Interpretable Adversarial Perturbation in Input Embedding Space for Text

Motoki Sato[1*], **Jun Suzuki**[2,4†], **Hiroyuki Shindo**[3,4], and **Yuji Matsumoto**[3,4]
[1]Preferred Networks, Inc.,
[2]NTT Communication Science Laboratories,
[3]Nara Institute of Science and Technology,
[4]RIKEN Center for Advanced Intelligence Project
sato@preferred.jp, suzuki.jun@lab.ntt.co.jp, {shindo, matsu}@is.naist.jp

Abstract

Following great success in the image processing field, the idea of adversarial training has been applied to tasks in the natural language processing (NLP) field. One promising approach directly applies adversarial training developed in the image processing field to the input word embedding space instead of the discrete input space of texts. However, this approach abandons such *interpretability* as generating adversarial texts to significantly improve the performance of NLP tasks. This paper restores interpretability to such methods by restricting the directions of perturbations toward the existing words in the input embedding space. As a result, we can straightforwardly reconstruct each input with perturbations to an actual text by considering the perturbations to be the replacement of words in the sentence while maintaining or even improving the task performance[1].

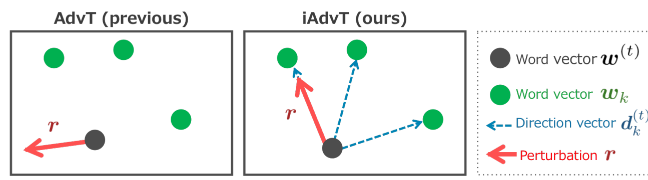

Figure 1: Intuitive sketch to explain our idea: our method (right) restricts perturbations in which words exist in the input word embedding space, whereas previous method (left) allows them to select any direction.

1 Introduction

The existence of (small) perturbations, which induce prediction error in machine learning models, was first discovered and discussed in [Szegedy *et al.*, 2014]. They called the perturbed inputs *adversarial examples*. Such perturbations can be easily found by optimizing the input to maximize the prediction error. After this discovery, a framework called *adversarial training* (AdvT) was proposed [Goodfellow *et al.*, 2015] whose basic idea was to train models that can correctly classify both the original training data and adversarial examples generated based on the training data. Using AdvT, we can further improve the generalization performance of models. This improvement implies that the loss function of adversarial examples works as a good *regularizer* during model training. Currently, a technique for generating adversarial examples is crucial to neural image processing for both improving the task performance and analyzing the behaviors of *black-box* neural models.

Unlike its great success in the image processing field, AdvT cannot be straightforwardly applied to tasks in the natural language processing (NLP) field. This is because we cannot calculate the *perturbed inputs* for tasks in the NLP field since the inputs consist of discrete symbols, which are not a continuous space used in image processing. A novel strategy was recently proposed to improve AdvT for NLP tasks [Miyato *et al.*, 2017] whose key strategy is simple and straightforward: applying AdvT to continuous word embedding space rather than the discrete input space of texts. Their method preserves a theoretical background developed in the image processing field and works well as a regularizer. In fact, this method significantly improved the task performance and achieved the state-of-the-art performance on several text classification tasks. Another notable merit of this method is succinct architecture. It only requires the gradient of the loss function to obtain adversarial perturbations (see Eq. 9). Note that the gradient calculation is a standard calculation procedure for updating the model parameters during training. We can obtain adversarial perturbations in the embedding space with a surprisingly small calculation cost without incorporating any additional sophisticated architecture.

In contrast, the main drawback of this method is that it abandons the generation of adversarial examples interpretable by people since how to appropriately reconstruct perturbed inputs in the input word embedding space to actual texts is not trivial. This implies that this approach lacks *inter-*

[*]This work was conducted when the first author worked at Nara Institute of Science and Technology and RIKEN AIP.

[†]His current affiliation is Tohoku University.

[1]Our code for reproducing our experiments is available at https://github.com/aonotas/interpretable-adv

pretability. In fact, they declared that since their training strategy is no longer intended as a defense against adversaries, they exclusively proposed it as a regularizer to stabilize the model [Miyato *et al.*, 2017]. It is often desirable for researchers and developers to generate adversarial examples (adversarial texts) to understand the behavior of *black-box* neural models. Therefore, a trade-off exists between well-formed and low-cost (gradient-based) approaches and the interpretability of the AdvT methods used in the NLP field.

The main topic of this paper is the reduction of this trade-off gap. This paper restores interpretability while preserving the good ability of regularizer. Our main idea is to only restrict the directions of the perturbations toward the locations of existing words in the word embedding space. Fig. 1 illustrates an intuitive explanation of our idea. With our method, we can straightforwardly interpret each input with a perturbation as an actual sentence by considering the perturbations to be substitutions of the words in the sentence. To the best of our knowledge, our study is the first trial that discusses the interpretability of AdvT based on adversarial perturbation applied to tasks in the NLP field.

2 Related Work

Several studies have applied the ideas of AdvT to certain NLP tasks. A method was proposed that fooled reading comprehension systems by adding sentences to the ends of paragraphs using crowdsourcing [Jia and Liang, 2017]. Random character swaps can break the output of neural machine translation systems [Belinkov and Bisk, 2018; Hosseini *et al.*, 2017], and thus they proposed AdvT methods that generate random character swaps and utilize the generated input sentences as additional training data for their models. Moreover, a method generated a large number of input sentences by replacing a word with its synonym [Samanta and Mehta, 2017]

The primary strategy for generating adversarial examples in the NLP field clearly differs from those developed in the image processing field, which are rather ad-hoc, e.g., using human knowledge [Jia and Liang, 2017], dictionaries [Samanta and Mehta, 2017], or require such costly procedures as exhaustive searches [Samanta and Mehta, 2017]. These methods are not essentially based on the previously discussed idea of perturbation that was first discussed [Szegedy *et al.*, 2014] for generating *adversarial examples*.

In contrast, our baseline method [Miyato *et al.*, 2017] preserves a theoretical background developed in the image processing field. Thus, note that the methods discussed in this paper borrow a distinct strategy from the current primal strategy taken in the NLP field as described above.

3 Target Tasks and Baseline Models

This section briefly explains the formal definitions of our target tasks, text classification and sequence labeling, and the baseline neural models for modeling these tasks. Fig. 2 shows the architecture of the baseline neural models.

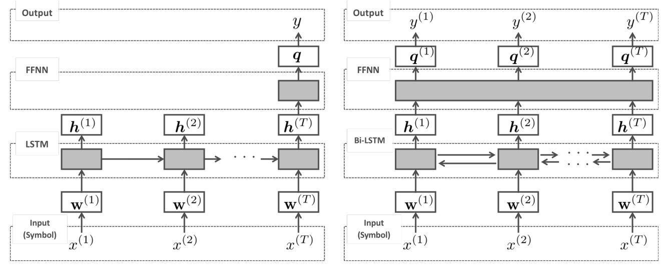

Figure 2: Overview of our baseline neural models: LSTM-based classifier for sentiment classification (left) and Bi-LSTM model for grammatical error detection (right).

3.1 Common Notation

Let X represent an input sentence. $\mathcal{V}$ denotes the vocabulary of the input words. $x^{(t)} \in \mathcal{V}$ is the t-th word that appears in given input sentence X, where $X = (x^{(1)}, \dots, x^{(T)})$ if the number of words in X is T. Here we introduce the following short notation of sequence $(x^{(1)}, \dots, x^{(T)})$ as $(x^{(t)})_{t=1}^{T}$. $\mathcal{Y}$ denotes a set of output classes. To explain the text classification and the sequence labeling tasks in a single framework, this paper assumes that output Y denotes sequence of class labels $Y = (y^{(t)})_{t=1}^{T}$, where $y^{(t)} \in \mathcal{Y}$ for all t in the case of sequence labeling, and class label $Y = y$, where $y \in \mathcal{Y}$ for the text classification case.

Let $\boldsymbol{w}^{(t)}$ be a word embedding vector that corresponds to $x^{(t)}$ whose dimension is D, where $\boldsymbol{w}^{(t)} \in \mathbb{R}^D$. Thus, sequence of word embedding vectors $\tilde{X}$ that corresponds to X can be written as $\tilde{X} = (\boldsymbol{w}^{(t)})_{t=1}^{T}$. Then for text classification, $\tilde{y}$ denotes a corresponding class ID of y in $\mathcal{Y}$. $\tilde{y}$ always takes one integer from 1 to $|\mathcal{Y}|$, where $\tilde{y} \in \{1, \dots, |\mathcal{Y}|\}$. $\tilde{y}^{(t)}$ also denotes a corresponding class ID of $\tilde{y}^{(t)}$ in $\mathcal{Y}$ for sequence labeling. Finally, $\tilde{Y}$ represents $\tilde{Y} = \tilde{y}$ for text classification and $\tilde{Y} = (\tilde{y}^{(t)})_{t=1}^{T}$ for sequence labeling.

Here, without loss of generality, we formulate a text classification task or a sequence labeling task whose inputs and outputs are respectively $\tilde{X}$ and $\tilde{Y}$ instead of X and Y. This is because we can uniquely convert from X to $\tilde{X}$ and from $\tilde{Y}$ to Y. Thus, training data $\mathcal{D}$ can be represented as a set of $\tilde{X}$ and $\tilde{Y}$ pairs, namely, $\mathcal{D} = \{(\tilde{X}^{(n)}, \tilde{Y}^{(n)})\}_{n=1}^{N}$, where N represents the amount of training data.

3.2 Baseline Model for Text Classification

We encode input $\tilde{X}$ with a recurrent neural network (RNN)-based model, which consists of an LSTM unit [Hochreiter and Schmidhuber, 1997]. The (forward) LSTM unit calculates a hidden state in each step t as $\boldsymbol{h}^{(t)} = \mathtt{LSTM}\big(\boldsymbol{w}^{(t)}, \boldsymbol{h}^{(t-1)}\big)$, where $\boldsymbol{h}^{(0)}$ is assumed to be a zero vector. Then we model the (conditional) probability of output $\tilde{Y}$ given input $\tilde{X}$ as follows:

$$p(\tilde{Y} \mid \tilde{X}, \mathcal{W}) = \frac{\exp(q_{\tilde{y}})}{\sum_{m=1}^{|\mathcal{Y}|} \exp(q_m)}, \tag{1}$$

where q_m is the m-th factor of $\boldsymbol{q}$ whose dimension is $|\mathcal{Y}|$. $\boldsymbol{q}$ is calculated through a standard feed-forward neural network from T-th final hidden state $\boldsymbol{h}^{(T)}$, where $\boldsymbol{q} = \mathtt{FFNN}(\boldsymbol{h}^{(T)})$. Here we omit an explanation of the detailed configurations of LSTM and FFNN, but they will be described in our experiment section, since the selection of their configurations affects none of this paper's discussion.

3.3 Baseline Model for Sequence Labeling

We employ a bi-directional LSTM to encode input $\tilde{X}$. The hidden state of each step t, that is, $\boldsymbol{h}^{(t)}$, can be obtained by the concatenation of two hidden states from forward and backward LSTMs: $\boldsymbol{h}^{(t)} = \mathtt{concat}(\boldsymbol{h}_{\mathrm{f}}^{(t)}, \boldsymbol{h}_{\mathrm{b}}^{(t)})$, where $\boldsymbol{h}_{\mathrm{f}}^{(t)} = \mathtt{LSTM}\big(\boldsymbol{w}^{(t)}, \boldsymbol{h}_{\mathrm{f}}^{(t-1)}\big)$, and $\boldsymbol{h}_{\mathrm{b}}^{(t)} = \mathtt{LSTM}\big(\boldsymbol{w}^{(t)}, \boldsymbol{h}_{\mathrm{b}}^{(t+1)}\big)$. We assume that $\boldsymbol{h}_{\mathrm{f}}^{(0)}$ and $\boldsymbol{h}_{\mathrm{b}}^{(T+1)}$ are always zero vectors. We also assume that probability $p(\tilde{Y} \mid \tilde{X}, \mathcal{W})$ can be decomposed into each step t. This means that probability $p(\tilde{Y} \mid \tilde{X}, \mathcal{W})$ can be calculated:

$$p(\tilde{Y} \mid \tilde{X}, \mathcal{W}) = \prod_{t=1}^{T} p(\tilde{y}^{(t)} \mid \tilde{X}, \mathcal{W}) \tag{2}$$

$$p(\tilde{y}^{(t)} \mid \tilde{X}, \mathcal{W}) = \frac{\exp(q_{\tilde{y}^{(t)}})}{\sum_{m=1}^{|\mathcal{Y}|} \exp(q_m)}, \tag{3}$$

where $q_m^{(t)}$ is the m-th factor of $\boldsymbol{q}^{(t)}$ whose dimension is $|\mathcal{Y}|$. $\boldsymbol{q}^{(t)}$ is calculated through a standard feed-forward neural network from t-th final hidden state $\boldsymbol{h}^{(t)}$: $\boldsymbol{q}^{(t)} = \mathtt{FFNN}(\boldsymbol{h}^{(t)})$.

3.4 Training

For training both the text categorization and the sequence labeling, we generally find the optimal parameters of an RNN-based model that can minimize the following optimization problem:

$$\hat{\mathcal{W}} = \underset{\mathcal{W}}{\mathrm{argmin}} \Big\{ \mathcal{J}(\mathcal{D}, \mathcal{W}) \Big\}, \tag{4}$$

where $\mathcal{W}$ represents the overall parameters in the RNN-based model. $\mathcal{J}(\mathcal{D}, \mathcal{W})$ is the loss function over entire training data $\mathcal{D}$, and $\ell(\tilde{X}, \tilde{Y}, \mathcal{W})$ is the loss function of individual training sample $(\tilde{X}, \tilde{Y})$ in $\mathcal{D}$:

$$\mathcal{J}(\mathcal{D}, \mathcal{W}) = \frac{1}{|\mathcal{D}|} \sum_{(\tilde{X}, \tilde{Y}) \in \mathcal{D}} \ell(\tilde{X}, \tilde{Y}, \mathcal{W}) \tag{5}$$

$$\ell(\tilde{X}, \tilde{Y}, \mathcal{W}) = -\log\big(p(\tilde{Y} \mid \tilde{X}, \mathcal{W})\big). \tag{6}$$

4 Adversarial Training in Embedding Space

Adversarial training (AdvT) [Goodfellow *et al.*, 2015] is a novel regularization method that improves the robustness of misclassifying small perturbed inputs. To distinguish the AdvT method in image processing, this paper specifically refers to AdvT that is applied to input word embedding space for NLP tasks as AdvT-Text, which was first introduced in [Miyato *et al.*, 2017].

Let $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$ be a (adversarial) perturbation vector for t-th word $x^{(t)}$ in input $\tilde{X}$. We assume that $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$ is a D-dimensional vector whose dimension always matches that of word embedding vector $\boldsymbol{w}^{(t)}$. Fig. 3 shows the AdvT-Text architecture and our baseline neural models by applying AdvT. See also Fig. 2 for a comparison of the architecture with our baseline neural models. Let $\boldsymbol{r}$ represent a concatenated vector of $\boldsymbol{r}^{(t)}$ for all t. We introduce $\tilde{X}_{+\boldsymbol{r}}$ that denotes $\tilde{X}$ with additional small perturbations, where $\tilde{X}_{+\boldsymbol{r}} = (\boldsymbol{w}^{(t)} + \boldsymbol{r}^{(t)})_{t=1}^{T}$. To obtain (worst case) perturbations $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$ for all t for maximizing the negative log-likelihood (equivalent to minimizing the log-likelihood), we seek optimal solution $\boldsymbol{r}_{\mathtt{AdvT}}$ by maximizing the following equation:

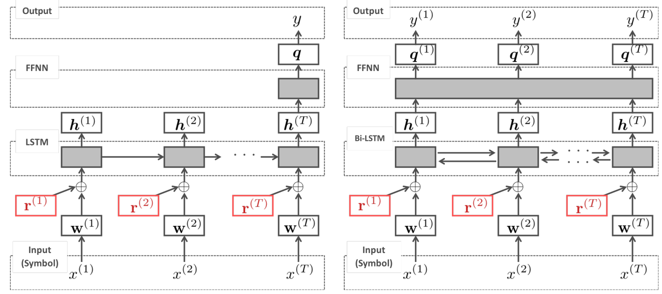

Figure 3: Overview of our neural models with perturbation: $\boldsymbol{r}$ denotes the perturbation, which is $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$, $\boldsymbol{r}_{\mathtt{VAT}}^{(t)}$, $\boldsymbol{r}(\boldsymbol{\alpha}_{\mathtt{iAdvT}}^{(t)})$, or $\boldsymbol{r}(\boldsymbol{\alpha}_{\mathtt{VAT}}^{(t)})$, depending on the method.

$$\boldsymbol{r}_{\mathtt{AdvT}} = \underset{\boldsymbol{r}, ||\boldsymbol{r}|| \le \epsilon}{\mathrm{argmax}} \Big\{ \ell(\tilde{X}_{+\boldsymbol{r}}, \tilde{Y}, \mathcal{W}) \Big\}, \tag{7}$$

where ϵ is a tunable hyper-parameter that controls the norm of the perturbation and $\boldsymbol{r}_{\mathtt{AdvT}}$ represents a concatenated vector of $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$ for all t that resemble $\boldsymbol{r}$. Then based on adversarial perturbation $\boldsymbol{r}_{\mathtt{AdvT}}$, the loss function for AdvT-Text can be defined:

$$\mathcal{J}_{\mathtt{AdvT}}(\mathcal{D}, \mathcal{W}) = \frac{1}{|\mathcal{D}|} \sum_{(\tilde{X}, \tilde{Y}) \in \mathcal{D}} \ell(\tilde{X}_{+\boldsymbol{r}_{\mathtt{AdvT}}}, \tilde{Y}, \mathcal{W}), \tag{8}$$

where $\tilde{X}_{+\boldsymbol{r}_{\mathtt{AdvT}}} = (\boldsymbol{w}^{(t)} + \boldsymbol{r}_{\mathtt{AdvT}}^{(t)})_{t=1}^{T}$, similar to $\tilde{X}_{+\boldsymbol{r}}$.

It is generally infeasible to exactly estimate $\boldsymbol{r}_{\mathtt{AdvT}}$ in Eq. 7 for sophisticated deep neural models. As a solution, an approximation method was proposed by linearizing $\ell(\tilde{X}, \tilde{Y}, \mathcal{W})$ around $\tilde{X}$ [Goodfellow *et al.*, 2015]. For our RNN-based models, the approximation method induces the following non-iterative solution for calculating $\boldsymbol{r}_{\mathtt{AdvT}}^{(t)}$ for all t:

$$\boldsymbol{r}_{\mathtt{AdvT}}^{(t)} = \frac{\epsilon \boldsymbol{g}^{(t)}}{||\boldsymbol{g}||_2}, \quad \boldsymbol{g}^{(t)} = \nabla_{\boldsymbol{w}^{(t)}} \ell(\tilde{X}, \tilde{Y}, \mathcal{W}), \tag{9}$$

where $\boldsymbol{g}$ is a concatenated vector of $\boldsymbol{g}^{(t)}$ for all t.

Finally, we jointly minimize objective functions $\mathcal{J}(\mathcal{D}, \mathcal{W})$ and $\mathcal{J}_{\mathtt{AdvT}}(\mathcal{D}, \mathcal{W})$:

$$\hat{\mathcal{W}} = \underset{\mathcal{W}}{\mathrm{argmin}} \Big\{ \mathcal{J}(\mathcal{D}, \mathcal{W}) + \lambda \mathcal{J}_{\mathtt{AdvT}}(\mathcal{D}, \mathcal{W}) \Big\}, \tag{10}$$

where λ is a coefficient that controls the balance of two loss functions.

5 Interpretable Adversarial Perturbation

As described above, we extended Adv-Text to restore the ability to generate adversarial texts that are interpretable by people while maintaining the task performance. We only restrict the directions of the perturbations in the embedding space toward existing words in the input word embedding space. The intuition behind our method is that the directions to other words can be interpreted as the substitution of another word in the sentence, which may reconstruct adversarial texts. We refer to our AdvT-Text extension as *interpretable AdvT-Text* or iAdvT-Text.

5.1 Definition of Interpretable AdvT-Text

Suppose $\boldsymbol{w}_k$ denotes a word embedding vector that corresponds to the k-th word in vocabulary $\mathcal{V}$. We define *direction vector* $\boldsymbol{d}_k^{(t)}$ that indicates the direction from $\boldsymbol{w}^{(t)}$ to $\boldsymbol{w}_k$ in the input word embedding space:

$$\boldsymbol{d}_k^{(t)} = \frac{\tilde{\boldsymbol{d}}_k^{(t)}}{||\tilde{\boldsymbol{d}}_k^{(t)}||_2}, \quad \text{where} \quad \tilde{\boldsymbol{d}}_k^{(t)} = \boldsymbol{w}_k - \boldsymbol{w}^{(t)}. \tag{11}$$

Note that $\boldsymbol{d}_k^{(t)}$ for all t and k is always a unit vector, $||\boldsymbol{d}_k^{(t)}||_2 = 1$. If the t-th word in the given input sentence is the k-th word in the vocabulary, then $\boldsymbol{w}_k = \boldsymbol{w}^{(t)}$, and thus, $\boldsymbol{d}_k^{(t)}$ becomes a zero vector[2].

Next let $\boldsymbol{\alpha}^{(t)}$ be a $|\mathcal{V}|$-dimensional vector, and let $\alpha_k^{(t)}$ be the k-th factor of $\boldsymbol{\alpha}^{(t)}$, where $\boldsymbol{\alpha}^{(t)} = (\alpha_k^{(t)})_{k=1}^{|\mathcal{V}|}$. We define $\boldsymbol{r}(\boldsymbol{\alpha}^{(t)})$ that denotes the perturbation generated for the t-th word in $\tilde{X}$, which is parameterized by $\boldsymbol{\alpha}^{(t)}$:

$$\boldsymbol{r}(\boldsymbol{\alpha}^{(t)}) = \sum_{k=1}^{|\mathcal{V}|} \alpha_k^{(t)} \boldsymbol{d}_k^{(t)}. \tag{12}$$

$\alpha_k^{(t)}$ is a weight for the direction from the t-th word in the input to the k-th word in the vocabulary. Then, similar to the definition of $\tilde{X}_{+\boldsymbol{r}}$, we also introduce $\tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha})}$ as follows:

$$\tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha})} = \big(\boldsymbol{w}^{(t)} + \boldsymbol{r}(\boldsymbol{\alpha}^{(t)})\big)_{t=1}^{T}. \tag{13}$$

Similar to Eq. 7, we seek the worst case weights of the direction vectors that maximize the loss functions as follows:

$$\boldsymbol{\alpha}_{\texttt{iAdvT}} = \underset{\boldsymbol{\alpha}, ||\boldsymbol{\alpha}|| \le \epsilon}{\operatorname{argmax}} \Big\{ \ell(\tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha})}, \tilde{Y}, \mathcal{W}) \Big\}. \tag{14}$$

Then we define the loss functions of our method, iAdvT-Text, based on $\boldsymbol{\alpha}_{\texttt{iAdvT}}$:

$$\mathcal{J}_{\texttt{iAdvT}}(\mathcal{D}, \mathcal{W}) = \frac{1}{|\mathcal{D}|} \sum_{(\tilde{X}, \tilde{Y}) \in \mathcal{D}} \ell(\tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha}_{\texttt{iAdvT}})}, \tilde{Y}, \mathcal{W}). \tag{15}$$

We substitute $\mathcal{J}_{\texttt{AdvT}}(\mathcal{D}, \mathcal{W})$ in Eq. 10 with $\mathcal{J}_{\texttt{iAdvT}}(\mathcal{D}, \mathcal{W})$ for our method, where the form of the optimization problem can be simply written:

$$\hat{\mathcal{W}} = \underset{\mathcal{W}}{\operatorname{argmin}} \Big\{ \mathcal{J}(\mathcal{D}, \mathcal{W}) + \lambda \mathcal{J}_{\texttt{iAdvT}}(\mathcal{D}, \mathcal{W}) \Big\}. \tag{16}$$

[2]If $\tilde{\boldsymbol{d}}_k^{(t)} = \boldsymbol{0}$, then we treat $\boldsymbol{d}_k^{(t)} = \boldsymbol{0}$.

To reduce the calculation cost, we also introduce an update formula derived by applying the same idea of the approximation method explained in Eq. 9:

$$\boldsymbol{\alpha}_{\texttt{iAdvT}}^{(t)} = \frac{\epsilon \boldsymbol{g}^{(t)}}{||\boldsymbol{g}||_2}, \quad \boldsymbol{g}^{(t)} = \nabla_{\boldsymbol{\alpha}^{(t)}} \ell(\tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha})}, \tilde{Y}, \mathcal{W}). \tag{17}$$

Similar to $\boldsymbol{r}_{\texttt{AdvT}}^{(t)}$, the intuitive interpretation of $\boldsymbol{\alpha}_{\texttt{iAdvT}}^{(t)}$ is the (normalized) strength of each direction $\boldsymbol{d}_k^{(t)}$ about how much to increase the loss function. Thus, we expect to evaluate which direction of words is a good adversarial perturbation.

5.2 Practical Computation

The most time-consuming part of our method is its calculation of the summation of all the words that appeared in Eq. 12, which includes the calculation of directions $\boldsymbol{d}_k^{(t)}$ for all the words from each word $\boldsymbol{w}^{(t)}$, as shown in Eqs. 11. At most, this creates a computational cost of $|\mathcal{V}|^2$, which might be unacceptable compared with the small computational cost of AdvT-Text (the previous method). Here we introduce $\mathcal{V}^{(t)}$ as individual vocabularies of step t, where $\mathcal{V}^{(t)} \subseteq \mathcal{V}$ for all t and $|\mathcal{V}^{(t)}| \ll |\mathcal{V}|$, i.e., $|\mathcal{V}^{(t)}| = 10$. In our method, we select the $|\mathcal{V}^{(t)}|$ nearest neighbor word embeddings around each $\boldsymbol{w}^{(t)}$ for all t in each iteration during the training. This approximation is equivalent to treating $\alpha_k^{(t)} = 0$ for all k if $w_k \notin \mathcal{V}^{(t)}$ for all t. The intuition behind this approximation is that words with a large distance can be treated as nearly unrelated words.

5.3 Extension to Semi-Supervised Learning

Suppose $\mathcal{D}'$ denotes a set of labeled and unlabeled data. *Virtual adversarial training* (VAT) [Miyato *et al.*, 2016] is a (regularization) method closely related to AdvT. VAT, a natural extension of AdvT to semi-supervised learning, can also be applied to tasks in NLP fields, which we refer to as VAT-Text. We borrow this idea and extend it to our iAdvT-Text for a semi-supervised setting, which we refer to as iVAT-Text.

VAT-Text uses the following objective function for estimating the loss of adversarial perturbation $\boldsymbol{r}_{\texttt{VAT}}$:

$$\mathcal{J}_{\texttt{VAT}}(\mathcal{D}', \mathcal{W}) = \frac{1}{|\mathcal{D}'|} \sum_{\tilde{X} \in \mathcal{D}'} \ell_{\texttt{KL}}(\tilde{X}, \tilde{X}_{+\boldsymbol{r}_{\texttt{VAT}}} \mathcal{W}) \tag{18}$$

$$\ell_{\texttt{KL}}(\tilde{X}, \tilde{X}_{+\boldsymbol{r}_{\texttt{VAT}}}, \mathcal{W}) = \texttt{KL}\big(p(\cdot \mid \tilde{X}, \mathcal{W}) || p(\cdot \mid \tilde{X}_{+\boldsymbol{r}_{\texttt{VAT}}}, \mathcal{W})\big),$$

where $\texttt{KL}(\cdot||\cdot)$ denotes the KL divergence. To obtain $\boldsymbol{r}_{\texttt{VAT}}$, we solve the following optimization problem:

$$\boldsymbol{r}_{\texttt{VAT}} = \underset{\boldsymbol{r}, ||\boldsymbol{r}|| \le \epsilon}{\operatorname{argmax}} \Big\{ \texttt{KL}\big(p(\cdot \mid \tilde{X}, \mathcal{W}) || p(\cdot \mid \tilde{X}_{+\boldsymbol{r}}, \mathcal{W})\big) \Big\}. \tag{19}$$

Then instead of solving the above optimization problem, an approximated method was proposed [Miyato *et al.*, 2017]:

$$\boldsymbol{r}_{\texttt{VAT}}^{(t)} = \frac{\epsilon \boldsymbol{g}^{(t)}}{||\boldsymbol{g}||_2}, \quad \boldsymbol{g}^{(t)} = \nabla_{\boldsymbol{w}^{(t)} + \boldsymbol{r}^{(t)}} \ell_{\texttt{KL}}(\tilde{X}, \tilde{X}_{+\boldsymbol{r}}, \mathcal{W}). \tag{20}$$

By using the same derivation technique to obtain the above approximation, we introduce the following equation to calculate $\boldsymbol{\alpha}_{\texttt{iVAT}}^{(t)}$ for an extension to semi-supervised learning:

$$\boldsymbol{\alpha}_{\texttt{iVAT}}^{(t)} = \frac{\epsilon \boldsymbol{g}^{(t)}}{||\boldsymbol{g}||_2}, \quad \boldsymbol{g}^{(t)} = \nabla_{\boldsymbol{\alpha}^{(t)}} \ell_{\texttt{KL}}(\tilde{X}, \tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha})}, \mathcal{W}). \tag{21}$$

Task	Dataset	Train	Dev	Test	Unlabeled
SEC	IMDB	21,246	3,754	25,000	50,000
	Elec	22,500	2,500	25,000	200,000
	Rotten Tomatoes	8,636	960	1,066	7,911,684
CAC	DBpedia	504,000	56,000	70,000	-
	RCV1	14,007	1,557	49,838	668,640
GED	FCE-public	28,731	2,222	2,720	-

Table 1: Summary of datasets

	Hyper-parameter	SEC	CAC	GED
Word embed.	dimensions	256		300
	dropout	0.5		
LSTM	state size	1024		200
	direction	Uni-LSTM		Bi-LSTM
FFNN	dimensions	30	128	50
	activation	ReLU		
Optimization	algorithm	Adam		
	batch size	32		
	initial learning rate	0.001		
	decay rate	0.9998		

Table 2: Summary of hyper-parameters

Method	Test error rate
Baseline	7.05 (%)
Random Perturbation (Labeled)	6.74 (%)
iAdvT-Rand (Ours)	6.69 (%)
iAdvT-Best (Ours)	6.64 (%)
AdvT-Text [Miyato *et al.*, 2017]	6.12 (%)
iAdvT-Text (Ours)	**6.08 (%)**
Random Perturbation (Labeled + Unlabeled)†	6.44 (%)
iVAT-Rand (Ours)†	6.08 (%)
iVAT-Best (Ours)†	6.30 (%)
VAT-Text [Miyato *et al.*, 2017]†	5.69 (%)
iVAT-Text (Ours)†	**5.66 (%)**
Full+Unlabeled+BoW [Maas *et al.*, 2011]	11.11 (%)
Paragraph Vectors [Le and Mikolov, 2014]	7.42 (%)
SA-LSTM [Dai and Le, 2015]†	7.24 (%)
One-hot bi-LSTM [Johnson and Zhang, 2016]†	5.94 (%)

Table 3: Test performance (error rate) on IMDB: lower is better. Semi-supervised learning models are marked with †.

Method	Elec	RCV1	Rotten
Baseline	6.24 (%)	12.01 (%)	17.36 (%)
AdvT-Text [Miyato *et al.*, 2017]	5.94 (%)	10.93 (%)	15.84 (%)
iAdvT-Text (Ours)	**5.58 (%)**	**10.07 (%)**	**14.24 (%)**
VAT-Text [Miyato *et al.*, 2017]†	5.66 (%)	11.80 (%)	14.26 (%)
iVAT-Text (Ours)†	**5.18 (%)**	**11.68 (%)**	**14.12 (%)**

Table 4: Test performance (error rate) on Elec, RCV1, and Rotten Tomatoes: lower is better. Semi-supervised learning models are marked with †.

Then the objective function for iVAT-Text can be written:

$$\mathcal{J}_{\mathtt{iVAT}}(\mathcal{D}', \mathcal{W}) = \frac{1}{|\mathcal{D}'|} \sum_{\tilde{X} \in \mathcal{D}'} \ell_{\mathtt{KL}}(\tilde{X}, \tilde{X}_{+\boldsymbol{r}(\boldsymbol{\alpha}_{\mathtt{iVAT}})} \mathcal{W}). \quad (22)$$

6 Experiments

We conducted our experiments on a sentiment classification (SEC) task, a category classification (CAC) task, and a grammatical error detection (GED) task to evaluate the effectiveness of our methods, iAdvT-Text and iVAT-Text. SEC is a text classification task that classifies a given text into either a positive or a negative class. GED is a sequence labeling task that identifies ungrammatical words.

6.1 Datasets

For SEC, we used the following well-studied benchmark datasets, IMDB [Maas *et al.*, 2011], Elec [Johnson and Zhang, 2015], and Rotten Tomatoes [Pang and Lee, 2005]. In our experiment with the Rotten Tomatoes dataset, we utilized unlabeled examples from the Amazon Reviews dataset[3]. For CAC, we utilized DBpedia [Lehmann *et al.*, 2015] and RCV1 [Lewis *et al.*, 2004]. Since the DBpedia dataset has no additional unlabeled examples, the DBpedia results are only for the supervised learning task. Following [Miyato *et al.*, 2017], we split the original training data into training and development sentences. For GED, we utilized the First Certificate in the English dataset (FCE-public) [Yannakoudakis *et al.*, 2011]. Table 1 summarizes the information about each dataset.

6.2 Model Settings

To fairly compare our methods with previous methods, we followed previously described model configurations [Miyato *et al.*, 2017] for SEC and [Rei and Yannakoudakis, 2016; Kaneko *et al.*, 2017] GED, shown in Fig. 3: left for SEC and right for GED. Moreover, following [Miyato *et al.*, 2017], we initialized the word embeddings and the LSTM weights with a pre-trained RNN-based language model [Bengio *et al.*, 2000] that was trained on labeled training and unlabeled data if they were available. To reduce the computational cost of softmax loss, we use the Adaptive Softmax [Grave *et al.*, 2017] for training language model. We utilized an early stopping criterion [Caruana *et al.*, 2000] based on the performance measured on development sets. The hyper-parameters are summarized in Table 2, with dropout [Srivastava *et al.*, 2014] and Adam [Kingma and Ba, 2014]. In addition, we set $\epsilon = 5.0$ for both AdvT-Text and VAT-Text and $\epsilon = 15.0$ for our method. We also set $\lambda = 1$ for all the methods. To find the best hyper-parameter, we picked models whose performances were best measured on development data.

In addition, we implemented our methods (iAdvT-Text and iVAT-Text) and re-implemented the previous methods (AdvT-Text and VAT-Text) using Chainer [Tokui *et al.*, 2015] with GPU support. All four methods share sub-modules, such as RNN-based models, in our implementation. Therefore, our internal experiments are fairly compared under identical conditions.

6.3 Evaluation by Task Performance

Table 3 shows the IMDB performance evaluated by the error rate. Random perturbation (Labeled) is the method with which we replaced $\boldsymbol{r}^{(t)}_{\mathtt{AdvT}}$ with a random unit vector, and Ran-

[3]http://snap.stanford.edu/data/web-Amazon.html

Method	Test error rate
Baseline	0.94 (%)
AdvT-Text [Miyato *et al.*, 2017]	**0.92 (%)**
iAdvT-Text (Ours)	0.99 (%)
VAT-Text [Miyato *et al.*, 2017]	**0.91 (%)**
iVAT-Text (Ours)	0.93 (%)

Table 5: Test performance (error rate) on DBpedia: lower is better

Method	$F_{0.5}$
Baseline	39.21
Random Perturbation	39.90
AdvT-Text [Miyato *et al.*, 2017]	**42.28**
iAdvT-Text (Ours)	42.26
VAT-Text [Miyato *et al.*, 2017]	41.81
iVAT-Text (Ours)	41.88
BiLSTM w/`Skipgram` [Rei and Yannakoudakis, 2016]	41.1
BiLSTM w/`GWE` [Kaneko *et al.*, 2017]	41.4

Table 6: Test performance ($F_{0.5}$) on GED task: larger is better

dom Perturbation (Labeled + Unlabeled) is the method with which we replaced $\boldsymbol{r}^{(t)}_{\mathtt{VAT}}$ with a random unit vector. We tried two simple methods, iAdvT-Rand and iAdvT-Best. iAdv-Rand is the method with which we replaced $\boldsymbol{r}^{(t)}_{\mathtt{AdvT}}$ with the nearest ten, randomly picked words vectors. iAdv-Best is the method from which we picked the best direction based on Eq. 13. Surprisingly, iAdvT-Text outperformed AdvT-Text, and iVAT-Text achieved the same performance level and slightly outperformed VAT-Text[4]. Note here that our method was mainly designed to restore the interpretability, not to improve the task performance. Before evaluating our methods, iAdvT-Text and iVAT-Text, we assumed that they would respectively degrade the AdvT-Text and VAT-Text performances, since our methods strongly restrict the degrees of freedom for the direction of the perturbations for interpretability. This suggests that the directions of the existence of actual words in the word embedding space provide useful information for improving the generalization performance. The performance of two simple methods (iAdvT-Rand and iAdvT-Best) is poor. Tables 4 and 5 show the performance on the other datasets[5].

Table 6 shows the test performance on the GED task. We used $F_{0.5}$ as an evaluation measure for GED, which was adopted in the CoNLL-14 shared task [Rozovskaya *et al.*, 2014][6]. Other reported results [Rei and Yannakoudakis, 2016; Kaneko *et al.*, 2017] are around the current state-of-the-art performance on this dataset. In our experiments, AdvT-Text achieved the highest $F_{0.5}$. Our experiments revealed that AdvT-Text can further outperform the current state-of-the-art methods. Moreover, our methods successfully reached performances that almost matched AdvT-Text and VAT-Text. Again, we emphasize that these results are substantially positive for our methods since they did not degrade the performance even when we added a strong restriction for calculating the perturbations.

In addition, in contrast to SEC, VAT-Text did not outperform AdvT-Text. Since the GED dataset does not contain a large amount of unlabeled data, we confirmed that it is hard for VAT-Text to improve the performance.

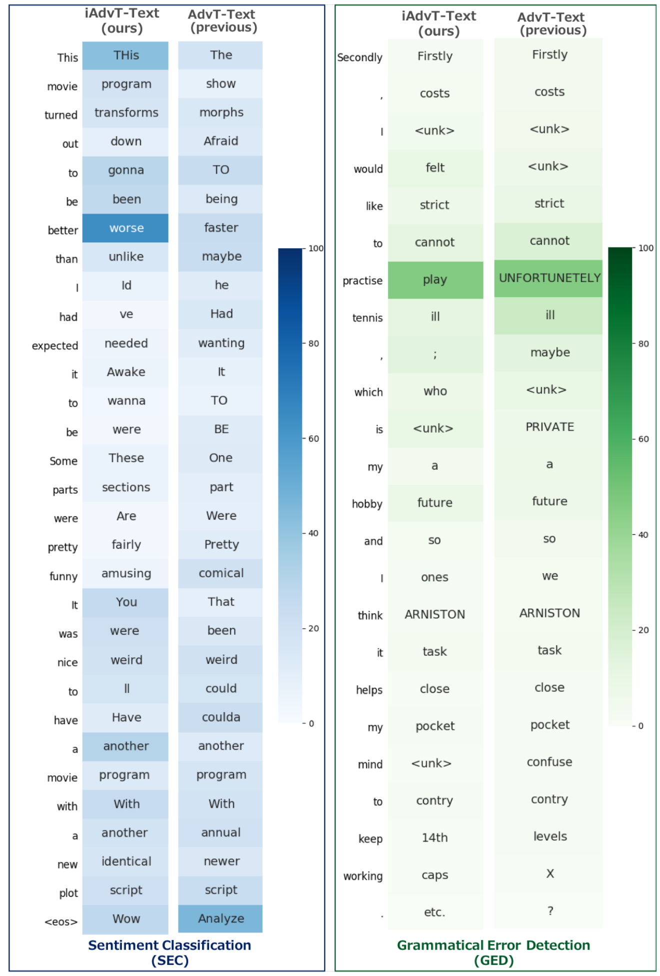

Figure 4: Visualization of perturbation at sentence-level: Texts at left of blue or green bars are sentences in datasets, and texts in blue or green bars are words reconstructed from perturbations.

6.4 Visualization of Sentence-Level Perturbations

We visualized the perturbations computed by our method (iAdvT-Text) in Fig. 4 for understanding its behavior. We also visualized the perturbations by the previous method (AdvT-Text) for comparison. The words at the left of each (blue or green) bar indicate the words in the (true) sentences in the dataset. We selected the highest direction toward a word from each word in the sentence. In our method, it can be easily obtained by selecting the maximum values of $\alpha^{(t)}_k$ for all t.

[4]The AdvT-Text and VAT-Text scores were obtained by our re-implemented code, which outperformed the original scores [Miyato *et al.*, 2017] (Adv:6.21 %, VAT:5.91 %).

[5]For RCV1 and DBpedia, our baselines were slightly weak due to the resource limitation of constracting the large-scale pre-trained language models.

[6]A previous study [Nagata and Nakatani, 2010] suggested that since accurate prediction is more important than coverage in error detection applications, $F_{0.5}$ was selected rather than F_1.

	Sentence (SEC)	Prediction
Original Sentence	The essence of this film falls on judgments by police officers who fortunately ethical and moral men act on situations within situations in a city with a super abundance of violence and killing Good compound interacting story lines and above average characterizations <eos>	Positive
Adversarial Example	The essence from THIS film falls on judgments by police officers who fortunately ethical and moral men act on situations within situations in a city with a super abundance of violence and killing Good compound interacting story lines and above average characterizations <eos>	Negative
Original Sentence	There is really but one thing to say about this sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness <eos>	Negative
Adversarial Example	There is really but one thing to say about that sorry movie It should never have been made The first one one of my favourites An American Werewolf in London is a great movie with a good plot good actors and good FX But this one It stinks to heaven with a cry of helplessness <eos>	Positive

	Sentence (GED)	Prediction	Correct Replacement
Original Sentence	We all want to thank you for having choose such good places in London .	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0	
Adversarial Example	We all want to thank you for having choosing such good places in London .	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	(choose → chosen)
Original Sentence	I am not really satisfied about it .	0 0 0 0 0 1 0 0	
Adversarial Example	I am not really satisfied more it .	0 0 0 0 0 0 0 0	(about → with)

Table 7: Adversarial examples, misclassified by trained model and reconstructed by iAdvT-Text

For AdvT-Text, we calculated the cosine similarities between the perturbation and direction to every word $\boldsymbol{w}_k$ and selected a word with the highest cosine similarity. Each word written in the (blue or green) bar represents the selected word by the above operations, and shades of color are the relative strengths of the obtained perturbations toward the selected words.

For the SEC task, the correct label of the sentence in the blue bar is *positive*. The iAdvT-Text successfully found the directions for replacing *better* with *worse* to increase the loss. In other words, the direction might change the class label from positive to negative. For the GED task, the sentence in the green bar contains a grammatical error word (*practise*), which should be replaced with *play*. The iAdvT-Text also found directions for replacing *practise* with *play*.

In contrast, the perturbations of AdvT-Text (Previous) were uninterpretable (replacing *<eos>* with *Analyze*, and replacing *practise* with *UNFORTUNETELY*). This is mainly because the perturbations of AdvT-Text barely matched the direction toward any existence points of word embeddings, and we just visualized the most cosine similar words with perturbation.

These results revealed that the directions of the perturbations in iAdvT-Text are understandable by humans, and thus, offer a chance for researchers to interpret black-box neural models, regardless whether the model properly learned certain phenomena that the researchers are interested in. We believe that such interpretability is critical, especially for sophisticated neural models. The usefulness of this visualization is the main claim of our proposed methods.

6.5 Adversarial Texts

We reconstructed adversarial examples, which misclassified the trained models, from the adversarial perturbations in the input word embedding space given by iAdvT-Text. To obtain adversarial texts, we first identified the largest perturbation and replaced the original word with one that matches the largest perturbation.

Table 7 shows typical examples, where the top two rows show an example for SEC, and the bottom two rows show an example for GED. For example, the second example in Table 7 was generated by changing *this* to *that*. Even though this example does not alter the meaning, the prediction was changed from Negative → Positive. The generated adversarial texts for GED still contain grammatical error; however the model predicts that they are grammatically correct. Thus, these two examples are adversarial texts.

Note that the previous methods, AdvT-Text and VAT-Text, hardly reconstruct such effective adversarial texts. Thus, this is a clear advantage of our methods compared with the previous ones.

7 Conclusion

This paper discussed the interpretability of adversarial training based on adversarial perturbation that was applied to tasks in the NLP field. Our proposal restricted the directions of perturbations toward the locations of existing words in the word embedding space. We demonstrated that our methods can successfully generate reasonable adversarial texts and interpretable visualizations of perturbations in the input embedding space, which we believe will greatly help researchers analyze a model's behavior. In addition, we confirmed that our methods, iAdvT-Text and iVAT-Text, maintained or improved the state-of-the-art performance obtained by our baseline methods, AdvT-Text and VAT-Text, in well-studied sentiment classification (SEC), category classification (CAC), and grammatical error detection (GED) benchmark datasets.

Acknowledgments

We thank four anonymous reviewers for their helpful comments. We also thank Takeru Miyato who suggested that we reproduce the result of a previous work [Miyato *et al.*, 2017] as well as Masahiro Kaneko and Tomoya Mizumoto who provided helpful comments.

References

[Belinkov and Bisk, 2018] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *ICLR*, 2018.

[Bengio *et al.*, 2000] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2000.

[Caruana *et al.*, 2000] Rich Caruana, Steve Lawrence, and C. Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NIPS*, pages 402–408, 2000.

[Dai and Le, 2015] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *NIPS*, pages 3079–3087, 2015.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[Grave *et al.*, 2017] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. In *ICML*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hosseini *et al.*, 2017] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments. *CoRR*, abs/1702.08138, 2017.

[Jia and Liang, 2017] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, pages 2021–2031, 2017.

[Johnson and Zhang, 2015] Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *NIPS*, volume 28, pages 919–927, 2015.

[Johnson and Zhang, 2016] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. In *ICML*, pages 526–534, 2016.

[Kaneko *et al.*, 2017] Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. Grammatical error detection using error-and grammaticality-specific word embeddings. In *IJCNLP*, volume 1, pages 40–48, 2017.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Le and Mikolov, 2014] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[Lehmann *et al.*, 2015] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195, 2015.

[Lewis *et al.*, 2004] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150, 2011.

[Miyato *et al.*, 2016] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2016.

[Miyato *et al.*, 2017] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. In *ICLR*, 2017.

[Nagata and Nakatani, 2010] Ryo Nagata and Kazuhide Nakatani. Evaluating performance of grammatical error detection to maximize learning effect. In *COLING*, pages 894–900, 2010.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[Rei and Yannakoudakis, 2016] Marek Rei and Helen Yannakoudakis. Compositional sequence labeling models for error detection in learner writing. In *ACL*, pages 1181–1191, 2016.

[Rozovskaya *et al.*, 2014] Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. The illinois-columbia system in the conll-2014 shared task. In *CoNLL Shared Task*, pages 34–42, 2014.

[Samanta and Mehta, 2017] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[Tokui *et al.*, 2015] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, volume 5, 2015.

[Yannakoudakis *et al.*, 2011] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading esol texts. In *ACL*, pages 180–189. ACL, 2011.