

# Attention and Transformers

## Deep Learning

Samuel Cheng

School of ECE  
University of Oklahoma

November 15, 2023

## 1 Word embedding

## Pre-Word2Vec

- A word embedding technique (word represented by a vector)
- Model probability of neighboring words given a center word

$$\begin{aligned} & \arg \max_{\theta} \prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} p(w_{t+j} | w_t; \theta) \\ &= \arg \min_{\theta} \left[ -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t; \theta) \right] \end{aligned}$$

$$p(o|c) = \text{softmax}(u_o^\top v_c)$$

# Word2Vec

- “Distributed representations of words and phrases and their compositionality” (Mikolov et al. 2013)
  - Try to reduce computational complexity
  - Also referred to as the skip-gram model

$$J_t(\theta) = - \left[ \log \sigma(\mathbf{u}_o^\top \mathbf{v}_c) + \sum_{j \sim p(w)} \log(1 - \sigma(\mathbf{u}_j^\top \mathbf{v}_c)) \right]$$

- Alternative model
  - Continuous bag of words (CBOW): model in an opposite manner. Model center word probability with surrounding words

# Latent semantic analysis

- Word2Vec uses a window and goes through entire document
- Latent semantic analysis (aka topic model) looks into co-occurrence count instead
  - Lower complexity
  - Simply generate vector using SVD

# GloVe

- Combine the idea of window and cooccurrence counting
- By Pennington, Socher, Manning (2014)

$$J(\theta) = \frac{1}{2} \sum_{i,j} f(p_{i,j})(u_i^\top v_j - \log p_{i,j})^2$$

# Evaluating word vector

- Intrinsic (intermediate task):
  - Word vector analogy: man to woman = king to ?
  - Word vector distances and their correlation with human judgments
- Extrinsic (real-world task):
  - Name entity recognition
  - Machine translation

## Fun word2vec analogies

Expression	Nearest token
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadians - Montreal + Toronto	Toronto Maple Leafs

Richard Socher



# Evaluation datasets

- Word vector analogies: syntactic and semantic examples  
<http://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt>
- Distances correlated with human judgments  
<http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/>

# Name Entity Recognition (NER)

- Goal: try to predict whether a given word in a sentence is a name and its category
  - Person (**PER**)
  - Organization (**ORG**)
  - Location (**LOC**)
  - Miscellaneous (**MISC**)
- For example,
  - **John** lives in **Oklahoma** and studies at the **University of Oklahoma**
  - The **Republicans** will repeal the **Affordable Care Act**

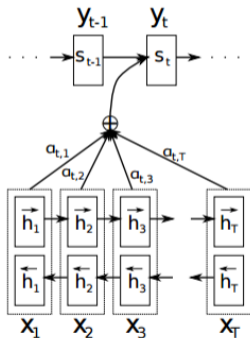
# Problem with RNNs

- Seq2seq models require RNNs to memorize the entire sentence before translating it. It works great for short sentences but performance drops significantly for long sentences
- RNNs are relatively hard and computationally very expensive to train

# Attention for language translation

Bahdanau et al. 2014 (Bengio's group)

- The original model summarizes the input with a single vector  $c$
- Different output position probably more relevant to a part of the input



$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$p(y_t | y_{t-1}, \dots, y_1, x) = g(s_t, y_{t-1}, c_t)$$

with

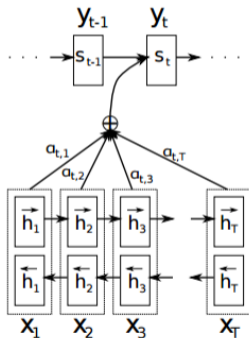
$$c_t = \sum_j \alpha_{t,j} h_j, \quad \alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_k \exp(e_{t,k})}$$

where  $e_{t,j} = a(s_{t-1}, h_j)$  is an alignment score to see how well the inputs around position  $j$  matches output at position  $i$

# Attention for language translation

Bahdanau et al. 2014 (Bengio's group)

- The original model summarizes the input with a single vector  $c$
- Different output position probably more relevant to a part of the input



$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$p(y_t | y_{t-1}, \dots, y_1, x) = g(s_t, y_{t-1}, c_t)$$

with

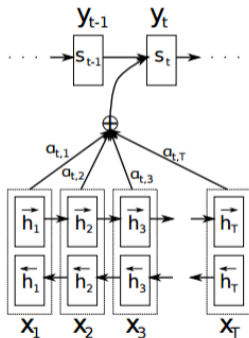
$$c_t = \sum_j \alpha_{t,j} h_j, \quad \alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_k \exp(e_{t,k})}$$

where  $e_{t,j} = a(s_{t-1}, h_j)$  is an alignment score to see how well the inputs around position  $j$  matches output at position  $i$

# Attention for language translation

Bahdanau et al. 2014 (Bengio's group)

- The original model summarizes the input with a single vector  $c$
- Different output position probably more relevant to a part of the input



$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$p(y_t | y_{t-1}, \dots, y_1, x) = g(s_t, y_{t-1}, c_t)$$

with

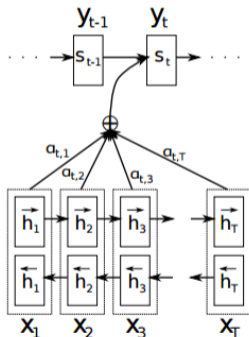
$$c_t = \sum_j \alpha_{t,j} h_j, \quad \alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_k \exp(e_{t,k})}$$

where  $e_{t,j} = a(s_{t-1}, h_j)$  is an alignment score to see how well the inputs around position  $j$  matches output at position  $i$

# Attention for language translation

Bahdanau et al. 2014 (Bengio's group)

- The original model summarizes the input with a single vector  $c$
- Different output position probably more relevant to a part of the input



$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$p(y_t | y_{t-1}, \dots, y_1, x) = g(s_t, y_{t-1}, c_t)$$

with

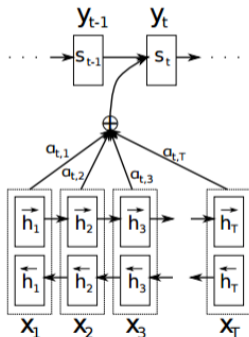
$$c_t = \sum_j \alpha_{t,j} h_j, \quad \alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_k \exp(e_{t,k})}$$

where  $e_{t,j} = a(s_{t-1}, h_j)$  is an alignment score to see how well the inputs around position  $j$  matches output at position  $i$

# Attention for language translation

Bahdanau et al. 2014 (Bengio's group)

- The original model summarizes the input with a single vector  $c$
- Different output position probably more relevant to a part of the input



$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

$$p(y_t | y_{t-1}, \dots, y_1, x) = g(s_t, y_{t-1}, c_t)$$

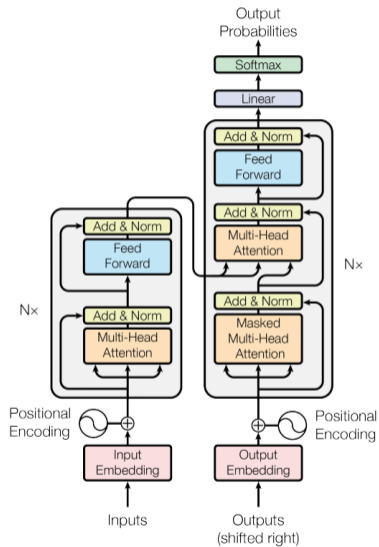
with

$$c_t = \sum_j \alpha_{t,j} h_j, \quad \alpha_{t,j} = \frac{\exp(e_{t,j})}{\sum_k \exp(e_{t,k})}$$

where  $e_{t,j} = a(s_{t-1}, h_j)$  is an alignment score to see how well the inputs around position  $j$  matches output at position  $i$



# Transformer



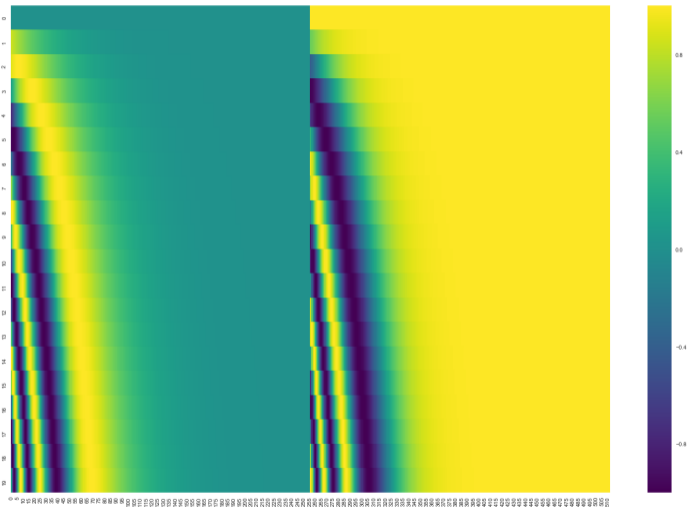
# Transformer

$$\text{Multihead}(Q, K, V) = W_0 \text{concat}(\text{Head}_1, \dots, \text{Head}_n)$$

where  $\text{Head}_i = \text{Attention}(W_i^Q Q, W_i^K K, W_i^V V)$

$$\text{Attention}(q, k, v) = \text{softmax}\left(\frac{q^\top k}{\sqrt{d_k}}\right) v$$

# Positional encoding



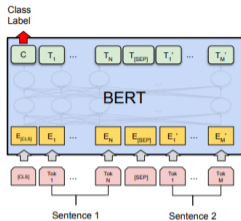
# BERT

- “Bidirectional Encoder Representations from Transformers”: encoder only model
- Quite a bit larger model size
  - Base model: 12 encoder blocks (layers), embedding (hidden) size 768, 12 heads (110M in total)
  - Large model: 24 encoder blocks, embedding size 1024, 16 heads (340M in total)
  - In contrast, the original transformer model has 6 encoder and 6 decoder blocks, 512 embedding size, and 8 heads

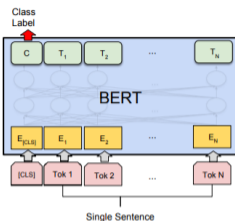
# BERT Pretraining

- The main idea is bidirectional. It is obvious but we can train such model with the original task
- The authors pre-train BERT with the following tasks
  - Mask LM (MLM)
  - Next Sentence Prediction (NSP)
    - Input = [CLS] the man went to [MASK] store [SEP]  
he bought a gallon [MASK] milk [SEP]  
Label = IsNext
    - Input = [CLS] the man [MASK] to the store [SEP]  
penguin [MASK] are flight ##less birds [SEP]  
Label = NotNext

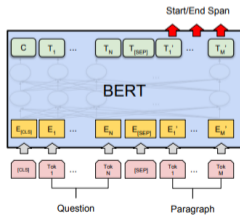
## BERT finetuning/applications



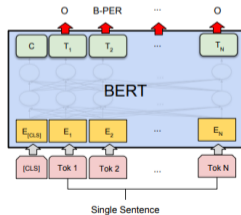
(a) Sentence Pair Classification Tasks:  
 MNLI, QQP, QNLI, STS-B, MRPC,  
 RTE, SWAG



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



(c) Question Answering Tasks:  
 SQuAD v1.1



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

# BERT Positional encoding

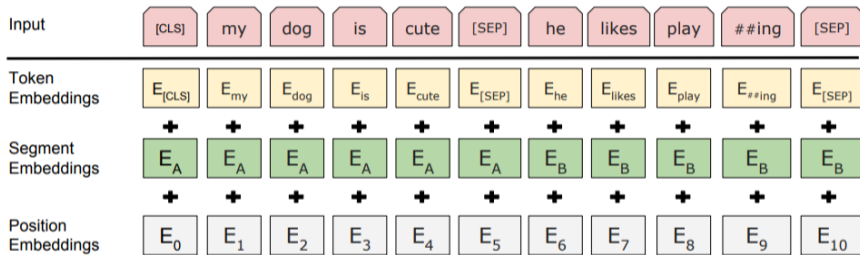


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

# Comparison

	<b>BERT</b>	<b>RoBERTa</b>	<b>DistilBERT</b>	<b>XLNet</b>
<b>Size (millions)</b>	<b>Base:</b> 110 <b>Large:</b> 340	<b>Base:</b> 110 <b>Large:</b> 340	<b>Base:</b> 66	<b>Base:</b> ~110 <b>Large:</b> ~340
<b>Training Time</b>	<b>Base:</b> 8 x V100 x 12 days* <b>Large:</b> 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	<b>Large:</b> 1024 x V100 x 1 day; 4-5 times more than BERT.	<b>Base:</b> 8 x V100 x 3.5 days; 4 times less than BERT.	<b>Large:</b> 512 TPU Chips x 2.5 days; 5 times more than BERT.
<b>Performance</b>	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
<b>Data</b>	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	<b>Base:</b> 16 GB BERT data <b>Large:</b> 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
<b>Method</b>	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

<https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8>



# GPT/GPT-2

- GPT means generative pre-training
- Language model from OpenAI
- If we only care about building a model (not translation), only need decoders
- Can be use for different task with little refinement (transfer learning)

# Early GPT applications

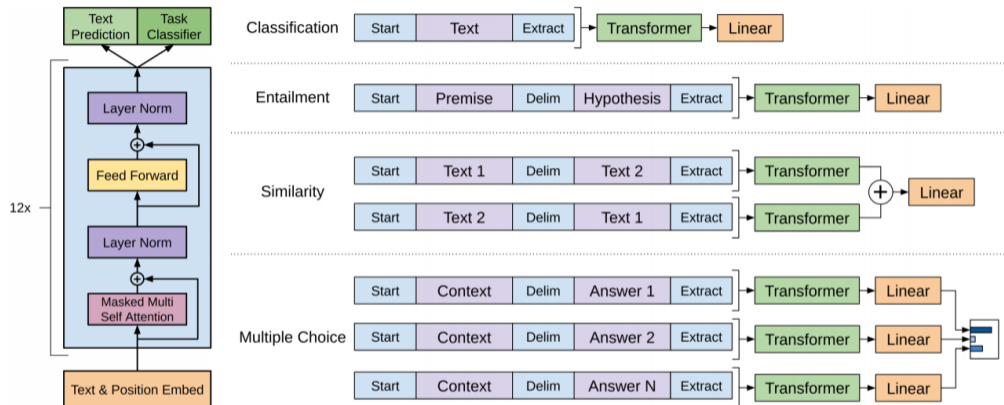


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

# GPT-3/GPT-4

- Number of parameters: 1.5B (GPT2)  $\rightarrow$  175B (GPT3)  $\rightarrow$  1T (GPT4)
- Improve tokenization: better implementation of Byte-Pair Encoding (BPE) in GPT3
- Layer scaling in GPT3
  - GPT2 scale the model based on the number of parameters alone
  - GPT3 scale depth, width, and attention heads differently
    - Depth (number of layers) grows logarithmically with the model size
    - Width (hidden layer size) grows linearly with the model size
    - Attention heads also grow linearly with the model size
- Sparse attention: GPT3 experimented several sparse attention
  - approximates the dense attention mechanism by only considering a subset of the token pairs
  - GPT3 is still mainly dense attention. Maybe used in GPT4?
- GPT3 can perform zero-shot or few-shot learning task

# Conclusion

- Unlike RNN, attention network can easily “attend” to elements far away without trying to remember everything all at once
- And in contrast to CNN where the reception field is usually limited to the neighborhood, the attention can extend to very far away
- One can think of the limitation in CNN is some form of inductive bias. As datasize increases, reducing inductive bias will improve performance
  - It may not be surprising that attention networks/transformers performs better than CNN
- Attention networks/transformers can be applied beyond NLP
  - E.g., Vision transformer takes embedded 16x16 patches as tokens and applies transformers as in the case of NLP