Interpretable Adversarial Perturbation in Input Embedding Space for Text

> *[PAPER]()*
>
> *[CODE]()*

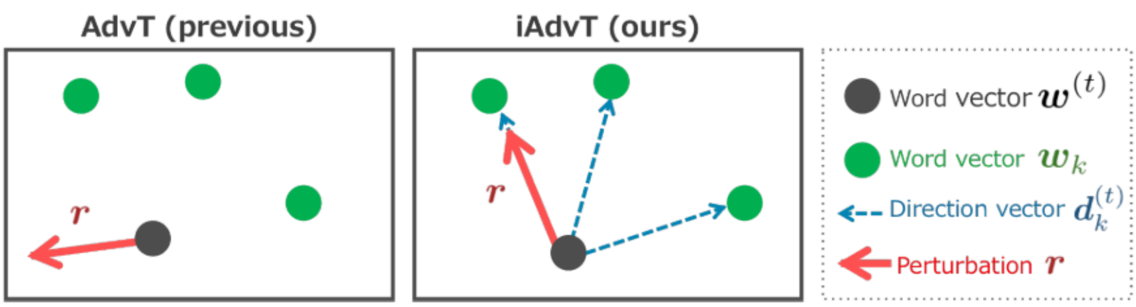
Abstract

> *The author propose an white & black box method to craft text adversarial samples. He designs three perturbation strategies:* insertion, modification, deletion*. The attack method is tested in SOTA character & word level DNN-based text classifiers.*

Motivation

- *Miyaota et al.2017* abandons the generation of adversarial examples interpretable by people, the perturbed embedding might have no meaning (corresponding to no words)
- trade-off exists between well- formed and low-cost (gradient-based) approaches and the in- terpretability of the AdvT methods used in the NLP field

Main Idea



Restrict the directions of the perturbations toward the locations of existing words in the word embedding space. Interpret each input with a perturbation as an actual sentence by considering the perturbations to be substitutions of the words in the sentence.

Training

Goal:

$$\hat{w} = \arg\min_{w} J(D, W) \tag{1}$$

D : entire training data, W: overall parameters of model

$$J(D, W) = \frac{1}{|D|} \sum_{(\tilde{X}, \tilde{Y}, W)} l(\tilde{X}, \tilde{Y}, W)) \tag{2}$$

$$l(\tilde{X}, \tilde{Y}, W) = -log(P(\tilde{Y}|\tilde{X}, W)) \tag{3}$$

Adversarial Training

Denote $r^{(t)}_{AdvT}$ as the adversarial perturbation vector for $t-th$ word $x^{(t)}$ word in input $\tilde{X}$

$\tilde{X}_{+r} = (w^{(t)} + r^{(t)})^{T}_{t=1}$ denotes $\tilde{X}$ with perturbations

Worst-case perturbations:

$$r_{AdvT} = \arg\max_{r,||r||<=\epsilon} l(\tilde{X}_{+r}, \tilde{Y}, W) \tag{4}$$

Loss for Adv text:

$$J_{AdvT}(D, W) = \frac{1}{|D|} \sum_{(\tilde{X}, \tilde{Y}) \in D} l(\tilde{X}_{+r_{AdvT}}, \tilde{Y}, W) \tag{5}$$

Approximating by **linearizing**

$$r^{(t)}_{AdvT} = \frac{\epsilon g^{(t)}}{||g||_2}, g^{(t)} = \nabla_{w^{(t)}} l(\tilde{X}, \tilde{Y}, W) \tag{6}$$

Goal

$$\hat{w} = \arg\min_{w} J(D, W) + J_{AdvT}(D, W) \tag{7}$$

Interpretable Adversarial Training

Let w_k denotes the word embedding vector corresponding the $k-$th word in vocabulary V

direction vector $d^{(t)}_{k}$ indicates the direction from $w^{(t)}$ to w_k in embedding space

$$d_k^{(t)} = \frac{\tilde{d}_k^{(t)}}{||\tilde{d}_k^{(t)}||_2}, \tilde{d}_k^{(t)} = w_k - w^{(t)} \tag{8}$$

Let $\alpha_k^{(t)}$ be the weight for direction from t-th word in the input, $\alpha^{(t)} = (\alpha_k^{(t)})_{k=1}^{|V|}$

The perturbation generated for the t-th word:

$$r(\alpha^{(t)}) = \sum_{k=1}^{|V|} \alpha_k^{(t)} d_k^{(t)} \tag{9}$$

Perturbation on $\tilde{X}$: $\tilde{X}_{+r(\alpha)} = (w^{(t)} + r(\alpha^{(t)}))_{t=1}^{T}$

Find the worst case weights of weight vectors that maximize the loss function

$$\alpha_{iAdvT} = \arg \max_{\alpha, ||\alpha|| <= \epsilon} l(\tilde{X}_{+r(\alpha)}, \tilde{Y}, W) \tag{10}$$

Loss of iAdv text:

$$J_{iAdvT}(D, W) = \frac{1}{|D|} \sum_{(\tilde{X}, \tilde{Y}) \in D} l(\tilde{X}_{+r(\alpha_{iAdvT})}, \tilde{Y}, W) \tag{11}$$

Approximating by **linearizing**

$$\alpha_{iAdvT}^{(t)} = \frac{\epsilon g^{(t)}}{||g||_2}, g^{(t)} = \nabla_{\alpha^{(t)}} l(\tilde{X}_{+r(\alpha)}, \tilde{Y}, W) \tag{12}$$

Codes:

```
# Classification loss
output = model(x, x_length)
output_original = output
loss = F.softmax_cross_entropy(output, y, normalize=True)
    if args.use_adv:
        output = model(x, x_length, first_step=True, d=None)
        # Adversarial loss (First step)
        loss_adv_first = F.softmax_cross_entropy(output, y,
normalize=True)
        model.cleargrads()
        loss_adv_first.backward()

        if args.use_attn_d:
```

```
            # iAdv
            attn_d_grad = model.attention_d_var.grad #g
            attn_d_grad = F.normalize(attn_d_grad, axis=1) #g/||g||
            # Get directional vector
            dir_normed = model.dir_normed.data #d^(t)
            attn_d = F.broadcast_to(attn_d_grad,
dir_normed.shape).data
            d = xp.sum(attn_d * dir_normed, axis=1) # r(\alpha)
        else:
            # Adv
            d = model.d_var.grad #r_adv
        output = model(x, x_length, d=d) #X+r(\alpha)
        # Adversarial loss
        loss_adv = F.softmax_cross_entropy(output, y, normalize=True)
        loss += loss_adv * args.nl_factor
```

Practical Computation

Equation (9) is the most time-consuming operation, cost: $|V|^2$

Solution: choose a small vocabulary $V^{(t)}$ for each step t.

select the $|V^{(t)}|$ nearest neighbor word embeddings around $w^{(t)}$ (let $\alpha_k^{(t)} = 0$ for all k if $w_k \notin V^{(t)}$)

(words with large distance can be treated as nearly unrelated words)