Light Language Pre-Training

Taking Notes on the Fly Helps Language Pre-Training, ICLR 2021
Rethinking Positional Encoding in Language Pre-training, ICLR 2021

Presented by Qiyu Wu
Machine Learning Group, MSRA

Microsoft

Motivation

- Pretraining (e.g., BERT) plays a critical role in NLP tasks
- However, the computational cost of pretraining is very high

	Model Parameter	#Tokens in training	GPU days (on V100)	Cost
GPT	117M	32B	~12.5	$825
BERT-Base	110M	131B	~50	$3,300
BERT-Large	335M	131B	~150	$9,900
XLNET	360M	524B	~600	$39,600
RoBERTa	356M	2000B	~2,400	$158,400
GPT-2	1,500M	520B	~2,500	$165,000
T5	11,000M	1000B	~15,000	$990,000
GPT-3	175,000M	300B	~178,000	$11,687,500

- Barrier for research and product development

Cost calculation is based on Azure ND40v2, which has 8 V100 GPUs and costs $22 per hour.
Besides, if considering the distributed overhead, the actual costs will be much larger.

Break the Curse Through the "Machine Learning" Glasses

Training Objective

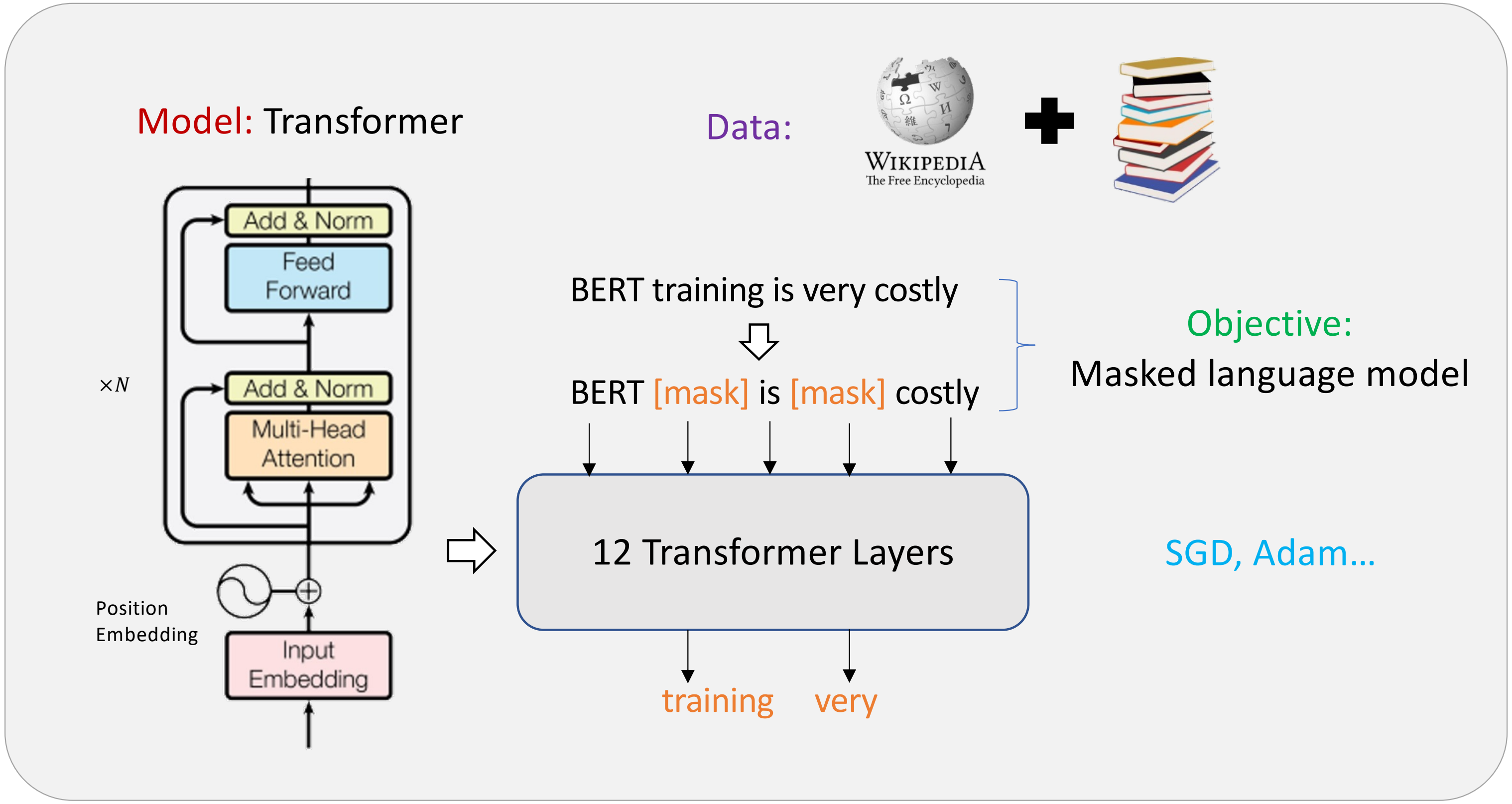
Optimization

$$\omega^* = \arg\min_{\omega \in \boldsymbol{\Omega}} \sum_{\substack{x_i \in X, y_i \in Y \\ (x_i, y_i) \sim P}}^{i=1,\ldots,N \to \infty} L(f_\omega(x_i), y_i)$$

Training Data

Model Architecture

BERT, as an Example



Computational Cost for Pretraining

Total Cost

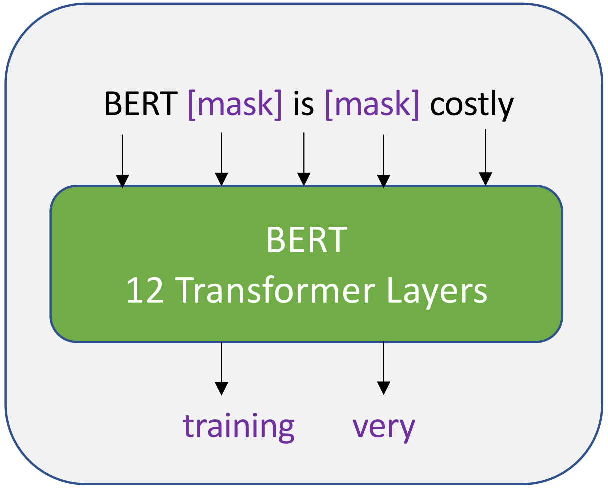
Model Training Cost

- The cost of learning a model from training data, given the current hyperparameters
- Depends on data quality/utilization, model capacity, self-supervised signal, and optimization strategy

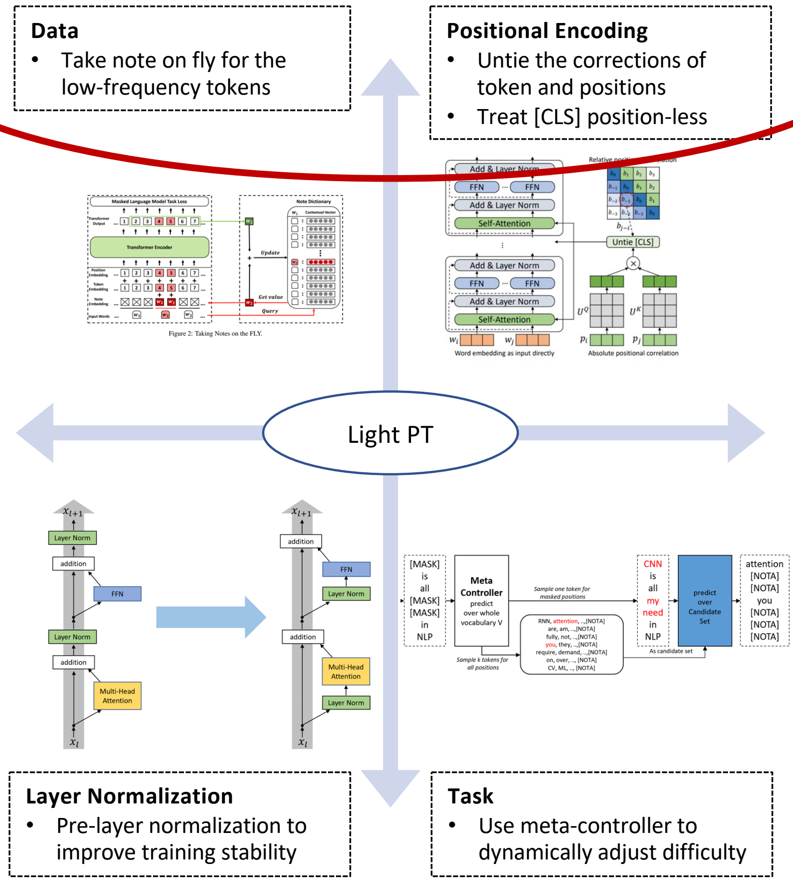
Hyperparameter Tuning Cost

- The cost of hyperparameter tuning, to select a model with good performance on validation data
- Related to the model robustness, and training instability

Holistic Solution for Algorithmic Acceleration



The pretraining cost of BERT very high. Requires several weeks to train a model by tens of GPUs



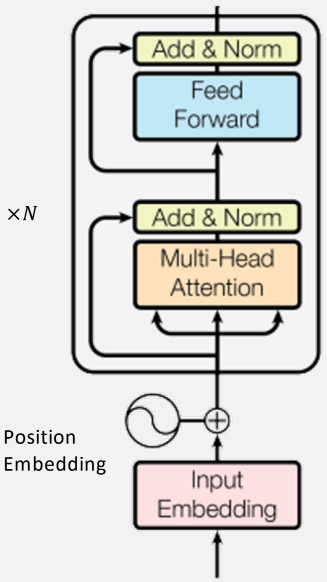
Taking Notes on the Fly Helps Language Pre-Training

ICLR 2021

Microsoft

Quality of Word Embeddings

Model: Transformer



Data: Wikipedia The Free Encyclopedia ✚

BERT training is very costly

⇩

BERT [mask] is [mask] costly

12 Transformer Layers

training very

Objective: Masked language model

SGD, Adam...

Word embedding is the input to the Transformer model.

Word embedding is optimized together with the model parameters, using gradient descent.

Do All Word Embeddings Have Good Quality?

- No. The embeddings of <u>low-frequency words</u> have low quality.
  - Rare words appear/update infrequently using gradient-descent approaches
  - The phenomena are observed in many practical scenarios, e.g., Transformer, LSTM, word2vec, Glove. [Bahdanau et al., 2017; Gong et al., 2018; Khassanov et al., 2019; Schick & Schutze, 2020.]
- If rare word embeddings are like noise, it hurts the training efficiency of other model parameters.

A Motivating Example

COVID-19 has cost thousands of _(lives)_ .

What is COVID-19?

dollars?
donuts?
puppies?
....tomatoes?

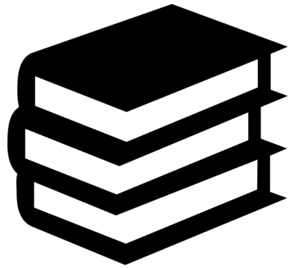
The embedding of COVID-19 is poor and contains much noise.

The model is slow to learn with very noisy input.

Training may be inefficient.

How to Treat Rare-Word Signals Better?

Thinking about we have a dictionary at hand.



When we meet some word that we don't know, we look it up from the dictionary and get its meaning by popular word sentence.

Dictionary helps us understand the sentence.

Improving the Representation of Rare Words

With Notes:

COVID-19 has cost thousands of lives .

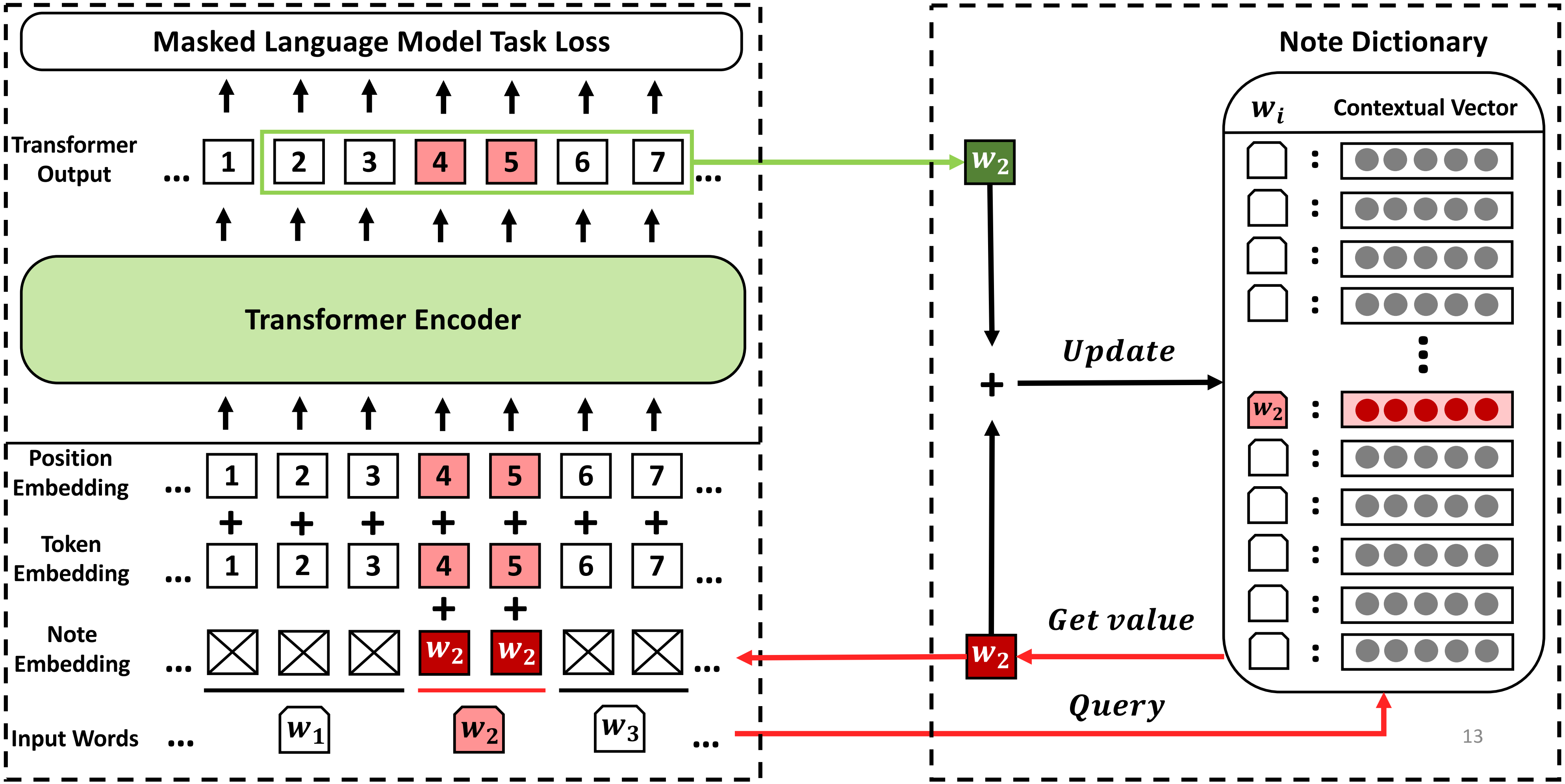
Pandemic;
global crisis

Note of COVID-19 that is taken before:
The COVID-19 pandemic is an ongoing global crisis.

A note dictionary saves historical contextual information of rare words.

Notes are used to enrich the understanding of the current sentence .

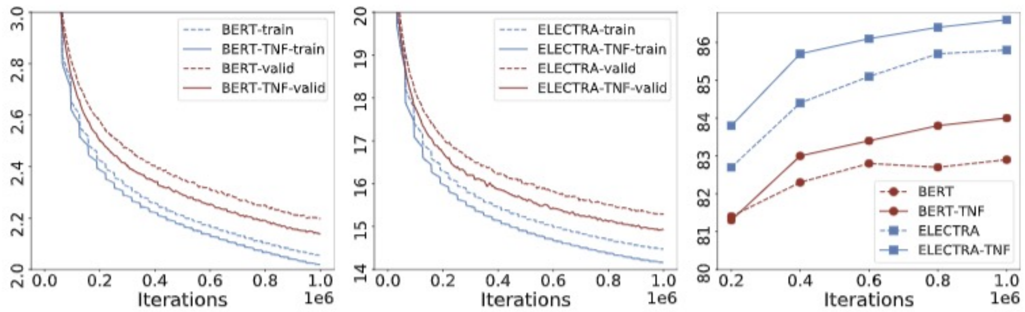
Transformer receives more accurate inputs and the training is efficient



Masked Language Model Task Loss
Transformer Output
Transformer Encoder
Position Embedding
Token Embedding
Note Embedding
Input Words
Note Dictionary
w_i
Contextual Vector
Update
Get value
Query

Results (See more BERT-base/large/ELECTRA perf in the paper)

	Params	Avg. GLUE
GPT-2	117 M	78.8
BERT	110 M	82.2
SpanBERT	110 M	83.9
ELECTRA	110 M	85.1
BERT (Ours)	110 M	83.1
BERT-TNF	110 M	83.9
ELECTRA (Ours)	110 M	86.0
ELECTRA-TNF	110 M	**86.7**



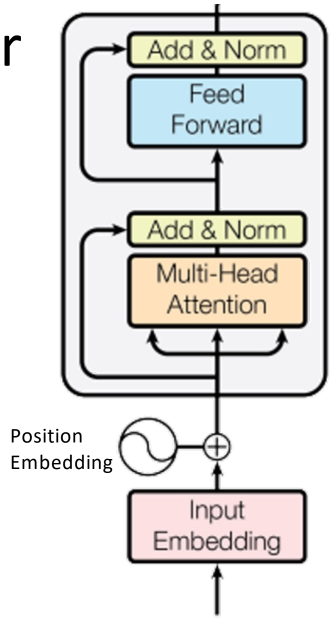
(a) Loss curves (BERT setting) (b) Loss curves (ELECTRA setting) (c) GLUE evaluation

Rethinking Positional Encoding in Language Pre-training

ICLR 2021

Positional Encoding

- Positional encoding is an important component in Transformer
  - Absolute positional encoding
    - An embedding vector p_i for each position i
    - $p_i + w_i$ is used as the input (w_i is the word embedding)
  - Relative positional encoding
    - An embedding vector p_{i-j} for each position i, j
    - Put inside the self-attention module



Rethinking Absolute Positional Encoding

- Is word + positional embedding reasonable?
  - Word embedding encodes word semantic.
  - Positional embedding encodes ``index''.
  - What can we obtain when we add this two heterogenous terms together?

- To answer the question above, we expand the self-attention calculation in the first layer

Rethinking Absolute Positional Encoding

$$\alpha_{ij} = \frac{\left((w_i + p_i)W^Q\right)\left((w_j + p_j)W^K\right)^T}{\sqrt{d}}$$

$$= \frac{(w_i W^Q)(w_j W^K)^T}{\sqrt{d}} + \frac{(w_i W^Q)(p_j W^K)^T}{\sqrt{d}} + \frac{(p_i W^Q)(w_j W^K)^T}{\sqrt{d}} + \frac{(p_i W^Q)(p_j W^K)^T}{\sqrt{d}}$$

word-word, position-position correlation

Words and positions are heterogeneous information.
It is not proper to use shared parameters (projections)

α_{ij} -- attention from position i to position j

W^Q, W^K -- parameters

Rethinking Absolute Positional Encoding

$$\alpha_{ij} = \frac{\left((w_i + p_i)W^Q\right)\left((w_j + p_j)W^K\right)^T}{\sqrt{d}}$$

$$= \frac{(w_iW^Q)(w_jW^K)^T}{\sqrt{d}} + \frac{(w_iW^Q)(p_jW^K)^T}{\sqrt{d}} + \frac{(p_iW^Q)(w_jW^K)^T}{\sqrt{d}} + \frac{(p_iW^Q)(p_jW^K)^T}{\sqrt{d}}$$

word-position, position-word correlation

In language pre-training, multiple sentences are patched into one sequence, then the position and word have very weak correlations.

α_{ij} -- attention from position i to position j

W^Q, W^K -- parameters

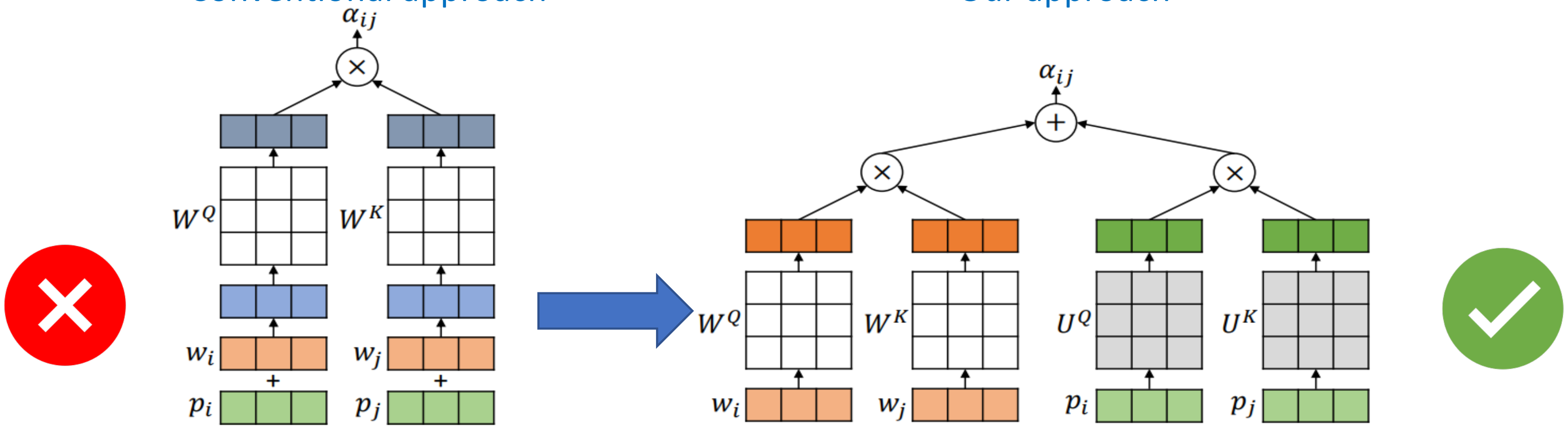
Our Modification I - Dealing with Absolute Positional Encoding

Ours: $\alpha_{ij} = \frac{(w_i W^Q)(w_j W^K)}{2\sqrt{d}} + \frac{(p_i U^Q)(p_j U^K)}{2\sqrt{d}}$

- Remove the two noisy terms in the middle
- Use different parameters to calculate word/position pair correlations

Conventional approach

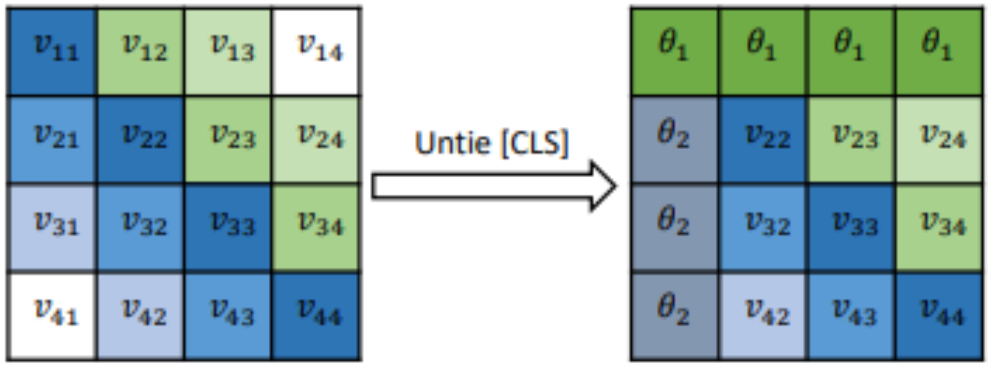
Our approach



(a) Absolute positional encoding.

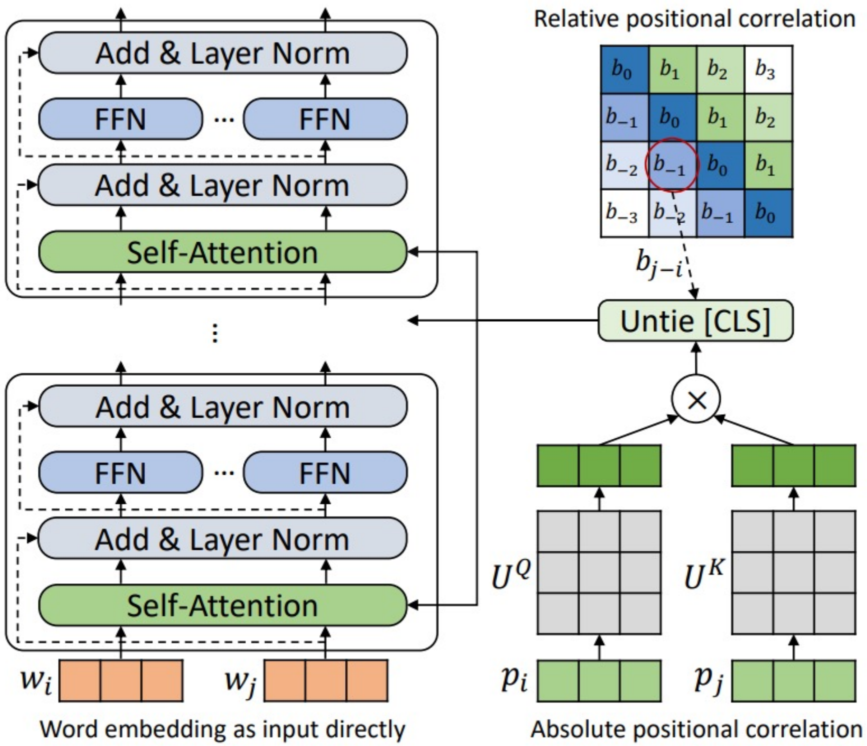
(b) Untied absolute positional encoding.

Our Modification II - Dealing with [CLS] Token

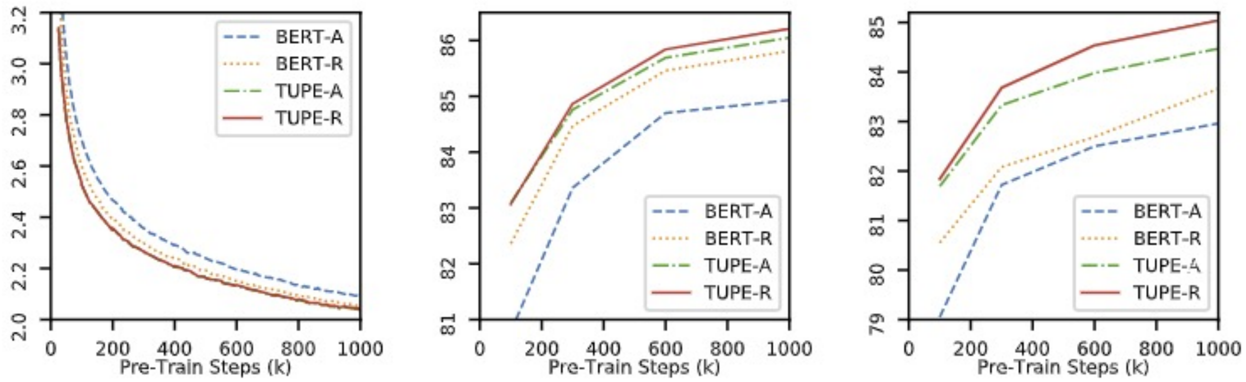


- [CLS] position summarizes the information of the whole sentence. It should be treated specifically compared to other natural words.
- Besides untying the word-position correlation, we also untie the [CLS] and natural word positions.
- We call our method TUPE (Transformer with Untied Positional Encoding)

TUPE (Transformer with Untied Positional Encoding)



Results (See more BERT-base/large/ELECTRA perf in the paper)



(a) Validation loss in pre-training. (b) MNLI-m score. (c) GLUE average score.

Figure 5: Both TUPE-A and TUPE-R convergence much faster than baselines, and achieve the better performance in downstream tasks while using much fewer pre-training steps.

Thank You！

Q&A