# CS 4248
# Natural Language Processing

**Professor NG Hwee Tou**
**Department of Computer Science**
**School of Computing**
**National University of Singapore**
**nght@comp.nus.edu.sg**

# Drawbacks of RNN

- Dealing with long range dependencies
- Need large number of training steps
- Inherently sequential nature of RNNs hinders parallel computation

# Transformer Networks

- Better exploit parallel processing
- Improved training speed (and improved accuracy)
- Self-attention neural networks
- Map a sequence of input vectors to a sequence of output vectors
- First used for machine translation

# Self-Attention

- Compute the similarity between a vector in a sequence and every vector in the sequence
- $n^2$ pairs in a sequence of $n$ vectors
- Relevance of a vector to another vector in the context (“attention”)

# Self-Attention

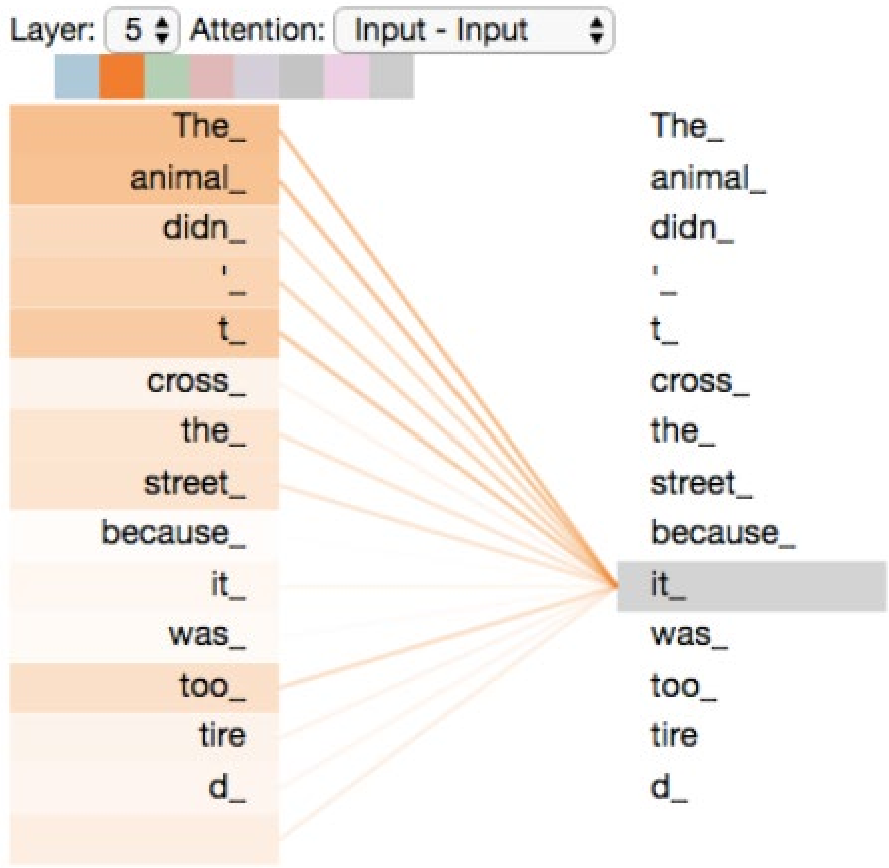



Diagram from "The illustrated transformer" blog post
http://jalammar.github.io/illustrated-transformer/

# Query, Key, Value

- Retrieves a value $\boldsymbol{v}$ for a query $\boldsymbol{q}$ by matching $\boldsymbol{q}$ to a key $\boldsymbol{k}$ and accessing its associated value $\boldsymbol{v}$
- $\sum_i (\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}_i)\boldsymbol{v}_i)$

$\boldsymbol{q} \longrightarrow$

| | |
|---|---|
| $\boldsymbol{k}_1$ | $\boldsymbol{v}_1$ |
| $\boldsymbol{k}_2$ | $\boldsymbol{v}_2$ |
| $\vdots$ | $\vdots$ |
| $\boldsymbol{k}_n$ | $\boldsymbol{v}_n$ |

$\boldsymbol{q}$ : query vector representation of the current token

# Self-Attention

$\boldsymbol{x}_i$: input vector

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

$$sim(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \frac{\exp(sim(\boldsymbol{x}_i, \boldsymbol{x}_j))}{\sum_{k=1}^{n} \exp(sim(\boldsymbol{x}_i, \boldsymbol{x}_k))}$$

$\boldsymbol{z}_i$: output vector

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij} \boldsymbol{v}_j$$

$d_k$: dimension of $\boldsymbol{k}$

# Self-Attention

- Expressed compactly in matrix form:

$$Q = XW^Q \quad K = XW^K \quad V = XW^V$$

$$Z = \mathrm{softmax}_{\mathrm{row}}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$\boldsymbol{x}_i$: row $i$ of $X$ $\quad$ $\boldsymbol{q}_i$: row $i$ of $Q$

$\boldsymbol{k}_i$: row $i$ of $K$ $\quad$ $\boldsymbol{v}_i$: row $i$ of $V$

$\boldsymbol{z}_i$: row $i$ of $Z$

# Multi-Head Self-Attention

- Head $h$: One set of $W_h^Q, W_h^K, W_h^V$ contributes $Z_h$
- $H$: total number of heads
- Each head models one "representation subspace" or one aspect of the relationships

$$Z = (Z_1 \oplus Z_2 \oplus \cdots \oplus Z_H) W^O$$

# Multi-Head Self-Attention

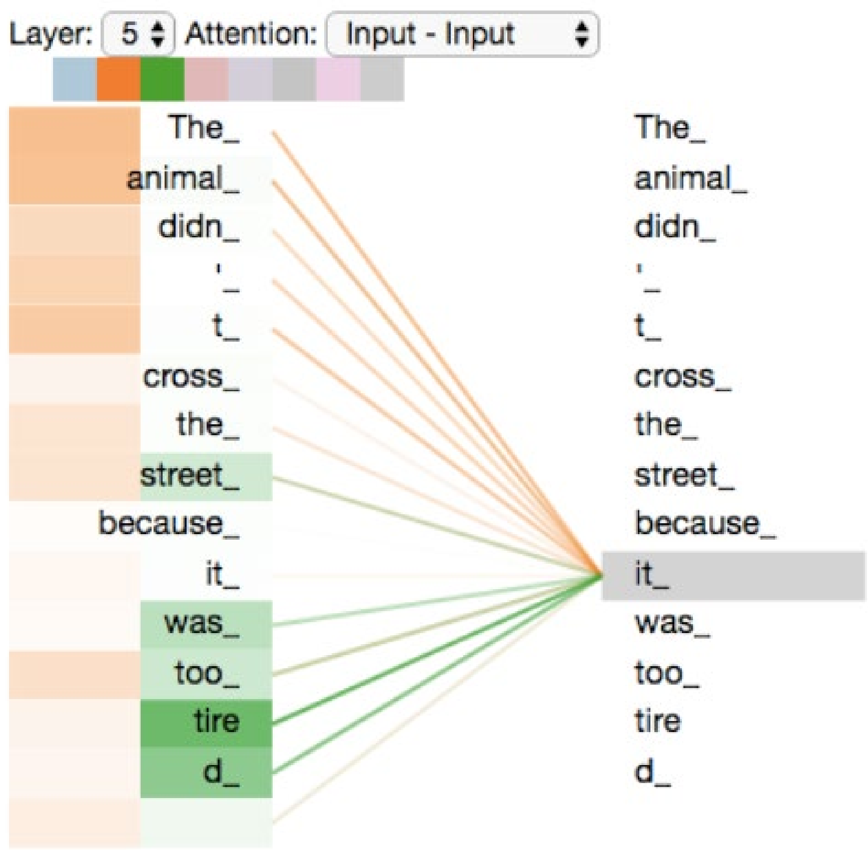



Diagram from “The illustrated transformer” blog post
http://jalammar.github.io/illustrated-transformer/

# Positional Encoding

- Order of words is important in language processing
- Need to indicate the position of each input vector in the sequence
- Positional embedding: a vector of sine and cosine function for each position
- Input vector: Add positional embedding to word embedding for each word

# Transformer Model Architecture

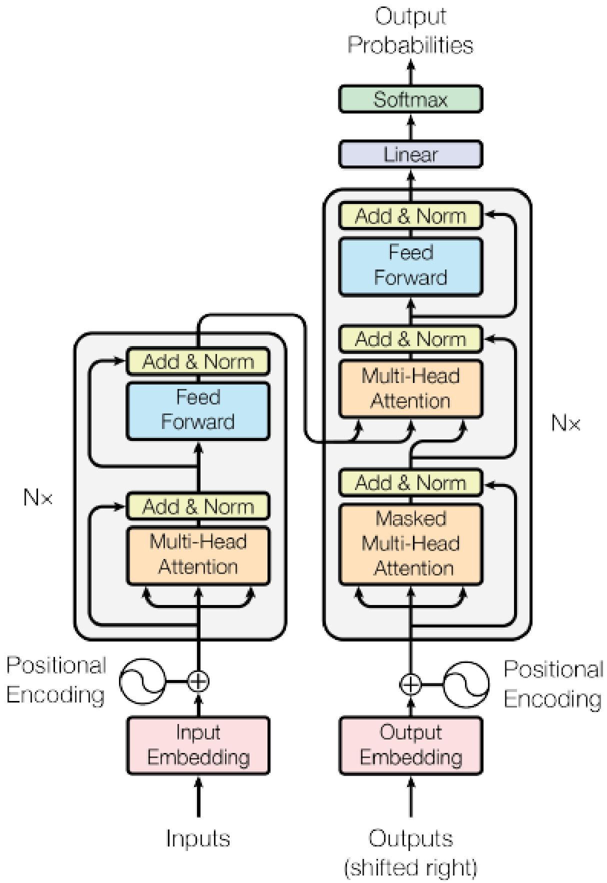



Diagram from “Attention is all you need”, NeurIPS 2017

# Transformer Model Architecture

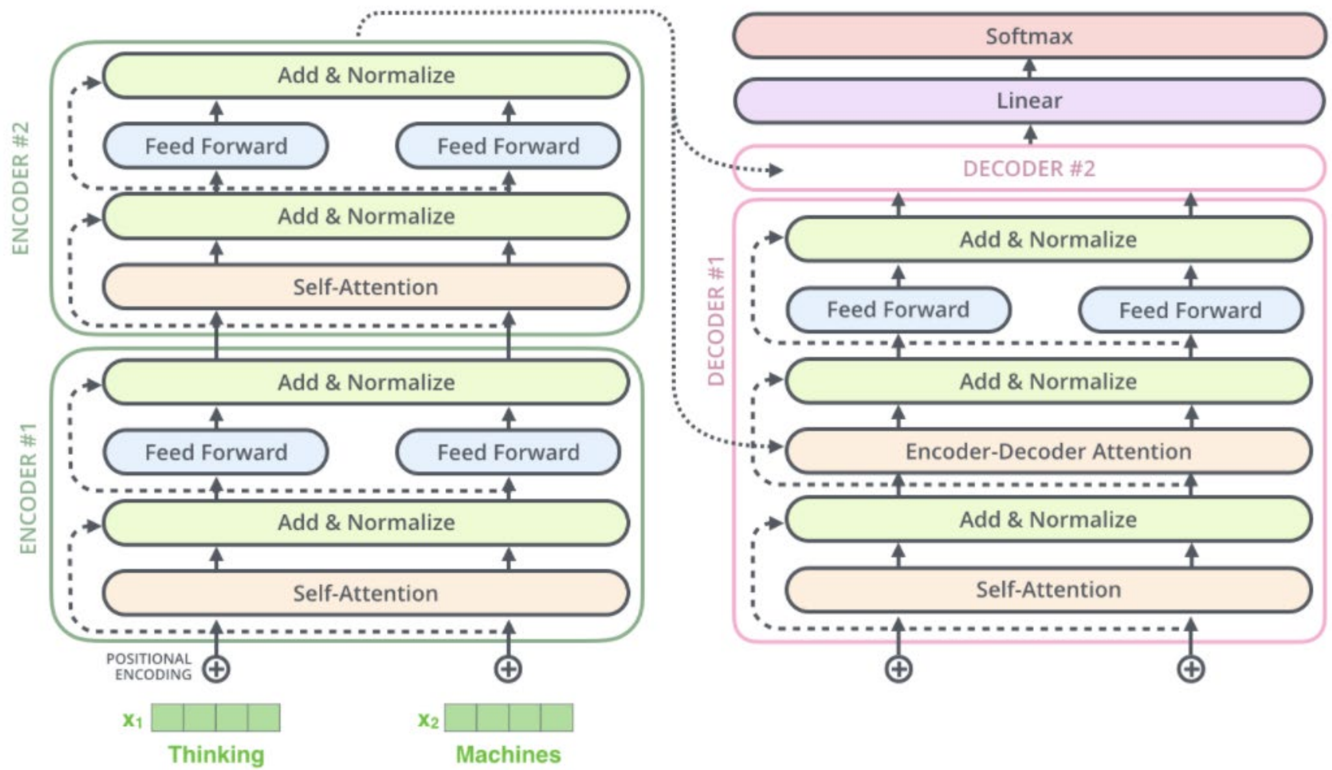



Diagram from “The illustrated transformer” blog post
http://jalammar.github.io/illustrated-transformer/

# Transformer Block

$$\boldsymbol{y} = \mathrm{LayerNorm}(\boldsymbol{x} + \mathrm{SelfAttn}(\boldsymbol{x}))$$

$$\boldsymbol{z} = \mathrm{LayerNorm}(\boldsymbol{y} + \mathrm{FFNN}(\boldsymbol{y}))$$

# Encoder-Decoder Attention

- The decoder has an extra cross-attention layer
  - $Q$: based on output of the previous layer of the decoder $X_{dec}$
  - $K, V$: based on final output of the encoder $X_{enc}$
  - $Q = X_{dec}W^Q \quad K = X_{enc}W^K \quad V = X_{enc}W^V$

# Layer Normalization

- Normalize values in each NN layer to have mean = 0 and variance = 1
- For each hidden unit $h_i$:

$$\mu = \frac{1}{H}\sum_{i=1}^{H} h_i \text{ and } \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(h_i - \mu)^2}$$

$$h_i \leftarrow g \times \left(\frac{h_i - \mu}{\sigma}\right) \quad \text{where } g \text{ is a parameter}$$

- Keep the values of a hidden layer in a range
- Fewer training iterations needed (converge faster)

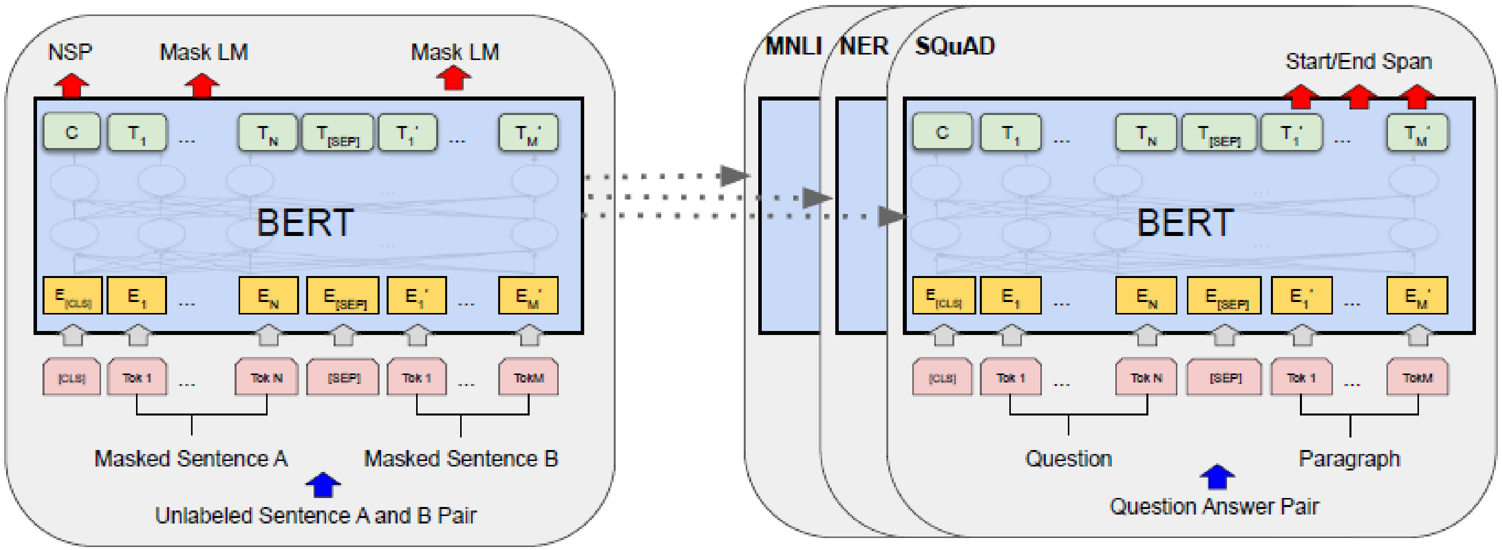
# BERT

- Bidirectional Encoder Representations from Transformers
- Pre-train contextualized representations from unlabeled text
- Consider both left and right context (bidirectional)
- Fine-tune with one additional output layer for many downstream (sentence-level and token-level) NLP tasks with excellent performance

# BERT

- Transfer learning
- Two steps: pre-training then fine-tuning
- Pre-training: Train the model on unlabeled text for masked language model and next sentence prediction
- Fine-tuning: Initialize the model with the pre-trained parameters, and all parameters are fine-tuned using labeled training data from the downstream NLP task

# BERT




Pre-training

Fine-Tuning

Diagram from the BERT paper, NAACL 2019
“BERT: Pre-training of deep bidirectional transformers for language understanding”

# Model Architecture of BERT

- Multi-layer bidirectional transformer encoder
- L: number of transformer layers (blocks)
- d: hidden vector size
- H: number of self-attention heads
- $\text{BERT}_{\text{BASE}}$: L = 12, d = 768, H = 12 (110M parameters)
- $\text{BERT}_{\text{LARGE}}$: L = 24, d = 1024, H = 16 (340M parameters)

# BERT Input Representation

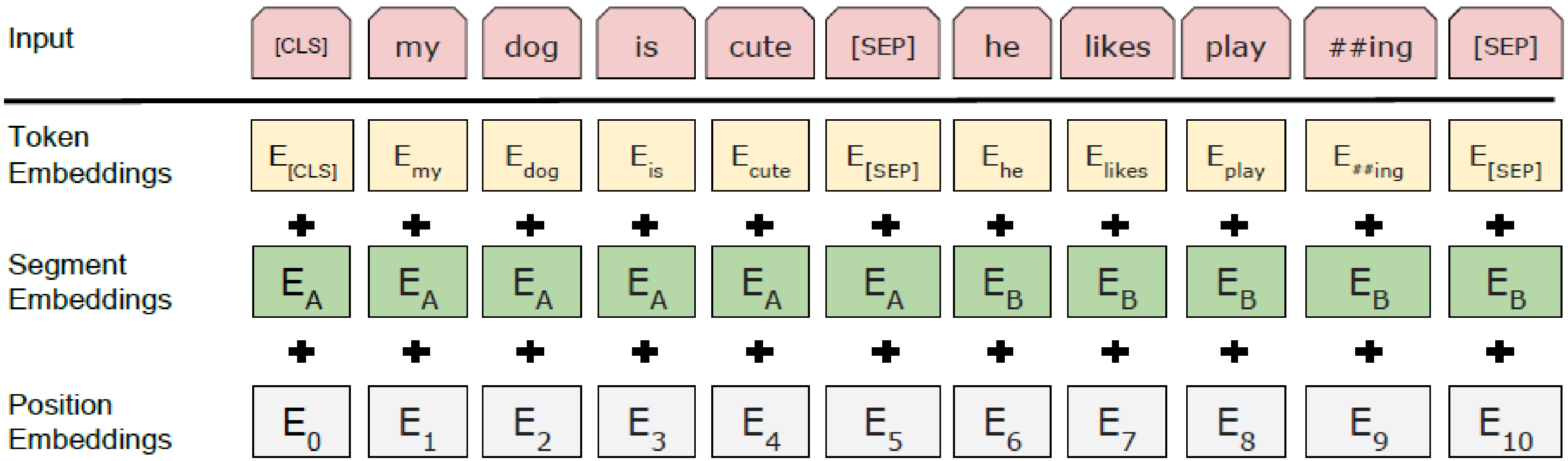



Diagram from the BERT paper, NAACL 2019

# Pre-Training of BERT

- Masked language model
  - Randomly mask 15% of all input tokens
  - Model to predict the masked tokens
- Next sentence prediction
  - Choose sentences A and B for each pre-training example
  - IsNext: B is the actual next sentence that follows A
  - NotNext: B is a random sentence from the corpus
  - Two-class prediction based on [CLS]
- Cross-entropy loss

# Pre-Training Corpora

- BooksCorpus (800M words)
- English Wikipedia (2.5B words)
- Total size = 3.3B words

# Contextual Embeddings

- Given pretrained BERT and a new input sentence of tokens $x_1, \ldots, x_n$, the output vector $z_i$ from the final layer of BERT is a contextual embedding or representation of $x_i$ in the context of the input sentence

# Fine-Tuning BERT

- Sentence A and B:
  - Natural language inference (NLI) determines the relation between the first sentence (premise) and the second sentence (hypothesis): entail or contradict or neutral
  - Question-passage pairs in question answering
  - Degenerate text – Ø pair in text classification or sequence tagging

# Fine-Tuning BERT

- Sentence-level task (classification): Final output vector for the [CLS] token
  - Sentiment classification
- Token-level task (sequence tagging): Final output vector for the $i^{th}$ input token
  - POS tagging, named entity recognition