Domain Adaptation of Thai Word Segmentation Models using Stacked Ensemble

Peerat Limkonchotiwat[*], Wannaphong Phatthiyaphaibun[§], Raheem Sarwar[‡], Ekapol Chuangsuwanich[†], Sarana Nutanong[*]

[*]School of Information Science and Technology, VISTEC, Thailand
[§]Faculty of Interdisciplinary Studies, Khon Kaen University, Thailand
[‡]RGCL, University of Wolverhampton, United Kingdom
[†]Department of Computer Engineering, Chulalongkorn University, Thailand
{Peerat.l_s19,snutanon}@vistec.ac.th,wannaphong@kkumail.com,
R.Sarwar4@wlv.ac.uk, ekapolc@cp.eng.chula.ac.th

Abstract

Like many Natural Language Processing tasks, Thai word segmentation is domain-dependent. Researchers have been relying on transfer learning to adapt an existing model to a new domain. However, this approach is inapplicable to cases where we can interact with only input and output layers of the models, also known as "*black boxes*". We propose a filter-and-refine solution based on the stacked-ensemble learning paradigm to address this black-box limitation. We conducted extensive experimental studies comparing our method against state-of-the-art models and transfer learning. Experimental results show that our proposed solution is an effective domain adaptation method and has a similar performance as the transfer learning method.

1 Introduction

Word Segmentation (WS) is an essential process for several Natural Language Processing (NLP) tasks such as Part-of-Speech (PoS) tagging and Machine Translation (MT). The accuracy of WS significantly affects the accuracy of these NLP tasks, as shown in experimental results from Nguyen et al. and Chang et al.

While WS is considered relatively simple in English, it is still an open problem in languages without explicitly defined word delimiters, such as Thai, Chinese, and Japanese. However, unlike Chinese and Japanese, Thai WS did not receive much research attention. There are only six notable publications (Chormai et al., 2019; Nararatwong et al., 2018; Kongyoung et al.; Noyunsan et al.; Thanadechteemapat and Fung; Tongtep and Theeramunkong) on Thai WS for the past ten years. On the other hand, there are at least eight papers from well-established conferences on Chinese and Japanese WS (Li et al., 2019; Aguirre and Aguiar, 2019; Zhou et al.; Ma et al., 2018; Gong et al., 2017; Chen et al., 2017; Zhou et al., 2017; Cai et al., 2017) within only the last two years. This investigation focuses on the segmentation of Thai words since it is a challenging problem that has an excellent opportunity to improve, especially in the area of *domain adaptation*.

Like many NLP tasks, Thai WS is domain-dependent. For instance, Chormai et al. (2019) recorded an accuracy drop from 91% to 81% when their model trained on a generic domain corpus (Kosawat et al., 2009) was tested on a social media one (bact' et al., 2019). Results from our analysis (Section 3) also conform to these findings.

One way to solve the domain dependency problem is through Transfer Learning (TL), which is a common technique in domain adaptations (Schuster et al.; Chang et al.). However, TL may not be applicable when working with a commercial API or a model that does not support weight adjustments (Chormai et al., 2019; Chuang, 2019; Ikeda, 2018). We call this type of model a *black box*.

In this paper, we propose a stacked-ensemble learning solution to overcome the black-box limitation. Instead of making changes to the existing model directly, we build a separate model to improve the accuracy of predictions made by the black box. Our solution comprises two parts, Domain-Generic (DG) and Domain-Specific (DS). The pre-trained black box handles the Domain-Generic part, and a new model is constructed to handle the Domain-Specific part. All samples go through Domain-Generic, which makes initial predictions. We rank all predictions according to uncertainty and send the top-k uncertain predictions to Domain-Specific for further consideration. We combine the predictions from Domain-Specific with the remaining from Domain-Generic to form the final predictive results.

We conducted extensive experimental studies to assess our solution's performance against a base-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 3841–3847,
November 16–20, 2020.

line model and transfer learning solutions. We also applied our *Stacked-Ensemble Filter-and-Refine (SEFR)* technique to Chinese and Japanese. Experimental results showed that our proposed solution achieved the accuracy level comparable to those of transfer learning solutions in Thai. For Chinese and Japanese, we showed that model adaptation using the SEFR technique could improve the performance of black-box models when used in a cross-domain setting.

Our contributions are as follows. First, we propose a novel solution for adapting a black-box model to a new domain by formulating the problem as an ensemble learning one. Second, we derive a filter-and-refine method to speed up the inference process without sacrificing accuracy in some cases. Third, we conducted extensive experimental studies; experimental results validate the effectiveness of our solution. Fourth, we make our code available at: `github.com/mrpeerat/SEFR_CUT`

2 Stacked-Ensemble Method

2.1 Pipeline Structure

Figure 1 displays the pipeline structure of the proposed SEFR method, which consists of a *Domain-Generic (DG) black box*, *uncertainty filtering*, and a *Domain-Specific (DS) model*. Each character enters the pipeline through the Domain-Generic black box, which gives a softmax or logistic score from the Domain-Generic model as output. We then use this output to calculate the uncertainty score. Uncertainty values are used to rank and filter samples that need reexamination by the Domain-Specific model. We then merge the results from Domain-Specific with the direct answers from Domain-Generic to form the final answers.

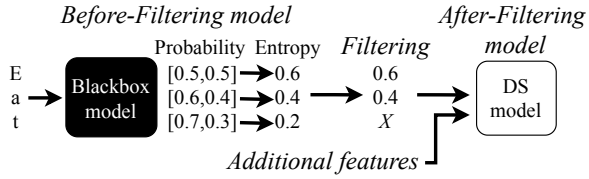


Figure 1: Overview of our *Stacked-Ensemble Filter-and-Refine (SEFR)* method

2.2 Pipeline Implementation

In this subsection, we consider how to implement the pipeline in Figure 1 effectively.

An effective filter and refine pipeline should have the following properties. First, before the filter, there is a general-decision maker that can make most decisions reasonably well. Second, the filter should be able to separate out decisions not requiring further consideration. Third, after the filter, there is a decision maker that can make the remaining decisions better than the general-decision maker. Using filter and refine can help reduce the computation time and avoid unnecessary errors from the Domain-Specific model which might not be as robust as the Domain-Generic model.

Before-Filtering Model. In our pipeline, the general-decision maker is the Domain-Generic black box. Specifically, we use the state-of-the-art pre-trained model (Rakpong Kittinaradorn, 2019; Chormai et al., 2019) constructed from a generic-domain corpus (Kosawat et al., 2009) to ensure the best possible performance in general cases.

Filtering. The prediction of an out-of-domain sample is likely to have an entropy higher than that of an in-domain one. Hence, we use the softmax entropy to separate the results from Domain-Generic into two groups: (i) *high-uncertainty predictions* that need further consideration from Domain-Specific; (ii) *low-uncertainty predictions* that we keep the results unchanged. The exact cut-off point can be fine-tune as a hyperparameter.

After-Filtering — Model. As stated earlier, the model placed after filtering should perform a certain task better than the one placed before filtering, which is domain specificity in this case. Hence, we use a Domain-Specific model trained with target-domain data to refine the uncertain predictions made by Domain-Generic. In theory, a Domain-Specific model can be constructed using any learning method. However, a DNN-based method may be inapplicable in a data-poor setting, which is the case in this investigation. As a result, we focus on classical learning methods that historically provide good results in WS problems, such as *Logistic Regression (LR)*, *Support Vector Machine (SVM)*, and *Conditional Random Field (CRF)*. Figure 2 shows performance evaluation results from different Domain-Specific implementations. As can be seen, CRF gave the best performance in comparison to other models.

After-Filtering — Input Features. We consider the following 4 sets of features. First, we use the n-gram windows to capture the context. Second, we use the dictionary index to identify whether each character can start a word (Horsuwan et al.). The next two feature sets are meta features obtained from the Domain-Generic model, i.e., the softmax output and the softmax entropy.

3 Performance Evaluation

We evaluated our SEFR solution against state-of-the-art models on nine benchmark corpora from three languages. Specifically we studied the effect of our SEFR method and report the performance by adapting a black-box model to a new domain by formulating the problem as an ensemble learning.

3.1 Performance Evaluation on Thai

Competitive Methods. Two state-of-the-art models for Thai WS were chosen as our competitive methods, i.e., DeepCut (Rakpong Kittinaradorn, 2019) and AttaCut-SC (Chormai et al., 2019). Both are deep learning models based on the *Convolution Neural Network (CNN)*. We also created two SEFR solutions using DeepCut and AttaCut-SC as the Domain-Generic model, and we called them SE+DeepCut and SE+AttaCut-SC, respectively. As domain-adaptation baselines, we applied transfer learning to DeepCut and AttaCut-SC and called them TL-DeepCut and TL-AttaCut-SC, respectively. We note that the authors of DeepCut provided the weights trained on the BEST corpus. We used the same architecture and parameter settings to update these weights on Wisesight and TNHC(Table 1). Attacut-SC does not provide weights to perform TL and requires retraining of the model. We trained the AttaCut-SC model using *BEST-2010* corpus (Kosawat et al., 2009) to obtain the best training weights to perform TL, where 90% of the data was used for training. We compared our method with a model pre-trained on BEST-2010 and then transferred to the target task.

Model	# Epoch	Batch Size	Optimizer	Learning rate
DeepCut	[3,10]	[256, 512, 2048, 4096, 8192]	Adam	0.001
AttaCut-SC	[3,10]	[256, 512, 2048, 4096, 8192]	Adam	0.001

Table 1: Parameter settings in Deepcut and AttaCut.

Datasets and Metrics. We evaluated our propose solution and the competitive methods using two Thai corpora. Wisesight (WS160) is a small scale corpus used for sentimental analysis on tweets. TNHC is a collection of Thai classical literature. Wisesight and TNHC are mostly used in domain adaptation experiments. See Table 2 for details. The model training was performed on 80% of the training set while the other 20% was used for tuning of hyperparameters, including the value of top-k.

The performance of the Thai WS is typically evaluated using F1 scores at the character level. However, if a word is wrongly tokenized, it may affect the tokenization of the following words. To avoid the overestimation of WS performance, we also evaluated the F1 scores at the word level.

Lang.	Corpora	# Sentence	# Word
TH	Wisesight	1K [0.16K]	22K [3.9K]
TH	TNHC	13K [7K]	374K [239K]
CN	AS	636K [13K]	4.8M [110K]
CN	CITYU	46K [1.1K]	1.2M [28K]
CN	MSR	56K [3.5K]	1.4M [91K]
CN	PKU	7.7K [1.1K]	371K [48K]
JP	GSD	7K [0.5K]	159K [12K]
JP	Modern	0.6K [0.16K]	11K [2.6K]
JP	PUD	0.7K [0.19K]	19K [5K]

Table 2: Summary of WS corpora (# Training [# testing]), TH = Thai, CN = Chinese, and JP = Japanese.

Experimental Studies. *Our Method vs Thai Competitive methods.* In this part of the paper, we compared our SEFR method against the state-of-the-art Thai WS methods on WS160 and TNHC. The experimental results given in Tables 3 and 4 show that for all corpora, i.e., WS160 and TNHC, our method (*SE+DeepCut*) outperformed the state-of-the-art DeepCut in the domain adaptation experiment. Moreover, *SE+AttaCut-SC* outperformed AttaCut-SC and TL-AttaCut-SC for all corpora. In particular, for the TNHC corpus, *SE+DeepCut* performed better than DeepCut by 1.7% and 0.3% at the character and word levels, i.e., char F1 and word F1, respectively. *SE+AttaCut-SC* outperformed AttaCut-SC and the different was 13.9% and 22.2% at the character and word levels respectively. The F1 Score gap between *SE+DeepCut* and TL-DeepCut was about 1.1% at the character level and 10.5% at the word level for both corpora. However, for AttaCut-SC, *SE+AttaCut-SC* performed better than TL in every corpus averaging about 12.9% at the character level and 12.7% at the word level. This result showed that our method could provide reasonable performance despite the low-accuracy predictions provided by the base model.

Effect of Top-k Percentage Entropy Selection. Figure 2 shows the effect of top-k percentage entropy selection on test sets of WS160 and TNHC using DeepCut as the Domain-Generic model. As expected of a filter and refine method, recall improves as the k value increases. Most of the recall improvements are from the lower range k values,

Method	Target = WS160	
	Char F1 (%)	Word F1 (%)
DeepCut	93.8	84.0
AttaCut-SC	93.5	84.0
TL-DeepCut	96.3	90.6
TL-AttaCut-SC	94.1	85.0
SE+DeepCut (k=100)	95.2	87.4
SE+AttaCut-SC (k=49)	94.5	85.6

Table 3: Performance comparison on WS160.

Method	Target = TNHC	
	Char F1 (%)	Word F1 (%)
DeepCut	93.5	75.4
AttaCut-SC	80.8	63.3
TL-DeepCut	95.4	88.6
TL-AttaCut-SC	81.2	71.8
SE+DeepCut (k=36)	95.2	84.1
SE+AttaCut-SC (k=100)	93.7	83.9

Table 4: Performance comparison on TNHC.

showing the effectiveness of the entropy-based filtering. For WS160, the F1 peaks at $k = 100$ due to the fact that precision also keeps increasing at every k value. In this case, our filter and refine method can be viewed as a re-scoring method. This is due to the effectiveness of *CRF*(Figures 2c and 2f) classifier. As shown in Figure 2, unlike *CRF*, increasing the k value past a certain threshold negatively affects the performance of *SVM*(Figures 2a and 2d) and *LR*(Figures 2b and 2e) models due to their weaker performance. Removing certain input features to the *CRF* model such as the sotfmax output also decrease the overall performance. For the TNHC dataset the best k value is around 30% showing the importance of filtering to reduce the potential candidates.

3.2 Evaluation on Chinese and Japanese.

Chinese Word Segmentation (CWS). In this experiment, we used the existing CWS model called PyWordSeg (Chuang, 2019) with character-level ELMO embedding. Normally, CWS categorizes characters into four classes: (i) beginning (B), (ii) internal (I), (iii) ending (E), and (iv) single-word (S) (Li et al., 2019). However, PyWordSeg classifies each character as boundary or non-boundary character which is similar to Thai WS, so we used the same feature as Thai WS in this experiment. We performed the experiment on *SIGHAN 2005* dataset (see Table 2) as a single corpora experiment not domain switch like Thai. However, PyWordSeg did not provide a probabilistic prediction that can be used to measure the uncertainty score on English and Pinyin characters. Therefore, we left out those sentences from the evaluation. Note that the released PyWordSeg code does not lend itself for straightforward transfer learning experiments, exemplifying our use case of treating models as black boxes.

For the CWS task, we evaluated our method on four corpora including AS, CityU, PKU, and MSR using the character-level F1. The results shown in Table 5 indicate that our method is better than the competitors.

Method	Corpus			
	AS	CITYU	MSR	PKU
PyWordSeg	98.4	98.6	85.3	83.2
SE+PyWordSeg (k=90,100,80,100)	98.5	98.8	94.5	94.0

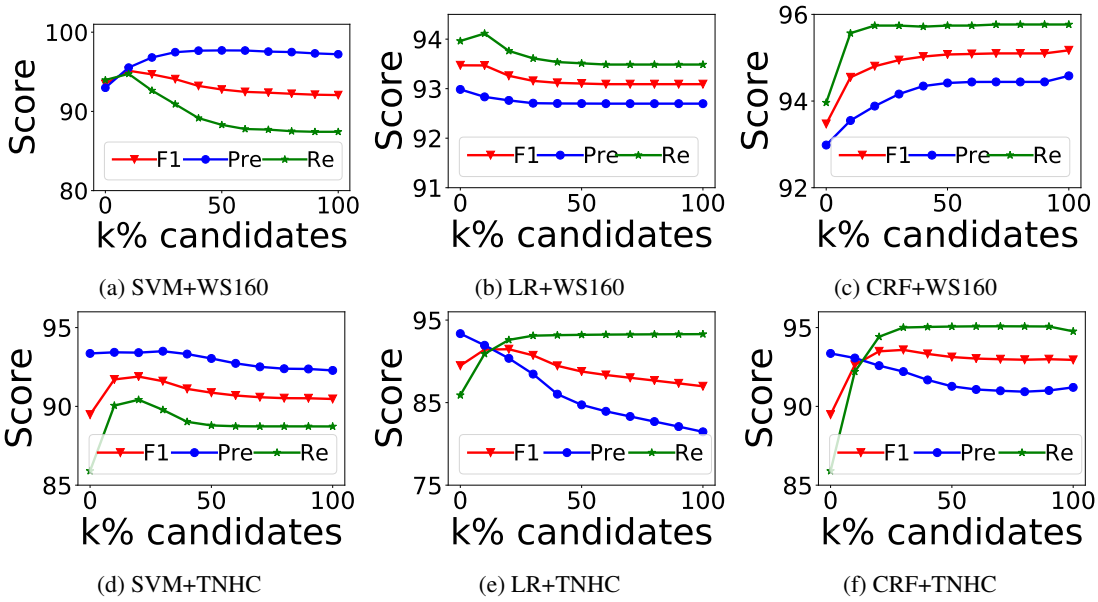
Table 5: Comparison between our method on CWS.

Japanese Word Segmentation (JWS). In this experiment, we performed JWS using Nagisa (Ikeda, 2018), trained on the *Balanced Corpus of Contemporary Written Japanese (BCCWJ)* (Maekawa et al.). This model categorizes characters into four classes: (i) beginning (B) (ii) middle (M) (iii) ending (E), and (iv) single-word (S) (Kitagawa and Komachi, 2018).

We performed experiments using the *Universal Dependencies 2.5* dataset (Asahara et al., 2018) on the GSD, Modern, and PUD subset (see Table 2). The results given in Table 6 indicate that our method outperformed JWS on all corpora. Specifically, our method reports performance improvement of 3% on GSD, 4.7% on Modern, and 11.5% on PUD using the character-level F1.

Method	Corpus		
	GSD	Modern	PUD
Nagisa	87.1	87.1	78.8
SE + Nagisa (k=100,100,100)	90.1	91.8	90.3

Table 6: Comparison between our method on JWS.



(a) SVM+WS160 (b) LR+WS160 (c) CRF+WS160

(d) SVM+TNHC (e) LR+TNHC (f) CRF+TNHC

Figure 2: Effect of top-k using SE+DeepCut (Chracter-level F1, Precision, and Recall)

4 Conclusion

We proposed a novel solution for adapting a black-box model to a new domain by formulating it as an ensemble learning problem. We conducted extensive experimental studies using nine benchmark corpora from three languages. For Thai Word Segmentation, the results showed that our method is an effective domain adaptation method and has similar performance as the transfer learning method. The results from Japanese and Chinese Word Segmentation experiments showed that our method could improve the performance of Japanese and Chinese black-box models.

References

Stalin Aguirre and Josafá Aguiar. 2019. A Japanese Word Segmentation Proposal. In *ACL*.

Masayuki Asahara, Hiroshi Kanayama, Takaaki Tanaka, Yusuke Miyao, Sumire Uematsu, Shinsuke Mori, Yuji Matsumoto, Mai Omura, and Yugo Murawaki. 2018. Universal Dependencies Version 2 for Japanese. In *LREC*.

bact', Pattarawat Chormai, Charin, and ekapolc. 2019. Pythainlp/wisesight-sentiment: First release.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and Accurate Neural Word Segmentation for Chinese. In *ACL*.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *WMT@ACL*.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-Criteria Learning for Chinese Word Segmentation. In *ACL*.

Pattarawat Chormai, Ponrawee Prasertsom, and Attapol Rutherford. 2019. Attacut: A Fast and Accurate Neural Thai Word Segmenter. *CoRR*, abs/1911.07056.

Yung-Sung Chuang. 2019. Robust Chinese Word Segmentation with Contextualized Word Representations. *CoRR*, abs/1901.05816.

Chen Gong, Zhenghua Li, Min Zhang, and Xinzhou Jiang. 2017. Multi-Grained Chinese Word Segmentation. In *EMNLP*.

Thanapapas Horsuwan, Kasidis Kanwatchara, Peerapon Vateekul, and Boonserm Kijsirikul. A comparative study of pretrained language models on thai social text categorization. In *ACIIDS*.

Taishi Ikeda. 2018. nagisa: A Japanese tokenizer based on recurrent neural networks. https://github.com/taishi-i/nagisa.

Yoshiaki Kitagawa and Mamoru Komachi. 2018. Long Short-Term memory for Japanese Word Segmentation. In *PACLIC*.

Sarawoot Kongyoung, Anocha Rugchatjaroen, and Krit Kosawat. Tlex+: A hybrid method using conditional random fields and dictionaries for thai Word Segmentation. In *KICSS*.

K. Kosawat, M. Boriboon, P. Chootrakool, A. Chotimongkol, S. Klaithin, S. Kongyoung, K. Kriengket, S. Phaholphinyo, S. Purodakananda, T. Thanakulwarapas, and C. Wutiwiwatchai. 2009. Best 2009 : Thai Word Segmentation software contest. In *SNLP*.

Xiaoya Li, Yuxian Meng, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is Word Segmentation Necessary for Deep Learning of Chinese Representations? In *ACL*.

Ji Ma, Kuzman Ganchev, and David Weiss. 2018. State-of-the-art Chinese Word Segmentation with Bi-LSTMs. In *EMNLP*.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. Balanced corpus of contemporary written Japanese. *Lang. Resour. Evaluation*.

Rungsiman Nararatwong, Natthawut Kertkeidkachorn, Nagul Cooharojananone, and Hitoshi Okada. 2018. Improving Thai Word and Sentence Segmentation Using Linguistic Knowledge. *IEICE Trans. Inf. Syst.*

Dat Quoc Nguyen, Thanh Vu, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. From Word Segmentation to POS tagging for Vietnamese. In *ALTA*.

Chaluemwut Noyunsan, Choochart Haruechaiyasak, Seksan Poltree, and Kanda Runapongsa Saikaew. A Multi-Aspect Comparison and Evaluation on Thai Word Segmentation programs. In *JIST*.

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

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual Transfer Learning for Multilingual Task Oriented Dialog. In *NAACL-HLT*.

Wigrai Thanadechteemapat and Lance Chun Che Fung. Thai word segmentation for visualization of Thai Web sites. In *ICMLC*.

Nattapong Tongtep and Thanaruk Theeramunkong. Simultaneous Character-Cluster-Based Word Segmentation and Named Entity Recognition in Thai Language. In *KICSS*.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, Xin-Yu Dai, and Jiajun Chen. 2017. Word-Context Character embeddings for Chinese Word Segmentation. In *EMNLP*.

Jianing Zhou, Jingkang Wang, and Gongshen Liu. Multiple Character Embeddings for Chinese Word Segmentation. In *ACL*.

A Appendices

A.1 Additional experimental details and results

Experimental environment. The experiments were conducted on Intel Core i9-9900X CPU @ 3.50GHz running on CentOS 7 with one Nvidia GeForce RTX 2080 Ti and 62 GB RAM. All the methods were implemented in Python and their performance and running time is provided in Table 7.

k **percentile**	**Char,Word F1**	**Running Time**
10	93.47, 89.48	0.34, 23.24
20	94.55, 92.63	0.37, 32.03
30	94.80, 93.49	0.42, 41.73
40	94.94, 93.58	0.47, 52.46
50	95.04, 93.33	0.51, 60.63
60	95.10, 93.16	0.56, 70.28
70	95.12, 93.04	0.61, 79.40
80	95.14, 92.98	0.67, 87.85
90	95.15, 92.97	0.70, 96.02
100	95.17, 92.95	0.75, 105.26

Table 7: Performance and Efficiency (Wisesight, TNHC): Effect of top-k.

c1	**c2**	**max_iterations**	**feature (possible_trainsitions)**
[1,0.1 **0.01**,0.001]	[1,0.1 **0.01**,0.001]	[200,500, **1000**]	True

Table 8: Parameter settings in CRF.

Evaluation measures. we measured the F1 scores for both character and world levels. For the character level, we used Sklearn (precision_recall_fscore_support) with binary average. For the word level, we applied the same practice as AttaCut (Chormai et al., 2019) in measuring the recall and precision. The performance comparison of our method on BEST corpus with DeepCut and AttaCut-SC is displayed in Table 10.

Ablation study: Feature. We also measured how different feature types affect the performance of our proposed solution, SE+DeepCut. Results are shown in Table 9. Note how the softmax features have the largest effect on performance, showing the importance of the DG model.

A.2 Error Analysis And Random Samples.

We performed an error analysis on Wisesight (test set) corpus for DeepCut and SE+DeepCut to show how we improve the baseline model. As shown

(Love Honda Civic because it's spacious and practical #HondaMyLove #EnjoyHondaThailand)

Actual ชอบ|Honda| |Civic| |เพราะ|มัน|กว้าง|เหมาะ|กับ|การ|ใช้|งาน| |#|ฮอนด้า|ที่รัก| |#|Enjoy|Honda|Thailand

DeepCut ชอบ|Honda| Civic |เพราะ|มัน|กว้าง|เหมาะ|กับ|การ|ใช้|งาน| |#ฮอนด้า|ที่|รัก| |#EnjoyHondaThailand

SE+DeepCut ชอบ|Honda| |Civic| |เพราะ|มัน|กว้าง|เหมาะ|กับ|การ|ใช้|งาน| |#|ฮอนด้า|ที่รัก| |#|Enjoy|Honda|Thailand

(Yesterday boba is delicioused, today senior is yummy // can't draw any more #TenSic)

Actual เมื่อ|วาน|ซานม|ไข่มุก|ก็|อร่อย| |วันนี้|รุ่น|พี่|ก็|อร่อย| |//|สูด|กาว|วาด|ไม่|ทัน|แล้ว| |#|เต็นสิค

DeepCut เมื่อ|วานซาน|ม|ไข่|มุก|ก็|อร่อย| |วัน|นี้|รุ่น|พี่|ก็|อร่อย| |/|/|สูด|กาววาด|ไม่|ทัน|แล้ว| |#เต็น|สิค

SE+DeepCut เมื่อ|วาน|ซานม|ไข่|มุก|ก็|อร่อย| |วัน|นี้|รุ่น|พี่|ก็|อร่อย| |//|สูด|กาววาด|ไม่|ทัน|แล้ว| |#|เต็น|สิค

Figure 3: Sentences from WS160 that DeepCut fails on

Dataset	Feature				Score (%)	
	N-gram Window	Softmax	Entropy	Dictionary Index	Char	Word
Train = Wisesight(WS1000) Test = Wisesight(WS160)		✓	✓	✓	93.7	83.8
	✓		✓	✓	91.3	76.6
	✓	✓		✓	95.0	86.4
	✓	✓	✓		92.1	83.6
	✓	✓	✓	✓	95.2	86.9

Table 9: Effect of Feature Types.

```
    Actual =|Mazda| |cx8| |มี|โอกาส|เข้า|มา|ใน|ไทย|มั้ย|ครับ
   DeepCut =|Mazda| |cx|8| |มี|โอกาส|เข้า|มา|ใน|ไทย|มั้ย|ครับ
SE+DeepCut =|Mazda| |cx8| |มี|โอกาส|เข้า|มา|ใน|ไทย|มั้ย|ครับ

    Actual =|สอบ|ถาม|หน่อย|นะ|คะ| |บลัชออน|ของ| |3ce| |สี|อะไร|คะ
   DeepCut =|สอบถาม|หน่อย|นะ|คะ| |บลัชออน|ของ| |3|ce| |สี|อะไร|คะ
SE+DeepCut =|สอบถาม|หน่อย|นะ|คะ| |บลัชออน|ของ| |3ce| |สี|อะไร|คะ

    Actual =|แต่|ผม|ชอบ|กิน|ช้าง|นะ| |เพราะ|ลีโอ|มัน|ออก|เปรี้ยว|ๆ
   DeepCut =|แต่|ผม|ชอบ|กิน|ช้าง|นะ| |เพราะ|ลีโอมัน|ออก|เปรี้ยว|ๆ
SE+DeepCut =|แต่|ผม|ชอบ|กิน|ช้าง|นะ| |เพราะ|ลีโอมัน|ออก|เปรี้ยว|ๆ
```

(a) Thai

```
    Actual =|这些|案件|反映|了|国有|企业|在|资金|管理|上|的|漏洞|。
 PyWordSeg =|这|些|案件|反映|了|国|有|企|业|在|资|金|管理|上|的|漏洞|。
SE+PyWordSeg =|这些|案件|反映|了|国有|企业|在|资金|管理|上|的|漏洞|。

    Actual =|机器人|迎|客|小姐|（|图片|）
 PyWordSeg =|机|器|人|迎客|小姐|（|图|片|）
SE+PyWordSeg =|机器|人|迎客|小姐|（|图片|）

    Actual =|数千|名|巴勒斯坦|和|以色列|妇女|在|耶路撒冷|举行|集会|（|图片|）
 PyWordSeg =|数|千|名|巴勒斯坦|和|以色列|妇|女|在|耶路撒冷|举|行集|会|（|图|片|）
SE+PyWordSeg =|数千|名|巴勒斯坦|和|以色列|妇女|在|耶路撒冷|举行|集会|（|图片|）
```

(b) Chinese

```
    Actual =|實用|に|適する|者|は|候文|に|し|て|既に|言ふ|所|と|異なり
    Nagasi =|實|用|に|適する|者|は|候|文|に|し|て|既に|言ふ|所|と|異なり
SE+Nagasi =|實用|に|適する|者|は|候|文|に|し|て|既に|言ふ|所|と|異なり

    Actual =|（|五十|八|葉）
    Nagasi =|（|五十|八|葉|）
SE+Nagasi =|（|五十|八|葉）

    Actual =|則ち|何|ぞ|獨り|文字|を|取ら|ざる|の|説|あら|んや
    Nagasi =|則ち|何|ぞ|獨り|文字|を|取ら|ざる|の|説|あら|ん|や
SE+Nagasi =|則ち|何|ぞ|獨り|文字|を|取ら|ざる|の|説|あら|んや
```

(c) Japanese

Figure 4: Random sentences from Thai, Chinese, and Japanese

Method	Target = BEST-2010	
	Char F1 (%)	Word F1(%)
DeepCut	98.1	92.6
AttaCut-SC	97.2	86.9
SE + DeepCut (k=1)	98.1	92.5
SE + AttaCut-SC (k=98)	98.0	89.8

Table 10: Performance comparison on BEST-2010.

in Figure 3, DeepCut was trained on BEST corpus where the annotation rules grouped words into a compound word resulting in a model that produced large word chunks. However, with not much English word in the corpus and no hashtag segmentation samples to train the model, DeepCut failed to segment English words correctly. On the other hand, the SE+DeepCut method was training on Wisesight (training set) therefore the behavior of our method is to split a compound word into multiple single word and our method can perform on English word and hashtag better than DeepCut. Thus, we need more data on the social media domain to support these domain characteristics with a good annotation guideline of data. We also add the random examples from Thai, Chinese, and Japanese the result are given in Figure 4.