BERT

Pretraining of Deep Bidirectional Transformers for Language Understanding

성주용, Causality Lab, SNU

3월 20일, 2025년

Abstract

- Bert is designed to pretrain deep bidirectional representations from **unlabeled text** by jointly conditioning on both left and right context in all layers.
  - Previous models relied on labeled text, which required significant time and had limited data.
  - By using unlabeled text, they were able to leverage a vast amount of training data and achieve better model representation.

Introduction

Natural language processing task

- Sentence-level tasks
  - Natural language inference
    - 참/거짓 판별
  - Paraphrasing
    - 같은 의미의 다른 문장을 생성

- Token-level tasks
  - Named entity recognition
    - 텍스트에서 특정 카테고리에 속하는 단어나 구를 식별하고 분류
  - Question answering

Introduction

Two existing strategies

- ► Feature-based
  - Use task-specific architecture

- ► Fine-tuning
  - Introduce minimal task-specific parameters
  - Trained on the downstream tasks by simply fine-tuning all pretrained parameters

Use **unidirectional language models** to learn general language representations
=> Restrict the power of the pre-trained representations

Introduction

Unidirectional drawback

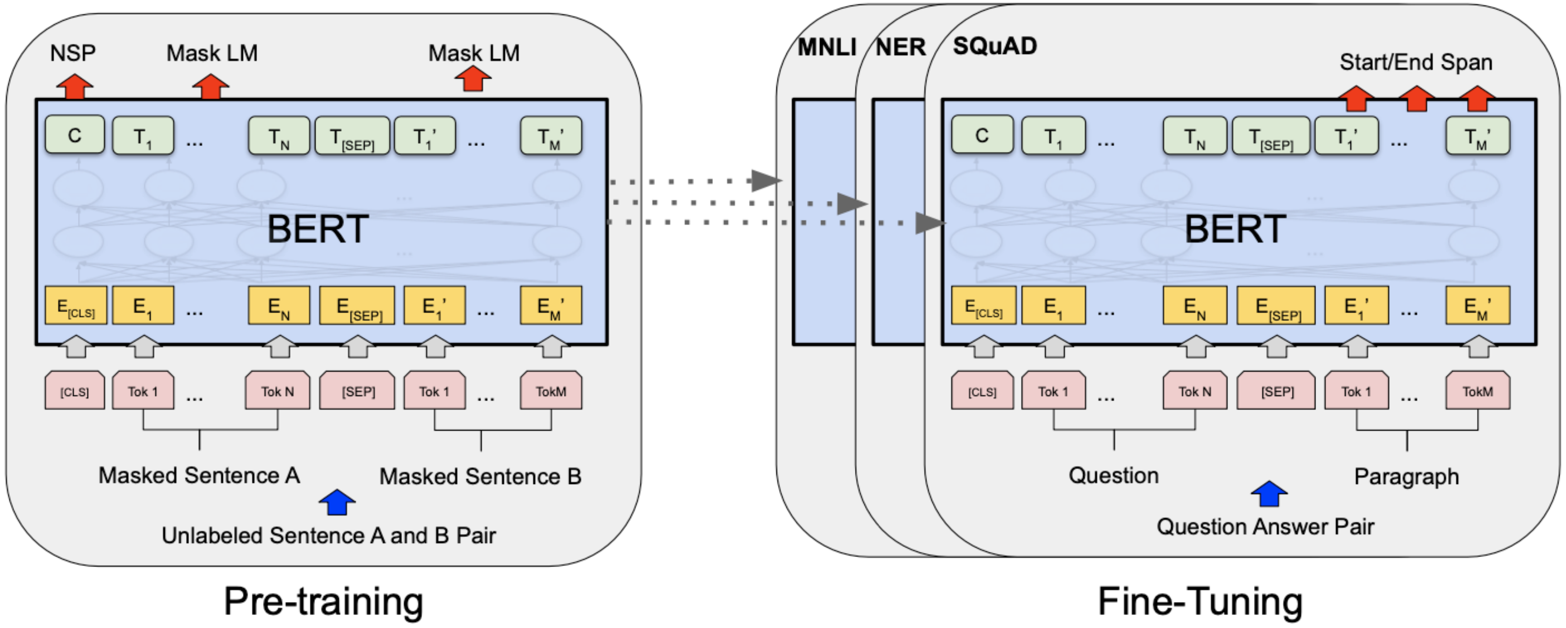
Use **unidirectional language models** to learn general language representations
$\Rightarrow$Restrict the power of the pre-trained representations

- ► The bank approved the loan.

- ► He sat by the bank and watched the river flow.

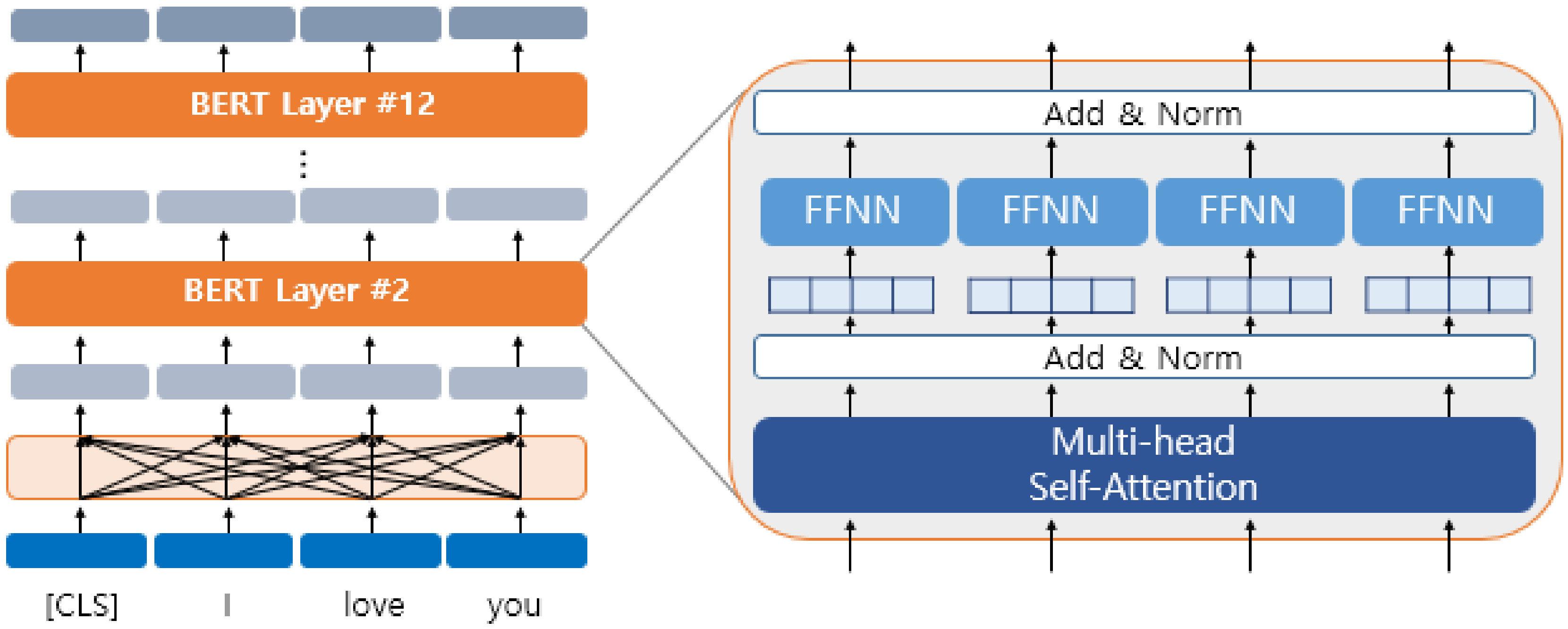
BERT

Architecture



BERT

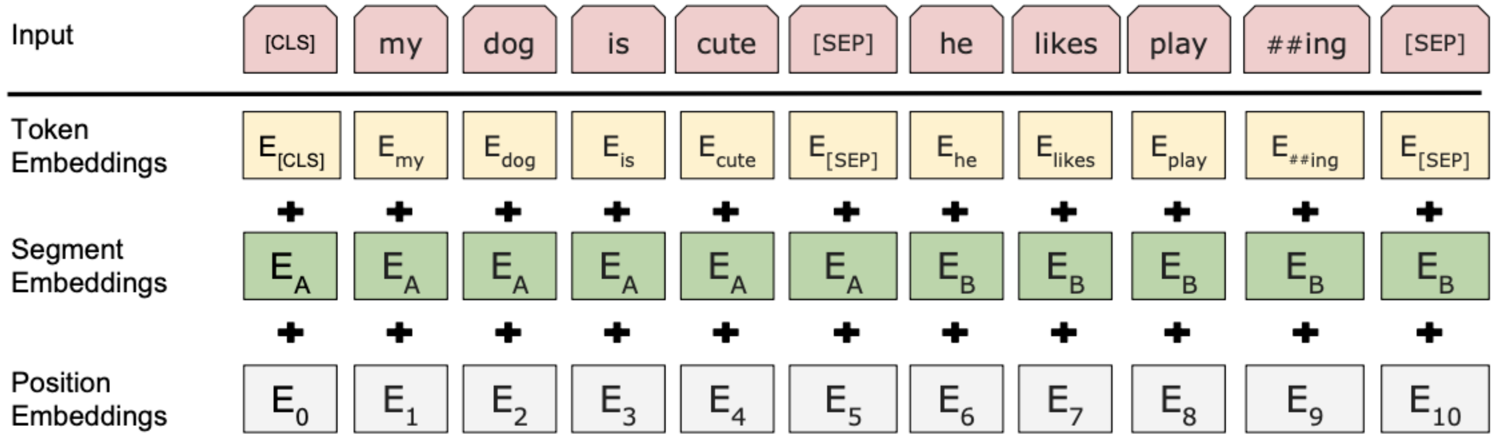
Architecture



BERT

Input/Output Representation

- Input
  - Word Piece embeddings as token embeddings.
  - First token of every sequence is always a [CLS]
    - Used as the aggregate sequence representation for classification tasks.
  - Sentence pairs are separated with [SEP] and introduce Segment embeddings.



BERT

Input/Output Representation

► Word Piece embedding

- Word Piece embeddings as token embeddings.

```
공연은 끝났어 -> ['공연-' + '-은' + '끝-' + '-났어']
공연을 끝냈어 -> ['공연-' + '-을' + '끝-' + '-냈어']
개막을 해냈어 -> ['개막-' + '-을' + '해-' + '-냈어']
```

수행하기 이전의 문장: Jet makers feud over seat width with big orders at stake

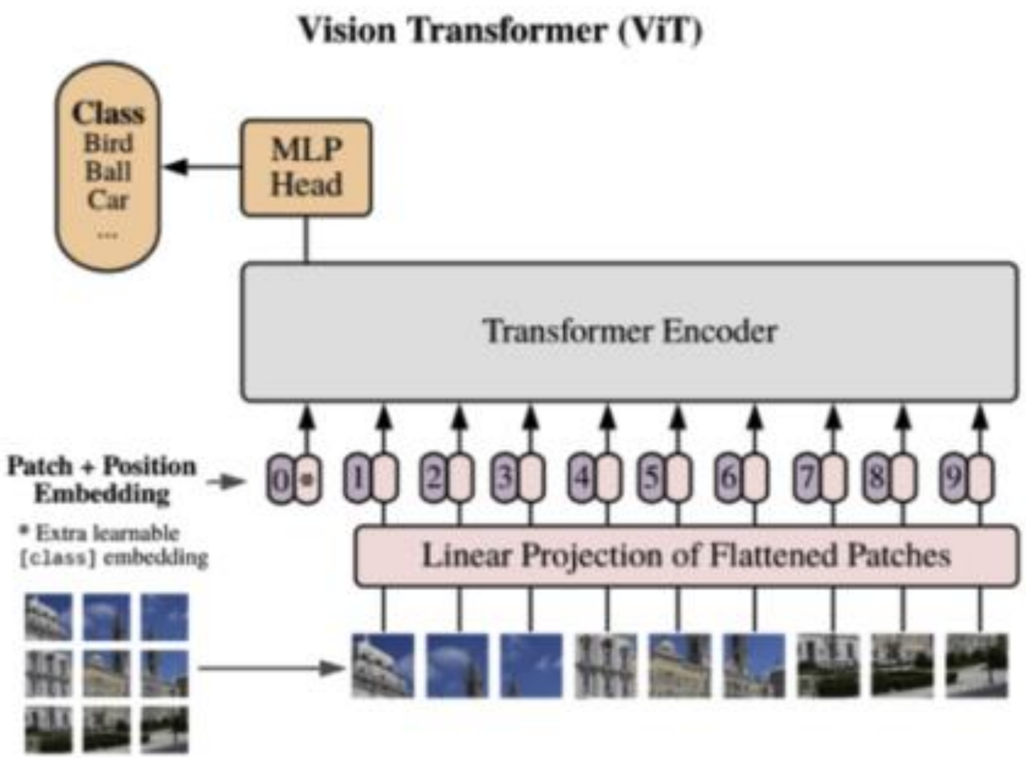
WordPiece Tokenizer를 수행한 결과(wordpieces): _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

BERT

Input/Output Representation

► CLS token

- First token of every sequence is always a [CLS]
  - Used as the aggregate sequence representation for classification tasks.

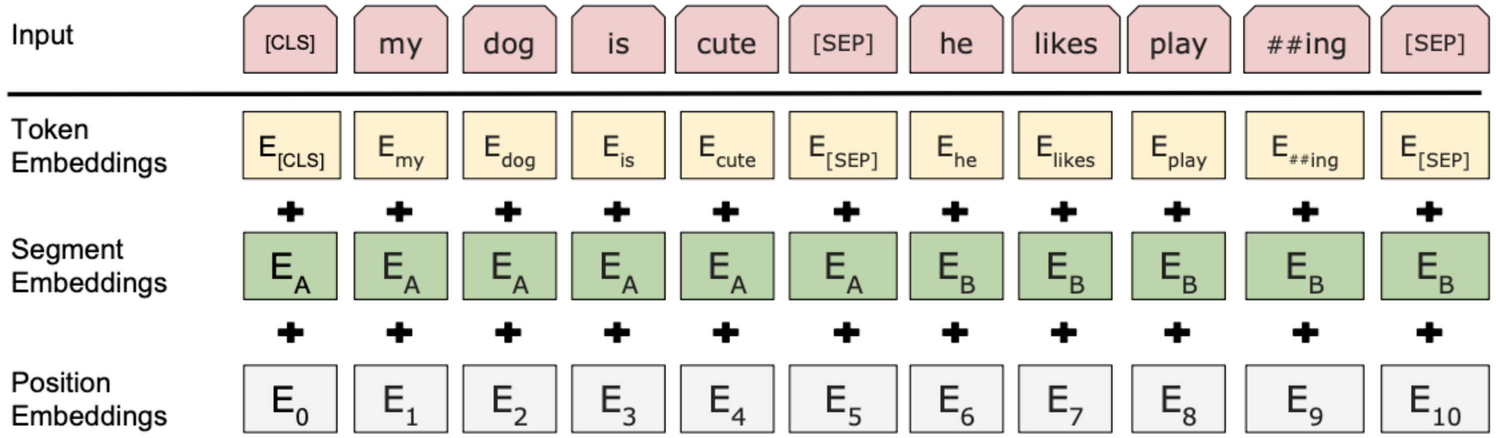


BERT

Input/Output Representation

► Input

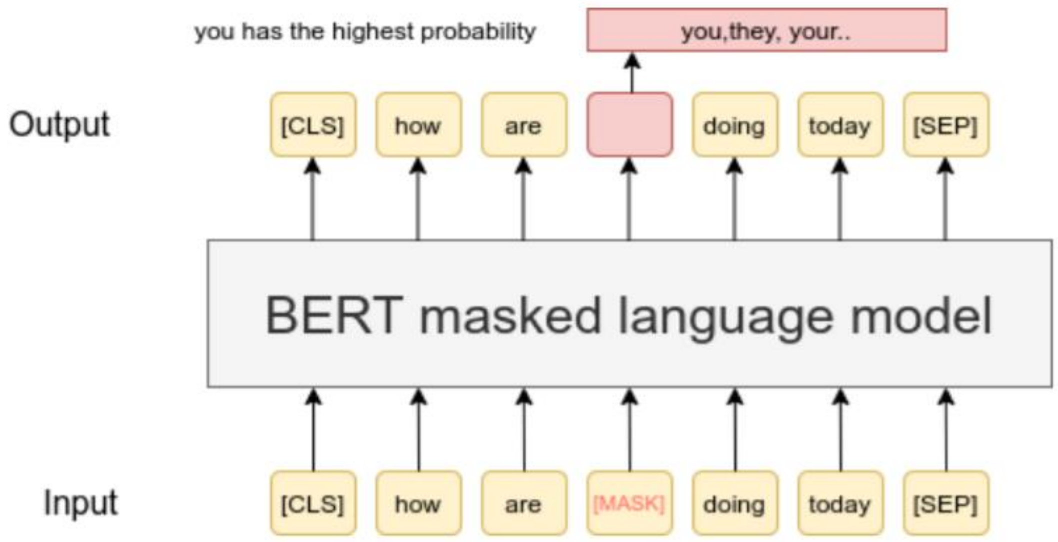
- Word Piece embeddings as token embeddings.
- First token of every sequence is always a [CLS]
  - Used as the aggregate sequence representation for classification tasks.
- Sentence pairs are separated with [SEP] and introduce Segment embeddings.



BERT

Masked language model

- ► Simply mask some percentage(15%) of the input tokens at random and predict.
- ► Final hidden vectors are fed into an output softmax.



문 1. 빈칸에 들어갈 표현으로 가장 적절한 것은?

> If you ________ when you are driving, it means that you stop.

① go through ② put off
③ pull over ④ get over

문 2. 빈칸에 들어갈 단어로 가장 적절한 것은?

> Journalists must be ________. For instance, they must be good at writing, listening to people, speaking, working quickly, and doing research.

① factual ② contemporary
③ extensive ④ versatile

BERT

Masked language model

```
def get_masked_lm_output(bert_config, input_tensor, output_weights, positions,
                         label_ids, label_weights):
  """Get loss and log probs for the masked LM."""
  input_tensor = gather_indexes(input_tensor, positions)

  output_bias = tf.get_variable(
      "output_bias",
      shape=[bert_config.vocab_size],
      initializer=tf.zeros_initializer())
  logits = tf.matmul(input_tensor, output_weights, transpose_b=True)
  logits = tf.nn.bias_add(logits, output_bias)
  log_probs = tf.nn.log_softmax(logits, axis=-1)
```

BERT

Masked language model

```
  label_ids = tf.reshape(label_ids, [-1])
  label_weights = tf.reshape(label_weights, [-1])

  one_hot_labels = tf.one_hot(
      label_ids, depth=bert_config.vocab_size, dtype=tf.float32)



  per_example_loss = -tf.reduce_sum(log_probs * one_hot_labels, axis=[-1])
  numerator = tf.reduce_sum(label_weights * per_example_loss)
  denominator = tf.reduce_sum(label_weights) + 1e-5
  loss = numerator / denominator

return (loss, per_example_loss, log_probs)
```

BERT

Next Sentence Prediction

- 50% actual next sentence,
  50% random sentence.

- C (CLS) is used for NSP task.

BERT

Next Sentence Prediction

```
def get_next_sentence_output(bert_config, input_tensor, labels):
  """Get loss and log probs for the next sentence prediction."""

  # Simple binary classification. Note that 0 is "next sentence" and 1 is
  # "random sentence". This weight matrix is not used after pre-training.
  with tf.variable_scope("cls/seq_relationship"):
    output_weights = tf.get_variable(
        "output_weights",
        shape=[2, bert_config.hidden_size],
        initializer=modeling.create_initializer(bert_config.initializer_range))
    output_bias = tf.get_variable(
        "output_bias", shape=[2], initializer=tf.zeros_initializer())

    logits = tf.matmul(input_tensor, output_weights, transpose_b=True)
    logits = tf.nn.bias_add(logits, output_bias)
    log_probs = tf.nn.log_softmax(logits, axis=-1)
    labels = tf.reshape(labels, [-1])
    one_hot_labels = tf.one_hot(labels, depth=2, dtype=tf.float32)
    per_example_loss = -tf.reduce_sum(one_hot_labels * log_probs, axis=-1)
    loss = tf.reduce_mean(per_example_loss)
    return (loss, per_example_loss, log_probs)
```

Experiment

GLUE

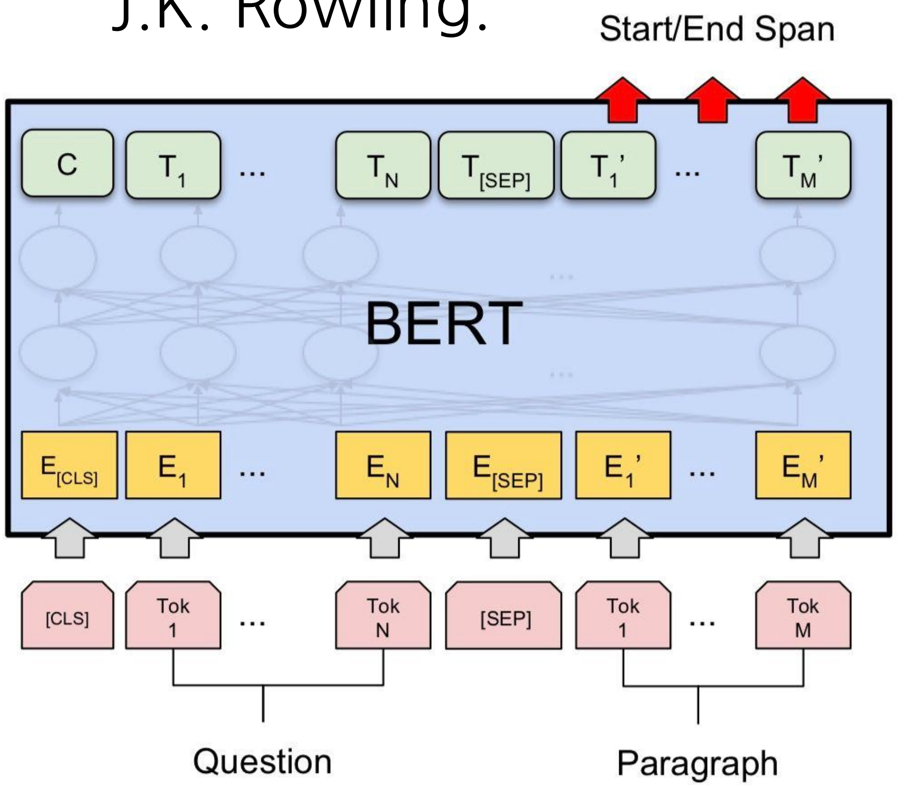
GLUE (General Language Understanding Evaluation) Benchmark Tasks:

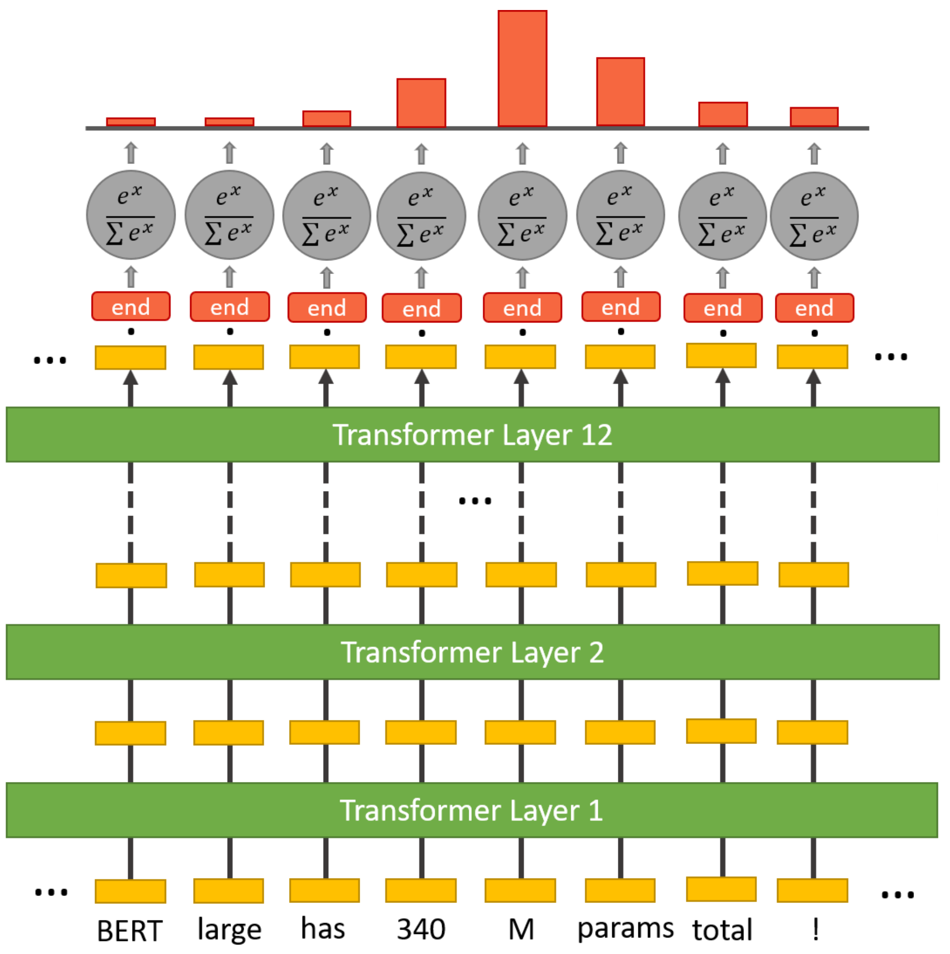
Task	Example	Dataset	Metric
Grammatical	"This toast is than that one." **= Ungrammatical**	CoLA	Matthews
Sentiment Analysis	"Toy Story 2 was okay." **= .543291 (neutral)**	SST-2	Accuracy
Similarity	a.) A pride of lions surrounded a monkey. b.) Lions encompassed a monkey. **= 4.7 (Very Similar)**	STS-B	Person / Spearman
Paraphrase	A. Last week, Seattle reported 12 new earthquakes. B. Seattle reported another 12 earthquakes yesterday. **= A Paraphrase**	MRPC	Accuracy / F1
Question Similarity	a.) How can I cook noodles over a campfire? b.) How do you make Mac & Cheese? **= Not Similar**	QQP	Accuracy / F1
Contradiction	a.) Glossier products are the best! b.) Glossier products are overpriced. **= Contradiction**	MNLI-mm	Accuracy
Answerable	a.) How does the Dyson Airwrap work? b.) The Airwarp uses the Coanda effect to create a vortex pulling the hair towards the attachments. **= Answerable**	QNLI	Accuracy
Entail	a.) In 2006, Paul David bought a Microprocessing center to create 30,000 jobs in Northern Minnesota. b.) Paul David created 30,000 jobs in MN. **= Entail**	RTE	Accuracy
Ambiguous pronouns	a.) Federico spoke to Marie, breaking her focus. b.) Federico spoke to Marie, breaking Federico's focus. **= Incorrect Referent**	WNLI	Accuracy

Experiment

SQuAD

- “Who wrote the Harry Potter series?”
- “The Harry Potter was written by J.K. Rowling.”





Ablation Studies

Effect of Pre-training Tasks

- ► No NSP : MLM o, NSP x

- ► LTR & NO NSP : MLM x (Left-to-Right), NSP x
  - Comparable to OPENAI GPT

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Ablation Studies

Effect of Model Size

- Larger model, better performance
- Task-specific models can benefit from the larger, more expressive pre-trained representations.

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Ablation Studies

Feature-based Approach with BERT

- Not all tasks can be easily represented by a Transformer encoder architecture, require a task-specific model architecture to be added.
- There are major computational benefits.
  - By using cheaper models on top of BERT representation.

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

End of Document