NLP: Advanced Architectures

Yuanzhi Zhu
SiDNN
Content

- Transformer
- GPT
- BERT
Content

• Transformer

• GPT

• BERT
NLP: Seq2Seq Problems

Seq2Seq Problems: More than word embeddings

**Machine Translation**

天星散落如雪  
The stars in the sky are scattered like snow

**Question Answering**

How are you?  
I am good
NLP: Seq2Seq Model

General Encoder Decoder Architecture

Source Sequence: 天星散落如雪

Target Sequence: The stars are scattered like snow

Encoder -> Decoder
NLP: Seq2Seq Model

RNN Encoder Decoder Architecture

Limitations:
- Long Sequences
- No Parallelization
Self-Attention: Better Info Mixer

Input Sequence $X$: $d_{model}$

$$\text{softmax} \left( \frac{Q \times K^T}{\sqrt{d_k}} \right)$$

Weighted sum of the values
Multi-head Attention ← Multi-channel CNN

All heads are initialized randomly to learn different attentions

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O
\]

where \( \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \)

output dimension = \( \mathcal{R}^{n \times d_{\text{model}}} \)
Seq2Seq Model with Attention

Encoder Decoder Architecture Using Attention

Source Sequence

天星散落如雪

The stars are scattered like snow

Decoder (Masked Attention)

(Attention)

Target Sequence
Mask: In Decoder

**Why?** Prediction should only rely on all the previous tokens
**Mask: In Decoder**

Masked attention

Product of Q*K (scale)

Masking

After Masking

SoftMax

Weight of Attention
Encoder & Decoder in Detail

**Encoder**

- LayerNorm
- +
- Feed Forward
- +
- Multi-Head Self-Attention
- X

**Decoder**

- Add & Norm
- Feed Forward
- Add & Norm
- Encoder-Decoder Attention
- Add & Norm
- Multi-Head Self-Attention
The Output Layer: (Language Generation Model)

- Output from Decoder with size $d_{model}$
- Linear
- Vector with vocabulary size
- Softmax
- Log-Probabilities
- Word with the biggest probability
- Argmax
Attention → Transformer
Self-Attention: Example

https://github.com/jessevig/bertviz
Encoder: Animation

Decoding time step: 1 2 3 4 5 6

OUTPUT

LINEAR + SOFTMAX

ENCODER

DECODER

EMBEDDING WITH TIME SIGNAL

EMBEDDINGS

INPUT: je suis étudiant

http://jalammar.github.io/illustrated-transformer
Decoder: Animation

http://jalammar.github.io/illustrated-transformer
**Transformer: Review**

*Attention is all you need (beyond RNN)*

\( n \): length of sequence \hspace{1cm} \( d \): dimension of word embedding

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity/Layer</th>
<th>Sequential Operation</th>
<th>Maximum Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>( O(n^2d) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Recurrent</td>
<td>( O(nd^2) )</td>
<td>( O(n) )</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>Convolutional</td>
<td>( O(knd) )</td>
<td>( O(1) )</td>
<td>( O(\log_k(n)) )</td>
</tr>
<tr>
<td>Self-Attention (restricted)</td>
<td>( O(rnd) )</td>
<td>( O(1) )</td>
<td>( O(n/r) )</td>
</tr>
</tbody>
</table>
Content

- Transformer
- GPT
- BERT
**Motivation:** Unlabeled data $\rightarrow$ General language model

Multi-layer Transformer **Decoders:** longer-range linguistic structure (Only one multi-head attention sub-layer)

GPT Example: Summarization

Alice’s Adventures in Wonderland by Lewis Carroll

According to O. Henry, the short story is a "flash fiction". It is a brief and vivid representation of a small moment of life. The story is a "flash" because it focuses on a single incident that captures a larger truth.

Summarizing Books with Human Feedback (openai.com)
GPT: Pre-Training

Two stages semi-supervised training

- Unsupervised Language Modelling (pre-training)

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

\[ h_0 = U W_e + W_p \]
\[ h_l = \text{transformer\_block}(h_{l-1}) \forall i \in [1, n] \]
\[ P(u) = \text{softmax}(h_n W_e^T) \]

I am good
\[ \arg\max P(u_i|I, am; \Theta) = \text{good} \]
Two stages semi-supervised training

- Supervised fine-tuning

\[ P(y|x^1, \ldots, x^m) = \text{softmax}(h_l^m W_y). \]

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

Including language modeling as an auxiliary objective

\[ L_3(C) = L_2(C) + \lambda \cdot L_1(C) \]

X: I am good
Y: positive
GPT: Task-Specific Input Transformations

GPT is SOTA: achieves new state-of-the-art results in 9 out of the 12 datasets
GPT: Experiment & Results

Zero-shot Behaviors

![Graph showing relative task performance vs. number of pre-training updates for different tasks and models. The x-axis represents the number of pre-training updates, ranging from $10^3$ to $10^6$. The y-axis represents relative task performance, ranging from 0.0 to 1.0. Different tasks and models are represented by distinct lines and markers. The tasks include sentiment analysis, winograd schema resolution, linguistic acceptability, question answering, Transformer, and LSTM. The performance curves show varying levels of improvement with increasing pre-training updates.]
GPT2 and GPT3

GPT-2
1.5B Parameters

GPT-3
175B Parameters
Content

• Transformer
• GPT
• BERT
Motivation: ELMo + GPT

BERT: Bidirectional Encoder Representations from Transformers

Multi-layer bidirectional Transformer Encoders (not masked)
Question = "What are some example applications of BERT?"

Passage = "...BERT model can be finetuned 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."

Answer: question answering and language inference
Pre-Training Task #1: Masked LM (MLM)

Standard bidirectional conditioning would allow each word to indirectly “see itself” in a multi-layered context

→ Masking 15% of the input tokens at random, then predicting only those masked tokens

\[ L(\mathcal{U}) = \sum_{i \in I} \log P(u_i | u_j \neq i; \Theta) \]

**Downside:** Mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.

- My dog is hairy. → choose hairy
  - 80% of time: my dog is [MASK]
  - 10% of time: my dog is apple
  - 10% of time: my dog is hairy
Question Answering / Natural Language Inference
→ Sentence relationships

Specifically, when choosing the sentence 1 and 2 for each pre-training example:

- 50% of time: 2 is the actual next sentence that follows 1
- 50% of time: 2 is a random sentence

Pre-Training Task #2: Next Sentence Prediction (NSP)
BERT: Bidirectional Encoder Representations from Transformers

Pre-training

Fine-Tuning
BERT: Experiment & Results

Importance of both tasks

<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>BERTBASE</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>

Influence of hyperparameters

<table>
<thead>
<tr>
<th>Hyper-parameters</th>
<th>Dev Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNLI-m</td>
</tr>
<tr>
<td>#L</td>
<td>#H</td>
</tr>
<tr>
<td>3</td>
<td>768</td>
</tr>
<tr>
<td>6</td>
<td>768</td>
</tr>
<tr>
<td>6</td>
<td>768</td>
</tr>
<tr>
<td>12</td>
<td>768</td>
</tr>
<tr>
<td>12</td>
<td>1024</td>
</tr>
</tbody>
</table>
## BERT: Experiment & Results

### BERT is SOTA

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm) 392k</th>
<th>QQP 363k</th>
<th>QNLI 108k</th>
<th>SST-2 67k</th>
<th>CoLA 8.5k</th>
<th>STS-B 5.7k</th>
<th>MRPC 3.5k</th>
<th>RTE 2.5k</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<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&lt;sub&gt;BASE&lt;/sub&gt;</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&lt;sub&gt;LARGE&lt;/sub&gt;</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>
Comparison: GPT & BERT

Word Embedding

\[ N_{vocab} \times d_{model} \]

BERT Encoder

\[ 12N d_{model} \times d_{model} \]

GPT Decoder

\[ 12N d_{model} \times d_{model} \]

<table>
<thead>
<tr>
<th>Model</th>
<th>Layer</th>
<th>( d_{model} )</th>
<th>#parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{BERT}_{\text{BASE}} )</td>
<td>12</td>
<td>768</td>
<td>110M</td>
</tr>
<tr>
<td>( \text{BERT}_{\text{LARGE}} )</td>
<td>24</td>
<td>1024</td>
<td>340M</td>
</tr>
<tr>
<td>GPT</td>
<td>12</td>
<td>768</td>
<td>110M</td>
</tr>
<tr>
<td>GPT2</td>
<td>48</td>
<td>1600</td>
<td>1542M</td>
</tr>
<tr>
<td>GPT3</td>
<td>96</td>
<td>12288</td>
<td>175B</td>
</tr>
</tbody>
</table>
Comparison: GPT & BERT

SuperGLUE Benchmark
Thank You!