Switch Transformers

Илья Федоров, 417 группа

Факультет вычислительной математики и кибернетики
МГУ им. М.В.Ломоносова

9 апреля 2021 г.

Содержание

Напоминание о трансформере

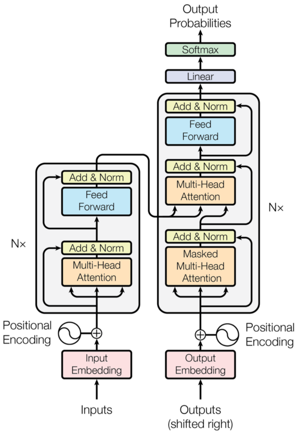
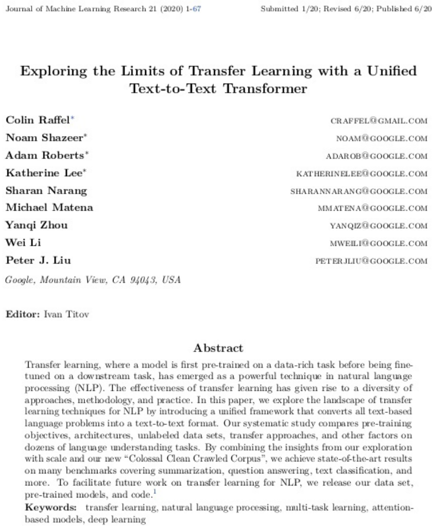


Figure 1: The Transformer - model architecture.

Attention Is All You Need, Vaswani et al. 2017

Модель T5


Journal of Machine Learning Research 21 (2020) 1-67 Submitted 1/20; Revised 6/20; Published 6/20

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Colin Raffel* CRAFFEL@GMAIL.COM
Noam Shazeer* NOAM@GOOGLE.COM
Adam Roberts* ADAROB@GOOGLE.COM
Katherine Lee* KATHERINELEE@GOOGLE.COM
Sharan Narang SHARANNARANG@GOOGLE.COM
Michael Matena MMATENA@GOOGLE.COM
Yanqi Zhou YANQIZ@GOOGLE.COM
Wei Li MWEILI@GOOGLE.COM
Peter J. Liu PETERJLIU@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Editor: Ivan Titov

Abstract

Transfer learning, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). The effectiveness of transfer learning has given rise to a diversity of approaches, methodology, and practice. In this paper, we explore the landscape of transfer learning techniques for NLP by introducing a unified framework that converts all text-based language problems into a text-to-text format. Our systematic study compares pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks. By combining the insights from our exploration with scale and our new "Colossal Clean Crawled Corpus", we achieve state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more. To facilitate future work on transfer learning for NLP, we release our data set, pre-trained models, and code.[1]

Keywords: transfer learning, natural language processing, multi-task learning, attention-based models, deep learning

1 / 67

Модель T5

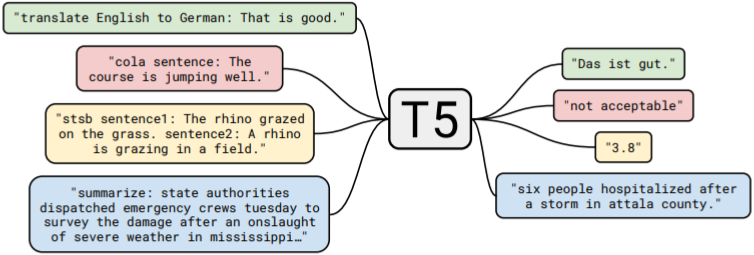


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Модель T5

Exploring the Limits of Transfer Learning

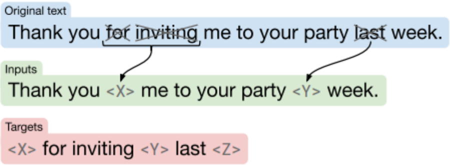


Figure 2: Schematic of the objective we use in our baseline model. In this example, we process the sentence "Thank you for inviting me to your party last week." The words "for", "inviting" and "last" (marked with an $\times$) are randomly chosen for corruption. Each consecutive span of corrupted tokens is replaced by a sentinel token (shown as `<X>` and `<Y>`) that is unique over the example. Since "for" and "inviting" occur consecutively, they are replaced by a single sentinel `<X>`. The output sequence then consists of the dropped-out spans, delimited by the sentinel tokens used to replace them in the input plus a final sentinel token `<Z>`.

Модель T5

Colossal **C**lean **C**rawled **C**orpus = C4

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Table 8: Performance resulting from pre-training on different data sets. The first four variants are based on our new C4 data set.

Также скоро на

Датасет C4 доступен уже с препроцессингом:
https://github.com/allenai/allennlp/discussions/5056

Switch Transformers

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
Google Brain
liamfedus@google.com

Barret Zoph*
Google Brain
barretzoph@google.com

Noam Shazeer
Google Brain
noam@google.com

Abstract

In deep learning, models typically reuse the same parameters for all inputs. Mixture of Experts (MoE) models defy this and instead select *different* parameters for each incoming example. The result is a sparsely-activated model – with an outrageous number of parameters – but a constant computational cost. However, despite several notable successes of MoE, widespread adoption has been hindered by complexity, communication costs, and training instability. We address these with the Switch Transformer. We simplify the MoE routing algorithm and design intuitive improved models with reduced communication and computational costs. Our proposed training techniques mitigate the instabilities, and we show large sparse models may be trained, for the first time, with lower precision (bfloat16) formats. We design models based off T5-Base and T5-Large (Raffel et al., 2019) to obtain up to 7x increases in pre-training speed with the same computational resources. These improvements extend into multilingual settings where we measure gains over the mT5-Base version across all 101 languages. Finally, we advance the current scale of language models by pre-training up to trillion parameter models on the "Colossal Clean Crawled Corpus", and achieve a 4x speedup over the T5-XXL model.[1]

Switch Transformers

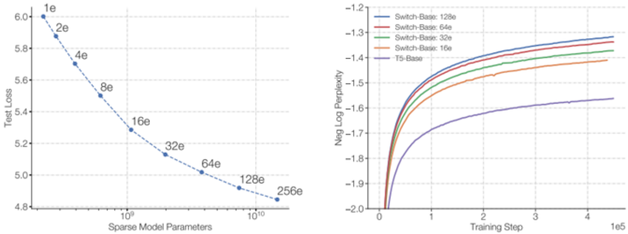


Figure 1: **Scaling and sample efficiency of Switch Transformers. Left Plot:** Scaling properties for increasingly sparse (more experts) Switch Transformers. **Right Plot:** Negative log-perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget.

Perplexity

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

equivalently:

$$PP(W) = 2^{-l}$$

$$\text{where } l = \frac{1}{N} \log P(w_1 w_2 \ldots w_N)$$

$$2^{-l} \text{ where } l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

Switch Transformers

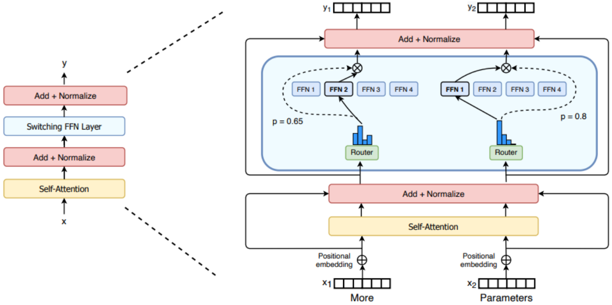
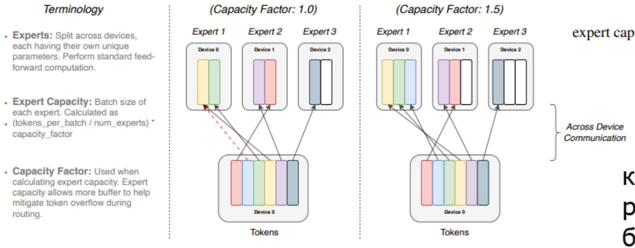


Figure 2: **Illustration of a Switch Transformer encoder block**. We replace the dense feed forward network (FFN) layer present in the Transformer with a sparse Switch FFN layer (light blue). The layer operates independently on the tokens in the sequence. We diagram two tokens (x_1 = "More" and x_2 = "Parameters" below) being routed (solid lines) across four FFN experts, where the router independently routes each token. The switch FFN layer returns the output of the selected FFN multiplied by the router gate value (dotted-line).

"We design our model with TPUs in mind, which require statically declared sizes"

=> нужно искать баланс между вместимостью и скоростью (из-за паддингов)



$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}$$

как бы научить сетку раскидывать токены более-менее равномерно?

Figure 3: **Illustration of token routing dynamics.** Each expert processes a fixed batch-size of tokens modulated by the *capacity factor*. Each token is routed to the expert with the highest router probability, but each expert has a fixed batch size of (total_tokens / num_experts) × capacity_factor. If the tokens are unevenly dispatched then certain experts will overflow (denoted by dotted red lines), resulting in these tokens not being processed by this layer. A larger capacity factor alleviates this overflow issue, but also increases computation and communication costs (depicted by padded white/empty slots).

Добавим вспомогательный лосс...

A Differentiable Load Balancing Loss. To encourage a balanced load across experts we add an auxiliary loss (Shazeer et al., 2017; 2018; Lepikhin et al., 2020). As in Shazeer et al. (2018); Lepikhin et al. (2020), Switch Transformers simplifies the original design in Shazeer et al. (2017) which had separate load-balancing and importance-weighting losses. For each Switch layer, this auxiliary loss is added to the total model loss during training. Given N experts indexed by $i = 1$ to N and a batch $\mathcal{B}$ with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P,

$$\text{loss} = \alpha N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax } p(x), i\} \tag{5}$$

and P_i is the fraction of the router probability allocated for expert i, [2]

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x) \tag{6}$$

Since we seek uniform routing of the batch of tokens across the N experts, we desire both vectors to have values of $1/N$. The auxiliary loss of Equation 4 achieves encourages uniform routing since it is minimized under a uniform distribution. The objective can also be differentiated as the P-vector is differentiable, but the f-vector is not. The final loss is multiplied by expert count N to keep the loss constant as the number of experts varies since under uniform routing $\sum_1^N (f_i \cdot P_i) = \sum_1^N (\frac{1}{N} \cdot \frac{1}{N}) = \frac{1}{N}$. Finally, a hyper-parameter α is a multiplicative coefficient for these auxiliary losses; throughout this work we use an $\alpha = 10^{-2}$ which was sufficiently large to ensure load balancing while small enough to not to overwhelm the primary cross-entropy objective.

Повышаем стабильность (с кототорой проблемы у оригинальной модели MoE)

1) В оригинальном MoE - все обучается в fp32 => долго
Сделали fp16 только в router'е - получилось хорошо.
Пересылать fp32 между девайсами долго

Selective precision with large sparse models. Model instability hinders the ability to train using efficient bfloat16 precision, and as a result, Lepikhin et al. (2020) trains with float32 precision throughout their MoE Transformer. However, we show that by instead *selectively casting* to float32 precision within a localized part of the model, stability may be achieved, without incurring expensive communication cost of float32 tensors. Table 2 shows that our approach permits nearly equal speed to bfloat16 training while conferring the training stability of float32.

Model (precision)	Quality (Neg. Log Perp.)	Speed (Examples/sec)
Switch-Base (float32)	-1.718	1160
Switch-Base (bfloat16)	-3.780 [*diverged*]	**1390**
Switch-Base (Selective precision)	**-1.716**	1390

Table 2: **Selective precision**. We cast the local routing operations to float32 while preserving bfloat16 precision elsewhere to stabilize our model while achieving nearly equal speed to (unstable) bfloat16-precision training. We measure the quality of a 32 expert model after a fixed step count early in training its speed performance.

2) Инициализация весов все еще очень важна!

Smaller parameter initialization for stability. Appropriate initialization is critical to successful training in deep learning and we especially observe this to be true for Switch Transformer. We initialize our weight matrices by drawing elements from a truncated normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{s/n}$ where s is a scale hyper-parameter and n is the number of input units in the weight tensor (e.g. fan-in).

As an additional remedy to the instability, we recommend reducing the default Transformer initialization scale $s = 1.0$ by a factor of 10. This both improves quality and reduces the likelihood of destabilized training in our experiments. Table 3 measures the improvement of the model quality and reduction of the variance early in training. We find that the average model quality, as mea-

Model (Initialization scale)	Average Quality (Neg. Log Perp.)	Std. Dev. of Quality (Neg. Log Perp.)
Switch-Base (0.1x-init)	**-2.72**	**0.01**
Switch-Base (1.0x-init)	-3.60	0.68

Table 3: **Reduced initialization scale improves stability**. Reducing the initialization scale results in better model quality and more stable training of Switch Transformer. Here we record the average and standard deviation of model quality, measured by the negative log perplexity, of a 32 expert model after 3.5k steps (3 random seeds each).

Сравненение моделей на downstream задачах

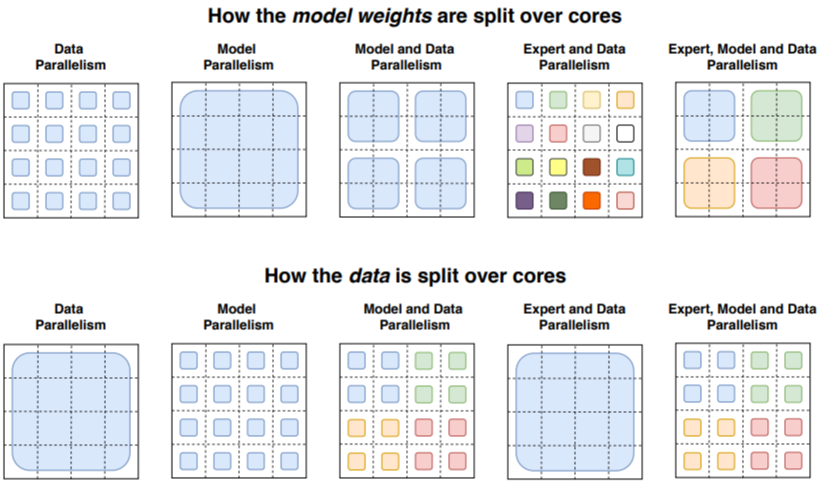
Model	Parameters	FLOPS
T5-Base	223M	124B
Switch-Base	7.4B	124B
T5-Large	739M	425B
Switch-Large	26.3B	425B

Model	GLUE	SQuAD	SuperGLUE	Winogrande (XL)
T5-Base	84.3	85.5	75.1	66.6
Switch-Base	**86.7**	**87.2**	**79.5**	**73.3**
T5-Large	87.8	88.1	82.7	79.1
Switch-Large	**88.5**	**88.6**	**84.7**	**83.0**

Model	XSum	ANLI (R3)	ARC Easy	ARC Chal.
T5-Base	18.7	51.8	56.7	**35.5**
Switch-Base	**20.3**	**54.0**	**61.3**	32.8
T5-Large	20.9	56.6	**68.8**	**35.5**
Switch-Large	**22.3**	**58.6**	66.0	**35.5**

Model	CB Web QA	CB Natural QA	CB Trivia QA
T5-Base	26.6	25.8	24.5
Switch-Base	**27.4**	**26.8**	**30.7**
T5-Large	27.7	27.6	29.5
Switch-Large	**31.3**	**29.5**	**36.9**

Switch Transformers



Switch Transformers

D Switch Transformers in Lower Compute Regimes

Switch Transformer is also an effective architecture at small scales as well as in regimes with thousands of cores and trillions of parameters. Many of our prior experiments were at the scale of 10B+ parameter models, but we show in Figure 12 as few as 2 experts produce compelling gains over a FLOP-matched counterpart. Even if a super computer is not readily available, training Switch Transformers with 2, 4, or 8 experts (as we typically recommend one expert per core) results in solid improvements over T5 dense baselines.

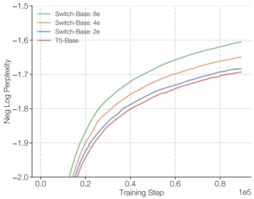


Figure 12: **Switch Transformer with few experts.** Switch Transformer improves over the baseline even with very few experts. Here we show scaling properties at very small scales, where we improve over the T5-Base model using 2, 4, and 8 experts.