Pre-training + Fine-tuning Paradigm

LING 575K Deep Learning for NLP
Shane Steinert-Threlkeld
May 5 2021
Announcements

● Regularization and training “speed”

● Sample next character: return a [batch_size] numpy array

● Schedule update:
  ● Multilingual guest lecture moved to May 26 [former overflow day]
  ● May 17: AMA / student question / discussion day
  ● Stay tuned for more info / anonymous question submission form
Note on Transformer Architecture

Do Transformer Modifications Transfer Across Implementations and Applications?

Sharan Narang* Hyung Won Chung Yi Tay William Fedus Thibault Fevry† Michael Matena† Karishma Malkan† Noah Fiedel Noam Shazeer Zhizhong Lan† Yanqi Zhou Wei Li Nan Ding Jake Marcus Adam Roberts Colin Raffel

Google Research

Abstract

The research community has proposed copious modifications to the Transformer architecture since it was introduced over three years ago, relatively few of which have seen widespread adoption. In this paper, we comprehensively evaluate many of these modifications in a shared experimental setting that covers most of the common uses of the Transformer in natural language processing. Surprisingly, we find that most modifications do not meaningfully improve performance. Furthermore, most of the Transformer (and, more generally, the Transformer) will yield equal-or-better performance on any task that the pipeline is applicable to. For example, residual connections in convolutional networks (He et al., 2016) are designed to ideally improve performance on any task where these models are applicable (image classification, semantic segmentation, etc.). In practice, when proposing a new improvement, it is impossible to test it on every applicable downstream task, so researchers must select a few representative tasks to evaluate it on. However, the proposals that are ultimately adopted by the research community and practitioners tend to be those that reliably improve performance across a wide variety of tasks “in
Today’s Plan

● Transfer learning in general
● Language model pre-training: initial steps
● Transformer-based pre-training
  ● Encoder only
  ● Decoder only
  ● Encoder-Decoder
● [Some] limitations [more later in course]
Transfer Learning
Standard Learning

Task 1 outputs

Task 1 inputs
Standard Learning

Task 1 inputs

Task 1 outputs

Task 2 inputs

Task 2 outputs
Standard Learning

Task 1 outputs

Task 2 outputs

Task 3 outputs

Task 1 inputs

Task 2 inputs

Task 3 inputs
Standard Learning

Task 1 outputs

Task 2 outputs

Task 3 outputs

Task 4 outputs

Task 1 inputs

Task 2 inputs

Task 3 inputs

Task 4 inputs
Standard Learning

- New task = new model
- Expensive!
  - Training time
  - Storage space
  - Data availability
    - Can be impossible in low-data regimes
Transfer Learning

“pre-training” task inputs

“pre-training” task outputs
Transfer Learning

“pre-training” task outputs
Transfer Learning

“pre-training” task outputs

Task 1 inputs
Transfer Learning

Task 1 inputs
Transfer Learning

Task 1 outputs

Task 1 inputs
Transfer Learning
Transfer Learning

Task 1 outputs

Task 2 inputs
Transfer Learning

Task 1 outputs

Task 2 inputs

Task 2 outputs
Transfer Learning

Task 1 outputs

Task 2 outputs
Transfer Learning

Task 1 outputs ➔ Task 2 outputs ➔ Task 3 inputs
Transfer Learning

Task 1 outputs → Task 2 outputs → Task 3 inputs

Task 2 outputs → Task 3 outputs
Transfer Learning

Task 1 outputs → Task 2 outputs → Task 3 outputs
Transfer Learning

Task 1 outputs → Pre-trained model, either:
- General feature extractor
- Fine-tuned on tasks

Task 2 outputs

Task 3 outputs
Pre-training + Fine-tuning

- Step 1: *pre-train* a model on a “general” task
  - Goal: produce general-purpose representations of the input, that will be useful when “transferred” to a more specific task.

- Step 2: *fine-tune* that model on the main task
  - Replace the “head” of the model with some task-specific layers
  - Run supervised training with the resulting model
Transfer Learning in Computer Vision

“We use features extracted from the OverFeat network as a generic image representation to tackle the diverse range of recognition tasks of object image classification, scene recognition, fine grained recognition, attribute detection and image retrieval applied to a diverse set of datasets. We selected these tasks and datasets as they gradually move further away from the original task and data the OverFeat network was trained to solve [cf. ImageNet]. Astonishingly, we report consistent superior results compared to the highly tuned state-of-the-art systems in all the visual classification tasks on various datasets”
### Current Benchmarks

**Leaderboard Version: 2.0**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Model</th>
<th>URL</th>
<th>Score</th>
<th>BGLS</th>
<th>CBR</th>
<th>COPA</th>
<th>MultiRL</th>
<th>ReCoRD</th>
<th>RTE</th>
<th>WRC</th>
<th>WSC</th>
<th>All-g</th>
<th>All-i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zhirong Wang</td>
<td>T5 + Momo, Single Model (Momo Team - Google Brain)</td>
<td>50.4</td>
<td>91.4</td>
<td>78.7</td>
<td>55.097.6</td>
<td>98.0</td>
<td>88.369.8</td>
<td>94.369.6</td>
<td>93.0</td>
<td>77.9</td>
<td>58.6</td>
<td>39.76</td>
<td>59.1</td>
</tr>
<tr>
<td>2</td>
<td>DeBERTa Team / Microsoft</td>
<td>DeBERTa / Tuning + Rus</td>
<td>58.3</td>
<td>96.4</td>
<td>66.707.6</td>
<td>98.4</td>
<td>88.269.7</td>
<td>94.964.1</td>
<td>93.2</td>
<td>77.6</td>
<td>59.0</td>
<td>39.393.8</td>
<td>68.7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SuperGLUE Human Baselines</td>
<td>SuperGLUE Human Baselines</td>
<td>92.3</td>
<td>71.4</td>
<td>72.908.9</td>
<td>100.0</td>
<td>81.851.9</td>
<td>97.6</td>
<td>89.0</td>
<td>100.0</td>
<td>99.399.7</td>
<td>78.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TS Team - Google</td>
<td>T5</td>
<td>58.3</td>
<td>91.2</td>
<td>71.908.8</td>
<td>94.8</td>
<td>88.163.3</td>
<td>94.163.4</td>
<td>92.5</td>
<td>78.9</td>
<td>59.5</td>
<td>52.781.9</td>
<td>65.5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Huawei Noah's Ark Lab</td>
<td>NICE-Plus</td>
<td>56.7</td>
<td>87.0</td>
<td>77.408.0</td>
<td>93.6</td>
<td>90.851.6</td>
<td>96.188.6</td>
<td>94.1</td>
<td>74.6</td>
<td>81.2</td>
<td>37.174.4</td>
<td>58.9</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Alibaba PANBEOU</td>
<td>PM Albert</td>
<td>56.1</td>
<td>86.1</td>
<td>82.456.4</td>
<td>91.0</td>
<td>84.854.7</td>
<td>84.854.3</td>
<td>68.8</td>
<td>74.1</td>
<td>50.2</td>
<td>28.399.2</td>
<td>72.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Infosys: DAMIR / AI Research</td>
<td>DeBERTa / CIETS</td>
<td>90.4</td>
<td>88.0</td>
<td>80.268.2</td>
<td>91.2</td>
<td>88.458.2</td>
<td>88.899.3</td>
<td>68.9</td>
<td>72.9</td>
<td>89.0</td>
<td>88.891.5</td>
<td>61.8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Talent / Nolos Lab</td>
<td>RubERTa (ensemble)</td>
<td>55.9</td>
<td>86.2</td>
<td>62.556.5</td>
<td>90.8</td>
<td>84.453.4</td>
<td>91.551.0</td>
<td>87.5</td>
<td>74.1</td>
<td>91.8</td>
<td>69.375.6</td>
<td>57.8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Zhupai Technology</td>
<td>RubERTa - adv</td>
<td>86.7</td>
<td>87.1</td>
<td>82.458.5</td>
<td>91.2</td>
<td>85.154.3</td>
<td>91.761.3</td>
<td>88.1</td>
<td>72.1</td>
<td>91.8</td>
<td>51.078.1</td>
<td>58.5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Facebook AI</td>
<td>RubERTa</td>
<td>87.1</td>
<td>87.1</td>
<td>80.555.2</td>
<td>90.8</td>
<td>84.453.5</td>
<td>90.650.0</td>
<td>88.2</td>
<td>69.9</td>
<td>69.0</td>
<td>51.078.1</td>
<td>57.9</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Anas Shabanovov</td>
<td>AIJoba Team, Transformers</td>
<td>62.1</td>
<td>86.1</td>
<td>91.604.8</td>
<td>88.8</td>
<td>92.154.7</td>
<td>82.879.8</td>
<td>68.9</td>
<td>74.1</td>
<td>79.8</td>
<td>10.010.0</td>
<td>103.0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Rahul Raha &amp; Shrikan Meher</td>
<td>ACHAFF (ALBERT) - few-shot</td>
<td>76.0</td>
<td>80.0</td>
<td>60.356.0</td>
<td>95.5</td>
<td>76.255.7</td>
<td>86.166.5</td>
<td>75.0</td>
<td>55.5</td>
<td>66.0</td>
<td>10.050.0</td>
<td>-0.4</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Tinu Sobh</td>
<td>PiET (ALBERT) - Fine-tuned (UIE Examples)</td>
<td>76.5</td>
<td>81.2</td>
<td>79.888.8</td>
<td>90.8</td>
<td>74.101.7</td>
<td>88.888.4</td>
<td>70.8</td>
<td>49.3</td>
<td>88.0</td>
<td>67.857.9</td>
<td>36.2</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Adhik de Wynter</td>
<td>Bert (Alfa AI)</td>
<td>74.1</td>
<td>83.7</td>
<td>81.086.4</td>
<td>88.6</td>
<td>69.754.1</td>
<td>48.840.6</td>
<td>81.2</td>
<td>70.1</td>
<td>55.8</td>
<td>56.181.5</td>
<td>49.8</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>IBM Research AI</td>
<td>Best (!)</td>
<td>73.5</td>
<td>84.6</td>
<td>69.654.0</td>
<td>73.8</td>
<td>73.205.5</td>
<td>74.674.0</td>
<td>64.1</td>
<td>68.2</td>
<td>61.0</td>
<td>57.857.3</td>
<td>29.5</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Ban Marn</td>
<td>GPT-3 (few-shot - OpenAI)</td>
<td>71.8</td>
<td>78.4</td>
<td>52.075.8</td>
<td>92.0</td>
<td>75.430.5</td>
<td>91.150.2</td>
<td>69.0</td>
<td>49.0</td>
<td>60.1</td>
<td>55.455.3</td>
<td>21.1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>SuperGLUE Baselines</td>
<td>SuperGLUE ++</td>
<td>56.5</td>
<td>70.0</td>
<td>64.506.4</td>
<td>72.6</td>
<td>70.504.1</td>
<td>72.571.3</td>
<td>79.6</td>
<td>69.6</td>
<td>64.4</td>
<td>48.457.4</td>
<td>38.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>BEST</strong></td>
<td><strong>Best</strong></td>
<td>68.0</td>
<td>77.4</td>
<td>75.765.8</td>
<td>70.6</td>
<td>70.504.1</td>
<td>72.571.3</td>
<td>71.7</td>
<td>68.6</td>
<td>64.4</td>
<td>57.851.7</td>
<td>22.0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Most Popular Class</td>
<td>Best</td>
<td>47.1</td>
<td>62.5</td>
<td>21.748.4</td>
<td>50.0</td>
<td>61.153.3</td>
<td>35.452.6</td>
<td>50.3</td>
<td>50.0</td>
<td>65.1</td>
<td>100.095.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CB512</td>
<td>CB512</td>
<td>44.5</td>
<td>62.2</td>
<td>49.071.2</td>
<td>51.6</td>
<td>6.056.5</td>
<td>14.913.6</td>
<td>49.7</td>
<td>52.1</td>
<td>65.1</td>
<td>100.095.0</td>
<td>-0.4</td>
<td></td>
</tr>
</tbody>
</table>
Language Model Pre-training
Where to transfer from?
Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations
Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / *transferable* representations

- Possibilities:
Where to transfer *from*?

- Goal: find a linguistic task that will build general-purpose / transferable representations

- Possibilities:
  - Constituency or dependency parsing
Where to transfer from?

- Goal: find a linguistic task that will build general-purpose / transferable representations

- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
Where to transfer from?

- Goal: find a linguistic task that will build general-purpose / transferable representations

- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
Where to transfer from?

● Goal: find a linguistic task that will build general-purpose / transferable representations

● Possibilities:
  ● Constituency or dependency parsing
  ● Semantic parsing
  ● Machine translation
  ● QA
Where to transfer \textit{from}?

- Goal: find a linguistic task that will build general-purpose / \textit{transferable} representations

- Possibilities:
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
  - QA
  - ...
Where to transfer *from*?

- **Goal:** find a linguistic task that will build general-purpose / *transferable* representations

- **Possibilities:**
  - Constituency or dependency parsing
  - Semantic parsing
  - Machine translation
  - QA
  - ...

- **Scalability issue:** all require expensive annotation
Language Modeling
Language Modeling

● A good language model should produce good general-purpose and transferable representations
Language Modeling

● A good language model should produce good *general-purpose* and *transferable* representations

● Linguistic knowledge:
  ● The bicycles, even though old, were in good shape because ____ …
  ● The bicycle, even though old, was in good shape because ____ …
Language Modeling

- A good language model should produce good general-purpose and transferable representations

- Linguistic knowledge:
  - The bicycles, even though old, were in good shape because ____ ...
  - The bicycle, even though old, was in good shape because ____ ...

- World knowledge:
  - The University of Washington was founded in _____
  - Seattle had a huge population boom as a launching point for expeditions to _____
Data for LM is cheap
Data for LM is cheap
Data for LM is cheap
Language Model Pre-training

- A currently powerful paradigm for training models for NLP tasks:
  - *Pre-train* a large language model on a large amount of raw text
  - *Fine-tune* a small model on top of the LM for the task you care about
  - [or use the LM as a general feature extractor]
Universal Language Model Fine-tuning for Text Classification (ACL ’18)
## ULMFiT

<table>
<thead>
<tr>
<th>Model</th>
<th>Test</th>
<th>Model</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoVe (McCann et al., 2017)</td>
<td>8.2</td>
<td>CoVe (McCann et al., 2017)</td>
<td>4.2</td>
</tr>
<tr>
<td>oh-LSTM (Johnson and Zhang, 2016)</td>
<td>5.9</td>
<td>TBCNN (Mou et al., 2015)</td>
<td>4.0</td>
</tr>
<tr>
<td>Virtual (Miyato et al., 2016)</td>
<td>5.9</td>
<td>LSTM-CNN (Zhou et al., 2016)</td>
<td>3.9</td>
</tr>
<tr>
<td>ULMFiT (ours)</td>
<td><strong>4.6</strong></td>
<td>ULMFiT (ours)</td>
<td><strong>3.6</strong></td>
</tr>
</tbody>
</table>
ULMFiT
Deep Contextualized Word Representations

Deep Contextualized Word Representations

Peters et. al (2018)

- NAACL 2018 Best Paper Award
Deep Contextualized Word Representations

Peters et. al (2018)

- NAACL 2018 Best Paper Award
- Embeddings from Language Models (ELMo)
  - [aka the OG NLP Muppet]
ELMo

Deep contextualized word representations

Matthew E. Peters†, Mark Neumann†, Mohit Iyyer†, Matt Gardner†,
{matthewp, markn, mohiti, mattg}@allenai.org

Christopher Clark*, Kenton Lee*, Luke Zettlemoyer†*
{csquared, kentonl, lsz}@cs.washington.edu

†Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pretrained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised
ELMo

Deep contextualized word representations
Matthew E. Peters¹, Mark Neumann¹, Mohit Iyyer¹, Matt Gardner¹, Christopher Clark*, Kenton Lee*, Luke Zettlemoyer¹*
{matthewp, markn, mohiti, mattg}@allenai.org
{csquared, kentonl, lsz}@cs.washington.edu

¹Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

We introduce a new type of deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP problems, including question answering, textual entailment and sentiment analysis. We also present an analysis showing that exposing the deep internals of the pre-trained network is crucial, allowing downstream models to mix different types of semi-supervision signals.

language model (LM) objective on a large text corpus. For this reason, we call them ELMo (Embeddings from Language Models) representations. Unlike previous approaches for learning contextualized word vectors (Peters et al., 2017; McCann et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Combining the internal states in this manner allows for very rich word representations. Using intrinsic evaluations, we show that the higher-level LSTM states capture context-dependent aspects of word meaning (e.g., they can be used without modification to perform well on supervised
ELMo Model

Source: BERT paper
ELMo Model

Source: BERT paper
ELMo Model

4096-d hidden state
512d projection

residual connection

Source: BERT paper
ELMo Model

Source: BERT paper
ELMo Model

4096-d hidden state
512d projection

residual connection

char CNN

Helps with rare / new words (no OOV)

Source: BERT paper
ELMo Training

● 10 epochs on 1B Word Benchmark

● NB: not SOTA perplexity even at time of publishing
  ● See “Exploring the Limits of Language Modeling” paper

● Regularization:
  ● Dropout
  ● L2 norm
Deep Contextualized Word Representations
Peters et. al (2018)

- Used in place of other embeddings on multiple tasks:

SQuAD = Stanford Question Answering Dataset
SNLI = Stanford Natural Language Inference Corpus
SST-5 = Stanford Sentiment Treebank

*Kitaev and Klein, ACL 2018  (see also Joshi et al., ACL 2018)
Global vs. Contextual Word Vectors

● Global vectors: one vector per word-type
  ● E.g. word2vec, GloVe
  ● No difference between e.g. “play” as a verb, noun, or its different senses

● Contextual vectors: one vector per word-occurrence
  ● “We saw a really great play last week.”
  ● “Do you want to play basketball tomorrow?”
  ● Each occurrence gets its own vector representation.
Deep Contextualized Word Representations

Peters et. al (2018)

- Comparison to GloVe:

<table>
<thead>
<tr>
<th>Source</th>
<th>Nearest Neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe play</td>
<td>playing, game, games, played, players, plays, player, Play, football, multiplayer</td>
</tr>
<tr>
<td>Chico Ruiz made a spectacular play on Alusik's grounder…</td>
<td>Kieffer, the only junior in the group, was commended for his ability to hit in the clutch, as well as his all-round excellent play.</td>
</tr>
<tr>
<td>Olivia De Havilland signed to do a Broadway play for Garson…</td>
<td>…they were actors who had been handed fat roles in a successful play, and had talent enough to fill the roles competently, with nice understatement.</td>
</tr>
</tbody>
</table>
Shallow vs Deep Pre-training

Model for task

Global embedding
Raw tokens

Contextual embedding (pre-trained)
Raw tokens

Model for task
Pre-trained Transformers: Encoder-only
BERT: Bidirectional Encoder Representations from Transformers

Devlin et al NAACL 2019
Overview

- Encoder Representations from Transformers: ✓
- Bidirectional: .........?
  - BiLSTM (ELMo): left-to-right and right-to-left
  - Self-attention: every token can see every other
- How do you treat the encoder as an LM (as computing $P(w_t | w_{t-1}, w_{t-2}, \ldots, w_1)$)?
  - Don’t: modify the task
Masked Language Modeling

- Language modeling: next word prediction
- **Masked** Language Modeling (a.k.a. cloze task): fill-in-the-blank
  - Nancy Pelosi sent the articles of ____ to the Senate.
  - Seattle ____ some snow, so UW was delayed due to ____ roads.
- I.e. $P(w_t | w_{t+k}, w_{t+(k-1)}, \ldots, w_{t+1}, w_{t-1}, \ldots, w_{t-(m+1)}, w_{t-m})$
  - (very similar to CBOW: continuous bag of words from word2vec)
- Auxiliary training task: next sentence prediction.
  - Given sentences A and B, binary classification: did B follow A in the corpus or not?
Schematically
Some details
Some details

- **BASE model:**
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M
Some details

- **BASE model:**
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M

- **LARGE model:**
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M
Some details

- **BASE model:**
  - 12 Transformer Blocks
  - Hidden vector size: 768
  - Attention heads / layer: 12
  - Total parameters: 110M

- **LARGE model:**
  - 24 Transformer Blocks
  - Hidden vector size: 1024
  - Attention heads / layer: 16
  - Total parameters: 340M

---

this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented...
Some details

● BASE model:
  ● 12 Transformer Blocks
  ● Hidden vector size: 768
  ● Attention heads / layer: 12
  ● Total parameters: 110M

● LARGE model:
  ● 24 Transformer Blocks
  ● Hidden vector size: 1024
  ● Attention heads / layer: 16
  ● Total parameters: 340M

this is the first work to demonstrate convincingly that scaling to extreme model sizes also leads to large improvements on very small scale tasks, provided that the model has been sufficiently pre-trained. Peters et al. (2018b) presented
# Input Representation

<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># of</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>$E_{# of}$</th>
<th>$E_{[SEP]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment Embeddings</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_A$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
<td>$E_B$</td>
</tr>
<tr>
<td>Position Embeddings</td>
<td>$E_0$</td>
<td>$E_1$</td>
<td>$E_2$</td>
<td>$E_3$</td>
<td>$E_4$</td>
<td>$E_5$</td>
<td>$E_6$</td>
<td>$E_7$</td>
<td>$E_8$</td>
<td>$E_9$</td>
<td>$E_{10}$</td>
</tr>
</tbody>
</table>
Input Representation

- [CLS], [SEP]: special tokens
[CLS], [SEP]: special tokens

Segment: is this a token from sentence A or B?
Input Representation

- **[CLS], [SEP]:** special tokens
- **Segment:** is this a token from sentence A or B?
- **Position embeddings:** provide position in sequence (*learned* in this case, not fixed)
Input Representation

- [CLS], [SEP]: special tokens
- Segment: is this a token from sentence A or B?
- Position embeddings: provide position in sequence (*learned* in this case, not fixed)
Training Details

- BooksCorpus (800M words) + Wikipedia (2.5B)
- Masking the input text. 15% of all tokens are chosen. Then:
  - 80% of the time: replaced by designated ‘[MASK]’ token
  - 10% of the time: replaced by random token
  - 10% of the time: unchanged
- Loss is cross-entropy of the prediction at the masked positions.
- Max seq length: 128 tokens for first 90%, 512 tokens for final 10%
- 1M training steps, batch size 256 = 4 days on 4 or 16 TPUs
## Initial 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>79.8</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>87.4</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.1</td>
</tr>
<tr>
<td>BERT\textsc{base}</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.5</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\textsc{large}</td>
<td>86.7/85.9</td>
<td>72.1</td>
<td>92.7</td>
<td>94.9</td>
<td>60.5</td>
<td>86.5</td>
<td>89.3</td>
<td>70.1</td>
<td>82.1</td>
</tr>
</tbody>
</table>
### Ablations

- Not a given (depth doesn’t help ELMo); possibly a difference between fine-tuning vs. feature extraction

<table>
<thead>
<tr>
<th>Hyperparams</th>
<th>Dev Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNLI-m</td>
</tr>
<tr>
<td>#L #H #A LM (ppl)</td>
<td></td>
</tr>
<tr>
<td>3  768  12  5.84</td>
<td>77.9</td>
</tr>
<tr>
<td>6  768  3   5.24</td>
<td>80.6</td>
</tr>
<tr>
<td>6  768  12  4.68</td>
<td>81.9</td>
</tr>
<tr>
<td>12 768  12  3.99</td>
<td>84.4</td>
</tr>
<tr>
<td>12 1024 16  3.54</td>
<td>85.7</td>
</tr>
<tr>
<td>24 1024 16  3.23</td>
<td>86.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MNLI-m (Acc)</th>
<th>QNLI (Acc)</th>
<th>MRPC (Acc)</th>
<th>SST-2 (Acc)</th>
<th>SQuAD (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_BASE</td>
<td>84.4</td>
<td>88.4</td>
<td>86.7</td>
<td>92.7</td>
<td>88.5</td>
</tr>
<tr>
<td>No NSP</td>
<td>83.9</td>
<td>84.9</td>
<td>86.5</td>
<td>92.6</td>
<td>87.9</td>
</tr>
<tr>
<td>LTR &amp; No NSP</td>
<td>82.1</td>
<td>84.3</td>
<td>77.5</td>
<td>92.1</td>
<td>77.8</td>
</tr>
<tr>
<td>+ BiLSTM</td>
<td>82.1</td>
<td>84.1</td>
<td>75.7</td>
<td>91.6</td>
<td>84.9</td>
</tr>
</tbody>
</table>
Other Prominent Encoders

- **RoBERTa**: robustly optimized BERT approach
  - BERT was very *under-trained*: give it more data, train it longer [keep model the same otherwise]
  - Good default encoder

- **ELECTRA**: replace Masked Language Modeling with “replaced token detection”, trains just as well with much less data

- **SpanBERT**: mask out entire *spans* instead of single tokens
Limitation of Encoders

● No left-to-right modeling assumption
● Good for NLU (understanding/comprehension) tasks
● Does not straightforwardly *generate* text
Pre-trained Transformers: Decoder-only
GPT(2)

- Generative Pre-training
  - Radford et al 2018; 2019 (GPT2); Brown et al 2020 (GPT3)

- Uses Transformer *decoder* instead of *encoder*
  - “Self”-attention: masked so that only can attend to *previous* tokens

- Pure LM training objective
- Can be used for text generation

- GPT: same params as BERT-BASE; GPT2 much bigger; GPT3 muuuuuch bigger (175B params)

- Training data: crawled from outbound Reddit links w/ >3 karma, not public
Comparison

Source: BERT paper
Samples from GPT2 (Cherry-picked)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.
Recycling is good for the world.

NO! YOU COULD NOT BE MORE WRONG!!

Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources. And THAT is why we need to get back to basics and get back to basics in our recycling efforts. One of the best ways to start is to look at the process of creating a paper product. When you make a paper product, it is basically a long chain of materials. Everything from the raw materials (wood, cardboard, paper, etc.), to the reagents (dyes, solvents, etc.) to the printing equipment (chemicals, glue, paper, ink, etc.), to the packaging, to the packaging materials (mercury, chemicals, etc.) to the processing equipment (heating, cooling, etc.), to the packaging materials, to the packaging materials that are shipped overseas and to the packaging materials that are used in the United States. Each step along the way creates tons of waste that we constantly have to clean up. The process
GPT2 Release Controversy

- Worry: it’s so good, it could be used by malicious actors to make deep-fakes for text (incl. fake news, character assassination, …)
- So the model was not initially released (but has been now)
- Some cynical interpretations

https://openai.com/blog/better-language-models/
Prompt: Studying computational linguistics is a good idea. There are many interesting problems.

However, there are many bad problems. The research into computational linguistics has not helped very much in the development of better solutions. One problem with computer science and machine learning is that, until recently, the problems they were supposed to solve were insufficiently well studied. Most of the people who are doing research in computer science (at least in my profession) have no understanding of how to design a particularly natural problem, for example. In the case of the reasoning problem above, most of the people I know who work on computer science or machine learning have no understanding of how to make computer programs work in ways which do not have unavoidable adverse consequences. In the best cases, computational linguistics was little help.
Prompt: I didn't ride my bike to campus today because of the snowy weather.

There is a lot of construction and I wanted to have a few hours of skiing. All that snow is a real pain in the ass. Good thing there was an LRT planned for that area. See you guys on campus in a few days!

Photo credit: Tapto@archivosos.gr – August 26, 2016: Windy Weather on the Mountain (photo credit: Diana Prat)
From Diana Prat, via tapto@archivosos.gr:

Ever since the camp that I organized to protest the modernisation of the Barranco, my beautiful alma mater, you know, going in for a press conference, I was asking why the Italian government
- Same approach: pure Transformer decoder trained on LM
- Scale: 175B params
- Data size: ~500 billion tokens, majority from filtered Common Crawl
- Few-shot “fine-tuning” paradigm:
  - Prompt with a few examples, ask to continue
  - No parameter updates
# GPT3 Few-Shot Results

<table>
<thead>
<tr>
<th></th>
<th>SuperGLUE Average</th>
<th>BoolQ Accuracy</th>
<th>CB Accuracy</th>
<th>CB F1</th>
<th>COPA Accuracy</th>
<th>RTE Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuned SOTA</td>
<td>89.0</td>
<td>91.0</td>
<td>96.9</td>
<td>93.9</td>
<td>94.8</td>
<td>92.5</td>
</tr>
<tr>
<td>Fine-tuned BERT-Large</td>
<td>69.0</td>
<td>77.4</td>
<td>83.6</td>
<td>75.7</td>
<td>70.6</td>
<td>71.7</td>
</tr>
<tr>
<td>GPT-3 Few-Shot</td>
<td>71.8</td>
<td>76.4</td>
<td>75.6</td>
<td>52.0</td>
<td>92.0</td>
<td>69.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>WiC Accuracy</th>
<th>WSC Accuracy</th>
<th>MultiRC Accuracy</th>
<th>MultiRC F1a</th>
<th>ReCoRD Accuracy</th>
<th>ReCoRD F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-tuned SOTA</td>
<td>76.1</td>
<td>93.8</td>
<td>62.3</td>
<td>88.2</td>
<td>92.5</td>
<td>93.3</td>
</tr>
<tr>
<td>Fine-tuned BERT-Large</td>
<td>69.6</td>
<td>64.6</td>
<td>24.1</td>
<td>70.0</td>
<td>71.3</td>
<td>72.0</td>
</tr>
<tr>
<td>GPT-3 Few-Shot</td>
<td>49.4</td>
<td>80.1</td>
<td>30.5</td>
<td>75.4</td>
<td>90.2</td>
<td>91.1</td>
</tr>
</tbody>
</table>

\( k=32 \)
Pretrained Transformers: Encoder-Decoder
BART

- Full Transformer, i.e. encoder-decoder transducer
- Many composable transformations of raw text, presented to encoder
- Goal of decoder is to reconstruct the original text

- Good for both discrimination and generation
High-level Overview

(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However, words can only condition on leftward context, so it cannot learn bidirectional interactions.

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitrary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.
### Comparison of Pre-training Objectives

<table>
<thead>
<tr>
<th>Model</th>
<th>SQuAD 1.1 F1</th>
<th>MNLI Acc</th>
<th>ELI5 PPL</th>
<th>XSum PPL</th>
<th>ConvAI2 PPL</th>
<th>CNN/DM PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT Base (Devlin et al., 2019)</td>
<td>88.5</td>
<td><strong>84.3</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Masked Language Model</td>
<td>90.0</td>
<td>83.5</td>
<td>24.77</td>
<td>7.87</td>
<td>12.59</td>
<td>7.06</td>
</tr>
<tr>
<td>Masked Seq2seq</td>
<td>87.0</td>
<td>82.1</td>
<td>23.40</td>
<td>6.80</td>
<td>11.43</td>
<td>6.19</td>
</tr>
<tr>
<td>Language Model</td>
<td>76.7</td>
<td>80.1</td>
<td><strong>21.40</strong></td>
<td>7.00</td>
<td>11.51</td>
<td>6.56</td>
</tr>
<tr>
<td>Permutated Language Model</td>
<td>89.1</td>
<td>83.7</td>
<td>24.03</td>
<td>7.69</td>
<td>12.23</td>
<td>6.96</td>
</tr>
<tr>
<td>Multitask Masked Language Model</td>
<td>89.2</td>
<td>82.4</td>
<td>23.73</td>
<td>7.50</td>
<td>12.39</td>
<td>6.74</td>
</tr>
<tr>
<td>BART Base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ Token Masking</td>
<td>90.4</td>
<td>84.1</td>
<td>25.05</td>
<td>7.08</td>
<td>11.73</td>
<td>6.10</td>
</tr>
<tr>
<td>w/ Token Deletion</td>
<td>90.4</td>
<td>84.1</td>
<td>24.61</td>
<td>6.90</td>
<td>11.46</td>
<td>5.87</td>
</tr>
<tr>
<td>w/ Text Infilling</td>
<td><strong>90.8</strong></td>
<td>84.0</td>
<td>24.26</td>
<td><strong>6.61</strong></td>
<td><strong>11.05</strong></td>
<td>5.83</td>
</tr>
<tr>
<td>w/ Document Rotation</td>
<td>77.2</td>
<td>75.3</td>
<td>53.69</td>
<td>17.14</td>
<td>19.87</td>
<td>10.59</td>
</tr>
<tr>
<td>w/ Sentence Shuffling</td>
<td>85.4</td>
<td>81.5</td>
<td>41.87</td>
<td>10.93</td>
<td>16.67</td>
<td>7.89</td>
</tr>
<tr>
<td>w/ Text Infilling + Sentence Shuffling</td>
<td><strong>90.8</strong></td>
<td>83.8</td>
<td>24.17</td>
<td>6.62</td>
<td>11.12</td>
<td><strong>5.41</strong></td>
</tr>
</tbody>
</table>
Advantages of Encoder-Decoder Models

● “Best of both worlds”
● On a par with RoBERTa on NLU / discrimination tasks
● State-of-the-art on many generation tasks (e.g. summarization)

● Others:
  ● MASS
  ● T5 [also uses labeled data]

<table>
<thead>
<tr>
<th>Source Document (abbreviated)</th>
<th>BART Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>The researchers examined three types of coral in reefs off the coast of Fiji. ... The researchers found when fish were plentiful, they would eat algae and seaweed off the corals, which appeared to leave them more resistant to the bacterium Vibrio corallilyticus, a bacterium associated with bleaching. The researchers suggested the algae, like warming temperatures, might render the corals’ chemical defenses less effective, and the fish were protecting the coral by removing the algae.</td>
<td></td>
</tr>
<tr>
<td>Fisheries off the coast of Fiji are protecting coral reefs from the effects of global warming, according to a study in the journal Science.</td>
<td></td>
</tr>
</tbody>
</table>

Sacoolas, who has immunity as a diplomat’s wife, was involved in a traffic collision ... Prime Minister Johnson was questioned about the case while speaking to the press at a hospital in Watford. He said, “I hope that Anne Sacoolas will come back ... if we can’t resolve it then of course I will be raising it myself personally with the White House.”

Boris Johnson has said he will raise the issue of US diplomat Anne Sacoolas’ diplomatic immunity with the White House.
Limitations of Pre-training + Fine-tuning
State of the Field

● Manning 2017: “The BiLSTM Hegemony”

● Right now: “The pre-trained Transformer Hegemony”
  ● By default: fine-tune a large pre-trained Transformer on the task you care about
  ● Will often yield the best results
  ● Beware: often not significantly better than very simple baselines (SVM, etc)
Some Reasons to Pause

- Leaderboard chasing (via larger models and more data) funnels research activity into one specific and limited goal
- Amplifies harmful biases
- Equity costs
- Climate costs
- Data documentation debt
- Does not promote human-like linguistic generalization ([Linzen 2020](Linzen2020) summary)

More from Angelina McMillan-Major on May 19 on [stochastic parrots paper](stochasticparrotspaper)
Transformers

https://huggingface.co/transformers
Overview of the Library

- Access to many variants of many very large LMs (BERT, RoBERTa, XLNET, ALBERT, T5, language-specific models, …) with fairly consistent API
  - Build tokenizer + model from string for name or config
  - Then use just like any PyTorch nn.Module

- Emphasis on ease-of-use
  - E.g. low barrier-to-entry to using the models, including for analysis

- Interoperable with PyTorch or TensorFlow 2.0
import torch
from transformers import BertTokenizer, BertModel, BertForMaskedLM

# OPTIONAL: if you want to have more information on what's happening under the hood, activate the logger as follows
import logging
logging.basicConfig(level=logging.INFO)

# Load pre-trained model tokenizer (vocabulary)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize input
text = '"[CLS] Who was Jim Henson? [SEP] Jim Henson was a puppeteer [SEP]"
tokenized_text = tokenizer.tokenize(text)

# Mask a token that we will try to predict back with `BertForMaskedLM`
masked_index = 8
tokenized_text[masked_index] = '"[MASK]"
assert tokenized_text == ['"[CLS]", '"who", '"was", '"jim", '"henson", '"?", '"[SEP]", '"jim", '"[MASK]", '"was", '"a", '"puppet", '"#eer", '"[SEP]"

# Convert token to vocabulary indices
indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)

# Define sentence A and B indices associated to 1st and 2nd sentences (see paper)
segments_ids = [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

# Convert inputs to PyTorch tensors
tokens_tensor = torch.tensor([indexed_tokens])
segments_tensors = torch.tensor([segments_ids])

See http://juditacs.github.io/2019/02/19/bert-tokenization-stats.html (h/t Naomi Shapiro)
Example: Forward Pass

```python
# Load pre-trained model (weights)
model = BertModel.from_pretrained('bert-base-uncased')

# Set the model in evaluation mode to deactivate the DropOut modules
# This is IMPORTANT to have reproducible results during evaluation!
model.eval()

# If you have a GPU, put everything on cuda
tokens_tensor = tokens_tensor.to('cuda')
segments_tensors = segments_tensors.to('cuda')
model.to('cuda')

# Predict hidden states features for each layer
with torch.no_grad():
    # See the models docstrings for the detail of the inputs
    outputs = model(tokens_tensor, token_type_ids=segments_tensors)
    # Transformers models always output tuples.
    # See the models docstrings for the detail of all the outputs
    # In our case, the first element is the hidden state of the last layer of the Bert model
    encoded_layers = outputs[0]
```
More on HuggingFace

- Main library: https://huggingface.co/transformers
- Model repository (w/ search, tags, etc): https://huggingface.co/models
- Datasets: https://huggingface.co/datasets