Improving Language Understanding by Generative Pre-Training
Motivation

- Semi-supervised learning: embeddings
  - Unsupervised learning of word-level or phrase-level stats
    - E.g. Word embeddings, ELMo vectors
  - Supervised training using these word-level features
    - ELMo Example:
      - Question Answering: Add ELMo to modified BiDAF model
      - Textual Entailment: Add ELMo to ESIM sequence model
      - Coreference Resolution: Add ELMo to end-to-end span-based neural model
ELMo: Different Models for Each Task
Generative Pre-Training (GPT)

- **Single Model: Transformers**
  - Make longer-distance connections
  - Faster training

- **Unsupervised pre-training**
  - Similar objective as Word2Vec
  - Predict context words

- **Supervised fine-tuning**
  - Use pre-trained model
  - *Only swap the last layer*

- **Takeaways**
  - Apply one pre-trained model to many tasks
  - BPE Tokens
  - Pre-trained Transformers learn something, even with no supervision
Transformer is more efficient than LSTM because it lends itself to parallelization.
Self-Attention
Self-Attention in Detail

Multiplying $x_1$ by the $W_Q$ weight matrix produces $q_1$, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.
Self-Attention in Detail

Input

Embedding

x₁

q₁

k₁

v₁

Queries

x₂

q₂

k₂

v₂

Keys

Values

Score

14

0.88

Divide by \( \sqrt{d_k} \)

q₁ \cdot k₁ = 112

q₁ \cdot k₂ = 96

12

0.12

Softmax
Transformers

Figure 1: The Transformer - model architecture.
GPT Framework

- Multi-layer Transformer decoder

\[
\begin{align*}
    h_0 &= UW_e + W_p \\
    h_l &= \text{transformer\_block}(h_{l-1}) \forall i \in [1, n]
\end{align*}
\]
Unsupervised Pre-Training

- Similar objective as Word2Vec
- Given tokens \( u \), maximize:

\[
L_1(U) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)
\]

\[
P(u) = \text{softmax}(h_n W_c^T)
\]

- \( k \) is context window size
- \( h_n W_c^T \) is score for each word
- softmax gives probability distribution
Supervised Fine-Tuning

- Keep the pre-trained Transformers
- Replace the final linear layer
  - Replace $W_e$ with $W_y$
- Data inputs $x$, label $y$
- Maximize

$$L_2(C) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

$$P(y|x^1, \ldots, x^m) = \text{softmax}(h^m_lW_y).$$

$$L_3(C) = L_2(C) + \lambda \cdot L_1(C)$$
Framework

- Multi-layer Transformer decoder
  \[ h_0 = UW_e + W_p \]
  \[ h_i = \text{transformer\_block}(h_{i-1}) \quad \forall i \in [1, n] \]

- Unsupervised pre-training
  - Similar objective as Word2Vec
  - Maximize:
    \[ L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \]
    \[ P(u) = \text{softmax}(h_n W^T_e) \]

- Supervised fine-tuning
  - Data inputs x, label y
  - Maximize:
    \[ L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \ldots, x^m) \]
    \[ P(y | x^1, \ldots, x^m) = \text{softmax}(h^m_l W_y) \]
    \[ L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda \times L_1(\mathcal{C}) \]
Task Adaptations

- How to adapt a single architecture to multiple input formats?
Task Overviews

- **Classification (e.g. sentiment analysis)**
  - Given a piece of text, is it positive or negative?
  - Answers: "Yes", "No"
  - Answers: "Very positive", "Positive", "Neutral", "Negative", "Very negative"

- **Entailment**
  - Given a premise $p$ and a hypothesis $h$, does $p$ imply $h$?
  - Answers: "entailment", "contradiction", or "neutral"

- **Similarity**
  - Are two sentences semantically equivalent?
  - Answers: "Yes", "No"

- **Multiple Choice (e.g. Story Cloze)**
  - Given a short story and two sentences, which is the sentence that ends the story?
  - Given a passage and a question, and some multiple-choice answers, which is the answer?
  - Answers: A_1, A_2, ... A_N
Task-Specific Adaptations

- Classification
- Entailment
- Similarity
- Multiple Choice

Special start token
Special delimiter token
Special end token

Pre-trained
Task-specific

Softmax
Experiments

- BooksCorpus for unsupervised training
  - About the same size as 1B Word Benchmark (used for ELMo)
  - **Preserves longer structure**
- Model
  - 12-layer transformer network
- Returns strong results on most tasks, especially question answering and commonsense reasoning
Data

Textual Entailment: ~2-6% improvement

<table>
<thead>
<tr>
<th>Method</th>
<th>MNLI-m</th>
<th>MNLI-mm</th>
<th>SNLI</th>
<th>SciTail</th>
<th>QNLI</th>
<th>RTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESIM + ELMo [44] (5x)</td>
<td>-</td>
<td>-</td>
<td>89.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAFE [58] (5x)</td>
<td>80.2</td>
<td>79.0</td>
<td>89.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Stochastic Answer Network [35] (3x)</td>
<td>80.6</td>
<td>80.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAFE [58]</td>
<td>78.7</td>
<td>77.9</td>
<td>88.5</td>
<td>83.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GenSen [64]</td>
<td>71.4</td>
<td>71.3</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
<td>59.2</td>
</tr>
<tr>
<td>Multi-task BiLSTM + Attn [64]</td>
<td>72.2</td>
<td>72.1</td>
<td>-</td>
<td>-</td>
<td>82.1</td>
<td>61.7</td>
</tr>
<tr>
<td>Finetuned Transformer LM (ours)</td>
<td>82.1</td>
<td>81.4</td>
<td>89.9</td>
<td>88.3</td>
<td>88.1</td>
<td>56.0</td>
</tr>
</tbody>
</table>
## Data

Question answering and story completion: 3-6% improvement

<table>
<thead>
<tr>
<th>Method</th>
<th>Story Cloze</th>
<th>RACE-m</th>
<th>RACE-h</th>
<th>RACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>val-LS-skip [55]</td>
<td>76.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hidden Coherence Model [7]</td>
<td>77.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic Fusion Net [67] (9x)</td>
<td>-</td>
<td>55.6</td>
<td>49.4</td>
<td>51.2</td>
</tr>
<tr>
<td>BiAttention MRU [59] (9x)</td>
<td>-</td>
<td>60.2</td>
<td>50.3</td>
<td>53.3</td>
</tr>
<tr>
<td>Finetuned Transformer LM (ours)</td>
<td><strong>86.5</strong></td>
<td><strong>62.9</strong></td>
<td><strong>57.4</strong></td>
<td><strong>59.0</strong></td>
</tr>
</tbody>
</table>
## Data

Semantic similarity and text classification: wide range

<table>
<thead>
<tr>
<th>Method</th>
<th>Classification</th>
<th></th>
<th></th>
<th>Semantic Similarity</th>
<th></th>
<th>GLUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CoLA (mc)</td>
<td>SST2 (acc)</td>
<td>MRPC (F1)</td>
<td>STSB (pc)</td>
<td>QQP (F1)</td>
</tr>
<tr>
<td>Sparse byte mLSTM [16]</td>
<td>-</td>
<td>93.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TF-KLD [23]</td>
<td>-</td>
<td>-</td>
<td>86.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ECNU (mixed ensemble) [60]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>81.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-task BiLSTM + ELMo + Attn [64]</td>
<td>35.0</td>
<td>90.2</td>
<td>80.2</td>
<td>55.5</td>
<td>66.1</td>
<td>64.8</td>
</tr>
<tr>
<td>Multi-task BiLSTM + ELMo + Attn [64]</td>
<td>18.9</td>
<td>91.6</td>
<td>83.5</td>
<td>72.8</td>
<td>63.3</td>
<td>68.9</td>
</tr>
<tr>
<td>Finetuned Transformer LM (ours)</td>
<td><strong>45.4</strong></td>
<td>91.3</td>
<td>82.3</td>
<td><strong>82.0</strong></td>
<td><strong>70.3</strong></td>
<td><strong>72.8</strong></td>
</tr>
</tbody>
</table>
Takeaways (discussion?)
Takeaways (discussion?)

- **Few new parameters** for each supervised task
  - One linear layer, plus delimiter embedding
- **Transformers**
  - Allow long-term dependencies to be made
  - Faster to train
- **BPE Tokens** (next)
- **Zero-shot Behavior** (next next)
Binary Pair Encoding (BPE) Tokens

- **Drawbacks of regular word tokens**
  - Other forms? (play vs. playing)
  - Compound words (overripe)
  - Large vocab size

- **Begin with a vocabulary**: 'A', 'B', 'C', ...

- **Add to your vocabulary**: Combine common character-pairs
  - 'A' + 'B' → 'AB'

- Also, add an end-of-word symbol *

- **Example**:  
  - \{ 'low', 'lowest', 'newer', 'wider' \} 
  - \{ 'low*', 'lowest*', 'newer*', 'wider*' \}

- Add \{ r*, lo, low, er* \} to vocabulary

- Before: \ l + o + w + e + r + *

- After: \ low + er*
Zero-shot Behavior

- Use heuristics, rather than supervised training
- Use pre-trained model directly
- E.g: Question answering: Pick the answer the generative model assigns the highest probability to, conditioned on the document and question
Analysis: Zero-Shot Behavior

State of Art

Random Guessing
Takeaways (discussion?)

- **Few new parameters** for each supervised task
  - One linear layer, plus delimiter embedding
- **Transformers**
  - Allow long-term dependencies to be made
  - Faster to train
- BPE Tokens (next)
- Zero-shot training (next next)
Analysis: Layer Transfer
Related Work

- **Pre-trained LSTM**
  - (Dai et al. 2015) and (Howard and Ruder 2018)
  - Pre-train LSTM's on sequence autoencoding, then apply to text classification

- **Auxiliary unsupervised objectives**
  - Add an unsupervised goal to your objective
    - E.g. Ask your model to do context-prediction and text classification
    - (Collobert and Weston 2008) and (Rei 2017)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Devlin et al. (Google AI Language)
Previous Work

- Language model pre-training has been used to improve many NLP tasks
  - Elmo (Peters et al., 2018)
  - OpenAI GPT (Radford et al., 2018)
  - ULMFit (Howard and Rudder, 2018)

- Language model pre-training has been used to improve many NLP tasks
  - Feature-based: include pre-trained representations as additional features (e.g., ELMo)
  - Fine-tuning: introduce task-specific parameters and fine-tune the pre-trained parameters
Limitations of Previous Techniques

● **Problem:** Language models only use left context or right context, but language understanding is bidirectional.
● Why are LMs unidirectional?
● Reason: Words can “see themselves” in a bidirectional encoder.
Main Ideas

● Propose a new training objective so that a deep bidirectional transformer can be trained
  ○ The masked language model
  ○ Next Sentence Prediction

● Merits of BERT
  ○ Just fine-tune BERT Model for specific tasks to achieve state-of-the-art performance
  ○ BERT advances the state-of-the-art for eleven NLP tasks
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

- Too little masking: Too expensive to train
- Too much masking: Not enough context
Masked LM

• Problem: Mask token never seen at fine-tuning
• Solution: 15% of the words to predict, but don’t replace with [MASK] 100% of the time.

  Instead:

• 80% of the time, replace with [MASK]
• 10% of the time, replace random word
• 10% of the time, keep same
Next Sentence Prediction

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

- The training loss is the sum of the mean masked Language Model likelihood and the mean next sentence prediction likelihood
- Use 30,000 WordPiece vocabulary on input
- Each token is sum of three embeddings
Model Architecture

Transformer encoder
Model Architecture

Differences in pre-training model architectures: BERT, OpenAI GPT, and ELMo
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
## Comparison of BERT and OpenAI GPT

<table>
<thead>
<tr>
<th>OpenAI GPT</th>
<th>BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained on BooksCorpus (800M)</td>
<td>Trained on BooksCorpus (800M) + Wikipedia (2,500M)</td>
</tr>
<tr>
<td>Use sentence separator ([SEP]) and classifier token ([CLS]) only at fine-tuning time</td>
<td>BERT learns [SEP], [CLS] and sentence A/B embeddings during pre-training</td>
</tr>
<tr>
<td>Trained for 1M steps with a batch-size of 32,000 words</td>
<td>Trained for 1M steps with a batch-size of 128,000 words</td>
</tr>
<tr>
<td>Use the same learning rate of 5e-5 for all fine-tuning experiments</td>
<td>BERT choose a task-specific learning rate which performs the best on the development set</td>
</tr>
</tbody>
</table>
Fine-Tuning Procedure
Fine-Tuning For Specific Tasks

(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
## Results

### 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/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>
<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_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_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
SQuAD 1.1

The training objective is the sum of the log-likelihoods of the correct start and end positions.

- Only new parameters: Start vector and end vector.
- Softmax predictions.

\[ P_i = \frac{e^{S_iT_i}}{\sum_j e^{S_jT_j}} \]
The Situations with Adversarial Generations (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.

The only task-specific parameters is a vector $V \in \mathbb{R}^H$

The probability distribution is the softmax over the four choices

$$P_i = \frac{e^{V \cdot c_i}}{\sum_{j=1}^{4} e^{V \cdot c_i}}$$
### Swag Result

<table>
<thead>
<tr>
<th>System</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESIM+GloVe</td>
<td>51.9</td>
<td>52.7</td>
</tr>
<tr>
<td>ESIM+ELMo</td>
<td>59.1</td>
<td>59.2</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;BASE&lt;/sub&gt;</td>
<td>81.6</td>
<td>-</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt;</td>
<td>86.6</td>
<td>86.3</td>
</tr>
<tr>
<td>Human (expert)†</td>
<td>-</td>
<td>85.0</td>
</tr>
<tr>
<td>Human (5 annotations)†</td>
<td>-</td>
<td>88.0</td>
</tr>
</tbody>
</table>

Table 4: SWAG Dev and Test accuracies. Test results were scored against the hidden labels by the SWAG authors. †Human performance is measured with 100 samples, as reported in the SWAG paper.
Ablation Study

Effect of Pre-training Task

- BERT-Base
- No Next Sent
- Left-to-Right & No Next Sent
- Left-to-Right & No Next Sent + BiLSTM

Accuracy

- MNLI
- QNLI
- MRPC
- SQuAD
Effect of 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.
Effects of Model Size

- Big models help a lot
- Going from 110M -> 340M params helps even on datasets with 3,600 labeled examples
## Effects of Masking Strategy

<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>10%</td>
</tr>
<tr>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>80%</td>
<td>0%</td>
</tr>
<tr>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td>0%</td>
<td>20%</td>
</tr>
<tr>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

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

Advantages of Feature-Based Approach:

- Not all tasks can be represented by a transformer encoder architecture, and therefore require a task-specific model architecture to be added.
- Major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation.
# Feature-Based BERT Results

<table>
<thead>
<tr>
<th>System</th>
<th>Dev F1</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo (Peters et al., 2018a)</td>
<td>95.7</td>
<td>92.2</td>
</tr>
<tr>
<td>CVT (Clark et al., 2018)</td>
<td>-</td>
<td>92.6</td>
</tr>
<tr>
<td>CSE (Akbik et al., 2018)</td>
<td>-</td>
<td>93.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fine-tuning approach</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{BERT}_{\text{LARGE}}$</td>
<td>96.6</td>
<td>92.8</td>
</tr>
<tr>
<td>$\text{BERT}_{\text{BASE}}$</td>
<td>96.4</td>
<td>92.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature-based approach (BERT$_{\text{BASE}}$)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Embeddings</td>
<td>91.0</td>
<td>-</td>
</tr>
<tr>
<td>Second-to-Last Hidden</td>
<td>95.6</td>
<td>-</td>
</tr>
<tr>
<td>Last Hidden</td>
<td>94.9</td>
<td>-</td>
</tr>
<tr>
<td>Weighted Sum Last Four Hidden</td>
<td>95.9</td>
<td>-</td>
</tr>
<tr>
<td>Concat Last Four Hidden</td>
<td>96.1</td>
<td>-</td>
</tr>
<tr>
<td>Weighted Sum All 12 Layers</td>
<td>95.5</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.
Comparison to Computer Vision

- Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer’s outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset.
Conclusions

● Pre-trained language models are increasingly adopted in many NLP tasks
● Major contribution of this paper is to propose a deep bidirectional architecture from Transformer