DeBERTa: Decoding-enhanced BERT with Disentangled Attention

Radek Bartyzal

GLAMI AI

2. 3. 2021

Motivation

Paper is by Microsoft Research from 2020/2021.

- improved BERT
- models available through Hugging Face
- new SOTA in Language Understanding and Lang. Generation downstream tasks as of January 6, 2021
- faster pre-training = less steps, but less time?

BERT

Transformer +

DeBERTa changes to BERT

disentangled attention mechanism: treat content and position information separately

relative positional encodings passed to each layer

enhanced mask decoder: provide absolute positional encodings right before the last softmax layer decoding the masked words

Scale invariant Fine Tuning: perturbe normalized word embeddings = adversarial training = better generalization

Classic attention mechanism

$$\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{q}}, \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{k}}, \boldsymbol{V} = \boldsymbol{H}\boldsymbol{W}_{\boldsymbol{v}}, \boldsymbol{A} = \frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{d}}$$

$$\boldsymbol{H}_{\boldsymbol{o}} = \mathrm{softmax}(\boldsymbol{A})\boldsymbol{V}$$

- before the first layer word vectors are summed with positional vectors
- these combined = entangled content vectors are then passed through Transformer layers
- each layer's attention creates Key, Query and Value vectors from these combined vectors

Disentangled attention mechanism

disentangled attention = at each layer, create separate Key and Query vectors = K^r and Q^r from relative positional vectors

$$\boldsymbol{Q_c} = \boldsymbol{HW_{q,c}}, \boldsymbol{K_c} = \boldsymbol{HW_{k,c}}, \boldsymbol{V_c} = \boldsymbol{HW_{v,c}}, \boldsymbol{Q_r} = \boldsymbol{PW_{q,r}}, \boldsymbol{K_r} = \boldsymbol{PW_{k,r}}$$

$$\tilde{A}_{i,j} = \underbrace{\boldsymbol{Q_i^c K_j^c}^\intercal}_{\text{(a) content-to-content}} + \underbrace{\boldsymbol{Q_i^c K_{\delta(i,j)}^r}^\intercal}_{\text{(b) content-to-position}} + \underbrace{\boldsymbol{K_j^c Q_{\delta(j,i)}^r}^\intercal}_{\text{(c) position-to-content}}$$

$$\boldsymbol{H_o} = \mathrm{softmax}\left(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}}\right)\boldsymbol{V_c}$$

- there is still only one Value vector created from content vector
- content vectors are the ones transformed by each layer, positional vectors stay the same
- final attention matrix is sum of attention matrices of possible combinations of Pos. and Content Q/Ks

Disentangled attention mechanism

$P_{i,j}$ = encoding of a position of i relative to j

$$A_{i,j} = \{\boldsymbol{H_i}, \boldsymbol{P_{i|j}}\} \times \{\boldsymbol{H_j}, \boldsymbol{P_{j|i}}\}^\intercal$$
$$= \boldsymbol{H_i H_j^\intercal} + \boldsymbol{H_i P_{j|i}^\intercal} + \boldsymbol{P_{i|j} H_j^\intercal} + \boldsymbol{P_{i|j} P_{j|i}^\intercal}$$

Possible attention matrices = combinations of P, C:

- C to C = classic attention
- C to P = what relative positions are important to this word
- P to C = what words are important to these relative positions
- P to P = what relative position are important to this relative position = does not make sense = not used

Disentangled attention: Efficient implementation

- an input sequence of length N
- requires a space complexity of $O(N^2d)$ to store the relative position embedding for each token
- set the maximum relative distance k to 512 for pre-training
- embeddings of all possible relative positions are always a subset of $K_r \in R^{2k \times d} \implies$ we can reuse K_r in the attention calculation for all the queries
- we do not need to allocate memory to store a relative position embedding for each query and thus reduce the space complexity to $O(kd)$ (probably $O(2kd)$) (for storing K_r and Q_r)

Relative positional encodings

Denote k as the maximum relative distance, $\delta(i,j) \in [0, 2k)$ as the relative distance from token i to token j, which is defined as:

$$\delta(i,j) = \begin{cases} 0 & \text{for} \quad i - j \leqslant -k \\ 2k - 1 & \text{for} \quad i - j \geqslant k \\ i - j + k & \text{others.} \end{cases} \tag{3}$$

- the positional vectors are relative = [..., -2, -1, 0, 1, 2, ...]
- the positional encodings are shared between layers
- max relative distance = $k = 512$, after that just pad: [-2, -2, -1, 0]

Scale invariant Fine Tuning = SiFT

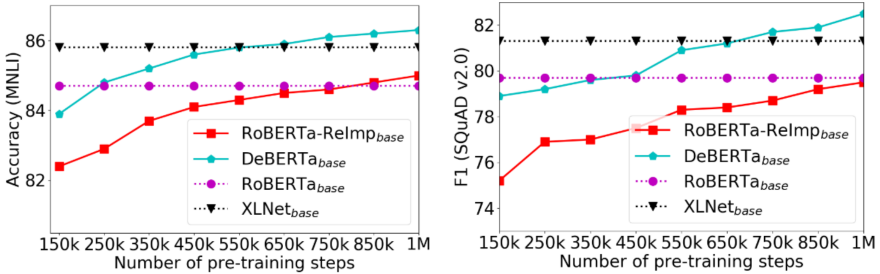
- The model is regularized: model produces the same output distribution as it produces on an adversarial perturbation of that example.
- perturbation is applied to the word embedding instead of the word sequence
- However, the value ranges (norms) of the embedding vectors vary among different words and models.
- variance gets larger for bigger models with billions of parameters, leading to some instability of adversarial training.
- SiFT improves the training stability by applying the perturbations to the normalized word embeddings.
- Specifically, when fine-tuning DeBERTa to a downstream NLP task in our experiments, SiFT first normalizes the word embedding vectors into stochastic vectors, and then applies the perturbation to the normalized embedding vectors.
- We find that the normalization substantially improves the performance of the fine-tuned models.

Ablation study: everything helps

Model	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	SQuAD v2.0 F1/EM	RACE Acc
BERT_{base} Devlin et al. (2019)	84.3/84.7	88.5/81.0	76.3/73.7	65.0
RoBERTa_{base} Liu et al. (2019c)	84.7/-	90.6/-	79.7/-	65.6
XLNet_{base} Yang et al. (2019)	85.8/85.4	-/-	81.3/78.5	66.7
$\text{RoBERTa-ReImp}_{base}$	84.9/85.1	91.1/84.8	79.5/76.0	66.8
DeBERTa_{base}	**86.3/86.2**	**92.1/86.1**	**82.5/79.3**	**71.7**
-EMD	86.1/86.1	91.8/85.8	81.3/78.0	70.3
-C2P	85.9/85.7	91.6/85.8	81.3/78.3	69.3
-P2C	86.0/85.8	91.7/85.7	80.8/77.6	69.6
-(EMD+C2P)	85.8/85.9	91.5/85.3	80.3/77.2	68.1
-(EMD+P2C)	85.8/85.8	91.3/85.1	80.2/77.1	68.5

Table 5: Ablation study of the DeBERTa base model.

Pre-training needs less steps than RoBERTa



(a) Results on MNLI development (b) Results on SQuAD v2.0 development

Figure 1: Pre-training performance curve between DeBERTa and its counterparts on the MNLI and SQuAD v2.0 development set.

Results: GLUE

Model	CoLA Mcc	QQP F1/Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT_{large}	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa_{large}	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet_{large}	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
$\text{ELECTRA}_{\text{large}}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa_{large}	70.5	92.3	**91.1/91.1**	96.8	92.8	**95.3**	88.3	**91.9**	**90.00**

Table 1: Comparison results on the GLUE development set.

We use 6 DGX-2 machines (96 V100 GPUs) to train the models. A single model trained with 2K batch size and 1M steps takes about 20 days.

Results: Comparison with similar sized models

Model	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	SQuAD v2.0 F1/EM	RACE Acc	ReCoRD F1/EM	SWAG Acc	NER F1
BERT_{large}	86.6/-	90.9/84.1	81.8/79.0	72.0	-	86.6	92.8
ALBERT_{large}	86.5/-	91.8/85.2	84.9/81.8	75.2	-	-	-
RoBERTa_{large}	90.2/90.2	94.6/88.9	89.4/86.5	83.2	90.6/90.0	89.9	93.4
XLNet_{large}	90.8/90.8	95.1/89.7	90.6/87.9	85.4	-	-	-
Megatron_{336M}	89.7/90.0	94.2/88.0	88.1/84.8	83.0	-	-	-
DeBERTa_{large}	**91.1/91.1**	**95.5/90.1**	**90.7/88.0**	**86.8**	**91.4/91.0**	**90.8**	**93.8**
$\text{ALBERT}_{xxlarge}$	90.8/-	94.8/89.3	90.2/87.4	86.5	-	-	-
$\text{Megatron}_{1.3B}$	90.9/91.0	94.9/89.1	90.2/87.1	87.3	-	-	-
$\text{Megatron}_{3.9B}$	91.4/91.4	95.5/90.0	91.2/88.5	89.5	-	-	-

Table 2: Results on MNLI in/out-domain, SQuAD v1.1, SQuAD v2.0, RACE, ReCoRD, SWAG, CoNLL 2003 NER development set. Note that missing results in literature are signified by "-".

Results: Scale up to 1.5B parameters

Model	BoolQ Acc	CB F1/Acc	COPA Acc	MultiRC F1a/EM	ReCoRD F1/EM	RTE Acc	WiC Acc	WSC Acc	Average Score
$\text{RoBERTa}_{\text{large}}$	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	84.6
NEXHA-Plus	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	86.7
T5_{11B}	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	89.3
T5_{11B}+Meena	**91.3**	**95.8/97.6**	97.4	88.3/63.0	94.2/93.5	92.7	**77.9**	95.9	90.2
Human	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	89.8
$\text{DeBERTa}_{1.5B}$	90.4	94.9/97.2	96.8	**88.2/63.7**	**94.5/94.1**	**93.2**	76.4	**95.9**	89.9
$\text{DeBERTa}_{Ensemble}$	90.4	95.7/97.6	**98.4**	88.2/63.7	94.5/94.1	93.2	77.5	95.9	**90.3**

Table 6: SuperGLUE test set results scored using the SuperGLUE evaluation server. All the results are obtained from https://super.gluebenchmark.com on January 6, 2021.

- we share the projection matrices of relative position embedding $W_{k,r}, W_q, r$ with $W_{k,c}, W_{q,c}$, respectively, in all attention layers = not treating the content/positional information differently?

Conclusion

The single 1.5B-parameter DeBERTa model substantially outperforms T5 with 11 billion parameters on the SuperGLUE benchmark by 0.6%(89.3% vs. 89.9%)

Sources

1.He, Pengcheng, et al. "Deberta: Decoding-enhanced bert with disentangled attention." arXiv preprint arXiv:2006.03654 (2020). https://arxiv.org/abs/2006.03654