Natural Language Engineering (2023), **29**, pp. 287–315
doi:10.1017/S1351324921000486

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Towards improving the robustness of sequential labeling models against typographical adversarial examples using triplet loss

Can Udomcharoenchaikit[1], Prachya Boonkwan[2] and Peerapon Vateekul[1,*]

[1]Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand and
[2]Language and Semantic Technology Lab (LST), NECTEC, Pathumthani, Thailand
*Corresponding author. E-mail: peerapon.v@chula.ac.th

(Received 12 September 2020; revised 20 December 2021; accepted 27 December 2021;
first published online 4 February 2022)

Abstract

Many fundamental tasks in natural language processing (NLP) such as part-of-speech tagging, text chunking, and named-entity recognition can be formulated as sequence labeling problems. Although neural sequence labeling models have shown excellent results on standard test sets, they are very brittle when presented with misspelled texts. In this paper, we introduce an adversarial training framework that enhances the robustness against typographical adversarial examples. We evaluate the robustness of sequence labeling models with an adversarial evaluation scheme that includes typographical adversarial examples. We generate two types of adversarial examples without access (black-box) or with full access (white-box) to the target model's parameters. We conducted a series of extensive experiments on three languages (English, Thai, and German) across three sequence labeling tasks. Experiments show that the proposed adversarial training framework provides better resistance against adversarial examples on all tasks. We found that we can further improve the model's robustness on the chunking task by including a triplet loss constraint.

Keywords: Tagging; Evaluation; Part-of-speech tagging; Information extraction

1. Introduction

Sequence labeling is a common natural language processing (NLP) task. Many NLP tasks such as part-of-speech (PoS) tagging, named-entity recognition (NER), and chunking can be formulated as sequence labeling tasks. The goal of sequence labeling tasks is to assign a categorical label to each word in a natural language sequence. Recent advancements in NLP have shown that neural networks can be a very effective sequence labeler. The BiLSTM-CRF architecture is one of the better-known methods which has become a standard technique for building a sequential tagger (Huang *et al.* 2015; Lample *et al.* 2016). Recent works improve the performance of BiLSTM-CRF models by using contextualized word embeddings (Peters *et al.* 2018; Akbik *et al.* 2018, 2019). These methods have been shown to obtain very high evaluation scores across multiple sequence labeling tasks. However, sequential taggers are often evaluated only on held-out datasets, and their performances drop substantially when evaluated against misspelled texts. Misspellings are common in real-world tasks and can adversely affect the performance of sequential labeling models. Nevertheless, most NLP systems are not explicitly trained to address spelling errors[a].

State-of-the-art machine learning systems are sensitive to subtle changes in inputs. These small changes can lead to a drastic drop in performance. In computer vision, it has been shown

[a]In this work, we consider both intentional and unintentional spelling mistakes as spelling errors.

that machine learning models consistently misclassify perturbed examples with differences that are indistinguishable from the original unperturbed examples (Goodfellow *et al.* 2015). These examples are also called *adversarial examples*. While adversarial examples are more common in computer vision, recent progress has also been made in NLP as well. Due to the discrete nature of textual inputs, it is almost impossible to create imperceptible perturbation on textual inputs. Adversarial examples in NLP are perturbations, such as spelling errors or word substitutions. NLP systems are also vulnerable to these adversarial examples. Textual adversarial examples have broken NLP systems for fundamental tasks in NLP such as text classification (Ebrahimi *et al.* 2018), machine translation (Belinkov and Bisk 2018), and reading comprehension systems (Jia and Liang 2017).

Previous literature on adversarial examples for NLP systems mostly focuses on text classification or machine translation. Few studies on adversarial examples for NLP have been done for sequential tagging tasks, even though these tasks are fundamental tasks in NLP. They are the building blocks for many NLP applications, especially applications without enough data to be trained end-to-end. We will illustrate the importance of studying adversarial examples in sequential tagging tasks with two use cases:

(1) **Privacy protection**: consider clinical de-identification, we can apply NER to identify and remove sensitive information from clinical records (Uzuner *et al.* 2007; Liu *et al.* 2017). Nevertheless, it is possible to train an attacker which changes some characters in the text to reveal a person's identity and renders a de-identification system useless.

(2) **Improving robustness**: adversarial examples are designed to maximize a loss function of a model. Hence, they provide a worst-case scenario benchmark to test the robustness of a model. Furthermore, we argue that sequential labeling tasks can provide a deeper understanding of robustness against spelling errors. Since sequential labeling tasks come with annotation for every word in the corpus, we can pinpoint precisely where a sequential tagger fails.

Previous literature on robust sequential taggers focuses on improving the performance of sequential taggers on out-of-distribution texts. They rely on adversarial training methods to improve the robustness of their models by perturbing the embeddings in continuous space instead of perturbing the input texts (Yasunaga *et al.* 2018; Zhou *et al.* 2019). Alternatively, Bodapati *et al.* (2019) augment training data with upper-cased and lower-cased sentences to improve the robustness of the target model. Previous literature on sequential labeling does not consider the worst-case attacks. These works also define the robustness of sequential taggers by testing them on social media data (Zhou *et al.* 2019), on infrequent words (Yasunaga *et al.* 2018) or on sentences with capitalization errors (Bodapati *et al.* 2019); however, none of them includes adversarial spelling errors as a benchmark for robustness. Given that adversarial spelling errors can be intentionally crafted to break a sequential tagger, it is important to use it as one of the benchmarks for robustness.

In this paper, we aim to build a sequential labeling system that is robust against spelling errors. Drawing inspiration from computer vision, we apply triplet loss (Schroff *et al.* 2015) to constrain the similarity between clean texts and their parallel perturbed texts. We propose an adversarial training framework that combines an adversarial training technique with a deep metric learning constraint such as triplet loss. In order to simulate the discrete nature of text, instead of adding noises to the embeddings layer, we explicitly perturb the textual inputs for adversarial training. We evaluate the robustness against typographical errors with black-box and white-box adversarial examples to reward robust systems. These adversarial examples are created by generating typos within an input sequence. Black-box adversarial examples do not assume any access to the model parameters, and they show us the robustness of the models against typographical errors in general. On the other hand, white-box adversarial examples are generated with full access to the model

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

parameters to find perturbations that maximize loss for each input sequence. Hence, white-box adversarial examples can provide an insight towards the worst-case perturbations to the textual inputs. We experimented on multiple sequential tagging tasks (part-of-speech tagging, named-entity recognition, and chunking) across 3 languages: English—CoNLL2003 (Tjong Kim Sang and De Meulder 2003), Thai[b]—ORCHID (Sornlertlamvanich *et al.* 1998), and German—TüBa-D/Z (Telljohann *et al.* 2004). Moreover, we also conducted experiments on datasets with natural spelling errors to benchmark sequential taggers in real-world settings. Here we show that the proposed adversarial training framework yields substantial gains over the baseline training method on adversarial test sets on all tasks. In addition, triplet loss can further improve the robustness of the model on the chunking task.

This paper has two main contributions:

(1) **An adversarial framework** that improves the robustness of a sequential tagger against adversarial examples and misspelled data. By incorporating adversarial training with triplet loss, we propose a novel sequential tagger training technique (AT-T) that constrains the similarity between clean and perturbed samples (Section 3.3.2).

(2) **An adversarial evaluation scheme** for sequential tagging tasks that uses both white-box and black-box adversarial examples to reveal weaknesses of sequential taggers (Section 3.4).

2. Related work

Before the deep learning era, most sequential tagging models were linear statistical models such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Lafferty *et al.* 2001; Nguyen and Guo 2007; Ratinov and Roth 2009). These sequential labeling models are heavily dependent on task-specific resources and hand-crafted features to improve their performances. It is important to highlight that these task-specific resources and features are costly to develop and often do not apply to other tasks (Ma and Xia 2014).

In recent years, deep learning techniques have been successfully applied to linguistic sequence labeling tasks without having to rely on hand-crafted features. Current top-performing approaches often use BiLSTM-CRF as a core component in their architectures (Akbik *et al.* 2018; Peters *et al.* 2018; Xin *et al.* 2018; Yasunaga *et al.* 2018; Straková *et al.* 2019). By finetuning or using fixed-features extracted directly from the transformer-based models such as Bidirectional Encoder Representations from Transformers (BERT), sequential taggers also achieve high performance scores across multiple sequential labeling datasets (Devlin *et al.* 2019; Heinzerling and Strube 2019). Other top-performing approaches use differentiable neural architecture search (NAS) to find an optimal architecture for a sequential tagging task (Jiang *et al.* 2019). However, these works have shown a strong performance on held-out datasets. They often overlook the ability to generalize to out-of-distribution cases such as spelling errors.

Previous literature on adversarial examples has exposed brittleness in machine learning systems by showing that subtle changes to the input can lead to failures in prediction outcomes (Szegedy *et al.* 2014; Goodfellow *et al.* 2015). While adversarial examples are more common in computer vision, there is a growing literature in NLP on adversarial examples. For instance, Miyato *et al.* (2017) construct adversarial examples by using perturbation on continuous word embeddings instead of discrete textual inputs. A problem with this approach is that it ignores the discrete nature of textual data.

In order to apply perturbation to the textual input, Jia and Liang (2017) evaluate the robustness of question-answering systems by adding a generated sentence to distract systems without changing the correct answer. Gao *et al.* (2018) introduce a black-box attack for text classifiers by

[b]For discussion on the implications of applying sequential tagger on the Thai language, please see Appendix B

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

injecting small misspelling errors into a text sequence. Belinkov and Bisk (2018) reveal weaknesses of character-based neural machine translation (NMT) models by evaluating them on misspelled texts. Belinkov and Bisk (2018) use rule-based synthetic spelling errors and natural spelling errors to evaluate the model. Ebrahimi *et al.* (2018) investigate white-box adversarial examples; they trick a character-level model by generating adversarial examples by choosing character-edit operations, such as deletion, insertion, and substitution ("flip"), based on the gradients of the target model. Michel *et al.* (2019) enforce constraints on adversarial examples so that they are meaning-preserving on the source side. Wallace *et al.* (2019) propose input-agnostic universal adversarial triggers with white-box access to a model's parameters. They study adversarial attacks to help analyze and diagnose NLP systems. Furthermore, Gardner *et al.* (2020) propose a new evaluation paradigm by creating contrast sets. A contrast set consists of test instances that are manually perturbed by domain experts in order to change their gold labels. This is in contrast to adversarial examples, where inputs are perturbed to change a model's decisions but their gold label do not change. Yet, to our knowledge, there is no adversarial attack framework for sequential tagging task.

To overcome these weaknesses, previous literature shows that NLP systems with subword-level or character-level embeddings are able to generalize to out-of-vocabulary words (Ling *et al.* 2015; Sennrich *et al.* 2016). Moreover, subword embeddings and character-level embeddings can help avoid filling up word-level embeddings with a large number of vocabulary. However, NLP systems with subword-level or character-level embeddings are often trained on clean data, and their performance drops drastically when they encounter misspelled words (Belinkov and Bisk 2018). Piktus *et al.* (2019) alleviate this issue by training subword embeddings to be robust against typographical errors by adding the spell correction loss function to the fastText (Bojanowski *et al.* 2017) loss function, which allows misspelled data to be incorporated. They benchmark the robustness of their models on misspelled texts by perturbing words in the test set. Their perturbations are based on spelling errors collected from a search engine's query logs, which is only publicly available in English. This is not applicable to other languages without such resources. Belinkov and Bisk (2018) artificially inject a training set with misspelled variants of each word. This technique of increasing robustness by training on noisy data is called adversarial training.

Adversarial training (Goodfellow *et al.* 2015) is a standard method for improving robustness against adversarial examples. It improves the robustness of a model by training on both unperturbed original examples and perturbed examples. Liu *et al.* (2020) improve robustness by including noisy sentences in the training process and also using a loss function to constraint the similarity between clean and perturbed representations. For sequential tagging tasks, Yasunaga *et al.* (2018) improve the robustness of the BiLSTM-CRF model on infrequent and unseen words using adversarial training method. They generate adversarial examples by adding small perturbation to continuous word embeddings. Pruthi *et al.* (2019) overcome adversarial misspellings on text classification tasks by using a word recognition model. Liu *et al.* (2020) use character embeddings, the adversarial training method, and a similarity constraint to improve robustness against character-level adversarial examples on text classification tasks.

Previous literature on robust sequential taggers focuses on improving the performance of sequential taggers on out-of-distribution texts. They rely on adversarial training methods to improve the robustness of their models by perturbing the embeddings in continuous space instead of perturbing the input texts (Yasunaga *et al.* 2018; Zhou *et al.* 2019). Yasunaga *et al.* (2018) conduct part-of-speech tagging experiments on multiple languages on the Pen Treebank WSJ corpus (English) and the Universal Dependencies dataset (27 languages). Yasunaga *et al.* (2018) find that adversarial training improves overall accuracy. In addition, adversarial training alleviates over-fitting in low resource languages, and it also increases part-of-speech tagging accuracy for infrequent and unseen words. Yasunaga *et al.* (2018) benchmark the robustness of the models on rare and unseen words. Zhou *et al.* (2019) focus on NER tasks. They address data imbalance issue by incorporating label statistics to the CRF loss and combine it with focal loss. Zhou *et al.* (2019)

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 1. The typographical errors for creating adversarial examples. * Note that capitalization error is not applicable to writing systems without capitalization such as Thai

Typographical error	English example	Thai example	German example
Insertion	Indonesia → Indonesiia	บรรทม /bant^{h}om/ → บรรีทม	Laufen → Laufjen
Transposition	Indonesia → Indnoesia	บรรทม /bant^{h}om/ → บรทรม	Laufen → Luafen
Deletion	Indonesia → Indnesia	บรรทม /bant^{h}om/ → บรทม	Laufen → Lafen
Substitution	Indonesia → Indonedia	บรรทม /bant^{h}om/ → บรรมม	Laufen → Laofen
Capitalization*	Indonesia → iNDONESIA		Laufen → laufen

benchmark the robustness of the models on user-generated data. (Bodapati *et al.* 2019) improve robustness to capitalization errors in NER by using data augmentation. Their data augmentation technique augments the training set with sentences with all upper-cased characters as well as sentences with all lower-cased characters. This allows a model to learn to exploit or ignore orthographic information depending on the contextual information.

The main drawback of previous literature on sequential taggers is that they rely on evaluating the held-out test sets to benchmark sequential tagging models. This leads to an overestimation of the models' performance that neglects their ability to generalize. Recent literature on robust sequential taggers fills this gap by benchmarking sequential taggers on out-of-distribution texts. In this work, we extend this idea by benchmarking sequential taggers on adversarial texts to estimate the worst-case scenario. We also explore triplet loss as a training strategy for improving the robustness of sequential taggers.

3. Methodology

The central idea of this work is that we want to train a robust model against misspelled texts. An ideal model should perform well on sequence labeling tasks on the normal held-out datasets and robust against text perturbation. In this section, we discuss methods for creating adversarial examples, our training methods, and an adversarial evaluation scheme that benchmarks the performance of our models in different scenarios.

3.1. Adversarial examples

Adversarial examples are inputs intentionally designed to degrade the performance of well-trained models. In this work, we create adversarial examples with textual perturbation to simulate a scenario where a model encounters misspelled text. Table 1 shows five types of typographical errors used to generate adversarial examples used in this paper. Previous literature used insertion, transposition, deletion, and substitution errors in their experiments (Belinkov and Bisk 2018; Heigold *et al.* 2018; Karpukhin *et al.* 2019). Because capital letters are essential features to sequential tagging tasks such as NER (Nebhi *et al.* 2015; Bodapati *et al.* 2019), we also add capitalization errors to our experiment, in which we swap lowercase characters to uppercase characters and vice-versa. By synthetically altering spellings of words within the existing corpora, we do not have to collect and annotate new corpora.

We apply the following constraints to ensure that these adversarial examples are intelligible to humans. We only perturb words with at least four characters. We do not alter the first or last characters except for transposition and capitalization errors. For transposition error, we do not alter the first character. For capitalization error, almost all alterations to capitalization do not stop humans from inferring the original text.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

3.1.1. Black-box adversarial examples

In a black-box setting, we do not assume any access to the model's parameters. For each word, we sample a type of perturbation from a discrete uniform distribution. Then, we sample one of all the possible perturbations of the sampled type from a discrete uniform distribution. Note that we only use insertion for evaluation only, and we exclude it from adversarial training.

3.1.2. White-box adversarial examples

In a white-box setting, we have full access to the model's parameters to create worst-case adversarial examples. We extend previous literature (Ebrahimi *et al.* 2018; Michel *et al.* 2019; Wallace *et al.* 2019) to make it suitable for our work, by including typographical errors as adversarial inputs. Given an input sequence of n words $X = \{x_1, \ldots, x_n\}$, it is typical to form a vector representation $\mathbf{x}_i$ for each word. We first start by selecting the position i^* with the largest gradient magnitude. Then, we find a word-level adversarial example $\hat{x}$ for the word x_i by satisfying the following optimization problem:

$$\arg\max_{\hat{x} \in \mathcal{C}} \mathcal{L}_{\text{target}} \left(x_0, \ldots, x_{i-1}, \hat{x}, x_{i+1}, \ldots, x_n\right) \tag{1}$$

where $\mathcal{C}$ is all possible combinations of perturbations for the word x_i, and $\hat{x}$ is one of the possible perturbations. $\mathcal{L}_{\text{target}}$ is the target loss function. For all white-box attacks, we use a standard loss function without auxiliary losses: $\mathcal{L}_{\text{target}}$. To save computational cost, Equation (1) can be approximated using the first-order approximation as discussed in (Ebrahimi *et al.* 2018; Michel *et al.* 2019; Wallace *et al.* 2019):

$$\arg\max_{\hat{x} \in \mathcal{C}} \left[\hat{\mathbf{x}} - \mathbf{x}_i\right]^\top \nabla_{\mathbf{x}_i} \mathcal{L}_{\text{target}} \tag{2}$$

As shown in Algorithm 1, at each iteration, we select an index with the largest magnitude. Then, we perturb the selected index by choosing a perturbation that causes the greatest loss. Then, we replace the perturbed word to the selected index in the input sentence. We repeat the same process without perturbing the already perturbed word until every word in the input sequence is perturbed. The final result is a perturbed text sequence (X_p).

Figure 1 illustrates a text sequence collected from the CoNLL2003 test set along with its black-box and white-box perturbed versions. Figure 1 also includes predictions from the baseline model (ELMo enhanced BiLSTM-CRF, (Peters *et al.* 2018)) for the NER task.

3.2. Adversarial training

Adversarial training (AT) is a standard technique for improving the robustness of deep learning models against adversarial examples. It withstands adversarial attacks by injecting adversarial examples into training data. We follow a black-box adversarial training procedure, where we augment a training set with their parallel perturbed text sequences without any internal knowledge of the target neural networks. We generate adversarial examples using the method discussed in Section 3.1.1.

At the beginning of every epoch, we dynamically generate black-box adversarial examples for all the training samples to train the model on both clean examples and typographical adversarial examples. Hence, we would have different data augmentation every epoch. This allows us to obtain robustness against variety of perturbed input texts. At each training step, we define the loss function for adversarial training as:

$$\mathcal{L}_{AT} = \mathcal{L}_{clean}(\boldsymbol{\theta}; X, y) + \gamma \mathcal{L}_{perturbed} \left(\boldsymbol{\theta}; X_{\text{p}}, y\right) \tag{3}$$

where $\mathcal{L}_{clean}(\boldsymbol{\theta}; X, y)$ and $\mathcal{L}_{perturbed} \left(\boldsymbol{\theta}; X_{\text{p}}, y\right)$ represent the loss from a clean text sequence and the loss from its perturbed pair, respectively. $\mathbf{y}$ represents a sequence of n predictions, where

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Algorithm 1: White-box adversarial attack.

Input: Original text sequence (X)
Output: Perturbed text sequence (X_p)
(1. Initialize the list of perturbed words and fill it with indexes of words that will not be perturbed e.g. words with less than 4 characters)
perturbed_list$\leftarrow \varnothing$;
foreach *word* x_i *in the sequence* X **do**
  if $|x_i| < 4$ *characters* **then**
    perturbed_list $\leftarrow$ perturbed_list $\cup\{i\}$;
  end
end
(2. Each iteration, generate an adversarial example and replace the original word with it)
while $\exists i \notin perturbed\_list$ **do**
  back-propagate $\mathcal{L}_{\text{target}}$ to get gradient $\nabla_{\mathbf{x}_i}\mathcal{L}_{\text{target}}$;
  $i^* \leftarrow \underset{i \notin perturbed\_list}{\arg\max} \|\nabla_{\mathbf{x}_i}\mathcal{L}_{\text{target}}\|_2$;
  perturbed_list $\leftarrow$ perturbed_list $\cup\{i^*\}$;
  $\hat{\mathbf{x}}^* \leftarrow \underset{\hat{x}\in C}{\arg\max} [\hat{\mathbf{x}} - \mathbf{x}_i]^\top \nabla_{\mathbf{x}_i}\mathcal{L}_{\text{target}}$;
  $X[i^*] \leftarrow \hat{x}^*$;
end
$X_p \leftarrow X$;
return X_p;

$\mathbf{y} = (y_1, y_2, \ldots, y_n)$. $\boldsymbol{\theta}$ represents learnable parameters of the model. $\gamma \in (0, 1)$ is a constant weight (hyperparameter) for the loss from the perturbed text input. We used $\gamma = 0.2$ for all of our experiments.

3.3. Auxiliary losses

Here we introduce two auxiliary losses used in our experiments: pair similarity loss and triplet loss. The main idea of these losses is that we want to have control over similarity between an encoded representation of a clean input text sequence and an encoded representation of a perturbed input text sequence.

A sequential encoder takes an input sequence of vectors $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and returns an output sequence of vectors $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$, which represents information about the sequence at each step.

By averaging the output representations, we produce a fixed dimensional embedding representation of a text sequence. Therefore, it is possible to compare similarity between multiple input sequences of different lengths.

$$s = \frac{\sum_i h_i}{|X|} \tag{4}$$

where s is a fixed length sequence encoding vector and $|X|$ is a number of words in the sequence.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

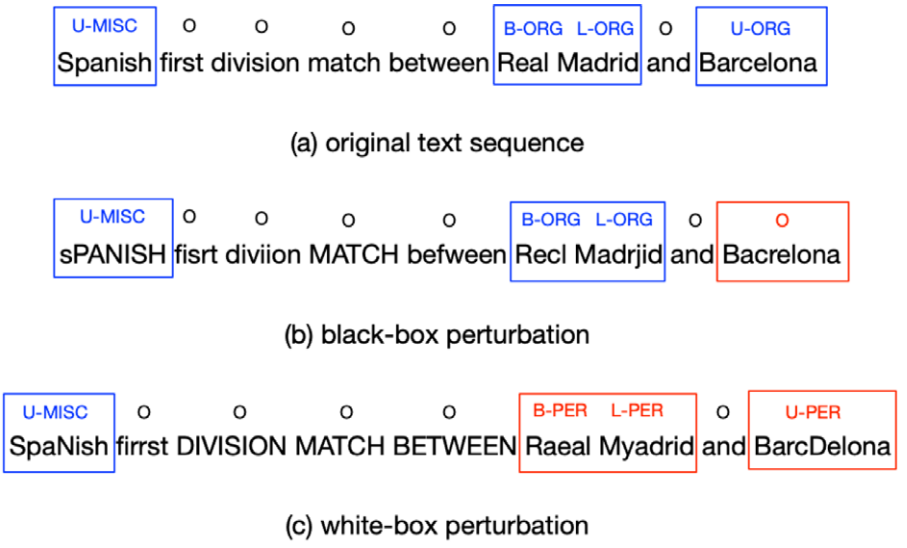

Figure 1. Our adversarial evaluation scheme (see section 3.4) expands one held-out test set into three parallel test sets: (1) **original test set** contains original unperturbed text, (2) **black-box adversarial test set** contains perturbed samples generated via black-box adversarial attacks, and (3) **white-box adversarial test set** contains perturbed samples generated using gradient-based white-box perturbation. This figure shows real samples collected from three parallel CoNLL2003 test sets with predictions from the baseline model (ELMo enhanced BiLSTM-CRF) from a supplementary experiment in Appendix A. Blue texts and boxes refer to correct predictions. Red texts and boxes refer to incorrect predictions.

We normalize s using L2 normalization since the squared Euclidean distance between two vectors is related to their cosine similarity when they are normalized to unit length.

$$z = \frac{s}{\max\left(\|s\|_2, \epsilon\right)} \tag{5}$$

where ϵ is a small positive constant value to circumvent division by zero. We used $\epsilon = 10^{-12}$ in all of our experiments.

3.3.1. Pair similarity loss

We can compute a loss function to constrain the difference between the encoded representation of a clean input text sequence (z^c) and the encoded representation of a perturbed input text sequence (z^p).

$$\mathcal{L}_{pair} = \left\| z^c - z^p \right\|_2^2 \tag{6}$$

A perturbed text sequence is dynamically generated from the clean text sequence as mentioned in Section 3.1.1; therefore, we would have a different perturbation for each text sequence in each epoch.

3.3.2. Triplet loss

Triplet loss is often used in retrieval tasks where it is important to measure similarity between two objects, such as facial recognition (Schroff *et al.* 2015) and answer selection for question answering (Kumar *et al.* 2019). We apply the triplet loss from (Schroff *et al.* 2015) as an auxiliary loss to ensure that an encoded representation of a clean text sequence is closer to encoded representations

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

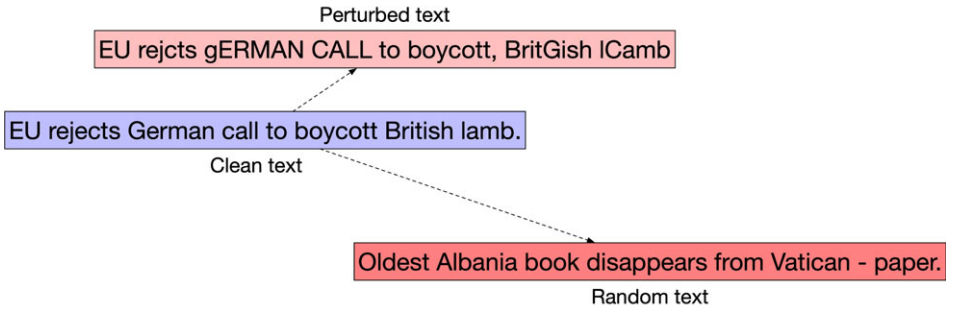

Figure 2. An example of a triplet from the CoNLL2003 corpus. With triplet loss as a part of training objective, we want the distance between a clean text sequence and its perturbed pair to be closer than the distance between a clean text sequence and a random text sequence as illustrated in this figure.

of its perturbed text than a random text sequence:

$$\mathcal{L}_{triplet} = \left[\left\| z^c - z^p \right\|_2^2 - \left\| z^c - z^r \right\|_2^2 + \alpha \right]_+ \tag{7}$$

where z^r an encoded representation of a random text sequence and α is a margin between perturbed and random pairs. This margin enforces a positive sample to be closer to the anchor (clean sample) than a random sample. Without α, the loss would be zero when a positive sample and a random sample have the same distance to the anchor or when a positive sample is only slightly closer to then anchor. We used $\alpha = 0.2$ in all of our experiments.

For each triplet, we sample a random text sequence by randomly selecting a sequence from a training set, excluding the clean text sequence. As shown in Figure 2, we provide an example of a triplet used for training one of our models; the distance between the clean text sequence and the perturbed text sequence is smaller than the distance between the clean text sequence and the random text sequence.

In summary, we experiment with four different training objectives:

(1) **baseline:** $\tilde{L} = \mathcal{L}_{clean}$
(2) **adversarial training (AT):** $\tilde{L} = \mathcal{L}_{AT}$
(3) **adversarial training + pair similarity loss (AT-P):** $\tilde{L} = \mathcal{L}_{AT} + \mathcal{L}_{pair}$
(4) **adversarial training + triplet loss (AT-T):** $\tilde{L} = \mathcal{L}_{AT} + \mathcal{L}_{triplet}$

The baseline method is an arbitrary differentiable loss function for a sequence labeling task for an unperturbed corpus. We use this as a baseline to examine whether we can gain robustness by changing our training strategy.

AT, AT-P, and AT-T methods are training objective functions that allow us to learn models that can label clean sequential inputs effectively and also learn to handle texts with typographical errors at the same time. The AT method presents perturbed text sequences to the model at training time, making it less susceptible to adversarial examples. As illustrated in Figure 3, the AT-P method is similar to the work by Liu *et al.* (2020), but instead of constraining similarity at the embeddings layer, we constrain similarity between clean and perturbed inputs at the encoder layer. This allows us to freeze pre-trained embeddings and save on computation time since the expensive forward pass on the frozen parts will only be computed once for the clean inputs. Figure 4 illustrates our proposed AT-T framework, which combines adversarial training technique with the triplet loss: this incorporates all the techniques from the first three methods and extends it by enforcing a margin between perturbed pair and a negative pair.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

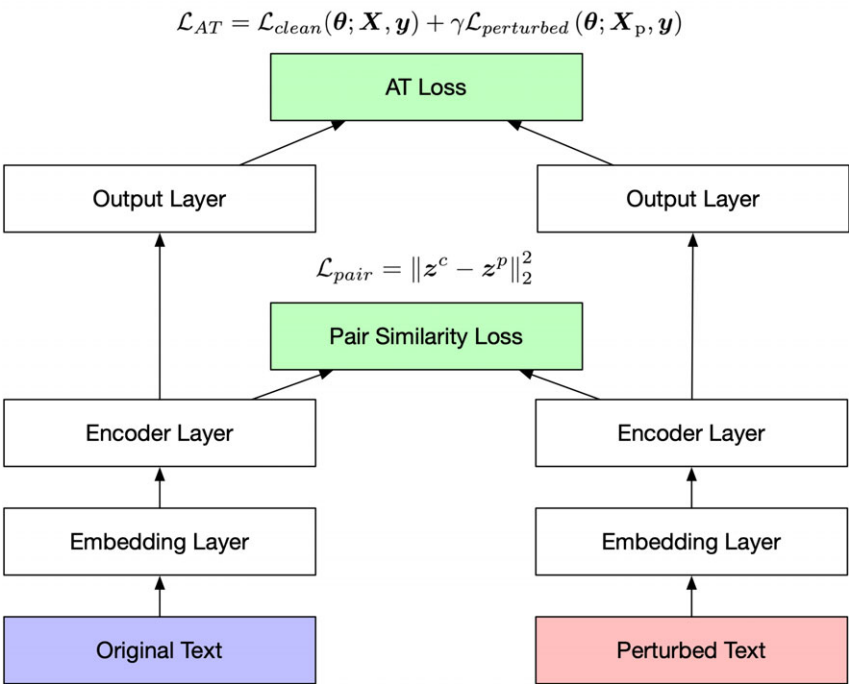

Figure 3. The AT-P framework for training robust neural networks against typographical adversarial examples. It combines adversarial training with the pair similarity loss.

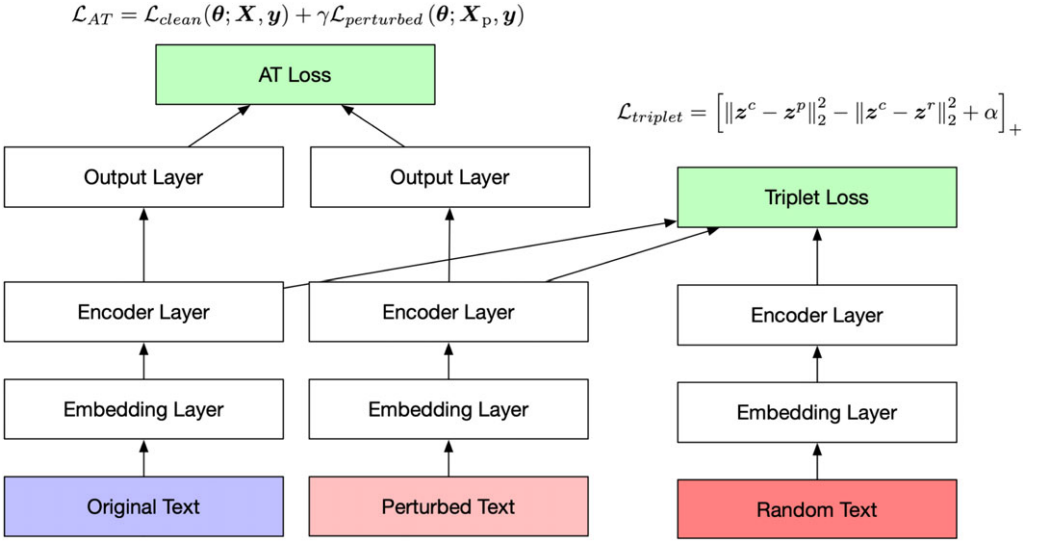

Figure 4. Our AT-T framework for training robust neural networks against typographical adversarial examples. It incorporates adversarial training with the triplet loss.

3.4. Adversarial evaluation scheme

Despite strong performances on standard evaluation schemes, current sequential tagging systems perform poorly under adversarial evaluation. To determine whether sequential tagging systems can handle inputs with spelling errors, we use an adversarial evaluation scheme that contains parallel test sets with altered input texts.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

We cannot directly evaluate the robustness of the models from the standard test sets. Therefore, it is necessary to simulate an environment where there are typographical errors. We do this by injecting typographical errors into our test sets. We generate typographical errors by using black-box and white-box methods as discussed in Sections 3.1.1 and 3.1.2. In order to demonstrate the ability to generalize to a different adversarial spelling error distribution, we exclude insertion permutation from the training data and use it only on our adversarial test sets. In addition, Jayanthi *et al.* (2020) show that insertion attack is the most effective permutation against their adversarial defense systems.

Our adversarial evaluation scheme tests whether NLP systems can tag text sequences that contain perturbed parallel text sequences without changing the original labels, while it also evaluates whether our models can maintain their performance on unperturbed texts. We propose that an evaluation scheme should evaluate models on three parallel test sets:

(1) **original test set:** We evaluate the models on an original test set without any perturbation to ensure that our models can also perform well on unperturbed texts.

(2) **black-box adversarial test set:** We alter and replace the original test set with black-box adversarial examples. A test set with black-box adversarial examples can show us the robustness of the models against typographical errors in general. A perturbation for each word is selected without access to the model's parameters.

(3) **white-box adversarial test set:** We alter and replace the original test set with gradient-based white-box adversarial examples. Instead of sampling a perturbation from a discrete uniform distribution, the white-box adversarial attack method selects an adversarial example using knowledge of the model's parameters to maximize the loss. Since there are multiple ways to generate an adversarial example for each word, the white-box adversarial attack method allows us to approximate the worst perturbation.

An example from each parallel test set can be found in Figure 1.

4. Experiments

In this section, we discuss the experimental setup, evaluation metric, and corpora used for training our sequence labeling models.

4.1. Experimental setup

Here we discuss the experimental setup details. First, we explain the setup for the main sequential tagger model used in all of our experiments. Then, we discuss spelling correction, a traditional method to combat spelling errors.

4.1.1. Main model

Our experiments rely on ELMo enhanced BiLSTM-CRF sequential tagger (Peters *et al.* 2018) as a baseline for comparison since it can exploit character-level information as well as contextual information, making it suitable for dealing with misspelled texts. In addition, El Boukkouri *et al.* (2020) suggest that ELMo's character-CNN module is more robust against spelling errors than BERT's wordpiece system.

We follow the AllenNLP (Gardner *et al.* 2017) implementation of this model. As shown in Figure 5, the ELMo enhanced BiLSTM-CRF sequential tagger's embedding layer concatenates 1024-dimensional pre-trained ELMo embeddings (512 dimensions for each direction) with

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

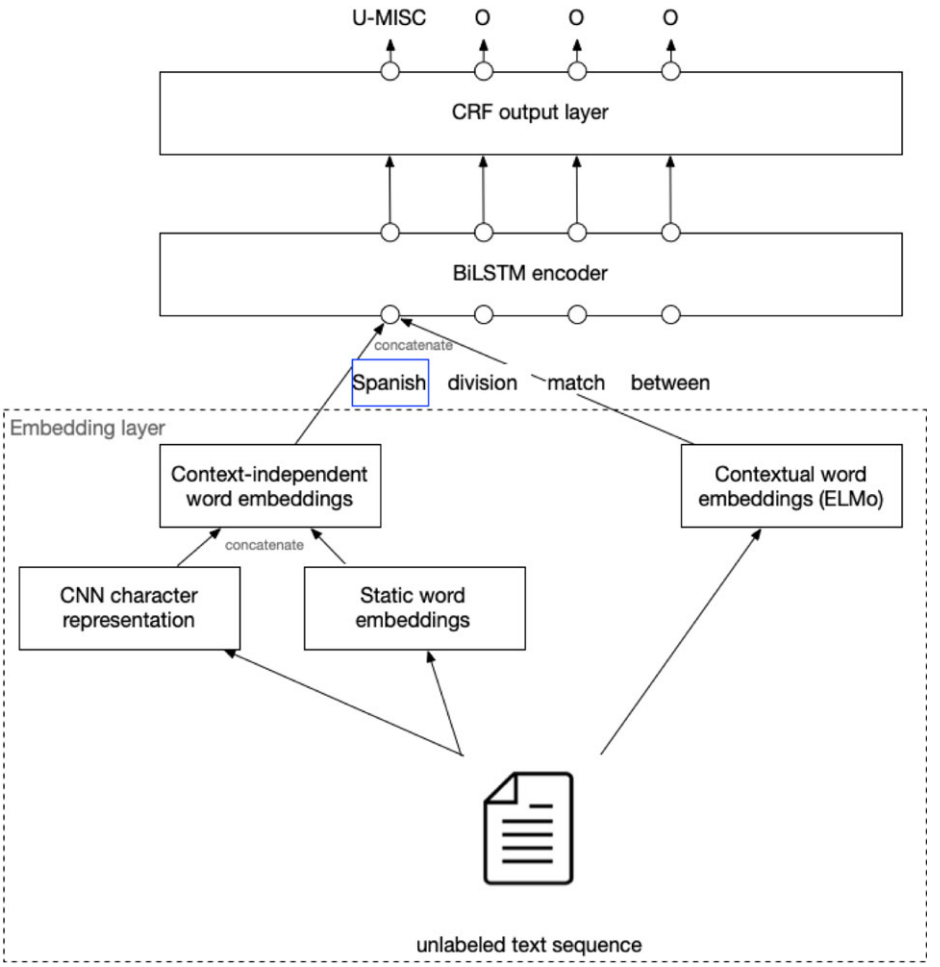

Figure 5. ELMo enhanced BiLSTM-CRF sequential tagger Peters *et al.* (2018) uses both context-independent representation and contextual representation. In this figure, the word representation for "Spanish" is the concatenation between context-independent word embeddings (CNN character representation and static word embeddings) and contextual word embeddings (ELMo).

50-dimensional static word embeddings (we use pre-trained GloVe when it is available for the language.) and 128 dimensional CNN character representation to create a token representation. The CNN character representation takes in 16-dimensional embeddings of each character and uses 128 convolutional filters of size three with ReLU activation and max-pooling. The character representation contains morphological information of each token, and the static word embeddings contains context-independent word-level information, while ELMo captures contextual information.

The token representation is forwarded to two stacked BiLSTM layers with 400 hidden units (200 hidden units for each direction) in each layer. The second BiLSTM layer's output is passed to a CRF model to predict a label for each token in the input sequence. The dropout rate between each layer is 0.5 (including dropout between the first and the second BiLSTM layer). During training, the parameters are optimized using Adam optimizer with a constant learning rate of 0.001. The gradient norms are rescaled to have a maximum value of 5.0. The pre-trained ELMo embeddings are not fine-tuned. We train all the models in this research for 40 epochs and use early stopping

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

to stop training after 10 epochs with no improvement on the clean validation set. We report the averaged score across three runs with different random seeds.

Since there is no standard pre-trained ELMo or GloVe for Thai, we used ELMo[c] trained from scratch with the word segmented data from the ORCHID (Sornlertlamvanich *et al.* 1998) training set. We trained the Thai ELMo on bidirectional language models' objective for 40 epochs. We also replaced GloVe with randomly initialized Thai word embeddings of the same dimensional size.

4.1.2. Spelling corrector

To compare our proposed methods with a traditional solution to spelling errors, we also include additional experiments where we preprocess adversarial examples with a spelling corrector at test time. A spelling corrector detects spelling mistakes and replaces them with correctly spelled words. We utilize Peter Norvig's algorithm[d] for spelling correction, considering that the off-the-shelf softwares that implemented this algorithm are widely available in multiple languages, including Thai. For English and German, we use pyspellcheker[e]. For Thai, we use PyThaiNLP[f].

4.2. Evaluation metric

We benchmark our models in three different environments as discussed in section 3.4. We evaluate the performance of our models using the F_1 metric:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})} \tag{8}$$

For sequential labeling tasks that a tag can span across multiple words, such as NER and chunking, a prediction is considered correct only if it is an exact match of a corresponding span in the corpus. A span is a unit that consists of one or more words. Part-of-Speech tags do not span over multiple words; therefore, we can calculate F_1 score at word-level.

4.3. Corpora

In this paper, we present the results obtained from seven benchmarks across four corpora: CoNLL2003 (NER, English), CoNLL2003 (chunking, English), CoNLL2003 (PoS, English), ORCHID (PoS, Thai), TüBa-D/Z (PoS, German), Twitter-41 (PoS, Thai), and Misspelled TüBa-D/Z (PoS, German).

CoNLL2003 is a language-independent named-entity recognition corpus (Tjong Kim Sang and De Meulder 2003). It contains data from two European languages: English and German. Only English data are freely available for research purposes. Therefore, we only use the English version in this study. Apart from NER annotations, CoNLL2003 also contains part-of-speech and syntactic chunk annotations.

ORCHID (**O**pen linguistic **R**esources **CH**anneled toward **I**nter**D**isciplinary research) is a Thai part-of-speech tagged corpus developed by researchers from National Electronics and Computer Technology Center (NECTEC) in Thailand and Communications Research Laboratory (CRL) in Japan (Sornlertlamvanich *et al.* 1998). ORCHID contains texts from technical papers that appeared in the proceedings of NECTEC annual conferences. The ORCHID corpus is freely

[c]We used the TensorFlow implementation of the pre-trained biLM for ELMo by AllenNLP (https://github.com/allenai/bilm-tf). We also provide the hyperparameters for training the Thai ELMo model and the model's weights (https://github.com/c4n/elmo_th).

[d]https://norvig.com/spell-correct.html

[e]https://pypi.org/project/pyspellchecker/

[f]https://pythainlp.github.io/

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 2. Data statistics of the English CoNLL2003, the Thai ORCHID, and the German TüBa-D/Z corpora (training set/validation set/test set). † - For English and German, an input sequence is a sentence. For Thai, an input sequence is a paragraph

	CoNLL2003	ORCHID	TüBa-D/Z
Language	English	Thai	German
Task	PoS/NER/Chunking	PoS	PoS
# of classes	44/4/9	47	54
# of input sequences†	14,987/3,466/3,684	925/103/115	15,000/5,000/5,000
# of words	203,621/51,263/46,425	274,006/25,401/43,226	274,729/91,843/93,409
# of characters	890,042/224,197/200,112	1,187,721/107,512/190,394	1,438,304/480,381/489,490
avg. char/word	4.4/4.4/4.3	4.3/4.2/4.4	5.2/5.2/5.2

available. ORCHID segments each text into paragraphs and sentences. Since the Thai sentence segmentation standard is still under development, we use each paragraph as an input sequence instead. Also, Thai has no explicit word boundary; therefore, we use gold segmentation provided by the Thai ORCHID corpus to define our word inputs. We split 10% of the ORCHID corpus for testing. Then, we split the remaining corpus into a training set (90%) and a validation set (10%).

TüBa-D/Z (Telljohann *et al.* 2005) is a German treebank developed by the Division of Computational Linguistics at the Department of Linguistics of the Eberhard Karl University of TÜbingenen, Germany. It contains POS tags as well as annotation for misspelled words. We use TüBa-D/Z version 11. TüBa-D/Z is a large dataset. There are 104,787 sentences and 1,959,474 tokens in the corpus, this is almost 10 times the amount of words in the CoNLL2003 training set, which might not be feasible for our experiments. Therefore, we sampled some of the sentences in the corpus for the experiments.

Table 2 shows the detailed statistics of the three corpora. For the CoNLL2003 corpus, the number of classes does not include the class "O" for non-named entities or non-syntactic chunks.

For each corpus, we use a separated set of characters for creating insertion and substitution errors. For the English CoNLL2003 and the German TüBa-D/Z corpora, we use all alphabets, both lowercase, and uppercase. For the Thai ORCHID, we use all the Thai consonants, the vowels, and the tonal marks.

In addition to the standard corpora, we extend our experiments by benchmarking on real datasets with natural misspelling errors. Table 3 shows the data statistics of the following corpora.

Twitter-41[g] is a small Thai twitter dataset with Universal POS tags[h] that we collected to benchmark the ability of sequential taggers to handle out-of-domain and misspelled data. Each text sample contains at least one spelling error. We use ORCHID word segmentation standard for this dataset. All ORCHID POS tags can be converted to Universal POS tags; this allows us to test sequential taggers trained on ORCHID data.

Misspelled TüBa-D/Z (Telljohann *et al.* 2005) includes all sentences with annotated spelling errors in the TüBa-D/Z corpus. We separated these misspelled sentences from the TüBa-D/Z corpus for benchmarking purpose. Hence, they do not appear in training, validation, and TüBa-D/Z test set. The Misspelled TüBa-D/Z test set allows us to assess a sequential tagger's performance on in-distribution data with a small amount of spelling errors. There are 4706 misspelled words out of 115,801 words.

[g]https://github.com/c4n/thai-political-tweets
[h]https://universaldependencies.org/u/pos/ Petrov *et al.* (2012)

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 3. Data statistics of datasets with natural misspelling errors: Twitter-41 and Misspelled TüBa-D/Z

	Twitter-41	Misspelled TüBa-D/Z
Language	Thai	German
Task	PoS	PoS
# of classes	16	54
# of input sequences	41	4,319
# of words	965	115,801
# of characters	3,472	620,764
avg. char/word	3.6	5.4

5. Experimental results

We performed experiments over three sequence labeling tasks across three languages to better understand the relative importance of our training objectives. We investigated the robustness of sequential taggers through a series of black-box and white-box attacks. We only use insertion perturbation to generate adversarial examples for evaluation. And we use transposition, deletion, substitution, and capitalization (for English and German) to generate adversarial examples for adversarial training. This allows us to test whether our adversarial training methods can generalize to different spelling error distributions. In addition, to test the ability to generalize on real-world data, we also benchmarked our methods on real datasets that contains natural spelling errors.

Table 4 illustrates the effectiveness of our adversarial attack methods. The baseline method performs much worse on white-box adversarial examples than black-box adversarial examples, indicating that white-box attacks can generate stronger adversarial examples. This suggests that we can select a misspelling variant that is more damaging than the others. It is important to point out that the NER task is more sensitive to typographical errors than the chunking and PoS tasks. For the CoNLL2003 corpus, the differences between F_1 scores on black-box and white-box test sets are much higher for the NER task. The degradation of the F_1 scores—when evaluated on noisy test sets—seems to be less severe for the chunking and PoS tasks. For the Thai ORCHID and the German TüBa-D/Z corpora, the gap between F_1 scores on the black-box and the white-box test sets is much closer than the CoNLL2003 corpus.

Comparing to the baseline, AT, AT-P, and AT-T can improve the robustness against misspelled text without degrading its performance on the original texts. AT, AT-P, and AT-T can all maintain competitive results on the original test sets compared to the baseline method. The AT-T method outperforms all other methods on both black-box and white-box test sets for the chunking task.

Furthermore, we also provide average cosine similarity scores between clean-perturbed pairs (z^c, z^p) and clean-random pairs (z^c, z^r) as shown in Table 5. This reveals the effectiveness of our similarity constraint techniques: pair similarity loss and triplet loss. Ideally, we want the cosine similarity score of a clean-perturbed pair to be high, and the cosine similarity score of a clean-random pair to be low.

5.1. Effects of adversarial training

We examined the performance of the models trained by the AT method compared to the baseline method. Table 4 indicates that adversarial training is a very effective method for improving the robustness of sequential labeling models against typographical adversarial examples across all

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 4. F_1 scores and associated standard deviations on the English CONLL2003, the Thai ORCHID, and the German TüBa-D/Z corpora. The black-box and white-box adversarial test sets only contain insertion perturbation. We use the rest of the perturbation types for adversarial training. We averaged the scores across three runs with different random seeds and calculated their standard deviations. A bolded number refers to the best F-1 score for each task in each test set. An underlined number denotes statistical significance against all other models for all random seeds using McNemar's Test (p-value $<= 0.05$)

	Original	Black-box	White-box	w/spelling correction Black-box	w/spelling correction White-box
			CoNLL2003 (English, NER)		
Baseline	91.72 ± 0.24	52.23 ± 4.07	39.01 ± 1.92	86.35 ± 0.57	83.95 ± 0.20
AT	**92.05 ± 0.04**	**80.17 ± 0.38**	**74.61 ± 0.99**	**88.21 ± 0.16**	**88.39 ± 0.99**
AT-P	91.85 ± 0.18	79.32 ± 1.09	72.32 ± 2.69	87.87 ± 0.07	88.27 ± 0.20
AT-T	91.75 ± 0.18	79.75 ± 0.40	74.20 ± 1.06	**88.21 ± 0.16**	88.30 ± 0.39
			CoNLL2003 (English, Chunking)		
Baseline	**91.97 ± 0.08**	73.14 ± 0.87	66.04 ± 0.97	87.34 ± 0.07	86.47 ± 0.10
AT	91.83 ± 0.13	83.82 ± 0.74	78.94 ± 1.41	87.88 ± 0.71	87.02 ± 0.12
AT-P	91.82 ± 0.15	84.27 ± 0.16	80.05 ± 0.36	87.85 ± 0.09	86.99 ± 0.16
AT-T	91.75 ± 0.08	**84.55 ± 0.62**	**80.57 ± 1.18**	**88.04 ± 0.14**	**87.26 ± 0.15**
			CoNLL2003 (English, PoS)		
Baseline	**95.83 ± 0.05**	83.11 ± 0.47	78.04 ± 0.74	92.95 ± 0.13	93.29 ± 0.22
AT	95.64 ± 0.01	**91.21 ± 0.01**	**89.14 ± 0.03**	**93.14 ± 0.01**	**94.06 ± 0.18**
AT-P	95.60 ± 0.04	91.19 ± 0.16	88.61 ± 0.28	93.11 ± 0.07	93.47 ± 0.02
AT-T	95.57 ± 0.05	91.06 ± 0.07	88.55 ± 0.05	**93.14 ± 0.03**	93.61 ± 0.16
			ORCHID (Thai, PoS)		
Baseline	93.98 ± 0.05	78.10 ± 0.37	75.05 ± 0.63	93.42 ± 0.49	92.87 ± 0.18
AT	**94.09 ± 0.10**	**90.04 ± 0.18**	**86.66 ± 0.62**	94.24 ± 0.05	**94.33 ± 0.29**
AT-P	93.88 ± 0.05	88.67 ± 0.75	84.19 ± 0.66	94.21 ± 0.13	92.70 ± 0.06
AT-T	94.02 ± 0.08	89.77 ± 0.44	85.33 ± 0.50	**94.36 ± 0.04**	93.17 ± 0.09
			TüBa-D/Z (German, PoS)		
Baseline	**99.07 ± 0.02**	75.85 ± 0.01	71.59 ± 0.83	96.12 ± 0.04	94.99 ± 0.18
AT	99.05 ± 0.02	**92.15 ± 0.02**	**88.43 ± 0.59**	96.76 ± 0.04	**95.90 ± 0.08**
AT-P	99.04 ± 0.01	91.79 ± 0.01	87.36 ± 0.60	**96.83 ± 0.03**	95.77 ± 0.08
AT-T	99.03 ± 0.01	91.89 ± 0.01	88.18 ± 1.00	96.73 ± 0.03	95.66 ± 0.10

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 5. Average cosine similarity scores between clean-perturbed pairs and clean-random pairs. We averaged the cosine similarity scores and their standard deviations across three runs with different random seeds. We want the similarity between each clean-random pair to be high and the similarity between each clean-random pair to be low. † - This shows a difference between the average cosine similarity scores of clean-perturbed pairs and clean-random pairs for each model

	Clean-perturbed pair	Clean-random pair	Δ avg sim. †
	CoNLL2003 (English,NER)		
Baseline	0.7704 ± 0.0140	0.5874 ± 0.0275	0.1830
AT	0.7501 ± 0.0043	0.5424 ± 0.0056	0.2077
AT-P	$\mathbf{0.8520 \pm 0.0039}$	0.8134 ± 0.0109	0.0385
AT-T	0.7103 ± 0.0079	$\mathbf{0.2877 \pm 0.0216}$	**0.4226**
	CoNLL2003 (English, Chunking)		
Baseline	0.7745 ± 0.0015	0.6721 ± 0.0026	0.1024
AT	0.7674 ± 0.0142	0.6575 ± 0.0209	0.1099
AT-P	$\mathbf{0.8465 \pm 0.0002}$	0.8088 ± 0.0003	0.0377
AT-T	0.7013 ± 0.0146	$\mathbf{0.3738 \pm 0.0347}$	**0.3275**
	CoNLL2003 (English, PoS)		
Baseline	0.6597 ± 0.0042	0.3770 ± 0.0043	0.2827
AT	0.6594 ± 0.0060	0.3635 ± 0.0118	0.2959
AT-P	$\mathbf{0.8202 \pm 0.0041}$	0.6922 ± 0.0164	0.1280
AT-T	0.6483 ± 0.0018	$\mathbf{0.2645 \pm 0.0074}$	**0.3837**
	ORCHID (Thai, PoS)		
Baseline	0.7173 ± 0.0068	0.3800 ± 0.0164	0.3373
AT	0.7218 ± 0.0059	0.3782 ± 0.0128	0.3436
AT-P	$\mathbf{0.7934 \pm 0.0019}$	0.5317 ± 0.0079	0.2618
AT-T	0.7437 ± 0.0060	$\mathbf{0.3643 \pm 0.0111}$	**0.3795**
	TüBa-D/Z (German, PoS)		
Baseline	0.6582 ± 0.0177	0.5454 ± 0.0212	0.1128
AT	0.6969 ± 0.0100	0.5480 ± 0.0142	0.1489
AT-P	$\mathbf{0.8619 \pm 0.0020}$	0.8022 ± 0.0041	0.0597
AT-T	0.6997 ± 0.0187	$\mathbf{0.4440 \pm 0.0347}$	**0.2557**

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

tasks on both black-box and white-box test sets. The adversarial training method also maintains competitive results on the original test sets.

For the English CoNLL2003 corpus, adversarial training obtains the best result on all test sets for the PoS tagging and NER tasks. For the Thai ORCHID corpus, the AT method obtained the best scores on all test sets. For the German TüBa-D/Z corpus, adversarial training obtains the best results on the black-box test set and the white-box test set. The differences between the F_1 scores of black-box and white-box test sets are lower for the PoS tagging task on the ORCHID corpus and the TüBa-D/Z corpus than the CoNLL2003 corpus.

Table 5 shows that adversarial training slightly improves the similarity between the encoded sequence representations of clean-perturbed pairs.

5.2. Effects of pair similarity loss

We investigated the performance of the models trained by the AT-P method compared to the AT method. The AT-P method improves the performance by constraining the similarity between original texts and perturbed texts. Table 4 shows an improvement on black-box and white-box test sets only on the chunking task. The AT-P method can also maintain competitive results on the original test sets. The results suggest that, for the chunking task, we can improve the robustness against typographical errors by constraining the similarity between original and misspelled text sequences.

Table 5 shows that the pair similarity loss improves the similarity between the encoded sequence representations of clean-perturbed pairs substantially. However, it also increases the similarity between clean-random pairs for which we want to keep low.

5.3. Effects of triplet loss

We observed the performance of the models trained by the AT-T method compared to the AT and AT-P methods. Triplet loss minimizes the distance between clean text sequence and its parallel perturbed text sequence and maximizes the distance between clean text sequence and a random perturbed text sequence. Table 4 indicates that the AT-T method outperforms the AT and AT-P methods on both black-box and white-box test sets on the chunking task. The AT-T method maintains competitive results on all original test sets and achieves better scores comparing to the AT-P method on all adversarial testsets except for the ConLL2003 PoS task. However, comparing to the AT method, the AT-T method does not improve the PoS tagging and NER performances.

Regarding the fact that both AT-P and AT-T methods did not provide a clear improvement over the AT method, we can conclude that our sequence-level similarity constraints can improve the performances over adversarial test sets only for the chunking task. It does not improve the robustness over adversarial examples for the PoS tagging and the NER tasks. Note that a span can be a large portion of the sequence; therefore, chunking can benefit more from sequence-level similarity constraints.

Table 5 shows that triplet loss lowers the similarity between clean-random pairs. However, it does not improve the similarity between the encoded sequence representations of clean-perturbed pairs. Nevertheless, the gaps between the similarity scores between clean-perturbed pairs and clean-random pairs are larger with triplet loss.

5.4. Effects of spelling correction

We found that using a spelling corrector at test time can tremendously improve the robustness over adversarial test sets. It is more effective than any of the adversarial training meth.ds we proposed. Nevertheless, all of our methods (AT, AT-P, AT-T) can boost the robustness for all tasks when combined with the spelling corrector. However, on the real misspelled datasets,

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 6. F_1 scores and theirs standard deviations on the Thai Twitter-41 and the German TüBa-D/Z (misspelled) corpora. We averaged the scores across three runs with different random seeds and calculated their standard deviations. A bolded number refers to the best F-1 score for each task in each test set. An underlined number denotes statistical significance against all other models for all random seeds using McNemar's Test (p-value $<= 0.05$)

			w/spelling correction	w/spelling correction
	Twitter-41	Misspelled TüBa-D/Z	Twitter-41	Misspelled TüBa-D/Z
Baseline	68.84 ± 0.87	98.18 ± 0.01	68.84 ± 0.85	97.76 ± 0.04
AT	$\mathbf{69.33 \pm 1.22}$	98.17 ± 0.02	69.12 ± 1.25	$\underline{\mathbf{97.85 \pm 0.04}}$
AT-P	68.81 ± 0.66	$\mathbf{98.21 \pm 0.01}$	68.74 ± 0.59	97.69 ± 0.03
AT-T	69.08 ± 0.26	98.15 ± 0.01	$\mathbf{69.15 \pm 0.18}$	97.68 ± 0.03

the spelling corrector does not improve the performance and can hinder the performance of sequential taggers.

5.5. Real datasets

To test the ability to improve the robustness in real data, we benchmarked our methods on two datasets. Since there is no need to exclude a perturbation type and keep it for evaluation, we use all the perturbation types to train the AT, AT-P, and AT-T methods to benchmark them on the real datasets.

Twitter-41 is a Thai social media dataset where each tweet contains at least one spelling error. Misspelled TüBa-D/Z contains all misspelled samples in the TüBa-D/Z corpus. Twitter-41 illustrates the models' performance on misspelled out-of-domain data, while the misspelled TüBa-D/Z corpus reveals the performance on misspelled in-domain data.

Table 6 shows the results on the real datasets. Since the result of each model is similar to each other, we confirm the result using McNemar's test. Each model was trained on three different random seeds. For each model pair that we want to compare, we run nine McNemar's tests to confirm the result.

On the Twitter-41 dataset, we found that the AT and the AT-T methods' improvements over the baseline model are not significant. This suggests that our methods do not generalize to out-of-domain data with natural spelling errors.

Our methods do not improve the performance on the misspelled TüBa-D/Z, since the models' performances on the misspelled TüBa-D/Z are already over 98 F1-score, leaving little room for improvement. The spelling corrector does not improve the performance on both corpora and could hinder the performance of the sequential taggers. Although the results on Misspelled TüBa-D/Z with spelling correction have a poorer performance, the AT method can boost the performance on spelling corrected data with a statistically significant result for all nine McNemar's tests against the baseline model.

The results on real datasets show that there is still a gap in improving the robustness for real world examples.

6. Error analysis

In this section, we examine which classes are harder to predict when there are misspelling errors. Studying errors on sequential labeling tasks has an advantage because we can analyze word-level errors by comparing them to their ground truths and whether they are affected by typographical errors. Therefore, we can study the impact of misspelling errors in each test set. For each task, we select top 5 categories with highest number of perturbations. We analyze white-box adversarial

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

test sets since they simulate a worst-case scenario for each model. For each dataset, first, we discuss the impact of white-box adversarial examples on the baseline model. Then, we discuss how adversarial training methods can improve the robustness over these adversarial examples.

Table 7 shows the error rates of top five most frequent classes with adversarial examples, and the numbers of perturbed words for the top five classes are shown in Table 8. We compare the error rates between original and white-box test sets. The baseline model is trained on unperturbed texts only. As shown in Table 7, the baseline model is most sensitive to white-box adversarial examples. However, there are classes where adversarial attack is more damaging.

CoNLL2003 (NER): Interestingly, the baseline is more sensitive to adversarial examples for the location (LOC) class than other classes. However, it has less error rate for LOC comparing to organization (ORG) and miscellaneous (MISC). The baseline model is also sensitive to person (PER) adversarial examples. Excluding the non-entity class (O), PER has the lowest error rate on the original test set. However, its error rate exceeds the ORG class which performs much poor in the original test set. The adversarial training methods (AT, AT-P, AT-T) improve the error rate for adversarial examples substantially, especially for the LOC class. MISC adversarial examples have the highest error rate for all adversarial training methods. On the other hand, the non-entity class (O) has a very low error rate for adversarial examples. This suggests that our methods can segregate named-entity classes from non-entity class.

CoNLL2003 (Chunking): For the white-box adversarial evaluation, the baseline model has the highest error rates for all classes except for noun phrase (NP), where it has the lowest error rate (2.63%) comparing to other methods. Moreover, there is a class imbalance in the data, 60.9% of all words in the training data are annotated with an NP tag. This suggests that there is a bias towards NP. In addition, NP also has a larger range of vocabulary comparing to other classes. This makes it less likely to overfit.

On the original dataset, the baseline model has a lower error rate (4.56%) for prepositional phrase (PP) than verb phrase (VP), adverb phrase (ADVP), and adjective phrase (ADJP). But despite that, for adversarial examples, PP has the highest error rate for the baseline model (98.49%). The following prediction examples of the baseline model, on the original text (1) and the perturbed text (2), show two errors caused by adversarial examples:

(1) Despite winning the Asian Games title two years ago
PP VP NP NP NP NP NP NP ADVP

(2) ∗ Dhespite wQinning the AsZian Glames tiZtle two yeaprs ago
NP VP NP NP NP VP NP NP ADVP

Since vocabulary for preposition is more limited than other classes, it is easier to fool the model by injecting spelling errors. Our adversarial methods help reduce the error rates on adversarial examples, the AT-T method alleviates the error rate of PP from 98.49% to 51.25%. However, there is still a big gap for improvement.

CoNLL2003 (POS): For adversarial examples, the baseline model has the worst performance on the adjective class (JJ) and gives a poorer performance to the singular noun class (NN) compared to the plural noun class (NNS). The plural noun class often contains a morphological clue (the letter "s" at the end of the word), since we are evaluating only on insertion perturbation, the adversarial examples can still contain the clue for NNS. Despite the fact that the cardinal number class (CD) has the lowest error rate on the original dataset for all methods, it is not the class with the lowest error rate on white-box adversarial evaluation.

(1) Fourteen years after he bludgeoned
CD NNS IN PRP VBD

(2) ∗ Fjourteen yeaers axfter he bludgeConed
NNP NNS RB PRP VBD

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 7. Error rates on perturbed words: We calculate the error rates at word-level only. We excluded non-perturbed words from the calculation. Only top five classes with the highest number of perturbed words are shown in this table.

	Original					White-box				
CoNLL2003 (NER)	O	PER	ORG	LOC	MISC	O	PER	ORG	LOC	MISC
baseline	1.32	3.23	**8.28**	7.94	18.45	20.14	44.69	36.64	73.87	61.26
AT	**1.22**	**3.20**	9.53	6.38	**18.20**	**2.44**	**16.24**	30.39	33.69	**39.80**
AT-P	1.24	3.44	9.50	**6.20**	19.08	4.23	21.48	29.45	**29.40**	44.39
AT-T	1.24	3.58	10.29	6.97	18.71	3.40	17.75	**29.42**	32.58	40.60
CoNLL2003 (CHUNK)	NP	VP	PP	ADVP	ADJP	NP	VP	PP	ADVP	ADJP
Baseline	**1.47**	7.64	4.56	24.76	34.59	**2.63**	67.76	98.49	91.56	86.48
AT	1.58	**6.68**	**3.64**	24.01	35.05	3.52	31.10	54.05	75.65	72.74
AT-P	1.84	7.21	4.53	**23.25**	**32.42**	4.43	**29.50**	57.17	**66.26**	69.76
AT-T	1.60	7.95	4.60	27.78	36.43	3.85	31.27	**51.25**	69.27	**69.64**
CoNLL2003 (POS)	NNP	NN	JJ	NNS	CD	NNP	NN	JJ	NNS	CD
Baseline	4.24	8.32	16.55	**6.35**	2.64	9.40	51.16	66.52	32.99	23.59
AT	**4.08**	**8.19**	**15.63**	6.41	**2.02**	**7.22**	**26.30**	**42.05**	**15.49**	13.45
AT-P	5.68	8.97	15.81	6.80	2.06	11.26	30.84	45.65	18.15	13.66
AT-T	5.25	8.83	17.50	6.50	2.08	10.48	30.04	48.85	18.98	**12.78**
ORCHID (POS)	NCMN	VACT	VSTA	JSBR	NPRP	NCMN	VACT	VSTA	JSBR	NPRP
baseline	**3.55**	5.71	15.91	17.13	22.98	17.55	61.78	86.21	83.20	44.58
AT	4.42	**5.48**	**15.69**	**16.63**	**15.83**	**11.52**	**29.95**	**42.20**	**39.63**	**27.76**
AT-P	5.38	5.85	18.91	20.41	18.54	15.27	32.36	55.04	50.56	33.21
AT-T	4.77	6.52	18.95	18.45	22.66	13.65	34.09	49.81	45.32	38.22
TüBa-D/Z (POS)	NN	ADJA	NE	ADV	VVFIN	NN	ADJA	NE	ADV	VVFIN
Baseline	0.77	**0.86**	4.05	2.33	0.93	24.38	40.69	28.79	83.02	38.99
AT	**0.69**	**0.86**	**2.33**	4.64	**0.69**	**7.54**	**10.49**	43.31	**26.08**	**14.07**
AT-P	0.89	0.96	4.27	**2.74**	1.08	9.83	14.17	**27.56**	48.07	18.41
AT-T	0.72	1.09	2.46	5.42	0.90	**7.54**	14.36	45.64	30.89	15.77

The above example shows the baseline model's prediction error of the CD class. Considering that the vocabulary for cardinal number in word form is limited, it is more likely to overfit and therefore easier to fool with adversarial perturbation. The three adversarial training methods can

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 8. Frequency of unperturbed/perturbed words in each of the top five classes with the highest number of perturbed words

CoNLL2003 (NER)	O	PER	ORG	LOC	MISC	Other classes
	Non-entity	Person	Organization	Location	Miscellaneous	
	20,761/17,501	219/2,515	375/2,088	143/1,865	142/826	0/0
CoNLL2003 (CHUNK)	**NP**	**VP**	**PP**	**ADVP**	**ADJP**	
	Noun	Verb	Prepositional	Adverb	Adjective	
	10,274/18,888	1,764/3916	3,088/906	106/486	40/291	6,368/308
CoNLL2003 (POS)	**NNP**	**NN**	**JJ**	**NNS**	**CD**	
	Proper noun	Noun (singular)	Adjective	Noun (plural)	Cardinal number	
	1,136/7,625	313/4,528	160/2,212	56/2,199	4,321/1,625	15,654/6,606
ORCHID (POS)	**NCMN**	**VACT**	**VSTA**	**JSBR**	**NPRP**	
	Common noun	Active verb	Stative verb	Subordinating conjunction	Proper noun	
	1,547/9047	1,231/3268	796/1,400	646/918	42/757	18,763/4,811
TüBa-D/Z (POS)	**NN**	**ADJA**	**NE**	**ADV**	**VVFIN**	
	Noun	Attributive adjective	Proper noun	Adverb	Finite main verb	
	439/17,310	140/5,038	687/4,362	861/4258	98/3,940	38,697/17,579

give a large improvement for the CD class. However, the AT-P and the AT-T methods yield a poorer performance on the proper noun (NNP) class.

ORCHID (POS): Comparing the performance of the baseline model between the original test set and the white-box adversarial example, we found that the class with the largest surge in error rate (15.91% → 86.21%) is the stative verb (VSTA). A stative verb refers to a state such as "to know," and "to be." Moreover, other classes with limited vocabulary such as active verb (VACT) and subordinating conjunction (JSBR) are less robust to adversarial examples compared to classes with larger set of vocabulary, such as common noun (NCMN) and proper noun (NPRP). The AT method consistently gives the lowest error rate on adversarial examples, closing the gap between the limited vocabulary and large vocabulary classes. Yet, there is still a lot of room for improvement.

TüBa-D/Z (POS): The baseline model's result shows that classes with more vocabulary, such as noun (NN) and proper noun (NE), are more robust against the adversarial examples. This result agrees with the results from the previously discussed datasets. Although the three adversarial training methods can improve an overall performance on adversarial examples, they can be detrimental to the performance on the NE class. This agrees with the result from the CoNLL2003 (POS) dataset. Note that for German, all nouns are capitalized. Therefore augmenting capitalization errors to the training data can reduce the benefit from this orthographic clue.

The error analysis confirms that the three adversarial training methods (AT, AT-P, and AT-T) can improve an overall robustness against adversarial examples. The analysis shows

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

that adversarial examples cause more errors for classes with limited vocabulary. The gap of performances between classes show us opportunities for future improvement.

7. Conclusion

In this paper, we study an adversarial training framework for sequential labeling model to improve the robustness against adversarial typographical examples. We also propose an adversarial evaluation scheme to conduct experiments on both original and adversarial test sets.

We have shown that our adversarial training framework can withstand typographical adversarial examples while maintaining high F_1 scores on the original test sets. By incorporating adversarial training method and triplet loss, the novel AT-T method further improves F_1 scores on both black-box and white-box adversarial test sets on the chunking task.

Adversarial training can strengthen the robustness of sequential taggers against spelling errors. Moreover, it increases the similarity between the encoded sequence representations of clean-perturbed pairs. Using similarity constraints, such as pair loss and triplet loss, at the sentence level improves the performance for the chunking task. For PoS tagging and NER, no substantial improvement is observed. Pair loss also increases the similarity between clean-perturb sequence pairs substantially. However, it also increases similarity between clean-random pairs, and the differences between the average cosine similarity scores of clean-perturbed pairs and clean-random pairs become lower as well. Triplet loss widens these differences.

Using a spelling correction program can tremendously increase the robustness against typographical adversarial examples. Combining the spelling corrector with our methods (AT, AT-P, and AT-T) can further improve the robustness of sequential taggers. However, on the real datasets, the spelling corrector does not improve the performance of the sequential taggers and can even hinder their performance.

Our adversarial evaluation scheme reveals the weaknesses of deep learning models on typographical errors. It approximates the worst-case scenario with white-box attacks and approximates a general performance on misspelled texts with black-box attacks. The adversarial evaluation scheme also includes the performance scores on the unperturbed test sets to ensure that we can enhance the performance over adversarial examples without degrading our performance in a standard setting. The experimental results have shown that typographical adversarial attacks can cause deterioration in the performance of sequential taggers. White-box adversarial examples can deteriorate the performance of the taggers much stronger than the black-box adversarial examples. This suggests that there is a perturbation variant that is much more effective than the others. This adversarial evaluation scheme reveals a weakness of sequential taggers, which can be exploited to cause harm, such as bypassing a de-identification system.

This paper only addresses typographical adversarial examples. The performance gap between natural misspelling examples and synthetic adversarial examples is still large. Hence, there is a research opportunity to close this gap by introducing more constraints to the adversarial example generation process to reflect natural spelling errors. In addition, there are many linguistics capabilities that we should consider to improve the model's robustness. Nevertheless, adversarial training and similarity constraint through metric learning are promising research directions for improving the overall robustness of current sequential labeling systems.

As for future work, we intend to further research into deep metric learning and spelling correction techniques, since a combination of these two techniques is effective against adversarial examples. Negative sampling for deep metric learning is one area we aim to explore to further enhance the triplet loss. In addition, we want to explore deeper into sequential tagging applications with a real societal impact, such as a clinical de-identification system to prevent harm

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

caused by adversarial examples. Furthermore, we intend to incorporate linguistic knowledge to create adversarial examples that reflect real language usage.

Acknowledgments. This work was supported in part by the Thailand Graduate Institute of Science and Technology (TGIST:SCA-CO-2561-7116TH), National Science and Technology Development Agency (NSTDA). We would like to thank Dr Sarana Nutanong for providing computational resources, and Eberhard Karls University of TÜbingen for granting us access to the TüBa-D/Z treebank. Furthermore, we thank Pattarawat Chormai for his feedback on an earlier draft of this work. Lastly, we would like to express our gratitude to the editor and anonymous reviewers for their comprehensive and insightful comments, which greatly aided us in improving this work.

References

Akbik A., **Bergmann T.** and **Vollgraf R.** (2019). 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)*, pp. 724–728.

Akbik A., **Blythe D.** and **Vollgraf R.** (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Belinkov Y. and **Bisk Y.** (2018). Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Bilmes L. (1995). The grammaticalization of thai'come'and'go'. In Annual Meeting of the Berkeley Linguistics Society, volume 21, pp. 33–46.

Bodapati S., **Yun H.** and **Al-Onaizan Y.** (2019). Robustness to capitalization errors in named entity recognition. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, Hong Kong, China. Association for Computational Linguistics, pp. 237–242.

Bojanowski P., **Grave E.**, **Joulin A.** and **Mikolov T.** (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **5**, 135–146.

Devlin J., **Chang M.-W.**, **Lee K.** and **Toutanova K.** (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Ebrahimi J., **Rao A.**, **Lowd D.** and **Dou D.** (2018). 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)*, pp. 31–36.

El Boukkouri H., **Ferret O.**, **Lavergne T.**, **Noji H.**, **Zweigenbaum P.** and **Tsujii J.** (2020). CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online). International Committee on Computational Linguistics, pp. 6903–6915.

Gao J., **Lanchantin J.**, **Soffa M.L.** and **Qi Y.** (2018). Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops (SPW), pp. 50–56.

Gardner M., **Artzi Y.**, **Basmova V.**, **Berant J.**, **Bogin B.**, **Chen S.**, **Dasigi P.**, **Dua D.**, **Elazar Y.**, **Gottumukkala A.**, et al. (2020). Evaluating nlp models via contrast sets. arXiv preprint arXiv:2004.02709.

Gardner M., **Grus J.**, **Neumann M.**, **Tafjord O.**, **Dasigi P.**, **Liu N.F.**, **Peters M.**, **Schmitz M.** and **Zettlemoyer L.S.** (2017). Allennlp: A deep semantic natural language processing platform.

Goodfellow I., **Shlens J.** and **Szegedy C.** (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Heigold G., **Varanasi S.**, **Neumann G.** and **van Genabith J.** (2018). How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume* 1: *Research Papers)*, Boston, MA. Association for Machine Translation in the Americas, pp. 68–80.

Heinzerling B. and **Strube M.** (2019). Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 273–291.

Huang Z., **Xu W.** and **Yu K.** (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Jayanthi S.M., **Pruthi D.** and **Neubig G.** (2020). NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 158–164, Online. Association for Computational Linguistics.

Jia R. and **Liang P.** (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics, pp. 2021–2031.

Jiang Y., **Hu C.**, **Xiao T.**, **Zhang C.** and **Zhu J.** (2019). Improved differentiable architecture search for language modeling and named entity recognition. 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, pp. 3585–3590.

Karpukhin V., **Levy O.**, **Eisenstein J.** and **Ghazvininejad M.** (2019). Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, Hong Kong, China. Association for Computational Linguistics, pp. 42–47.

Kumar S., **Garg S.**, **Mehta K.** and **Rasiwasia N.** (2019). Improving answer selection and answer triggering using hard negatives. 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, pp. 5911–5917.

Lafferty J.D., **McCallum A.** and **Pereira F.C.N.** (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, *ICML '01*, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc, 282–289.

Lample G., **Ballesteros M.**, **Subramanian S.**, **Kawakami K.** and **Dyer C.** (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California. Association for Computational Linguistics, pp. 260–270.

Ling W., **Dyer C.**, **Black A.W.**, **Trancoso I.**, **Fermandez R.**, **Amir S.**, **Marujo L.** and **Lus T.** (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal. Association for Computational Linguistics, pp. 1520–1530.

Liu H., **Zhang Y.**, **Wang Y.**, **Lin Z.** and **Chen Y.** (2020). Joint character-level word embedding and adversarial stability training to defend adversarial text. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*. AAAI Press, pp. 8384–8391.

Liu Z., **Tang B.**, **Wang X.** and **Chen Q.** (2017). De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of Biomedical Informatics*, **75**, S34–S42. *Supplement: A Natural Language Processing Challenge for Clinical Records: Research Domains Criteria (RDoC) for Psychiatry*.

Ma X. and **Xia F.** (2014). Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume* 1: *Long Papers)*, Baltimore, Maryland. Association for Computational Linguistics, pp. 1337–1348.

Michel P., **Li X.**, **Neubig G.** and **Pino J.** (2019). On evaluation of adversarial perturbations for sequence-to-sequence models. 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)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 3103–3114.

Minegishi M. (2011). Description of thai as an isolating language. *Social Science Information* **50**(1), 62–80.

Miyato T., **Dai A.M.** and **Goodfellow I.J.** (2017). Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Nebhi K., **Bontcheva K.** and **Gorrell G.** (2015). Restoring capitalization in #tweets. In *Proceedings of the 24th International Conference on World Wide Web*, *WWW 2015 Companion*, New York, NY, USA. Association for Computing Machinery, 1111–1115.

Nguyen N. and **Guo Y.** (2007). Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, *ICML 2007*, 681–688, New York, NY, USA. Association for Computing Machinery.

Peters M., **Neumann M.**, **Iyyer M.**, **Gardner M.**, **Clark C.**, **Lee K.** and **Zettlemoyer L.** (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 2227–2237.

Petrov S., **Das D.** and **McDonald R.** (2012). A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA), pp. 2089–2096.

Piktus A., **Edizel N.B.**, **Bojanowski P.**, **Grave E.**, **Ferreira R.** and **Silvestri F.** (2019). Misspelling oblivious word embeddings. 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)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 3226–3234.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Pruthi D., **Dhingra B.** and **Lipton Z.C.** (2019). Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 5582–5591.

Ratinov L. and **Roth D.** (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, Boulder, Colorado. Association for Computational Linguistics, pp. 147–155.

Schroff F., **Kalenichenko D.** and **Philbin J.** (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823.

Sennrich R., **Haddow B.** and **Birch A.** (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume* 1: *Long Papers)*, Berlin, Germany. Association for Computational Linguistics, pp. 1715–1725.

Sornlertlamvanich V., **Takahashi N.** and **Isahara H.** (1998). Thai part-of-speech tagged corpus: Orchid. In *Proceedings of the Oriental COCOSDA Workshop*, pp. 131–138.

Straková J., **Straka M.** and **Hajic J.** (2019). Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5326–5331, Florence, Italy. Association for Computational Linguistics.

Szegedy C., **Zaremba W.**, **Sutskever I.**, **Bruna J.**, **Erhan D.**, **Goodfellow I.** and **Fergus R.** (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Telljohann H., **Hinrichs E.**, **KÜbler S.** and **KÜbler R.** (2004). The tÜba-d/z treebank: Annotating german with a context-free backbone. In *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Citeseer.

Telljohann H., **Hinrichs E.W.**, **KÜbler S.**, **Zinsmeister H.** and **Beck K.** (2005). Stylebook for the tÜbingen treebank of written german (tÜba-d/z). In Seminar fÜr Sprachwissenschaft, *Universität Tübingen, Germany*.

Tirasaroj N. and **Aroonmanakun W.** (2009). Thai named entity recognition based on conditional random fields. In *2009 Eighth International Symposium on Natural Language Processing*, pp. 216–220.

Tjong Kim Sang E.F. and **De Meulder F.** (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147.

Uzuner Z., **Luo Y.** and **Szolovits P.** (2007). Evaluating the State-of-the-Art in Automatic De-identification. *Journal of the American Medical Informatics Association* **14**(5), 550–563.

Wallace E., **Feng S.**, **Kandpal N.**, **Gardner M.** and **Singh S.** (2019). Universal adversarial triggers for attacking and analyzing NLP. 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, pp. 2153–2162.

Xin Y., **Hart E.**, **Mahajan V.** and **Ruvini J.-D.** (2018). Learning better internal structure of words for sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics, pp. 2584–2593.

Yasunaga M., **Kasai J.** and **Radev D.** (2018). Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana. Association for Computational Linguistics, pp. 976–986.

Zhou J.T., **Zhang H.**, **Jin D.**, **Peng X.**, **Xiao Y.** and **Cao Z.** (2019). Roseq: Robust sequence labeling. IEEE Transactions on Neural Networks and Learning Systems.

A. Supplementary results

This section shows the results of adversarial models trained and evaluated on all perturbation types (insertion, transposition, deletion, substitution, and capitalization). This evaluation setting is more suitable to build an adversarial defense system against known adversarial examples. The main difference is that the AT-T method performs better on the NER task, as shown in Table 9. The error analysis on black-box adversarial examples in Table 10 suggests that the NER task is more sensitive to insertion adversarial examples.

Table 10 shows the error rates of the models on five typographical error type. The baseline model is trained on vanilla texts only. As shown in Table 10, the baseline model is the most sensitive to all types of typographical errors on all tasks and corpora. The baseline model is also sensitive to words without perturbation (Clean) for span-based tasks (NER and chunking); this suggests that they were also affected by their surrounding perturbed words. The error rate of the baseline

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 9. F_1 scores and theirs standard deviations on the English CONLL2003 and the Thai ORCHID corpora. We averaged the scores across three runs with different random seeds, and calculated their standard deviations. A bolded number refers to the best F-1 score for each task in each test set. All perturbation types are used to generate adversarial examples for both adversarial training and adversarial evaluation

	original test set	black-box test set	white-box test set
CoNLL2003 (English,NER)			
baseline	91.72 ± 0.24	55.87 ± 1.61	39.53 ± 1.69
AT	91.85 ± 0.00	83.42 ± 0.28	69.06 ± 0.51
AT-P	91.78 ± 0.07	83.74 ± 0.20	69.17 ± 3.07
AT-T	**92.04** ± 0.00	**83.87** ± 1.30	**71.55** ± 2.42
CoNLL2003 (English, Chunking)			
baseline	**91.92** ± 0.11	76.48 ± 0.90	66.03 ± 0.60
AT	91.87 ± 0.07	81.62 ± 0.41	77.90 ± 0.99
AT-P	**91.92** ± 0.04	86.97 ± 0.13	79.65 ± 0.57
AT-T	91.68 ± 0.08	**87.37** ± 0.27	**80.79** ± 1.16
CoNLL2003 (English, PoS)			
baseline	**95.74** ± 0.12	84.01 ± 0.42	77.31 ± 0.29
AT	95.73 ± 0.06	**93.03** ± 0.04	88.79 ± 0.14
AT-P	95.65 ± 0.03	92.93 ± 0.05	**88.86** ± 0.05
AT-T	95.56 ± 0.00	92.83 ± 0.09	88.64 ± 0.33
ORCHID (Thai, PoS)			
baseline	93.89 ± 0.08	77.36 ± 0.31	75.55 ± 1.13
AT	**94.13** ± 0.02	**90.37** ± 0.26	**85.44** ± 0.52
AT-P	93.89 ± 0.03	89.65 ± 0.44	83.21 ± 1.10
AT-T	93.91 ± 0.07	88.76 ± 0.65	83.19 ± 1.11

model on words without perturbation is 25.04 percent for the NER task and 6.31 percent on the chunking task, while the error rate remains low at 2.53 and 3.30 for the part-of-speech tagging task on the ORCHID corpus and the CoNLL2003 corpus, respectively. For the English CoNLL2003 corpus, insertion error has the highest error rates for all tasks for the baseline model. Interestingly, substitution error has lower error rates than transposition error, which contains all the original letters for NER and chunking tasks. For the chunking task, transposition error rates are higher than deletion error, which also alters the length of each word. Surprisingly, capitalization error also has a high error rate even we normalized it for word embeddings. The only difference is that the character-level part of ELMo and the CNN character-to-word embeddings are sensitive to capital letters. Therefore, capitalization is still an important feature for these sequential tagging tasks.

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

Table 10. Error rates and their standard deviations of models on different typographical errors. We calculate the error rates at word-level only. All perturbation types are used to generate adversarial examples for both adversarial training and adversarial evaluation. We excluded words with "O" (non-named entities, or non-syntactic chunks) as their ground truth from the calculation. ‡ - "Clean" refers to unperturbed words, these are words that are shorter than 4 characters. We do not alter words that are shorter than 4 character to ensure that the perturbed words are intelligible to humans. † - Thai has no capitalization

	Insertion	Deletion	Transposition	Substitution	Capitalization	Clean‡
			CoNLL2003 (English,NER)			
Baseline	42.56 ± 1.13	41.61 ± 0.63	40.68 ± 0.38	37.98 ± 0.94	38.96 ± 3.74	25.04 ± 0.07
AT	18.00 ± 0.99	18.53 ± 0.54	18.21 ± 0.54	15.81 ± 0.15	15.00 ± 1.57	**11.38** ± 0.97
AT-P	**17.69** ± 1.80	17.71 ± 3.53	17.82 ± 2.26	15.98 ± 1.76	13.81 ± 5.76	12.02 ± 2.59
AT-T	18.21 ± 0.57	**17.56** ± 1.70	**16.88** ± 0.97	**15.09** ± 1.14	**12.93** ± 0.83	11.68 ± 1.00
			CoNLL2003 (English, Chunking)			
Baseline	17.21 ± 0.94	14.39 ± 0.69	16.13 ± 1.09	15.32 ± 1.32	11.62 ± 0.62	6.31 ± 0.64
AT	7.64 ± 0.34	7.29 ± 0.16	7.64 ± 0.21	6.95 ± 0.14	5.52 ± 0.08	4.21 ± 0.08
AT-P	7.39 ± 0.29	6.57 ± 0.07	7.12 ± 0.19	6.59 ± 0.07	**5.08** ± 0.07	4.16 ± 0.15
AT-T	**7.18** ± 0.58	**6.44** ± 0.40	**7.07** ± 0.61	**6.54** ± 0.17	5.18 ± 0.37	**4.13** ± 0.05
			CoNLL2003 (English, PoS)			
Baseline	34.63 ± 0.23	25.64 ± 2.06	30.76 ± 1.41	30.79 ± 1.11	39.18 ± 1.20	2.53 ± 0.08
AT	**12.92** ± 0.20	12.13 ± 0.10	14.62 ± 0.28	13.25 ± 0.08	10.62 ± 0.37	**2.48** ± 0.08
AT-P	12.97 ± 0.37	**12.04** ± 0.19	**14.10** ± 0.16	**13.15** ± 0.18	**10.53** ± 0.13	2.53 ± 0.00
AT-T	13.27 ± 0.53	12.17 ± 0.25	14.77 ± 0.53	13.29 ± 0.18	10.62 ± 0.59	2.61 ± 0.03
			ORCHID (Thai†, PoS)			
Baseline	44.31 ± 0.80	42.72 ± 0.56	44.73 ± 0.84	46.91 ± 0.68	N/A	3.30 ± 0.16
AT	**16.61** ± 0.81	**16.89** ± 0.47	**16.90** ± 0.87	**20.28** ± 0.77	N/A	**2.57** ± 0.03
AT-P	19.35 ± 1.80	19.75 ± 1.82	19.92 ± 1.58	23.46 ± 2.12	N/A	2.70 ± 0.02
AT-T	20.20 ± 1.81	20.47 ± 1.59	20.11 ± 1.14	24.05 ± 1.72	N/A	2.79 ± 0.06

B. Implications of applying sequential tagger on the Thai language

Thai is an isolating language with SVO word order (Minegishi 2011). Here we introduce three characteristics of Thai language that have implication on the performance of sequential taggers:

- Thai has no word boundary. This introduces more errors early on in the NLP pipeline when applying word segmentation.
- Thai writing system does not have direct orthographic features or special characters such as capital letters to help identify named entities and proper nouns. Instead, we rely on clue words to identify named entities and proper nouns (Tirasaroj and Aroonmanakun 2009).

https://doi.org/10.1017/S1351324921000486 Published online by Cambridge University Press

- There are Thai words that have gone through grammaticalization which introduces ambiguity to part-of-speech tagging. For text with a higher rate of misspellings, we hypothesize that it is harder to disambiguate the POS tags due to noisy context and spurious ambiguity. The following examples show the word paɪ in two different part-of-speech categories:

(1) นิดไปตลาด /nít paɪ tàlàːt/ (Nid goes to the market.)
PN go market
PROPN VERB NOUN

(2) เราทดสอบไปแล้ว /rau tʰótsɔ̀ːp paɪ lɛ́ːʊ/ (We have tested already.)
we test go PAST
PRON VERB AUX ADV

In sentence 1, paɪ is the main verb. Using paɪ here can be compared with "to go" in English. It describes a physical motion away from one place to another (Bilmes 1995). In sentence 2, paɪ is a post-verb auxiliary. Here paɪ is a posthead following the main verb. Using paɪ here can be compared with "already" in English. It describes an orientation of the main verb with respect to time (Bilmes 1995).

Cite this article: Udomcharoenchaikit C, Boonkwan P and Vateekul P (2023). Towards improving the robustness of sequential labeling models against typographical adversarial examples using triplet loss. *Natural Language Engineering* **29**, 287–315. https://doi.org/10.1017/S1351324921000486