Pretraining and Finetuning

He He

NEW YORK UNIVERSITY

February 28, 2023

Logistics

- Section will be in-person, starting at 4:55pm.
  - Review and Q&A about the lecture recording.
  - Lab material.
- Online midterm next week
- Spring break no lecture
- Project: start early! Proposal due after spring break

Table of Contents

Representation learning

What are good representations?

- Enable a notion of distance over text (word embeddings)
- Contains good features for downstream tasks

Representation learning

What are good representations?

- Enable a notion of distance over text (word embeddings)
- Contains good features for downstream tasks

negative the food is good but doesn't worth an hour wait

Simple features (e.g. BoW) require complex models.
Good features only need simple models (e.g. linear classifier) .

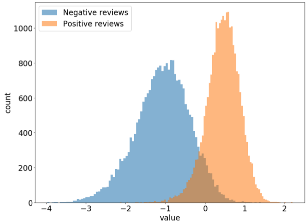


Figure: Sentiment neuron [Radford et al., 2017]

Representation learning

What can we do with good representations:

- Learning with small data: fine-tuning on learned representations
- Transfer learning: one representation for many tasks
- Metric learning: get a similarity metric for free

Representation learning

What can we do with good representations:

- Learning with small data: fine-tuning on learned representations
- Transfer learning: one representation for many tasks
- Metric learning: get a similarity metric for free

Training a neural network on any task gives us a representation good for *that task*.

But on which task can we learn good *general* representations?

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping.

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party.

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party. *coreference*
- ________ is the capital of Tanzania.

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party. *coreference*
- ________ is the capital of Tanzania. *knowledge*
- The boy is ________ because he lost his keys.

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party. *coreference*
- ________ is the capital of Tanzania. *knowledge*
- The boy is ________ because he lost his keys. *commonsense*
- John took 100 bucks to Vegas. He won 50 and then lost 100. Now he only has ________ to go home.

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party. *coreference*
- ________ is the capital of Tanzania. *knowledge*
- The boy is ________ because he lost his keys. *commonsense*
- John took 100 bucks to Vegas. He won 50 and then lost 100. Now he only has ________ to go home. *numerical reasoning*

What can we learn from word guessing?

- The cats that are raised by my sister ________ sleeping. *syntax*
- Jane is happy that John invited ________ friends to his birthday party. *coreference*
- ________ is the capital of Tanzania. *knowledge*
- The boy is ________ because he lost his keys. *commonsense*
- John took 100 bucks to Vegas. He won 50 and then lost 100. Now he only has ________ to go home. *numerical reasoning*

Word guessing entails lots of tasks related to language understanding!

Self-supervised learning

Key idea: predict parts of the input from the rest

- No supervision is needed—both input and output are from the raw data.
- Easy to scale—only need unlabeled data.
- Learned representation is general—related to many tasks.

Self-supervised learning

Key idea: predict parts of the input from the rest

- No supervision is needed—both input and output are from the raw data.
- Easy to scale—only need unlabeled data.
- Learned representation is general—related to many tasks.

Approach:

- **Pretrain**: train a model using self-supervised learning objectives on large data.
- **Finetune**: update part or all of the parameters of the pretrained model (which provides an initialization) on supervise data of a task.

A bit of history

- Pretrain an RNN model on unlabeled data and finetune on supervised tasks [Dai et al., 2015] [ULMFiT; Howard et al., 2018]
    - Promising results on a small scale

A bit of history

- Pretrain an RNN model on unlabeled data and finetune on supervised tasks [Dai et al., 2015] [ULMFiT; Howard et al., 2018]
  - Promising results on a small scale
- ELMo: replace static word embedding by contextual word embeddings from pretrained bi-LSTM [Peters et al., 2018]
  - First impactful result in NLP

A bit of history

- Pretrain an RNN model on unlabeled data and finetune on supervised tasks [Dai et al., 2015] [ULMFiT; Howard et al., 2018]
  - Promising results on a small scale
- ELMo: replace static word embedding by contextual word embeddings from pretrained bi-LSTM [Peters et al., 2018]
  - First impactful result in NLP
- Pretrain a Transformer model and finetune on supervised tasks
  - GPT [Radford et al., 2018], BERT [Devlin et al., 2018]

A bit of history

- Pretrain an RNN model on unlabeled data and finetune on supervised tasks [Dai et al., 2015] [ULMFiT; Howard et al., 2018]
  - Promising results on a small scale
- ELMo: replace static word embedding by contextual word embeddings from pretrained bi-LSTM [Peters et al., 2018]
  - First impactful result in NLP
- Pretrain a Transformer model and finetune on supervised tasks
  - GPT [Radford et al., 2018], BERT [Devlin et al., 2018]
- Scale the pretrained model to larger sizes
  - GPT-2 (1.5B), T5 (11B), GPT-3 (175B), PaLM (540B)
  - We will talk about 100B+ models in the third module

Types of pretrained models

- **Encoder models**, e.g., BERT
  - Encode text into vector representations that can be used for downstream classification tasks

Types of pretrained models

- **Encoder models**, e.g., BERT
  - Encode text into vector representations that can be used for downstream classification tasks
- **Encoder-decoder models**, e.g., T5
  - Encode input text into vector representations and generate text conditioned on the input

Types of pretrained models

- **Encoder models**, e.g., BERT
  - Encode text into vector representations that can be used for downstream classification tasks
- **Encoder-decoder models**, e.g., T5
  - Encode input text into vector representations and generate text conditioned on the input
- **Decoder models**, e.g., GPT-2 (later)
  - Read in text (prefix) and continue to generate text

Types of pretrained models

- **Encoder models**, e.g., BERT
  - Encode text into vector representations that can be used for downstream classification tasks
- **Encoder-decoder models**, e.g., T5
  - Encode input text into vector representations and generate text conditioned on the input
- **Decoder models**, e.g., GPT-2 (later)
  - Read in text (prefix) and continue to generate text

Types of pretrained models

- **Encoder models**, e.g., BERT
  - Encode text into vector representations that can be used for downstream classification tasks
- **Encoder-decoder models**, e.g., T5
  - Encode input text into vector representations and generate text conditioned on the input
- **Decoder models**, e.g., GPT-2 (later)
  - Read in text (prefix) and continue to generate text

All models are transformer based.

Encoder models

An encoder takes a sequence of tokens and output their contextualized representations:

$$h_1, \ldots, h_n = \mathrm{Encoder}(x_1, \ldots, x_n)$$

We can then use $h_1, \ldots, h_n$ for other tasks.

Encoder models

An encoder takes a sequence of tokens and output their contextualized representations:

$$h_1, \dots, h_n = \mathrm{Encoder}(x_1, \dots, x_n)$$

We can then use $h_1, \dots, h_n$ for other tasks.

How do we train $\mathrm{Encoder}$?

- Use any supervised task: $y = f(h_1, \dots, h_n)$
- Use self-supervised learning: predict a word from its neighbors

Masked language modeling

Learning objective:

$$\max \sum_{x \in \mathcal{D}, i \sim p_{\text{mask}}} \log p(x_i \mid x_{-i}; \theta)$$

- x_{-i}: noisy version fo x where x_i is corrupted
- p_{mask}: mask generator

BERT: objective

- **Masked language modeling**:
  - Randomly sample 15% tokens as prediction targets
  - Replace the target tokens in the input by either `[MASK]` (10%) or a random token (10%), or leave it unchanged
    cats are cute $\rightarrow$ cats `[MASK]`/is/are cute
  - Later work has shown that just use `[MASK]` is sufficient

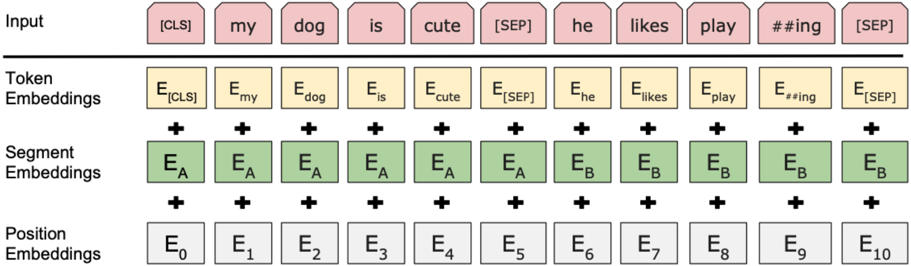
BERT: objective

- **Masked language modeling**:
  - Randomly sample 15% tokens as prediction targets
  - Replace the target tokens in the input by either `[MASK]` (10%) or a random token (10%), or leave it unchanged
    cats are cute $\rightarrow$ cats `[MASK]`/is/are cute
  - Later work has shown that just use `[MASK]` is sufficient
- **Next sentence prediction**: predict whether a pair of sentences are consecutive

$$\max \sum_{x \sim \mathcal{D}, x_n \sim p_{\text{next}}} \log p(y \mid x, x_n; \theta)$$

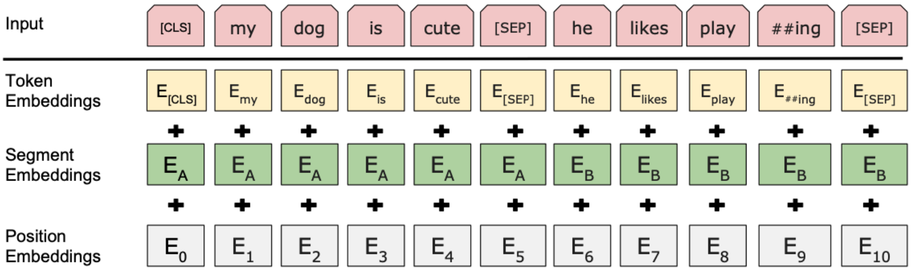
  - x_n: either the sentence following x or a randomly sampled sentence
  - y: binary label of whether x_n follows x
  - Later work has shown that this objective is not necessary

BERT: architecture



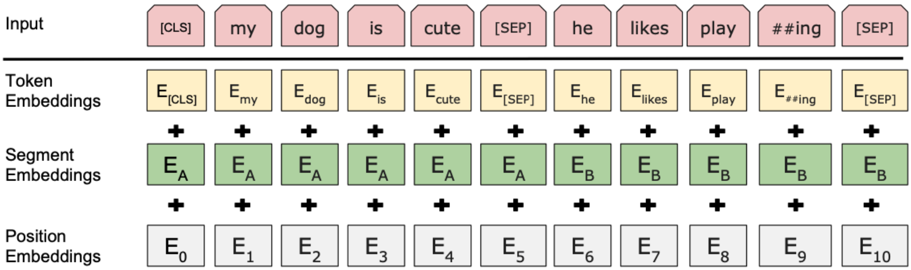
- Subword unit: wordpiece (basically byte pair encoding)

BERT: architecture



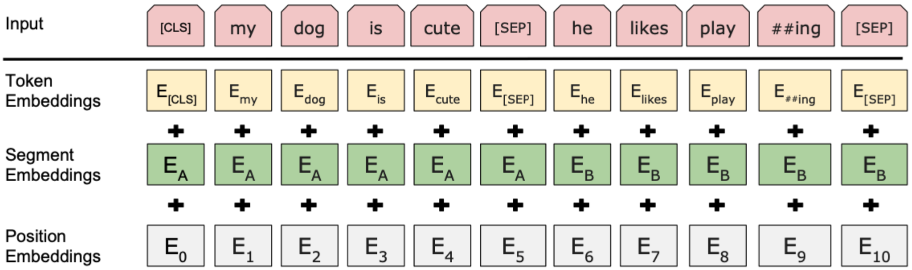
- Subword unit: wordpiece (basically byte pair encoding)
- `[CLS]`: first token of all sequences; used for next sentence prediction

BERT: architecture



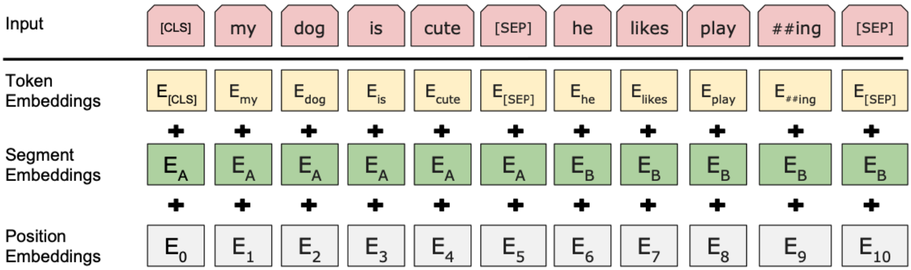
- Subword unit: wordpiece (basically byte pair encoding)
- `[CLS]`: first token of all sequences; used for next sentence prediction
- Distinguish two sentences in a pair: `[SEP]` and segment embedding

BERT: architecture



- Subword unit: wordpiece (basically byte pair encoding)
- `[CLS]`: first token of all sequences; used for next sentence prediction
- Distinguish two sentences in a pair: `[SEP]` and segment embedding
- Learned position embedding

BERT: architecture

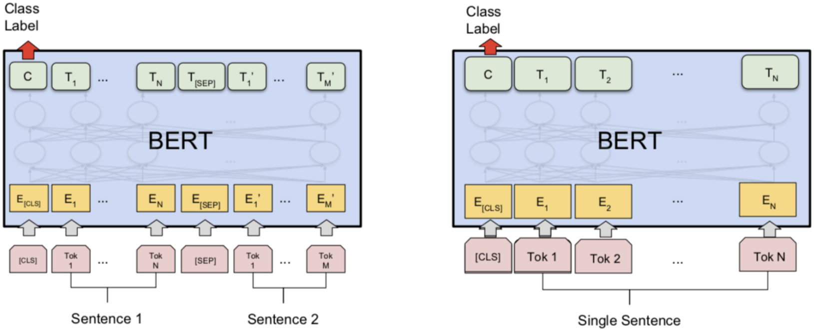


- Subword unit: wordpiece (basically byte pair encoding)
- `[CLS]`: first token of all sequences; used for next sentence prediction
- Distinguish two sentences in a pair: `[SEP]` and segment embedding
- Learned position embedding
- 12 (base; 110M params) or 24 (large; 340M params) layer Transformer

Finetuning BERT

Classification tasks: Add a linear layer (randomly initialized) on top of the `[CLS]` embedding

$$p(y \mid x) = \mathrm{softmax}(W h_{\texttt{[CLS]}})$$



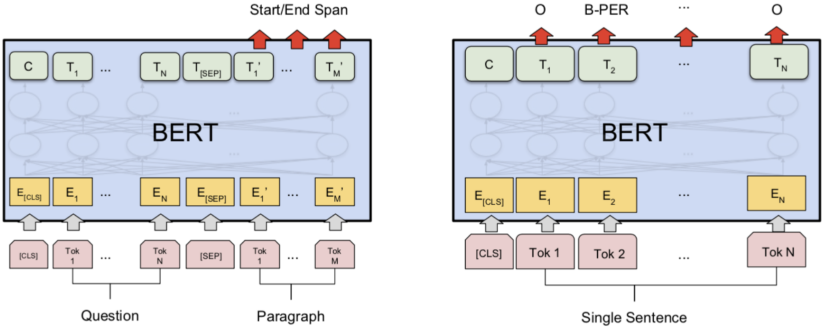
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

Finetuning BERT

Sequence labeling tasks: Add linear layers (randomly initialized) on top of every token

$$p(y_i \mid x) = \mathrm{softmax}(Wh_i)$$



(c) Question Answering Tasks:
SQuAD v1.1

(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Finetuning BERT

- Finetune all parameters (both the newly added layer and the pretrained weights)
- Use a small learning rate (e.g., 1e-5)
- Train for a small number of epochs (e.g, 3 epochs)
- Led to SOTA results on many NLU tasks
- Not straightforward to use on text generation tasks

Encoder-decoder models

An encoder-decoder model encodes input text to a sequence of contextualized representations, and decodes a sequence of tokens autoregressively.

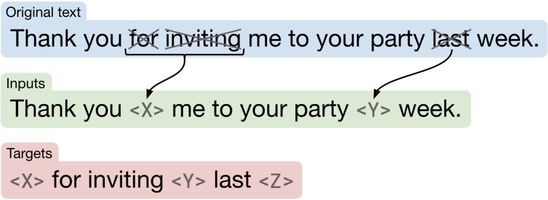
$$
\begin{aligned}
h_1, \ldots, h_n &= \mathrm{Encoder}(x_1, \ldots, x_n) \\
s_1, \ldots, s_m &= \mathrm{Decoder}(y_0, \ldots, y_{m-1}, h_1, \ldots, h_n) \\
p(y_i \mid x, y_{<i}) &= \mathrm{softmax}(W s_i)
\end{aligned}
$$

How do we train the encoder-decoder?

- Use any supervised task, e.g., machine translation
- Use self-supervised learning: predict text spans from their neighbors

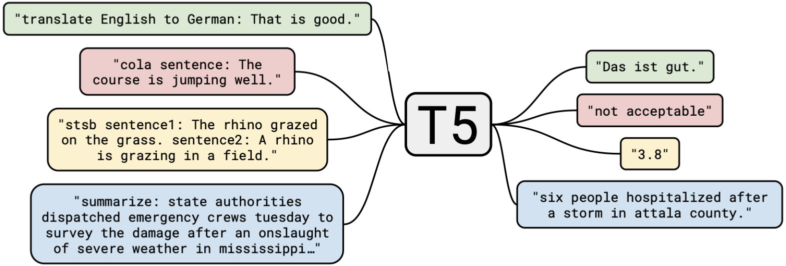
Masked language modeling using an encoder-decoder

Input: text with corrupted spans
Output: recovered spans



T5

- First train on unlabele data by **masked language modeling**
  - Predict corrupted spans as a sequence
- Then continue training by **supervised multitask learning**
  - Formulate tasks as text-to-text format
  - Use a prefix to denote the task
  - Mixing examples from different datasets when constructing batches



- Jointly training with the two objectives works slightly worse

Finetuning T5

- Formulate the task in text-to-text format
- Fine-tune all parameters (similar to BERT fine-tuning)
- Advantages over encoder models: unified modeling of many different tasks

Efficient pretraining

An obvious downside of pretrained models is that they are quite expensive to train!

How can we make them more efficient?

Efficient pretraining

An obvious downside of pretrained models is that they are quite expensive to train!

How can we make them more efficient?

Idea 1: reducing the number of parameters smartly

Example: ALBERT (a lite BERT) [Lan et al., 2020]

- **Parameter sharing**:
  - Share feedforward network weights across layers
  - Share self-attention weights across layers
  - ALBERT: share all params across layers

Efficient pretraining

Idea 2: design harder learning objectives

ALBERT: Inter-sentence coherence loss

- Motivation: the next sentence prediction task is too easy
- Design hard negative examples
- Input: take two consecutive sentences, swap their order randomly
- Output: predict if they are in natural order

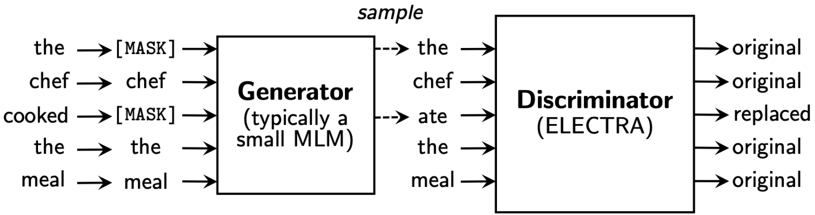
  I went home. SEP *I slept.* +1

  I slept. SEP *I went home.* -1
- What is needed to perform this task well?

Efficient pretraining

Idea 2: design harder learning objectives

ELECTRA [Clark et al., 2020]: discriminate from true vs guessed tokens



- First train the generator for n steps using the MLM objective.
- Freeze generator weights. Then train the discriminator using the sequence classification objective.
- The discriminator and generator share weights except for the input token embeddings.

Efficient pretraining

ELECTRA result:

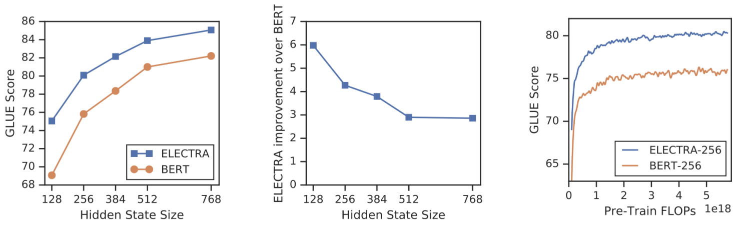


Figure: Finetuning result on the GLUE benchmark

- Larger improvement at smaller model sizes
- Faster training
- An effective approach if you don't have large compute for pretraining

What are these models trained on?

Both quantity and quality are important

- Wikipedia: encyclopedia articles (clean, single domain)
- Toronto Books Corpus: e-books (diverse domain)
- WebText (40GB): content submitted to Reddit with a vote ≥ 3 (diverse, bias)
- CommonCrawl (20TB): scraped HTML with markers removed (diverse, large, noisy, bias)
  - A cleaned version: C4 (750GB)

Summary

Lots of learning happens from just observing the world (data).

- Self-supervised learning: benefits from large data and compute
  - Basic: predict parts from other parts based on the structure of data (works beyond text)
  - Advanced: design hard negatives to improve efficiency
- Finetuning: adapt pretrained models to downstream tasks on a small amount of labeled data