COMP 3225

Natural Language Processing

Sequence Processing, Transformer and Attention

Stuart E. Middleton

sem03@soton.ac.uk

University of Southampton

Overview

- LSTM
- GRU
- Gated Units, Layers and Networks
- Transformer

- <break - discussion point>

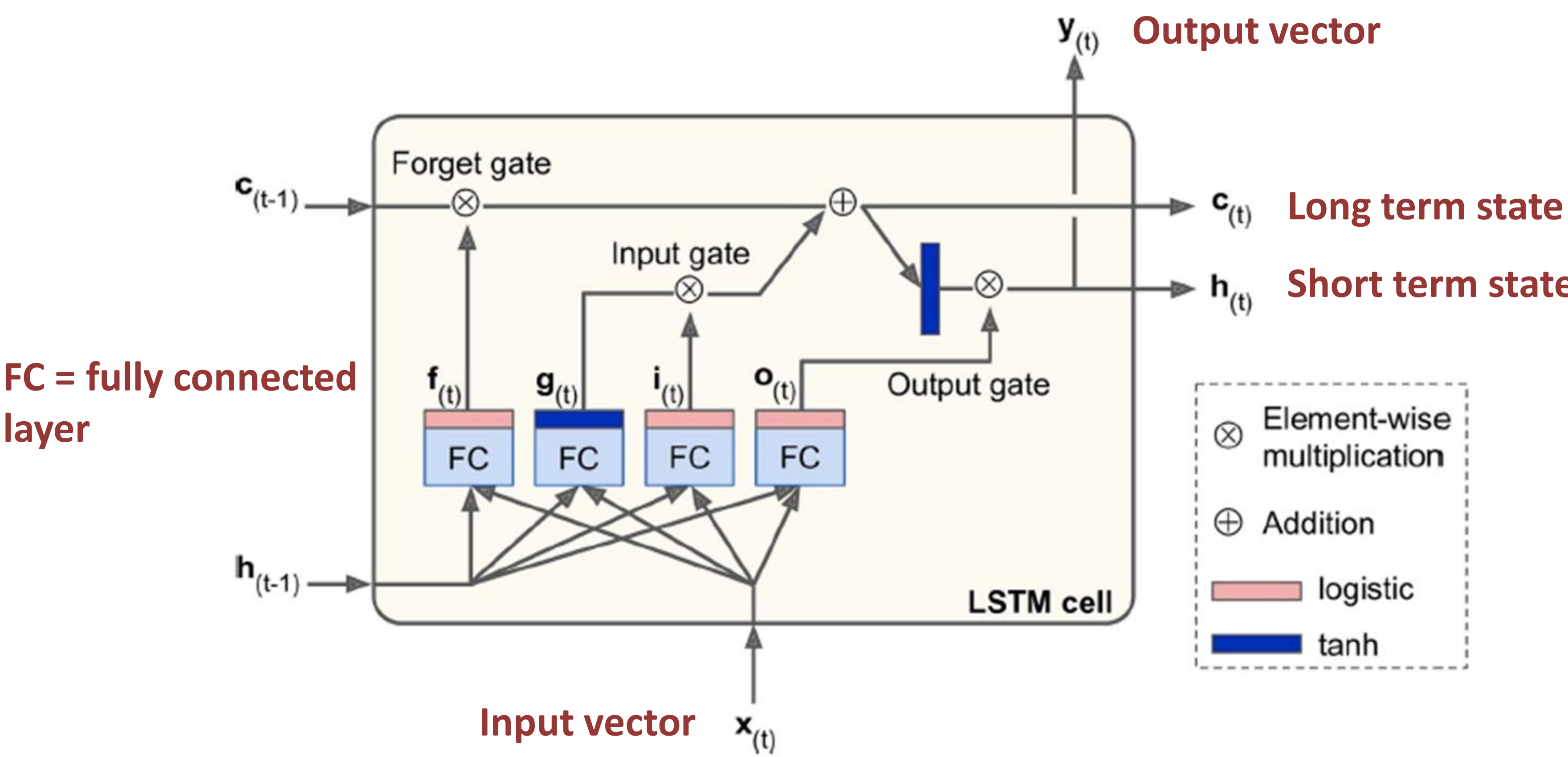
- Self-attention
- Transformer for Text Completion Task

LSTM

- Long Short-Term Memory (LSTM)
- Long-distant information is critical to many language apps
  - Information encoded by a single RNN layer tends to be fairly local
  - Information is lost each training step, so encoded memory of tokens far away in sequence degrades pretty quickly
- Vanishing gradients problem
  - Gradients get smaller and smaller as Gradient Decent progresses to deeper stacked layers - they 'vanish'
  - Connection weights are virtually unchanged, so training loss does not converge
- Exploding gradients problem
  - The opposite can also happen, where gradients get larger and larger and the training loss again does not converge
  - In general these problems lead to unstable gradients

LSTM

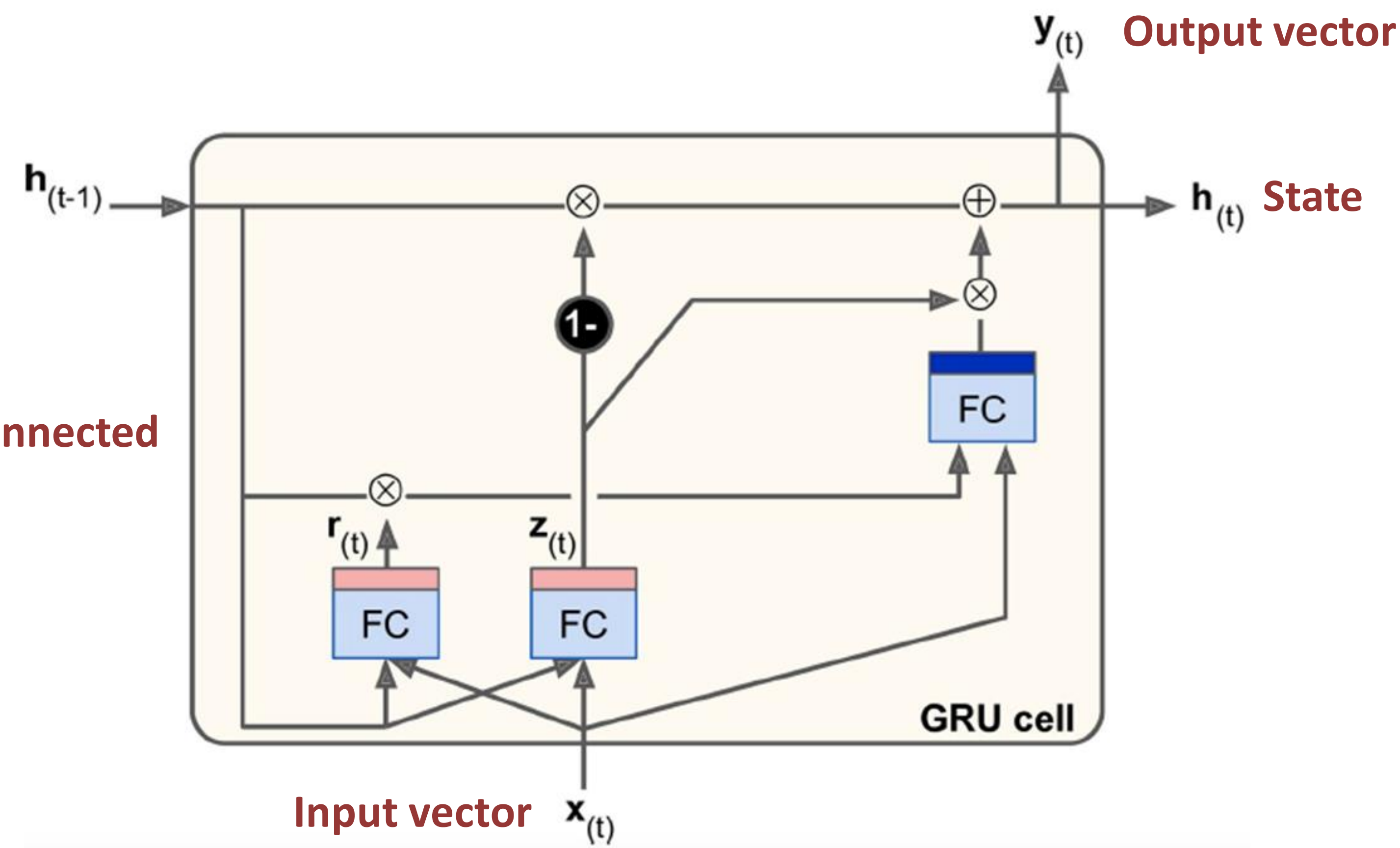
- Long short-term memory (LSTM)
  - Forgets information no longer relevant (forget gate)
  - Adds new information (input gate sometimes called add gate)
  - Gate = layers → logistic activation function σ (value 0..1), followed by a pointwise multiplication to provide a type of binary mark



GRU

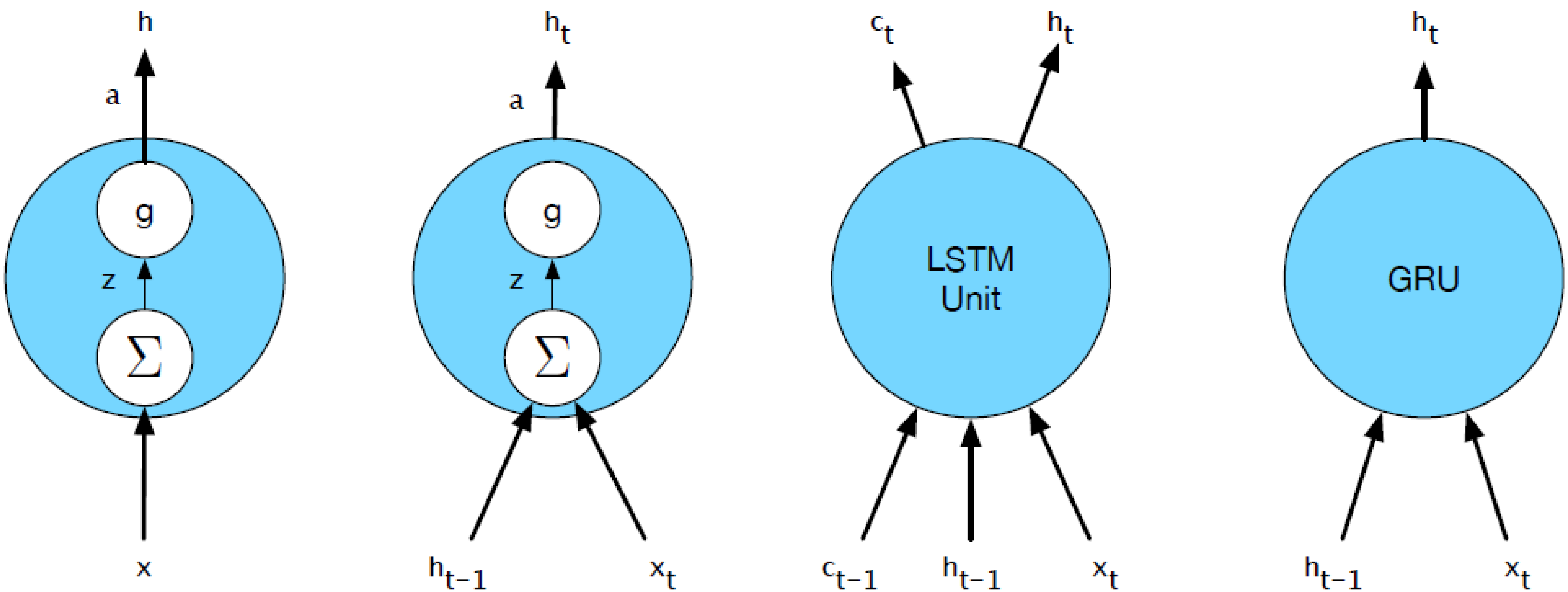
- Gated Recurrent Unit (GRU)
  - Merges both long and short term state vectors
  - Simpler (less layers) and often performs as well as LSTM

FC = fully connected layer



Gated Units, Layers and Networks

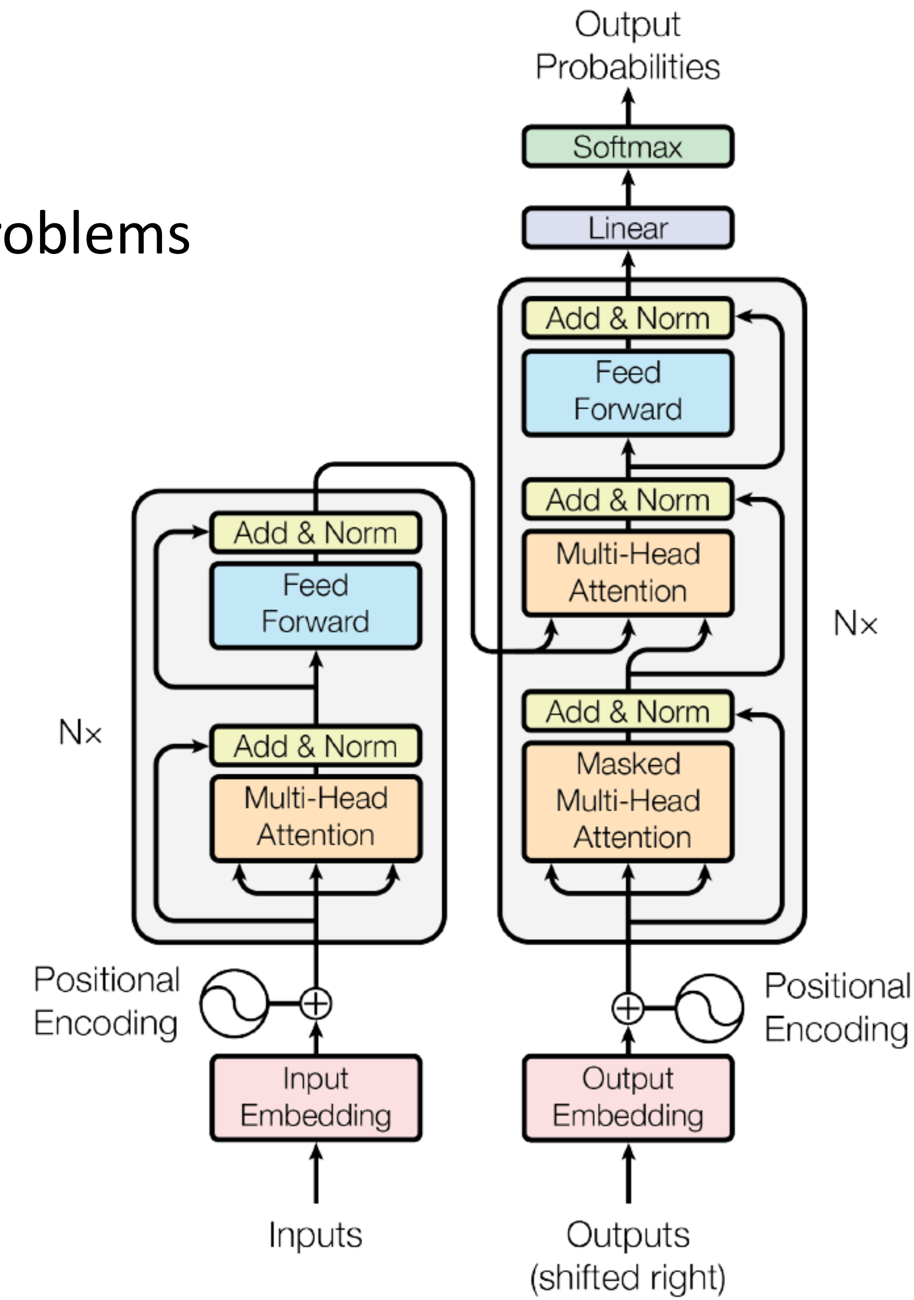
- Stacked RNN, LSTM and GRU layers can be experimented with easily using frameworks like Tensorflow and PyTorch



MLP cell **RNN cell** **LSTM cell** **GRU cell**

Transformer

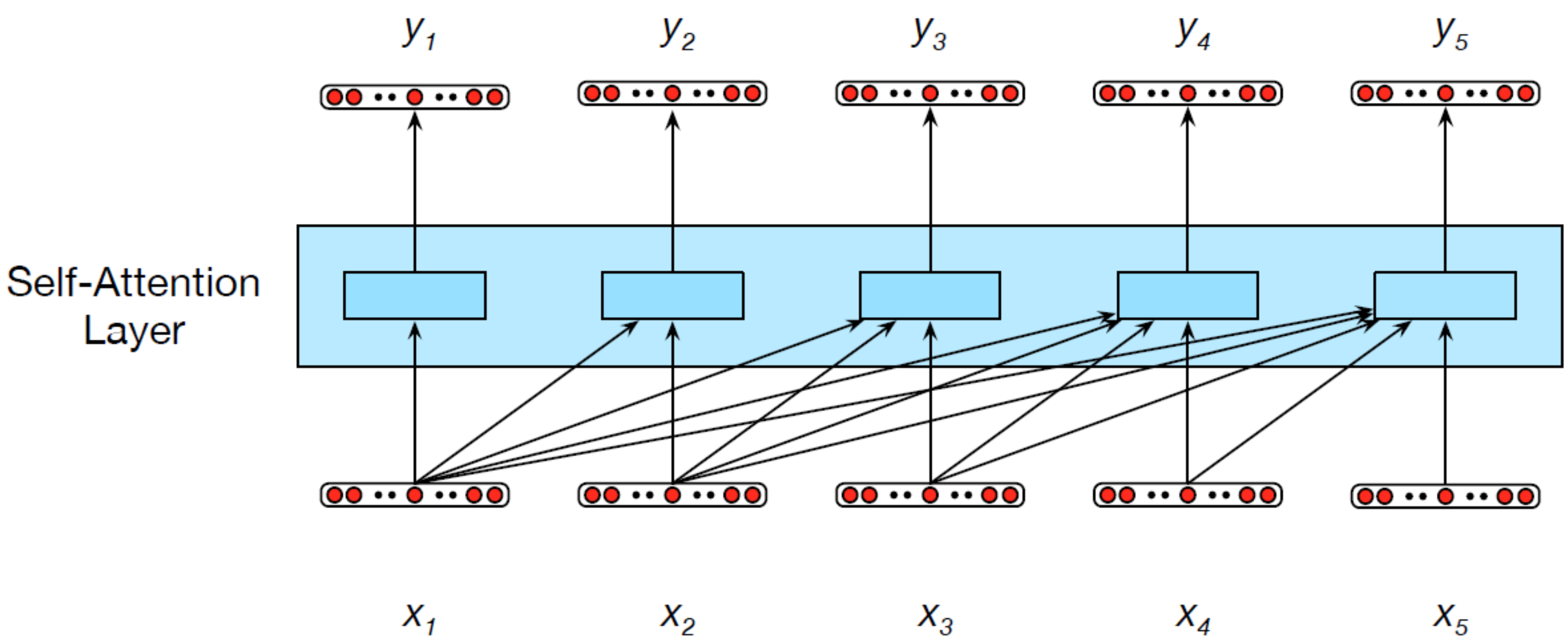
- Even LSTM and GRU's have problems with very long sequences
- Transformer architecture
  - Uses attention layers
    not sequential RNN layers
  - This architecture has led to many advances in NLP model performance



Cite: Vaswani, A. Shazeer, N. Parmar, N. Uszkoreit, J. Jones, L. Gomez, A.N. Kaiser, L. Polosukhin, I. (2017) Attention Is All You Need, arXiv:1706.03762v5

Self-attention

- Attention layers >> compare item to other items to reveal their relevance in the current context
  - Self-attention compares to other elements within same sequence
- Self-attention layer maps input sequences $(x_1,..., x_n)$ to output sequences of the same length $(y_1,..., y_n)$
  - Layer computes for position i the y_i using input $x_1...x_i$ (uses context)
  - Layer computation independent of other layers (parallelizable)



Self-attention

- Compare items (embedding vector for a word) in a sequence using dot product using a score function

$$score(x_i, x_j) = x_i \cdot x_j$$

$$\alpha_{ij} = \text{softmax}(score(x_i, x_j)) \;\; \forall j \leq i$$

j = all values up to i

$$y_i = \sum_{j \leq i} \alpha_{ij} x_j$$

- For seq $x_1..x_3$ we need compute score(x_3,x_1), score(x_3,x_2), score(x_3,x_3)
- Normalize scores using softmax to create a vector of attention weights α
- Compute output by summing inputs in seq $x_1..x_i$ weighted by α

- But ... to allow learning we need trainable weights

Break

- Panopto Quiz - discussion point

- How can we provide trainable weights for the attention function?

We don't need to! we have the attention weights α already
Use a LSTM layer
Use an weight vector
Use an weight matrix

$$y_i = \sum_{j \leq i} \alpha_{ij} x_j$$

Break

- Panopto Quiz - discussion point

- How can we provide trainable weights for the attention function?

We don't need to! we have the attention weights α already >> we need weights to train
Use a LSTM layer >> there is no sequence structure in an attention layer
Use an weight vector >> No
Use an weight matrix >> Yes, actually several matricies as you will see next

$$y_i = \sum_{j \leq i} \alpha_{ij} x_j$$

Self-attention

- Imagine encoder learns a dictionary
  - concept → lexical term == key → query
    'They played chess' >> 'subject' → 'They'; 'verb' → 'played'
  - Concepts are learnt in latent space do not map directly to single words
- Decoder can 'lookup' values for concepts it think go next in seq
- Transformer uses embeddings to provide this dictionary which is then differentiable (for gradient decent)
  - Query Q = matrix [$n_{queries}$, d_{keys}], where $n_{queries}$ = input seq length
  - Key K = matrix [n_{keys}, d_{keys}], where n_{keys} = number of keys
  - Value V = matrix [n_{keys}, d_{values}]
  - d_{keys} = dimensions of query and key embedding

$$q_i = W^Q x_i; \ \ k_i = W^K x_i; \ \ v_i = W^V x_i$$

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

$$y_i = \sum_{j \le i} \alpha_{ij} v_j$$

layer output =
weighted sum of (attention x value)
for each word in seq

Self-attention

- Since computation of output q_i can be done independently, we can use matrix multiplication to parallelize training

$$q_i = W^Q x_i;\ \ k_i = W^K x_i;\ \ v_i = W^V x_i$$

$$Q = W^Q X;\ \ K = W^K X;\ \ V = W^V X$$

$$SelfAttention(Q, K, V) \ = \ \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

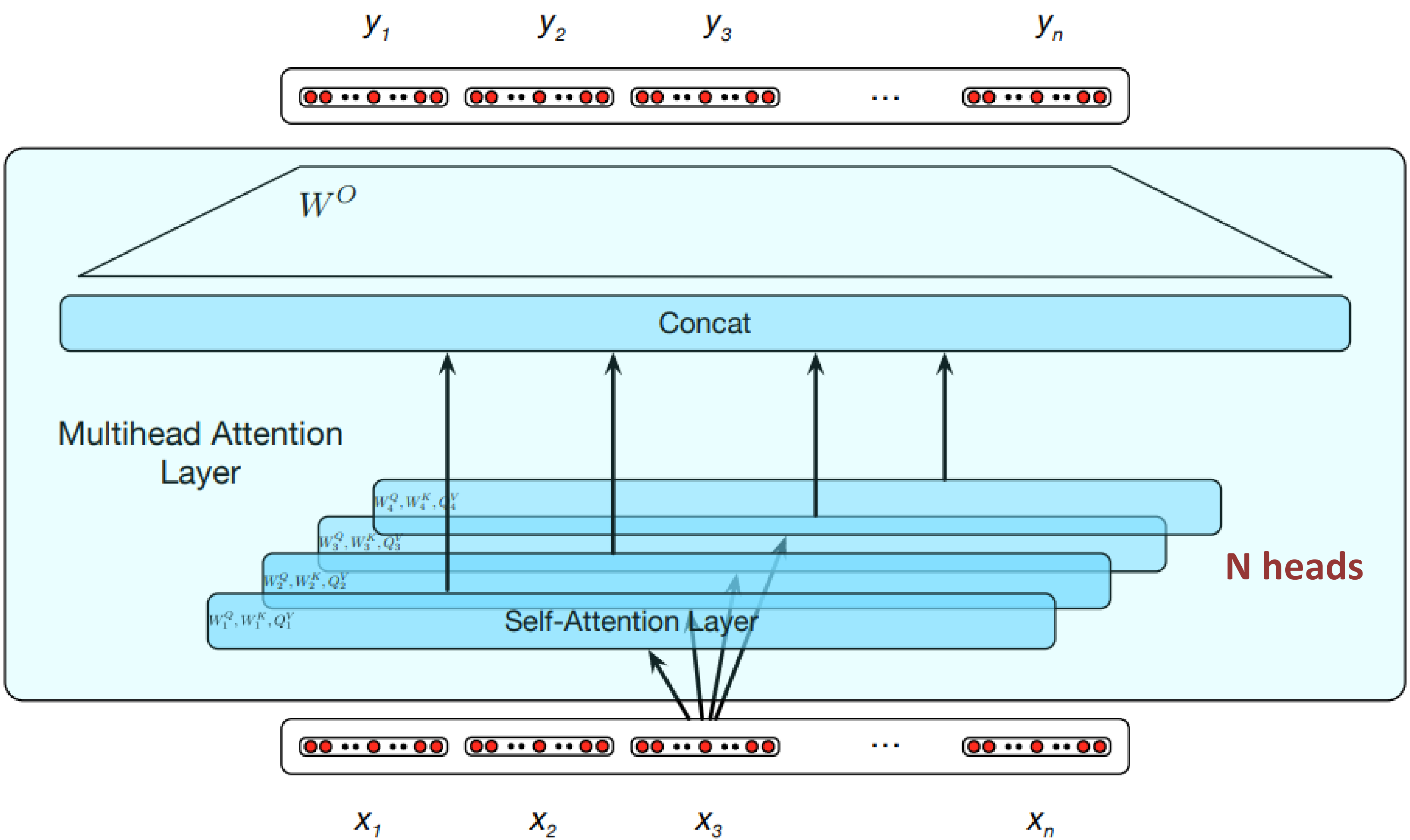
- For language modeling we do not want to use sequence information after the current position (that would be cheating)
- We can blank (i.e. give a value of -∞) matrix values after the current position to hide them

Self-attention

- Positional embeddings
  - The position of words in sequences is important for language problems
  - Transformer attention layers do not encode the position of words (unlike RNNs), so it uses an additional positional embedding
  - These can be learned during training (e.g. additional position input alongside words) or generated using a positional encoding function (e.g. cosine/sine as per Vaswani 2017 paper)
  - Positional embeddings have the same dimensions as word embeddings, so are just summed together to give a position-aware input embedding

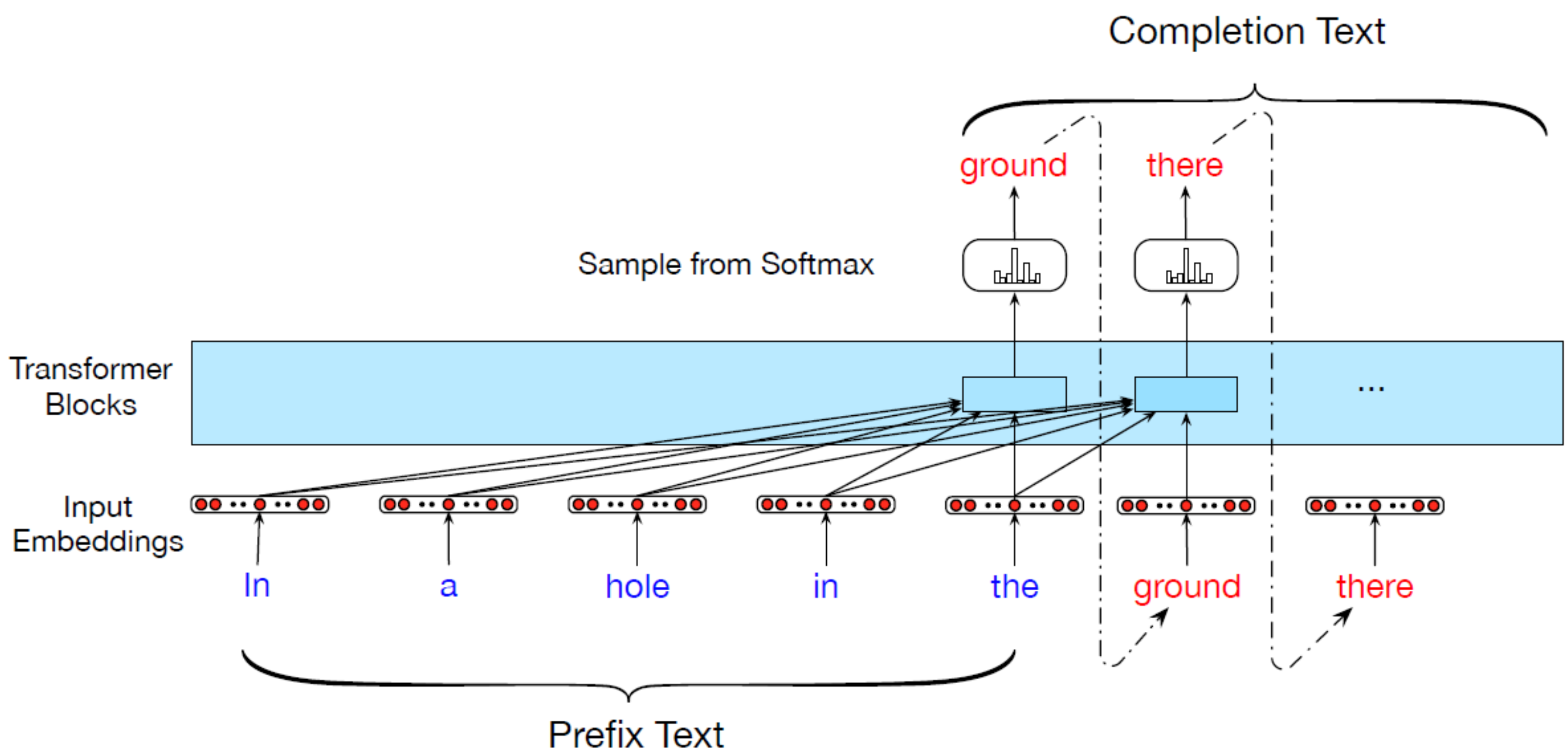
Self-attention

- Multi-head Self-attention Layers
  - A self-attention layer is called a head
  - There can be N heads in a model providing a deep learning stack



Transformer for Text Completion Task

- Example using Transformer model for text completion
- Input = sequence of words
- Output = prediction of words to complete sequence
- Transformer architectures have become a 'go to' model for NLP



Required Reading

- LSTM, GRU, Transformer and Self-attention
  - Jurafsky and Martin, Speech and Language Processing, 3rd edition (online)
    >> chapter 9
- Transformer and Self-attention
  - Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O’Reilly , 2017
    >> Chapter 16 'attention mechanisms'
- LSTM, GRU (optional)
  - Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O’Reilly , 2017
    >> Chapter 15 'handling long sequences'

Questions

- Panopto Quiz - 1 minute brainstorm for interactive questions

  Please write down in Panopto quiz in **1 minute** two or three questions that you would like to have answered at the next interactive session.

  Do it **right now** while its fresh.

  Take a screen shot of your questions and **bring them with you** at the interactive session so you have something to ask.