Exploring the limits of Transfer learning with a Unified Text-to-Text Transformer

2.1. Model

- roughly equivalent to the original Transformer
- removing the Layer Norm bias
- placing the layer normalization outside the residual path
- using different position embedding scheme.

2.4. Input & Output Format

- "text-to-text" format
- provides a consistent training objective both for pre-training & fine-tuning

3.1. Baseline

- to reflect modern, typical practice
- pre-train a standard Transformer using a simple denoising objective
- separately fine-tune on each of downstream task.

3.1.2 Training

- "inverse square root" learning rate schedule

$$1 / \sqrt{\max(n, k)}$$

(n: current training iteration; k: number of warm-up steps (set to 10^4 in all experiments))

  - constant learning rate of 0.01 for the first 10^4 steps
  - then exponentially decays the learning rate until pre-training is over.

3.1.4. Unsupervised Objective

- mask consecutive spans of tokens & only predict dropped-out tokens
  → reduce computational cost of pre-training

3.1.5. Baseline Performance

- pre-training provides significant gains across almost all benchmarks
- inter-run variance
  - GLUE & SuperGLUE : average of scores of each benchmark
  - CoLA, CB, COPA : ↑ inter-run variance...
  - harder to compare models using GLUE & SuperGLUE scores alone.

3.2 Architectures

3.2.1. Model Structures

- major distinguishing factor for different architectures

  : "mask" used by different attention mechanisms in the model.

- Transformer

  - input : sequence
  - output : a new sequence of the same length
    - each entry of output sequence is produced by computing a weighted average of entries of the input sequence

    - $Y_i = \sum_j W_{i,j} X_j$

      Y_i : i^{th} element of the output sequence

      $W_{i,j}$: scalar weight self-attention mechanism as a function of X_i & X_j

      X_j : j^{th} entry of the input sequence

  - attention mask is used to zero out certain weights in order to constrain which entries of the input can be attended to at a given output timestep.

① encoder-decoder Transformer

- encoder :
  - fully-visible attention mask
  - allows a self-attention mechanism to attend to any entry of the input when producing each entry of its output.
  - BERT's output at the timestep corresponding to the classification token is then used to make a prediction for classifying the input sequence.

- decoder :
  - causal masking pattern
  - used to autoregressively produce an output sequence
  - language models can learn to perform some text-to-text tasks without supervision.
  - drawback of causal masking in the text-to-text setting

    → avoided by changing the masking pattern.

3.2.2. Comparing Different Model Structure

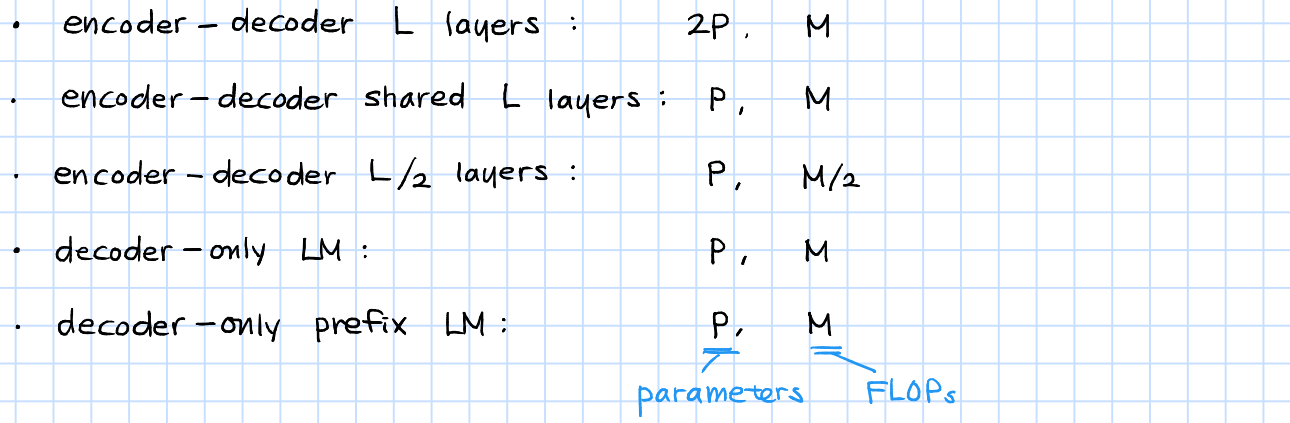
- equivalent in some meaningful way : same number of parameters, same amount of computation
- hard! why?

  ↓

  · encoder – decoder model $\overset{\text{parameters}}{=\!=}$ Language Model

  L layers L layers 2L layers

  $\overset{\text{computation}}{=\!=}$



⇒ 2L encoder – decoder $=\!=$ L layer Language Model

- Comparing different model structures

  · encoder – decoder L layers : 2P, M

  · encoder – decoder shared L layers : P, M

  · encoder – decoder L/2 layers : P, M/2

  · decoder – only LM : P, M

  · decoder – only prefix LM : P, M

  parameters, FLOPs

3.2.3. Objectives

- for models that ingest a prefix before making predictions :

  (encoder – decoder model & prefix LM)

  · sample a span of text from unlabeled data set

  · choose a random point to split it into prefix and target points.

- for standard language model :

  · train the model to predict the entire span from beginning to end.

3.2.4. Results

- shared parameter encoder – decoder outperforms the decoder – only prefix LM
- explicit encoder – decoder attention is beneficial.

3.3.1. Disparate High – Level Approaches

① prefix language modeling

- splits a span of text into two components : input to the encoder / target sequence predicted by the decoder
- Example :

  Thank you for inviting me to your party last night.

  inputs targets

② masked language modeling (MLM) objected used in BERT

- takes a span of text & corrupts 15% of the tokens
- 90% of corrupted tokens : replaced with a special mask token
- 10% " : replaced with a random token
- BERT goal: reconstruct masked tokens at the output of the encoder.
- encoder – decoder case : use the entire uncorrupted sequence as target
- Example :

Thank you <M> <M> me to your party apple week. (original text)

inputs target

③ Deshuffling objective

- denoising sequential autoencoder
- takes a sequence of tokens, shuffles it, uses the original deshuffled sequence as a target
- Example :

party me for your to. last fun you inviting week Thank (original text)

inputs target

3.3.4. Corrupting Spans

- specifically corrupt contiguous, randomly-spaced spans of tokens.
- Example :
  - a sequence of 500 tokens, 15% corrupted, 25 total spans
    ↓
  - total number of corrupted tokens : 500 x 0.15 = 75
  - average span length : 75 / 25 = 3

→ given the original sequence length & corruption rate,
we can equivalently parametrize this objective
by the average span length or total number of spans.