BERT-ATTACK: Adversarial Attack Against BERT Using BERT

EMNLP 2020
발표: Baek, Hyeongryeol

Abstract

1. Task: generating adversarial samples
2. Contribution
   a. effectively generate fluent and semantically-preserved adversarial samples
   b. a higher **attacking success rate** and **a lower perturb percentage** compared with previous attacking algorithms

Introduction

1. Neural Networks are vulnerable to adversarial samples
    a. imperceptible to human judges
    b. misleading the neural networks to incorrect predictions

Heuristic	Premise	Hypothesis	Label
Lexical overlap heuristic	The banker near the judge saw the actor.	The banker saw the actor.	E
	The lawyer was advised by the actor.	The actor advised the lawyer.	E
	The doctors visited the lawyer.	The lawyer visited the doctors.	N
	The judge by the actor stopped the banker.	The banker stopped the actor.	N
Subsequence heuristic	The artist and the student called the judge.	The student called the judge.	E
	Angry tourists helped the lawyer.	Tourists helped the lawyer.	E
	The judges heard the actors resigned.	The judges heard the actors.	N
	The senator near the lawyer danced.	The lawyer danced.	N
Constituent heuristic	Before the actor slept, the senator ran.	The actor slept.	E
	The lawyer knew that the judges shouted.	The judges shouted.	E
	If the actor slept, the judge saw the artist.	The actor slept.	N
	The lawyers resigned, or the artist slept.	The artist slept.	N

< Examples of misleading NN. NN is prone to label all examples as E (entailment). >

[1] T. McCoy, E. Pavlick, and T. Linzen, “Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, (Florence, Italy), pp. 3428–3448, Association for Computational Linguistics, July 2019.

Introduction

1. Neural Networks are vulnerable to adversarial samples
    a. imperceptible to human judges
    b. misleading the neural networks to incorrect predictions

Dataset					Label
MNLI	Ori	Some rooms have balconies .	Hypothesis	All of the rooms have balconies off of them .	Contradiction
	Adv	Many rooms have balconies .	Hypothesis	All of the rooms have balconies off of them .	Neutral
IMDB	Ori	it is hard for a lover of the novel northanger abbey to sit through this bbc adaptation and to keep from throwing objects at the tv screen... why are so many facts concerning the tilney family and mrs . tilney ' s death altered unnecessarily ? to make the story more ' horrible ? '			Negative
	Adv	it is hard for a lover of the novel northanger abbey to sit through this bbc adaptation and to keep from throwing objects at the tv screen... why are so many facts concerning the tilney family and mrs . tilney ' s death altered unnecessarily ? to make the plot more ' horrible ? '			Positive
IMDB	Ori	i first seen this movie in the early 80s .. it really had nice picture quality too . anyways , i 'm glad i found this movie again ... the part i loved best was when he hijacked the car from this poor guy... this is a movie i could watch over and over again . i highly recommend it .			Positive
	Adv	i first seen this movie in the early 80s .. it really had nice picture quality too . anyways , i 'm glad i found this movie again ... the part i loved best was when he hijacked the car from this poor guy... this is a movie i could watch over and over again . i inordinately recommend it .			Negative

Table 10: Some generated adversarial samples. Origin label is the correct prediction while label is adverse prediction. Only red color parts are perturbed. We only attack premises in MNLI task. Text in FAKE dataset and IMDB dataset is cut to fit in the table. Original text contains more than 200 words.

< Adversarial samples. Label should be invariant. >

Introduction

2. Key to generating adversarial samples
    a. imperceptible to human judges yet **misleading to neural models**
    b. fluent in grammar and semantically **consistent with original inputs**
3. Proposed methods
    a. **finding the vulnerable words** in one given input sequence for the target model
    b. applying BERT in **a semantic-preserving way** to generate substitutes for the vulnerable words

Methods: Finding vulnerable words

1. Finding important words
   a. comparing the prediction probability b/w S with and without token w_i
   b. taking \epsilon percent of the most important words

Let $S = [w_0, \cdots, w_i \cdots]$ denote the input sentence, and $o_y(S)$ denote the logit output by the target model for correct label y, the importance score I_{w_i} is defined as

$$I_{w_i} = o_y(S) - o_y(S_{\backslash w_i}), \tag{1}$$

where $S_{\backslash w_i} = [w_0, \cdots, w_{i-1}, [\mathtt{MASK}], w_{i+1}, \cdots]$ is the sentence after replacing w_i with $[\mathtt{MASK}]$.

Methods: Word replacement

2. Word replacements
    a. feeding original sequence to get candidates of token w_j
        i. not using [MASK] token: more semantic-consistent
    b. filtering stop words and antonyms using synonym dictionaries
        i. preserving original labels

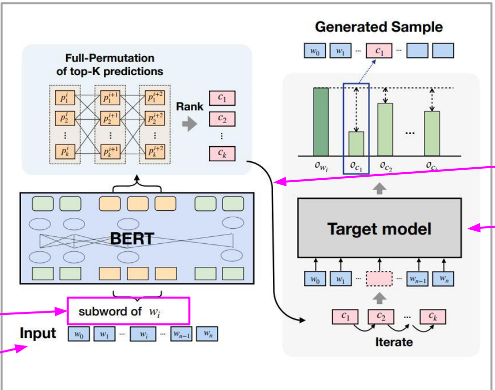

Figure 1: One step of our replacement strategy.

Filtering stop words & antonyms

Black box inside the target model.
(Querying is the only way to attack it)

Important words

Original input without [MASK]

Methods: Pseudo-code

Algorithm 1 BERT-Attack

procedure WORD IMPORTANCE RANKING
  $S = [w_0, w_1, \cdots]$ // input: tokenized sentence
  $Y \leftarrow$ gold-label
  for w_i in S **do**
    calculate importance score I_{w_i} using Eq. 1
  select word list $L = [w_{top-1}, w_{top-2}, \cdots]$
  // sort S using I_{w_i} in descending order and collect $top - K$ words

Calculating word importance

procedure REPLACEMENT USING BERT
  $H = [h_0, \cdots, h_n]$ // sub-word tokenized sequence of S
  generate top-K candidates for all sub-words using BERT and get $P^{\in n \times K}$
  for w_j in L **do**
    if w_j is a whole word **then**
      get candidate $C = Filter(P^j)$
      replace word w_j
    else
      get candidate C using PPL ranking and Filter
      replace sub-words $[h_j, \cdots, h_{j+t}]$

Iterating tokens according to importance scores
And getting candidates

    Find Possible Adversarial Sample
    for c_k in C **do**
      $S' = [w_0, \cdots, w_{j-1}, c_k, \cdots]$ // attempt
      if argmax$(o_y(S'))! = Y$ **then**
        return $S^{adv} = S'$ // success attack
      else
        if $o_y(S') < o_y(S^{adv})$ **then**
          $S^{adv} = [w_0, \cdots, w_{j-1}, c, \cdots]$ // do one perturbation

Replacing a token with a candidate one by one
And calculating a prediction score

  return None

Metrics

- **After-attack-accuracy (lower is better)**

  : the acc. scores on the adversarial samples

- **Query number per sample (lower is better)**

  : the total times of sending a text to the target model to get the prediction score

- **Semantic consistency (higher is better)**

  : sim. score b/w the adversarial sample and the original sequence using Universal Sentence Encoder

- **Perturbed percentage (lower is better)**

  : the ratio of the number of perturbed words to the text length

Experiments

- Model comparison across metrics
- Human evaluation of adversarial samples w.r.t grammar and fluency
- Transferability: examining if adversarial samples curated based on one model can also fool another
- Adversarial training: finetuning with adversarial samples

Experiments: Model comparison across metrics

Dataset	Method	Original Acc	Attacked Acc	Perturb %	Query Number	Avg Len	Semantic Sim
Fake	BERT-Attack(ours)	97.8	**15.5**	**1.1**	**1558**	885	**0.81**
	TextFooler(Jin et al., 2019)		19.3	11.7	4403		0.76
	GA(Alzantot et al., 2018)		58.3	1.1	28508		-
Yelp	BERT-Attack(ours)	95.6	**5.1**	**4.1**	**273**	157	**0.77**
	TextFooler		6.6	12.8	743		0.74
	GA		31.0	10.1	6137		-
IMDB	BERT-Attack(ours)	90.9	**11.4**	**4.4**	**454**	215	**0.86**
	TextFooler		13.6	6.1	1134		**0.86**
	GA		45.7	4.9	6493		-
AG	BERT-Attack(ours)	94.2	**10.6**	**15.4**	**213**	43	**0.63**
	TextFooler		12.5	22.0	357		0.57
	GA		51	16.9	3495		-
SNLI	BERT-Attack(ours)	89.4(H/P)	7.4/**16.1**	**12.4/9.3**	**16/30**	8/18	0.40/**0.55**
	TextFooler		**4.0**/20.8	18.5/33.4	60/142		**0.45**/0.54
	GA		14.7/-	20.8/-	613/-		-
MNLI matched	BERT-Attack(ours)	85.1(H/P)	**7.9/11.9**	**8.8/7.9**	**19/44**	11/21	0.55/**0.68**
	TextFooler		9.6/25.3	15.2/26.5	78/152		**0.57**/0.65
	GA		21.8/-	18.2/-	692/-		-
MNLI mismatched	BERT-Attack(ours)	82.1(H/P)	**7/13.7**	**8.0/7.1**	**24/43**	12/22	0.53/**0.69**
	TextFooler		8.3/22.9	14.6/24.7	86/162		**0.58**/0.65
	GA		20.9/-	19.0/-	737/-		-

Table 1: Results of attacking against various fine-tuned BERT models. TextFooler is the state-of-the-art baseline. For MNLI task, we attack the hypothesis(H) or premises(P) separately.

Purturbed percentage and query number are significantly low

Experiments: Human evaluation & Transferability

- More difficult; semantics-preserved
- More transferable in NLI task

Dataset		Accuracy	Semantic	Grammar
MNLI	Original	0.90	3.9	4.0
	Adversarial	0.70	3.7	3.6
IMDB	Original	0.91	4.1	3.9
	Adversarial	0.85	3.9	3.7

Table 2: Human-Evaluation Results.

Dataset	Model	LSTM	BERT-base	BERT-large
IMDB	Word-LSTM	-	0.78	0.75
	BERT-base	0.83	-	0.71
	BERT-large	0.87	0.86	-
Dataset	**Model**	ESIM	BERT-base	BERT-large
MNLI	ESIM	-	0.59	0.60
	BERT-base	0.60	-	0.45
	BERT-large	0.59	0.43	-

Table 6: Transferability analysis using attacked accuracy as the evaluation metric. The column is the target model used in attack, and the row is the tested model.

Experiments: Adversarial training

- The models becomes more robust to adversarial-attacks.

Dataset	Method	Ori Acc	Atk Acc	Perturb %
MNLI matched	BERT-Atk	85.1	7.9	8.8
	+Adv Train	84.6	23.1	10.5

Table 5: Adversarial training results.

Lesson learned

- Adversarial samples make the model more robust to heuristic methods.
- Transferability of adversarial samples is poor (can generate target model specific samples).
- It is harder to attack pretrained model, i.e., requiring more query times and perturbed words.