Slim but Smart -- a Review on ALBERT

01 Dec 2021 | # natural_lanugage_inference # NLP # deep_learning # model_reduction

Anonymous

What does it take to build an intelligent robot that comprehends text and responds appropriately to relevant questions? For example, how good can a robot be at taking the English reading comprehension exams (RACE) administered to Chinese middle-schoolers and high-schoolers? In 2018, Google's AI Language team published BERT, a breakthrough in natural language processing designed to answer this very question. Named for its bidirectional transformer structure, BERT established the significant performance advantages of pre-trained models for learning tasks; the model utilized **340 million** parameters to achieve an accuracy of 72%. Even with Tensor Processing Units (TPUs) – Google's proprietary hardware developed specifically for large-scale deep learning tasks – BERT required four days of training on 64 TPUs!

Passage:
In a small village in England about 150 years ago, a mail coach was standing on the street. It didn't come to that village often. People had to pay a lot to get a letter. The person who sent the letter didn't have to pay the postage, while the receiver had to.
"Here's a letter for Miss Alice Brown," said the mailman.
" I'm Alice Brown," a girl of about 18 said in a low voice.
Alice looked at the envelope for a minute, and then handed it back to the mailman.
"I'm sorry I can't take it, I don't have enough money to pay it", she said.
A gentleman standing around were very sorry for her. Then he came up and paid the postage for her.
When the gentleman gave the letter to her, she said with a smile, " Thank you very much, This letter is from Tom. I'm going to marry him. He went to London to look for work. I've waited a long time for this letter, but now I don't need it, there is nothing in it."
"Really? How do you know that?" the gentleman said in surprise.
"He told me that he would put some signs on the envelope. Look, sir, this cross in the corner means that he is well and this circle means he has found work. That's good news."
The gentleman was Sir Rowland Hill. He didn't forgot Alice and her letter.
"The postage to be paid by the receiver has to be changed," he said to himself and had a good plan.
"The postage has to be much lower, what about a penny? And the person who sends the letter pays the postage. He has to buy a stamp and put it on the envelope." he said . The government accepted his plan. Then the first stamp was put out in 1840. It was called the "Penny Black". It had a picture of the Queen on it.

Questions:

1): The first postage stamp was made _.
A. in England B. in America C. by Alice D. in 1910

2): The girl handed the letter back to the mailman because _ .
A. she didn't know whose letter it was
B. she had no money to pay the postage
C. she received the letter but she didn't want to open it
D. she had already known what was written in the letter

3): We can know from Alice's words that _ .
A. Tom had told her what the signs meant before leaving
B. Alice was clever and could guess the meaning of the signs
C. Alice had put the signs on the envelope herself
D. Tom had put the signs as Alice had told him to

4): The idea of using stamps was thought of by _ .
A. the government
B. Sir Rowland Hill
C. Alice Brown
D. Tom

5): From the passage we know the high postage made _ .
A. people never send each other letters
B. lovers almost lose every touch with each other
C. people try their best to avoid paying it
D. receivers refuse to pay the coming letters

Answer: ADABC

Here's an example question taken from the RACE test (not the clearest abbreviations, I know). If you have a second to scan through the story above, you will realize that it is pretty simple for humans, but it takes a lot of compute for a computer to understand the passage.

As with any machine learning breakthrough, there are two questions that immediately come to mind. Can we achieve the same results more efficiently? And can we improve upon this performance?

Let's start out with the first question: we need a way to slim down this gigantic model! At a basic level, memory limitations are an immediate bottleneck. Even with dedicated hardware, there is always an incentive to shrink our models down to reduce the memory demands of training. Additionally, large models like BERT are typically trained in a distributed manner across many machines; the communication overhead associated with such an approach is also significant. Ideally, we can address both these issues by shrinking the number of parameters in the model.

Enter ALBERT – short for A Lite BERT. ALBERT is a smaller version of BERT that strives to circumvent these exact limitations while also making performance advancements.

Improvements Offered by ALBERT

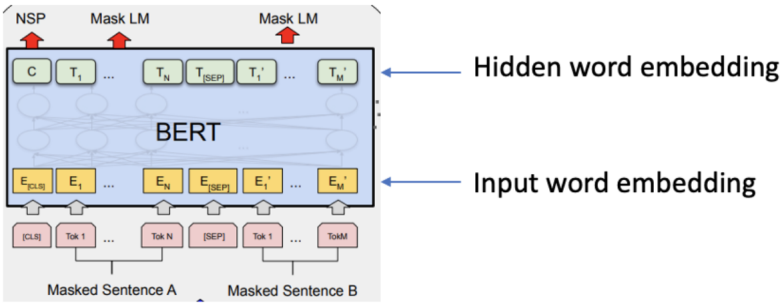
ALBERT utilizes three key features to be slim but smart. The first two are techniques used to slim down the model and the last one boosts its performance.

1. Factorized embedding parameterization
2. Cross-layer parameter sharing
3. Self-supervised loss for sentence-order prediction.

Factorized Embedding Parameterization

Machines use numerical values, but words are categorical. The way we can translate words to trainable metrics is by

using word embeddings, which are a set of numerical values unique to each word or word-related token (such as WordPieces embeddings used in BERT). These values can be tweaked after being propagated across multiple blocks of the transformer to reflect higher level relationships between words.



The architecture of BERT, showing *E* and *T*.

(Annotations added by authors of this blog)

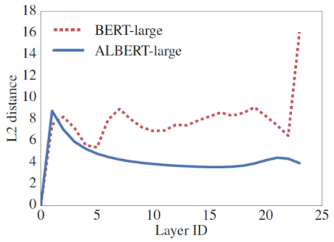
In the original BERT model, the input word embeddings *E* have the same size as the hidden word embeddings *T*. However, ALBERT separates the two embedding matrices. This is because WordPiece (input) embedding E are designed to learn context-independent representations, whereas hidden-layer embeddings T are meant to learn context-dependent representations, and BERT performs well precisely because of these context-dependent representations. ALBERT delinks the size of *E* vectors from that of *T*. It projects the input through a lower dimensional space before projecting it into the hidden space. This change led to a significant decrease in parameters.

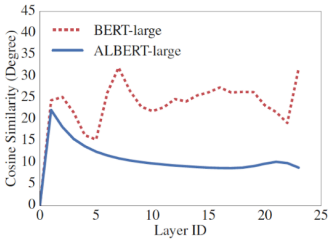
Cross-layer Parameter Sharing

In a transformer layer, there is an attention network and a feed-forward network. In the original BERT model, the multi-head self-attention segment and feed-forward segment each have their own parameters. ALBERT instead has parameters that are shared between these subsegments. By sharing parameters across all layers, the total number of parameters within ALBERT is greatly reduced relative to BERT. In addition to decreasing parameter redundancy and improving overall model

efficiency, this feature also stabilizes the parameters such that the model is more generalizable.

> **Sidenote**: stabilizing the parameters means that the difference between the inputs and outputs of different layers are not widely different. This can be analyzed using the L2 distances and cosine similarity of the input and output embeddings of each layer. As shown here, ALBERT's curve is smooth, suggesting more stable parameters.





Sentence Order Prediction Loss

BERT used Next-Sentence Prediction (NSP) as a loss function, which predicts "yes" or "no" as to whether two segments of text appear consecutively in the original text. The NSP's objective was designed to improve performance on other tasks, such as understanding whether two sentences have the same meaning, but was later deemed ineffective. This might be due to the fact that NSP tries to combine the prediction of topics and the coherence of the sentences into one task. Insead, ALBERT uses a new loss called Sentence Order Prediction (SOP), which only focuses on modeling inter-sentence coherence. This has helped ALBERT achieve better results.

ALBERT's Experimental Setup

For comparison, the authors preprocessed their data and performed training with the same methodology that was used for BERT. ALBERT was trained on the English Wikipedia and BOOKCORPUS datasets, which together contain ~16 GB of uncompressed data. ALBERT also uses a

vocabulary size of 30,000 and groups together the vocabulary in groups of 1-grams, 2-grams, or 3-grams.

> **Side note**: contiguous sequences of n words can be grouped together into n-grams.

Impacts of ALBERT's Improvements

Let's take a deeper dive into just how much better ALBERT is compared to its bigger brother.

High level Architecture/Parameter Comparison

ALBERT is released in 4 different model sizes, with all versions being a fraction of the size of BERT. For example, ALBERT-base has 9x fewer parameters than BERT-base, and ALBERT-large has about 18x fewer parameters compared to BERT-large.

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
	xlarge	1270M	24	2048	2048	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	59M	24	2048	128	True
	xxlarge	233M	12	4096	128	True

Overall Improvement

The ALBERT-xxlarge model performs significantly better than BERT-large while having 70% fewer parameters! Additionally, the biggest BERT model performed significantly worse than BERT-base on all metrics, which indicates that **bigger is not always better**. After training for roughly the same amount of time, ALBERT-xxlarge is significantly better than BERT-large: +1.5% better on average, with the difference on the RACE dataset as high as +5.2%.

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.5/83.3	80.3/77.3	84.1	91.7	68.3	82.1	17.7x
	large	334M	92.4/85.8	83.9/80.8	85.8	92.2	73.8	85.1	3.8x
	xlarge	1270M	86.3/77.9	73.8/70.5	80.5	87.8	39.7	76.7	1.0
ALBERT	base	12M	89.3/82.1	79.1/76.1	81.9	89.4	63.5	80.1	21.1x
	large	18M	90.9/84.1	82.1/79.0	83.8	90.6	68.4	82.4	6.5x
	xlarge	59M	93.0/86.5	85.9/83.1	85.4	91.9	73.9	85.5	2.4x
	xxlarge	233M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

Models	Steps	Time	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
BERT-large	400k	34h	93.5/87.4	86.9/84.3	87.8	94.6	77.3	87.2
ALBERT-xxlarge	125k	32h	**94.0/88.1**	**88.3/85.3**	87.8	**95.4**	**82.5**	**88.7**

The results of ALBERT's performance on a variety of benchmark datasets, including the Reading Comprehension test (RACE) mentioned before

The experimental data demonstrates that ALBERT makes notable performance gains over BERT while having considerably fewer parameters. Let's now assess the individual impact of each of ALBERT's key design choices:

Factorized Embeddings Parameterization

In order to assess the impact of the factorized embeddings, four different vocabulary embedding sizes E were used. For each embedding E, the score was collected for the different downstream tasks. This experiment was performed twice: once for BERT-style models with no parameter sharing and once with ALBERT-style models featuring parameter sharing. For the BERT-style models, larger embedding sizes correlated to superior performance, although the performance difference was relatively small. For the ALBERT-style model, it was found that an embedding size of 128 consistently yielded the best performance. Therefore, the authors decided to use an embedding size of 128 for all future ALBERT experiments/settings.

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Cross-Layer Parameter Sharing

To assess the cross-layer parameter sharing, four types of parameter-sharing were assessed on the downstream tasks: not-shared (BERT-style), all-shared (ALBERT-style), only sharing attention parameters, and only sharing feed-forward parameters. The latter two are essentially intermediate strategies where only some parameters are shared. All four strategies were trialed with an embedding size of 768 and 128. It was found that the all-shared

strategy employed by ALBERT actually impairs performance for both embedding sizes. However, in comparing the attention-only sharing with the feed-forward-only sharing, it appears that the feed-forward sharing contributes much more to the overall performance drop seen by ALBERT's all-shared approach. Despite these results, it was found that for the chosen embedding size setting of 128, ALBERT's performance deficit is relatively minimal.

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

Sentence Order Prediction

To assess the effectiveness of sentence order prediction, SOP was compared against NSP (BERT-style) as well as the case where no inter-sentence loss is used. The scores were collected for each of these three loss strategies across the different downstream tasks. The results demonstrate that NSP loss doesn't aid in the SOP task, indicating that NSP only models topic shifts but not inter-sentence coherence. Conversely, the findings show that SOP loss performs well for both tasks. Most importantly, SOP loss consistently improves performance for multi-sentence encoding tasks.

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	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**

Final Words

Model pruning is commonly used to reduce the size of complex models for faster inference, but we can also save training time (along with inference time) and compute resources by building a smaller model from the very beginning. ALBERT did this through making the vocabulary embedding layer smaller, sharing parameters across all of its layers, and using a loss function better suited for the task of comprehending sentence coherence.

The key takeaway from ALBERT is that larger models are not always better! Clever design choices can make the models smaller but perform better.

Older | Newer