Post-BERT Era

BERT-based architectures

Learning goals

- Understand the developments of the post-BERT era
- Get to know different self-supervised objectives
- Understand how to tackle BERTs critical shortcomings

SUCESSORS OF BERT

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

10/2018

SUCESSORS OF BERT

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

10/2018 → 07/2019

July 2019 - RoBERTa

Liu et al., 2019 concentrate on improving the original BERT architecture by (1) careful hyperparameter tuning (2) abandoning the additional Next Sentence Prediction objective (3) increasing the pre-training corpus *massively*.

Other approaches now more and more concentrate on improving, down-scaling or understanding BERT. A new research direction called **BERTology** emerges.

SUCESSORS OF BERT

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

September 2019 - ALBERT

Lan et al., 2019 design a new pre-training objective (SOP) and introduce several parameter reduction techniques to allow for faster and more efficient training of BERT.

Ultimately, they are able to improve the performance of BERT by scaling up the smaller and more efficient model.

July 2019 - RoBERTa

Liu et al., 2019 concentrate on improving the original BERT architecture by (1) careful hyperparameter tuning (2) abandoning the additional Next Sentence Prediction objective (3) increasing the pre-training corpus *massively*.

Other approaches now more and more concentrate on improving, down-scaling or understanding BERT. A new research direction called **BERTology** emerges.

SUCESSORS OF BERT

10/2018 — 07/2019 — 09/2019 — 10/2019

October 2018 - BERT

BERT (**Devlin et al., 2018**) is a bidirectional contextual embedding model purely relying on Self-Attention by using multiple **Transformer encoder** blocks.

BERT (and its successors) rely on the **Masked Language Modelling objective** during pre-training on huge unlabelled corpora of text.

July 2019 - RoBERTa

Liu et al., 2019 concentrate on improving the original BERT architecture by (1) careful hyperparameter tuning (2) abandoning the additional Next Sentence Prediction objective (3) increasing the pre-training corpus *massively*.

Other approaches now more and more concentrate on improving, down-scaling or understanding BERT. A new research direction called **BERTology** emerges.

September 2019 - ALBERT

Lan et al., 2019 design a new pre-training objective (SOP) and introduce several parameter reduction techniques to allow for faster and more efficient training of BERT.

Ultimately, they are able to improve the performance of BERT by scaling up the smaller and more efficient model.

October 2019 - DistilBERT

Sanh et al., 2019 employed the concept of 'model distillation' to create a smaller BERT-type model (contrary to the current trend of building ever larger models).

DistilBERT shows an impressive performance when fine-tuned on downstream tasks despite only exhibiting half the size of the ordinary BERT-BASE model.

ROBERTA – PRE-TRAINING IMPROVEMENTS

Robustly **o**ptimizied **BERT a**pproach ▸ Liu et al., 2019

Short summary:

- Change of the `MASK`ing strategy
  $\rightarrow$ BERT masks the sequences once before pre-training
  $\rightarrow$ RoBERTa uses dynamic `MASK`ing
  $\rightarrow$ RoBERTa sees the same sequence `MASK`ed differently
- RoBERTa **does not use** the additional NSP objective during pre-training
- Authors claim that BERT is seriously "undertrained"
  - 160 GB pre-training corpus instead of 13 GB
  - Pre-training is performed with larger batch sizes (8k)
- Training on full-length sequences only (512 tokens)

ROBERTA – THE ARCHITECTURE

Architectural differences:

- Architecture (layers, heads, embedding size) identical to BERT
- 50k token BPE vocabulary instead of 30k
- Model size differs (due to the larger embedding matrix)
  $\Rightarrow \sim$ 125M (360M) for the BASE (LARGE) variant

Performance differences:

	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**	-

▸ Source: Liu et al., 2019

Note: Liu et al. (2019) report the accuracy for QQP while Devlin et al. (2019) report the F1 score (cf. results displayed in chapter 6.2.3); XLNet: see next Chapter.

DYNAMIC VS. STATIC MASKING

Static Masking (BERT):

- Apply `MASK`ing procedure to pre-training corpus once
- (additional for BERT: Modify the corpus for NSP)
- Train for approximately 40 epochs

Dynamic Masking (RoBERTa):

- Duplicate the training corpus *ten* times
- Apply `MASK`ing procedure to each duplicate of the pre-training corpus
- Train for 40 epochs
- Model sees each training instance in ten different "versions" (each version four times) during pre-training

DYNAMIC VS. STATIC MASKING

- **BERT:**

The	quick	brown	[MASK]	jumps	over	the	[MASK]	dog	.

The	quick	brown	[MASK]	jumps	over	the	[MASK]	dog	.

⋮

The	quick	brown	[MASK]	jumps	over	the	[MASK]	dog	.

- **RoBERTa:**

The	quick	brown	[MASK]	jumps	over	the	[MASK]	dog	.

The	[MASK]	brown	fox	[MASK]	over	the	lazy	dog	.

[MASK]	quick	[MASK]	fox	jumps	over	the	lazy	dog	.

⋮

[MASK]	quick	brown	fox	jumps	over	the	lazy	[MASK]	.

DYNAMIC VS. STATIC MASKING

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).

Source: Liu et al., 2019

NO NEXT SENTENCE PREDICTION

- Described as important part of the pre-training process in BERT
- ▸ Liu et al., 2019 report that it hurts performance

→ Especially for QNLI, MNLI, and SQuAD 1.1

- Conduct experiments in multiple settings:
  - SEGMENT-PAIR+NSP
    (Exactly like BERT)
  - SENTENCE-PAIR+NSP
    (Like BERT, but with natural sentences)
  - FULL-SENTENCES
    (No NSP, inputs may cross document boundaries)
  - DOC-SENTENCES
    (No NSP, only sentences from one document)

NO NEXT SENTENCE PREDICTION

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).

▸ Source: Liu et al., 2019

Note: XLNet: see next Chapter.

BATCH SIZE

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

▸ Source: Liu et al., 2019

FURTHER CHANGES IN PRE-TRAINING

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**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
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 $\rightarrow$ 160GB of text) and pretrain for longer (100K $\rightarrow$ 300K $\rightarrow$ 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

▸ Source: Liu et al., 2019

Note: XLNet: see next Chapter.

ALBERT

A Lite BERT ‣ Lan et al., 2019

Short summary:

- Major change to the architecture
  $\rightarrow$ Size of embedding layer independent from hidden layer
  (rember in Transformer/BERT: $E = H$)
  $\rightarrow$ Saves memory and compute
- Parameter sharing across layers
- Substitution of the NSP objective during pre-training

ARCHITECTURAL CHANGES

Disentanglement of E and H

- WordPiece-Embeddings (size E)
  - first layer of the model
  - each token is initially mapped to this embedding
  - context-independent
- In Transformer/BERT:
  - $H = E$
  - down-project E to keys, queries and values of size H/A
  - concatenate resulting embeddings from all A heads
  - results in hidden layer representation of size H
- **Question:** What are the implications?

ARCHITECTURAL CHANGES

Thoughts / Implications

- WordPiece-Embeddings (size E)
  - required representational capacity?
  - probably could be limited w/o loosing much
- Hidden-Layer-Embedding (size H)
  - required representational capacity?
  - depending on how polysemous a word/token might be
  - difficult to say "one size fits all"
  - probably might be better to rather increase this, compared to the WordPiece embeddings

$\rightarrow$ *Setting $E = H$ does not allow us to pursue these considerations*

ARCHITECTURAL CHANGES

Disentanglement solves this

- Hidden-Layer-Embeddings (size H) context-dependent
  $\rightarrow$ providing more capacity makes more sense here
- Setting $H >> E$ enlargens model capacity in the hidden layers without increasing the size of the embedding matrix
- $O(V \times H) > O(V \times E + E \times H)$ if $H >> E$

CROSS-LAYER PARAMETER SHARING

- Pre-trained transformer-based models are deep stacks of identically parametrized layers and thus have many parameters
- Sharing them as a way to gain parameter efficiency
- Two different places in the network, where sharing can be done
  - Attention parameters
  - FFN parameters
- Ablation studies:
  - both
  - both individually
  - none

CROSS-LAYER PARAMETER SHARING

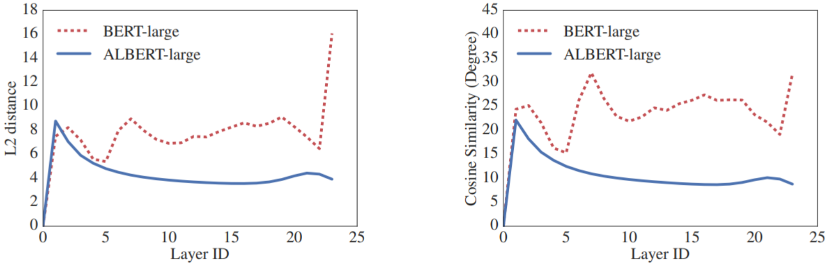
Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

Table 4: The effect of cross-layer parameter-sharing strategies, ALBERT-base configuration.

Source: Lan et al., 2019

- (Drastic) reduction of model size (more for sharing FFN weights)
- Sharing parameters hurts performance
  - Worse for models with larger E
  - Worse for sharing FNN compared to Attention weights
  - **Question: Why? What is the intuition here?**

CROSS-LAYER PARAMETER SHARING



▸ Source: Lan et al., 2019

- Distince/Similarity of input and output embeddings per layer
- Smoother transitions for ALBERT models

CHANGES IN PRE-TRAINING

Substitution of the NSP objective

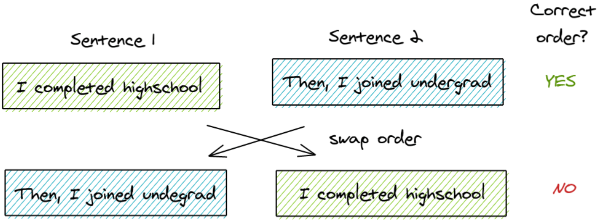
- Previous works questioned the usefulness of NSP

$\rightarrow$ Lan et al. assume that this is due to lacking difficulty

- Introduction of *Sentence-Order Prediction* (SOP) as a new pre-training task
- Positive examples created alike to those from NSP (take two consecutive sentences from the same document)
- Negative examples: Just swap the ordering of sentences

CHANGES IN PRE-TRAINING

Illustration:



▸ Source: Amit Chaudhary

Effectiveness:

SP tasks	Intrinsic Tasks MLM	NSP	SOP	Downstream Tasks SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

▸ Source: Lan et al., 2019

CHANGES IN PRE-TRAINING

$n - gram$ **masking for the MLM task**

- During pre-training BERT single tokens are masked
- Lan et al. mask up to three consecutive tokens
- Choice of n:

$$p(n) = \frac{1/n}{\sum_{k=1}^{N} 1/k}$$

PERFORMANCE

Performance differences:

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

▸ Source: Lan et al., 2019

Notes:

- In General: Smaller model size (because of parameter sharing)
- Nevertheless: Scale model up to almost similar size (`xxlarge` version)
- Strong performance compared to BERT