ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators

Лысенко Иван 193

Outline

Masked language modeling vs. replaced token detection

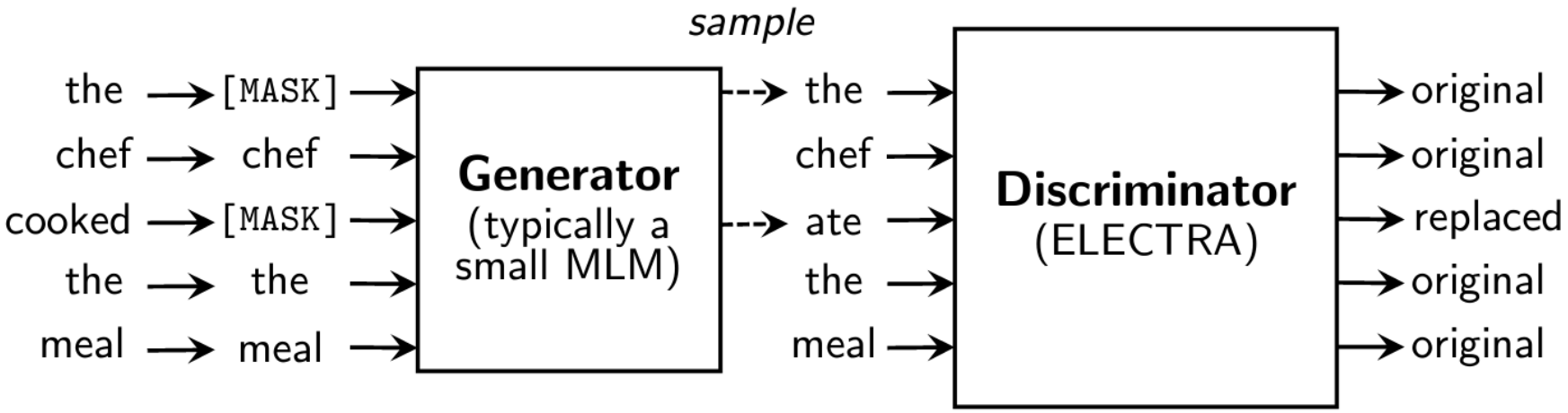
- MLM: заменяем 15% токенов на "[MASK]" и пытаемся восстановить исходные токены
- RTD: заменяем 15% токенов на синтетические из какого-то конкретного распределения (обычно выход какой-то MLM модели) и пытаемся отличить настоящие токены от синтетических

Masked language modeling vs. replaced token detection

Плюсы replaced token detection по сравнению с masked language modeling:

- В MLM модель учится только на 15% токенов, которые мы заменили на "[MASK]", тогда как в RTD модель учится на всех токенах
- При pre-training на MLM модель видит искусственные токены "[MASK]", но при обучении на downstream task – нет, что может негативно влиять на качество модели

Процесс pre-training: overview



Мы тренируем две модели (энкодеры, чаще всего трансформерные): генератор G и дискриминатор D.
Для каждой из моделей:

- Вход: $\boldsymbol{x} = [x_1, \ldots, x_n]$ – последовательность токенов
- Выход: $h(\boldsymbol{x}) = [h_1, \ldots, h_n]$ – их контекстуализированные векторные представления
- Голова: MLM для генератора, RTD для дискриминатора

Процесс pre-training: генератор

Для всех входных позиций t, на которых стоит токен "[MASK]", генератор возвращает распределение над словарем:

$$p_G(x_t \mid \boldsymbol{x}) = \frac{\exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right)}{\sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)},$$

где e – эмбеддинг токена.

Процесс pre-training: дискриминатор

Для всех своих входных позиций t дискриминатор пытается понять, "настоящий" ли токен x_t, то есть является ли он выходом генератора или нет. **Если генератор выдал тот же токен, что и был в данных, то мы считаем его настоящим.**

$$D(\boldsymbol{x}, t) = \text{sigmoid}(w^T h_D(\boldsymbol{x})_t),$$

где w – обучаемый вектор.

Процесс pre-training: forward pass

1. Случайная замена входных токенов на "[MASK]"

$$k = \lceil 0.15n \rceil$$

$$\boldsymbol{m} = [m_1, \ldots, m_k],\ m_i \sim \mathcal{U}\{1, n\}\ \forall i = 1, \ldots k$$

$$\boldsymbol{x}^{masked} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, [MASK])$$

Процесс pre-training: forward pass

1. Случайная замена входных токенов на "[MASK]"

$$k = \lceil 0.15n \rceil$$
$$\boldsymbol{m} = [m_1, \ldots, m_k],\ m_i \sim \mathcal{U}\{1, n\}\ \forall i = 1, \ldots k$$
$$\boldsymbol{x}^{masked} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, [MASK])$$

2. Генератор

$$\hat{\boldsymbol{x}} = [\hat{x}_1, \ldots, \hat{x}_k],\ \hat{x}_i \sim p_G(x_i \mid \boldsymbol{x}^{masked})\ \forall i \in \boldsymbol{m}$$
$$\boldsymbol{x}^{corrupt} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

Процесс pre-training: forward pass

1. Случайная замена входных токенов на "[MASK]"

$$k = \lceil 0.15n \rceil$$
$$\boldsymbol{m} = [m_1, \ldots, m_k],\ m_i \sim \mathcal{U}\{1, n\}\ \forall i = 1, \ldots k$$
$$\boldsymbol{x}^{masked} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, [MASK])$$

2. Генератор

$$\hat{\boldsymbol{x}} = [\hat{x}_1, \ldots, \hat{x}_k],\ \hat{x}_i \sim p_G(x_i \mid \boldsymbol{x}^{masked})\ \forall i \in \boldsymbol{m}$$
$$\boldsymbol{x}^{corrupt} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

3. Дискриминатор

$$[D(\boldsymbol{x}^{corrupt}, 1), \ldots D(\boldsymbol{x}^{corrupt}, n)]$$

Процесс pre-training: лоссы

1. Лосс генератора (средний negative log-likelihood):

$$\mathcal{L}_{MLM}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i \mid \boldsymbol{x}^{masked})\right)$$

Процесс pre-training: лоссы

1. Лосс генератора (средний negative log-likelihood):

$$\mathcal{L}_{MLM}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i \mid \boldsymbol{x}^{masked})\right)$$

2. Лосс дискриминатора (средний log loss):

$$\mathcal{L}_{Disc}(\boldsymbol{x}, \theta_D) = \mathbb{E}\Bigg(\sum_{t=1}^{n} -\mathbb{1}(x_t^{corrupt} = x_t)\log D(\boldsymbol{x}^{corrupt}, t) - \\ -\mathbb{1}(x_t^{corrupt} \neq x_t)\log(1 - D(\boldsymbol{x}^{corrupt}, t))\Bigg)$$

Авторы аппроксимируют ожидания одним сэмплом.

Процесс pre-training: задача оптимизации

$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{MLM}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{Disc}(\boldsymbol{x}, \theta_D),$$

где $\mathcal{X}$ – корпус текстов, а λ – гиперпараметр.

Benchmarks: GLUE

Чтобы получить хороший score на GLUE, модель должна быть хороша на многих доменах одновременно: question answering, sentiment analysis, и др. Таким образом, модель должна делиться общими лингвистическими знаниями между задачами. Результатом является средняя оценка по всем корпусам.

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

Benchmarks: SQuAD

Question answering dataset из статей Википедии.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.

Model extensions: weight sharing

Авторы попробовали связать матрицы эмбеддингов генератора и дискриминатора, а также полностью связать все веса двух моделей.

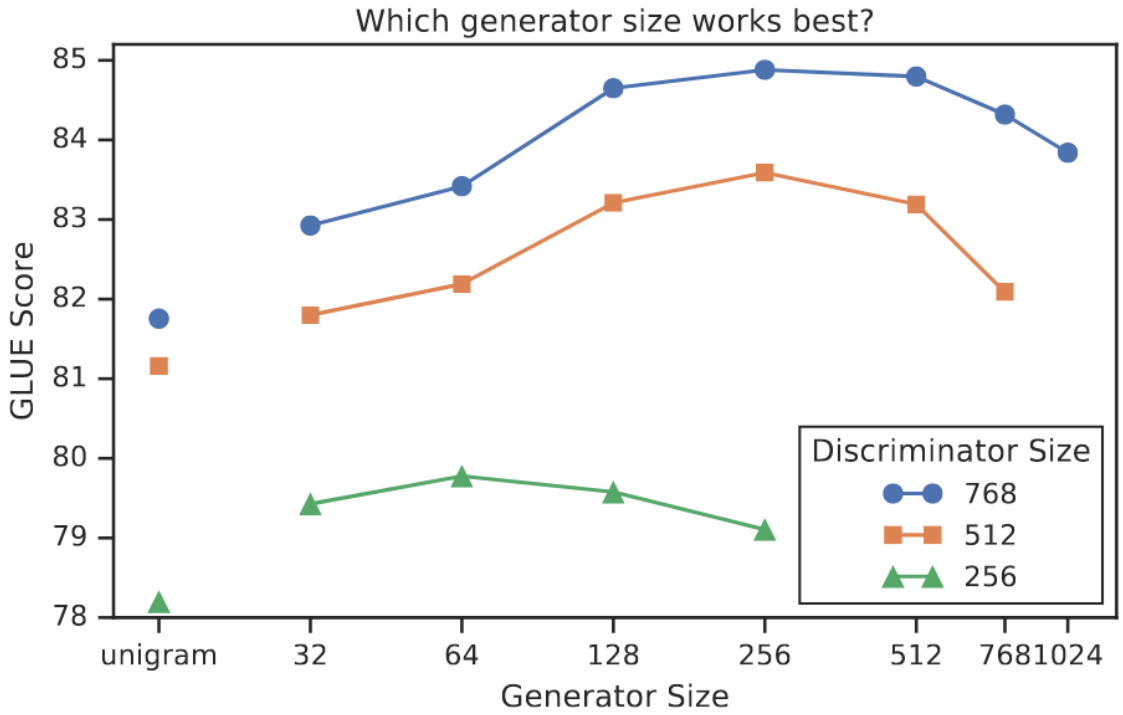
Маленький генератор + связь матриц эмбеддингов оказалась эффективнее всего.

Сравнение GLUE для моделей одинакового размера:

- No weight tying: 83.6
- Tying token embeddings: 84.3
- Tying all weights: **84.4**

Model extensions: smaller generators

Если генератор и дискриминатор одинакового размера, то тренировка ELECTRA занимает примерно в 2 раза больше времени за шаг, по сравнению с тренировкой только на MLM, поэтому авторы предлагают использовать маленькие генераторы.



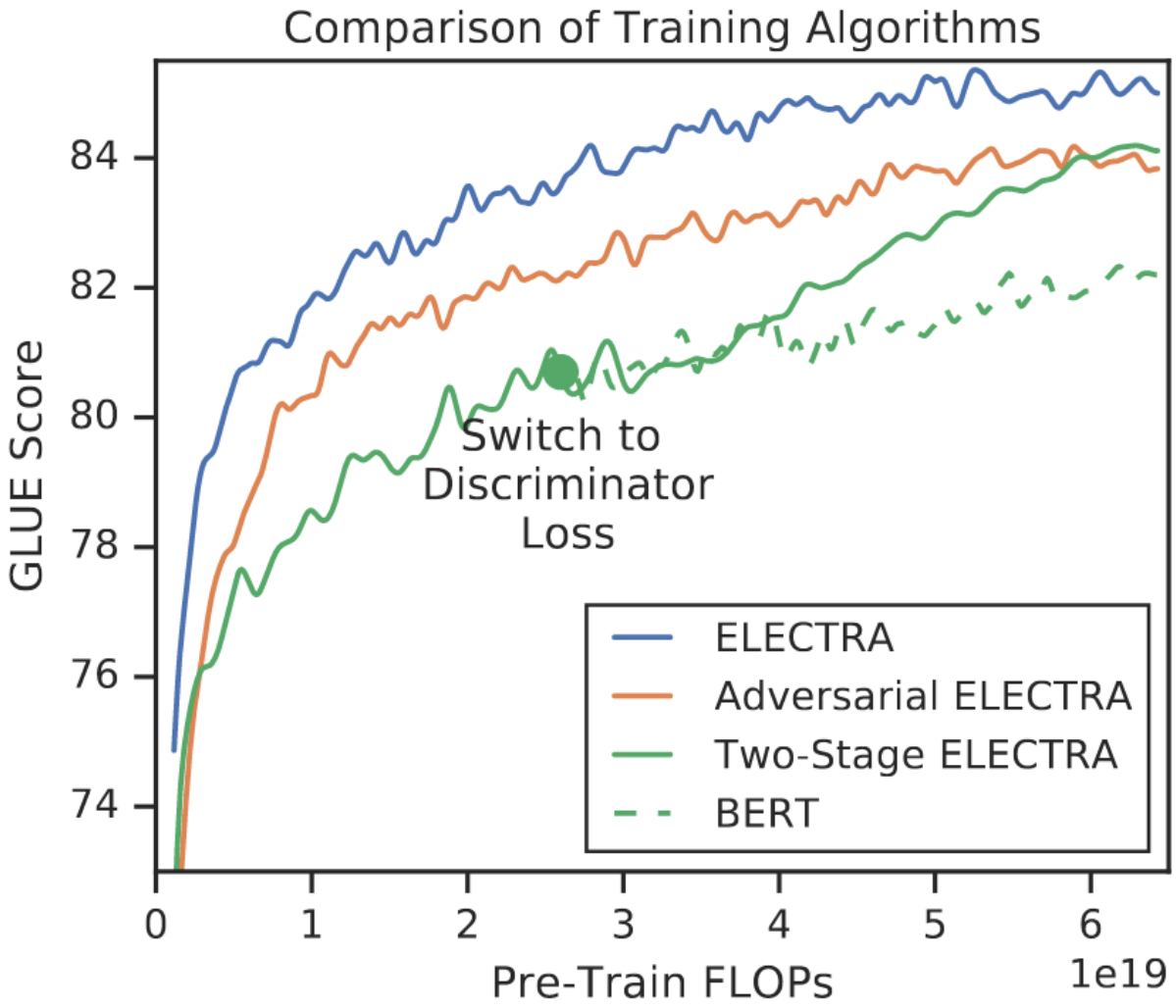
Model extensions: training algorithms

Авторы попробовали вместо одновременного обучения генератора и дискриминатора следующую двухступенчатую процедуры:

1. Тренируем только генератор n шагов
2. Инициализируем веса дискриминатора весами генератора. Тренируем дискриминатор n шагов, не трогая веса генератора

Model extensions: training algorithms

Но это не улучшило результат.



Результаты: ELECTRA-Small и ELECTRA-Base на GLUE

Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

Результаты: ELECTRA-Large на GLUE

Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	**97.1**	**91.2**	92.0	90.5	**91.3**	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	**97.1**	90.5	**92.6**	90.4	90.9	–	88.5	**92.5**	89.1	–
ELECTRA	3.1e21 (1x)	**71.7**	**97.1**	90.7	92.5	**90.8**	**91.3**	**95.8**	**89.8**	**92.5**	**89.5**	**89.4**

Table 3: GLUE test-set results for large models. Models in this table incorporate additional tricks such as ensembling to improve scores (see Appendix B for details). Some models do not have QNLI scores because they treat QNLI as a ranking task, which has recently been disallowed by the GLUE benchmark. To compare against these models, we report the average score excluding QNLI (Avg.*) in addition to the GLUE leaderboard score (Score). "ELECTRA" and "RoBERTa" refer to the fully-trained ELECTRA-1.75M and RoBERTa-500K models.

Результаты: ELECTRA-Base и ELECTRA-Large на SQuAD

Model	Train FLOPs	Params	SQuAD 1.1 dev		SQuAD 2.0 dev		SQuAD 2.0 test	
			EM	F1	EM	F1	EM	F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	–	–	–	–
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–	–	–
XLNet	3.9e21 (5.4x)	360M	**89.7**	**95.1**	87.9	**90.6**	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	–	94.0	–	87.7	–	–
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	–	–
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	–	–
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	–	–
ELECTRA-1.75M	3.1e21 (4.4x)	335M	**89.7**	94.9	**88.0**	**90.6**	**88.7**	**91.4**

("EM" – "exact match", percentage of predictions that match any one of the ground truth answers exactly; "-...K" – количество FLOPs для pre-training)