Contextual word representations

Christopher Potts

Stanford Linguistics

CS224u: Natural language understanding

Guiding ideas

Static vector representations of words

1. Feature-based (sparse): Classical lexical representations
2. Count-based methods (sparse): PMI, TF-IDF, etc.
3. Classical dimensionality reduction (dense): PCA, SVD, LDA, etc.
4. Learned dimensionality reduction (dense): autoencoders, word2vec, GloVe, etc.

Hands-on review:
`https://web.stanford.edu/class/cs224u/background.html`

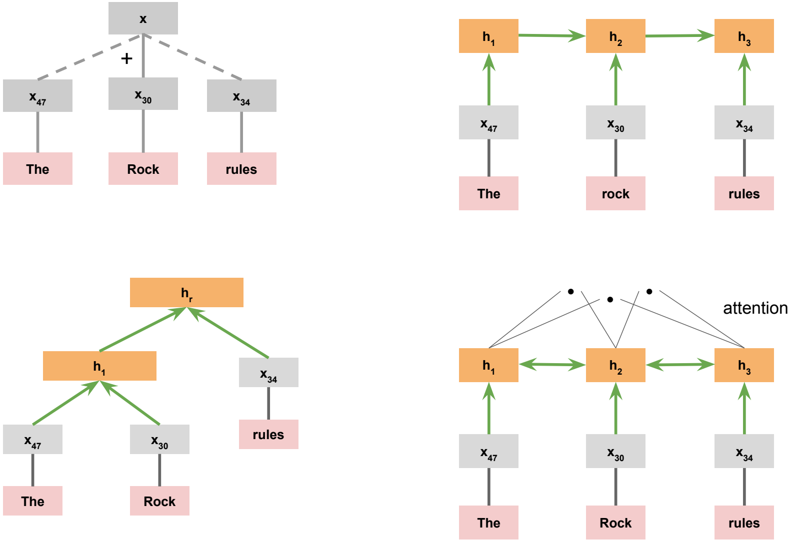
Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burglar broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.
2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit
3. a. A crane caught a fish.
   b. A crane picked up the steel beam.
   c. I saw a crane.
4. a. Are there typos? I didn't see any.
   b. Are there bookstores downtown? I didn't see any.

A brief history of contextual representation

1. November 2015: Dai and Le (2015) showed the value of LM-style pretraining for downstream tasks.
2. August 2017: McCann et al. (2017) (CoVe) pretrained bi-LSTMs for machine translation and showed that this was a useful start-state for downstream tasks.
3. February 2018: Peters et al. (2018) (ELMo) first showed how very large-scale pretraining of bidirectional LSTMs can lead to rich multipurpose representations.
4. June 2018: Radford et al. (2018) introduced GPT.
5. October 2018: Devlin et al. (2019) introduced BERT.

Model structure and linguistic structure



Attention

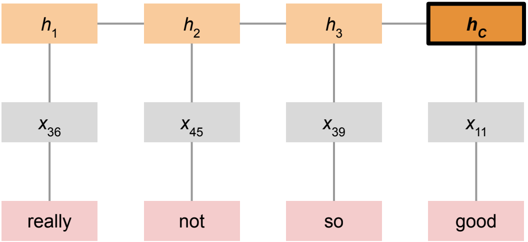
classifier $y = \textbf{softmax}(\tilde{h}W + b)$

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$

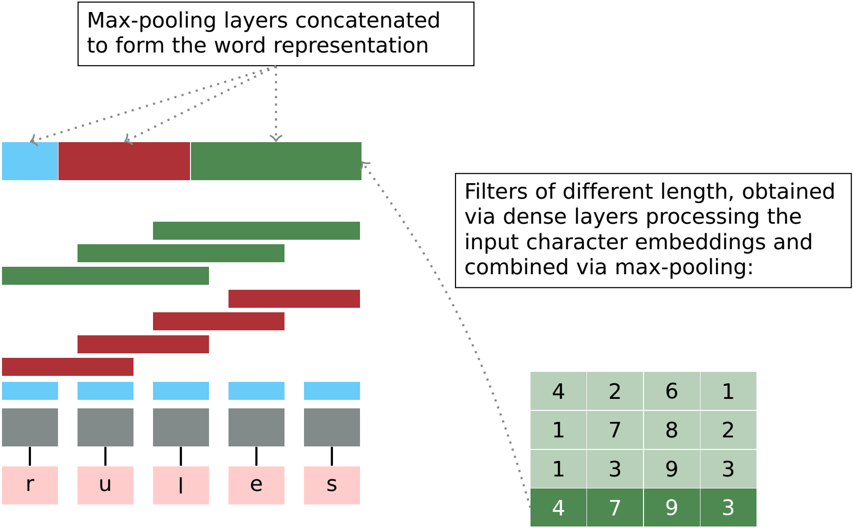
context $\kappa = \textbf{mean}\left([\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3]\right)$

attention weights $\alpha = \textbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Subword modeling in ELMo

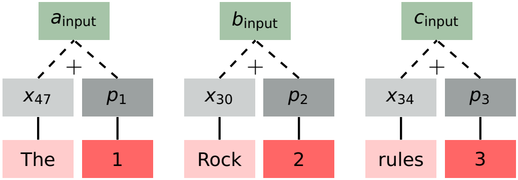


Guiding idea: Word pieces

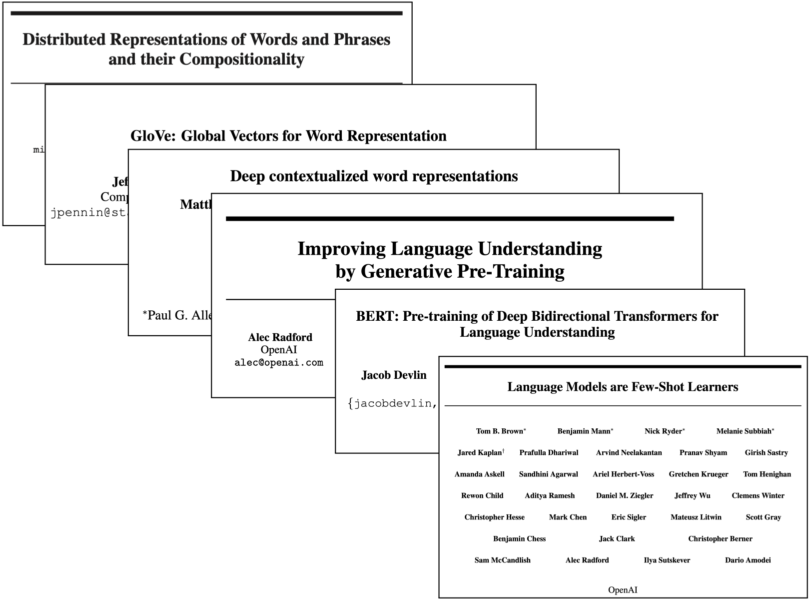
```
[1]: from transformers import BertTokenizer

[2]: tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

[3]: tokenizer.tokenize("This isn't too surprising.")

[3]: ['This', 'isn', "'", 't', 'too', 'surprising', '.']

[4]: tokenizer.tokenize("Encode me!")

[4]: ['En', '##code', 'me', '!']

[5]: tokenizer.tokenize("Snuffleupagus?")

[5]: ['S', '##nu', '##ffle', '##up', '##agu', '##s', '?']

[6]: tokenizer.vocab_size

[6]: 28996
```

Sennrich et al. 2016,
https://github.com/google/sentencepiece

Guiding idea: Positional encoding



Guiding idea: Massive scale pretraining



Distributed Representations of Words and Phrases and their Compositionality

mi

GloVe: Global Vectors for Word Representation

Jef
Comp
jpennin@st

Deep contextualized word representations

Matt

*Paul G. Alle

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin

{jacobdevlin,

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Guiding idea: Fine-tuning

1. 2016–2018: Static word representation as RNN embeddings
2. 2018–:

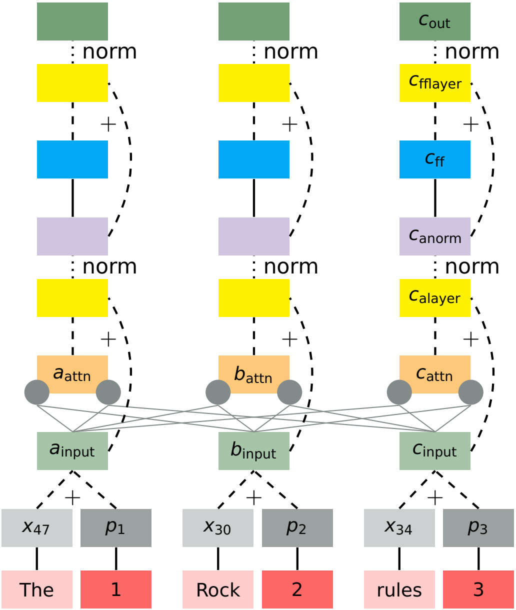
[31]:
```
class HfBertClassifierModel(nn.Module):
    def __init__(self, n_classes, weights_name='bert-base-cased'):
        super().__init__()
        self.n_classes = n_classes
        self.weights_name = weights_name
        self.bert = BertModel.from_pretrained(self.weights_name)
        self.bert.train()
        self.hidden_dim = self.bert.embeddings.word_embeddings.embedding_dim
        # The only new parameters -- the classifier:
        self.classifier_layer = nn.Linear(
            self.hidden_dim, self.n_classes)

    def forward(self, indices, mask):
        reps = self.bert(
            indices, attention_mask=mask)
        return self.classifier_layer(reps.pooler_output)
```

3. 2021–:

```
openai api fine_tunes.create -t <TRAIN_FILE_ID_OR_PATH> -m <BASE_MODEL>
```

The Transformer

Core model structure



$$c_{\text{out}} = \frac{c_{\text{fflayer}} - \textbf{mean}(c_{\text{fflayer}})}{\textbf{std}(c_{\text{fflayer}}) + \epsilon}$$

$$c_{\text{fflayer}} = c_{\text{anorm}} + \textbf{Dropout}(c_{\text{ff}})$$

$$c_{\text{ff}} = \textbf{ReLU}(c_{\text{anorm}} W_1 + b_1) W_2 + b_2$$

$$c_{\text{anorm}} = \frac{c_{\text{alayer}} - \textbf{mean}(c_{\text{alayer}})}{\textbf{std}(c_{\text{alayer}}) + \epsilon}$$

$$c_{\text{alayer}} = \textbf{Dropout}\left(c_{\text{attn}} + c_{\text{input}}\right)$$

$$c_{\text{attn}} = \textbf{sum}\left([\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}]\right)$$

$$\alpha = \textbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{c_{\text{input}}{}^{\top} a_{\text{input}}}{\sqrt{d_k}}, \frac{c_{\text{input}}{}^{\top} b_{\text{input}}}{\sqrt{d_k}}\right]$$

$$c_{\text{input}} = x_{34} + p_3$$

Computing the attention representations

Calculation as previously given

$$
\begin{aligned}
c_{\text{attn}} &= \textbf{sum}\left(\left[\alpha_1 a_{\text{input}}, \alpha_2 b_{\text{input}}\right]\right) \\
\alpha &= \textbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{{c_{\text{input}}}^{\top} a_{\text{input}}}{\sqrt{d_k}}, \frac{{c_{\text{input}}}^{\top} b_{\text{input}}}{\sqrt{d_k}}\right]
\end{aligned}
$$

Matrix format

$$
\textbf{softmax}\left(\frac{c_{\text{input}} \begin{bmatrix} a_{\text{input}} \\ b_{\text{input}} \end{bmatrix}^{\top}}{\sqrt{d_k}}\right) \begin{bmatrix} a_{\text{input}} \\ b_{\text{input}} \end{bmatrix}
$$

Computing the attention representations

```
[1]: import numpy as np
```

```
[2]: seq_length = 3
     d_k = 4
```

```
[3]: inputs = np.random.uniform(size=(seq_length, d_k))
     inputs
```

```
[3]: array([[0.31436922, 0.66969307, 0.270804  , 0.72023504],
            [0.87180132, 0.27637445, 0.43091867, 0.34138704],
            [0.20292054, 0.6345131 , 0.01058343, 0.22846636]])
```

```
[4]: a_input = inputs[0]
     b_input = inputs[1]
     c_input = inputs[2]
```

Computing the attention representations

[5]:
```
def softmax(X):
    z = np.exp(X)
    return (z / z.sum(axis=0)).T
```

[6]:
```
c_alpha = softmax([
    (c_input.dot(a_input) / np.sqrt(d_k)),
    (c_input.dot(b_input) / np.sqrt(d_k))])
```

[7]:
```
c_attn = sum([c_alpha[0]*a_input, c_alpha[1]*b_input])
c_attn
```

[7]:
```
array([0.57768027, 0.48390338, 0.34643646, 0.54128076])
```

[8]:
```
ab = inputs[:-1]
```

[9]:
```
softmax(c_input.dot(ab.T) / np.sqrt(d_k)).dot(ab)
```

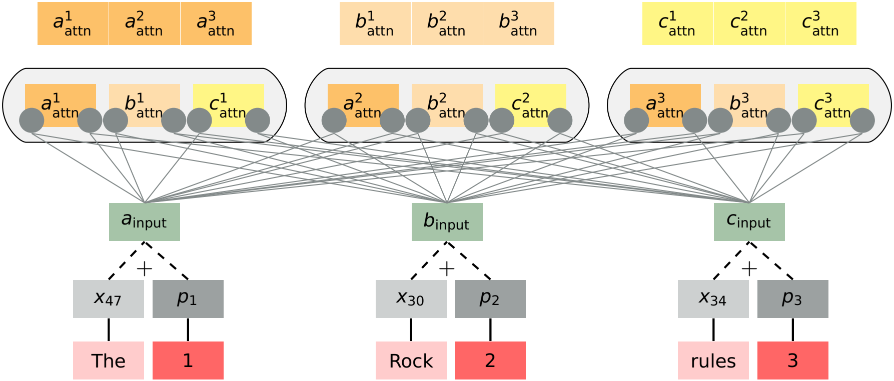
[9]:
```
array([0.57768027, 0.48390338, 0.34643646, 0.54128076])
```

[10]:
```
# If we allow every input to attend to itself:
softmax(inputs.dot(inputs.T) / np.sqrt(d_k)).dot(inputs)
```

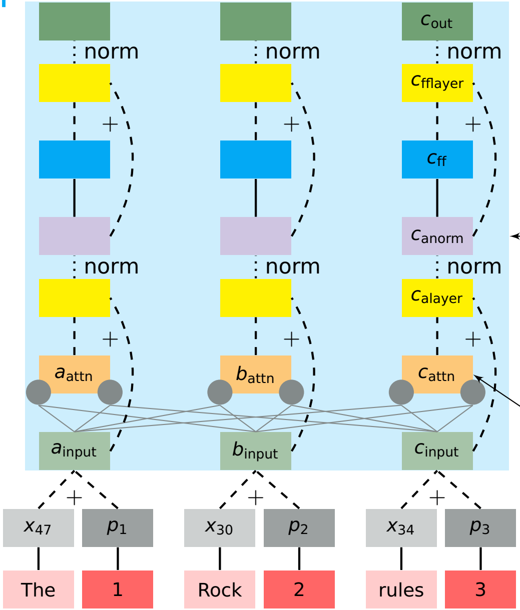
[10]:
```
array([[0.4614388 , 0.53204444, 0.2451212 , 0.45136127],
       [0.50173123, 0.50618272, 0.26184404, 0.43678288],
       [0.45493467, 0.5332328 , 0.23643403, 0.4388242 ]])
```

Multi-headed attention

$$
\begin{aligned}
c^3_{\text{attn}} &= \textbf{sum}\left(\left[\alpha_1(a_{\text{input}} W^V_3), \alpha_2(b_{\text{input}} W^V_3)\right]\right) \\
\alpha &= \textbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(c_{\text{input}} W^Q_3)^\top (a_{\text{input}} W^K_3)}{\sqrt{d_k}}, \frac{(c_{\text{input}} W^Q_3)^\top (b_{\text{input}} W^K_3)}{\sqrt{d_k}}\right]
\end{aligned}
$$



Repeated transformer blocks



Repeated N times with c_{out} serving as c_{input} at each successive layer.

Includes multi-headed attention in each block

The architecture diagram

A view from PyTorch

```python
from transformers import AutoModel
```

```python
model = AutoModel.from_pretrained('bert-base-cased')
```

```python
model
```

```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(28996, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (1): BertLayer(
```

Positional encoding

The role of positional encoding

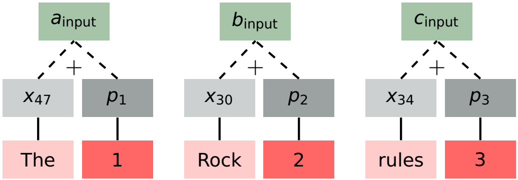
- The Transformer has a very limited capacity to keep track of word order:
  - the attention connections are not directional, and
  - there are no other interactions between the columns.
- Positional encodings ensure differences between `A B C` and `C B A`.
- Positional encodings have also been used to keep track of hierarchical notions of position like premise/hypothesis in Natural Language Inference.

Evaluating positional encoding schemes

1. Does the set of positions need to be decided ahead of time?
2. Does the scheme hinder generalization to new positions?

Models will tend to impose a max length on the sequences they can process for reasons relating to their learned weights. We will ask whether different positional encoding schemes are imposing anything about length generalization *separate from this*.

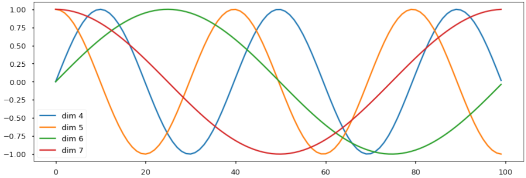
Absolute positional encoding



Limitations

1. Set of position needs to be decided ahead of time.
2. May hinder generalization to new positions, even for familiar phenomena:

Frequency-based positional encoding




```python
import numpy as np
def posenc(pos, d_model):
    div_term = np.exp(
        np.arange(0, d_model, 2) * -(np.log(10000.0) / d_model))
    rep = np.zeros(d_model)
    rep[0::2] = np.sin(pos * div_term)
    rep[1::2] = np.cos(pos * div_term)
    return rep
```

Limitations

1. ~~Set of position needs to be decided ahead of time.~~
2. May hinder generalization to new positions, even for familiar phenomena.

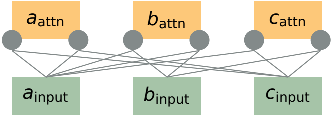
'The Annotated Transformer'

Relative positional encoding: Basics

Previously

$$\boxed{c_{\text{attn}}} = \textbf{sum}\left(\left[\alpha^c_1 a_{\text{input}}, \alpha^c_2 b_{\text{input}}\right]\right)$$

$$\alpha^c = \textbf{softmax}\left(\left[\frac{{c_{\text{input}}}^\top a_{\text{input}}}{\sqrt{d_k}}, \frac{{c_{\text{input}}}^\top b_{\text{input}}}{\sqrt{d_k}}\right]\right)$$



Relative encoding

$$\boxed{c_{\text{attn}}} = \textbf{sum}\left(\left[\alpha^c_1 a_{\text{input}} + a^V_{3,1}, \alpha^c_2 b_{\text{input}} + a^V_{3,2}\right]\right)$$

$$\alpha^c = \textbf{softmax}\left(\left[\frac{{c_{\text{input}}}^\top (a_{\text{input}} + a^K_{3,1})}{\sqrt{d_k}} \frac{{c_{\text{input}}}^\top (b_{\text{input}} + a^K_{3,2})}{\sqrt{d_k}}\right]\right)$$

Shaw et al. 2018

Relative positional encoding: Windows

$$\boxed{c_{\text{attn}}} = \textbf{sum}\left(\left[\alpha_1^c a_{\text{input}} + a_{3,1}^V, \alpha_2^c b_{\text{input}} + a_{3,2}^V\right]\right)$$

$$\alpha^c = \textbf{softmax}\left(\left[\frac{{c_{\text{input}}}^{\mathsf{T}}(a_{\text{input}} + a_{3,1}^K)}{\sqrt{d_k}} \; \frac{{c_{\text{input}}}^{\mathsf{T}}(b_{\text{input}} + a_{3,2}^K)}{\sqrt{d_k}}\right]\right)$$

With window size $d = 2$:

1	2	3	4	5	6	7
a_{input}	b_{input}	c_{input}	d_{input}	e_{input}	f_{input}	g_{input}
$a_{4,1}^K = w_{-2}^K$	$a_{4,2}^K = w_{-2}^K$	$a_{4,3}^K = w_{-1}^K$	$a_{4,4}^K = w_{0}^K$	$a_{4,5}^K = w_{1}^K$	$a_{4,6}^K = w_{2}^K$	$a_{4,7}^K = w_{2}^K$
$a_{3,1}^K = w_{-2}^K$	$a_{3,2}^K = w_{-1}^K$	$a_{3,3}^K = w_{0}^K$				

Same logic for the representations a_{ij}^V.

Shaw et al. 2018

Relative positional encoding: Full definition

With learned attention parameters:

$$\boxed{\mathrm{attn}_i} = \sum_{j=1}^{n} \alpha_{ij} \left(x_j W^V + a_{ij}^V \right)$$

$$\alpha_{ij} = \mathbf{softmax}\left(\frac{(x_i W^Q)^\top (x_j W^K + a_{ij}^K)}{\sqrt{d_k}} \right)$$

Limitations

1. ~~Set of position needs to be decided ahead of time.~~
2. ~~May hinder generalization to new positions, even for familiar phenomena.~~

The	Rock

w_0^K	w_1^K	w_{-1}^K

Shaw et al. 2018

GPT

GPT: Autoregressive loss function

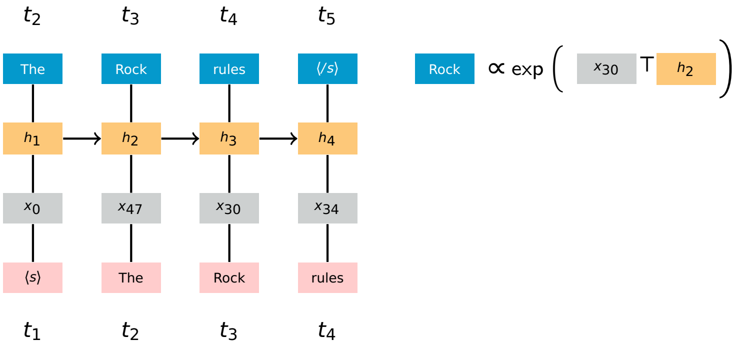
For vocabulary $\mathcal{V}$, sequence $\mathbf{x} = [x_1, \ldots, x_T]$, and word-level embedding e:

$$\max_{\theta} \sum_{t=1}^{T} \log \frac{\exp\left(e(x_t)^{\top} h_{\theta}(\mathbf{x}_{1:t-1})\right)}{\sum_{x' \in \mathcal{V}} \exp\left(e(x')^{\top} h_{\theta}(\mathbf{x}_{1:t-1})\right)}$$

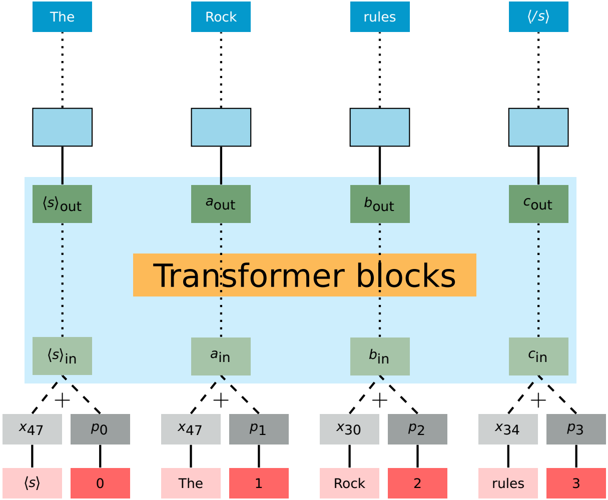
for model parameters h_{θ}.

Conditional language modeling

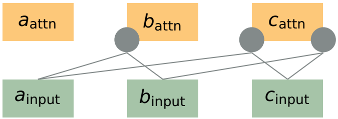
$$\mathbf{x} = [\langle s \rangle, \text{The}, \text{Rock}, \text{rules}, \langle /s \rangle]$$



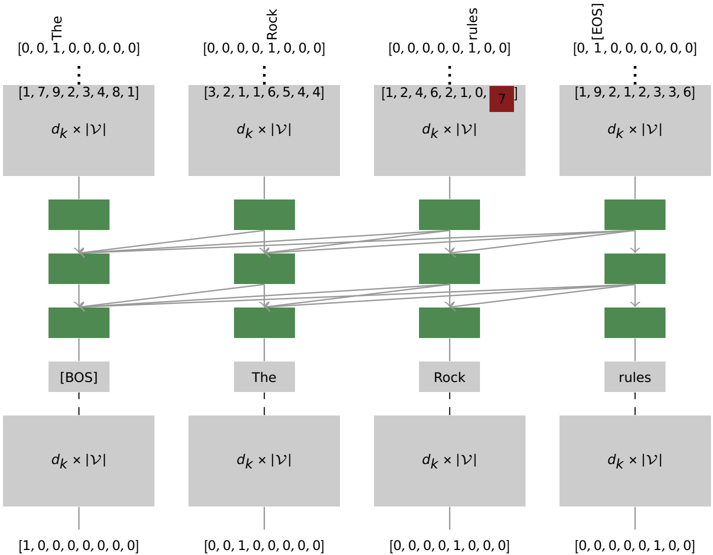
GPT


The
Rock
rules
⟨/s⟩
⟨s⟩out
aout
bout
cout
Transformer blocks
⟨s⟩in
ain
bin
cin
+
x47
p0
x47
p1
x30
p2
x34
p3
⟨s⟩
0
The
1
Rock
2
rules
3

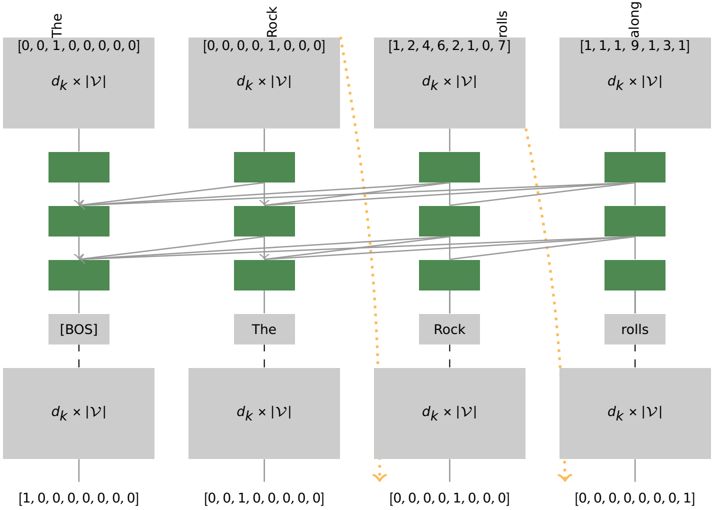
GPT: Attention masking



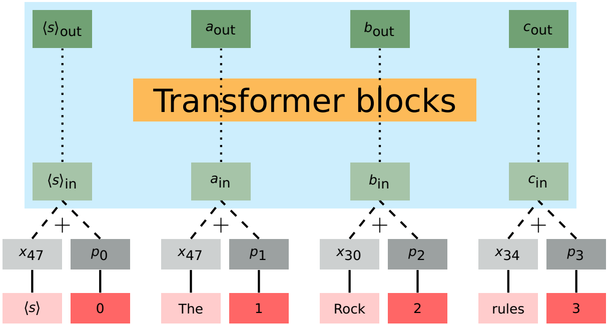
GPT: Training with teacher forcing



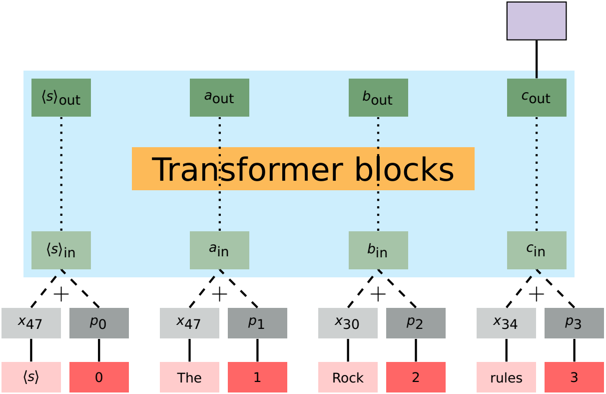
Generation


The
Rock
rolls
along
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[1, 2, 4, 6, 2, 1, 0, 7]
[1, 1, 1, 9, 1, 3, 1]
$d_k \times |\mathcal{V}|$
[BOS]
The
Rock
rolls
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]

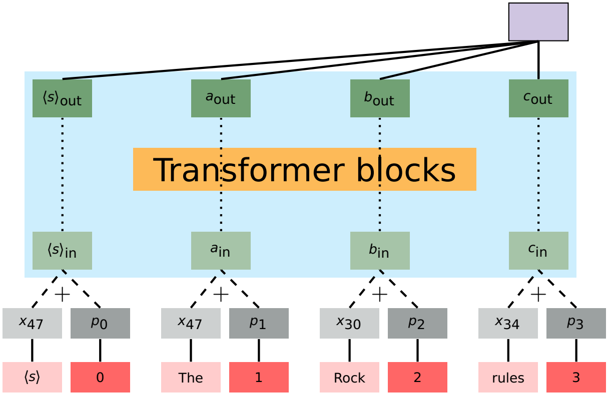
GPT: Fine-tuning



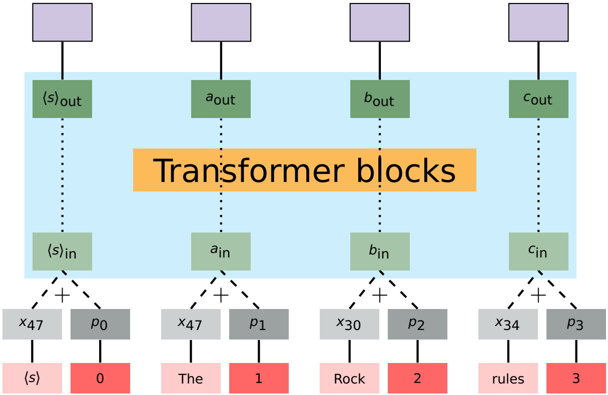
GPT: Fine-tuning



GPT: Fine-tuning



GPT: Fine-tuning



GPT: Scaling up from OpenAI

	Layers	d_k	d_{ff}	Parameters
GPT (Radford et al. 2018)	12	768	3,072	117M
GPT-2 (Radford et al. 2019)	48	1,600	1,600	1,542M
GPT-3 (Brown et al. 2020)	96	12,288	?	175,000M

The GPT-3 paper also reports on models ranging in size from 125M to 13B (Table 2.1).

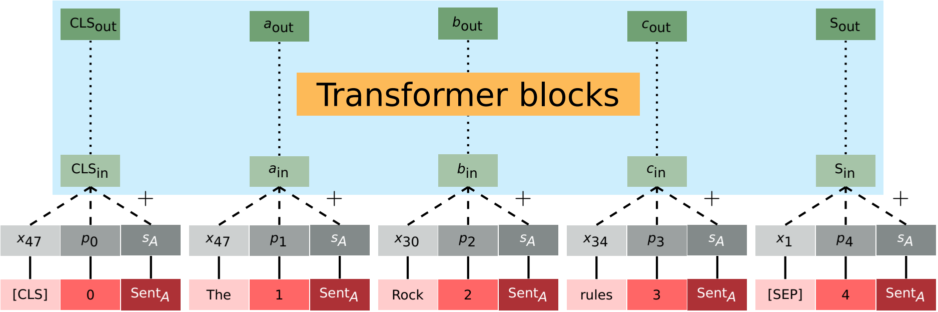
GPT: Scaling up truly open models

	Layers	d_k	d_{ff}	Parameters
GPT-Neo (Eleuther)	24	2,048	2,048	2,700M
GPT-J (Eleuther)	28	4,096	16,384	6,000M
GPT-NeoX (Eleuther)	44	6,144	6,144	20,000M
OPT-66B (Zhang et al. 2022)	64	9,216	9,216	66,000M
BLOOM (Scao et al. 2022)	70	14,336	14,336	176,247M

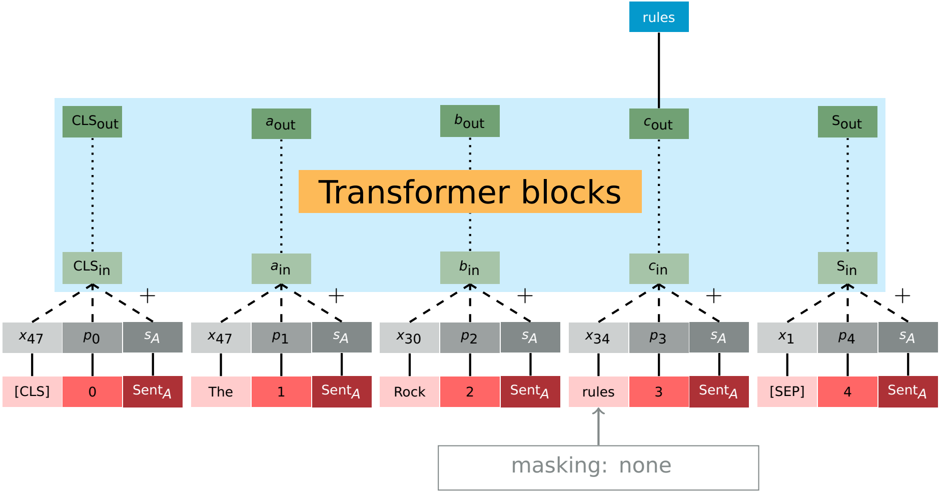
This table will be out of date by the time anyone reads it, if not before!

BERT

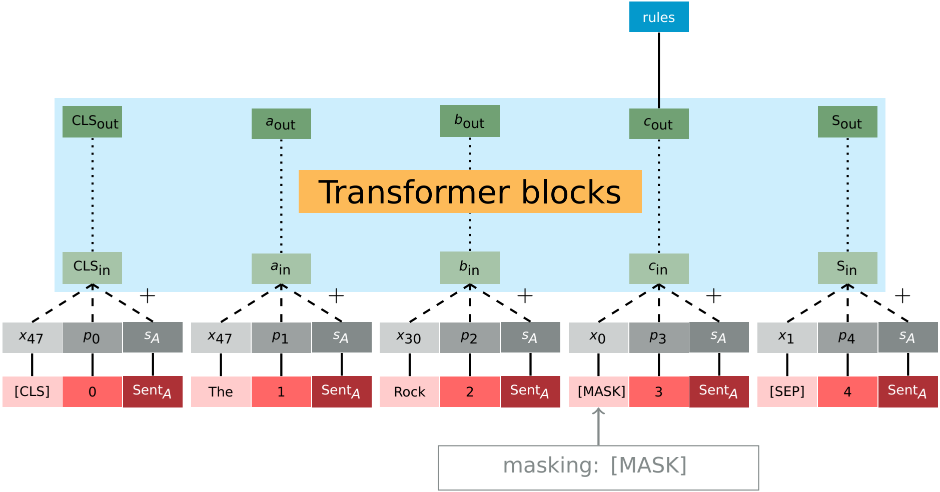
BERT: Core model structure



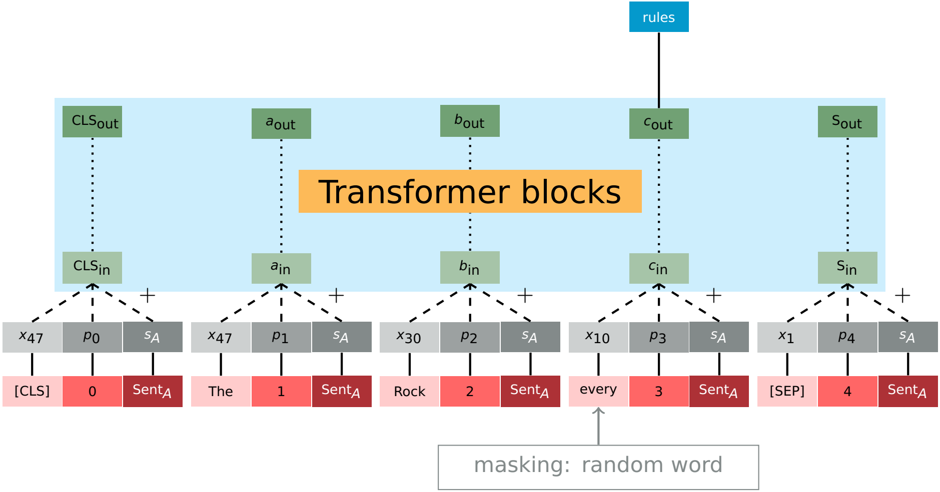
Masked Language Modeling (MLM)



Masked Language Modeling (MLM)



Masked Language Modeling (MLM)



BERT: MLM loss function

For Transformer parameters H_θ and sequence $\mathbf{x} = [x_1, \ldots, x_T]$ with masked version $\hat{\mathbf{x}}$:

$$\max_\theta \sum_{t=1}^{T} m_t \log \frac{\exp\left(e(x_t)^\top H_\theta(\hat{\mathbf{x}})_t\right)}{\sum_{x' \in \mathcal{V}} \exp\left(e(x')^\top H_\theta(\hat{\mathbf{x}})_t\right)}$$

where $\mathcal{V}$ is the vocabulary, x_t is the actual token at step t, $m_t = 1$ if token t was masked, else 0, and $e(x)$ is the embedding for x.

Binary next sentence prediction pretraining

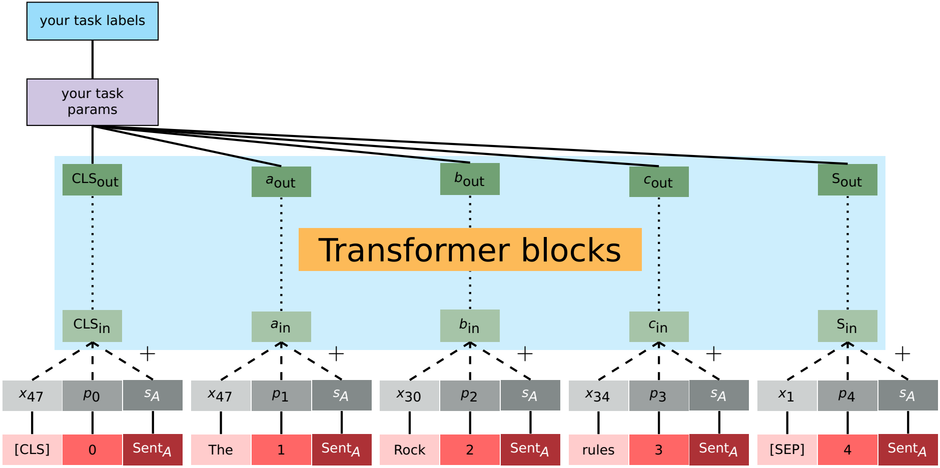
Positive: Actual sentence sequences

- [CLS] the man went to [MASK] store [SEP]
- he bought a gallon [MASK] milk [SEP]
- Label: `IsNext`

Negative: Randomly chosen second sentence

- [CLS] the man went to [MASK] store [SEP]
- penguin [MASK] are flight ##less birds [SEP]
- Label: `NotNext`

BERT: Transfer learning and fine-tuning



Tokenization and the BERT embedding space

```
[1]: from transformers import BertTokenizer

[2]: tokenizer = BertTokenizer.from_pretrained('bert-base-cased')

[3]: tokenizer.tokenize("This isn't too surprising.")

[3]: ['This', 'isn', "'", 't', 'too', 'surprising', '.']

[4]: tokenizer.tokenize("Encode me!")

[4]: ['En', '##code', 'me', '!']

[5]: tokenizer.tokenize("Snuffleupagus?")

[5]: ['S', '##nu', '##ffle', '##up', '##agu', '##s', '?']

[6]: tokenizer.vocab_size

[6]: 28996
```

BERT: Core model releases

	Layers	d_k	d_{ff}	Parameters
BERT-tiny	2	128	512	4M
BERT-mini	4	246	1,024	11M
BERT-small	4	512	2,048	29M
BERT-medium	8	512	2,048	41M
BERT-base	12	768	3,072	110M
BERT-large	24	1,024	4,096	340M

Limited to sequences of 512 tokens due to dimensionality of the positional embeddings.

Many new releases at the project site and on Hugging Face, including BERT variants for different languages.

BERT: Known limitations

1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.
2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.”
3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch”
4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language”

RoBERTa

Addressing the known limitations with BERT

1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.
2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.”
3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch”
4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language”

Robustly **o**ptimized **BERT a**pproach

BERT	RoBERTa
Static masking/substitution	Dynamic masking/substitution
Inputs are two concatenated document segments	Inputs are sentence sequences that may span document boundaries
Next Sentence Prediction (NSP)	No NSP
Training batches of 256 examples	Training batches of 2,000 examples
Word-piece tokenization	Character-level byte-pair encoding
Pretraining on BooksCorpus and English Wikipedia	Pretraining on BooksCorpus, Wikipedia, CC-News, OpenWebText, Stories
Train for 1M steps	Train for up to 500K steps
Train on short sequences first	Train only on full-length sequences

Additional differences in the optimizer and data presentation (sec 3.1).

RoBERTa results informing final system design

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

RoBERTa results informing final system design

RoBERTa choice for efficient batching, and comparisons with related work.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

RoBERTa results informing final system design

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

RoBERTa results informing final system design

Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
BERT$_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
XLNet$_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of BERT$_{\text{LARGE}}$. Results for BERT$_{\text{LARGE}}$ and XLNet$_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

RoBERTa: Core model releases

	Layers	d_k	d_{ff}	Parameters
RobERTa-base	12	768	3,072	125M
RobERTa-large	24	1,024	4,096	355M

Related work

A Primer in BERTology: What we know about how BERT works

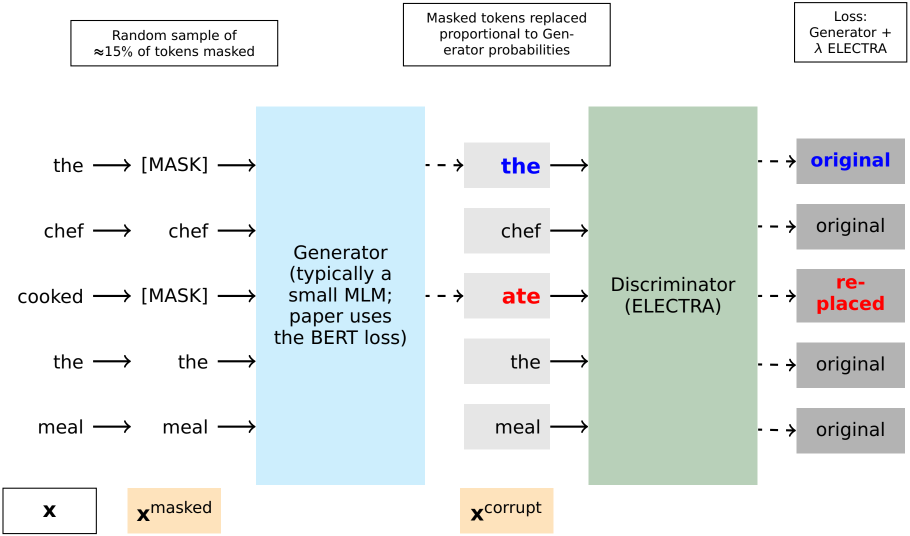
Anna Rogers, Olga Kovaleva, Anna Rumshisky
Department of Computer Science, University of Massachusetts Lowell
Lowell, MA 01854
{arogers, okovalev, arum}@cs.uml.edu

ELECTRA

Addressing the known limitations with BERT

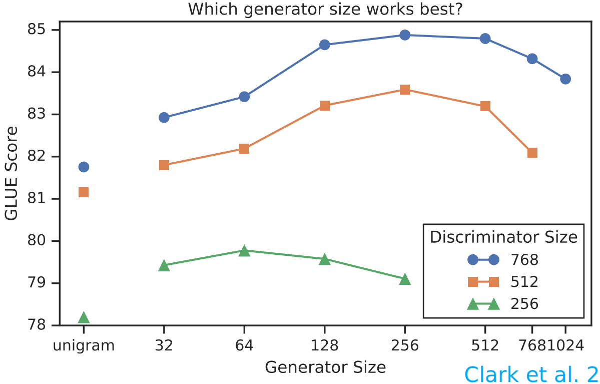
1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.
2. Devlin et al. (2019): "The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning."
3. Devlin et al. (2019): "The second downside of using an MLM is that only 15% of tokens are predicted in each batch"
4. Yang et al. (2019): "BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language"

Core model structure (Clark et al. 2019)



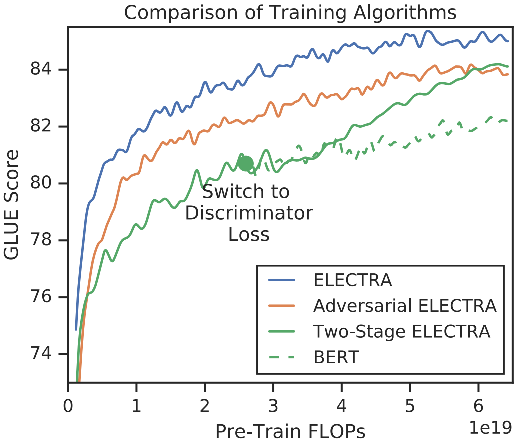
Generator/Discriminator relationships

Where Generator and Discriminator are the same size, they can share Transformer parameters, and more sharing is better. However, the best results come from having a Generator that is small compared to the Discriminator:



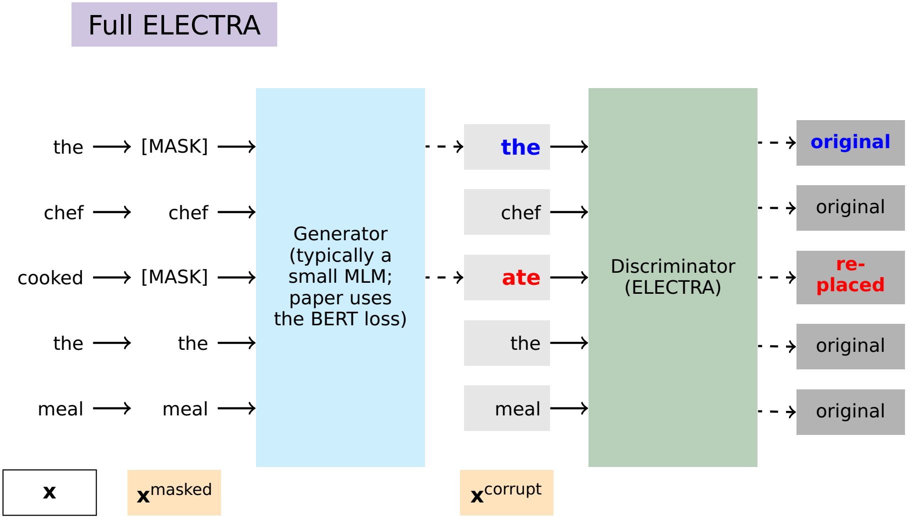
Clark et al. 2019, Figure 3

Efficiency



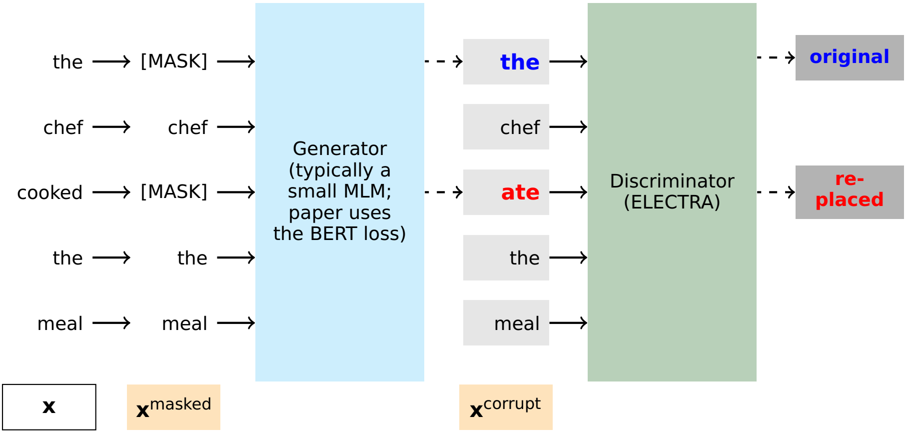
Clark et al. 2019, Figure 3

ELECTRA efficiency analyses



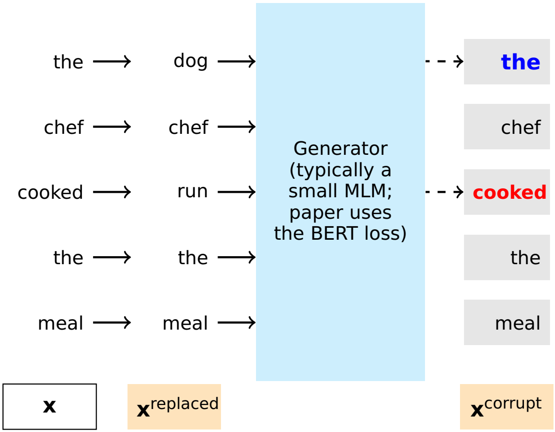
ELECTRA efficiency analyses

ELECTRA 15%



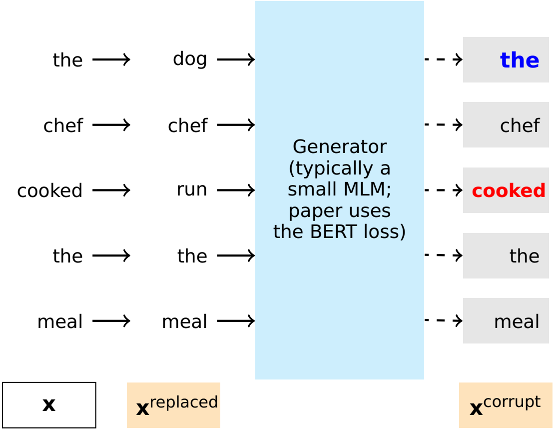
ELECTRA efficiency analyses

Replace MLM



ELECTRA efficiency analyses

All-tokens MLM



ELECTRA efficiency analyses

Model	GLUE score
ELECTRA	**85.0**
All-tokens MLM	84.3
Replace MLM	82.4
ELECTRA 15%	82.4
BERT	82.2

ELECTRA model releases

Available from the project site:

Model	Layers	Hidden Size	Params	GLUE test
Small	12	256	14M	77.4
Base	12	768	110M	82.7
Large	24	1024	335M	85.2

'Small' is the model designed to be "quickly trained on a single GPU".

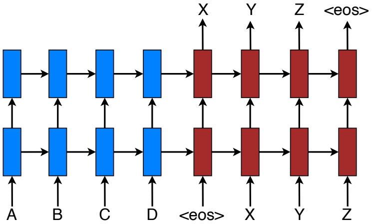
seq2seq architectures

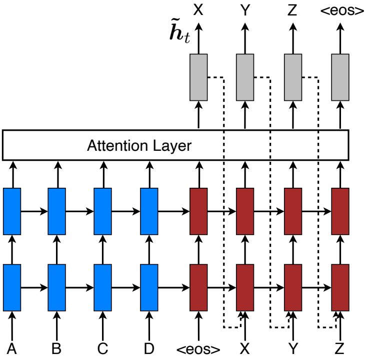
Some tasks with natural seq2seq structure

1. Machine translation (language to language)
2. Summarization (text to shorter text)
3. Free-form question answering (question to answer)
4. Dialogue (utterance to utterance)
5. Semantic parsing (sentence to logical form)
6. Code generation (sentence to program)
7. ...

More general class: encoder–decoder (agnostic about whether the encoding and decoding involve sequences).

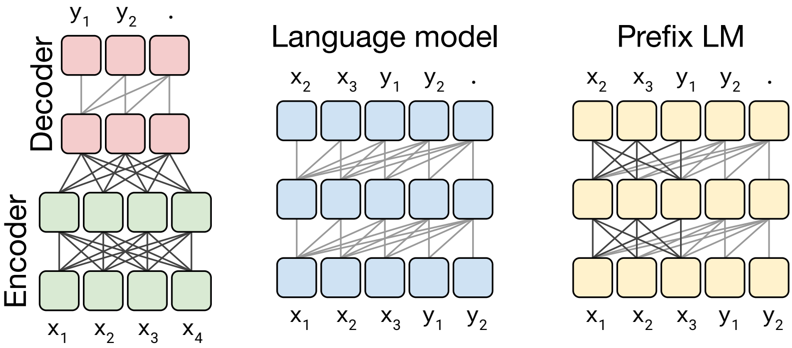
From the RNN era





From Luong et al. 2015, Figures 1 and 4

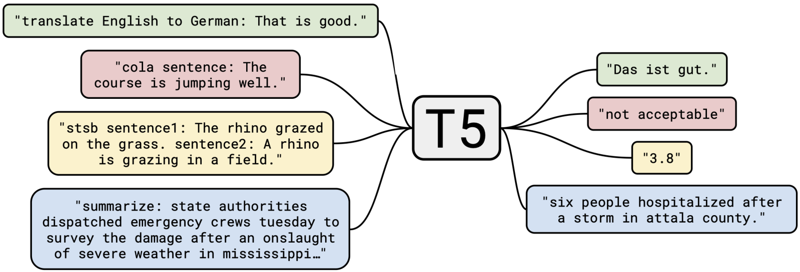
Transformer-based options



From Raffel et al. 2019, Figure 4

T5

Encoder–Decoder variant with extensive multi-task supervised and unsupervised training. Input task-prefixes guide model behavior. Lots of pretrained versions.



From Raffel et al. 2019, Figure 1

T5 model releases

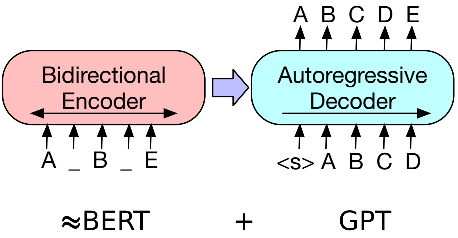
See https://huggingface.co/docs/transformers/model_doc/t5

	Layers	d_k	d_{ff}	Parameters
T5-small	6	512	2,048	60M
T5-base	12	768	3,072	220M
T5-large	24	1,024	4,096	770M
T5-3B	24	1,024	16,384	3,000M
T5-11B	24	1,024	65,536	11,000M

Also: FLAN-T5 models (instruction-tuned) of Chung et al. 2022

BART

- Text infilling
- Sentence shuffling
- Token masking
- Token deletion
- Document rotation



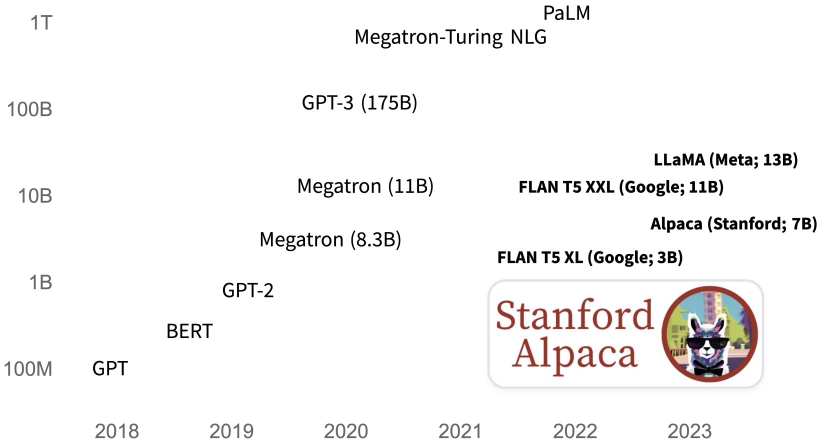
Fine-tuning

- Classification: Uncorrupted copies of the input are fed to the encoder and the decoder. The final decoder state is usually the basis for classification.
- seq2seq: Standard encoder–decoder usage.

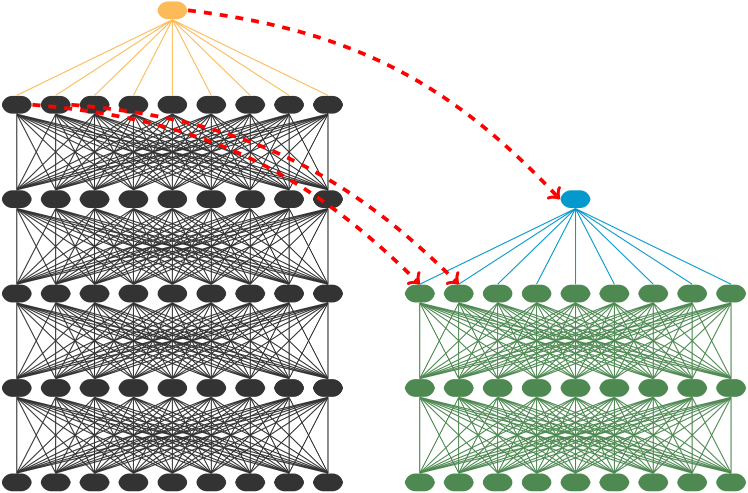
Lewis et al. 2019

Distillation

Trends in model size



Teachers and students



Distillation objectives

From least to most heavy duty; weighted averages of these are common:

0. Gold data for the task.
1. Teacher's output labels.
2. Teacher's output scores (Hinton et al. 2015).
3. Teacher's final output states (cosine loss; Sanh et al. 2019).
4. Other teacher hidden states and embeddings (Romero et al. 2015)
5. Student is trained to mimic the counterfactual behavior of the teacher under interventions (Wu et al. 2022).

For more: Gou et al. 2021

Distillation objectives

These can be used in combination to some extent:

1. Standard: One teacher with frozen parameters; only student parameters are updated.
2. Multi-teacher: An ensemble for the same task or potentially multi-task.
3. Co-distillation: The teacher and student are trained jointly. Also called 'online distillation' (Anil et al. 2018).
4. Self-distillation: The objective includes terms that seek to make some model components align with others (Zhang et al. 2019).

For more: Gou et al. 2021

Distillation performance

Distillation has been applied in many domains. The following point to a core result for GLUE (Wang et al. 2018): via distillation, we can increase efficiency with almost no loss in performance.

1. Sanh et al. (2019): Distill 12-layer BERT-base into 6 layers retaining 97% of GLUE performance.

2. Sun et al. (2019): Distill BERT-base into 3-layer and 6-layer variants, also maintaining good performance on GLUE.

3. Jiao et al. (2020): Distill BERT-base into 4-layers with similarly strong GLUE results.

Wrap-up

Other noteworthy architectures

- Transformer XL (Dai et al. 2019): Long contexts via recurrent connections to previous (frozen) states.
- XLNet (Yang et al. 2019): Bidirectional context with an autoregressive loss, via sampling of different sequence orders.
- DeBERTa (He et al. 2021): Separate representations for word and positions, with distinct attention connections.

BERT: Known limitations

1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies. RoBERTa

2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.” ELECTRA

3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch” ELECTRA

4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language” XLNet!

Pretraining data

1. OpenBookCorpus (Bandy and Vincent 2021):
   https://huggingface.co/datasets/bookcorpusopen
2. The Pile (Gao et al. 2020):
   https://pile.eleuther.ai
3. Big Science Data (Laurençon et al. 2022):
   https://huggingface.co/bigscience-data
4. Wikipedia processing:
   https://github.com/attardi/wikiextractor
5. Pushshift Reddit Data (Baumgartner et al. 2020):
   https://files.pushshift.io/reddit/

Current trends

1. Autoregressive architectures seem to have taken over, possibly just because the field is focused on generation.
2. Bidirectional models may still have the edge when it comes to representation.
3. seq2seq is still a dominant choice for tasks with that structure.
4. People are still obsessed with scaling up, but we are seeing a counter movement towards “smaller” models (still $\approx$10B parameters).

References I

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. 2018. Large scale distributed neural network training through online distillation. *ArXiv*, abs/1804.03235.

Jack Bandy and Nicholas Vincent. 2021. Addressing" documentation debt" in machine learning research: A retrospective datasheet for bookcorpus. *arXiv preprint arXiv:2105.05241*.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The PushShift Reddit dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 830–839.

T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, P. Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. ArXiv:1901.02860.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *Proceedings of the International Conference on Learning Representations*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. ArXiv:1503.02531.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

References II

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. 2022. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ArXiv:1910.13461.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. ArXiv:1907.11692.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Leon Bottou, Max Welling, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Ms, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. ArXiv:2002.12327.

References III

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*.

Alexander Rush. 2018. The annotated Transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. ArXiv:1803.02155.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhengxuan Wu, Atticus Geiger, Josh Rozner, Elisa Kreiss, Hanson Lu, Thomas Icard, Christopher Potts, and Noah D. Goodman. 2022. Causal distillation for language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4295, Seattle, United States. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

References IV

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3712–3721.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained Transformer language models. *arXiv preprint arXiv:2205.01068*.