ELECTRA :
Pre-Training Text Encoders as Discriminators rather than Generators

QRAFT | AXE
GUIJIN.SON

2021.08.11

Computational Inefficiency of Past Models

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

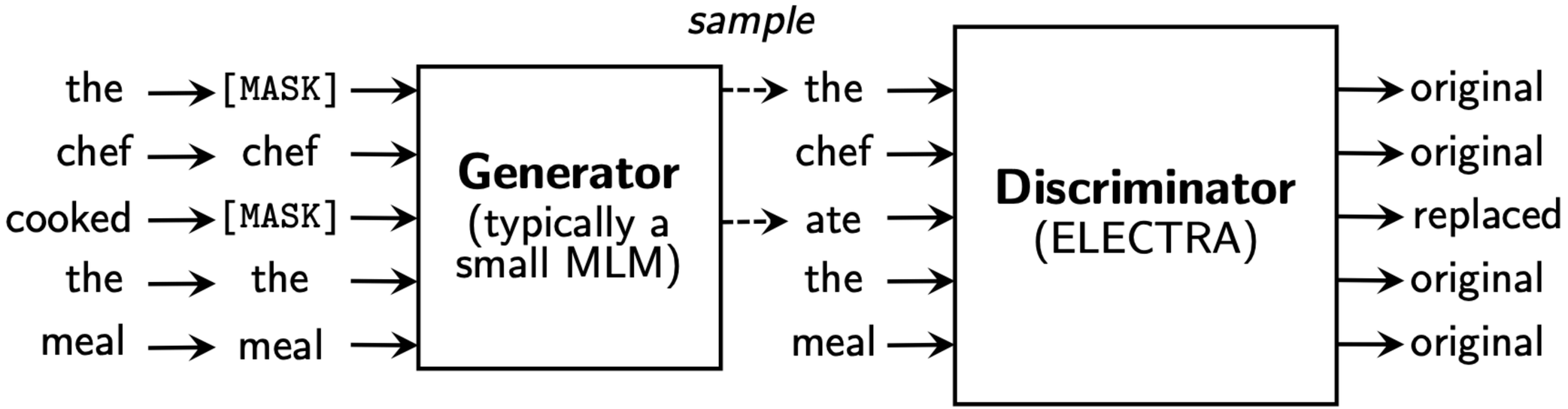
1. **PreTrain - FineTune Discrepancy : [MASK] Token**

2. **Inefficient Usage of Data**

> **“In all of our experiments, we mask 15% of all WordPiece to- kens in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than recon- structing the entire input. “**

Alternative : Replaced Token Detection

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators



1 INTRODUCTION

Current state-of-the-art representation learning methods for language can be viewed as learning denoising autoencoders (Vincent et al., 2008). They select a small subset of the unlabeled input sequence (typically 15%), mask the identities of those tokens (e.g., BERT; Devlin et al. (2019)) or attention to those tokens (e.g., XLNet; Yang et al. (2019)), and then train the network to recover the original input. While more effective than conventional language-model pre-training due to learning bidirectional representations, these masked language modeling (MLM) approaches incur a substantial compute cost because the network only learns from 15% of the tokens per example.

As an alternative, we propose *replaced token detection,* a pre-training task in which the model learns to distinguish real input tokens from plausible but synthetically generated replacements. Instead of masking, our method corrupts the input by replacing some tokens with samples from a proposal distribution, which is typically the output of a small masked language model. This corruption procedure solves a mismatch in BERT (although not in XLNet) where the network sees artificial `[MASK]` tokens during pre-training but not when being fine-tuned on downstream tasks. We then pre-train the network as a discriminator that predicts for every token whether it is an original or a replacement. In contrast, MLM trains the network as a generator that predicts the original identities of the corrupted tokens. A key advantage of our discriminative task is that the model learns from *all* input tokens instead of just the small masked-out subset, making it more computationally efficient. Although our

Model Details

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

Generator Output : $p_G(x_t|\boldsymbol{x}) = \exp\left(e(x_t)^T h_G(\boldsymbol{x})_t\right) / \sum_{x'} \exp\left(e(x')^T h_G(\boldsymbol{x})_t\right)$

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

$$m_i \sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k \qquad \boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]})$$

$$\hat{x}_i \sim p_G(x_i|\boldsymbol{x}^{\text{masked}}) \text{ for } i \in \boldsymbol{m} \qquad \boldsymbol{x}^{\text{corrupt}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

Loss Function : Formation

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

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

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

We minimize the combined loss

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

Loss Function : Implementation

```
with tf.variable_scope("discriminator_predictions"):
  hidden = tf.layers.dense(
      discriminator.get_sequence_output(),
      units=self._bert_config.hidden_size,
      activation=modeling.get_activation(self._bert_config.hidden_act),
      kernel_initializer=modeling.create_initializer(
          self._bert_config.initializer_range))
  logits = tf.squeeze(tf.layers.dense(hidden, units=1), -1)
  if self._config.electric_objective:
    log_q = tf.reduce_sum(
        tf.nn.log_softmax(cloze_output.logits) * tf.one_hot(
            inputs.input_ids, depth=self._bert_config.vocab_size,
            dtype=tf.float32), -1)
    log_q = tf.stop_gradient(log_q)
    logits += log_q
    logits += tf.log(self._config.mask_prob / (1 - self._config.mask_prob))

  weights = tf.cast(inputs.input_mask, tf.float32)
  labelsf = tf.cast(labels, tf.float32)
  losses = tf.nn.sigmoid_cross_entropy_with_logits(
      logits=logits, labels=labelsf) * weights
  per_example_loss = (tf.reduce_sum(losses, axis=-1) /
                      (1e-6 + tf.reduce_sum(weights, axis=-1)))
```

Isn’t this GAN?

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

1. “Real” and “Fake”
2. Adversarial Loss X | Maximum Likelihood O
3. No Noise Input

Model Extension : Weight Sharing

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

Weight Sharing We propose improving the efficiency of the pre-training by sharing weights between the generator and discriminator. If the generator and discriminator are the same size, all of the transformer weights can be tied. However, we found it to be more efficient to have a small generator, in which case we only share the embeddings (both the token and positional embeddings) of the generator and discriminator. In this case we use embeddings the size of the discriminator's hidden states.[4] The "input" and "output" token embeddings of the generator are always tied as in BERT.

We compare the weight tying strategies when the generator is the same size as the discriminator. We train these models for 500k steps. GLUE scores are 83.6 for no weight tying, 84.3 for tying token embeddings, and 84.4 for tying all weights. We hypothesize that ELECTRA benefits from

Performance

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

Does Pretrain-FineTune Discrepancy Exist?

ELECTRA: Pre-Training Text Encoders as Discriminators rather than Generators

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 8: Ablation over different masking strategies.

100% RND < 80% MASK