Large Language Models

CSCI 601 471/671
NLP: Self-Supervised Models

https://self-supervised.cs.jhu.edu/sp2023/

[Slide credit: Chris Tanner, Jacob Devlin and many others]
Logistics Update

- The midterm:
  - will be on March 7 during class time.
  - I will not be here; Adam (TA) will run the show.
  - it will be on paper
  - It will be based on the ideas you have seen in homework and lectures. If you understand them, you’re set!
  - Scope HW 1-5 and lectures until last Thursday (Feb 23)
Recap: Self-Attention

- $b_i$ is obtained based on the whole input sequence.
- can be parallelly computed.

Idea: replace any thing done by RNN with self-attention.

“Neural machine translation by jointly learning to align and translate” Bahdanau etl. 2014;
“Attention is All You Need” Vaswani et al. 2017

[adopted from Hung-yi Lee]
Recap: RNN vs Transformer
Recap: Attention Block

Given input $\mathbf{x}$:

$$ Q = \mathbf{W}^q \mathbf{x} $$
$$ K = \mathbf{W}^k \mathbf{x} $$
$$ V = \mathbf{W}^v \mathbf{x} $$

$$ \text{Attention}(\mathbf{x}) = \text{softmax} \left( \frac{QK^T}{\sqrt{h}} \right) V $$

[Attention Is All You Need, Vaswani et al. 2017]
Recap: Transformer [Vaswani et al. 2017]

- An encoder-decoder architecture
- 3 forms of attention

[Attention Is All You Need, Vaswani et al. 2017]
After Transformer ...
<table>
<thead>
<tr>
<th>Module Level</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local Transformer [156], Gaussian Transformer [42]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictive Attention Transformer [143], Reformer [51], Lasyformer [159]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAMTL [98]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Attention [164], Hard-Coded Gaussian Attention [165], Synthesizer [131]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Li et al. [73], Deepspe and Nacenshpu [27], Talking-head Attention [119]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaborative MHA [7]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-head Attention</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive Attention Span [120], Multi-Scale Transformer [64]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic Routing [40, 74]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Position Encoding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Absolute</td>
<td>BERT [28], Wang et al. [198], FLOADER [85]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relative</td>
<td>Shaw et al. [116], Music Transformer [56], T5 [104], Transformer-XL [74]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeBERTa [59]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Rep.</td>
<td>TUF [43], Referencer [124]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implicit Rep.</td>
<td>Complex Embedding [140], R-Transformer [141], CPR [28]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Placment</td>
<td>post-LN [28, 83, 157], pre-LN [6, 17, 67, 136, 141]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substitutes</td>
<td>AdaNorm [153], scaled $L_2$ normalization [93], PowerNorm [121]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerNorm</td>
<td>ReZero-Transformer [1]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activa. Func.</td>
<td>Swish [106], GELU [14, 28], GLU [18]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPN</td>
<td>Enlarge Capacity</td>
<td>Product-key Memory [69], Gohard [71], Switch Transformer [36], Expert Prototyping [155], Hash Layer [119]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropping</td>
<td>All-Attention layer [137], Yang et al. [197]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arch. Level</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X-formers</td>
<td>Lite Transformer [48], Funnel Transformer [23], DeLigt [12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reformer [11], Predictive Attention Transformer [48], Transparent Attention [5]</td>
<td>Feedback Transformer [34]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACT</td>
<td>UT [26], Conditional Computation Transformer [7], DeoBERT [150], PABEE [171], Li et al. [179], Jia et al. [197]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reversability</td>
<td>Transformer-XL [24], Compressive Transformer [109], Memformer [147]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yoshida et al. [169], ERNIE-Doc [38]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchy</td>
<td>Micali et al. [92], HIBERT [166], Liu and Lapatka [86], Hi-Transformer [145], TENER [154], TNT [48]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alt. Arch.</td>
<td>ET [12], Macron Transformer [89], Sandwich Transformer [99], MAN [35], DARTSformer [167]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pre-Train</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder</td>
<td>BERT [28], RoBERTa [87], BigBird [163]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decoder</td>
<td>GPT [101], GPT-X [102], GPT-3 [12]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enc. Dec.</td>
<td>BART [72], T5 [104], Switch Transformer [36]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>App.</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>BERT [28], ELMo [121], Transformer-XL [24], Compressive Transformer [109], TENER [154]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CV</td>
<td>Image Transformer [94], DETR [113], ViT [133], Swin Transformer [88], ViViT [3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>Speech Transformer [11], Streaming Transformer [15], Reformer TTS [97], Music Transformer [56]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multimodal</td>
<td>VisualBERT [12], VLEHR [125], VideoBERT [128], M6 [81], Chimera [46], DALL-E [107], CogView [129]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Impact of Transformers

- A building block for a variety of LMs

  - **Encoders**
    - Examples: BERT, RoBERTa, SciBERT.
    - Captures bidirectional context. Wait, how do we pretrain them?

  - **Decoders**
    - Examples: GPT-2, GPT-3, LaMDA
    - Other name: causal or auto-regressive language model
    - Nice to generate from; can’t condition on future words

  - **Encoder-Decoders**
    - Examples: Transformer, T5, Meena
    - What’s the best way to pretrain them?
BERT Encoders
BERT

Bidirectional Encoder Representations from Transformers
**BERT**

*Bidirectional Encoder Representations from Transformers*

Like Bidirectional LSTMs (ELMo), let’s look in both directions.
BERT

Bidirectional Encoder Representations from Transformers

Let’s only use Transformer Encoders, no Decoders
BERT

**Bidirectional Encoder Representations from Transformers**

It’s a language model that builds rich representations via self-supervised learning (pre-training).
BERT (2018)

- Transformer based network to learn representations of language

- Improvements
  - Bi-directional LSTM -> Self-attention
  - Massive data
  - Masked-LM objective

Abstract

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement).

There are two existing strategies for applying pre-trained language representations to downstream tasks: feature-based and fine-tuning. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right sampling strategy.
BERT: Architecture

- Stacks of Transformer encoders

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
BERT: Architecture

- Model output dimension: 512

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
BERT is trained to uncover masked tokens.

brown 0.92
lazy 0.05
playful 0.03
Probing BERT Masked LM

- Making words forces BERT to use context in both directions to predict the masked word.

Paris is the [MASK] of France.

![Computed Results]

https://huggingface.co/bert-base-uncased
Probing BERT Masked LM

- Making words forces BERT to use context in both directions to predict the masked word.

Today is Tuesday, so tomorrow is [MASK].

Compute

Computation time on cpu: cached

<table>
<thead>
<tr>
<th>Word</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>friday</td>
<td>0.274</td>
</tr>
<tr>
<td>wednesday</td>
<td>0.211</td>
</tr>
<tr>
<td>thursday</td>
<td>0.139</td>
</tr>
<tr>
<td>monday</td>
<td>0.108</td>
</tr>
<tr>
<td>sunday</td>
<td>0.077</td>
</tr>
</tbody>
</table>

https://huggingface.co/bert-base-uncased
BERT: Pre-training Objective (1): Masked Tokens

- Randomly mask 15% of the tokens and train the model to predict them.

Use the output of the masked word’s position to predict the masked word.

Possible classes:
- All English words
- Aardvark
- Improvisation
- Zzyzyva

Input

[CLS] Let’s stick to [MASK] in this skit

FFNN + Softmax
BERT: Pre-training Objective (1): Masked Tokens

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

- Too little masking: Too expensive to train
- Too much masking: Underdefined (not enough context)
BERT: Pre-training Objective (2): Sentence Ordering

- Predict sentence ordering
  - Predict likelihood that sentence B belongs after sentence A
  - 1% IsNext
  - 99% NotNext

- 50% correct ordering, and 50% random incorrect ones

![Diagram of sentence ordering](image)

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
BERT: Pre-training Objective (2): Sentence Ordering

- Learn relationships between sentences, predict whether Sentence B is actual sentence that proceeds Sentence A, or a random sentence

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
**BERT: Input Representation**

- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
  - Addition to transformer encoder: sentence embedding

---

<table>
<thead>
<tr>
<th>Input</th>
<th>[CLS]</th>
<th>my</th>
<th>dog</th>
<th>is</th>
<th>cute</th>
<th>[SEP]</th>
<th>he</th>
<th>likes</th>
<th>play</th>
<th>#/ing</th>
<th>[SEP]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Token Embeddings</th>
<th>( E_{[CLS]} )</th>
<th>( E_{my} )</th>
<th>( E_{dog} )</th>
<th>( E_{is} )</th>
<th>( E_{cute} )</th>
<th>( E_{[SEP]} )</th>
<th>( E_{he} )</th>
<th>( E_{likes} )</th>
<th>( E_{play} )</th>
<th>( E_{#/ing} )</th>
<th>( E_{[SEP]} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Segment Embeddings</th>
<th>( E_A )</th>
<th>( E_A )</th>
<th>( E_A )</th>
<th>( E_A )</th>
<th>( E_A )</th>
<th>( E_A )</th>
<th>( E_B )</th>
<th>( E_B )</th>
<th>( E_B )</th>
<th>( E_B )</th>
<th>( E_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Position Embeddings</th>
<th>( E_0 )</th>
<th>( E_1 )</th>
<th>( E_2 )</th>
<th>( E_3 )</th>
<th>( E_4 )</th>
<th>( E_5 )</th>
<th>( E_6 )</th>
<th>( E_7 )</th>
<th>( E_8 )</th>
<th>( E_9 )</th>
<th>( E_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

---

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al., 2018]
Training

● Trains model on unlabeled data over different pre-training tasks (self-supervised learning)
● **Data:** Wikipedia (2.5B words) + BookCorpus (800M words)
● **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 TPUs for 4 days

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
BERT in Practice

**TensorFlow:** [https://github.com/google-research/bert](https://github.com/google-research/bert)

**PyTorch:** [https://github.com/huggingface/transformers](https://github.com/huggingface/transformers)
Fine-tuning BERT

“Ideation once, finetune many times.”

- **Idea:** Make pre-trained model **usable** in **downstream tasks**
- **Initialized** with pre-trained model parameters
- **Fine-tune** model parameters using labeled data from downstream tasks
An Example Result: SWAG

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 ([CLS])
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*

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al. 2018]
Why did no one think of this before?

- Concretely, why wasn’t contextual pre-training popular before 2018 with ELMo?

- Good results on pre-training is $>1,000 \times$ to $100,000$ more expensive than supervised training.
What Happened After BERT?

- RoBERTa (Liu et al., 2019)
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date

<table>
<thead>
<tr>
<th>Model</th>
<th>data</th>
<th>bsz</th>
<th>steps</th>
<th>SQuAD (v1.1/2.0)</th>
<th>MNLI-m</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>16GB</td>
<td>8K</td>
<td>100K</td>
<td>93.6/87.3</td>
<td>89.0</td>
<td>95.3</td>
</tr>
<tr>
<td>+ additional data (§3.2)</td>
<td>160GB</td>
<td>8K</td>
<td>100K</td>
<td>94.0/87.7</td>
<td>89.3</td>
<td>95.6</td>
</tr>
<tr>
<td>+ pretrain longer</td>
<td>160GB</td>
<td>8K</td>
<td>300K</td>
<td>94.4/88.7</td>
<td>90.0</td>
<td>96.1</td>
</tr>
<tr>
<td>+ pretrain even longer</td>
<td>160GB</td>
<td>8K</td>
<td>500K</td>
<td>94.6/89.4</td>
<td>90.2</td>
<td>96.4</td>
</tr>
<tr>
<td>BERT\textsubscript{LARGE}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>13GB</td>
<td>256</td>
<td>1M</td>
<td>90.9/81.8</td>
<td>86.6</td>
<td>93.7</td>
</tr>
</tbody>
</table>
What Happened After BERT?

- **RoBERTa** (Liu et al., 2019)
  - Drops the next sentence prediction loss!
  - Trained on 10x data (the original BERT was actually under-trained)
  - Much stronger performance than BERT (e.g., 94.6 vs 90.9 on SQuAD)
  - Still one of the most popular models to date

- **ALBERT** (Lan et al., 2020)
  - Increasing model sizes by sharing model parameters across layers
  - Less storage, much stronger performance but runs slower.

- **ELECTRA** (Clark et al., 2020)
  - Two models generator and discriminator
  - It provides a more efficient training method
What Happened After BERT?

- Models that handle long contexts (512 tokens)
  - Longformer, Big Bird, ...

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

- BERT extended to different domains
  - SciBERT, BioBERT, FinBERT, ClinicalBERT, ...

- Making BERT smaller to use
  - DistillBERT, TinyBERT, ...
Text generation using BERT

**BERT has a Mouth, and It Must Speak:**
BERT as a Markov Random Field Language Model

Alex Wang  
New York University  
alexwang@nyu.edu

Kyunghyun Cho  
New York University  
Facebook AI Research  
CIFAR Azrieli Global Scholar  
kyunghyun.cho@nyu.edu

**Mask-Predict: Parallel Decoding of Conditional Masked Language Models**

Marjan Ghazvininejad*  
Omer Levy*  
Yinhan Liu*  
Luke Zettlemoyer

Facebook AI Research  
Seattle, WA

---

Exposing the Implicit Energy Networks behind Masked Language Models via Metropolis--Hastings

Kartik Goyal, Chris Dyer, Taylor Berg-Kirkpatrick

---

Leveraging Pre-trained Checkpoints for Sequence Generation Tasks

Sascha Rothe, Shashi Narayan, Aliaksei Severyn

---

<table>
<thead>
<tr>
<th>src</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Der Abzug der franzischen Kampftruppen wurde am 20. November abgeschlossen.</td>
<td>The departure of the French combat troops was completed on 20 November.</td>
</tr>
<tr>
<td>completed</td>
<td>completed on 20 November.</td>
</tr>
<tr>
<td>completed</td>
<td>completed on November 20th.</td>
</tr>
</tbody>
</table>

**t = 0** The departure of the French combat troops was completed on 20 November.

**t = 1** The departure of the French combat troops was completed on 20 November.

**t = 2** The withdrawal of French combat troops was completed on November 20th.
Summary Thus Far

- BERT and the family
- An encoder; Transformer-based networks trained on massive piles of data.
- Incredible for learning contextualized embeddings of words
- It’s very useful to pre-train a large unsupervised/self-supervised LM then fine-tune on your particular task (replace the top layer, so that it can work)
- However, they were not designed to generate text.