Lucerne University
of Applied Sciences
and Arts

Pretraining and BERT

NLP
Andreas Marfurt

Information Technology
17.04.2025

FH Zentralschweiz

Motivation

In 2019, Google announced that it had begun leveraging BERT in its search engine, and by late 2020 it was using BERT in almost every English-language query. A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model.

All kinds of adaptations:

- ALBERT
- SentenceBERT
- SpanBERT
- CamemBERT
- BioBERT

Overview

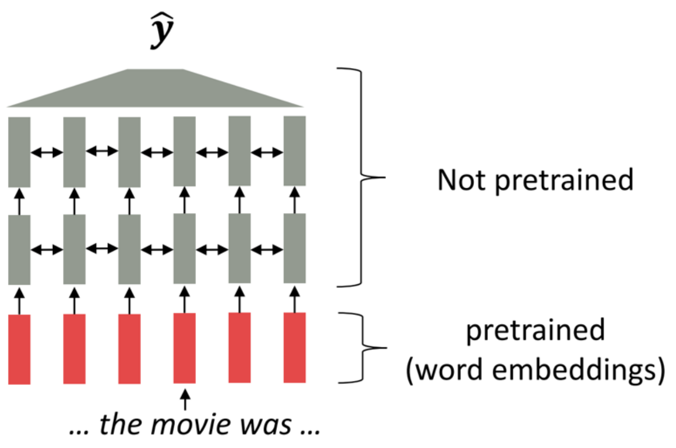
- Pretraining
  - Self-supervision
- BERT
  - Pretraining objectives
  - Segment embeddings
  - Training
  - Tasks
  - Results

Pretraining

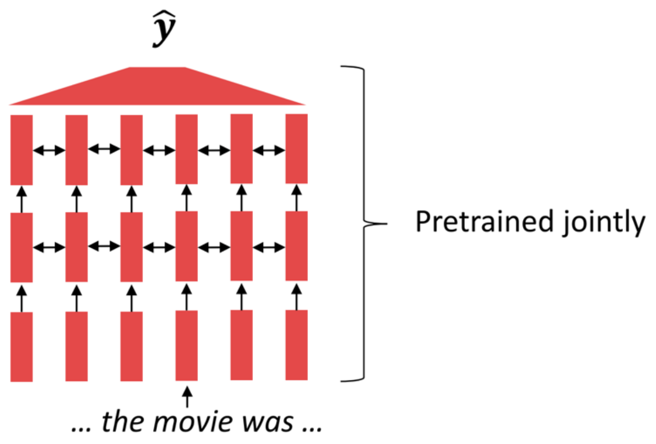
- Train a general-purpose model on large data
- Goal: Be useful for many downstream tasks
  - Transfer learning: Transfer the knowledge from one task to a different task
- Computer vision: Pretraining on ImageNet
- NLP
  - ELMo (2018) trained on One Billion Word dataset
  - BERT (2019) trained on BooksCorpus + English Wikipedia (3.3B words)

What to pretrain?

- word2vec: Pretrain word embeddings
- Now: Pretrain entire neural network



vs.



Pretrain-then-finetune Paradigm

Two stages:

1. Pretrain the model to learn general language understanding
2. Finetune it on a specific task to learn task-specific features

Pretrain-then-finetune Paradigm

Two stages:

1. Pretrain the model to learn general language understanding
2. Finetune it on a specific task to learn task-specific features

This happens 1x

This happens 1x per task

Pretrain-then-finetune Paradigm

Two stages:

1. Pretrain the model to learn general language understanding
2. Finetune it on a specific task to learn task-specific features

This happens 1x

This happens 1x per task

Expensive

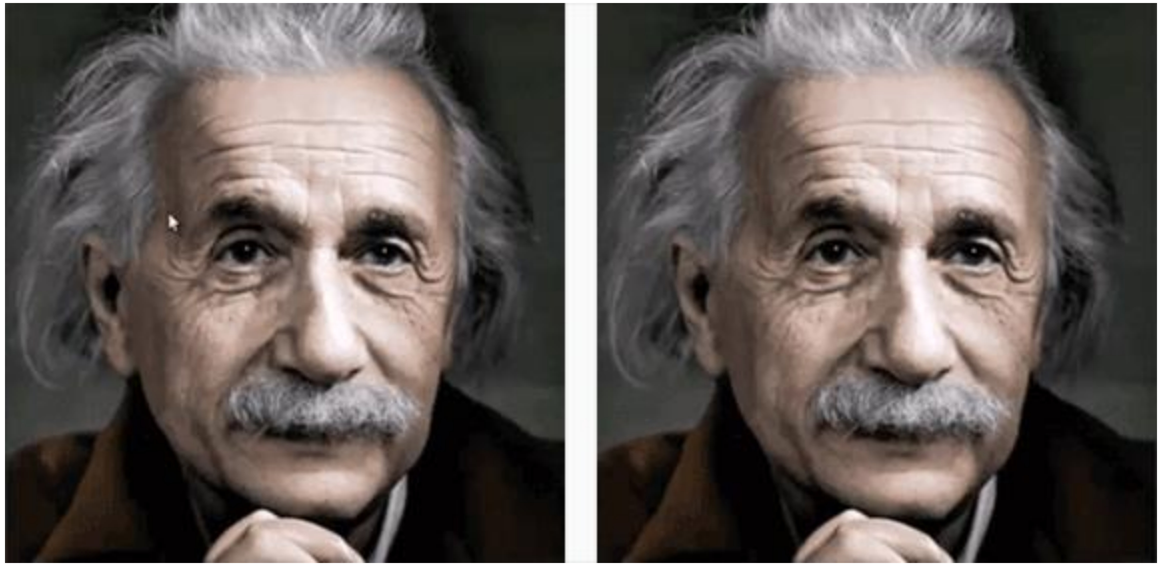
Cheap

Self-supervised vs. Unsupervised

- Discussion about terminology
- Supervised: Data contains both inputs X and targets/labels Y
  - Examples: Image classification, machine translation
- Unsupervised: Unlabeled inputs X
  - Examples: Clustering, dimensionality reduction (e.g. PCA)
- Self-supervised: Create your own labels from unlabeled data
  - Examples: Inpainting, reconstruction with autoencoders, language modeling

Self-supervision

- Create your own labels
- Inpainting: Masking



Self-supervision

- Language modeling: Predict the next word

  I went to the store to buy a _____

- Masked language modeling: Predict the masked word

  I [MASK] to the store to buy a banana.

  → In contrast to (causal) language modeling: Can use the right context (= following words) as well to predict the masked word

BERT

BERT

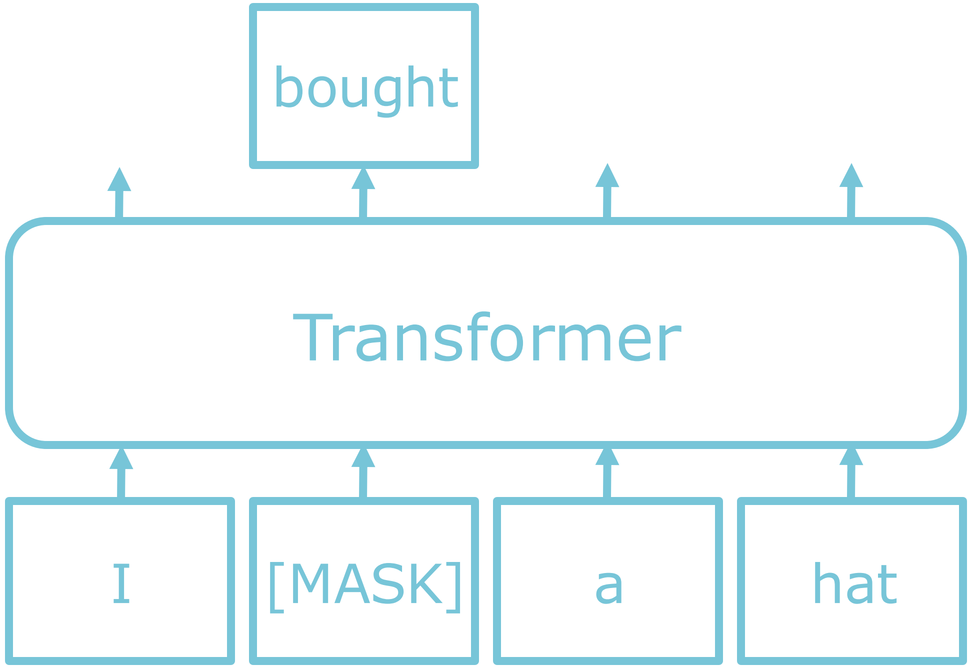
- Devlin et al., 2019
- BERT = Bidirectional Encoder Representations from Transformers
- Transformer encoder only

BERT Objective 1: Masked Language Modeling (MLM)

- Mask a % of input tokens and ask the model to predict them

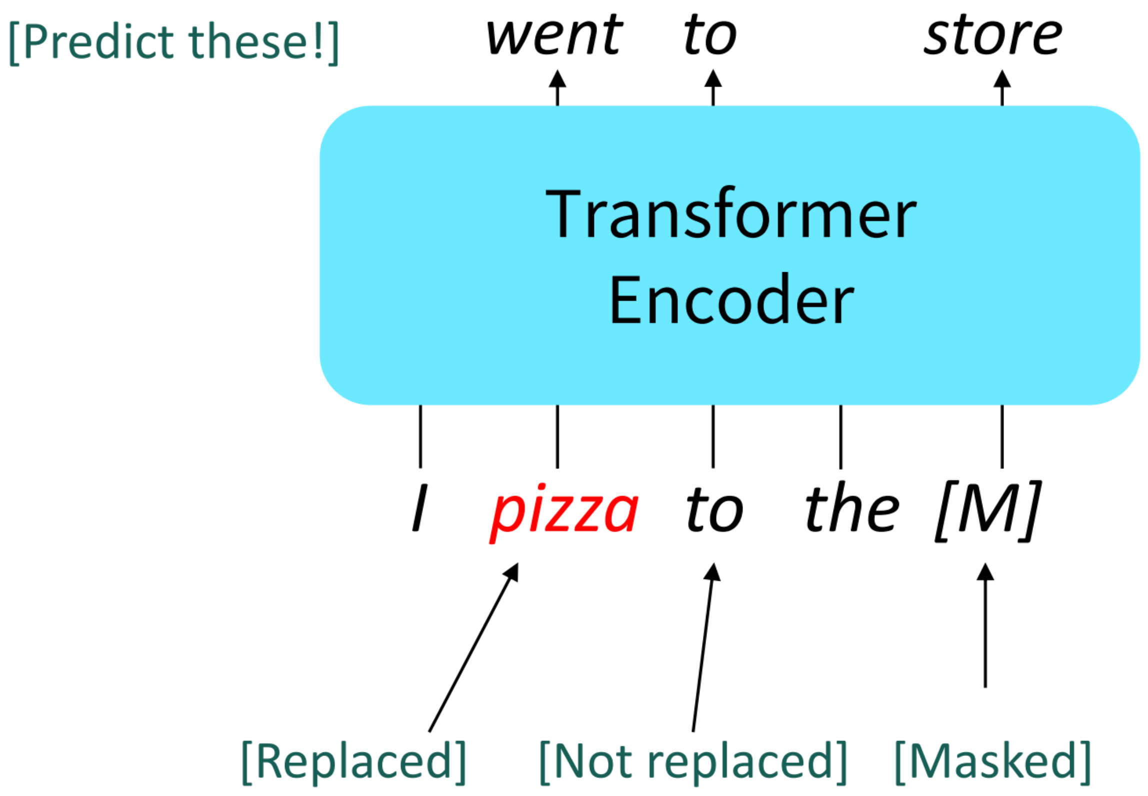
How do we predict masked tokens in the Transformer?

- Input: masked sequence
- Output: hidden state for each input token
- Predict the word from the output hidden state at the position of [MASK]
  - Compute a loss on this token
  - Ignore the outputs at other positions



BERT MLM Objective

- Predict 15% of tokens (randomly selected for each example)
- Of those 15% …
  - 80% are replaced with [MASK]
  - 10% are replaced with a random token
  - 10% are unchanged (but we still compute a loss here)



BERT Objective 2: Next Sentence Prediction (NSP)

- Pick two sentences from the data
- Model has to predict whether s_2 follows s_1 in the text
- Format:
  [CLS] s_1 [SEP] s_2 [SEP]

Special tokens

- Use the output hidden state of [CLS] to predict if sentences are consecutive

HSLU

BERT Embeddings

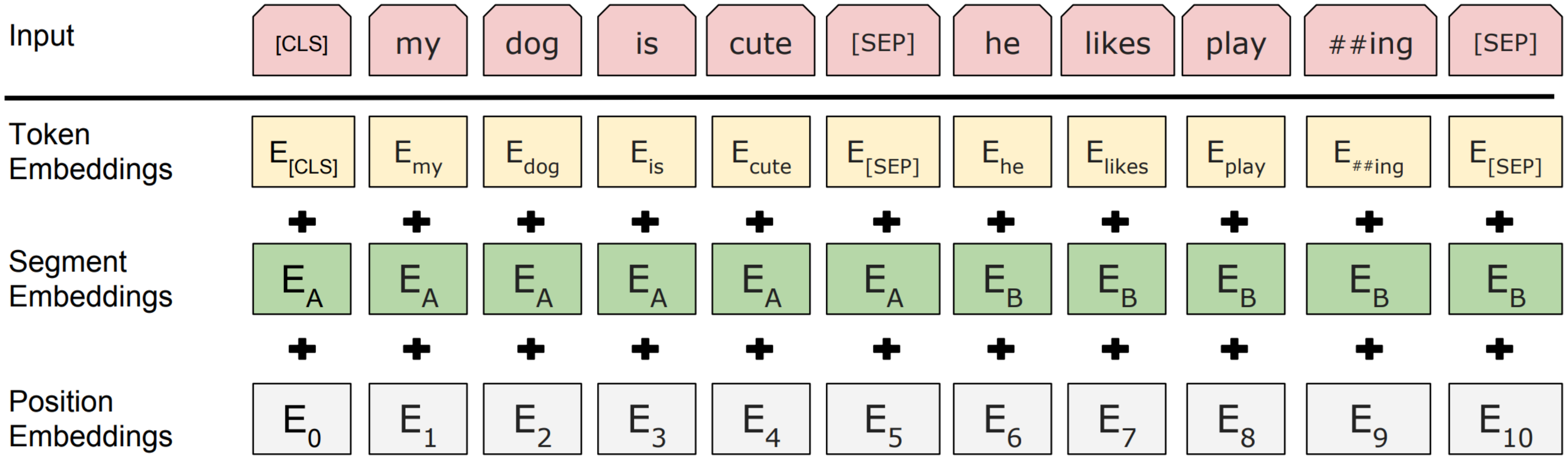


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Segment embeddings are called “token type embeddings” in Hugging Face

BERT NSP Objective

- Not clear if the next sentence prediction objective is really necessary
- Follow-up study (RoBERTa, Liu et al., 2019) found that removing it improved results on language understanding tasks
  - The paper doesn't say anything about the segment embeddings, so they are probably removed
  - In general, adding additional information through embeddings is a promising method, for example in TableFormer

BERT Training

- Pretraining: 64 TPUs for 4 days
- Finetuning: Typically a few hours on a single GPU

- Two model sizes:
  - BERT-base: 12 layers, 768 hidden_dim, 12 attention heads, 110M parameters
  - BERT-large: 24 layers, 1024 hidden_dim, 16 attention heads, 340M parameters

BERT Configuration

Look at it yourself:

```
>>> from transformers import BertConfig
>>> config = BertConfig.from_pretrained('bert-base-uncased’)
>>> print(config)
BertConfig {
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.23.1",
  "use_cache": true,
  "vocab_size": 30522
}
```

BERT Finetuning

- Need to train task-specific model parameters
  - E.g. classifier
  - … whose weights are randomly initialized
- Should we freeze BERT’s weights?
  - Better to update the entire network
  - Use a smaller learning rate for pretrained parameters than for randomly initialized parameters

Transformer Learning Rate

- With all Transformers: Use learning rate warmup
  - Increase learning rate (linearly is fine) from lr_start to lr_max
  - Decrease learning rate from lr_max to lr_end
  - For example: [torch.optim.OneCycleLR]
- Typical values from my own experience:
  - lr_max: Tune this with a hyperparameter search
    - BERT paper uses 5e-5, 3e-5, 2e-5
  - lr_start = lr_max/100
  - lr_end = lr_start/100
  - warmup_steps = total_steps/10

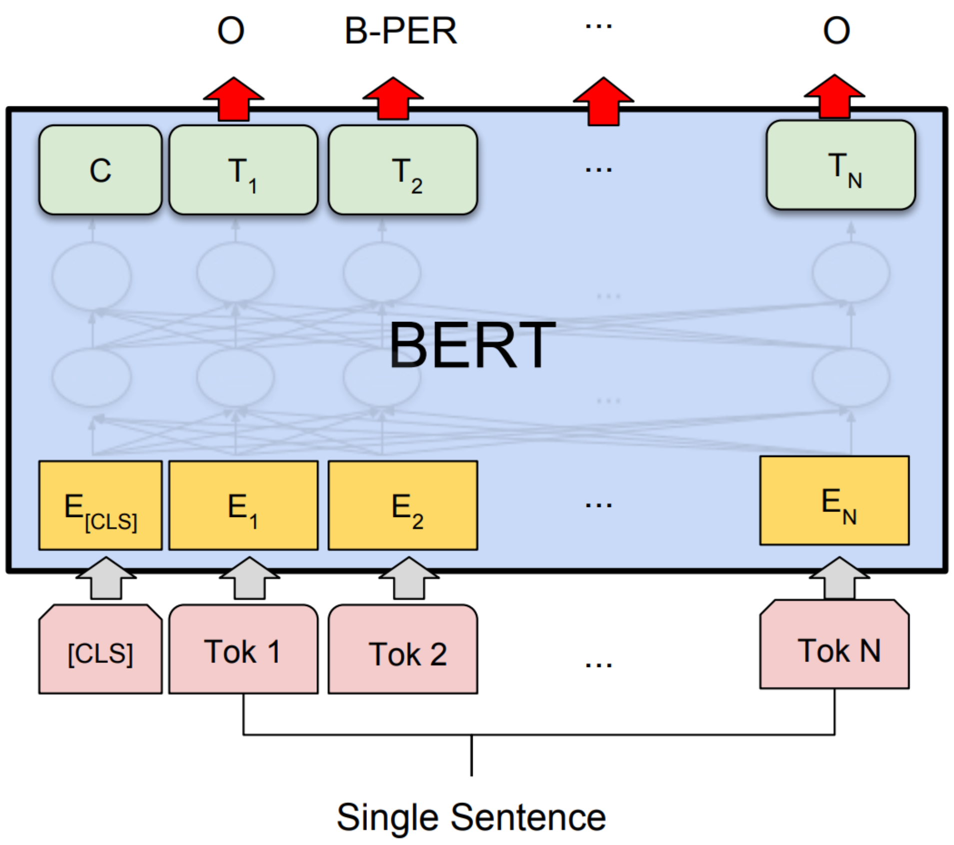
How do we use BERT for task X?

- Bring the input into a format that BERT can understand (= has seen in a similar form during pretraining)
- Run input through BERT → One output per input token
- Train a classifier to predict a label from each/specific output positions
  - Depends on the task

Single Sentence Tagging

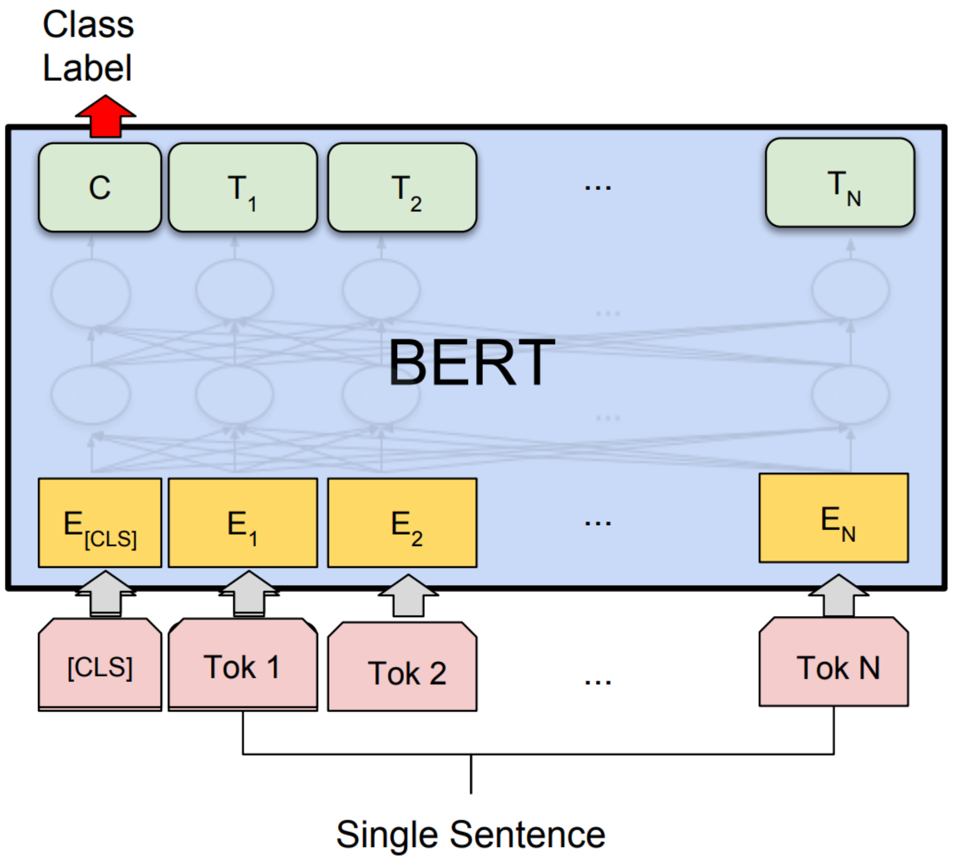
(= Token classification)

- Named Entity Recognition



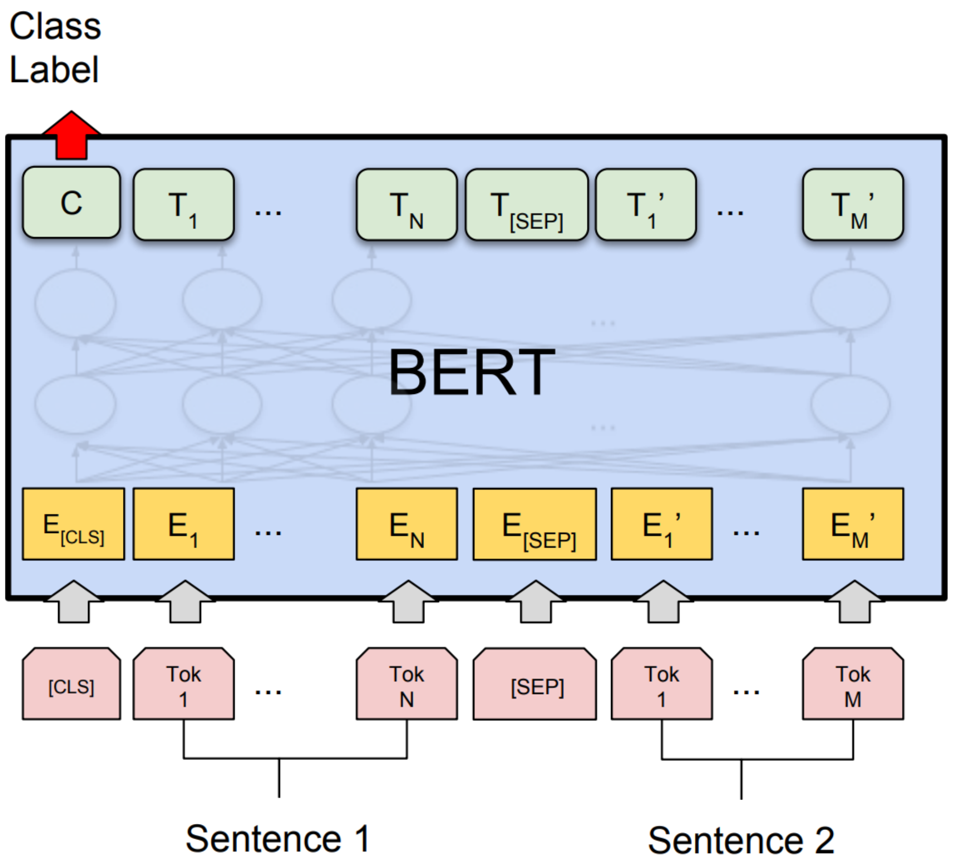
Single Sentence Classification

- Sentiment classification
- Linguistic acceptability judgment



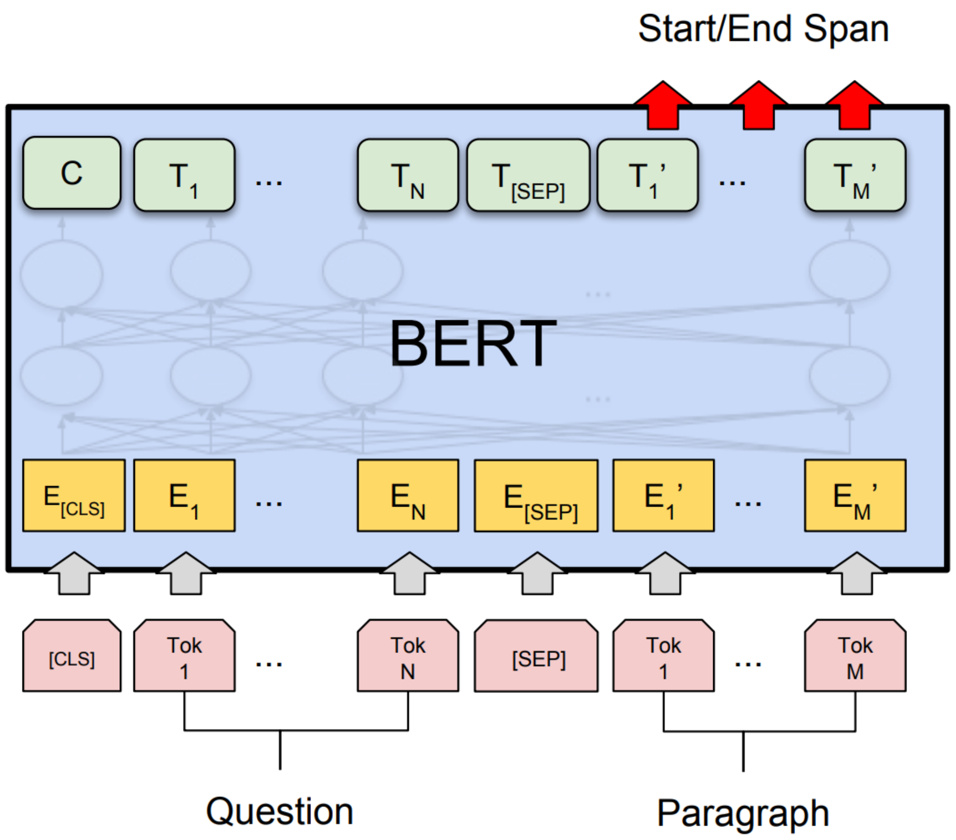
Sentence Pair Classification

- Natural language inference
- Sentence similarity
- Paraphrase detection
- Selecting the most probable continuation sentence



Question Answering

- SQuAD v1.1 (Stanford Question Answering Dataset)



BERT Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Why was BERT such a big thing?

- One model beat many state-of-the-art models specialized for their task
  - Translation
  - Question answering
  - Text classification
  - Summarization
  - Parsing
  - Natural Language Inference

→ Step towards holy grail of AI: artificial general intelligence (AGI)

In-class exercise: QA with BERT