Transformers and transfer learning

CS 585, Fall 2019

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst

some slides from Emma Strubell, Matt Peters, and Jacob Devlin
stuff from last time

- HW2 out now! please start early, as it is fairly long and may prove difficult to implement
- can you repeat questions asked during class? ok
Self-attention

Layer $p$

Q K V

Nobel committee awards Strickland who advanced optics

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]

Layer $p$

Q: Nobel committee
K: awards
V: Strickland

who advanced optics
Self-attention

[Vaswani et al. 2017]

Layer $p$

$Q$

$K$

$V$

optics
advanced
who
Strickland
awards
committee
Nobel

Nobel committee awards Strickland who advanced optics
Self-attention

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]

Layer $p$

Optics
Advanced
Who
Strickland
Awards
Committee
Nobel

Q
K
V

Layer $p$

Nobel
Committee
Awards
Strickland
Who
Advanced
Optics
Self-attention

\[ M \]

\[ Q \]

\[ K \]

\[ V \]

Layer \( p \)

Nobel committee awards Strickland who advanced optics

[optics advanced who Strickland awards committee Nobel]

[Vaswani et al. 2017]
Self-attention

\[ M \]
\[ A \]
\[ Q \]
\[ K \]
\[ V \]

Layer \( p \)

Nobel committee awards Strickland who advanced optics

[Vaswani et al. 2017]
Multi-head self-attention

[Vaswani et al. 2017]
Multi-head self-attention

Layer \( p \)

Nobel, committee, awards, Strickland, who, advanced, optics

optics, advanced, who, Strickland, awards, committee, Nobel

\([\text{Vaswani et al. 2017}]\)
Multi-head self-attention

Layer $p$

Layer $p+1$

Nobel committee awards Strickland who advanced optics

[optics advanced who Strickland awards committee Nobel A]

[Viswanati et al. 2017]
Multi-head self-attention

[Vaswani et al. 2017]

Layer $p+1$

Nobel committee awards Strickland who advanced optics
Multi-head self-attention

Layer 1

Layer p

Layer J

[Vaswani et al. 2017]
These are residual connections
Positional encoding

**EMBEDDING WITH TIME SIGNAL**

**POSITIONAL ENCODING**

**EMBEDDINGS**

**INPUT**

Je

suis

étudiant
Positional encoding

\[ PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \]

\[ PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \]

Positional encoding is a 512d vector

\( i \) = a particular dimension of this vector

\( pos \) = position of the word in the text

\( d_{model} = 512 \)
What does this look like?*

Sin
Even parts

Cos
Odd parts
Absolute vs relative difference?

<table>
<thead>
<tr>
<th>Model</th>
<th>Position Information</th>
<th>EN-DE BLEU</th>
<th>EN-FR BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (base)</td>
<td>Absolute Position Representations</td>
<td>26.5</td>
<td>38.2</td>
</tr>
<tr>
<td>Transformer (base)</td>
<td>Relative Position Representations</td>
<td><strong>26.8</strong></td>
<td><strong>38.7</strong></td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>Absolute Position Representations</td>
<td>27.9</td>
<td>41.2</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>Relative Position Representations</td>
<td><strong>29.2</strong></td>
<td><strong>41.5</strong></td>
</tr>
</tbody>
</table>

Shaw et al., NAACL 2018
What's going on here?
Last major missing piece:

- Decoder self-attention masking
## Ablations

<table>
<thead>
<tr>
<th></th>
<th>(N)</th>
<th>(d_{model})</th>
<th>(d_{ff})</th>
<th>(h)</th>
<th>(d_k)</th>
<th>(d_v)</th>
<th>(P_{drop})</th>
<th>(\epsilon_{ls})</th>
<th>train steps</th>
<th>PPL (dev)</th>
<th>BLEU (dev)</th>
<th>params (\times 10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
<td>512</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>100K</td>
<td>4.92</td>
<td>25.8</td>
<td>65</td>
</tr>
<tr>
<td>(A)</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>512</td>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td>5.29</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>128</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td>5.00</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>4.91</td>
<td>25.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td></td>
</tr>
<tr>
<td>(B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.16</td>
<td>25.1</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td>60</td>
</tr>
<tr>
<td>(C)</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.11</td>
<td>23.7</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.19</td>
<td>25.3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.88</td>
<td>25.5</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.75</td>
<td>24.5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.66</td>
<td>26.0</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.12</td>
<td>25.4</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.75</td>
<td>26.2</td>
<td>90</td>
</tr>
<tr>
<td>(D)</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.77</td>
<td>24.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.95</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.67</td>
<td>25.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.47</td>
<td>25.7</td>
<td></td>
</tr>
<tr>
<td>(E)</td>
<td></td>
<td>positional embedding instead of sinusoids</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.92</td>
<td>25.7</td>
<td></td>
</tr>
<tr>
<td>big</td>
<td>6</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td></td>
<td></td>
<td>0.3</td>
<td></td>
<td>300K</td>
<td>4.33</td>
<td>26.4</td>
<td>213</td>
</tr>
</tbody>
</table>
Hacks to get it to work:
Optimizer

We used the Adam optimizer (cite) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula: $lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$. This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup\_steps = 4000$.

Note: This part is very important. Need to train with this setup of the model.
Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has **confidence** of the correct word and the rest of the **smoothing** mass distributed throughout the vocabulary.

---

I went to class and took ___

cats   TV    notes    took    sofa

0      0      1       0      0

0.025  0.025  0.9     0.025  0.025

with label smoothing
Get penalized for overconfidence!

Loss

Target word confidence
### Byte pair encoding (BPE)

- Deal with rare words / large vocabulary by instead using *subword* tokenization

<table>
<thead>
<tr>
<th>system</th>
<th>sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>health research institutes</td>
</tr>
<tr>
<td>reference</td>
<td>Gesundheitsforschungsinstitute</td>
</tr>
<tr>
<td>WDict</td>
<td>Forschungsinstitute</td>
</tr>
<tr>
<td>C2-50k</td>
<td>Forschungs</td>
</tr>
<tr>
<td>BPE-60k</td>
<td>Gesundheits</td>
</tr>
<tr>
<td>BPE-J90k</td>
<td>Gesundheits</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>system</th>
<th>sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>asinine situation</td>
</tr>
<tr>
<td>reference</td>
<td>dumme Situation</td>
</tr>
<tr>
<td>WDict</td>
<td>asinine situation → UNK → asinine</td>
</tr>
<tr>
<td>C2-50k</td>
<td>as</td>
</tr>
<tr>
<td>BPE-60k</td>
<td>as</td>
</tr>
<tr>
<td>BPE-J90K</td>
<td>as</td>
</tr>
</tbody>
</table>

Sennrich et al., ACL 2016
exercise
transfer learning
Do NNs really need millions of labeled examples?

- Can we leverage *unlabeled* data to cut down on the number of labeled examples we need?
What is transfer learning?

• In our context: take a network trained on a task for which it is easy to generate labels, and adapt it to a different task for which it is harder.

• In computer vision: train a CNN on ImageNet, transfer its representations to every other CV task

• In NLP: train a really big language model on billions of words, transfer to every NLP task!
can we use language models to produce word embeddings?

Deep contextualized word representations. Peters et al., NAACL 2018
Word vectors are ubiquitous

Most if not all current state-of-the-art NLP systems use pre-trained word embeddings* (as of 2018)

* With the exception of data-rich tasks like machine translation
word2vec represents each word as a single vector

\[ \text{play} = [0.2, -0.1, 0.5, \ldots] \]

\[ \text{bank} = [-0.3, 1.4, 0.7, \ldots] \]

\[ \text{run} = [-0.5, -0.3, -0.1, \ldots] \]
The new-look *play* area is due to be completed by early spring 2010.
Gerrymandered congressional districts favor representatives who play to the party base.
The freshman then completed the three-point play for a 66-63 lead.
**Nearest neighbors**

\[
\text{play} = [0.2, -0.1, 0.5, \ldots]
\]

**Nearest Neighbors**

<table>
<thead>
<tr>
<th>playing</th>
<th>plays</th>
</tr>
</thead>
<tbody>
<tr>
<td>game</td>
<td>player</td>
</tr>
<tr>
<td>games</td>
<td>Play</td>
</tr>
<tr>
<td>played</td>
<td>football</td>
</tr>
<tr>
<td>players</td>
<td>multiplayer</td>
</tr>
</tbody>
</table>
Multiple senses entangled

\[
\text{play} = [0.2, -0.1, 0.5, \ldots]
\]

Nearest Neighbors

<table>
<thead>
<tr>
<th>playing</th>
<th>VERB</th>
<th>plays</th>
</tr>
</thead>
<tbody>
<tr>
<td>game</td>
<td></td>
<td>player</td>
</tr>
<tr>
<td>games</td>
<td></td>
<td>Play</td>
</tr>
<tr>
<td>played</td>
<td></td>
<td>football</td>
</tr>
<tr>
<td>players</td>
<td></td>
<td>multiplayer</td>
</tr>
</tbody>
</table>
Multiple senses entangled

\[
\text{play} = [0.2, -0.1, 0.5, \ldots]
\]

**Nearest Neighbors**

- playing
- game
- games
- played
- players

**VERB**

- plays
- player
- Play
- football
- multiplayer

**NOUN**
Multiple senses entangled

\[ \text{play} = [0.2, -0.1, 0.5, \ldots] \]

Nearest Neighbors

- playing
- game
- games
- played
- players

- VERB: plays, player, Play
- NOUN: games, football, multiplayer
- ADJ:
Contextual Representations

- **Problem**: Word embeddings are applied in a context free manner

  \[
  \begin{align*}
  \text{open a bank account} & \quad \text{on the river bank} \\
  [0.3, 0.2, -0.8, \ldots] & \\
  \end{align*}
  \]

- **Solution**: Train *contextual* representations on text corpus

  \[
  \begin{align*}
  [0.9, -0.2, 1.6, \ldots] & \quad [-1.9, -0.4, 0.1, \ldots] \\
  \uparrow & \quad \uparrow \\
  \text{open a bank account} & \quad \text{on the river bank} \\
  \end{align*}
  \]
History of Contextual Representations

- **Semi-Supervised Sequence Learning**, Google, 2015

```
Train LSTM Language Model

open LSTM open a bank ...

Fine-tune on Classification Task

LSTM LSTM LSTM
very funny movie POSITIVE
```
History of Contextual Representations

- **ELMo: Deep Contextual Word Embeddings**, AI2 & University of Washington, 2017

Train Separate Left-to-Right and Right-to-Left LMs

Apply as “Pre-trained Embeddings”

Existing Model Architecture
Deep bidirectional language model

... download new games or play ??
Deep bidirectional language model

… download new games or play ??
Deep bidirectional language model

... download  new  games  or  play  ??
Deep bidirectional language model

… download new games or play ??
Deep bidirectional language model

... download new games or

play ??
Deep bidirectional language model

… download new games or play ??
Deep bidirectional language model

... download     new      games        or

play        ??
Deep bidirectional language model

... download     new      games        or     play
Use all layers of language model

… games or play online via …

0.25

0.6

0.15

embeddings from language models

ELMo
Learned task-specific combination of layers

ELMo embeddings from language models

... games or play online via ...
The biLM produces $2L + 1$ intermediate representations:

$$R_k = \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j = 1, \ldots, L\}$$

$$= \{h_{k,j}^{LM} \mid j = 0, \ldots, L\}$$

where $h_{k,0}^{LM} = x_k^{LM}$ is the token layer and

$h_{k,j}^{LM} = [\overrightarrow{h}_{k,j}^{LM} ; \overleftarrow{h}_{k,j}^{LM}]$, for each biLSTM layer.

ELMo: A task specific combination of these features:

$$\text{ELMo}_{k}^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}.$$ 

where $s_j^{task}$ are softmax-normalized weights and $\gamma^{task}$ is a scaling parameter.
The biLM produces $2L + 1$ intermediate representations:

$$R_k = \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} \mid j = 1, \ldots, L\}$$

$$= \{h_{k,j}^{LM} \mid j = 0, \ldots, L\}$$

where $h_{k,0}^{LM} = x_k^{LM}$ is the token layer and $h_{k,j}^{LM} = [\overrightarrow{h}_{k,j}^{LM}; \overleftarrow{h}_{k,j}^{LM}]$, for each biLSTM layer.

ELMo: A task specific combination of these features:

$$\text{ELMo}_{k}^{\text{task}} = E(R_k; \Theta^{\text{task}}) = \gamma^{\text{task}} \sum_{j=0}^{L} s_{j}^{\text{task}} h_{k,j}^{LM}.$$

where $s_{j}^{\text{task}}$ are softmax-normalized weights and $\gamma^{\text{task}}$ is a scaling parameter.
ELMo representations are **contextual** – they depend on the entire sentence in which a word is used.

how many different embeddings does ELMo compute for a given word?
ELMo improves NLP tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Previous SOTA</th>
<th>Our</th>
<th>ELMo + Baseline</th>
<th>Increase (Absolute/Relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQuAD</td>
<td>Liu et al. (2017)</td>
<td>84.4</td>
<td>81.1</td>
<td>85.8</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>88.7 ± 0.17</td>
</tr>
<tr>
<td>SRL</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>84.6</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>70.4</td>
</tr>
<tr>
<td>NER</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>92.22 ± 0.10</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>54.7 ± 0.5</td>
</tr>
</tbody>
</table>
Large-scale recurrent neural language models learn contextual representations that capture basic elements of semantics and syntax.

Adding ELMo to existing state-of-the-art models provides significant performance improvement on all NLP tasks.

```
elmo = hub.Module("https://tfhub.dev/google/elmo/1", trainable=True)
embeddings = elmo(["the cat is on the mat", "dogs are in the fog"],
signature="default",
as_dict=True)["elmo"]
```
Problem with Previous Methods

- **Problem**: Language models only use left context or right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
Problem with Previous Methods

- **Problem**: Language models only use left context or right context, but language understanding is bidirectional.
- Why are LMs unidirectional?
- **Reason 1**: Directionality is needed to generate a well-formed probability distribution.
  - We don’t care about this. **Why not?**
Problem with Previous Methods

- **Problem**: Language models only use left context or right context, but language understanding is bidirectional.

- Why are LMs unidirectional?
  - **Reason 1**: Directionality is needed to generate a well-formed probability distribution.
    - We don’t care about this.
  - **Reason 2**: Words can “see themselves” in a bidirectional encoder.
Unidirectional context
Build representation incrementally

Bidirectional context
Words can “see themselves”
Masked LM

● **Solution:** Mask out $k\%$ of the input words, and then predict the masked words
  ○ We always use $k = 15\%$

The man went to the [MASK] to buy a [MASK] of milk

What are the pros and cons of increasing $k$?
Problem: Mask token never seen at fine-tuning
Solution: 15% of the words to predict, but don’t replace with \[\text{[MASK]}\] 100% of the time. Instead:
- 80% of the time, replace with \[\text{[MASK]}\]
  went to the store → went to the the \[\text{[MASK]}\]
- 10% of the time, replace random word
  went to the store → went to the running
- 10% of the time, keep same
  went to the store → went to the store
Next Sentence Prediction

- To learn *relationships* between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence.

<table>
<thead>
<tr>
<th>Sentence A</th>
<th>Sentence B</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>The man went to the store.</td>
<td>He bought a gallon of milk.</td>
<td>IsNextSentence</td>
</tr>
<tr>
<td>The man went to the store.</td>
<td>Penguins are flightless.</td>
<td>NotNextSentence</td>
</tr>
</tbody>
</table>
**Input Representation**

- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
- Single sequence is much more efficient.
Model Architecture

Transformer encoder

- **Multi-headed self attention**
  - Models context
- **Feed-forward layers**
  - Computes non-linear hierarchical features
- **Layer norm and residuals**
  - Makes training deep networks healthy
- **Positional embeddings**
  - Allows model to learn relative positioning
Model Architecture

- Empirical advantages of Transformer vs. LSTM:

  What are they?
Empirical advantages of Transformer vs. LSTM:

1. Self-attention == no locality bias
   - Long-distance context has “equal opportunity”
2. Single multiplication per layer == efficiency on TPU
   - Effective batch size is number of words, not sequences
Model Details

- **Data:** Wikipedia (2.5B words) + BookCorpus (800M words)
- **Batch Size:** 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- **Training Time:** 1M steps (~40 epochs)
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay
- **BERT-Base:** 12-layer, 768-hidden, 12-head
- **BERT-Large:** 24-layer, 1024-hidden, 16-head
- **Trained on:** 4x4 or 8x8 TPU slice for 4 days
Fine-Tuning Procedure
Fine-Tuning Procedure

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

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

(c) Question Answering Tasks: SQuAD v1.1

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

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</td>
<td></td>
</tr>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT\text{BASE}</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT\text{LARGE}</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>91.1</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>81.9</td>
</tr>
</tbody>
</table>

### MultiNLI

**Premise:** Hills and mountains are especially sanctified in Jainism.

**Hypothesis:** Jainism hates nature.

**Label:** Contradiction

### CoLa

**Sentence:** The wagon rumbled down the road.

**Label:** Acceptable

**Sentence:** The car honked down the road.

**Label:** Unacceptable
A girl is going across a set of monkey bars. She
(i) jumps up across the monkey bars.
(ii) struggles onto the bars to grab her head.
(iii) gets to the end and stands on a wooden plank.
(iv) jumps up and does a back flip.

- Run each Premise + Ending through BERT.
- Produce logit for each pair on token 0 (\([\text{CLS}]\))

\[
P_i = \frac{e^{V \cdot C_i}}{\sum_{j=1}^{4} e^{V \cdot C_j}}
\]
Effect of Pre-training Task

- Masked LM (compared to left-to-right LM) is very important on some tasks, Next Sentence Prediction is important on other tasks.
- Left-to-right model does very poorly on word-level task (SQuAD), although this is mitigated by BiLSTM
Effect of Directionality and Training Time

- Masked LM takes slightly longer to converge because we only predict 15% instead of 100%
- But absolute results are much better almost immediately
Effect of Model Size

- Big models help a lot
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples
- Improvements have not asymptoted
# Effect of Masking Strategy

- Masking 100% of the time hurts on feature-based approach
- Using random word 100% of time hurts slightly

<table>
<thead>
<tr>
<th>Masking Rates</th>
<th>Dev Set Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNLI</td>
</tr>
<tr>
<td></td>
<td>Fine-tune</td>
</tr>
<tr>
<td>80%</td>
<td>84.2</td>
</tr>
<tr>
<td>100%</td>
<td>84.3</td>
</tr>
<tr>
<td>80%</td>
<td>84.1</td>
</tr>
<tr>
<td>80%</td>
<td>84.4</td>
</tr>
<tr>
<td>0%</td>
<td>83.7</td>
</tr>
<tr>
<td>0%</td>
<td>83.6</td>
</tr>
</tbody>
</table>
Multilingual BERT

- Trained single model on 104 languages from Wikipedia. Shared 110k WordPiece vocabulary.

- XNLI is MultiNLI translated into multiple languages.
- Always evaluate on human-translated Test.
- **Translate Train**: MT English Train into Foreign, then fine-tune.
- **Translate Test**: MT Foreign Test into English, use English model.
- **Zero Shot**: Use Foreign test on English model.

<table>
<thead>
<tr>
<th>System</th>
<th>English</th>
<th>Chinese</th>
<th>Spanish</th>
</tr>
</thead>
<tbody>
<tr>
<td>XNLI Baseline - Translate Train</td>
<td>73.7</td>
<td>67.0</td>
<td>68.8</td>
</tr>
<tr>
<td>XNLI Baseline - Translate Test</td>
<td>73.7</td>
<td>68.4</td>
<td>70.7</td>
</tr>
<tr>
<td>BERT - Translate Train</td>
<td>81.9</td>
<td>76.6</td>
<td>77.8</td>
</tr>
<tr>
<td>BERT - Translate Test</td>
<td>81.9</td>
<td>70.1</td>
<td>74.9</td>
</tr>
<tr>
<td>BERT - Zero Shot</td>
<td>81.9</td>
<td>63.8</td>
<td>74.3</td>
</tr>
</tbody>
</table>
Common Questions

- Is deep bidirectionality really necessary? What about ELMo-style shallow bidirectionality on bigger model?
- Advantage: Slightly faster training time
- Disadvantages:
  - Will need to add non-pre-trained bidirectional model on top
  - Right-to-left SQuAD model doesn’t see question
  - Need to train two models
  - Off-by-one: LTR predicts next word, RTL predicts previous word
  - Not trivial to add arbitrary pre-training tasks.
Common Questions

● Why did no one think of this before?
● Better question: Why wasn’t contextual pre-training popular before 2018 with ELMo?
● Good results on pre-training is >1,000x to 100,000 more expensive than supervised training.
  ○ E.g., 10x-100x bigger model trained for 100x-1,000x as many steps.
  ○ Imagine it’s 2013: Well-tuned 2-layer, 512-dim LSTM sentiment analysis gets 80% accuracy, training for 8 hours.
  ○ Pre-train LM on same architecture for a week, get 80.5%.
  ○ Conference reviewers: “Who would do something so expensive for such a small gain?”
Common Questions

● The model must be learning more than “contextual embeddings”

● Alternate interpretation: Predicting missing words (or next words) requires learning many types of language understanding features.
  ○ syntax, semantics, pragmatics, coreference, etc.

● Implication: Pre-trained model is much bigger than it needs to be to solve specific task

● Task-specific model distillation words very well
Common Questions

● Is modeling “solved” in NLP? I.e., is there a reason to come up with novel model architectures?
  ○ But that’s the most fun part of NLP research :(

● Maybe yes, for now, on some tasks, like SQuAD-style QA.
  ○ At least using the same deep learning “lego blocks”

● Examples of NLP models that are not “solved”:
  ○ Models that minimize total training cost vs. accuracy on modern hardware
  ○ Models that are very parameter efficient (e.g., for mobile deployment)
  ○ Models that represent knowledge/context in latent space
  ○ Models that represent structured data (e.g., knowledge graph)
  ○ Models that jointly represent vision and language
Common Questions

● Personal belief: Near-term improvements in NLP will be mostly about making clever use of “free” data.
  ○ Unsupervised vs. semi-supervised vs. synthetic supervised is somewhat arbitrary.
  ○ “Data I can get a lot of without paying anyone” vs. “Data I have to pay people to create” is more pragmatic distinction.

● No less “prestigious” than modeling papers:
  ○ *Phrase-Based & Neural Unsupervised Machine Translation*, Facebook AI Research, EMNLP 2018 Best Paper
Conclusions

- Empirical results from BERT are great, but biggest impact on the field is:
- With pre-training, bigger == better, without clear limits (so far).
- Unclear if adding things on top of BERT really helps by very much.
  - Good for people and companies building NLP systems.
  - Not necessary a “good thing” for researchers, but important.