittinaradorn et al., 2019), which claim

[*]Part of the work done while interning at Dr. Attapol T. Rutherford's lab.

good performance according to their own respective benchmarks. The accuracies of these word segmentation systems are not benchmarked on the same datasets for a rigorous comparison within and across domains. In this paper, we reimplement some of these baselines for comparison. More importantly, these techniques formulate the word segmentation problem as a character-sequence tagging problem and ignore the fact that characters form a syllable as an intermediate step to word formation.

Our systems take advantage of the fact that every word can be parsed into orthographical syllables. A sequence of Thai characters can be thought of as a sequence of orthographical syllables, each of which consists of at least two consonant characters or a pair of a consonant and a vowel. We hypothesize that syllable segmentation should simplify the downstream task of word segmentation. We explore the use of syllables in two ways. We use syllables as features (embeddings) for the word segmentation model, and we also try posing the word segmentation as a syllable-tagging problem, as opposed to character-tagging problem. We present a neural word segmentation model that utilizes character and syllable embeddings as the representation and achieve state-of-the-art result. We conclude that underlying linguistic structures (such as syllables) can serve as a suitable structure for neural architecture, which yields better performance as a result.

Our contributions can be summarized as follows:

- We propose a neural Thai word segmentation model that outperforms the previous state-of-the-art system in both in-domain and out-of-domain evaluations at $F_1$ scores 0.95 and 0.86 respectively . Our model also operates $8.9\times$ and $3.5\times$ faster than the previous system on CPU and GPU instances respectively. Our code is publicly available[1].
- We develop the first ML-based Thai orthographical syllable segmenter, which achieves 0.96 syllable-level $F_1$ and 0.99 character-level $F_1$ score and allows very little error to propagate to the downstream word segmentation task.
- We show that syllable segmentation helps word segmentation as a feature and as a structure for word segmentation. Our syllable segmenter is now part of PyThaiNLP package, which is widely used by the Thai NLP community.

## 2 Problem Statement: Thai Word and Syllable Segmentation

Given a string of Thai text, a Thai word segmenter identifies the word boundaries within the string. This problem is formulated as a classification problem or sequence-tagging problem where we try to identify a character that starts a new word in the string. A Thai word is defined as the smallest lexical unit that still conveys the meaning (Aroonmanakun, 2007). Most ambiguous and debatable cases revolve around the degree of compositionality of nouns and verbs. The word การบ้าน (/kaːn.bâːn /, homework) is one word, although การ (/kaːn/, nominalization morpheme) and บ้าน (/bâːn/, house) are also words in other context.

The meaning of การบ้าน is not compositional from the two potential morphemes and therefore should be treated as one word. As a more uncertain example, it is arguable that the meaning of กอดอก (/kɔ̀ːt ʔòk/, to cross arms) is composed of กอด (/kɔ̀ːt/, to hug) and อก (/ʔòk/, chest) and therefore should not be treated as one word. Our dataset follows the guidelines that favor the segmentation takes the degree of compositionality into account and not grammatical function changes such as nominalization or verbalization.

Syllable segmentation task is defined similarly, but we try to find syllable boundaries instead of word boundaries. It is noteworthy that word boundaries are always subset of syllable boundaries because each syllable belongs to exactly one word. We use this fact as a basis for our model architecture. An orthographical syllable is defined as a substring in a word that can be pronounced as one or one and a half phonological syllable. For example, the word จินตนา can be segmented into two orthographical syllables จิน (/tɕin/), which is also phonologically one syllable, and ตนา (/ta.nāː/), which is phonologically two syllables. The mismatch between orthographical and phonological syllables is very common, and we believe orthographical syllable is more tractable computationally because each syllable is guaranteed to be at least two characters long and each syllable can also be a word itself in some cases.

[1] github.com/heytitle/Syllable-based-Neural-Thai-Word-Segmentation

## 3 Related Works

Thai word segmentation share some similarities with Chinese and Japanese word segmentation, but the linguistic differences require different types of models, but some insights from their techniques inspire our approaches. Both Chinese and Japanese word segmentation problems are formulated as character-sequence tagging problem. BiLSTM has been found to be very effective for Chinese (Ma et al., 2018) and Japanese (Kitagawa and Komachi, 2018). Interestingly, length-sensitive BIES tagset proves to be a better tagset that the standard BI tagset (Nakagawa, 2004). The BIES tagset cannot be readily applied to Thai word segmentation since Thai words are at least two characters long. In this study, we automatically group characters into syllables first, so that such length-sensitive tagset can be applied.

Previous approaches to Thai word segmentation can be categorized into two categories: dictionary-based and learning-based. (Poowarawan, 1986) proposes the first dictionary-based method using a greedy algorithm to decide when a word should be formed. Dictionary-based methods inevitably suffer from unseen words, and hence are harder to generalize to other domains. (Sornlertlamvanich, 1993) uses Maximal Matching, to handle such unseen word cases.

Past machine-learning word segmenters utilize subword features. (Theeramunkong et al., 2000) propose Thai Character Clusters (TCCs). TCCs differ from the notion of syllables used in our work in that TCCs are based loosely on the orthographical syllables and designed with information retrieval applications in mind. Using TCCs, (Theeramunkong and Usanavasin, 2001) develop a decision tree classifier to determine whether a word should be formed from TCCs, based on a predefined metric. (Aroonmanakun, 2002) presents a two-stage word segmentation that incorporates handcrafted syllable features with dynamic programming to form the most reasonable segmentation, while (Bheganan et al., 2009) use a hidden Markov model to form words that are then verified with a dictionary.

Modern approaches to this problem are used by practitioners, but it is unclear which approach is the most accurate or fastest. (Kittinaradorn et al., 2019) present DeepCut, a 1-dimension CNN for Thai word segmentation. The alleged state-of-the-art system (Sertis Co., Ltd., 2017) and (Lapjaturapit et al., 2018) use a BiRNN to segment words with multiple possible segmentation candidates. These results are not systematically benchmarked or published, but we use these architectures as our baselines.

## 4 Our Approach

We propose two Thai word segmentation systems based on BiLSTM and CNN architectures. The BiLSTM-based system follows the standard formulation of sequence-tagging BiLSTM model.

For the CNN-based system, unlike DeepCut that uses multiple convolutions on features, we employ Iterated Dilated Convolutions proposed by Strubell et al. (2017). Iterated Dilated Convolutions is a hierarchical dilated convolutional layers that the dilation number increases doubly at each layer. It allows the model to consume sufficient context with fewer parameters, and Strubell et al. (2017) demonstrate its advantages in prediction performance on entity recognition tasks and computational efficiency. We use a stack of three levels with width three and dilation numbers 1, 2, and 4; In total, it covers the context of seven characters on each side.

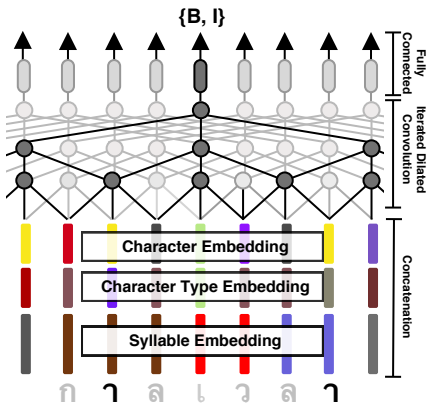


Figure 1: CNN-based word segmenter with character and syllable features using 3-level Iterated Dilated Convolutions (Strubell et al., 2017). Colors represent different embeddings. Word in figure is กาล เวลา (/ka:n we: la:/, time).

After the convolution layers or the BiLSTM layers, we have two fully-connected layers, combined extract features to the probability of tags. Figure 1 depicts the convolution architecture in details.

Our systems use syllables as additional features and structures to the model. Both systems use character embeddings and syllable embeddings as features. For a given character, we first find the syllable that contains the character and then look up the syllable embedding. Syllable embeddings should provide higher-level information than a character type or a character embedding can afford. As an additional experiment, we reformulate the task as a syllable-sequence tagging problem instead of a character-sequence tagging one. In this formulation, we use syllable embeddings only for both BiLSTM and CNN-based models.

We use new tagsets to take advantage of the fact that words with different lengths are distributed differently. The standard tagset uses binary tags: B for the beginning of a new word and I for the inside of a word. We explore a length-sensitive tagset: BI-short, BI-mid, and BI-long for words that have 1-2 syllables, 3-4 syllables, and 5+ syllables, which we call Scheme A. We also experiment with another length-sensitive tagset: BI-k tag where $k = \{1, 2, 3, 4+\}$ is the length of the word in terms of number of syllables, which we call Scheme B.

All of the proposed models require an automatic syllable segmenter as a preprocessing system for the word segmenters. Thai syllables are mostly bound by rules, but ambiguity and typos are natural occurrences in real data. To the best of our knowledge, there is no previous work in Thai syllable segmentation, and syllables have never been used as features for ML-based word segmentation systems. We propose a Conditional Random Fields (CRF) syllable segmenter primarily to be used as a preprocessing step for the word segmenter. We use template-based n-gram features to capture the local context of the sequence.

## 5 Experiments

### 5.1 Evaluation Metrics

Evaluating the quality of word segmenters is typically done on a character-level (CL) basis. Standard measured metrics are precision, recall, and $F_1$ of starting-word characters. However, intuitively, when a token is tokenized wrongly, it would consequentially affect the tokenization of following tokens. Thus, measuring only the character-level metrics would overestimate the tokenization performance of syllable or word tokenizers. Therefore, in this work, we also measure the chunk-level metrics—the syllable level (SL) or world level (WL). SL and WL $F_1$ are calculated based on the number of correct syllables and words respectively rather than characters. Figure 2 illustrates the character-level and chunk-level measures.

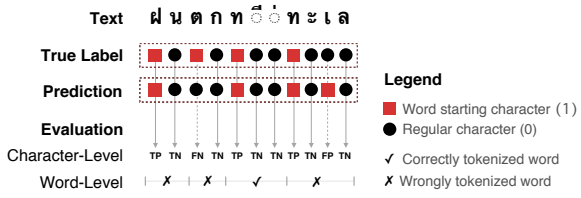


Figure 2: Segmentation evaluation metrics. Correct segmentation is ฝน|ตก|ที่|ทะเล (/fŏn tók tʰɯ̄ː tʰáʔ leː/, rain at sea).

### 5.2 Experiment 1: Syllable Segmenter

We hypothesize that syllable segmentation is simple enough that we can treat it as a preprocessing step for word segmentation. We use pycrfsuite implementation of CRF (Peng and Korobov, 2018; Okazaki, 2007) and train the model on Thai National Corpus (TNC) (Aroonmanakun et al., 2009). The TNC contains a subcorpus of 2.56M annotated syllables (around 8M characters). The dataset is split three-way for training, development and testing using the 70:20:10 scheme. The training, development, and test sets contain 1.8M syllables, 0.5M syllables, and 0.25M syllables respectively.

We hypothesize that CRF is suitable for syllable segmentation because of its inclusion of sequential information. We test this hypothesis by comparing it against a maximum entropy model (MaxEnt), trained using the scikit-learn implementation (Pedregosa et al., 2011). For both algorithms, we experiment with the following features, with $N$ and window size $W$ of 1 to 4: i) individual characters within $W$ places around on both sides of a potential boundary (Chr); ii) two N-grams on both sides (ChrSpan); iii) N-gram features that include all N-grams within $W$ places on both sides.

### 5.3 Experiment 2: Word Segmenter

Our training data is BEST-2010 (NECTEC, 2010). Annotated with word boundaries and name entities, the corpus contains 415 Thai documents from four categories: news, articles, encyclopedias, and novels, accounting for 134,107 samples (split by line), around 5.11M words, and 18.74M characters. Balancing the distribution of categories, we take 10% of the training set as a validation split. We use the official provided test set (officially named as TEST_100K) for evaluation, which has 2,252 samples, about 128K words, and 496K characters.

Apart from in-domain evaluation on BEST-2010 test set, we also perform cross-domain word segmentation evaluations on another two datasets: i) Thai National Historical Corpus (TNHC) (Sawatphol and Rutherford, 2019) contains 20,791 samples, around 599K words, and 2.14M characters of Thai classical literature documents with word boundaries annotated by humans, and ii) Wisesight corpus (Wisesight (Thailand) Co., Ltd., 2019) contains 26,700 social media messages. Because the Wisesight dataset does not have word boundary annotation, we randomly take 1,000 samples (with 7 spam messages removed) from the test split and manually segment them using the same word segmentation standard proposed by Aroonmanakun (2007). We call this Wisesight-1000: it has around 22K words, and 75K characters; our annotation is available at Wisesight's repository. Appendix B summarizes the statistics of the datasets.

Our word segmenters are from two models families, namely BiLSTMs and CNNs with Iterated Dilated Convolutions, or (ID-CNNs) for short. These word segmenters come with variants of:

- **CRF**: CRF can be employed in the last step of prediction.
- **Features sets**: we experiment with i) character and character type embeddings and ii) syllable embeddings for character-sequence models. See Appendix C.4 for the details of our character type and syllable dictionaries;
- **Tagset**: discussed in Section 4, we also experiment with three tagsets: BI, SchemeA, and SchemeB (BI-$k$ tagset);

We use the convention of `(Model-Family)[-CRF](Features)-(Tagset)` to refer to the specific configuration of each model.

We compare our word segmenters with two strong baselines: i) PyThaiNLP (Phatthiyaphaibun et al., 2016) with its maximal matching engine (Sornlertlamvanich et al., 1997); ii) DeepCut (Kittinaradorn et al., 2019), a character-level CNN (Zhang et al., 2015) with a stack of 13 convolutional filters, followed by pooling and fully-connected layers, comprising around 500K trainable parameters. iii) BiLSTM(CH)-BI and ID-CNN(CH)-BI, which are similar to the existing Thai word segmenters: (Sertis Co., Ltd., 2017) and (Kittinaradorn et al., 2019) respectively.

We use PyTorch (Paszke et al., 2019) for implementing word segmenters and train them using Adam (Kingma and Ba, 2015). For each model family, we perform 20 hyperparameter search trials using the random search strategy (Bergstra and Bengio, 2012). We use batch size of 32 and 128 for character-level and syllable-level models respectively. Detailed information about our training settings, hyperparameter search space, and the best configuration of each model configuration from the search can be found in Appendix D.

## 6 Results and Discussion

### 6.1 Syllable Segmentation Performance

Both MaxEnt and CRF achieve near-perfect performance (Table 1). This suggests that the output from syllable segmentation is suitable for downstream as it allows very little error to propagate. Our top performing models are CRF-based with character and trigram features ($N = 3$, $W = \{3, 4\}$) for both feature types). We observe that syllable boundaries from the model with $W = 4$ better align with word boundaries in BEST-2010 validation set; hence, we use this syllable model in our word segmentation experiment.

### 6.2 Word Segmentation Performance

| Model | Features | Syllable-seg | |
|---|---|---|---|
| | | Character-level $F_1$ | Syllable-level $F_1$ |
| CRF | Chr (W=3), Trigram (W=3) | 99.46% | 98.58% |
| CRF | Chr (W=3), ChrSpan (W=3) | 99.29% | 98.10% |
| CRF$^\star$ | Chr (W=4), Trigram (W=4) | 99.44% | 98.54% |
| MaxEnt | Chr (W=4) | 99.18% | 97.84% |
| MaxEnt | Chr (W=4), Trigram (W=4) | 99.29% | 98.02% |

Table 1: Our methods achieve near-perfect syllable-level $F_1$ scores on Thai National Corpus (TNC) test set. ($\star$): we use the model to train word segmenters.

| Method | Sequence unit | Features | | Tagset | BEST-2010 Val. $\text{WL}_{F_1}$ | |
|---|---|---|---|---|---|---|
| | | Character$^\ddagger$ | Syllable | | Best | AVG $\pm$ STD |
| BiLSTM | Character | ✓ | ✗ | BI | 97.01% | 96.55 $\pm$ 0.44 |
| | Character | ✓ | ✓ | BI | **97.34%** | 97.06 $\pm$ 0.28 |
| | Syllable | ✗ | ✓ | BI | 97.09% | 96.49 $\pm$ 0.45 |
| | Syllable | ✗ | ✓ | SchemeA | 97.08% | 96.27 $\pm$ 0.56 |
| | Syllable | ✗ | ✓ | SchemeB | 96.94% | 96.47 $\pm$ 0.42 |
| BiLSTM-CRF | Syllable | ✗ | ✓ | BI | **97.20%** | 96.73 $\pm$ 0.29 |
| | Syllable | ✗ | ✓ | SchemeA | 96.90% | 96.57 $\pm$ 0.40 |
| | Syllable | ✗ | ✓ | SchemeB | 96.97% | 96.41 $\pm$ 0.35 |
| ID-CNN | Character | ✓ | ✗ | BI | 96.53% | 95.75 $\pm$ 0.83 |
| | Character | ✓ | ✓ | BI | **97.11%** | 96.88 $\pm$ 0.16 |
| | Syllable | ✗ | ✓ | BI | 97.06% | 96.47 $\pm$ 0.41 |
| | Syllable | ✗ | ✓ | SchemeA | 97.02% | 96.51 $\pm$ 0.33 |
| | Syllable | ✗ | ✓ | SchemeB | 96.86% | 96.43 $\pm$ 0.39 |
| ID-CNN-CRF | Syllable | ✗ | ✓ | BI | 97.22% | 96.63 $\pm$ 0.39 |
| | Syllable | ✗ | ✓ | SchemeA | **97.26%** | 96.70 $\pm$ 0.38 |
| | Syllable | ✗ | ✓ | SchemeB | 97.09% | 96.76 $\pm$ 0.33 |

Table 3: Word segmentation quality ($\text{WL}_{F_1}$) on BEST-2010 validation set from different methods. Best $\text{WL}_{F_1}$'s of each model family and sequence level are in bold. Best, mean and standard deviation of each setting's $\text{WL}_{F_1}$ are statistics from 20 search trials. See Appendix D for the search space and each setting's best configuration.

BiLSTM(CH+SY)-BI and ID-CNN-CRF(SY)-SchemeA achieve the state-of-the-art performance on the BEST-2010 test set benchmark and out-of-domain datasets (Table 2). Our results confirm the hypothesis that syllables help improve word segmentation performance. On the validation set, syllable features improve BiLSTM-CH $\text{WL}_{F_1}$ from 97.01% to 97.34% and improve ID-CNN from 96.53% to 97.11% (Table 3). We observe similar improvement on the test set (Table 2).

The effects of syllable embeddings are more pronounced when we only consider out-of-vocabulary (OOV) cases (Table 4), where the models need to generalize rather than simply memorize words found in the training set. Syllable embeddings improve OOV recall from 59.58% to 67.42% for BiLSTM and from 51.92% to 64.09% for ID-CNN. This further confirms that syllable embeddings help improve

| Method | Dataset ($\text{WL}_{F_1}$) | | |
|---|---|---|---|
| | In-Domain | Out-Domain | |
| | BEST-2010 | Wisesight | TNHC |
| Previous work | | | |
| Dictionary-based | 71.18% | 78.97% | 72.70% |
| DeepCut | 94.46% | 84.45% | 78.17% |
| Ours | | | |
| BiLSTM | | | |
| (CH)-BI | 95.05% | 85.85% | 79.31% |
| (CH+SY)-BI | **95.59%** | **86.15%** | 78.70% |
| -CRF(SY)-BI | 95.51% | 86.10% | **79.89%** |
| ID-CNN | | | |
| (CH)-BI | 94.31% | 85.80% | 79.22% |
| (CH+SY)-BI | 95.45% | **86.43%** | **79.87%** |
| -CRF(SY)-SchA$^\star$ | **95.60%** | 86.15% | 79.64% |

Table 2: Word segmentation quality ($\text{WL}_{F_1}$) on BEST-2010 test set, Wisesight-1000 and TNHC datasets from different methods. Syllables help improve out-of-domain performance for both BiLSTM and ID-CNN models.

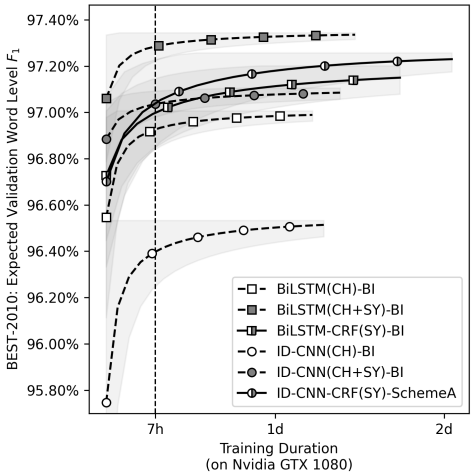


Figure 3: Expected validation performance $\text{WL}_{F_1}$ and training duration from ID-CNNs and BiLSTMs trained on a different set of features.

| **Method** | **OOV Recall** |
|---|---|
| Dictionary-based | 21.31% (24.70%) |
| DeepCut | 52.09% (60.47%) |
| BiLSTM | |
| (CH)-BI | 49.51% (59.58%) |
| (CH+SY)-BI | 58.62% (67.42%) |
| -CRF(SY)-BI | 58.37% (65.09%) |
| ID-CNN | |
| (CH)-BI | 44.83% (51.92%) |
| (CH+SY)-BI | 48.52% (64.09%) |
| -CRF(SY)-SchemeA | 56.53% (65.35%) |

Table 4: Out-of-Vocabulary (OOV) recalls from BEST-2010 test set and different methods. Values in parentheses are calculated by tokens. BiLSTM(CH)-BI and ID-CNN(CH)-BI are our control models, and the other BiLSTMs and ID-CNNs are the best model selected from hyperparameter search.

word segmentation.

We observed modest or inconsistent improvement from the CRF layers and the length sensitive tagsets (SchemeA and SchemeB). This might be because BiLSTMs and CNNs already take the context of the sequence into account. The pattern of tagsets is likely to be found by these models without explicitly given. This contradicts the previous findings where BIE tagsets were used effectively in Chinese (Nakagawa, 2004).

Additionally, we also used the methods proposed by Dodge et al. (2019) to calculate the expected performance given computational budget. Figure 3 shows the expected validation performance ($\text{WL}_{F_1}$) versus the computational budget displayed as training duration on Nvidia GTX 1080, which is the average training time times $n$, for different model configurations from BiLSTMs and ID-CNNs. We see that models with syllable features have higher expected validation performance than models using character features onlys (BiLSTM(CH)-BI and ID-CNN(CH)-BI). Hence, one needs significantly less amount of computation budget for hyperparameter optimization when incorporating syllable features.

Interestingly, when considering training duration budget below seven hours, Figure 3 illustrates that ID-CNN(CH+SY)-BI would yield higher expected performance than BILSTM-CRF(SY)-BI and ID-CNN-CRF(SY)-SchemeA. This might be useful information for practical settings. Although BiLSTM(CH+SY)-BI provides the most optimal expected validation performance and training duration curve, using such a model might not be efficient in certain settings that only CPU instances available. We shall discuss the issue in Section 8.

## 7 Explaining Word Segmenters

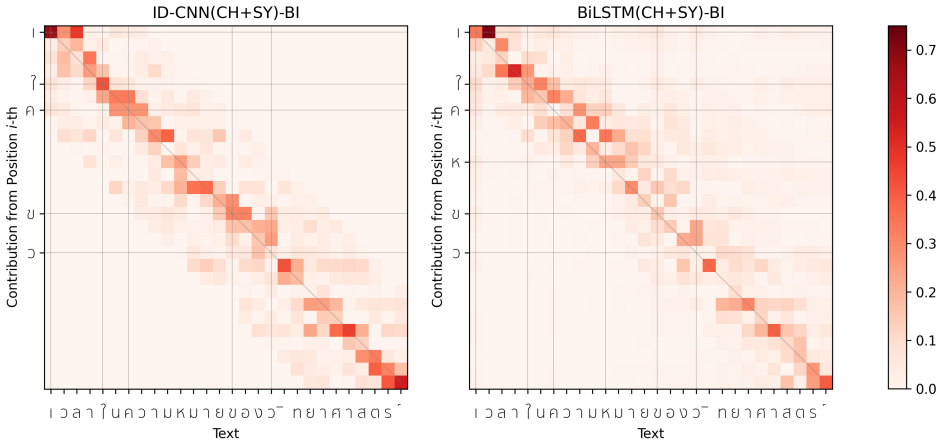


Figure 4: Explaining tagging predictions of เวลา|ใน|ความหมาย|ของ|วิทยาศาสตร์ (/weː laː naj kʰwaːm màːj kʰɔ́ːŋ wít jaː sáːt/, time in a scientific definition) from ID-CNN(CH+SY)-BI and BiLSTM(CH+SY)-BI using Integrated Gradients. The visualized attribution scores are absolute and normalized. Vertical and horizontal lines illustrate predicted word boundaries.

We perform further analysis on how ID-CNN(CH+SY)-BI and BiLSTM(CH+SY)-BI utilize the given features across the sequence. We use Integrated Gradients (Sundararajan et al., 2017) to explain tag predictions of these models by computing the attribution of each position to the prediction. We set the interpolation step size to 100 and use Captum[2] for Integrated Gradients computation. We use a sequence of padding characters as the baseline input sequence (Mudrakarta et al., 2018) and take the absolute values of the attribution scores.

Figure 4 shows the amount of attributions determining word segmentation prediction for เวลา|ใน|ความหมาย|ของ|วิทยาศาสตร์ (/weː laː naj kʰwaːm màːj kʰɔ́ːŋ wít jaː sáːt/, time in a scientific definition). One can see that the attributions concentrates around a narrow band surrounding the diagonal line for both ID-CNN(CH+SY)-BI and BiLSTM(CH+SY)-BI, suggesting high contributions from neighbor characters. Therefore, this evidence implies that going through all the locations in the sequence as done in BiLSTM(CH+SY)-BI might be not optimal for the Thai word segmentation task.

To provide quantitative measures, we select samples whose length is between 21 and 50 characters from the BEST-2010 test set, resulting in 338 samples. We then calculate the attribution amount allocated to each surrounding location up to 10 character left and right (Figure 5). Attribution scores are concentrated around three characters on both sides. This suggests that models such as ID-CNNs that consume features from only a selective set of neighbor locations are sufficient for our segmentation purpose, rather than going through the sequence as in BiLSTMs.

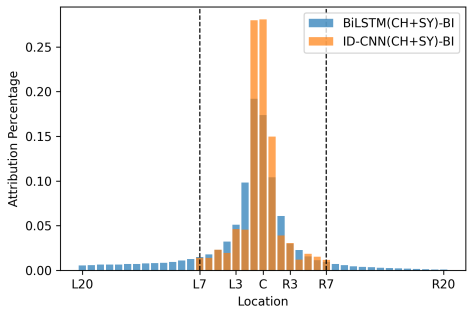


Figure 5: Distributions of normalized attribution scores from ID-CNN(CH+SY)-BI and BiLSTM(CH+SY)-BI using Integrated Gradients. Two dashed lines indicate the size of CNN filter.

Moreover, we also see here that the two attribution distributions are slightly left-skewed, suggesting models rely more on neighboring characters from the left side than the right side. This is reasonable because it aligns with the writing direction of Thai language.

[2] github.com/pytorch/captum/releases/tag/v0.2.0

| Method | BEST-2010 Test Set Segmentation Time (Wall Clock) | | |
|---|---|---|---|
| | t2.medium | p2.xlarge(bs=1) | p2.xlarge(bs=128) |
| DeepCut | 252.89 ± 9.93 (1×) | 134.54 ± 1.20 (1×) | - |
| PyThaiNLP† | 2.93 ± 0.07 (86×) | - | - |
| BiLSTM | | | |
| (CH)-BI | 123.29 ± 0.19 (2.1×) | 48.14 ± 0.05 (2.8×) | 10.12 ± 0.04 (1×) |
| (CH+SY)-BI | 77.04 ± 0.53 (3.3×) | 40.42 ± 0.38 (3.3×) | 17.65 ± 0.15 (0.6×) |
| -CRF(SY)-BI | 62.05 ± 0.56 (0.5×) | 52.73 ± 0.49 (2.6×) | 20.62 ± 0.08 (0.5×) |
| ID-CNN | | | |
| (CH)-BI | 11.87 ± 0.11 (21.3×) | 7.22 ± 0.01 (18.6×) | 5.28 ± 0.02 (1.9×) |
| (CH+SY)-BI | 23.48 ± 0.25 (10.8×) | 18.60 ± 0.09 (7.2×) | 16.56 ± 0.09 (0.6×) |
| -CRF(SY)-SchemeA | 28.32 ± 0.26 (8.9×) | 38.94 ± 0.54 (3.5×) | 19.28 ± 0.12 (0.5×) |

Table 5: BEST-2010 test set segmentation time from different methods and computing instances. `t2.medium` is a CPU instance, while `p2.xlarge` is a GPU instance. Numbers in parentheses indicate a speed factor (higher is faster) to the reference method displayed with (1×). *bs* stands for batch size.

## 8 Speed Benchmark

We perform speed benchmark of two existing methods, namely PyThaiNLP and DeepCut, and our word segmenters. We conduct the benchmark on two AWS cloud instances: `t2.medium` and `p2.xlarge`; the former is a CPU instance, while the latter is a GPU instance. We use UNIX's `time` to measure the wall clock of segmentation and AWS's official deep learning image[3]. For each method or configuration, we perform five runs in which the first two runs are burn-in, and we average segmentation time from the last three. We refer to Appendix A for our benchmark code.

Table 5 shows that ID-CNNs are generally faster than BiLSTMs on `t2.medium` and `p2.xlarge` when batch size is small (`bs=1`); they are on par for a large batch size (`bs=128`). Comparing to DeepCut, our syllable models are faster. More precisely, the best syllable variants of BiLSTMs and ID-CNNs are $3.3\times$ and $8.9\times$ faster than DeepCut respectively, on `t2.medium`. The speed factor of our best models reduce to around $3\times$ on `p2.xlarge(bs=1)`.

For `p2.xlarge(bs=128)`, we did not benchmark DeepCut in this setting because its API does not allow batch size configuration. We use BiLSTM(CH)-BI's time as the reference instead. From the table, we see that both BiLSTM-CRF(SY)-BI and ID-CNN-CRF(SY)-SchemeA are on par in terms of speed. Nevertheless, we consider here the results on `t2.medium` the most important one because word segmentation is one of the first step in NLP, and it should be efficient without a special hardware.

## 9 Conclusion

In this work, we propose to use syllable knowledge for Thai word segmentation. We perform a systematic comparison between existing Thai word segmenters and our different model configurations based on LSTMs and CNNs. Our results show that incorporating syllable can improve word segmentation performance significantly both in-domain and out domain evaluations. We also analyze results from hyperparameter search, and the analysis shows that using syllable features require less computational budget for the search to find word segmenters with decent performance.

We also explain two character-level word segmenters using Integrated Gradients, showing that the LSTM-based word segmenter relies only a faction of neighbor character when predicting tags of the sequence, suggesting that using CNNs models are more efficient than LSTMs. Our speed benchmark shows that our best and fastest syllable model ID-CNN-CRF(SY)-SchemeA is $8.9\times$ and $3.5\times$ faster than DeepCut, the current state of the art, on CPU and GPU instances respectively.

In future, we would like to investigate 1) the end-to-end learning paradigm, combining syllable and word segmentation into one model; 2) the effect of pretrained syllable embeddings; and 3) the effect of word segmentation on downstream tasks.

[3] AWS's Deep Learning AMI Linux 2 v.29

## 10 Acknowledgements

We thank Can Udomcharoenchaikit and Arthit Suriyawongkul for their helpful comments on the early version of this manuscript. We also appreciate the organizers of NeurIPS 2019's NewInML workshop for organizing the event, giving us an opportunity to present and receive feedback on the prior version of this work. PC is supported by the German Federal Ministry of Education and Research (BMBF) and the Max Planck Society. The project is also supported by Special Task Force for Activating Research (STAR) Ratchadapiseksompoch Fund from Chulalongkorn university, and Research Grant for New Scholars from Thailand Research Fund (MRG6280175) and Chulalongkorn University.

## References

Wirote Aroonmanakun, Kachen Tansiri, and Pairit Nittayanuparp. 2009. Thai National Corpus: A Progress Report. In *Proceedings of the 7th Workshop on Asian Language Resources*, ALR7, pages 153–158. Association for Computational Linguistics.

Wirote Aroonmanakun. 2002. Collocation and thai word segmentation. In *Proceedings of the 5th SNLP & 5th Oriental COCOSDA Workshop*, pages 68–75.

Wirote Aroonmanakun. 2007. Thoughts on word and sentence segmentation in Thai. In *Proceedings of the Seventh Symposium on Natural language Processing, Pattaya, Thailand, December 13–15*, pages 85–90.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Poramin Bheganan, Richi Nayak, and Yue Xu. 2009. Thai Word Segmentation with Hidden Markov Model and Decision Tree. In Thanaruk Theeramunkong, Boonserm Kijsirikul, Nick Cercone, and Tu-Bao Ho, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5476, pages 74–85. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of EMNLP*, pages 2185–2194, Hong Kong, China, November. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yoshiaki Kitagawa and Mamoru Komachi. 2018. Long short-term memory for Japanese word segmentation. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong, 1–3 December. Association for Computational Linguistics.

Rakpong Kittinaradorn, Titipat Achakulvisut, Korakot Chaovavanich, Kittinan Srithaworn, Chanwit Kaewkasi, Tulakan Ruangrong, and Krichkorn Oparad. 2019. DeepCut: A Thai word tokenization library using Deep Neural Network, September.

Theerapat Lapjaturapit, Kobkrit Viriyayudhakom, and Thanaruk Theeramunkong. 2018. Multi-Candidate Word Segmentation using Bi-directional LSTM Neural Networks. In *2018 International Conference on Embedded Systems and Intelligent Technology & International Conference on Information and Communication Technology for Embedded Systems (ICESIT-ICICTES)*, pages 1–6.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese word segmentation with bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium, October-November. Association for Computational Linguistics.

Pramod Kaushik Mudrakarta, Ankur Taly, Mukund Sundararajan, and Kedar Dhamdhere. 2018. Did the model understand the question? In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1896–1906. Association for Computational Linguistics.

Tetsuji Nakagawa. 2004. Chinese and Japanese word segmentation using word-level and character-level information. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 466–472, Geneva, Switzerland, aug 23–aug 27. COLING.

NECTEC. 2010. BEST: Benchmark for Enhancing the Standard of Thai Language Processing.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8024–8035.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. Version 0.20.2.

Terry Peng and Mikhail Korobov. 2018. Python-CRFSuite, August. Version 0.9.6.

Wannaphong Phatthiyaphaibun, Korakot Chaovavanich, Charin Polpanumas, Arthit Suriyawongkul, and Lalita Lowphansirikul. 2016. PyThaiNLP: Thai Natural Language Processing in Python, June.

Yuen Poowarawan. 1986. Dictionary-based Thai Syllable Separation.

Jitkapat Sawatphol and Attapol Rutherford. 2019. TNHC: Thai National Historical Corpus.

Sertis Co., Ltd. 2017. Thai word segmentation with bi-directional RNN. Commit: ce696cfad6bf3d949d0834a12e1569e1f7cf9e67.

Virach Sornlertlamvanich, Thatsanee Charoenporn, and Hitoshi Isahara. 1997. ORCHID: Thai part-of-speech tagged corpus. *National Electronics and Computer Technology Center Technical Report*, pages 5–19.

Virach Sornlertlamvanich. 1993. Word segmentation for Thai in machine translation system. *Machine Translation, NECTEC*, pages 556–561.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In Octavian Popescu and Carlo Strapparava, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2670–2680. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Thanaruk Theeramunkong and Sasiporn Usanavasin. 2001. Non-dictionary-based Thai word segmentation using decision trees. In *Proceedings of the first international conference on Human language technology research - HLT '01*, pages 1–5, San Diego. Association for Computational Linguistics.

Thanaruk Theeramunkong, Virach Sornlertlamvanich, Thanasan Tanhermhong, and Wirat Chinnan. 2000. Character cluster based thai information retrieval. In Kam-Fai Wong, Dik Lun Lee, and Jong-Hyeok Lee, editors, *Proceedings of the Fifth International Workshop on Information Retrieval with Asian Languages, 2000, Hong Kong, China, September 30 - October 01, 2000*, pages 75–80. ACM.

Wisesight (Thailand) Co., Ltd. 2019. Wisesight Sentiment Corpus.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

## A Code

Our code is available at github.com/heytitle/Syllable-based-Neural-Thai-Word-Segmentation.

## B Dataset Statistics

| Dataset | # Characters | # Words |
|---|---|---|
| BEST-2010 | | |
| Train | 18.7M | 5M |
| Validation | 2M | 0.55M |
| Test | 0.50M | 0.19M |
| TNHC | 2M | 0.67M |
| WiseSight-1000 | 76K | 22K |
| TNC | 8M | 2.56M† |

Table 6: Dataset statistics. (†): number of syllables.

## C Data Preprocessing

### C.1 Syllable Segmentation for Word Segmenters

For word segmenters that rely on syllable features, we first split the input sentence to separate parts based on punctuation signs. Then, each part is syllable-segmented, and the list of syllables of the sentence is formed from these parts' syllables.

### C.2 BEST-2010 Out of Vocabulary Experiment

We first select words that are in BEST-2010 test set but not in the training set. We then select only Thai words. More details can be found at `oov.py` in our repository.

### C.3 TNHC

We remove one file whose name is จดหมายเหตุรายวันของสมเด็จเจ้าฟ้ามหาวชิรุณหิศ.json out due to encoding issues. For the rest, we remove meta tags, such as author, data, or filename from each file. Our preprocess script can be found at `process-tnhc.py` in our repository.

### C.4 Character Type and Syllable Dictionaries

Our character type dictionary is drawn from DeepCut's character type dictionary [4]. It separates characters into 12 categories, shown in Table 7.

For the syllable dictionary, we extract all syllables from BEST-2010 training set and then map each syllable into a category if one of our regular expression rules applies. We have four categories, namely 1) punctuation, 2) URLs, 3) sequence of English alphabets, and 4) number.

### C.5 Benchmark

We remove name-entity tags and separators (|), both repeated and tailing ones from the reference and given input. We refer to our `benchmark.py` in the repository for more details.

## D Training Parameters and Hyperparameter Optimization Details

Table 8 shows our setting and training parameters, and Table 9-24 shows hyperparameter search space for each model family. During the training, we keep the best checkpoint of each model, i.e. the final model is the checkpoint of the epoch thas has the lowest validation loss.

[4] `https://github.com/rkcosmos/deepcut/blob/master/deepcut/utils.py`

| Category | Description |
|---|---|
| Alphabet 1 | กขฃคฅงจฉชซญฎฏฐฑฒฑณดตถทธนบปพฟภมยรลวศษสฬอ |
| Alphabet 2 | ฅฉผฟฌหฮ |
| Vowel 1 | ะำิีึืุูเ |
| Vowel 2 | แไใโ |
| Tone mark | ่ ้ ๊ ๋ |
| Symbol | ์ๆฯ. |
| Digit | 0123456789๑๒๓๔๕๖๗๘๙ |
| Quotation | ” ’ ‘ ’ |
| Lower case English alphabet | abcdefghijklmnopqrstuvwxyz |
| Upper case English aphabet | ABCDEFGHIJKLMNOPQRSTUVWXYZ |
| Space | |
| Others | |

Table 7: Character type mapping.

| **Training Parameter** | **Value** |
|---|---|
| Computing infrastructure | Nvidia GTX 1080 |
| Number of search trials | 20 |
| Optimizer | Adam |
| Learning rate schedule | ReduceOnPlateau |
| Learning rate patient | 0 epoch |
| Number of epochs | 20 |
| Batch size | |
| Character level models | 32 |
| Syllable level models | 128 |
| Embedding Dimension | |
| Character | 32 |
| Character type | 32 |
| Syllable | 64 |

Table 8: Training parameters.

| | |
|---|---|
| **Average training duration** | 92 minutes |
| **Average validation word-level $F_1$** | 96.55±0.44% |
| **Best validation word-level $F_1$** | 97.01% |
| **Best model's number of trainable parameters** | 2,209,348 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 8.10e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 3.04e-06 |
| LSTM cells | *uniform-interger(128, 512)* | 489 |
| linear layer | *uniform-interger(16, 48)* | 34 |
| dropout | *uniform(0, 0.5)* | 0.4048 |

Table 9: Best hyperparameter and search space for BiLSTM(CH)-BI.

| | |
|---|---|
| **Average training duration** | 112 minutes |
| **Average validation word-level $F_1$** | 97.06±0.28% |
| **Best validation word-level $F_1$** | 97.34% |
| **Best model's number of trainable parameters** | 1,391,035 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 7.99e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 1.94e-04 |
| LSTM cells | *uniform-interger(128, 512)* | 204 |
| linear layer | *uniform-interger(16, 48)* | 27 |
| dropout | *uniform(0, 0.5)* | 0.3850 |

Table 10: Best hyperparameter and search space for BiLSTM(CH+SY)-BI.

| | |
|---|---|
| **Average training duration** | 25 minutes |
| **Average validation word-level $F_1$** | 96.49±0.45% |
| **Best validation word-level $F_1$** | 97.09% |
| **Best model's number of trainable parameters** | 1,595,134 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 7.21e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 4.77e-04 |
| LSTM cells | *uniform-interger(128, 512)* | 274 |
| linear layer | *uniform-interger(16, 48)* | 36 |
| dropout | *uniform(0, 0.5)* | 0.1567 |

Table 11: Best hyperparameter and search space for BiLSTM(SY)-BI.

| | |
|---|---|
| **Average training duration** | 25 minutes |
| **Average validation word-level $F_1$** | 96.27±0.56% |
| **Best validation word-level $F_1$** | 97.08% |
| **Best model's number of trainable parameters** | 1,869,546 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 9.95e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 5.67e-04 |
| LSTM cells | *uniform-interger(128, 512)* | 326 |
| linear layer | *uniform-interger(16, 48)* | 26 |
| dropout | *uniform(0, 0.5)* | 0.2167 |

Table 12: Best hyperparameter and search space for BiLSTM(SY)-SchemeA.

| | |
|---|---|
| **Average training duration** | 23 minutes |
| **Average validation word-level $F_1$** | 96.47±0.42% |
| **Best validation word-level $F_1$** | 96.94% |
| **Best model's number of trainable parameters** | 2,173,258 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 8.96e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 2.53e-04 |
| LSTM cells | *uniform-interger(128, 512)* | 376 |
| linear layer | *uniform-interger(16, 48)* | 18 |
| dropout | *uniform(0, 0.5)* | 0.4452 |

Table 13: Best hyperparameter and search space for BiLSTM(SY)-SchemeB.

| | |
|---|---|
| **Average training duration** | 132 minutes |
| **Average validation word-level $F_1$** | 96.73±0.29% |
| **Best validation word-level $F_1$** | 97.20% |
| **Best model's number of trainable parameters** | 2,697,374 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 6.45e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 4.00e-04 |
| LSTM cells | *uniform-interger(128, 512)* | 448 |
| linear layer | *uniform-interger(16, 48)* | 28 |
| dropout | *uniform(0, 0.5)* | 0.1465 |

Table 14: Best hyperparameter and search space for BiLSTM-CRF(SY)-BI.

| | |
|---|---|
| **Average training duration** | 135 minutes |
| **Average validation word-level $F_1$** | 96.57±0.40% |
| **Best validation word-level $F_1$** | 96.90% |
| **Best model's number of trainable parameters** | 2,997,807 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 6.37e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 3.70e-05 |
| LSTM cells | *uniform-interger(128, 512)* | 484 |
| linear layer | *uniform-interger(16, 48)* | 39 |
| dropout | *uniform(0, 0.5)* | 0.4632 |

Table 15: Best hyperparameter and search space for BiLSTM-CRF(SY)-SchemeA.

| | |
|---|---|
| **Average training duration** | 128 minutes |
| **Average validation word-level $F_1$** | 96.41±0.35% |
| **Best validation word-level $F_1$** | 96.97% |
| **Best model's number of trainable parameters** | 2,572,503 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 7.81e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 6.53e-05 |
| LSTM cells | *uniform-interger(128, 512)* | 431 |
| linear layer | *uniform-interger(16, 48)* | 33 |
| dropout | *uniform(0, 0.5)* | 0.3202 |

Table 16: Best hyperparameter and search space for BiLSTM-CRF(SY)-SchemeB.

| | |
|---|---|
| **Average training duration** | 98 minutes |
| **Average validation word-level $F_1$** | 95.75±0.83% |
| **Best validation word-level $F_1$** | 96.53% |
| **Best model's number of trainable parameters** | 391,291 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 3.88e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 4.41e-04 |
| convolution filters | *uniform-interger(128, 256)* | 235 |
| linear layer | *uniform-interger(16, 48)* | 39 |
| dropout | *uniform(0, 0.5)* | 0.0008 |

Table 17: Best hyperparameter and search space for ID-CNN(CH)-BI.

| | |
|---|---|
| **Average training duration** | 105 minutes |
| **Average validation word-level $F_1$** | 96.88±0.16% |
| **Best validation word-level $F_1$** | 97.11% |
| **Best model's number of trainable parameters** | 1,161,791 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 9.20e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 1.17e-05 |
| convolution filters | *uniform-interger(128, 256)* | 200 |
| linear layer | *uniform-interger(16, 48)* | 47 |
| dropout | *uniform(0, 0.5)* | 0.2167 |

Table 18: Best hyperparameter and search space for ID-CNN(CH+SY)-BI.

| | |
|---|---|
| **Average training duration** | 46 minutes |
| **Average validation word-level $F_1$** | 96.47±0.41% |
| **Best validation word-level $F_1$** | 97.06% |
| **Best model's number of trainable parameters** | 1,136,281 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 9.01e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 2.27e-04 |
| convolution filters | *uniform-interger(128, 256)* | 208 |
| linear layer | *uniform-interger(16, 48)* | 29 |
| dropout | *uniform(0, 0.5)* | 0.2329 |

Table 19: Best hyperparameter and search space for ID-CNN(SY)-BI.

| | |
|---|---|
| **Average training duration** | 45 minutes |
| **Average validation word-level $F_1$** | 96.51±0.33% |
| **Best validation word-level $F_1$** | 97.02% |
| **Best model's number of trainable parameters** | 1,136,339 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 7.02e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 2.79e-04 |
| convolution filters | *uniform-interger(128, 256)* | 207 |
| linear layer | *uniform-interger(16, 48)* | 41 |
| dropout | *uniform(0, 0.5)* | 0.3650 |

Table 20: Best hyperparameter and search space for ID-CNN(SY)-SchemeA.

| | |
|---|---|
| **Average training duration** | 43 minutes |
| **Average validation word-level $F_1$** | 96.43±0.39% |
| **Best validation word-level $F_1$** | 96.86% |
| **Best model's number of trainable parameters** | 1,169,615 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 8.24e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 3.22e-05 |
| convolution filters | *uniform-interger(128, 256)* | 219 |
| linear layer | *uniform-interger(16, 48)* | 40 |
| dropout | *uniform(0, 0.5)* | 0.0875 |

Table 21: Best hyperparameter and search space for ID-CNN(SY)-SchemeB.

| | |
|---|---|
| **Average training duration** | 158 minutes |
| **Average validation word-level $F_1$** | 96.63±0.39% |
| **Best validation word-level $F_1$** | 97.22% |
| **Best model's number of trainable parameters** | 1,206,435 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 9.67e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 1.77e-04 |
| convolution filters | *uniform-interger(128, 256)* | 232 |
| linear layer | *uniform-interger(16, 48)* | 35 |
| dropout | *uniform(0, 0.5)* | 0.1696 |

Table 22: Best hyperparameter and search space for ID-CNN-CRF(SY)-BI.

| | |
|---|---|
| **Average training duration** | 155 minutes |
| **Average validation word-level $F_1$** | 96.70±0.38% |
| **Best validation word-level $F_1$** | 97.26% |
| **Best model's number of trainable parameters** | 997,375 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 8.55e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 5.86e-04 |
| convolution filters | *uniform-interger(128, 256)* | 150 |
| linear layer | *uniform-interger(16, 48)* | 19 |
| dropout | *uniform(0, 0.5)* | 0.1658 |

Table 23: Best hyperparameter and search space for ID-CNN-CRF(SY)-SchemeA.

| | |
|---|---|
| **Average training duration** | 158 minutes |
| **Average validation word-level $F_1$** | 96.76±0.33% |
| **Best validation word-level $F_1$** | 97.09% |
| **Best model's number of trainable parameters** | 1,092,547 |

| **Hyperparameter** | **Search Space** | **Best Assignment** |
|---|---|---|
| learning rate | *loguniform(1e-4, 1e-3)* | 9.67e-04 |
| weight decay | *loguniform(1e-6, 1e-3)* | 2.03e-04 |
| convolution filters | *uniform-interger(128, 256)* | 192 |
| linear layer | *uniform-interger(16, 48)* | 19 |
| dropout | *uniform(0, 0.5)* | 0.1111 |

Table 24: Best hyperparameter and search space for ID-CNN-CRF(SY)-SchemeB.