Adversarial Training for Pre-trained Models

Dong Wang

Overview

- FreeLB: Enhanced Adversarial Training for Natural Language Understanding （ICLR 2020）
- SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization (ACL 2020)
- TextAT: Adversarial Training with Token-Aware Perturbation for Natural Language Understanding (arxiv 2004.14543)
- ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators (ICLR 2020)
- Revisiting Pre-Trained Models for Chinese Natural Language Processing (arxiv 2004.13922)

Introduction

- **Adversarial training** is a method for **creating robust neural networks**. During adversarial training, **mini-batches of training samples are contaminated with adversarial perturbations** (alterations that are small and yet cause misclassification), and then used to update network parameters until the resulting model learns to resist such attacks.

- In CV(Computer Vision), adversarial training can **improve the robustness**, but it usually leads to the **reduction of generalization**. In NLP, adversarial training **improves both robustness and generalization.**

Introduction

- **Adversarial Training**

  Adds adversarial perturbations to word embeddings and minimizes the resultant adversarial loss around input samples.

  - e.g. PGD, FreeLB, SMART, TextAT

- **Adversarial Example in Natural Languages**

  Produce actual adversarial examples.

  - e.g. ELECTRA, MacBERT

FreeLB: Enhanced Adversarial Training for Natural Language Understanding

Chen Zhu[1], **Yu Cheng**[2], **Zhe Gan**[2], **Siqi Sun**[2], **Tom Goldstein**[1], **Jingjing Liu**[2]
[1]University of Maryland, College Park [2]Microsoft Dynamics 365 AI Research
{chenzhu,tomg}@cs.umd.edu, {yu.cheng,zhe.gan,siqi.sun,jingjl}@microsoft.com

Method

Standard adversarial training seeks to find optimal parameters $\boldsymbol{\theta}^*$ to minimize the maximum risk for any $\boldsymbol{\delta}$ within a norm ball as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{Z},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|\leq\epsilon} L(f_{\boldsymbol{\theta}}(\boldsymbol{X}+\boldsymbol{\delta}), y) \right], \tag{1}$$

where $\mathcal{D}$ is the data distribution, y is the label, and L is some loss function. We use the Frobenius norm to constrain $\boldsymbol{\delta}$. For neural networks, the outer "min" is non-convex, and the inner "max" is non-concave.

Increase loss in input and decrease loss in parameter

Method

- PGD

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{Z},y)\sim\mathcal{D}} \left[\max_{\|\boldsymbol{\delta}\|\leq\epsilon} L(f_{\boldsymbol{\theta}}(\boldsymbol{X}+\boldsymbol{\delta}), y) \right],$$

$$\boldsymbol{\delta}_{t+1} = \Pi_{\|\boldsymbol{\delta}\|_F\leq\epsilon} \left(\boldsymbol{\delta}_t + \alpha g(\boldsymbol{\delta}_t)/\|g(\boldsymbol{\delta}_t)\|_F \right), \tag{2}$$

where $g(\boldsymbol{\delta}_t) = \nabla_{\boldsymbol{\delta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{X}+\boldsymbol{\delta}_t), y)$ is the gradient of the loss with respect to $\boldsymbol{\delta}$, and $\Pi_{\|\boldsymbol{\delta}\|_F\leq\epsilon}$ performs a projection onto the ϵ-ball. To achieve high-level robustness, multi-step adversarial examples are needed during training, which is computationally expensive. The K-step PGD (K-PGD) requires K forward-backward passes through the network, while the standard SGD update requires only one. As a result, the adversary generation step in adversarial training increases run-time by an order of magnitudea catastrophic amount when training large state-of-the-art language models.

Method

Algorithm 1 "Free" Large-Batch Adversarial Training (FreeLB-K)

Require: Training samples $X = \{(\boldsymbol{Z}, y)\}$, perturbation bound ϵ, learning rate τ, ascent steps K, ascent step size α

Initialize $\boldsymbol{\theta}$
for epoch $= 1 \dots N_{ep}$ **do**
    for minibatch $B \subset X$ **do**
        $\boldsymbol{\delta}_0 \leftarrow \frac{1}{\sqrt{N_{\boldsymbol{\delta}}}} U(-\epsilon, \epsilon)$
        $\boldsymbol{g}_0 \leftarrow 0$
        for $t = 1 \dots K$ **do**
            Accumulate gradient of parameters θ
                $\boldsymbol{g}_t \leftarrow \boldsymbol{g}_{t-1} + \frac{1}{K}\mathbb{E}_{(\boldsymbol{Z},y)\in B}[\nabla_{\boldsymbol{\theta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{X} + \boldsymbol{\delta}_{t-1}), y)]$
            Update the perturbation δ via gradient ascend
                $\boldsymbol{g}_{adv} \leftarrow \nabla_{\boldsymbol{\delta}} L(f_{\boldsymbol{\theta}}(\boldsymbol{X} + \boldsymbol{\delta}_{t-1}), y)$
                $\boldsymbol{\delta}_t \leftarrow \Pi_{\|\boldsymbol{\delta}\|_F \le \epsilon}(\boldsymbol{\delta}_{t-1} + \alpha \cdot \boldsymbol{g}_{adv} / \|\boldsymbol{g}_{adv}\|_F)$
        end for
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \tau \boldsymbol{g}_K$
    end for
end for

Experiment

Method	MNLI (Acc)	QNLI (Acc)	QQP (Acc)	RTE (Acc)	SST-2 (Acc)	MRPC (Acc)	CoLA (Mcc)	STS-B (Pearson)
Reported	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4
ReImp	-	-	-	85.61 (1.7)	96.56 (.3)	90.69 (.5)	67.57 (1.3)	92.20 (.2)
PGD	90.53 (.2)	94.87 (.2)	92.49 (.07)	87.41 (.9)	96.44 (.1)	90.93 (.2)	69.67 (1.2)	92.43 (7.)
FreeAT	90.02 (.2)	94.66 (.2)	92.48 (.08)	86.69 (15.)	96.10 (.2)	90.69 (.4)	68.80 (1.3)	92.40 (.3)
FreeLB	**90.61** (.1)	**94.98** (.2)	**92.60** (.03)	**88.13** (1.2)	**96.79** (.2)	**91.42** (.7)	**71.12** (.9)	**92.67** (.08)

Table 1: Results (median and variance) on the dev sets of GLUE based on the RoBERTa-large model, from 5 runs with the same hyperparameter but different random seeds. ReImp is our reimplementation of RoBERTa-large. The training process can be very unstable even with the vanilla version. Here, both PGD on STS-B and FreeAT on RTE demonstrates such instability, with one unconverged instance out of five.

Experiment

Model	Score	CoLA 8.5k	SST-2 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX
BERT-base[1]	78.3	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	65.1	34.2
FreeLB-BERT	79.4	54.5	93.6	88.1/83.5	87.7/86.7	72.7/89.6	85.7/84.6	91.8	70.1	65.1	36.9
MT-DNN[2]	87.6	**68.4**	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9/87.4	96.0	86.3	89.0	42.8
XLNet-Large[3]	88.4	67.8	**96.8**	93.0/90.7	91.6/91.1	74.2/90.3	90.2/89.8	98.6	86.3	**90.4**	47.5
RoBERTa[4]	88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	**98.9**	88.2	89.0	48.7
FreeLB-RoB	**88.8**	68.0	**96.8**	**93.1/90.8**	**92.4/92.2**	**74.8/90.3**	**91.1/90.7**	98.8	**88.7**	89.0	**50.1**
Human	87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-

Table 2: Results on GLUE from the evaluation server, as of Sep 25, 2019. Metrics are the same as the leaderboard. Number under each task's name is the size of the training set. FreeLB-BERT is the single-model results of BERT-base finetuned with FreeLB, and FreeLB-RoB is the ensemble of 7 RoBERTa-Large models for each task. References: [1]: (Devlin et al., 2019); [2]: (Liu et al., 2019a); [3]: (Yang et al., 2019); [4]: (Liu et al., 2019b).

Summary

- The method leverages recently proposed “free” training strategies (accumulate gradient of parametes) to enrich the training data with diversified adversarial samples at no extra cost than PGD-based adversarial training.
- Perform diversified adversarial training on large-scale state-of-the-art models.
- Only adversarial examples are used for training.

SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization

Haoming Jiang *
Georgia Tech
jianghm@gatech.edu

Pengcheng He, Weizhu Chen
Microsoft Dynamics 365 AI
{penhe,wzchen}@microsoft.com

Xiaodong Liu, Jianfeng Gao
Microsoft Research
{xiaodl,jfgao}@microsoft.com

Tuo Zhao
Georgia Tech
tourzhao@gatech.edu

Introduction

- Due to the limited data from the target task/domain and the extremely **high complexity** of the pre-trained model, **aggressive fine-tuning** often makes the adapted model overfit the training data of the target task/domain and therefore does not generalize well to unseen data.

- To effectively control the extremely high complexity of the model, this method propose a **Smoothness-inducing Adversarial Regularization** technique.

Smoothness-inducing Adversarial Regularization

- This method solves the following optimization for fine-tuning:

$$\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\mathrm{s}} \mathcal{R}_{\mathrm{s}}(\theta), \qquad (1)$$

where $\mathcal{L}(\theta)$ is the loss function defined as

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i),$$

Here we define $\mathcal{R}_{\mathrm{s}}(\theta)$ as

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\widetilde{x}_i - x_i\|_p \le \epsilon} \ell_{\mathrm{s}}(f(\widetilde{x}_i; \theta), f(x_i; \theta)),$$

Smoothness-inducing Adversarial Regularization

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\widetilde{x}_i - x_i\|_p \le \epsilon} \ell_{\mathrm{s}}(f(\widetilde{x}_i;\theta), f(x_i;\theta)),$$

By minimizing the objective, we can encourage f to be smooth within the neighborhoods of all input. Such a smoothness-inducing property is particularly helpful to **prevent overfitting and improve generalization on a low resource target domain for a certain task.**

Smoothness-inducing Adversarial Regularization

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\widetilde{x}_i - x_i\|_p \leq \epsilon} \ell_{\mathrm{s}}(f(\widetilde{x}_i; \theta), f(x_i; \theta)),$$

For classification tasks, $f(\cdot;\theta)$ outputs a probability simplex, and $l_s$is chosen as the symmetrized KL-divergence:

$$\ell_{\mathrm{s}}(P, Q) = \mathcal{D}_{\mathrm{KL}}(P||Q) + \mathcal{D}_{\mathrm{KL}}(Q||P);$$

For regression tasks, $f(\cdot;\theta)$ outputs a scalar, and $l_s$is chosen as the squared loss:

$$l_s(p,q) = (p-q)^2$$

Experiment

Model	MNLI-m/mm Acc	QQP Acc/F1	RTE Acc	QNLI Acc	MRPC Acc/F1	CoLA Mcc	SST Acc	STS-B P/S Corr
$BERT_{BASE}$								
BERT (Devlin et al., 2019)	84.4/-	-	-	88.4	-/86.7	-	92.7	-
$BERT_{ReImp}$	84.5/84.4	90.9/88.3	63.5	91.1	84.1/89.0	54.7	92.9	89.2/88.8
$SMART_{BERT}$	**85.6/86.0**	**91.5/88.5**	**71.2**	**91.7**	**87.7/91.3**	**59.1**	**93.0**	**90.0/89.4**
$RoBERTa_{LARGE}$								
RoBERTa (Liu et al., 2019c)	90.2/-	92.2/-	86.6	94.7	-/90.9	68.0	96.4	92.4/-
PGD (Zhu et al., 2020)	90.5/-	92.5/-	87.4	94.9	-/90.9	69.7	96.4	92.4/-
FreeAT (Zhu et al., 2020)	90.0/-	92.5/-	86.7	94.7	-/90.7	68.8	96.1	92.4/-
FreeLB (Zhu et al., 2020)	90.6/-	**92.6/-**	88.1	95.0	-/91.4	**71.1**	96.7	92.7/-
$SMART_{RoBERTa}$	**91.1/91.3**	92.4/89.8	**92.0**	**95.6**	**89.2/92.1**	70.6	**96.9**	**92.8/92.6**

Table 1: Main results on GLUE development set. The best result on each task produced by a single model is in **bold** and “-” denotes the missed result.

Experiment

Model /#Train	CoLA 8.5k	SST 67k	MRPC 3.7k	STS-B 7k	QQP 364k	MNLI-m/mm 393k	QNLI 108k	RTE 2.5k	WNLI 634	AX	**Score**	#param
Human Performance	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0/92.8	91.2	93.6	95.9	-	87.1	-
Ensemble Models												
RoBERTa[1]	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8/90.2	98.9	88.2	89.0	48.7	88.5	356M
FreeLB[2]	68.0	96.8	93.1/90.8	92.4/92.2	**74.8**/90.3	91.1/90.7	98.8	88.7	89.0	50.1	88.8	356M
ALICE[3]	69.2	97.1	93.6/91.5	92.7/92.3	74.4/**90.7**	90.7/90.2	**99.2**	87.3	89.7	47.8	89.0	340M
ALBERT[4]	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3/91.0	**99.2**	89.2	91.8	50.2	89.4	235M*
MT-DNN-SMART[†]	69.5	**97.5**	**93.7/91.6**	**92.9/92.5**	73.9/90.2	91.0/90.8	**99.2**	89.7	94.5	50.2	**89.9**	356M
Single Model												
BERT$_{\text{LARGE}}$[5]	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5	335M
MT-DNN[6]	62.5	95.6	90.0/86.7	88.3/87.7	72.4/89.6	86.7/86.0	93.1	75.5	65.1	40.3	82.7	335M
T5[8]	**70.8**	97.1	91.9/89.2	92.5/92.1	74.6/90.4	**92.0/91.7**	96.7	**92.5**	**93.2**	**53.1**	89.7	11,000M
SMART$_{\text{RoBERTa}}$	65.1	**97.5**	**93.7/91.6**	**92.9/92.5**	74.0/90.1	91.0/90.8	95.4	87.9	91.8[8]	50.2	88.4	356M

Table 2: GLUE test set results scored using the GLUE evaluation server. The state-of-the-art results are in **bold**. All the results were obtained from https://gluebenchmark.com/leaderboard on December 5, 2019. SMART uses the classification objective on QNLI. Model references: [1] Liu et al. (2019c); [2]Zhu et al. (2020); [3]Wang et al. (2019); [4]Lan et al. (2019); [5] Devlin et al. (2019); [6] Liu et al. (2019b); [7] Raffel et al. (2019) and [8] He et al. (2019), Kocijan et al. (2019). * ALBERT uses a model similar in size, architecture and computation cost to a 3,000M BERT (though it has dramatically fewer parameters due to parameter sharing). [†] Mixed results from ensemble and single of MT-DNN-SMART and with data augmentation.

Summary

- This method propose an **explicit regularization** to effectively control the model complexity at the fine-tuning stage.

- This method **compare the adversarial example with the normal example**.

TextAT: Adversarial Training with Token-Aware Perturbation for Natural Language Understanding

Linyang Li, Xipeng Qiu
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{linyangli19,xpqiu}@fudan.edu.cn

Introduction

Different from pixels in images or signals in audios, embeddings used in texts **possess abundant semantic information**. Therefore, perturbations are less focused on certain tokens when randomly initialized within the batch processing. To tackle this problem, this paper accumulate the perturbations of discrete tokens **throughout the training process**.

In this paper, we introduce two steps to create better adversarial samples:

(1) global accumulated token perturbation;

(2) discrete token normalization ball.

Method

(1) global accumulated token perturbation

We create global accumulated perturbation $Z \in \mathbb{R}^{N \times D}$, where N is the vocabulary size of model embedding space. For each batch, **adversarial perturbations are initialized by the corresponding perturbation from the global accumulated perturbation Z**. After K steps of adversarial training forward pass, we accumulate the gradients calculated by the given data and update the global accumulated perturbation Z.

$$\boldsymbol{\eta}_0^i \leftarrow \boldsymbol{Z}[w_i]$$

$$\boldsymbol{g}_t \leftarrow \boldsymbol{g}_{t-1} + \frac{1}{K}\mathbb{E}_{(X,y)\in B}[\nabla_\theta L(f_\theta(X + \boldsymbol{\delta}_{t-1} + \boldsymbol{\eta}_{t-1}), y)]$$

$$\boldsymbol{Z}[w_i] \leftarrow \boldsymbol{\eta}_t^i$$

Method

(2) Normalization Ball of Discrete Tokens

Since our core idea is to take the **discrete nature of texts** into consideration, we constrain perturbations with a tighter **token-level normalization ball** instead of naive **Frobenius normalization ball.**

We add a token-level scaling index: $n^i = \frac{||\delta^i||_F}{\max_j(||\delta^j||_F)}$

We can rewrite the normalization ball constraint as:

$$\boldsymbol{\delta}_t^i = n^i * (\boldsymbol{\delta}_{t-1}^i + \alpha g(\boldsymbol{\delta}_{t-1}^i)/||g(\boldsymbol{\delta}_{t-1}^i)||_F) \quad (3)$$

$$\boldsymbol{\delta}_t = \prod_{||\delta||_F \leq \epsilon}(\boldsymbol{\delta}_t) \quad (4)$$

Experiment

Model	RTE	QNLI	MRPC	CoLA	SST	STS-B	MNLI-m/mm	QQP
	Acc	Acc/f1	Mcc	Acc	P/S Corr	Acc	Acc/f1	Acc
BERT-BASE	66.4	90.5	88.9/84.8	52.1	93.5	87.1/85.8	84.6/83.4	71.2/89.2
FreeLB	70.1	91.8*	88.1/83.5	54.5	93.6	87.7/86.7	85.7/84.6	72.7/89.6
TextAT(ours)	**71.0**	**91.7**	**88.9/84.5**	**55.9**	**94.5**	86.8/85.7	85.2/**84.7**	**72.8**/89.5

Table 2: Evaluation results on the test set of GLUE benchmark. Results use the evaluation server on GLUE website. QNLI* in FreeLB is formed as pairwise ranking task.

Summary

- PGD generate multi-step adversarial examples to achieve high-level robustness, but only use the last gradient.
- FreeLB take the average gradient in K iterations.
- TextAT propose a global accumulated token perturbation and a token-aware Normalization Ball.

ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS

Kevin Clark
Stanford University
kevclark@cs.stanford.edu

Minh-Thang Luong
Google Brain
thangluong@google.com

Quoc V. Le
Google Brain
qvl@google.com

Christopher D. Manning
Stanford University & CIFAR Fellow
manning@cs.stanford.edu

Method

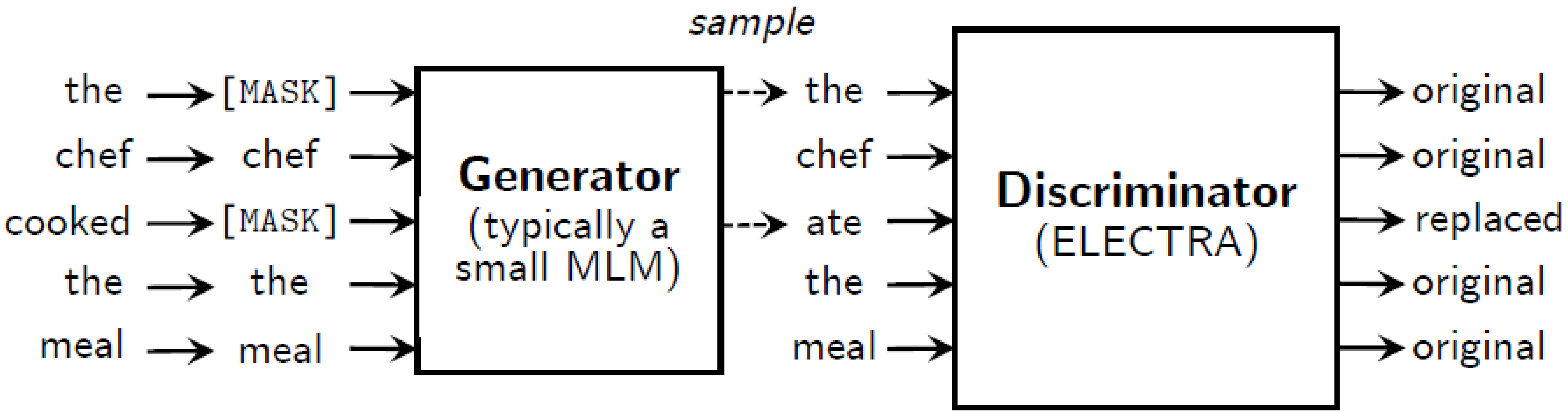

Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-r
1	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	9
2	Alibaba DAMO NLP	StructBERT + TAPT		90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	9
3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	9
4	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	9
5	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	9
6	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	9
7	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4	9
8	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	9
9	Huawei Noah's Ark Lab	NEZHA-Large		89.1	69.9	97.3	93.3/91.0	92.4/91.9	74.2/90.6	91.0	9
10	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	9

Revisiting Pre-Trained Models for Chinese Natural Language Processing

Yiming Cui[†‡], **Wanxiang Che**[†], **Ting Liu**[†], **Bing Qin**[†], **Shijin Wang**[‡§], **Guoping Hu**[‡]
[†]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[‡]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[§]iFLYTEK AI Research (Hebei), Langfang, China
[†]{ymcui,car,tliu,qinb}@ir.hit.edu.cn
[‡§]{ymcui,sjwang3,gphu}@iflytek.com

Introduction

Instead of masking with [MASK] token, which never appears in the fine-tuning stage, we propose to **use similar words for the masking purpose**. A similar word is obtained by using Synonyms toolkit (Wang and Hu, 2017), which is based on word2vec similarity calculations. If an N-gram is selected to mask, we will find similar words individually. In rare cases, when there is no similar word, we will degrade to use random word replacement.

	Chinese	English
Original Sentence	使用语言模型来预测下一个词的概率。	we use a language model to predict the probability of the next word.
+ CWS	使用 语言 **模型** 来 **预测** 下 一个 词 的 **概率** 。	-
+ BERT Tokenizer	使 用 语 言 **模 型** 来 **预 测** 下 一 个 词 的 **概** 率 。	we use a language **model** to **pre ##di ##ct** the **pro ##ba ##bility** of the next word .
Original Masking	使 用 语 言 [M] 型 来 [M] 测 下 一 个 词 的 概 率 。	we use a language [M] to [M] ##di ##ct the pro [M] ##bility of the next word .
+ WWM	使 用 语 言 [M] [M] 来 [M] [M] 下 一 个 词 的 概 率 。	we use a language [M] to [M] [M] [M] the [M] [M] [M] of the next word .
++ N-gram Masking	使 用 [M] [M] [M] [M] 来 [M] [M] 下 一 个 词 的 概 率 。	we use a [M] [M] to [M] [M] [M] the [M] [M] [M] [M] [M] next word .
+++ Mac Masking	使 用 **语 法 建 模** 来 **预 见** 下 一 个 词 的 **几** 率 。	we use a **text system** to **ca ##lc ##ulate** the **po ##si ##bility** of the next word .

Figure 1: Examples of the masking strategies. For clarity, we also include an English example.

Experiment

Sentence Pair Matching	XNLI		LCQMC		BQ Corpus	
	Dev	Test	Dev	Test	Dev	Test
BERT	77.8 (77.4)	77.8 (77.5)	89.4 (88.4)	86.9 (86.4)	86.0 (85.5)	84.8 (84.6)
ERNIE	79.7 (79.4)	78.6 (78.2)	89.8 (89.6)	87.2 (87.0)	86.3 (85.5)	85.0 (84.6)
BERT-wwm	79.0 (78.4)	78.2 (78.0)	89.4 (89.2)	87.0 (86.8)	86.1 (85.6)	85.2 (84.9)
BERT-wwm-ext	79.4 (78.6)	78.7 (78.3)	89.6 (89.2)	87.1 (86.6)	86.4 (85.5)	85.3 (84.8)
RoBERTa-wwm-ext	80.0 (79.2)	78.8 (78.3)	89.0 (88.7)	86.4 (86.1)	86.0 (85.4)	85.0 (84.6)
MacBERT-base	80.4 (79.5)	79.3 (78.9)	89.6 (89.3)	86.5 (86.3)	86.0 (85.4)	85.1 (84.7)
RoBERTa-wwm-ext-large	82.1 (81.3)	81.2 (80.6)	90.4 (90.0)	87.0 (86.8)	**86.3** (85.7)	**85.8** (84.9)
MacBERT-large	**82.4** (81.8)	**81.3** (80.6)	**90.6** (90.3)	**87.6** (87.1)	86.2 (85.7)	85.6 (85.0)

Table 6: Results on sentence pair matching tasks: XNLI, LCQMC, and BQ Corpus.

Summary

- Adversarial training **improves both robustness and generalization.**

- Many recent studies try to add adversarial training in the **pre-trained models**, and achieve better results.

- Token-level adversarial training (including token-level perturbations and token-level word replacement) can benefit the NLU task.

Thanks and QA