RoBERTa A Robustly Optimized BERT Pretraining Approach

DeBERTa Decoding–Enhanced BERT w/ Disentangles Attention

Sunkyung Park

Contents (RoBERTa)

Background：BERT Transformer−based LM

Setup

INPUT

$segment\ 1{:}\ x_1, \dots, x_N$
$segment\ 2{:}\ y_1, \dots, y_M$

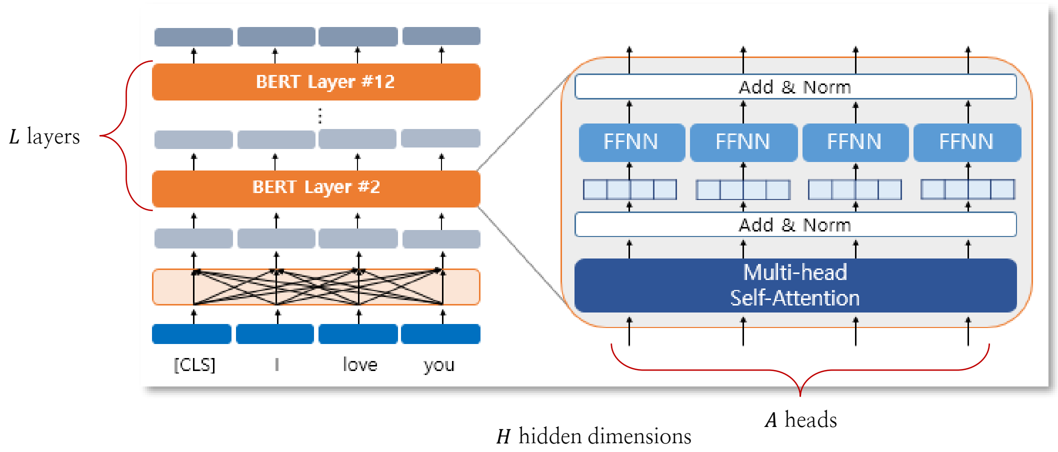
\# of Segment ≥ 1 sentence ⟩⟩⟩

$M + N < T$ maximum sequence length

MODEL

- Pretrained w/ a large unlabeled text corpus
- Finetuned w/ end−task labeled data

Architecture

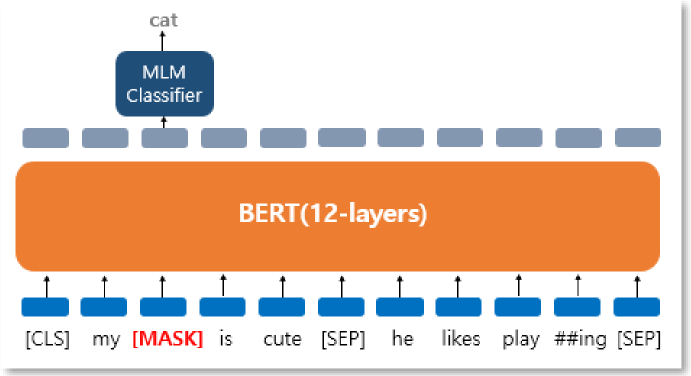


Background：BERT Transformer-based LM

Training Objectives

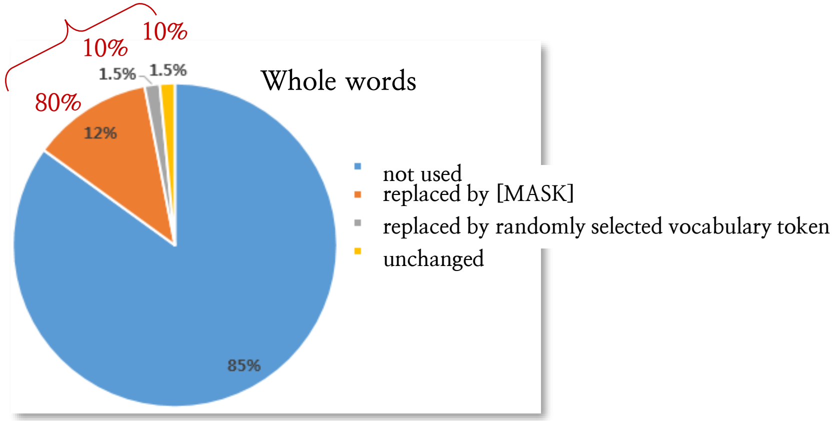
MSM Masked LM

Example

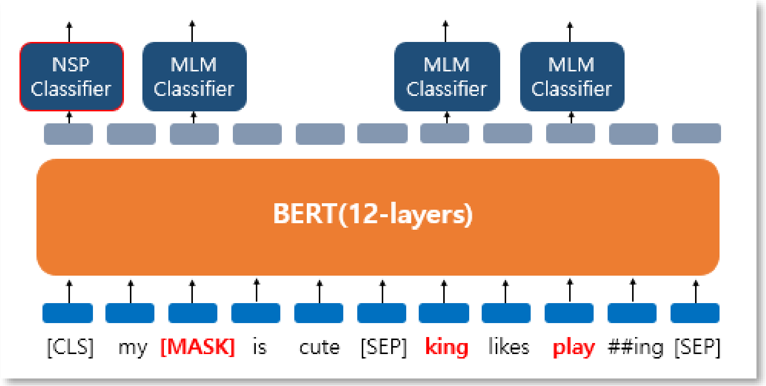


[original] My dog is cute, he likes playing.

15% of the input tokens



NSP Next Sentence Prediction



- [CLS]
  - Binary classification
  - Whether two segments follow each other in the original text
    - Positive and negatives examples are sampled w/ equal prob
    - Positive
      - Case: Consecutive sentences from the text corpus
      - Label: IsNextSentence
    - Negative
      - Case: Pair segments from different documents
      - Label: NotNextSentence

》 Reasoning downstream task

Background : BERT Transformer–based LM

Optimization	Optimizer	• Adam
		  • β_1 : 0.9
		  • β_2 : 0.999
		  • ϵ : $1e-6$

	L_2 weight decay	• 0.01

	Learning rate	• ≤10,000 steps
		  • warm up to a peak value of 1e−4
		• >10,000 steps
		  • linearly decayed
		• Learning rate warmup : (a small learning rate $_{\text{beginning}}$ → warm–up → training $_{\text{stable}}$ → return to small one)

	Dropout Activation function Steps [S] Batch size [B]	• 0.1 on all layers and attention weights • GELU • 1,000,000 updates • 256 sequences of maximum length 512 tokens [T]
Data	BERT is trained on… • BOOKCORPUS • English WIKIPEDIA 16GB of uncompressed text	

Experimental Setup for replication study of BERT

- Implementation
  - Make BERT in FAIRSEQ sequence modeling toolkit(tool set) providing reference implementation
  - Hyperparameters
    - Learning rate
      - The original BERT optimization hyperparameters except for the peak learning rate and # of warmup steps
      - (original) BERT
        - $\leq$ 10,000 steps
          - warm up to a peak value of 1e−4
        - $>$ 10,000 steps
          - linearly decayed
    - Optimizer
      - Find out training to be very sensitive to $\boldsymbol{\epsilon}$ the Adam epsilon term and setting $\boldsymbol{\beta_2 = 0.98}$ to improve stability when training w/ large batch size
      - (original) BERT
        - Adam ($\boldsymbol{\beta_1 = 0.9}$, $\boldsymbol{\beta_2 = 0.999, \epsilon = 1e-6}$)

Experimental Setup for replication study of BERT

- Implementation
  - Sequence length
    - Pretrain w/ sequences of at most $T = 512$ tokens
    - Train only w/ full−length sequences (not intentionally shortening)
  - Setting
    - 8 × 32GB NVIDIA V100 GPUs interconnected by InfiniBand computer networking communications standard
      - Train w/ mixed precision floating point arithmetic NVIDIA DGX−1 machines

Experimental Setup for replication study of BERT

- Data
  - BERT-style pretraining relies on large quantities of text
    - Dataset larger and more diverse than BERT results in improved end-task performance
  - Five English-language corpora of varying sizes and domains
    - 160 G of uncompressed text as much as possible
    - (original) BERT
      - BOOKCORPUS
      - English WIKIPEDIA 16GB of uncompressed text

Experimental Setup for replication study of BERT

- Data
    - Five English-language corpora of varying sizes and domains
        - Details

BOOKCORPUS & English WIKIPEDIA	CC-NEWS	OPENWEBTEXT	STORIES
• 16 GB • Original data of BERT	• 76 GB • English portion of the CommonCrawl News • 63 million English news article between Sep 2016 and Feb 2019	• 38 GB • Open-source recreation of the WebText corpus • Web context extracted from URLs shared on Reddit w/ at least 3 upvotes.	• 31 GB • Containing a subset of CommonCrawl data filtered to match the story like of Winograd schemas • Winograd schemas : • It tests whether the machine understands the language

- The trophy would not fit in the brown suitcase because it was too big.

What was too big?

Answer 0: the trophy

Answer 1: the suitcase

Select the answer

Experimental Setup for replication study of BERT

- Evaluation
    - Evaluate pretrained models on downstream tasks using the following three benchmarks

GLUE The General Language Understanding Evaluation

- A collection of 9 datasets
- Tasks
    - Single-sentence classification
        - Finding spam
        - Categorizing docs
        - Sentiment
    - Sentence-pair classification
        - Relation between 2 sentences
            - Paraphrase detection
            - Entailment
            - Contradiction
            - Neutral
- Provides
    - training and development data
    - a submission server and leader board
    - evaluation on a private held-out test data
- RoBERTa
    - Reports results on the development sets after finetuning the pretrained models

SQuAD The Stanford Question Answering Dataset

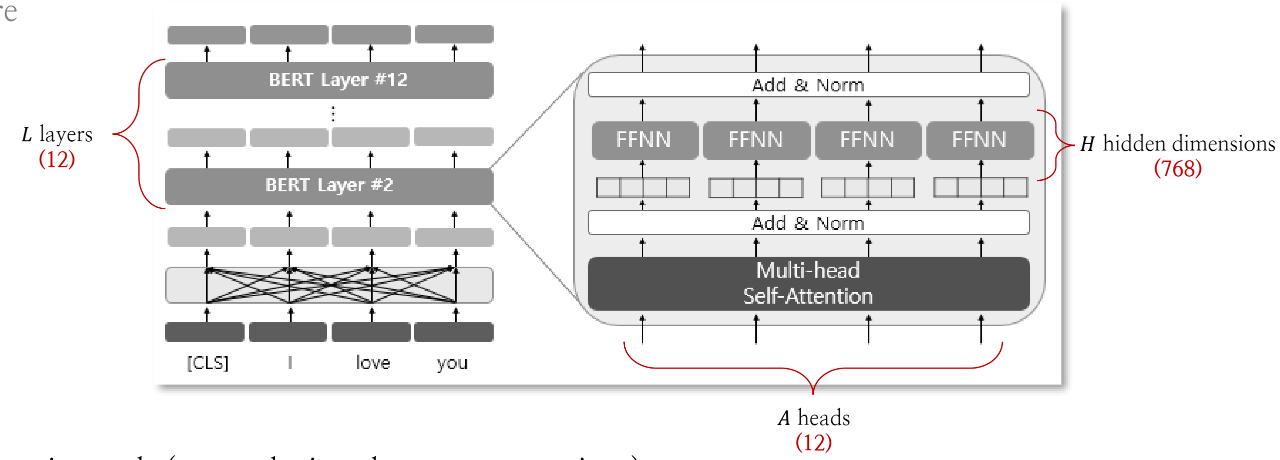
- A paragraph of context and a question
- Task
    - Answer the question by extracting the relevant span from the context.
- RoBERTa
    - Evaluates on two versions of SQuAD V1.1 and V2.0
    - V1.1 : context always contains an answer
        - The same span prediction method
    - V2.0 : some Qs are not answered in the context.
        - (+) binary classification whether the Q is answerable
    - Train jointly by summing the classification and span loss terms
        - Predicts span indices on pairs answerable

RACE The ReAding Comprehension from Examinations

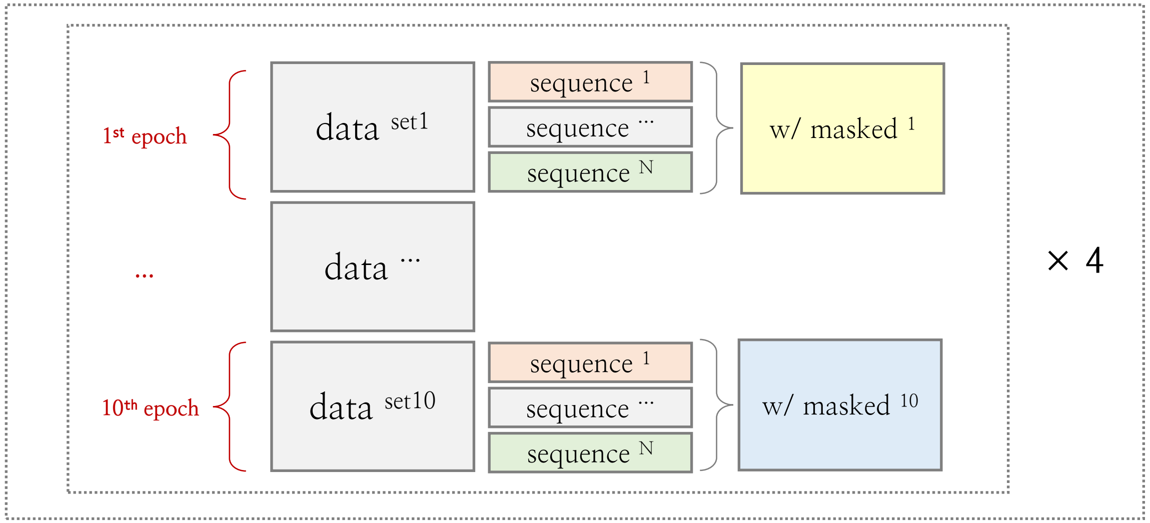
- Collected from English exams in China designed for middle and high school students
- Large-scale reading comprehension dataset
    - ≥ 28,000 passages
        - each w/ multiple questions
    - 100,000 Qs.
        - 1 correct answer w/ 4 options
- Longer context than others
- Large proportion of questions needed for reasoning
- Task
    - Select the correct answer

Training Procedure Analysis

- Choices for successfully pretraining BERT models : Static vs. Dynamic Masking
  - (Fixed) Model architecture



  - (Original BERT) a single static mask (once during data preprocessing)
  - (RoBERTa static mask) training data duplicated 10 times each sequence masked in 10 different ways (over 40 epochs) to avoid using the same mask in every epoch



- Each training sequence seen w/ the same mask 4 times during training.
- Dynamic masking Every pattern is different
  - RoBERETa dynamic mask
    - Make masking pattern every time the sequence is fed to the model.

Training Procedure Analysis

- Choices for successfully pretraining BERT models
  - (RoBERTa) Compare RoBERTa masking strategies w/ $\text{BERT}_{\text{BASE}}$

F1 Score

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

- Compares the official $\text{BERT}_{\text{BASE}}$ results (■) to our reimplementation w/ static (■) or dynamic masking (■).
- (■) our static masking ≒ (■) $\text{BERT}_{\text{BASE}}$ results
- (■) our dynamic masking > (■) $\text{BERT}_{\text{BASE}}$ results and (■) static masking
- (in this paper) Use dynamic masking

Training Procedure Analysis

- Choices for successfully pretraining BERT models : Model Input Format and Next Sentence Prediction
    - Original BERT
        - Trained to predict whether the observed document segments come from the same or distinct documents via an auxiliary NST loss.
        - NSP loss is an important factor in training model
    - Discrepancy between papers published later
        - (+) NSP loss helpful for performance on ANLI, MNLI, and SQuAD 1.1
        - (−) Recent works questions the necessity of the NSP loss
    - (RoBERTa) Compare alternative training formats

Training Procedure Analysis

- Choices for successfully pretraining BERT models : Model Input Format and Next Sentence Prediction

  - (RoBERTa) Compare alternative training formats

SEGMENT-PAIR + NSP	SENTENCE-PAIR + NSP	FULL-SENTENCE	DOC-SENTENCES

• Original input format in BERT
• Each input has a pair of segments
  • each w/ multiple sentence
  • total length $<_{\text{must be}}$ 512 tokens
	• Each input has a pair of sentences
	  • Sampled from
	    • contiguous adjacent portion of one document
	    • separate documents
	  • total length $<_{\text{generally}}$ 512 tokens
	    • to make total number of tokens like SEGMENT-PAIR+NSP
	    • BATCH SIZE ↑ (retrain)
		• Each input packed w/ full sentences
		  • Sampled from one or more docs
		  • Total length $<_{\text{must be}}$ 512 tokens
		• Inputs cross docs' boundaries
		  • Reach the end of one doc
		  • Begin sampling from the next doc
		  • Add a [SEP] separator token between docs
		  • Remove NSP loss
			• Unlike FULL-SENTENCES
			  • Inputs don't cross docs' boundaries
			    • Reach the end of one doc
			    • Inputs $<_{\text{likely}}$ 512 tokens
			• BATCH SIZE ↑ like FULL-SENTENCES
			• Remove NSP loss

(+) NSP loss	(+) NSP loss (+) BATCH SIZE tuning	(−) NSP loss (+) BATCH SIZE tuning	(−) NSP loss (+) BATCH SIZE tuning

Training Procedure Analysis

- Choices for successfully pretraining BERT models : Model Input Format and Next Sentence Prediction

  - (RoBERTa) Compare alternative training formats

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

- (■) individual sentences SENTENCE-PAIR → performance on downstream tasks ↓
  - impossible to learn long-range dependencies

- (■) outperforms $\text{BERT}_{\text{BASE}}$
  - without the NSP loss → performance on downstream tasks ↑
  - $\text{BERT}_{\text{BASE}}$
    - (+) SEGMENT-PAIR input format
    - (−) NSP loss
  - seq from single doc DOC-SENTENCES > seq from multiple docs FULL-SENTENCES

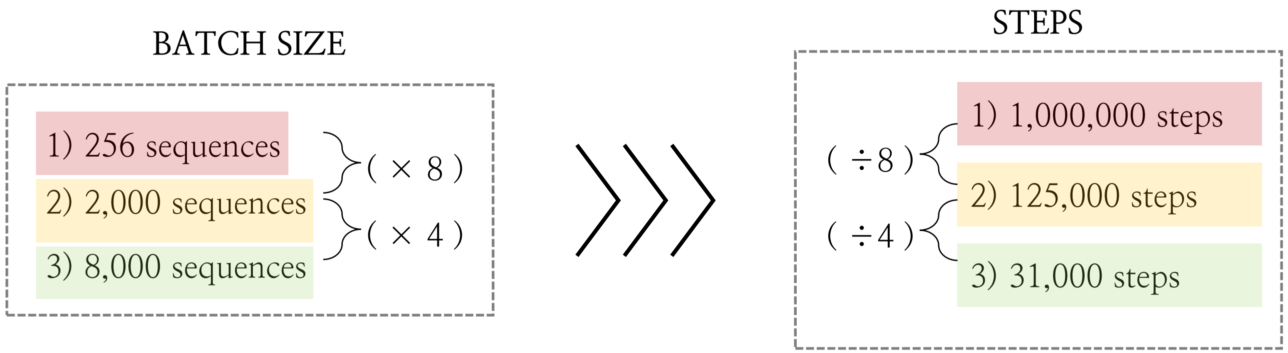
- DOC-SENTENCES w/ variable batch sizes

→ (in this paper) Use FULL-SENTENCES easier comparison w/ related works

Training Procedure Analysis

- Choices for successfully pretraining BERT models : Training w/ large batches
  - (■) $\text{BERT}_{\text{BASE}}$ trained for 1M steps w/ a batch size of 256 seq.

    - $STEPS^{\text{\# of times to update } w \text{ and } b} = (\ SAMPLES^{FIXED} \ * \ EPOCHS^{SAME}\ )/BATCH\ SIZE$
    - in gradient accumulation, it is same as (■), (■) and (■) → The same computational cost



Training Procedure Analysis

- Choices for successfully pretraining BERT models : Text Encoding
  - BPE Character-level Byte-Pair Encoding
    - A hybrid between character and word-level representations and it allows handling the large vocabularies
    - Relies on sub-word units extracted by statistical analysis → 10K-100K sub-word units
    - Uses Unicode characters as the base sub-word units
    - Encodes any input text without introducing any "unknown" tokens. → Robust on "OOV"
  - Original BERT
    - Preprocesses the input w/ heuristic tokenization rules
    - Makes a character-level BPE vocabulary of size 30K sub-word units
  - This paper
    - Without any additional preprocessing or tokenization of the input
    - Trains BERT w/ BBPE Byte-level Byte-Pair Encoding
      - a larger byte-level BPE vocabulary of size 50K sub-word units
      - Adds additional parameters for BERTs

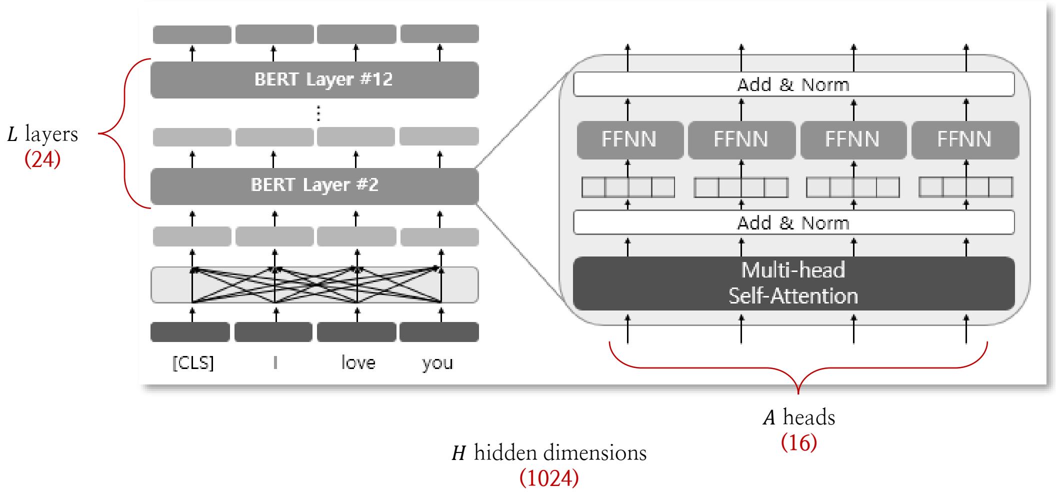
RoBERTa Robustly optimized BERT approach

- It is trained w/
  - Dynamic masking
  - FULL-SENTENCES without NSP loss
  - Large mini-batches
  - A larger than BPE BBPE

- Emphasizes two important factors
  - Data used for pretraining
  - Num of passes through the data
  - XLNet FOR EXAMPLE
    - $\text{SAMPLES} = \text{SAMPLES}_{\text{BERT}} \times 10$
    - $\text{STEP} = \text{STEP}_{\text{BERT}}/2$
    - $\text{BATCH SIZE} = \text{BATCH SIZE}_{\text{BERT}} \times 8$

See $\text{SEQUENCES}_{\text{BERT}} \times 4$ in a batch in pretraining

RoBERTa Robustly optimized BERT approach



- Architecture under the BERT Large
- 100K steps over a BOOKCORPUS + WIKIPEDIA dataset BERT
- 1024 V100 GPUs for one day

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$BERT_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$XLNet_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $BERT_{LARGE}$. Results for $BERT_{LARGE}$ and $XLNet_{LARGE}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

- (■) Results on training RoBERTa on the combined data w/ additional data
  - CC-NEWS, OPENWEBTEXT, STORIES
  - 100K steps
  - Further improvements on all downstream tasks
    - Importance of data size and diversity
- (■) Pretraining steps from 100K to 300K, and the 500 K
  - Significant gains in downstream task performance
  - The 300K and 500K step models > XLNet LARGE
    - Overfitting doesn't appear and Additional training benefits appear

RoBERTa Robustly optimized BERT approach

- (Downstream task) GLUE The General Language Understanding Evaluation Results

Median development set results for each task over five random initializations

GLUE tasks

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$BERT_{LARGE}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$XLNet_{LARGE}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $BERT_{LARGE}$ and $XLNet_{LARGE}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

- Two finetuning settings on RoBERTa
  - (■) single-task, dev
    - A limited hyperparameter sweep for each task
      - Batch size ∈ {16, 32}
      - Lr ∈ {1e−5, 2e−5, 3e−5}
        - Linear warmup for the first 6% steps and followed by linear decay to 0.
    - Finetunes RoBERTa using only the training data for each task
    - Performs early stopping on an evaluation metric on each task on the dev set.
  - (■) ensembles 5 and 7 models, test
    - Submissions on GLUE leaderboard are with multi−task finetuning but our submission depends only on single−task finetuning.

RoBERTa Robustly optimized BERT approach

- (Downstream task) GLUE The General Language Understanding Evaluation Results

GLUE tasks

Median development set results for each task over five random initializations

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

- Task-specific modifications on RoBERTa
  - (■) QNLI
    - Submission on leaderboard
      - A pairwise ranking formulation
        - (question, candidate)
          - Candidates are mined from the training set and compared to one another
          - A single (question, candidate) is classified as positive
    - Results on paper
      - Based on a pure classification approach for direct comparison w/ BERT
  - (■) WNLI
    - Data from Super-GLUE Wang et al.
      - Span of the query pronoun and referent thing that the pronoun stands for
      - Extracts additional candidate noun phrases
    - Finetunes RoBERTa to put higher scores to positive referent phrases

RoBERTa Robustly optimized BERT approach

- (Downstream task) GLUE The General Language Understanding Evaluation Results

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

- (■) Single-task, dev
  - RoBERTa achieves SOTA STATE-OF-THE-ART results on all the GLUE task dev sets
  - RoBERTa uses the same things(masked language modeling, architecture) as $\text{BERT}_{\text{LARGE}}$
  - RoBERTa outperforms $\text{BERT}_{\text{LARGE}}$ & XLNet $_{\text{LARGE}}$

    → Dataset size and training time are relatively more important ?!

- (■) Ensembles 5 and 7 models, test to the GLUE leaderboard
  - RoBERTa achieves SOTA results on 4 out of 9 tasks and the highest
    - (□) the highest average score
  - RoBERTa depends on only single-task finetuning

    → Further improvements w/ more sophisticated multi-task finetuning

RoBERTa Robustly optimized BERT approach

- (Downstream task) GLUE The General Language Understanding Evaluation Results

Hyperparam	RACE	SQuAD	GLUE
Learning Rate	1e-5	1.5e-5	{1e-5, 2e-5, 3e-5}
Batch Size	16	48	{16, 32}
Weight Decay	0.1	0.01	0.1
Max Epochs	4	2	10
Learning Rate Decay	Linear	Linear	Linear
Warmup ratio	0.06	0.06	0.06

Table 10: Hyperparameters for finetuning $\text{RoBERTa}_{\text{LARGE}}$ on RACE, SQuAD and GLUE.

Hyperparam	$\text{RoBERTa}_{\text{LARGE}}$	$\text{RoBERTa}_{\text{BASE}}$
Number of Layers	24	12
Hidden size	1024	768
FFN inner hidden size	4096	3072
Attention heads	16	12
Attention head size	64	64
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Warmup Steps	30k	24k
Peak Learning Rate	4e-4	6e-4
Batch Size	8k	8k
Weight Decay	0.01	0.01
Max Steps	500k	500k
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Gradient Clipping	0.0	0.0

Table 9: Hyperparameters for pretraining $\text{RoBERTa}_{\text{LARGE}}$ and $\text{RoBERTa}_{\text{BASE}}$.

RoBERTa Robustly optimized BERT approach

- (Downstream task) RACE The Stanford Question Answering Dataset Results
    - one of three benchmarks that evaluate pretrained models on downstream tasks

Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Single models on dev, w/o data augmentation				
$\text{BERT}_{\text{LARGE}}$	84.1	90.9	79.0	81.8
$\text{XLNet}_{\text{LARGE}}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$\text{XLNet}_{\text{LARGE}}$			$86.3^{\dagger}$	$89.1^{\dagger}$
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

Table 6: Results on SQuAD. † indicates results that depend on additional external training data. RoBERTa uses only the provided SQuAD data in both dev and test settings. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively.

- Only finetunes RoBERTa using the provided SQuAD training data.
- Only use the same learning rate for all layers
- ( ) Follows the same finetuning procedure as Devlin et al.
- ( ) (+) Classifies whether a given question is answerable
    - Train this classifier w/ span predictor (summing classification and span loss terms).
- ( ) Matches the SOTA $_{\text{XLNet}}$
- ( ) Outperforms the EM, F1 $_{\text{XLNet}}$
- ( ) Others rely on additional external training data but, RoBERTa does not use any additional data

RoBERTa Robustly optimized BERT approach

- (Downstream task) RACE The ReAding Comprehension from Examinations Results
  - one of three benchmarks that evaluate pretrained models on downstream tasks
  - RACE
    - Offers
      - A passage of text
      - Associated question
      - 4 candidates answers
    - Task
      - Classify which of the candidate answers is correct.

Model	**Accuracy**	**Middle**	**High**
Single models on test (as of July 25, 2019)			
$\text{BERT}_{\text{LARGE}}$	72.0	76.6	70.1
$\text{XLNet}_{\text{LARGE}}$	81.7	85.4	80.2
RoBERTa	**83.2**	**86.5**	**81.3**

Table 7: Results on the RACE test set. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Yang et al. (2019).

- Only finetunes RoBERTa by concatenate…
  - [one candidate answer, a passage of text, an associated question]
    - (Question, answer) length truncates at most 128 tokens
    - Passage length at most 512 tokens
  - passes [CLS] representations
- RoBERTa achieves SOTA on both settings Middle-school, High-school

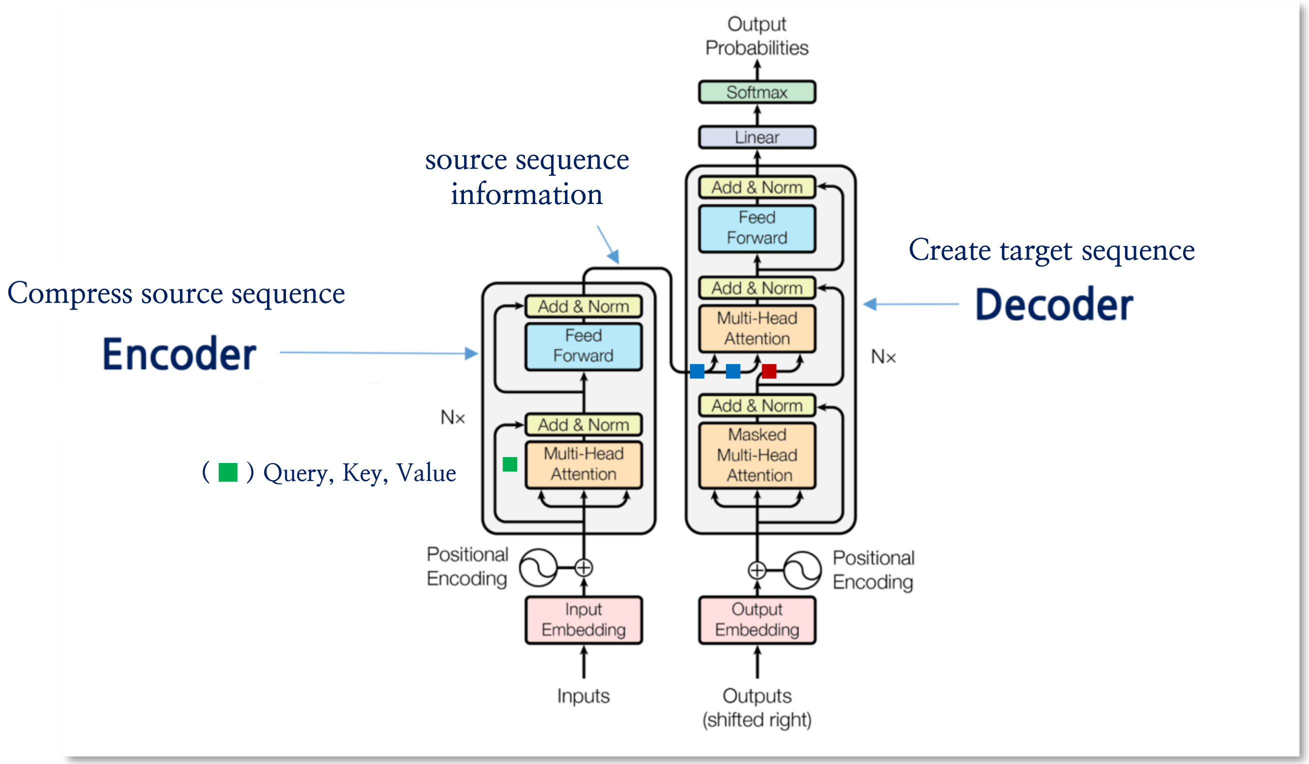
Contents (DeBERTa Decoding–Enhanced BERT w/ Disentangles Attention)

PLM Pre–training LM

Downstream Task

DeBERTa Decoding-Enhanced BERT w/ Disentangled (to separate things that are twisted together) attention

- Background
  - Transformer seq2seq model proposed by Google in 2017 on which BERT, GPT are based



( ) source word vector sequence from last block of encoder

( ) target word vector sequence derived from previous decoder block

Positional Encoding

- Limit
  - Self-attention lacks a way to encode word position information
- Positional bias
  - Each input word is represented by a vector w/ content and position
  - Absolute embedding < Relative embedding
- DeBERTa
  - Propose attention mechanism
    - each input word represented by two separate vectors to encode a word's content and position, respectively.
    - computes attention weights using matrices on contents and relative positions, respectively

DeBERTa Decoding-Enhanced BERT w/ Disentangled (to separate things that are twisted together) attention

- Background
  - Masked Language Model

$$\max_\theta \log p_\theta(\boldsymbol{X}|\tilde{\boldsymbol{X}}) = \max_\theta \sum_{i\in\mathcal{C}} \log p_\theta(\tilde{x}_i = x_i|\tilde{\boldsymbol{X}})$$

- $X = \{x_i\}$ a given sequence , C = index set of the masked tokens in the sequence
- X is corrupted into $\tilde{X}$ by masking 15% of tokens at random
- train a LM parameterized by θ to reconstruct X
  - It predicts the masked token $\tilde{x}$ conditioned on $\tilde{X}$

DeBERTa Decoding–Enhanced BERT w/ Disentangled (to separate things that are twisted together) attention

- Disentangled Attention: a two–vector approach to content and position embedding
  - A token at position i in a sequence is represented by $\{H_i\}$ and $\{P_{i\mid j}\}$
    - $\{H_i\}$: its content ($\neq$ context)
    - $\{P_{i\mid j}\}$: relative position w/ the token at position j
  - Cross attention score between token i and j

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

- (■) : Content–to–content existing approaches to relative position encoding
- (■) : Content–to–position existing approaches to relative position encoding
- (■) : Position–to–content
  - The attention weight of a word pair ($Word_i$, $Word_j$) depends on their contents and relative positions.
- (■) : Position–to–position
  - It is removed because relative position embedding is used.

DeBERTa

- Disentangled Attention: a two-vector approach to content and position embedding

The standard self-attention

$$Q = HW_q,\ K = HW_k,\ V = HW_v,\ A = \frac{QK^\intercal}{\sqrt{d}}$$

$$H_o = \text{softmax}(A)V$$

- $H \in R^{N \times d}$: input hidden vectors
    - N : the length of the input sequence
    - d : the dimension of hidden states
- (■) $H_o \in R^{N \times d}$: The output of self-attention
- (■) $W_{q,k,v} \in R^{d \times d}$: The projection matrices
- (■) $A \in R^{N \times N}$: attention matrix

DeBERTa

- Disentangled Attention: a two-vector approach to content and position embedding

"The disentangled self-attention w/ relative position bias"

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

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

Attention score from token i to token j

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

Scaling factor

- (■) Q_c, K_c, V_c : the projected content vectors generated using projection matrices $W_{q,c}$, $W_{k,c}$, $W_{v,c} \in R^{d \times d}$
- (■) Q_r, K_r : the projected relative position vectors generated using projection matrices $W_{q,r}$, $W_{k,r} \in R^{d \times d}$
  - $P \in R^{2k+d}$: the relative position embedding vector shared across all layers
    - $k = \delta^{delta}(i,j)$: the relative distance from token i to token j
      - $\delta(i,j) \in [_{\geq} 0, _{<} 2k)$

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

DeBERTa

- Disentangled Attention: a two−vector approach to content and position embedding

"The disentangled self−attention w/ relative position bias"

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

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

Attention score from token i to token j

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

Scaling factor

- (■) Q_i^c , K_j^c : i−th row of $Q_c^{\text{content vector}}$, j−th row of K_c content vector
  - (■) $K_j^c\ Q_{\delta(j,i)}^r$ (position $^{\text{query}}$−to−content $^{\text{key}}$)
    - Key : j−th row of K_c content vector
    - Query : $\delta(j,i)$−th row of Q_r relative position vector
      - $\delta(j\,,i)$: the attention weight of the key content at j w.r.t. query position at i
  - (■) $Q_i^c\ K_{\delta(i,j)}^r$ (content $^{\text{query}}$−to−position $^{\text{key}}$)
    - Query : i−th row of Q_c content vector
    - Key : $\delta(i,j)$−th row of K_r relative position vector
      - $\delta(i\,,j)$: the attention weight of the key position at j w.r.t. query content at i
- $\delta(i,j) \leq 512$ (for pre-training)

DeBERTa

- (is pretrained using MLM Masked LM) To predict what the masked word should be, it is trained to use the words surrounding a mask token
- (limitation) the disentangled attention mechanism does not consider the absolute positions
- Enhanced Mask Decoder accounts for absolute word positions

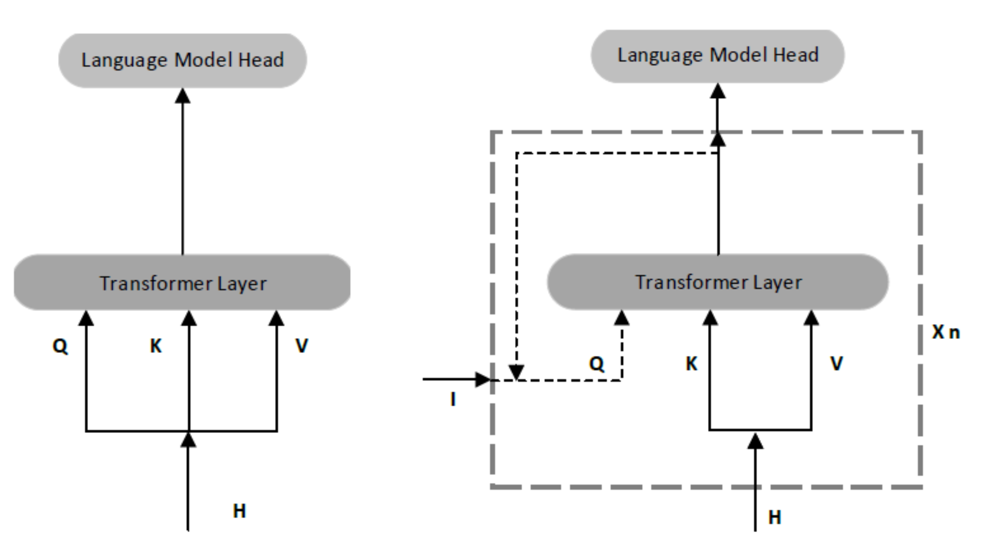
[MASK] (to be predicted) — [MASK] (to be predicted)

"a new **store** opened beside the new **mall**"

- Target
  - distinguish store and mall
- Limitation
  - only using local context relative positions and surrounding words is insufficient
  - *store*, *mall* follows the word *new* with the same relative positions (one step ago)
- Alternative
  - Syntactical nuances(subject is *store* not *mall*) would be caught with absolute positions
- Result
  - DeBERTa captures the relative positions in all the Transformer layers and uses the absolute positions when decoding the masked words

DeBERTa

- Enhanced Mask Decoder accounts for absolute word positions
  - Two methods to incorporate absolute positions
    - BERT : in the input layer
    - DeBERTa : (dashed line) right after all the Transformer layers but before the soft-max layer for masked token prediction



(a) BERT decoding layer (b) Enhanced Mask Decoder

Figure 2: Comparison of the decoding layer.

n stacked EMD layers

- Inputs of the structure of EMD enhanced mask decoder
  - ***I*** : any necessary information for decoding H
    - H
    - Absolute position embedding
    - Output from previous EMD

    → EMD is more general and flexible than BERT
  - ***H***: hidden states from the previous Transformer layer
- Comparison
  - $Performance_{BERT} < Performance_{EMD}$

DeBERTa

- SiFT Scale-invariant-Fine-Tuning
  - (Regularization method) Virtual adversarial training algorithm Miyato et al. (2018); Jiang et al. (2020)
    - improving a model's robustness to adversarial examples
    - Adversarial examples are created by making small perturbations small change to the input
  - Application on NLP
    - The perturbation is applied to the word embedding, not the original word.
    - DeBERTa inspired by layer normalization
      - Proposes SiFT algorithm to apply the perturbations to **the normalized** word embeddings.
      - Finds the normalization improves the performance of fine-tuned models.
      - Only applies SiFT to DeBERTa $_{1.5B}$ on SuperGLUE task

DeBERTa

- Settings
  - 6 DGX-2 machines (96 V100 GPUs)
  - Batch size, Steps : 2K, 1M (in 20 days)
- Experiment
  - Main Results on NLU Natural Language Understanding tasks : Performance on large models (L=24, H=1024, A=16) w/ GLUE

Transformed-based PLMs of similar structures

Pre-trained for 500K steps w/ 8K samples

→ 4B training samples

Model	CoLA Mcc	QQP 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
ELECTRA$_{large}$	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.

Pre-trained on 160G training data

- (training data) DeBERTa is pre-trained on 76G training data
  - Wikipedia(16G) + BookCorpus(6G) + OPENWEBTEXT(38G) + STORIES(31G) → Deduplication → 78G
- (pre-training) DeBERTa is pre-trained for 1M steps w/ 2K samples → 2B training samples
- (■) DeBERTa > BERT and RoBERTa
- (■) DeBERTa > XLNet (6 of 8 tasks)
- (■) DeBERTa > XLNet and ELECTRA (SOTA PLMs in terms of avg GLUE score)
- (■) DeBERTa SOTA in MNLI indicative task to monitor the research progress

DeBERTa

- Settings
  - 6 DGX-2 machines (96 V100 GPUs)
  - Batch size, Steps : 2K, 1M (in 20 days)
- Experiment
  - Main Results on NLU Natural Language Understanding tasks : Performance on large models w/ Question Answering, NLI, NER

NLI inference

Question and Answering

Similar model size

Using the same dataset

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**
$ALBERT_{xxlarge}$	90.8/-	94.8/89.3	90.2/87.4	86.5	-	-	-
$Megatron_{1.3B}$	90.9/91.0	94.9/89.1	90.2/87.1	87.3	-	-	-
$Megatron_{3.9B}$	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 "-".

- DeBERTa > Other models w/ similar model size
- (■) DeBERTa > $Megatron_{1.3B}$ (3 of 4 tasks)

Size of $Megatron_{1.3B}$ = Size of DeBERTa ×3

DeBERTa

- Settings
  - 4 DGX−2 machines (64 V100 GPUs)
  - Batch size, Steps : 2048, 1M (in 10 days)
- Experiment
  - Main Results on NLU Natural Language Understanding tasks : Performance on base models (L=12, H=768, A=12) w/ Question Answering, NLI

on 160G training data

on 78G training data

Model	MNLI-m/mm (Acc)	SQuAD v1.1 (F1/EM)	SQuAD v2.0 (F1/EM)
RoBERTa$_{base}$	87.6/-	91.5/84.6	83.7/80.5
XLNet$_{base}$	86.8/-	-/-	-/80.2
DeBERTa$_{base}$	**88.8/88.5**	**93.1/87.2**	**86.2/83.1**

Table 3: Results on MNLI in/out-domain (m/mm), SQuAD v1.1 and v2.0 development set.

- DeBERTa > XLNet and RoBERTa (with a larger margin than in large models)

DeBERTa

- Ablation study quantify the relative contributions of different components
  - Settings
    - (dataset) pre-trained using the Wikipedia+Bookcorpus dataset
    - (training) 1M steps w/ batch size 256 (in 7 days)
    - (h/w) 1 DGX-2 machine w/ 16 V100 GPUs

To verify experimental settings,
pre-train the RoBERTa base model from scratch
on this settings (RoBERTa-ReImp $_{base}$)

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
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

Enhanced masked decoder
Component-to-position
Position-to-component

Table 4: Ablation study of the DeBERTa base model.

- (■) RoBERTa ≒ RoBERTa-ReImp $_{base}$ → Setting is reasonable
- (■) Removing any one component in DeBERTa results in a sheer large amount performance drop

RoBERTa vs. DeBERTa

- Compare attention patterns of the last layer

Vertical strip

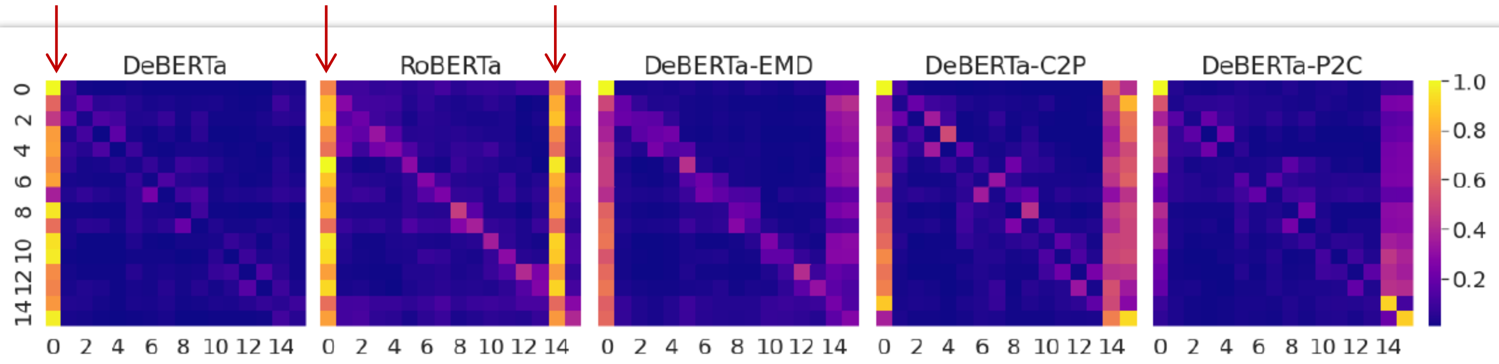


Figure 3: Comparison of attention patterns of the last layer among DeBERTa, RoBERTa and DeBERTa variants (i.e., DeBERTa without EMD, C2P and P2C respectively).

- Two differences between RoBERTa and DeBERTa
  - Diagonal line effect for a token attending to itself
    - (result) RoBERTa has a clear diagonal line effect, but DeBERTa does not have
    - (conjecture) DeBERTa has EMD the absolute position embedding added to the hidden state of content as query vector
    - DeBERTa−EMD w/o EMD has more visible than DeBERTa diagonal line effect
  - Vertical strips
    - (result) in RoBERTa w/ FULL−SENTENCE, they are caused by high−frequent tokens “a”, “the”, “.” punctuation
    - (result) in DeBERTa, only in first column, which represents [CLS] token
    - (conjecture) emphasis on [CLS] is desirable since the feature vector of [CLS] is used as a contextual representation of the entire input sequence in downstream tasks.