CS5242 : Neural Networks and Deep Learning

Lecture 12: Attention Neural Networks

Semester 1 2021/22

Xavier Bresson

https://twitter.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

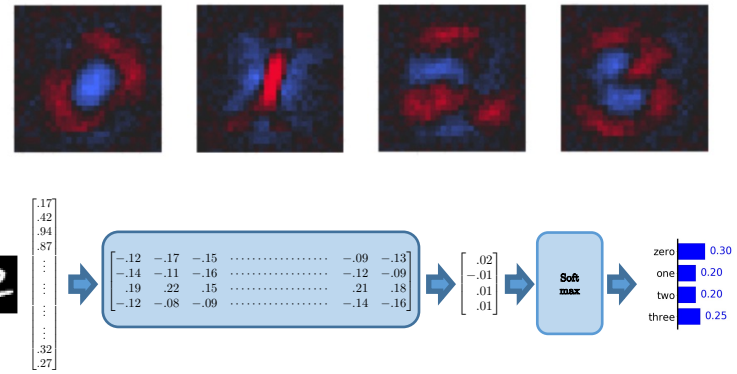
Neural Networks

- Main goal of neural networks is to learn the best possible (continuous) data representation that can be used to solve any arbitrary downstream tasks s.a. classification, regression, recommendation, etc.
- How to design neural networks ?
  - Identify data properties/structures/invariances that are as most universal and minimalist as possible,
  - Design layers that capture this(ese) structure(s), and stack many of them.
  - Be aware that NNs are hard to debug ! Bad NNs seem to work, but are slow to train, require lots of data, does not generalize well ($\neq$ software 1.0).
- Let us review next the main classes of NNs.
  - MLP, CNNs, RNNs

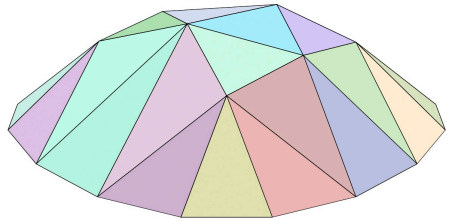
MLP/FC Networks

- Data structure
  - Input data is composed of fixed-size templates/patterns.
- MLP performs pattern matching for fixed input size.
- Compositional function :

$$F_0 \circ F_1 \circ ... \circ F_{L-1} \circ F_L$$
$$\text{where } F_l(x_{l-1}) = \sigma(A_l x_{l-1} + b_l) = x_l$$
$$\text{where } A_l, b_l \text{ are learned by SGD}$$

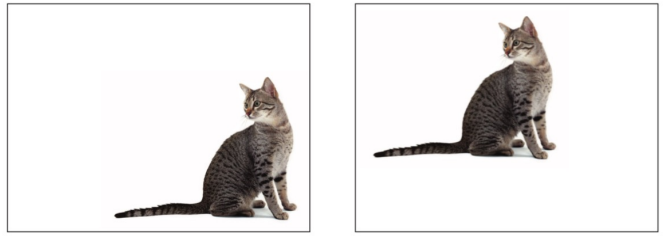


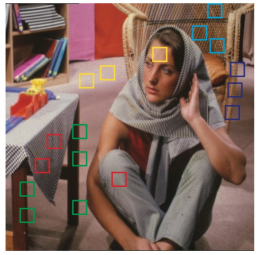
- Performance
  - Great for (piecewise-)linear data.

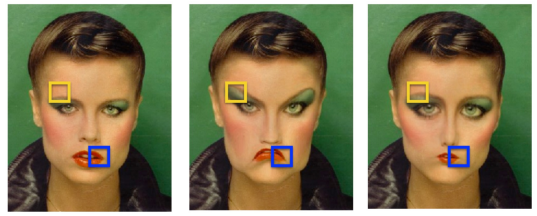


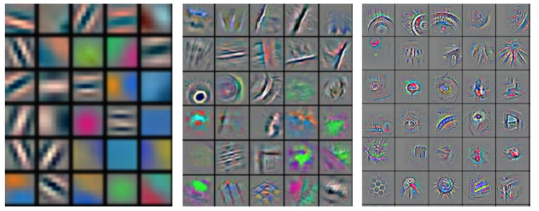
CNNs

- Data structure
  - Input data has a grid structure like 2D/3D images and videos.
- CNNs capture image properties
  - Local and global translation invariance on grids (stationarity)
    - Object is recognized independently of its position.
  - Local deformation invariance on grids (diffeomorphism)
  - Hierarchical data representation (multi-scale property)
- Performance
  - Great for grid-structured data
  - Revolution in Computer Vision since 2012









CNNs

- Compositional function :

$$F_0 \circ F_1 \circ ... \circ F_{L-1} \circ F_L$$

where $F_l(x_{l-1}) = \sigma(A_l * x_{l-1} + b_l) = x_l$

$*$ is the convolutional operator

A_l is the fixed-size kernel with compact support

A_l, b_l are learned by SGD

- Convolution operation
  - It is a linear operation (like MLP) but specialized to template matching with sliding window (translation).
  - It is independent to the input size (MLP is not), although images are usually resized to the same shapes (for batch computational efficiency/GPU).

RNNs

- Data structure
  - Input is an ordered sequence (or 1D grid structure).
  - Input length and output length can be variable.
- RNNs are designed for sequences
  - They learn a representation of sequence independently of its length.
    - Recurrence formula which summarizes the sequence with a vector h :

$$h \leftarrow f_W(h, x)$$

    - Weight sharing across time (translation invariance)
  - They learn to keep or ignore information in the sequence for the downstream task.
    - Gating mechanism to forget/remember the past or the new input :

$$\sigma \odot h$$

RNNs

- Performance
  - Significant progress in NLP but not a breakthrough.
  - Dominant in NLP for MT, QA, summarization, etc up to 2018.
- Limitation
  - RNNs cannot learn long-term dependencies (no more than 50 steps).
  - Hard to train because they are non-linear dynamical systems
    - Any small perturbation can amplify or vanish.
  - Slow to train because of their sequential nature (unlike CNNs).
    - Important limitation when training on Big Data.

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

Data Structure

- CNNs
  - Data on multi-dimensional grids
- RNNs
  - Data on ordered sequences
- Graph NNs
  - Data on graphs (not discussed in this module)
- What about data not supported by any grid?
  - Inputs are data with unknown relationships.
  - Data known as set of vectors (bags of features).
  - How to define neural networks for sets?

Neural Networks for Sets

- What invariances are necessary to design NNs for processing sets?
  - Permutation invariance (item indexing does not matter).
  - Size invariance (set representation must be independent of the set size).
- Naive solution
  - Pooling : Functions of the mean/sum/max operator for sets of data

$$\mathrm{NN}(\{x_0, ..., x_{n-1}\}) = f_W\left(\frac{1}{n}\sum_i x_i\right)$$

Mean

where vector $\mathbf{x}_i$ is a feature vector (continuous vector, result of discrete embedding or output of NN layer)

  - Bags of visual features (SIFT) work well, but not great.
  - Bags of NLP features (Word2Vec) work well, but not great.

Neural Networks for Sets

- Attention mechanism :
  - Focus on the most relevant data (weighted mean pooling)

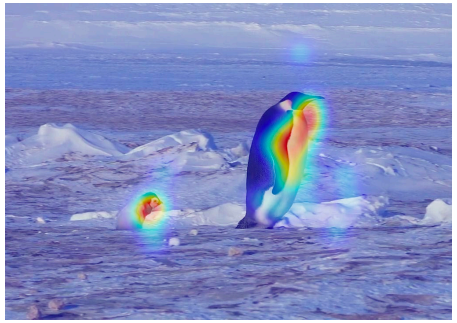
$$\mathrm{NN}(\{x_0, ..., x_{n-1}\}) = f_W\left(\frac{1}{n}\sum_i \underbrace{a(x_i, X\backslash x_i)}_{\text{Attention weights}} \cdot x_i\right)$$

Weighted mean

- Attention score $a_i = a(x_i, X\backslash x_i)$ is a probability distribution over all data X, which can change dynamically as a function of the data and the state of the system.
  - Score a_i can be binary (hard attention) or continuous (soft attention).
- Attention-based NNs or DeepSets (Zaheer-etal.'17 , Ilse-etal.'18).

Use of Attention

- Attention in CV :
  - Attention over grid



Visual attention

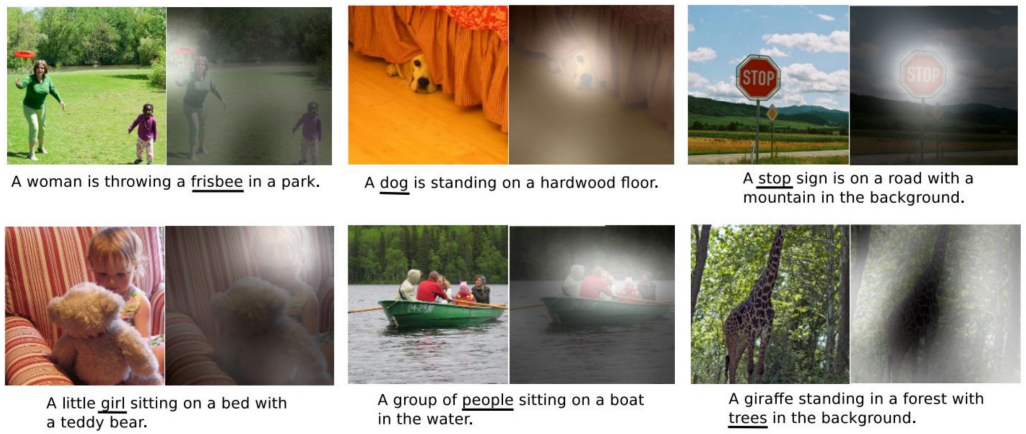
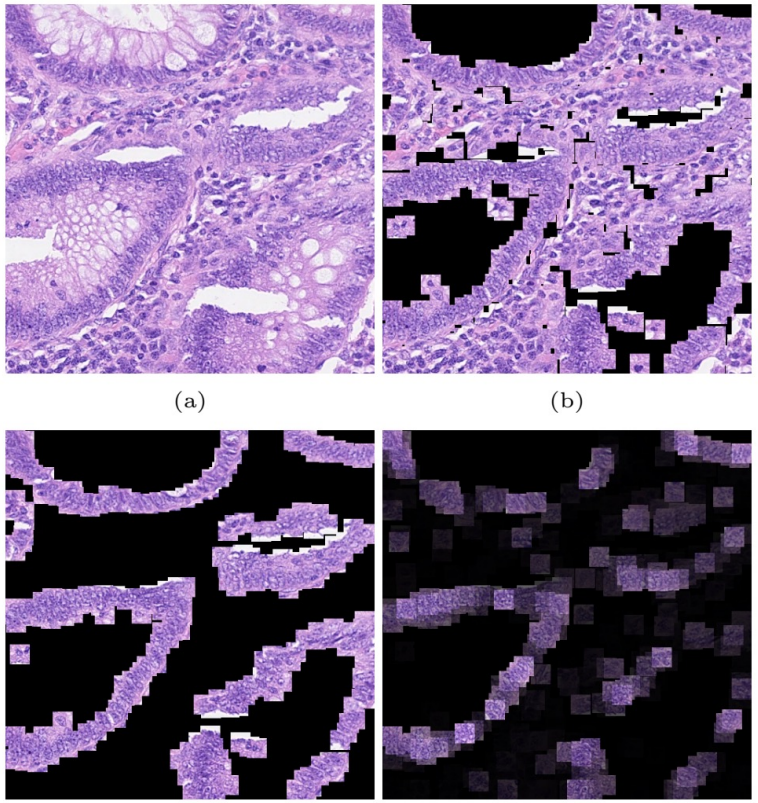


Image captioning

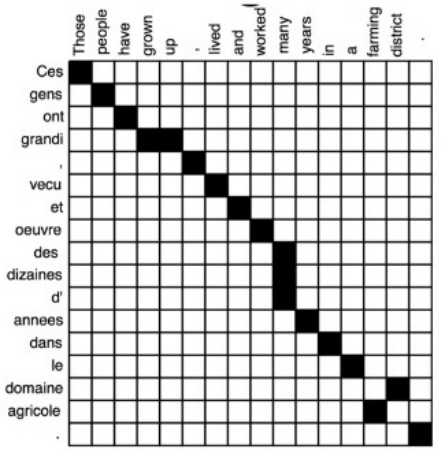


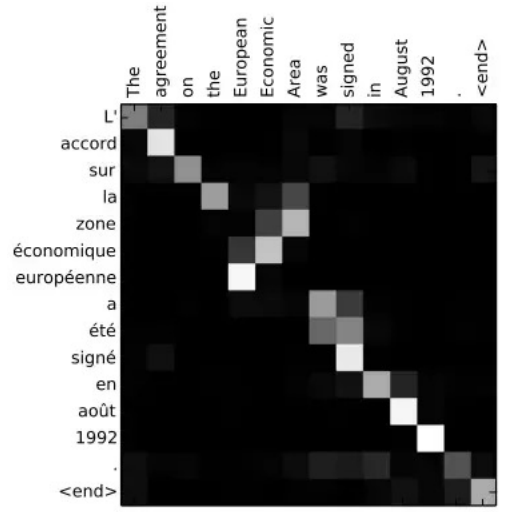
Ground truth patches that belong to the class epithelial.

Patches of (b) multiplied by its attention weights learned to make the classification decision.

Use of Attention

- Attention in NLP :
  - Alignment in Machine Translation (MT) :
    - Hard attention :
      - (Binary) alignment is a (difficult) combinatorial optimization problem.
    - Soft attention :
      - For each word in target sequence, get a probability over words in the source sequence [Brown-et-al'93].
      - Better approach as continuous relaxation of the combinatorial matching can be solved by backpropagation !



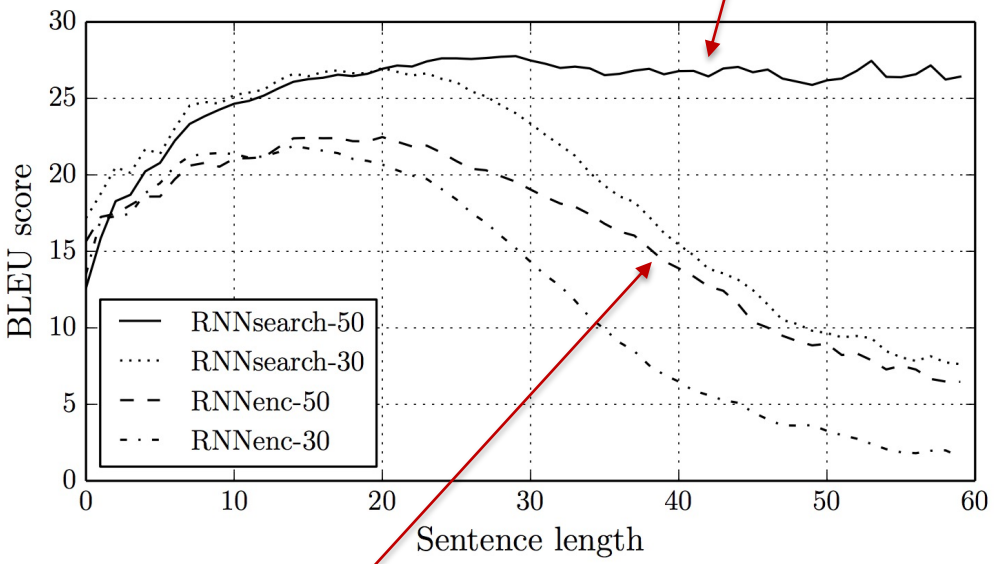


Use of Attention

- Attention in NLP :
  - First paper to do soft alignment with attention in MT is [Bahdanau, Cho, Bengio'14], followed by [Luong, Pham, Manning'15].
    - Improved SOTA in MT
    - Precursor of Transformers

- Breakthrough idea in NLP !
  - As revolutionary as CNNs in CV

RNNs with Attention
[Bahdanau, Cho, Bengio'14]



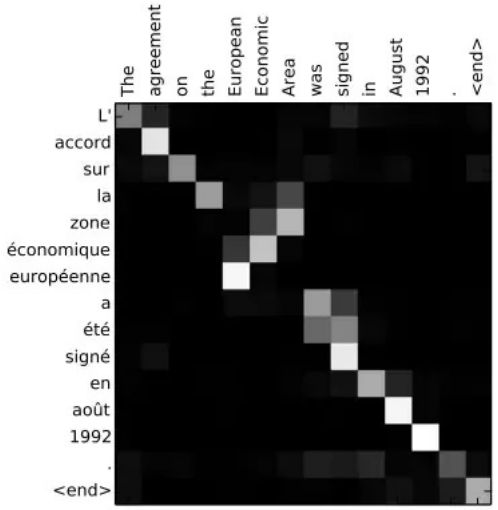
RNNs (No Attention)
(degrades quickly after 30 words)
[SOTA LSTM'14]

Attention in NLP

- Why casting MT as a soft alignment/attention problem is a great idea ?
  - With RNNs, the very long sequence requires to be memorized and represented by a single vector (that will be later decoded).
    - (The memory idea is good but) RNN architectures with a single memory vector cannot simply deal with long sequences (limit of non-linear dynamic systems).
    - We ask too much to memorize everything with one vector!
  - With attention, we distribute the memorization load over each word.
    - Each word in the target sequence only needs to find its match with the word (or a few words) in the source target.
      - It solves the limitation of long-term dependencies in RNNs (a word in the target sequence communicates with all words in the source sequence).
    - The matching is made easy by transforming the words with hidden representations.
    - Attention is a key mathematical structure/property for NLP!
      - SOTA for all NLP tasks since 2019.

Attention in NLP

- Illustrations



  - Example 1 : Machine Translation
    - Easy to match source-target words.

  - Example 2 : Sentiment Analysis
    - Easy to focus on relevant words.
      - Is the book's review good ?
        - Steve is arrogant but his book is awesome.

Focus on this positive word,
and ignore everything else !

Outline

- Neural Networks
- Neural Networks for Sets
- **Memory Networks**
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

Memory Networks

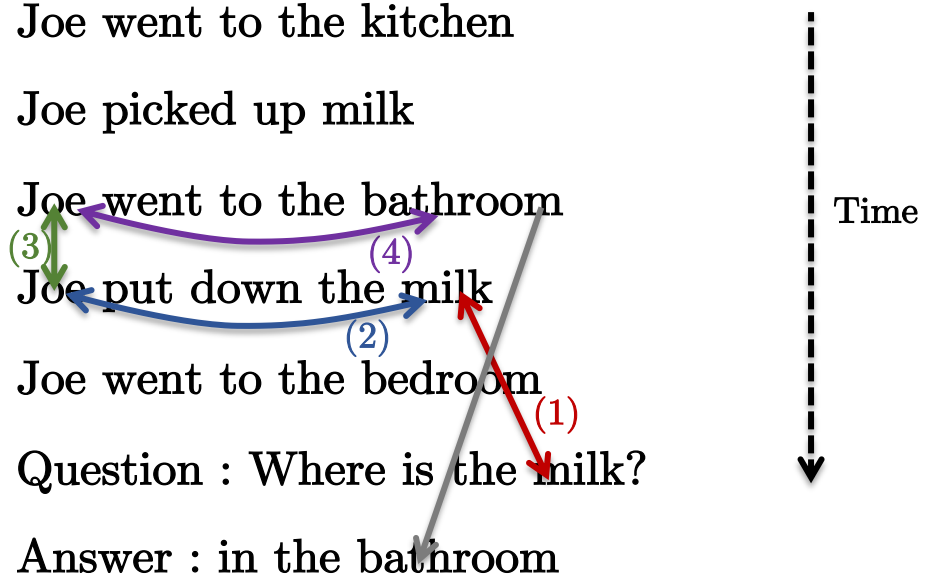
- [Bahdanau, Cho, Bengio'14] has a single-hop attention mechanism (1 layer of attention).
- Memory networks
  - Differentiable Memory Computers
  - Precursors of Transformers
  - Based on the principle that intelligence requires an adaptive long-term memory.
  - [Weston, Chopra, Bordes, Memory networks'14], [Graves, Wayne, Danihelka, Neural turing machines'14] introduced the idea of
    - Multiple-hop attention – stacking attention layers
    - Network keeps updated its memory to perform multi-step reasoning.

Memory Networks

- How do they work?
- An example with the bAbI dataset (Facebook Research)
  - A dataset to test reasoning on simple stories.
  - Example :
    - Joe went to the kitchen
    - Joe picked up milk
    - Joe went to the bathroom
    - Joe put down the milk
    - Joe went to the bedroom
    - Question : Where is the milk?
    - Answer : ----

Memory Networks

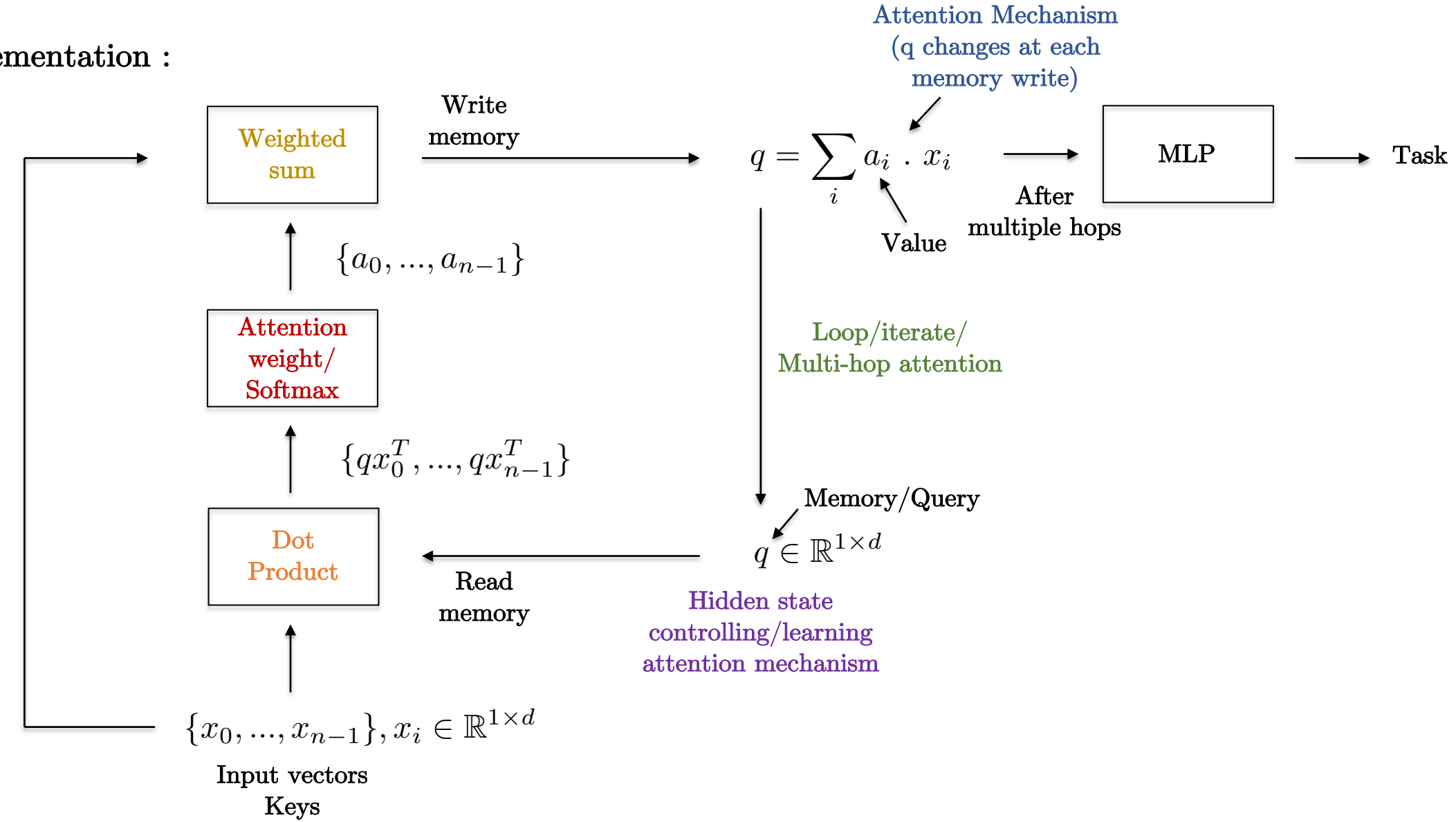
- Attention networks will solve the problem with multi-step word matching :
  - Joe went to the kitchen
  - Joe picked up milk
  - Joe went to the bathroom
  - Joe put down the milk
  - Joe went to the bedroom
  - Question : Where is the milk?
  - Answer : in the bathroom



- Multi-hop attention is the mechanism that is designed to do this multi-step reasoning process !
- Memory networks are fully trainable end-to-end systems [Sukhbaatar, Szlam, Weston, Fergus'15].

Memory Networks

- Implementation :



Memory Networks

- Input word set : $\{w_0, ..., w_{n-1}\}$
- Continuous representation : $\{x_0, ..., x_{n-1}\},\ x_i = f_E(w_i)$
  (world embedding)
- Define :

$$K = \begin{bmatrix} f_K(x_0) \\ \vdots \\ f_K(x_{n-1}) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

$$q \in \mathbb{R}^{1 \times d}$$

$$V = \begin{bmatrix} f_V(x_0) \\ \vdots \\ f_V(x_{n-1}) \end{bmatrix} \in \mathbb{R}^{n \times d}$$

- Repeat K hops :

$$a \leftarrow \text{Softmax}(qK^T) \in \mathbb{R}^{1 \times n}$$
$$q \leftarrow aV = \text{Softmax}(qK^T)V \in \mathbb{R}^{1 \times d}$$

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

Beyond Memory Networks

- Memory networks were promising, but not ground-breaking.
- Transformers [Vaswani-et-al'17] design the first efficient version of attention networks !
  - Ground-breaking architecture in NLP
  - Best NN architecture not only for NLP but for problems with sets in general.
  - Transformer improvements over Memory networks :
    - Multiple hidden states (one per word)
    - Multi-head attention mechanism (more learning capacity)
    - Residual blocks (better backpropagation)

Transformers

- Multiple hidden states
  - Memory nets have one single hidden state q for all inputs x_i.
  - Transformers have one hidden state h_i for any input x_i.

Vanilla Transformers

- Input set (continuous representation) :

$$\{x_0, ..., x_{n-1}\}, \quad X = \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix} \in \mathbb{R}^{n \times d}$$

- Initiate hidden state :

$$H = X \in \mathbb{R}^{n \times d}$$

- Repeat K hops :

$$\begin{aligned} A &\leftarrow \text{Softmax}(QK^T) \in \mathbb{R}^{n \times n} \\ H &\leftarrow AV = \text{Softmax}(QK^T)V \in \mathbb{R}^{n \times d} \\ &\text{with } Q = K = V = H \end{aligned}$$

Self-attention !

Transformers

- Input set (continuous representation) :

$$\{x_0, ..., x_{n-1}\}, \quad X = \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix} \in \mathbb{R}^{n \times d}$$

- Initiate hidden state :

$$H = X \in \mathbb{R}^{n \times d}$$

- Repeat K hops :

$$H \leftarrow \text{Softmax}(QK^T)V \in \mathbb{R}^{n \times d}$$

with $Q = HW^Q \in \mathbb{R}^{n \times d}$, $W^Q \in \mathbb{R}^{d \times d}$

$K = HW^K \in \mathbb{R}^{n \times d}$, $W^K \in \mathbb{R}^{d \times d}$

$V = HW^V \in \mathbb{R}^{n \times d}$, $W^V \in \mathbb{R}^{d \times d}$

Differentiable dictionary

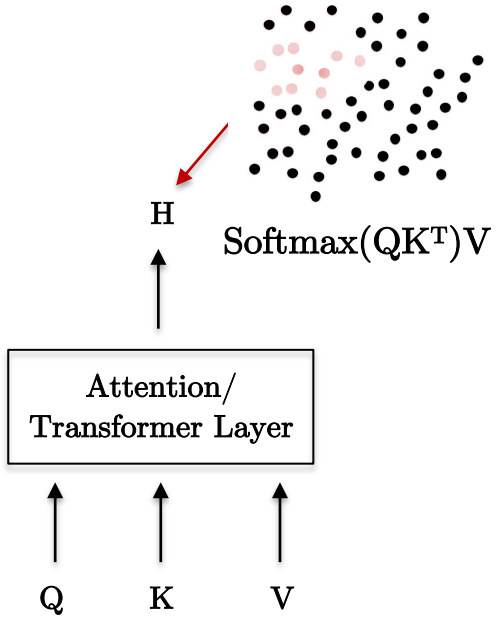
Dict is a standard structure in CS

Dict=(Query,Key,Value)

Context-To-Word Representation

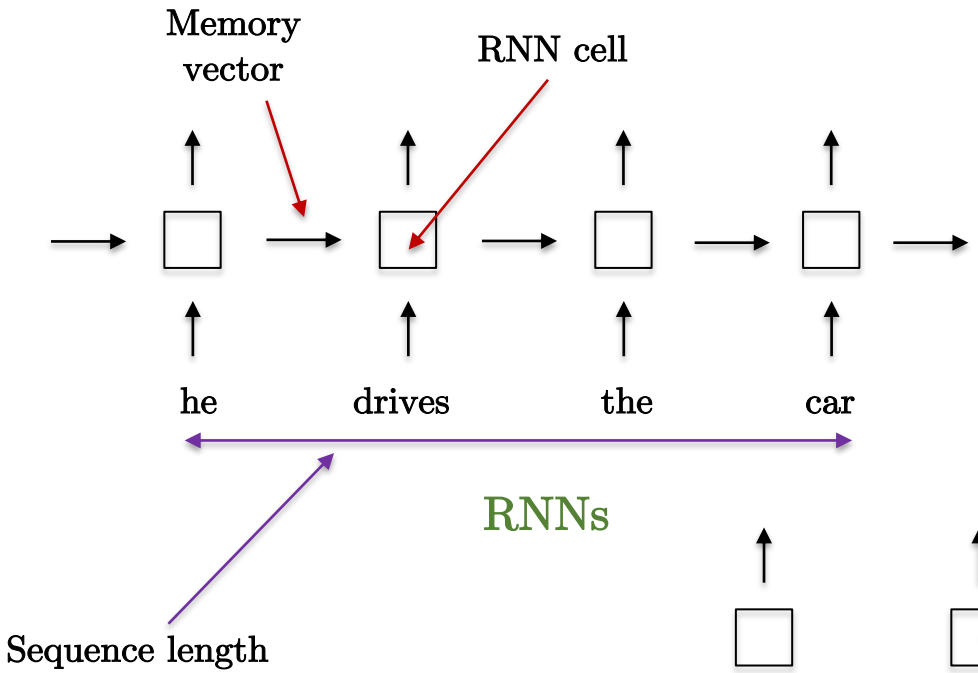
- From Word representation to Context-to-Word representation :

$$H \leftarrow \text{Softmax}(QK^T)V \in \mathbb{R}^{n \times d}$$

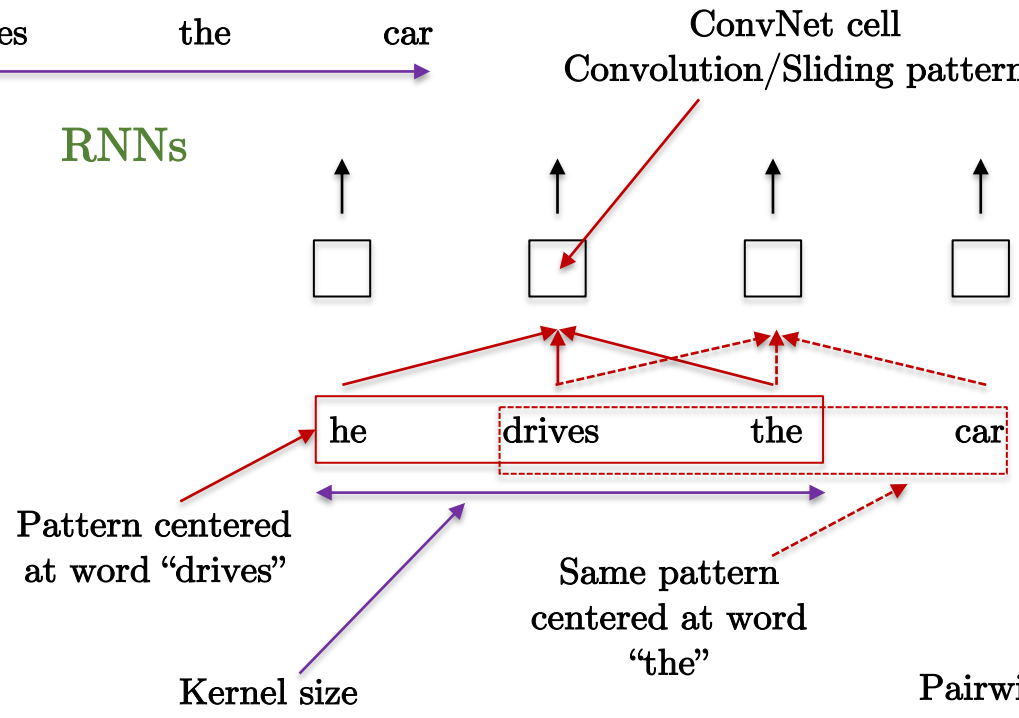


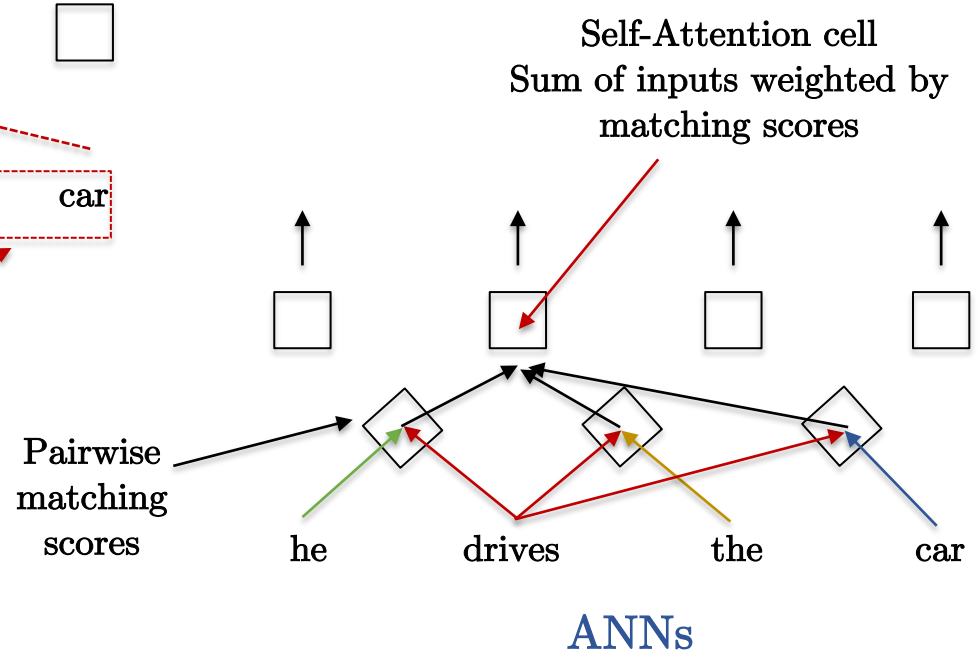
- The new data representation is a sum of all input data weighted by the pairwise matching (or attention) scores.
- The subset of data with non-zero attention scores forms the context.
- The attention mechanism allows to dynamically change the word representation according to its context.
- Context-to-Word is a powerful idea in NLP because a word may have different meanings, that can only be clarified in a particular context :
  - The vase broke. The news broke. Sandy broke the world record. Sandy broke the law. We broke even. The burglar broke into the house. Etc.

Computational Cost



n : sequence length
d : hidden feature size
k : kernel size





Computational Cost

- RNN layer : $O(n.d^2)$
- ConvNet layer : $O(n.d^2.k)$
- Transformer layer : $O(n^2.d)$

Seems bad !

with n : sequence length, d : hidden feature size, k : kernel size

- Attention networks have actually less parameters as long as $n \leq d$!
  - Example 1 : n=100, d=1000, k=3

    RNN: $O(10^8)$, ConvNet: $O(3.10^8)$, Transformer: $O(10^7)$
  - Example 2 : n=1000, d=1000, k=3

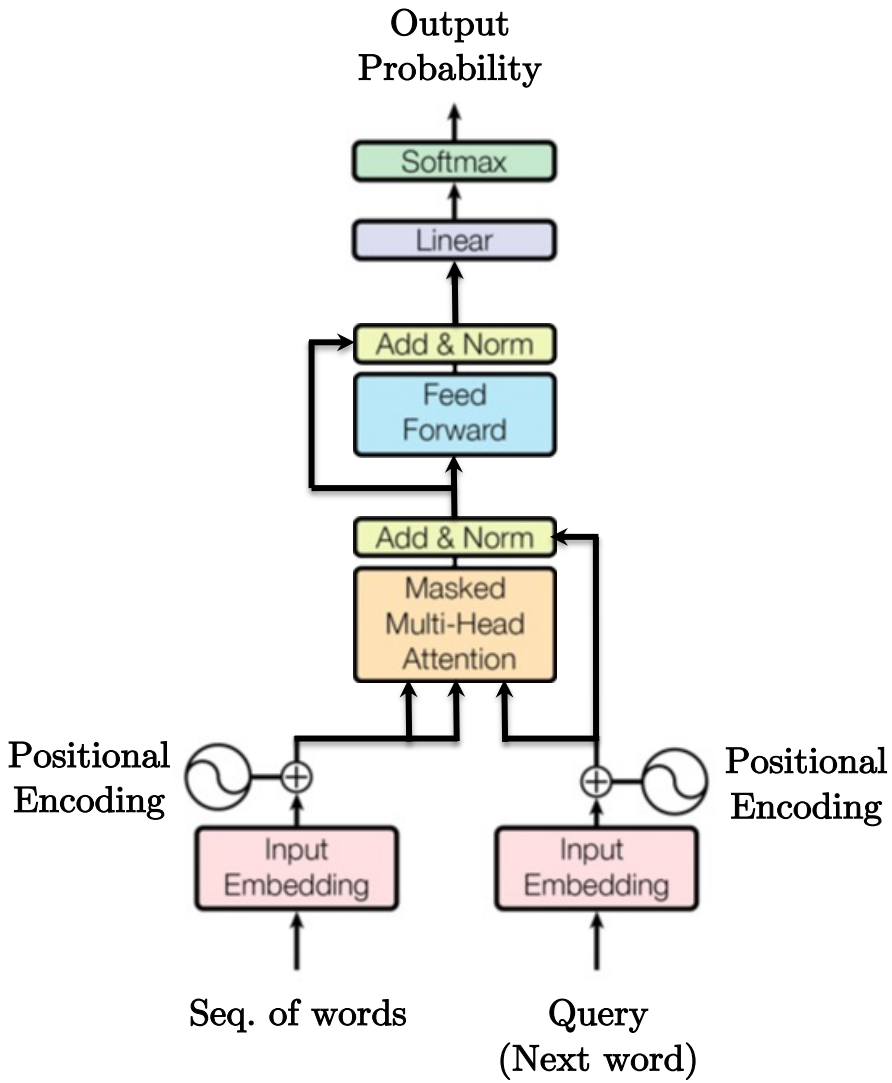
    RNN: $O(10^9)$, ConvNet: $O(3.10^9)$, Transformer: $O(10^9)$

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- **Language Model Transformers**
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

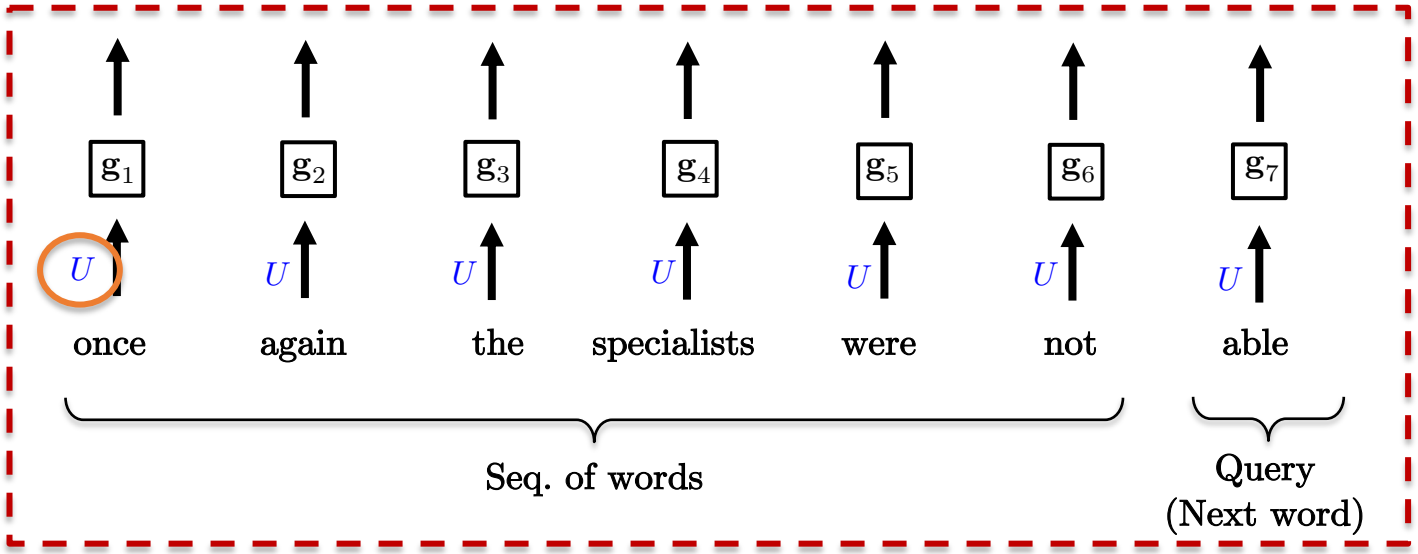
Language Model Transformers

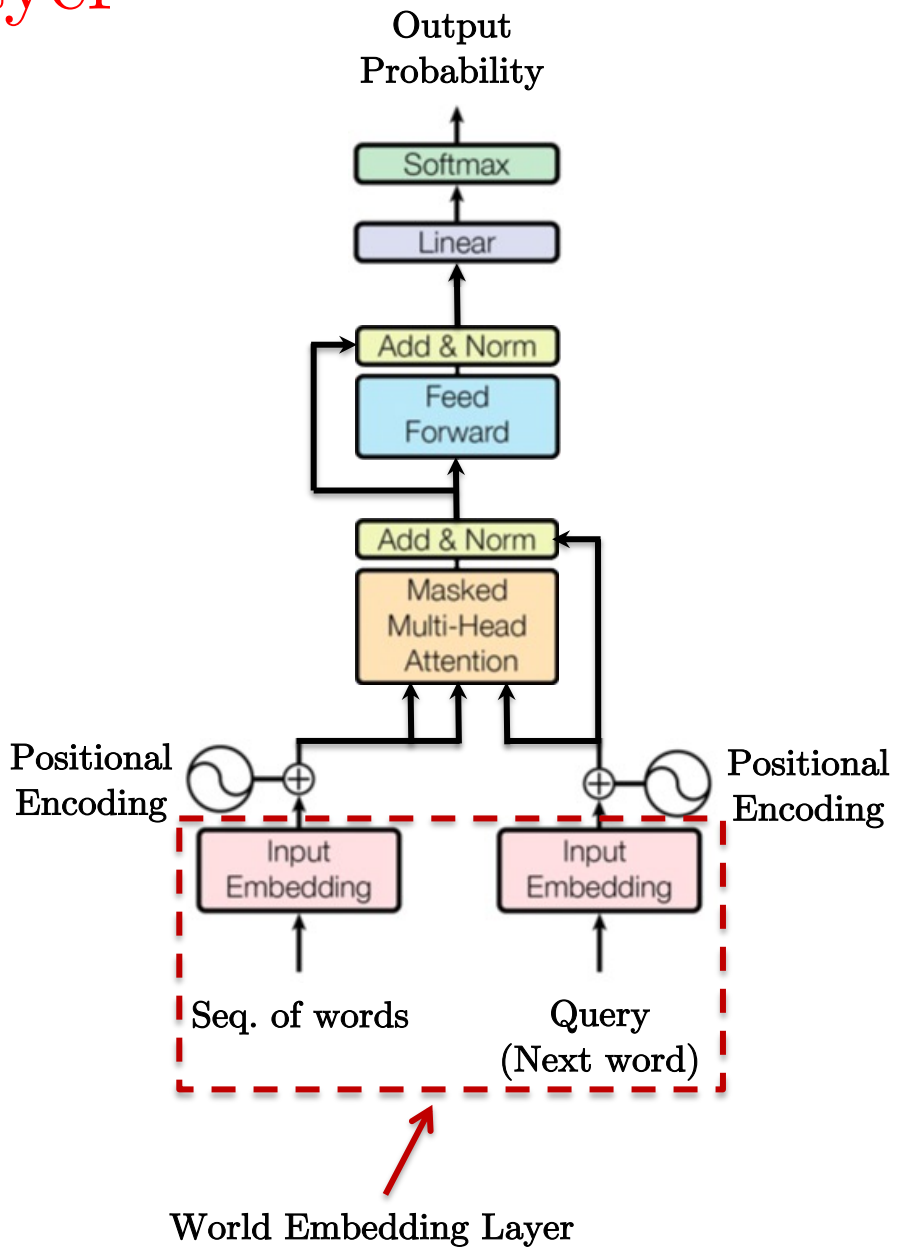
- Given a sequence of words, predict the next word.
- Fundamental task in NLP
  - Requires a world representation that can be flexible/changed in different contexts.
  - Transformers improved word representation to word-in-context representation.
- A LM Transformer is composed of three layers :
  - Word embedding layer
  - Attention layer
  - Classification layer



Word Embedding Layer

- Categorical variables (dictionary of 10,000 words) are
  - represented by one-hot vectors and then
  - embedded in a linear space.
- This is the same input embedding as in RNNs.
  - PyTorch nn.Embedding()





Multi-Head Attention (MHA)

- MHA update equation
  - PyTorch nn.MultiheadAttention()

$$h = \text{MHA}(q, K, V) \in \mathbb{R}^d,\ q \in \mathbb{R}^d, K \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{n \times d}$$

$$= \Big(\|_{h=1}^{H}\ \text{HA}_h(q, K, V) \Big) W^O,\ W^O \in \mathbb{R}^{d \times d}$$

Concatenation operation

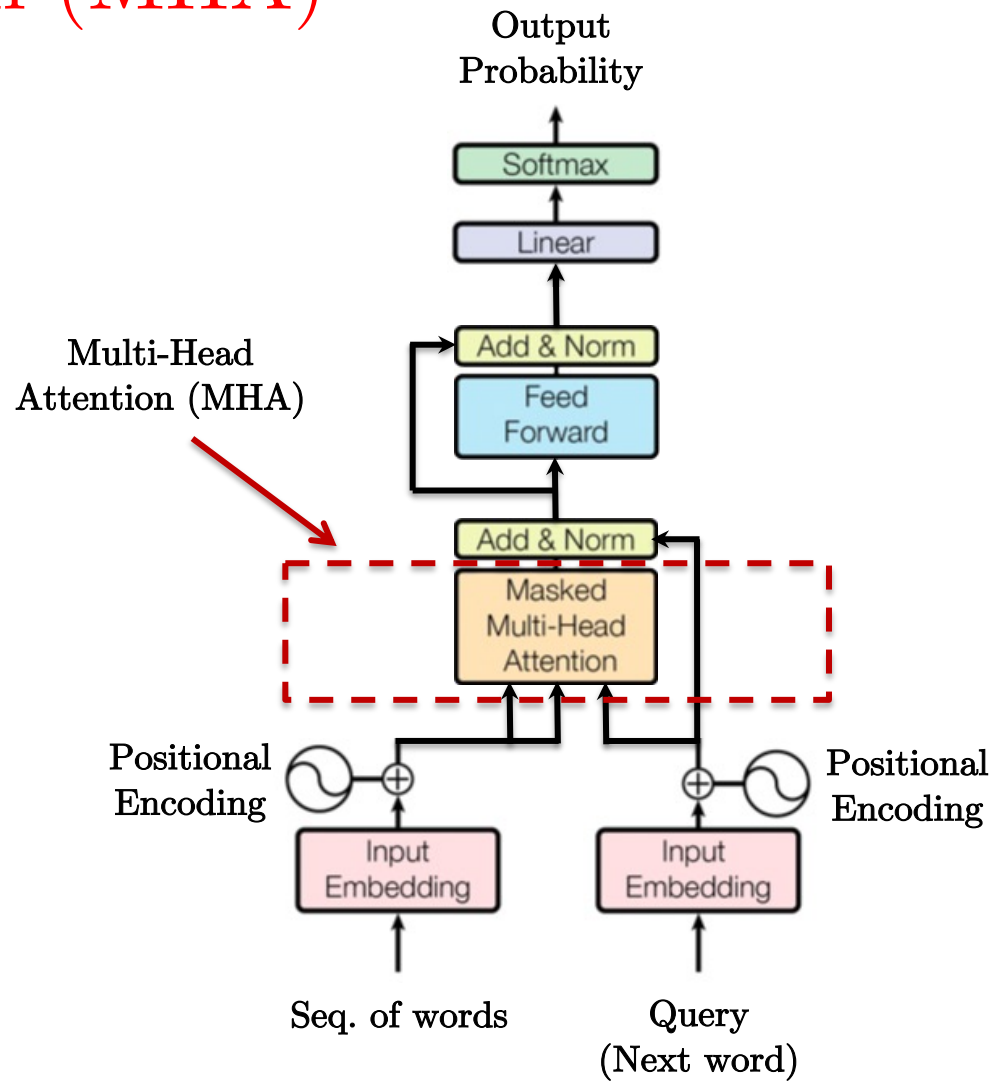
$$\text{with } q = \text{g}_t \in \mathbb{R}^d,\ K = V = \{\text{g}_{t-1}, ..., \text{g}_{t-n}\} \in \mathbb{R}^{n \times d}$$

$$\text{HA}_h(q, K, V) = \text{Softmax}\Big(\frac{q_h K_h^T}{\sqrt{d/H}}\Big) V_h \in \mathbb{R}^{d/H}$$

$$\text{with } q_h = q W_h^q \in \mathbb{R}^{d/H}, W_h^q \in \mathbb{R}^{d \times d/H}$$

$$K_h = K W_h^K \in \mathbb{R}^{n \times d/H}, W_h^K \in \mathbb{R}^{d \times d/H}$$

$$V_h = V W_h^V \in \mathbb{R}^{n \times d/H}, W_h^V \in \mathbb{R}^{d \times d/H}$$

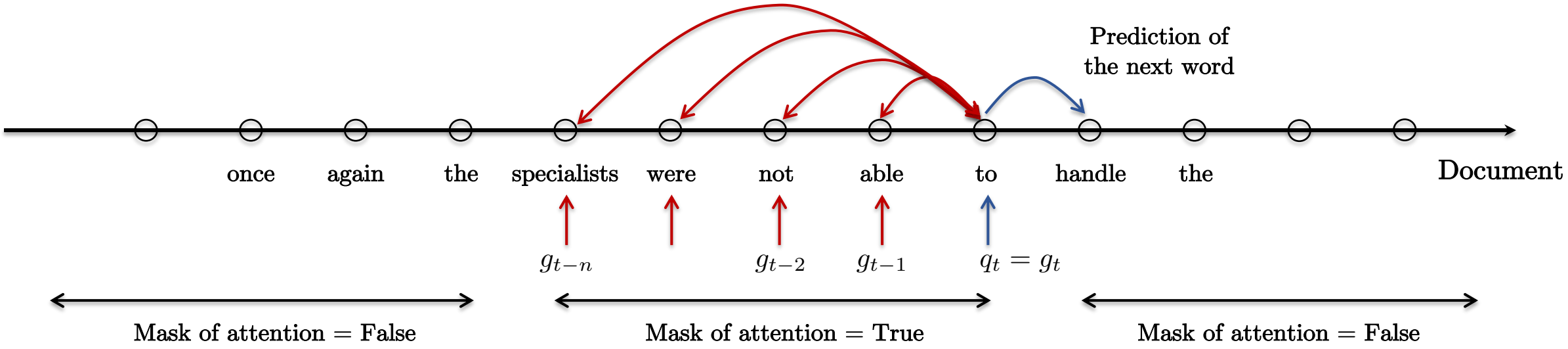


Masked MHA

- Masked MHA equation :

$$\mathrm{HA}(q, K, V) = \mathrm{Mask} \odot \mathrm{Softmax}\left(\frac{qK^T}{\sqrt{d}}\right)V$$

$$\text{with } \mathrm{Mask}_{ij} = \begin{cases} 1 \text{ if attention between } i \text{ and } j \\ 0 \text{ if no attention} \end{cases}$$



Attention Layer

- Attention layer update equation :

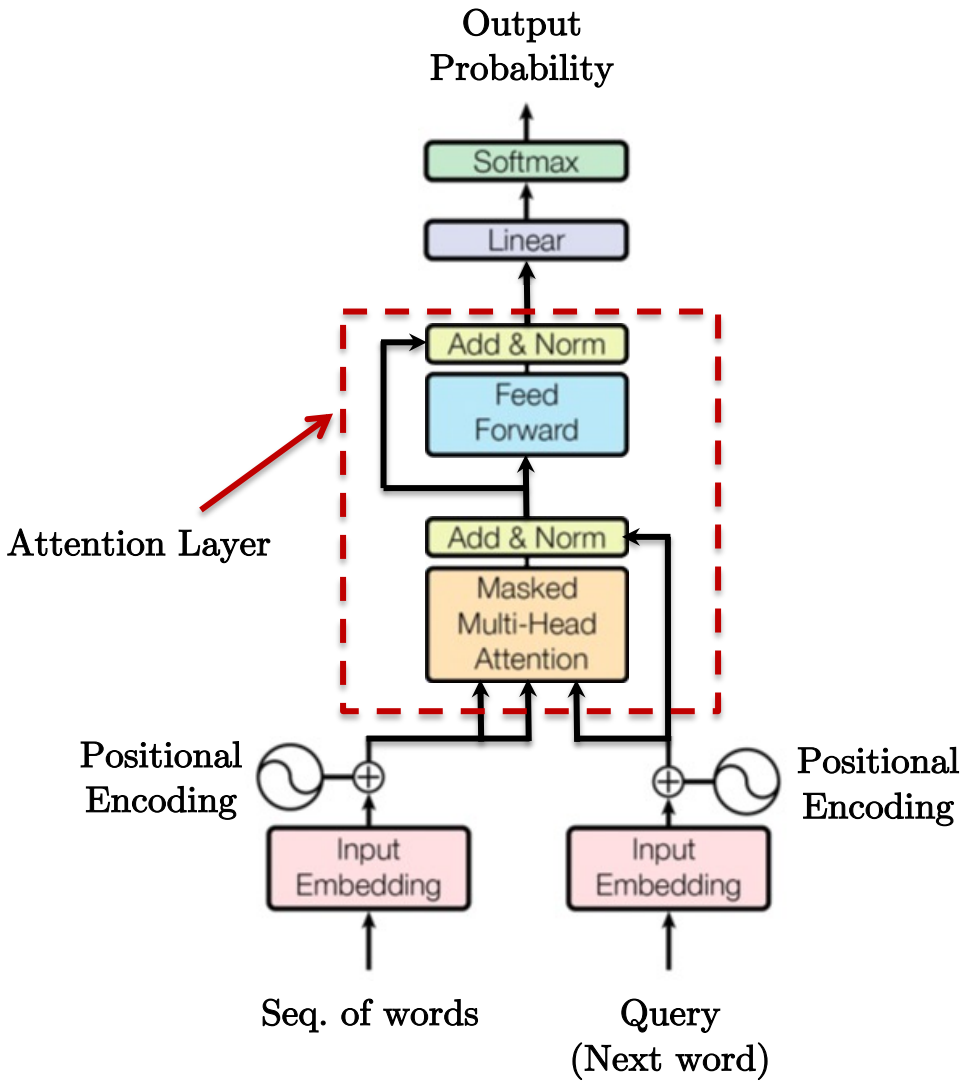
$$\text{for } \ell = 0, ..., L-1$$

$$\bar{q}^{\ell+1} = \text{LN}\big(q^{\ell} + \text{MHA}(q^{\ell}, K^{\ell}, V^{\ell})\big) \in \mathbb{R}^d$$

$$q^{\ell+1} = \text{LN}\big(\bar{q}^{\ell+1} + \text{MLP}(\bar{q}^{\ell+1})\big) \in \mathbb{R}^d$$

Residual connection

Layer normalization
z-scoring with learnable parameters
nn.LayerNorm()



Attention Layer

- Layer normalization :

z-scoring

$$\mathrm{LN}(q) = a \odot \left(\frac{q-\mu}{\sigma}\right) + b \in \mathbb{R}^d$$

$$\text{with } \mu = \mathrm{Mean}(q) \in \mathbb{R}, \sigma = \mathrm{Std}(q) \in \mathbb{R}, a, b \in \mathbb{R}^d$$

- Residual/skip connection :

$$x \longrightarrow \boxed{\cdot + f_W(\cdot)} \longrightarrow y = x + f_W(x)$$

Forward pass

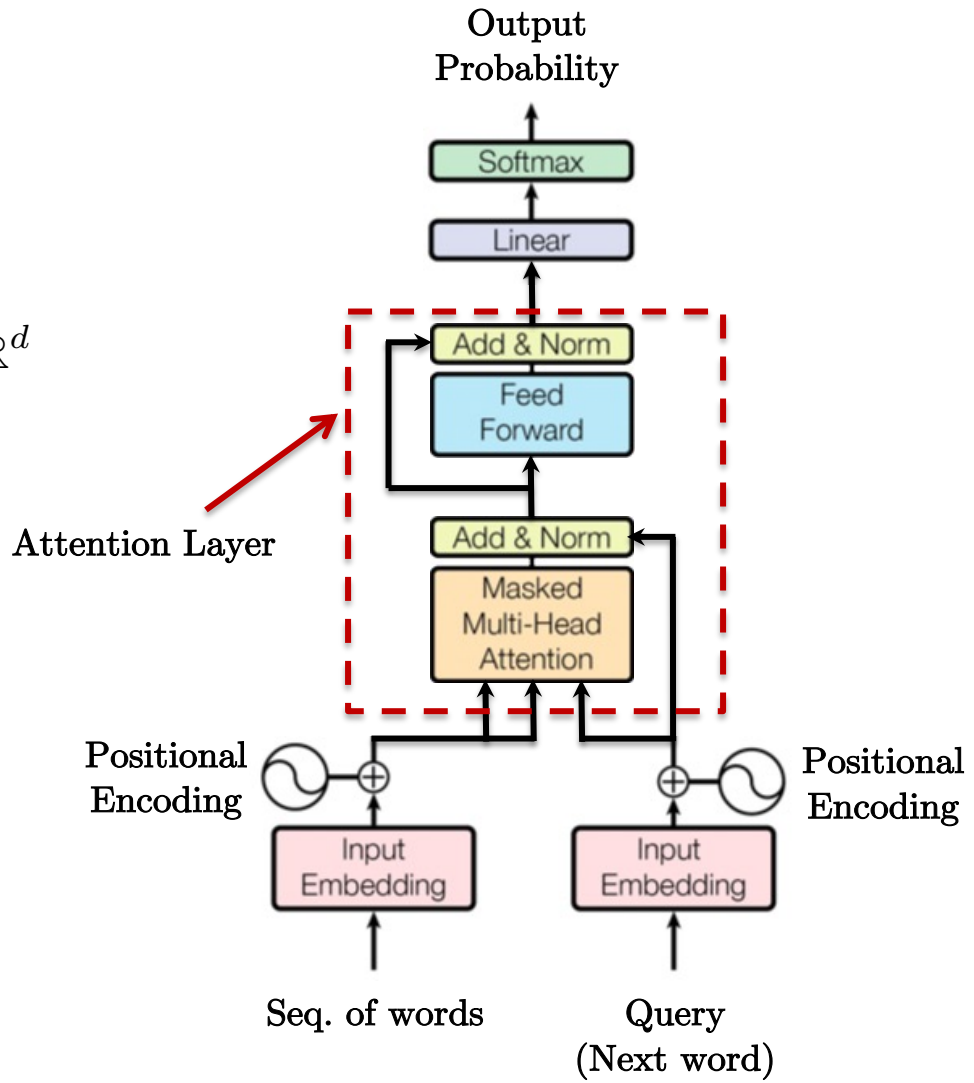
$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} \longleftarrow \boxed{\cdot + f_W(\cdot)} \longleftarrow \frac{\partial L}{\partial y}$$

Backward pass

$$= \frac{\partial L}{\partial y}\left(\mathrm{Id} + \nabla_x f_w\right)$$

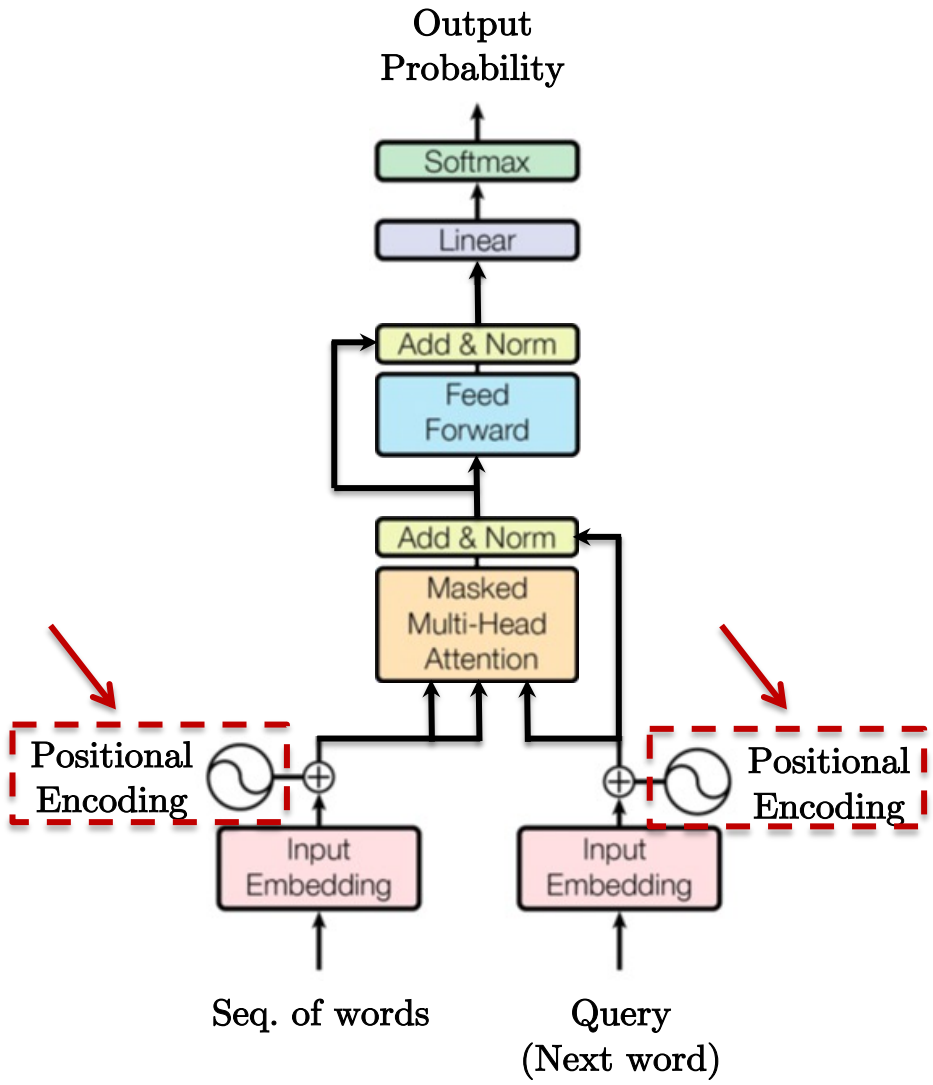
$$= \frac{\partial L}{\partial y} + \frac{\partial L}{\partial y}\nabla_x f_w$$

No vanishing gradient
for residual connection



Positional Encoding

- ANNs like Transformers are designed to process sets but items in a set are not ordered.
- This is an issue for NLP tasks.
  - An additional ordering feature is required to inject temporal information in the attention network.
- Two classes of Positional Encoding (PE) :
  - Learnable vs non-learnable PE
- Learnable PE :
  - Embedding of discrete ordering index 0,1,2,3,...,L-1, with L is the sequence length.
  - Two issues :
    - Requires to know the maximum L value among all training sequences.
    - Some test sequences may have lengths not present in the train set.

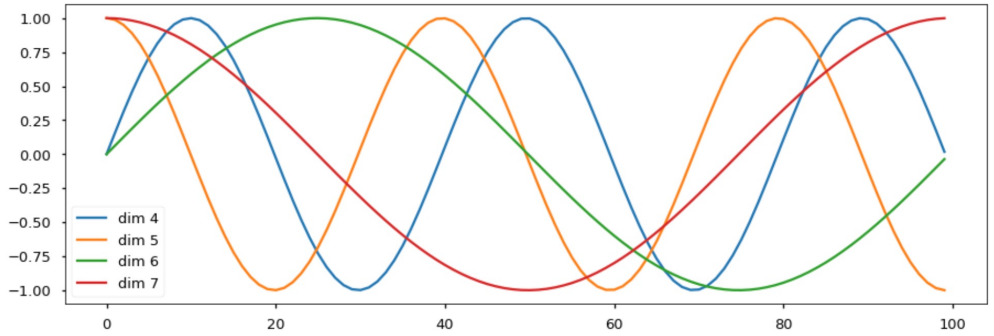


Positional Encoding

- Non-learnable PE :
  - Continuous ordering with sin and cos functions.
  - Advantages :
    - No training necessary
    - No need to know the maximum length in the train set.
    - Test sequences may have lengths not present in the train set.

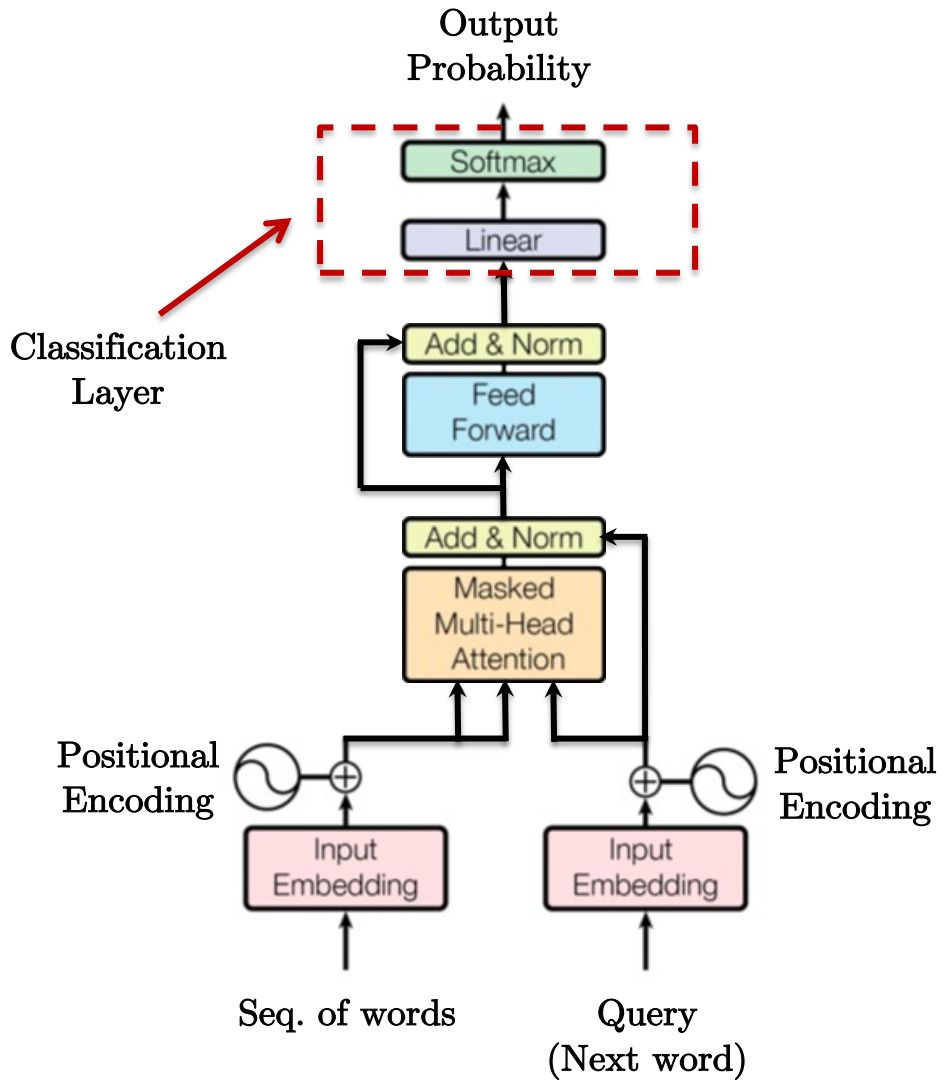
$\text{PE}_t \in \mathbb{R}^d$ is defined as

$$\text{PE}_{t,i} = \begin{cases} \sin(2\pi f_i t) \text{ if } i \text{ is even,} \\ \cos(2\pi f_i t) \text{ if } i \text{ is odd,} \end{cases} \quad \text{with } f_i = \frac{10,000^{\frac{d}{\lfloor 2i \rfloor}}}{2\pi}.$$



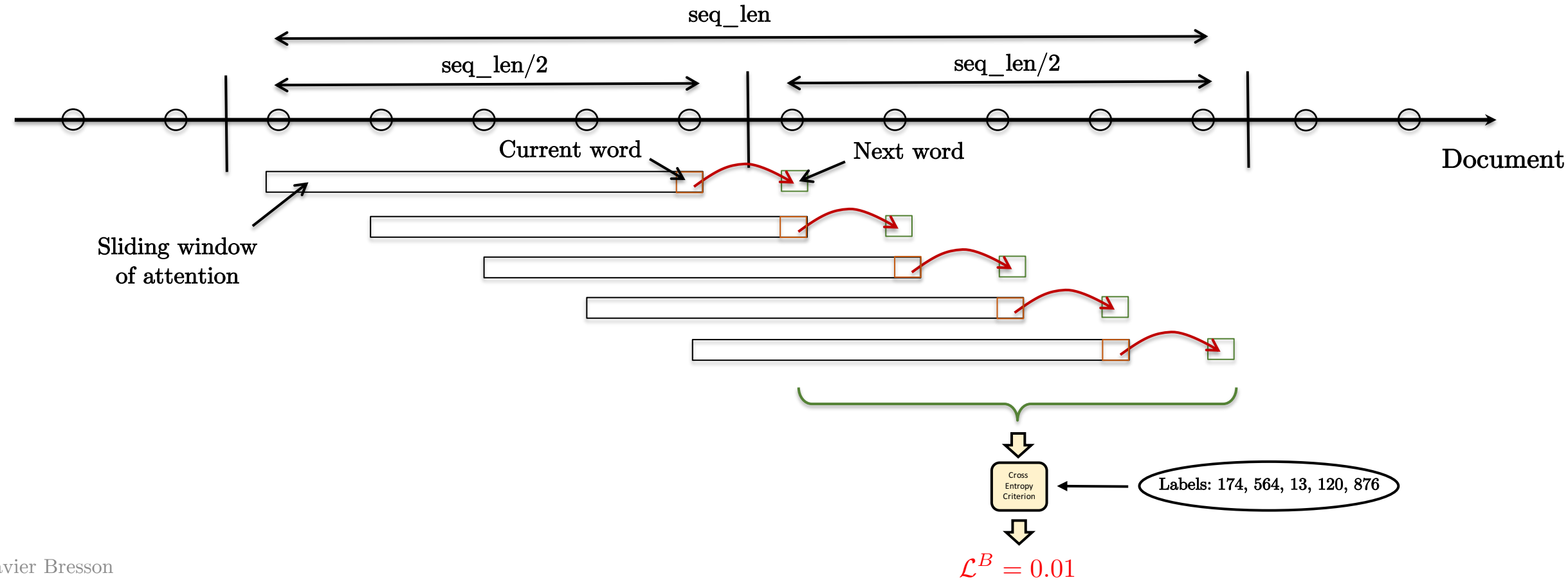
Classification Layer

- The last layer is a standard linear layer to generate the scores of the next word in the sequence, followed by a Softmax function to produce the probability vector.



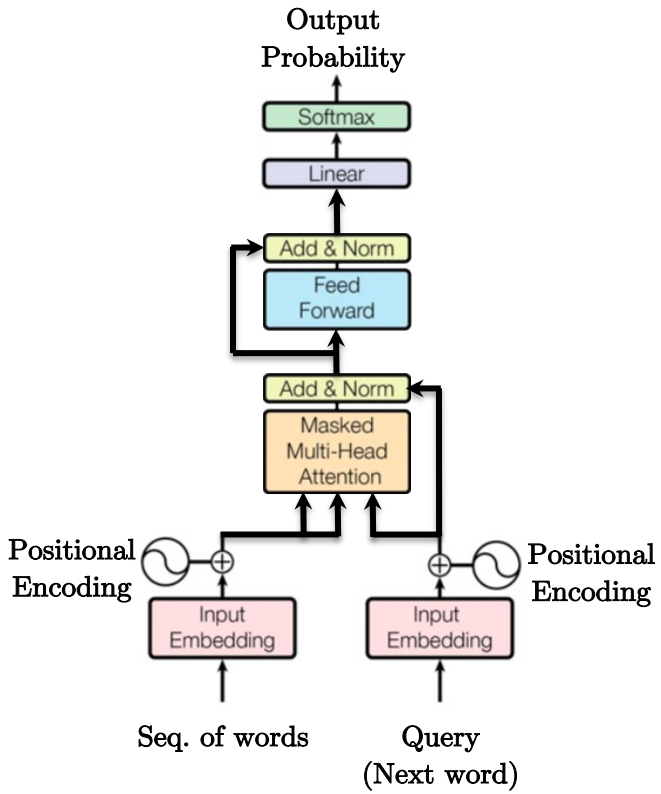
Training

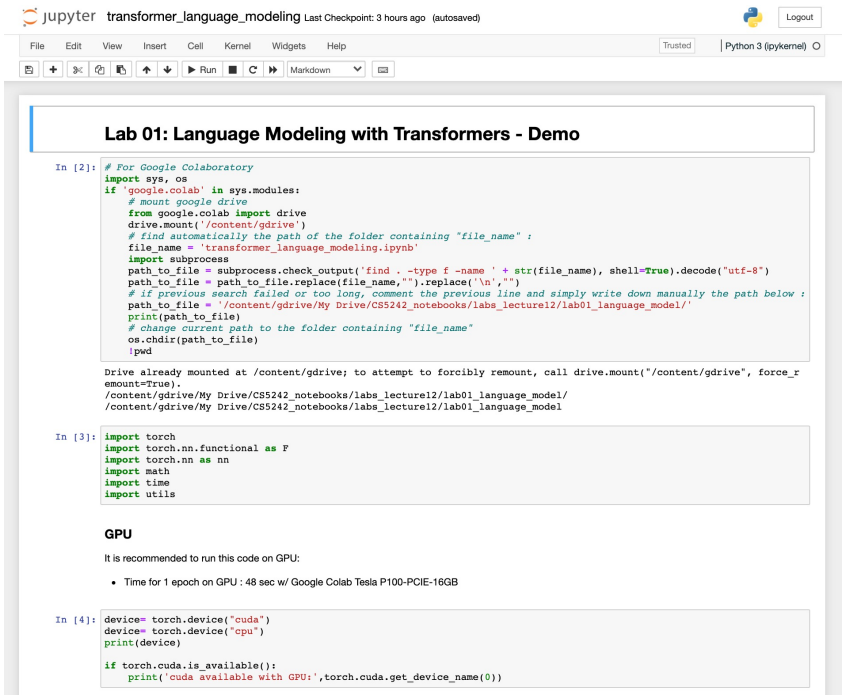
- Training is done with a fixed-size and sliding window of attention.
  - If train_data size is of size (seq_len x batch_size) then the size of the window of attention is seq_len/2.



Lab 01

- PyTorch implementation of Language Model Transformers





Lab 01

- Numerical results on PTB :
  - Vanilla RNN: $\exp(\text{train\_loss}) = 111$ $\exp(\text{test\_loss}) = 155$ 3 million parameters
  - LSTM: $\exp(\text{train\_loss}) = 59$ $\exp(\text{test\_loss}) = 106$ 7 million parameters
  - LSTM state-of-the-art: $\exp(\text{test\_loss}) = 50$ 50 million parameters
  - Vanilla LM Transformer:

    $\exp(\text{train\_loss}) = 54$ $\exp(\text{test\_loss}) = 174$ 3 million parameters
  - LM Transformer state-of-the-art (with pre-training):

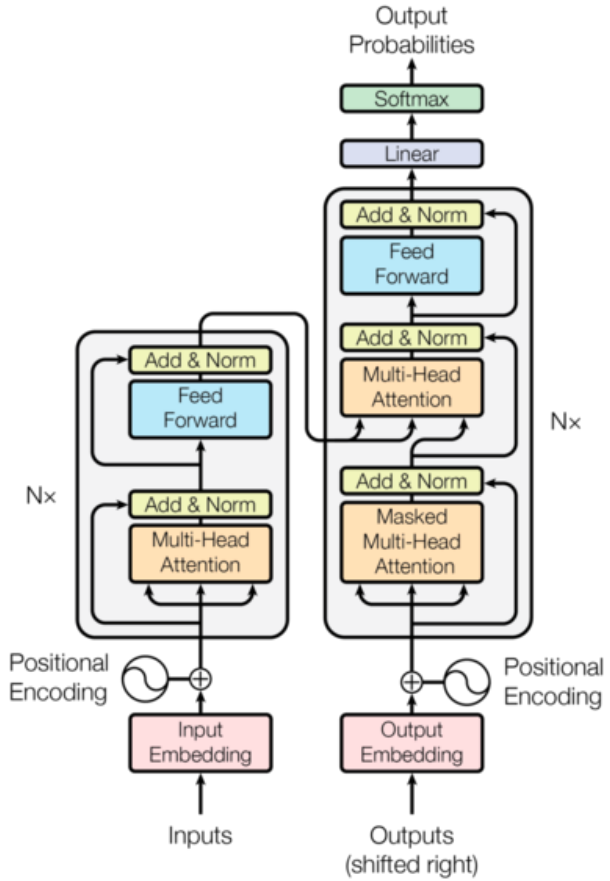
    $\exp(\text{test\_loss}) = 20.5$ 175,000 million parameters

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- **Sequence-To-Sequence Transformers**
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- Conclusion

Seq2Seq Transformers

- Given a sequence of words, convert it into a different sequence.
- Basic task in NLP
  - Translation
  - Question & Answer
  - Summarization
- A Seq2Sep Transformer is composed of
  - World embedding layer
  - Self-Attention encoding layer
  - Self-Attention decoding layer
  - Cross-Attention decoding layer
  - Classification layer



Input(EN): He drives the car

Output(FR): Il conduit la voiture

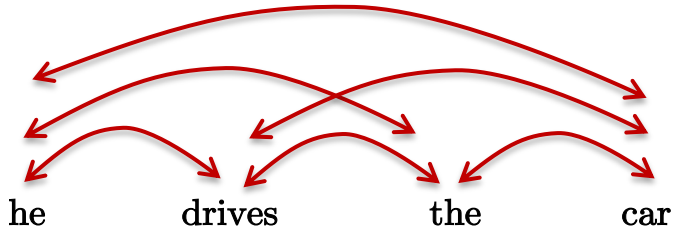
Seq2Seq Transformer Architecture
Vaswani-etal, Attention is all you need, 2017

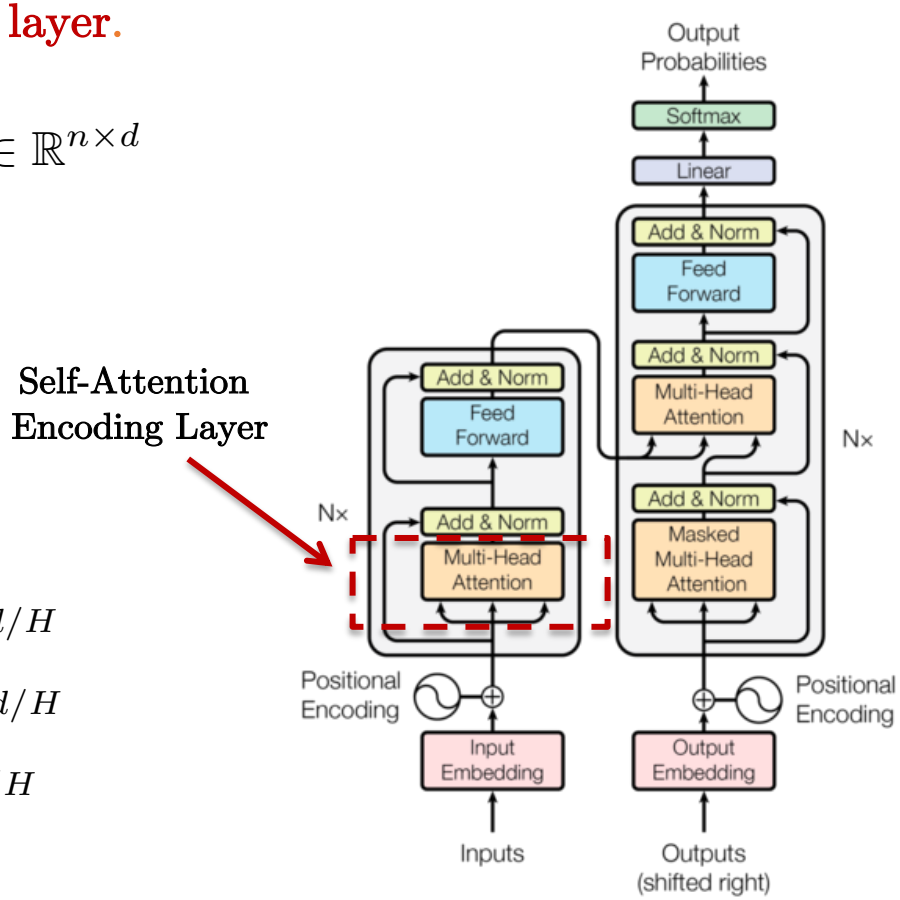
Self-Attention Encoding Layer

- Encode the whole input sequence with a self-attention layer.

$$H = \text{MHA}(Q, K, V) \in \mathbb{R}^{n \times d},\ Q \in \mathbb{R}^{n \times d}, K \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{n \times d}$$
$$= \Big(\|_{h=1}^{H} \text{HA}_h(Q, K, V) \Big) W^O,\ W^O \in \mathbb{R}^{d \times d}$$
$$\text{with } Q = K = V = \{\underbrace{\text{g}_1^{\text{in}}, ..., \text{g}_n^{\text{in}}}_{\text{Word embedding}}\} \in \mathbb{R}^{n \times d}$$

$$\text{HA}_h(Q, K, V) = \text{Softmax}\Big(\frac{Q_h K_h^T}{\sqrt{d/H}}\Big) V_h \in \mathbb{R}^{n \times d/H}$$
$$\text{with } Q_h = Q W_h^Q \in \mathbb{R}^{n \times d/H}, W_h^Q \in \mathbb{R}^{d \times d/H}$$
$$K_h = K W_h^K \in \mathbb{R}^{n \times d/H}, W_h^K \in \mathbb{R}^{d \times d/H}$$
$$V_h = V W_h^V \in \mathbb{R}^{n \times d/H}, W_h^V \in \mathbb{R}^{d \times d/H}$$





Encoder

- Encoder layer is composed of MHA, MLP, Residual Connection, Layer Normalization, PE, and possibly stacking multiple layers.

$$\text{for } \ell = 0, ..., L^{\text{Enc}} - 1$$

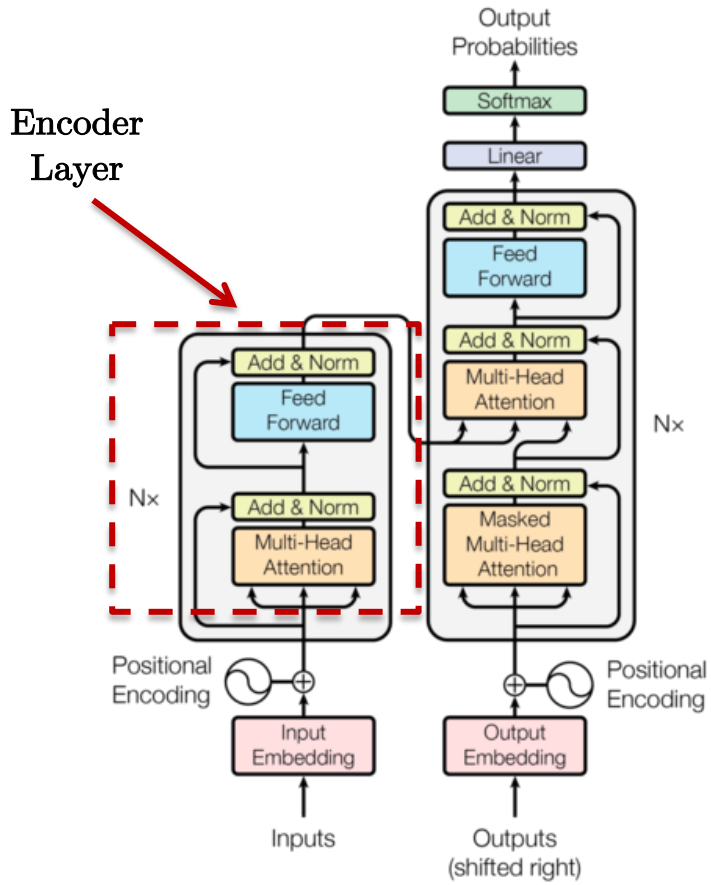
$$\bar{H}^{\ell+1} = \text{LN}\left(H^{\ell} + \text{MHA}(H^{\ell}, H^{\ell}, H^{\ell})\right) \in \mathbb{R}^{n \times d}$$

$$H^{\ell+1} = \text{LN}\left(\bar{H}^{\ell+1} + \text{MLP}(\bar{H}^{\ell+1})\right) \in \mathbb{R}^{n \times d}$$

$$\text{with initialization } H^{\ell=0} = \text{LL}(\{\text{g}_1^{\text{in}}, ..., \text{g}_n^{\text{in}}\} + \text{PE}) \in \mathbb{R}^{n \times d}$$

Linear layer (linear embedding)

Positional Encoding



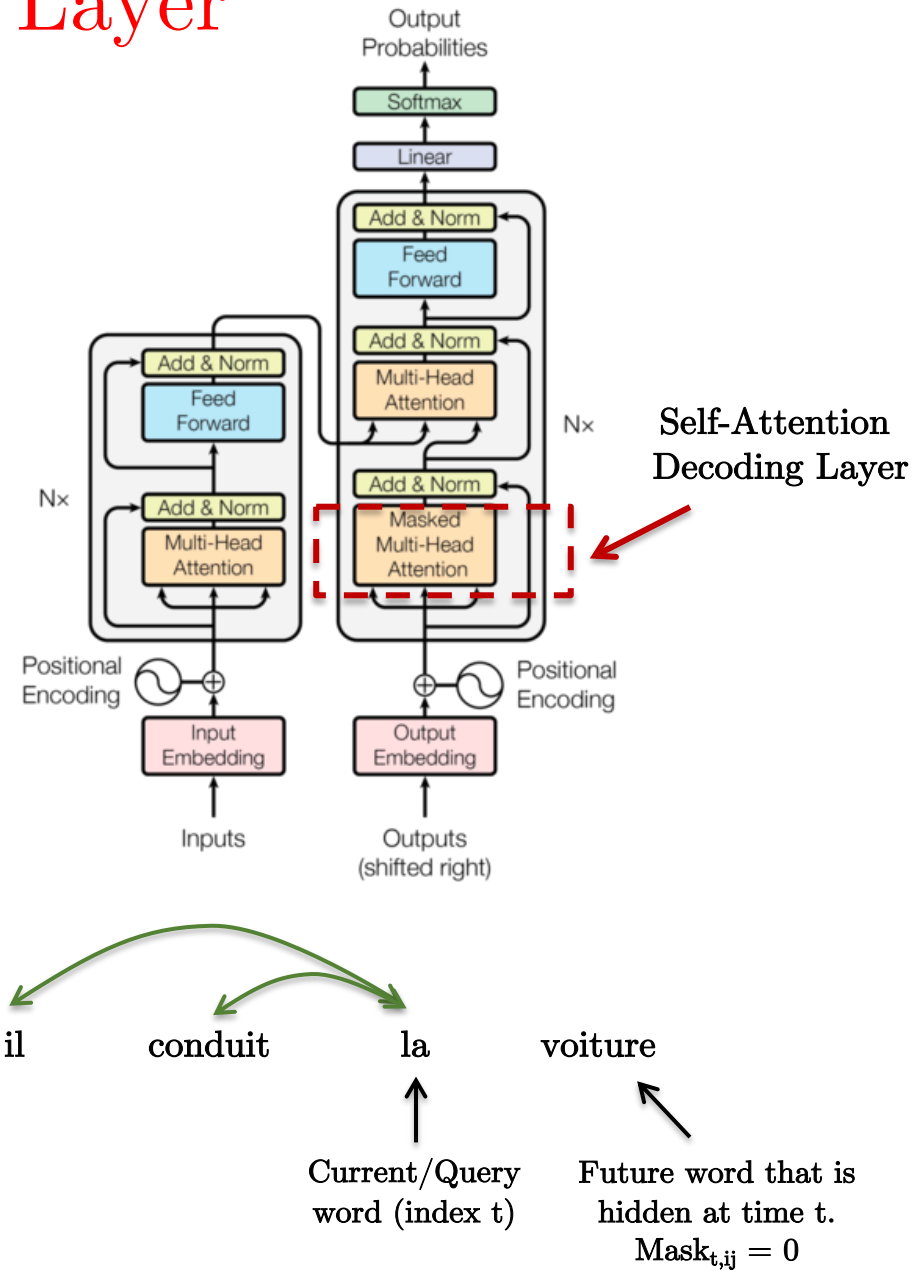
Self-Attention Decoding Layer

- Represent the query (here the current word in the partially decoded output sequence) with a masked self-attention layer.
  - Only make attention to words before the current world.
  - Use a mask to hide future words in the output sequence.

$$\bar{q}_t = \text{MHA}(q_t, K, V) \in \mathbb{R}^d,\ q_t \in \mathbb{R}^d, K \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{n \times d}$$
$$= \Big(\|_{h=1}^{H} \text{HA}_h(q_t, K, V) \Big) W^O,\ W^O \in \mathbb{R}^{d \times d}$$
$$\text{with } q_t = \text{g}_t^{\text{out}} \in \mathbb{R}^d,\ K = V = \{\text{g}_{t-1}^{\text{out}}, ..., \text{g}_{t-n}^{\text{out}}\} \in \mathbb{R}^{n \times d}$$

$$\text{HA}(q_t, K, V) = \text{Mask}_t \odot \text{Softmax}\Big(\frac{q_t K^T}{\sqrt{d/H}}\Big) V$$

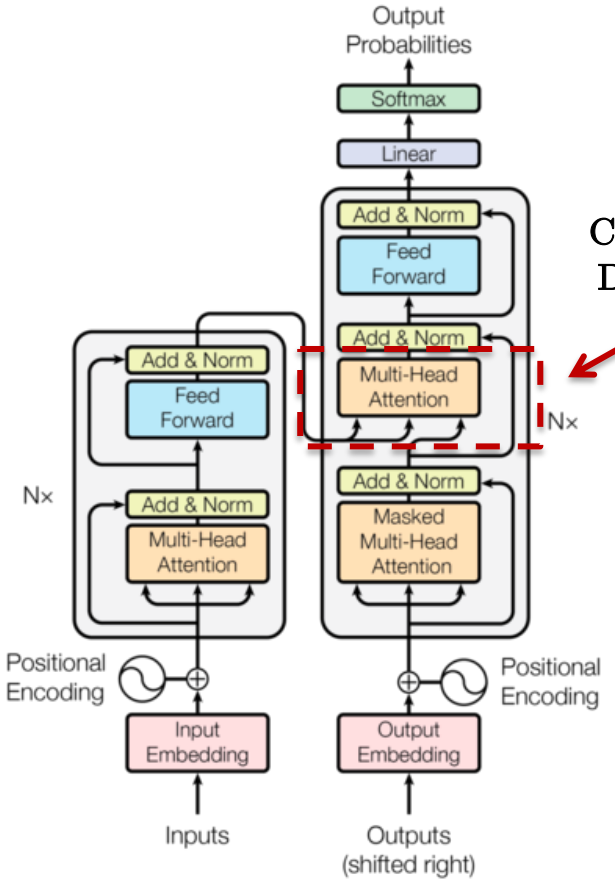
$$\text{with } \text{Mask}_{t,ij} = \begin{cases} 1 \text{ if attention between } i \text{ and } j \text{ at index } t \\ 0 \text{ if no attention} \end{cases}$$

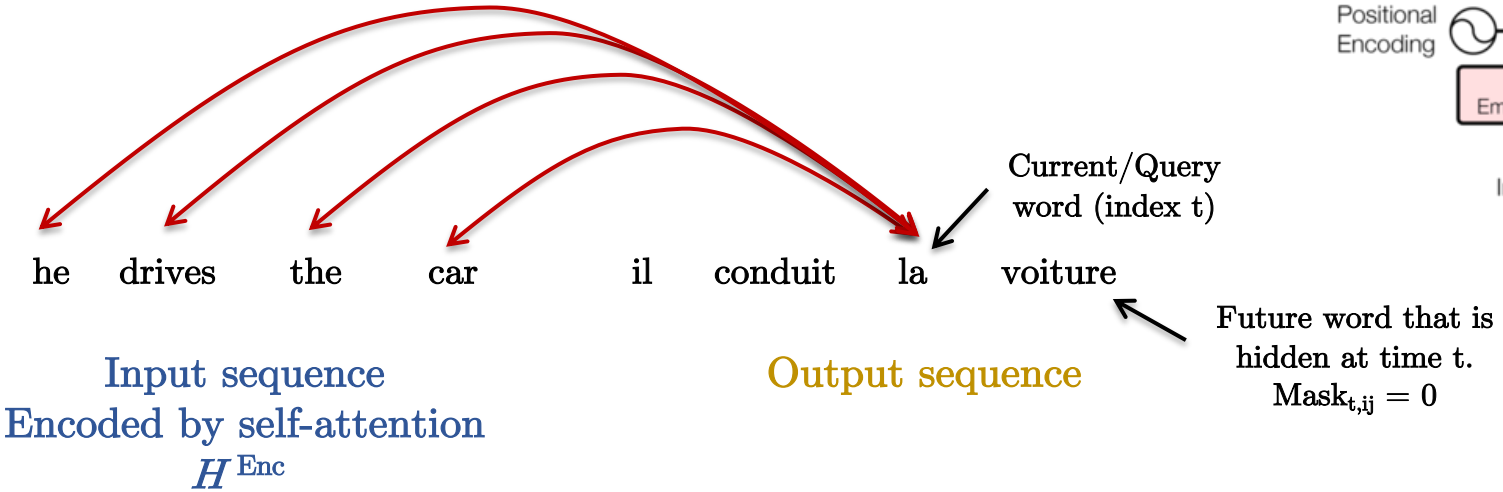


Cross-Attention Decoding Layer

- Compute attention between pairs of words coming from the whole encoded input sequence and the query in the output sequence.

$$\hat{q}_t = \text{MHA}(\bar{q}_t, K, V) \in \mathbb{R}^d,\ \bar{q}_t \in \mathbb{R}^d, K \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{n \times d}$$
$$= \left(\|_{h=1}^{H} \text{HA}_h(\bar{q}_t, K, V) \right) W^O,\ W^O \in \mathbb{R}^{d \times d}$$
$$\text{with } \bar{q}_t \in \mathbb{R}^d,\ K = V = H^{\text{Enc}} \in \mathbb{R}^{n \times d}$$





Decoder

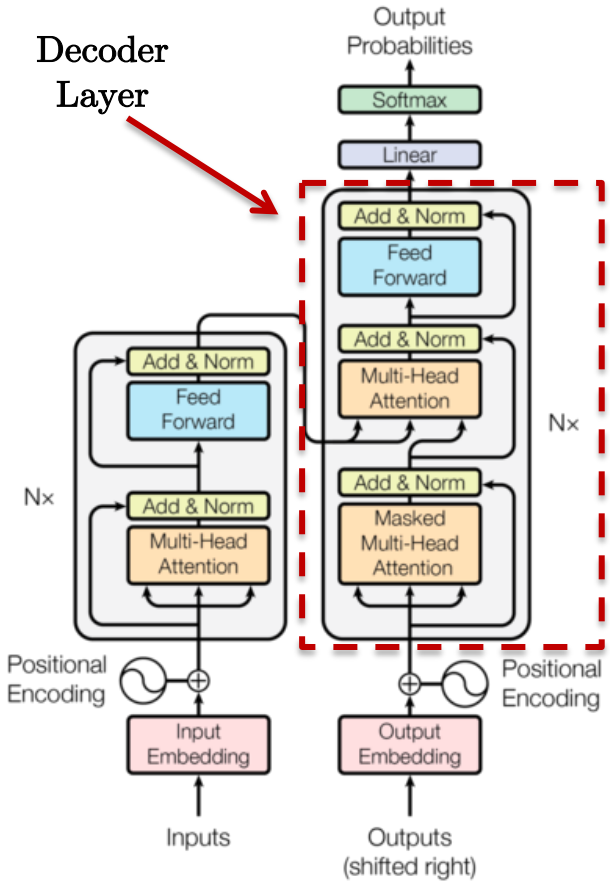
- Output of the decoder :

for $\ell = 0, ..., L^{\text{Dec}} - 1$

$$\bar{q}^{\ell+1} = \text{LN}\big(q^{\ell} + \text{MHA}(q^{\ell}, K^{\ell}, V^{\ell})\big) \in \mathbb{R}^d$$
$$\tilde{q}^{\ell+1} = \text{LN}\big(\bar{q}^{\ell+1} + \text{MLP}(\bar{q}^{\ell+1})\big) \in \mathbb{R}^d$$
$$\hat{q}^{\ell+1} = \text{LN}\big(\tilde{q}^{\ell} + \text{MHA}(\tilde{q}^{\ell}, H^{\text{Enc}}, H^{\text{Enc}})\big) \in \mathbb{R}^d$$
$$q^{\ell+1} = \text{LN}\big(\hat{q}^{\ell+1} + \text{MLP}(\hat{q}^{\ell+1})\big) \in \mathbb{R}^d$$

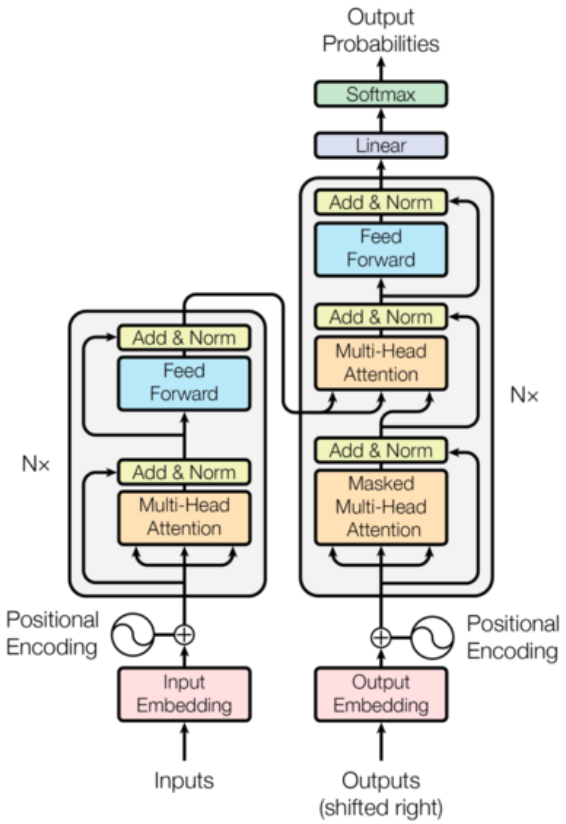
with initialization $q^{\ell=0} = \text{LL}(\text{g}_t^{\text{out}} + \text{PE}_t) \in \mathbb{R}^d$,

and $K^{\ell=0} = V^{\ell=0} = \text{LL}(\{\text{g}_{t-1}^{\text{out}}, ..., \text{g}_{t-n}^{\text{out}}\} + \text{PE}) \in \mathbb{R}^{n \times d}$



Lab 02

- PyTorch implementation of Seq2Seq Transformers



jupyter transformer_translation Last Checkpoint: 3 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Markdown

Lab 02: Sequence-To-Sequence with Transformers - Demo

The task is to learn to memorize an input sequence of length 100 and output the same sequence of length 100 but shifted by one word in the future.

For example, the input sequence is "some analysts expect oil prices to remain relatively"
and the output sequence is "analysts expect oil prices to remain relatively high".

In [50]:
```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    # find automatically the path of the folder containing "file_name" :
    file_name = 'transformer_translation.ipynb'
    import subprocess
    path_to_file = subprocess.check_output('find . -type f -name ' + str(file_name), shell=True).decode("utf-8")
    path_to_file = path_to_file.replace(file_name,"").replace('\n',"")
    # if previous search failed or too long, comment the previous line and simply write down manually the path below :
    path_to_file = '/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation/'
    print(path_to_file)
    # change current path to the folder containing "file_name"
    os.chdir(path_to_file)
    !pwd
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_r
emount=True).
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation/
/content/gdrive/My Drive/CS5242_notebooks/labs_lecture12/lab02_translation
```

In [51]:
```
import torch
import torch.nn.functional as F
import torch.nn as nn
import math
import time
import utils
```

GPU

It is recommended to run this code on GPU:

- Time for 1 epoch on GPU : xxx sec w/ Google Colab Tesla P100-PCIE-16GB

In [52]:
```
device= torch.device("cuda")
#device= torch.device("cpu")
print(device)
```

```
cuda
```

Transformers in 2017

- Machine Translation
  - WMT-2014 dataset
  - BLEU score

LSTM + Attention
(Google Brain)

Yonghui Wu et al, Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv:1609.08144, 2016

	EN-DE	EN-FR
GNMT	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformers	28.4	41.8

ConvNet
(FAIR)

Gehring et al, Convolutional sequence to sequence learning, arXiv:1705.03122, 2017

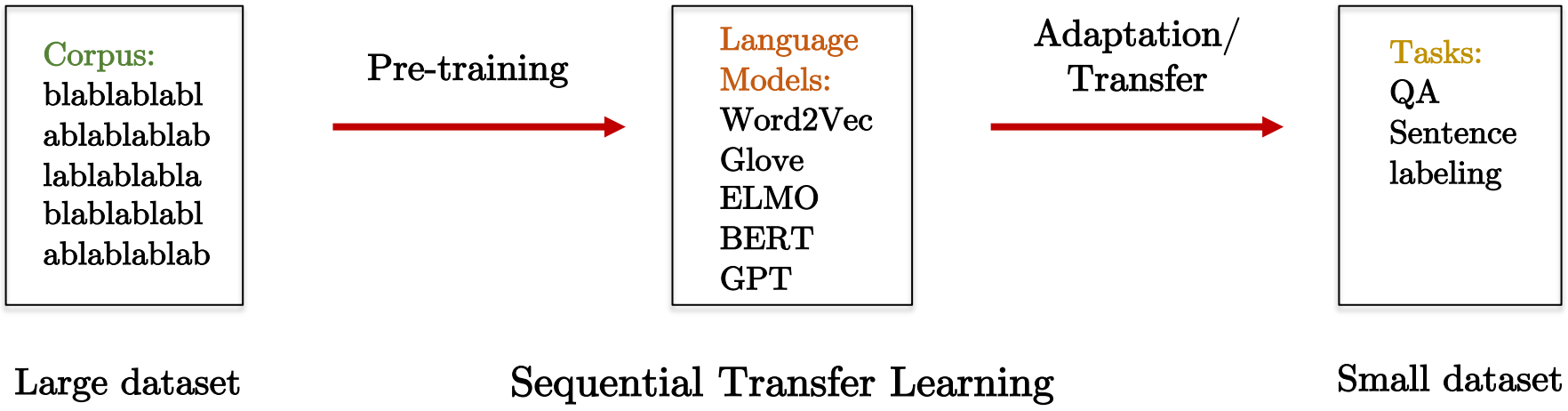
- 3x faster than LSTMs and ConvNets !
- Deep NLP architecture with 24 layers vs LSTM (3 layers).
  - ConvNets use 40 layers.

Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- **Transfer Learning with Language Models**
- Graph NNs vs Attention NNs
- Conclusion

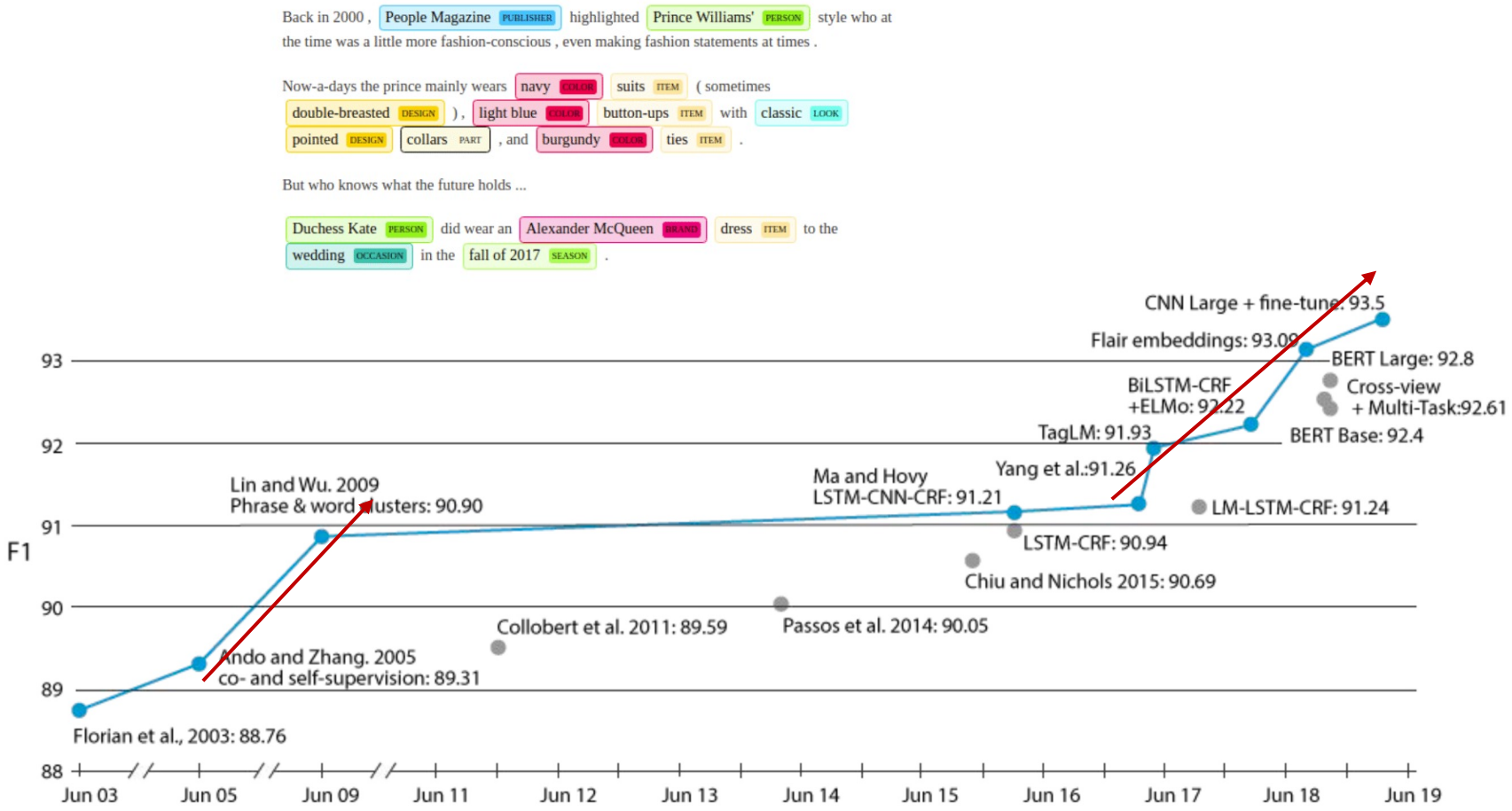
Transfer Learning with Language Modeling

- A major theme in NLP 2019 is sequential transfer learning
  - Pre-trained language models on large-scale corpus (for capturing language prior) and
  - Transfer to new tasks s.a. document classification, Q&A, named-entity recognition, etc by fine-tuning some layers on top of the pre-trained network (or by zero-shot learning).
  - Best performance for NLP tasks.



Example of Transfer Learning

- **Named-Entity Recognition :**



Language/Word Modeling

- Language Modeling (LM) provides a vectorial representation of a word given a context.
- This vectorial representation can be used to
  - Predict the word in a sequence.
  - Estimate the class of a sequence of words.

Once again the ??? were not able to handle the imbalances on the floor of the new york stock exchange said christopher senior ...

PTB dataset

Word: specialists

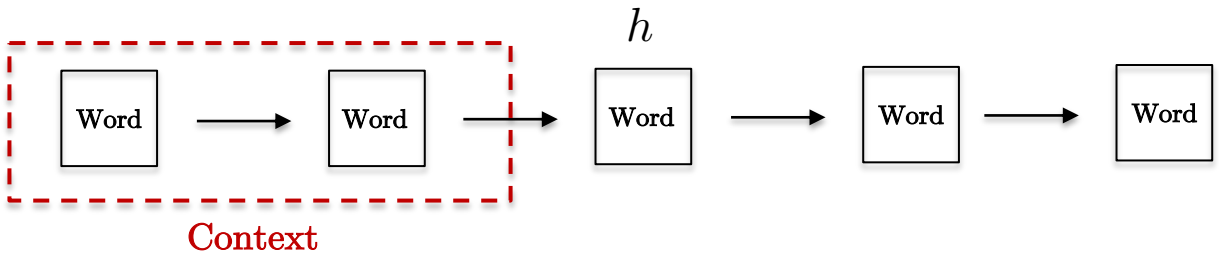
I had such a nice day. Too bad the rain comes in tomorrow at 5am.

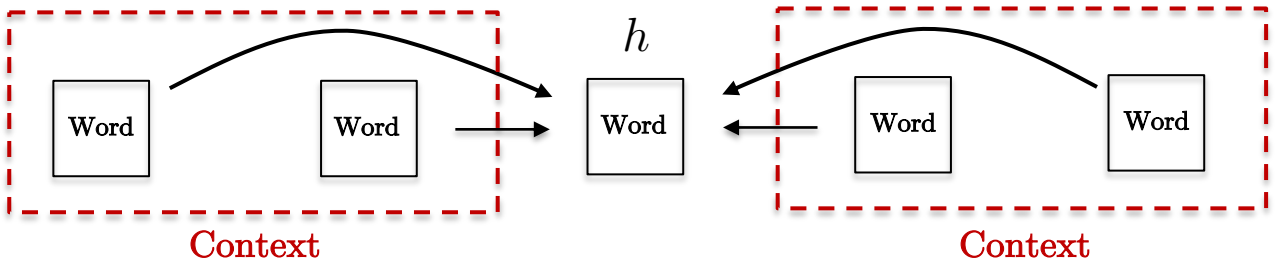
Twitter Dataset

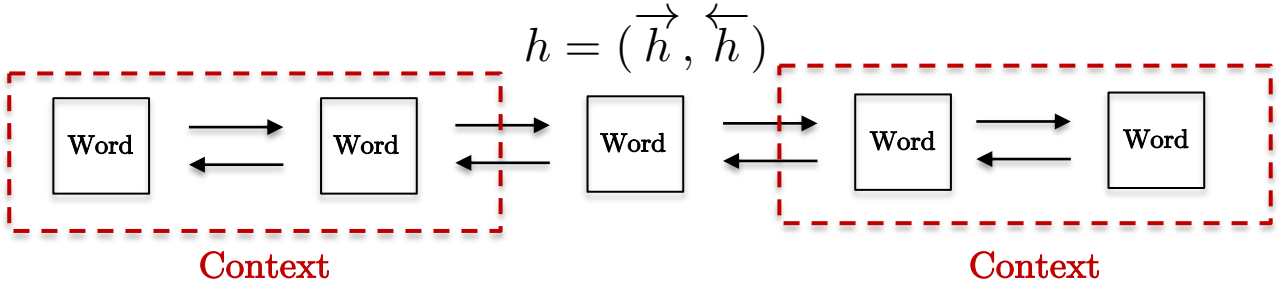
Class : Negative

Autoregressive vs Non-Autoregressive LM

- Autoregressive tasks
  - Sequence generation
  - Translation
  - Question & answer
  - Summarization
- Non-autoregressive tasks
  - Sentiment analysis
  - Name entity recognition
- Using autoregressive LM for non-autoregressive tasks







Language Modeling

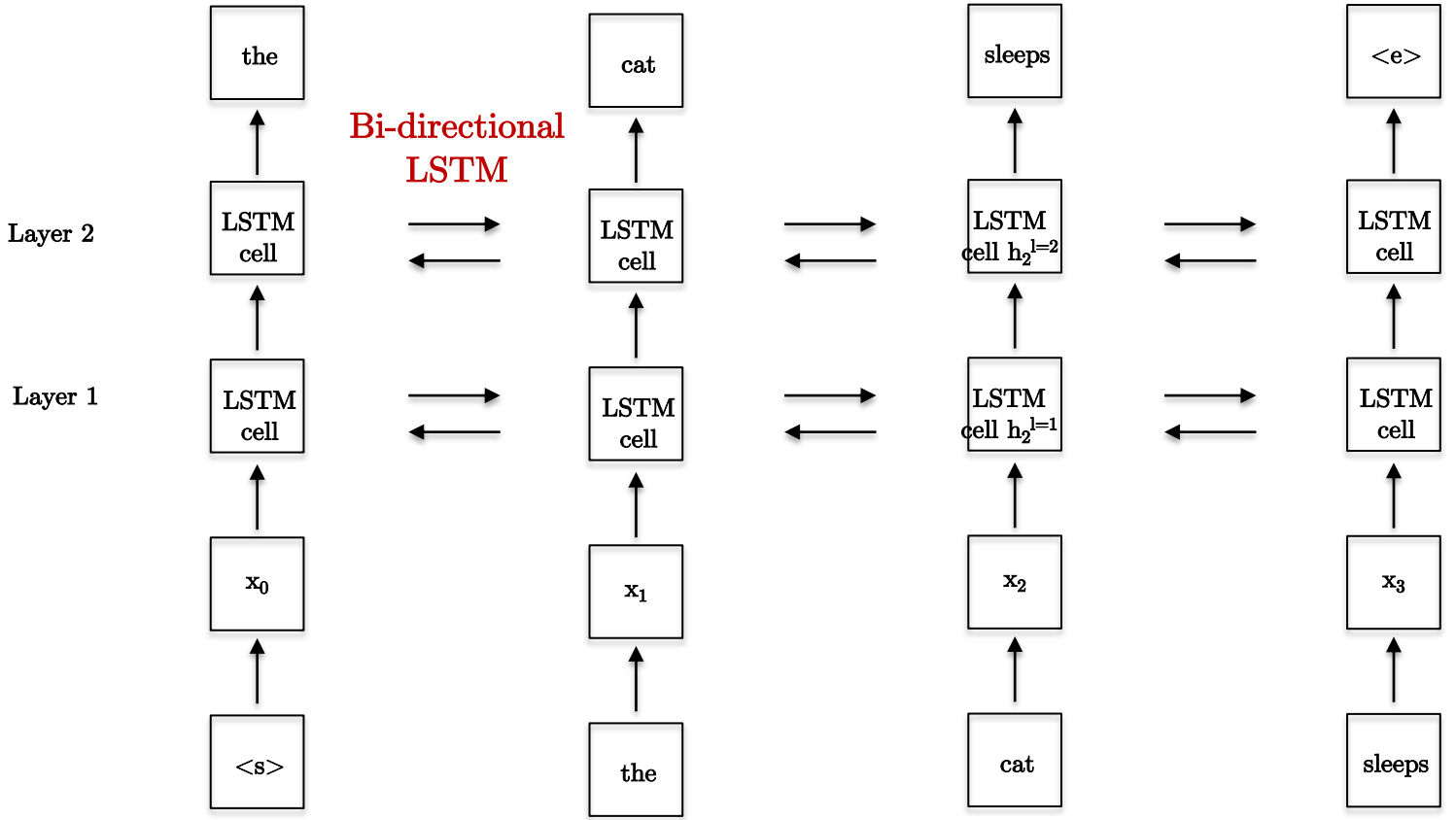
- Language Modeling (autoregressive or non-autoregressive) is an unsupervised/ self-supervised task.
  - No need for labeled data.
  - Consequence : Big (unlabeled) datasets are available for training.
    - Train with billions of words from Wikipedia, Reddit, etc.

English Wikipedia alone has
3.9 billion words (2021)

Words Posted to Reddit:
72 billion (2015)

ELMo

- Embedding for Language Modeling, Peters-et-al'18 (Allen Institute)



Transfer learning :
Represent word "cat" as a linear combination of x_2, $h_2^{l=1}$, $h_2^{l=2}$:

$$\text{cat} = W \begin{bmatrix} x_2 \\ h_2^{\ell=1} \\ h_2^{\ell=2} \end{bmatrix}$$

Transfer weight (learned by fine-tuning)

Context of word "cat" (given by bi-directional LSTM)

ELMo

- ELMo has proved to be effective at transfer learning.
  - One of the first to demonstrate the importance of context-to-word representation.

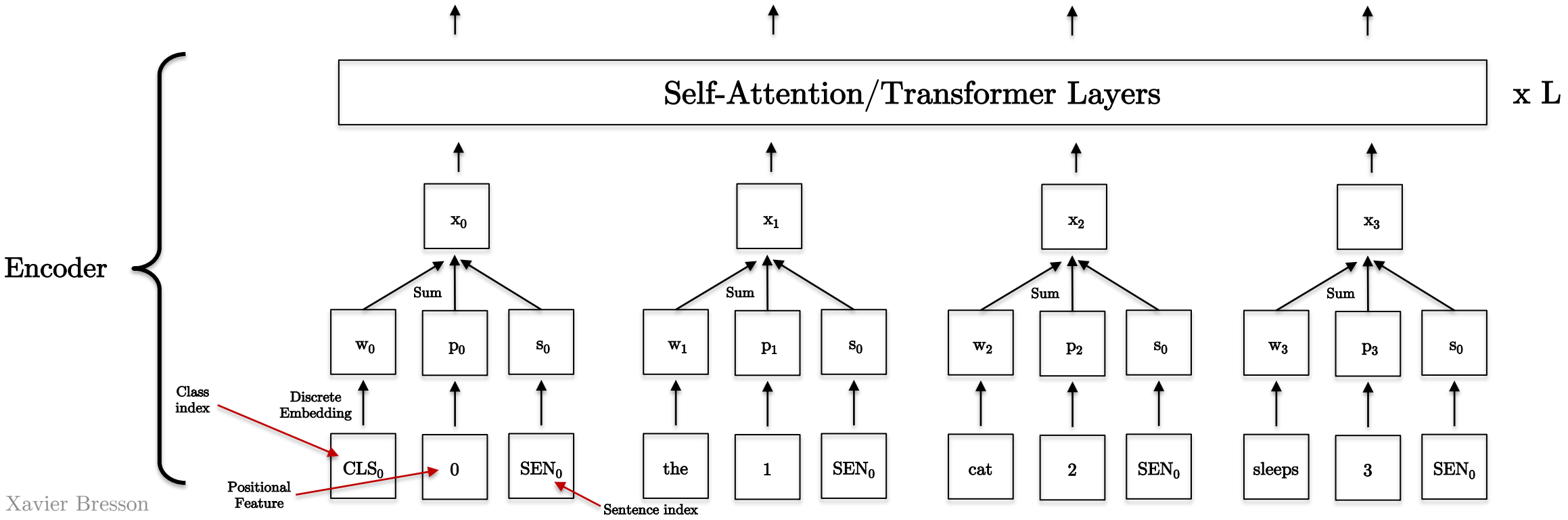
Use pre-trained context-to-word representation

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

- But ELMo is also limited by LSTM (RNN) which cannot be parallelized (sequential technique) and does not improve performance with more than 2-3 layers.

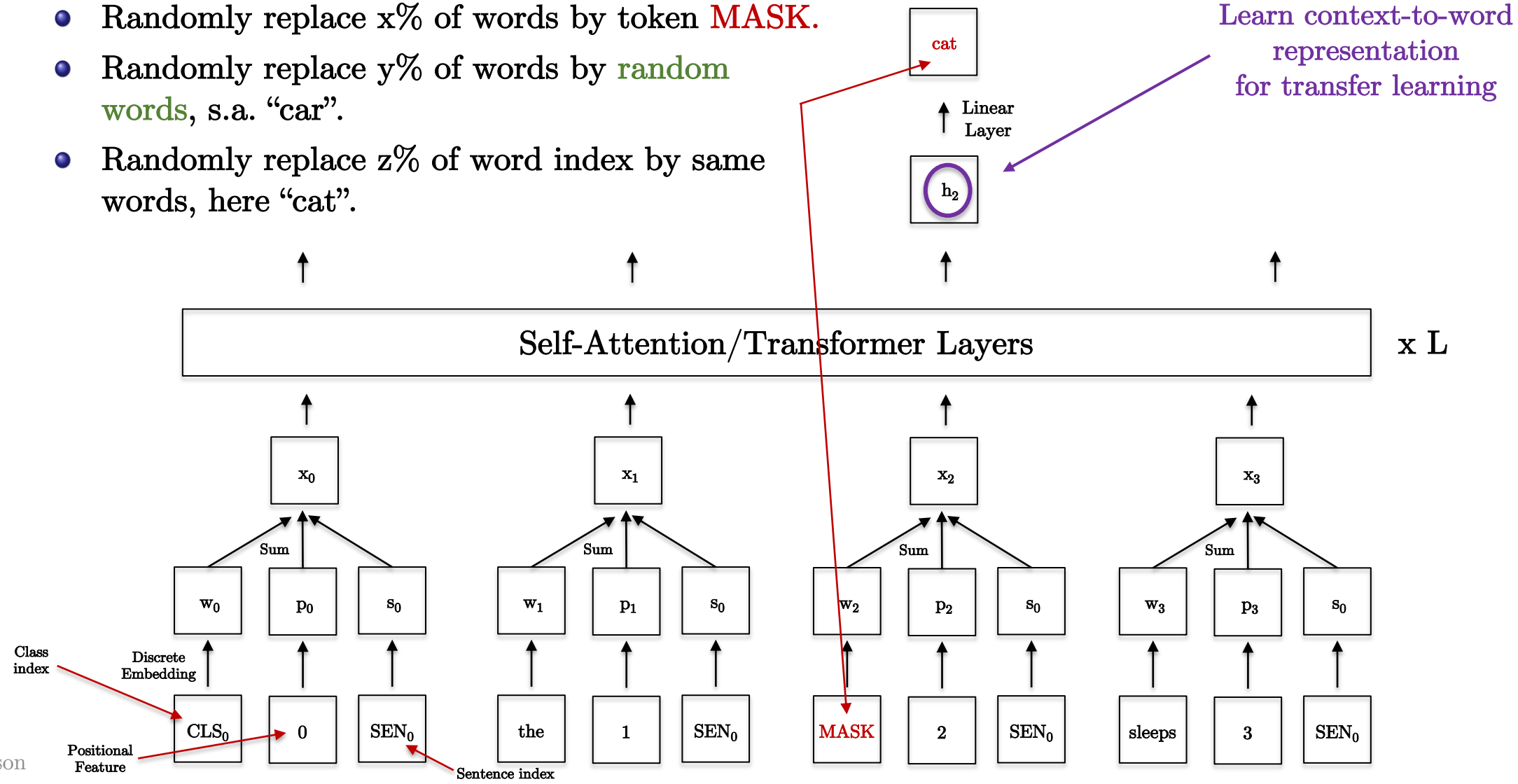
BERT

- Bi-directional Encoder Representation for Transformers, Devlin-et.al'19 (Google)
  - Use positional encoding, class index and sentence index.
  - Trained with two levels of hierarchical context :
    - Local dependencies with word prediction.
    - Global dependencies with sequence prediction.



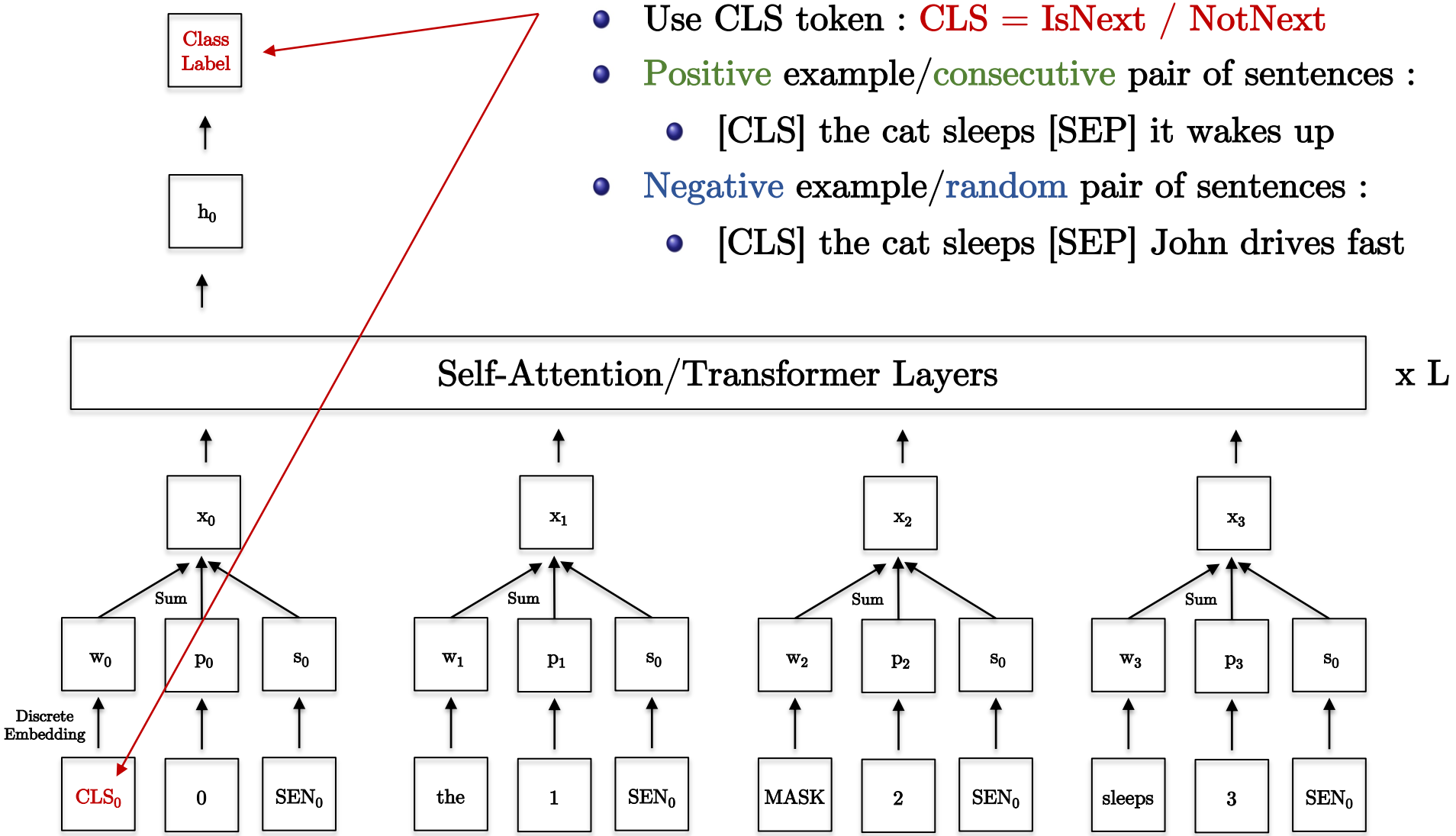
Word Prediction

- Train by predicting words
  - Randomly replace x% of words by token MASK.
  - Randomly replace y% of words by random words, s.a. “car”.
  - Randomly replace z% of word index by same words, here “cat”.



Sentence Prediction

- Train by predicting the next sentence
  - Use CLS token : CLS = IsNext / NotNext
  - Positive example/consecutive pair of sentences :
    - [CLS] the cat sleeps [SEP] it wakes up
  - Negative example/random pair of sentences :
    - [CLS] the cat sleeps [SEP] John drives fast

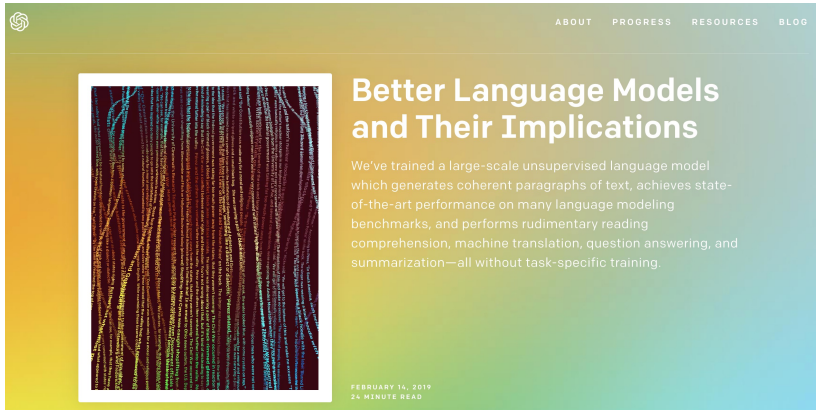


Training

- BERT base
  - 12 Transformers layers
  - 768 hidden features
  - 12 Attention heads
  - 110M parameters
- BERT large
  - 340M parameters
- Special tokenization of words with only 30K tokens.
- Dataset of 3B words
- Training took 256 TPU days (Oct 2018)
- Fine-tune on sentence classification, named-entity recognition (word classification), Q&A, etc.

GPT-2

- Improving Language Understanding by Generative Pre-Training, Radford et-al 2019 (OpenAI)
  - Pre-trained on 8M webpages, WebText 40GB.
  - SOTA on 7 NLP tasks without fine-tuning, simply by zero-shot learning !
  - 1.5B parameters
  - 2048 GPUs



To wit: when GPT-2 was tasked with writing a response to the prompt, "Recycling is good for the world, no, you could not be more wrong," the machine spat back:

> "Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources."

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. …

GPT-3

- Introduced in May 2020
- Pre-trained on multiple datasets.
- 175B parameters
- "The supercomputer developed for OpenAI is a single system with more than 285,000 CPU cores, 10,000 GPUs and 400 gigabits per second of network connectivity for each GPU server"
- US$12 Million to train

Dataset	Quantity (tokens)
Common Crawl (filtered)	410 billion
WebText2	19 billion
Books1	12 billion
Books2	55 billion
Wikipedia	3 billion

GPT-3 Powers the Next Generation of Apps

Over 300 applications are delivering GPT-3–powered search, conversation, text completion, and other advanced AI features through our API.

JOIN THE WAITLIST ↗

Nine months since the launch of our first commercial product, the OpenAI API, more than 300 applications are now using GPT-3, and tens of thousands of developers around the globe are building on our platform. We currently generate an average of 4.5 billion words per day, and continue to scale production traffic.

Given any text prompt like a phrase or a sentence, GPT-3 returns a text completion in natural language. Developers can "program" GPT-3 by showing it just a few examples or "prompts." We've designed the API to be both simple for anyone to use but also flexible enough to make machine learning teams more productive.

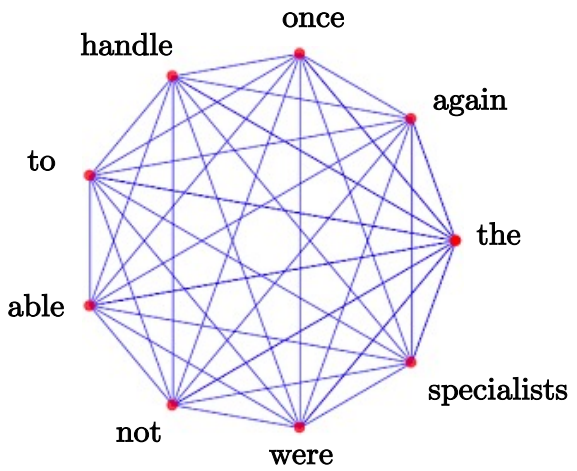
Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- **Graph NNs vs Attention NNs**
- Conclusion

ANNs vs GNNs

- Connections with GNNs
  - Attention learns a graph.
    - A graph represents the pairwise relationships between data.
    - Attention scores represent the strength of the pairwise relationships.

- ANNs are a special case of GNNs when
  - Each data is connected to all other data.
    - Fully connected graph (Vaswani et-al'17)
  - Or the attention is specifically selected to a subset of data.
    - Sparse/Graph Transformers (Dwivedi, Bresson'21)



Outline

- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Language Model Transformers
- Sequence-To-Sequence Transformers
- Transfer Learning with Language Models
- Graph NNs vs Attention NNs
- **Conclusion**

Human Level Translation

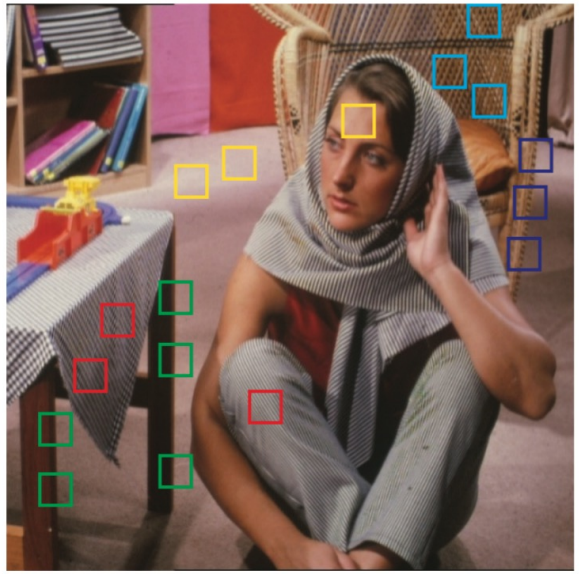
- Human have common sense.
  - What is obvious to humans is not to machines.
  - Winograds Schemas
    - The machine must identify the antecedent of an ambiguous pronoun in a statement.
    - Example 1 :
      - He didn't put the trophy in his suitcase because it was too small.
        - "it" refers to the suitcase.
      - He didn't put the trophy in his suitcase because it was too large.
        - "it" refers to the trophy.
    - Example 2 :
      - The women stopped taking pills because they were pregnant.
        - "they" refers to women.
      - The women stopped taking pills because they were carcinogenic.
        - "they" refers to pills.

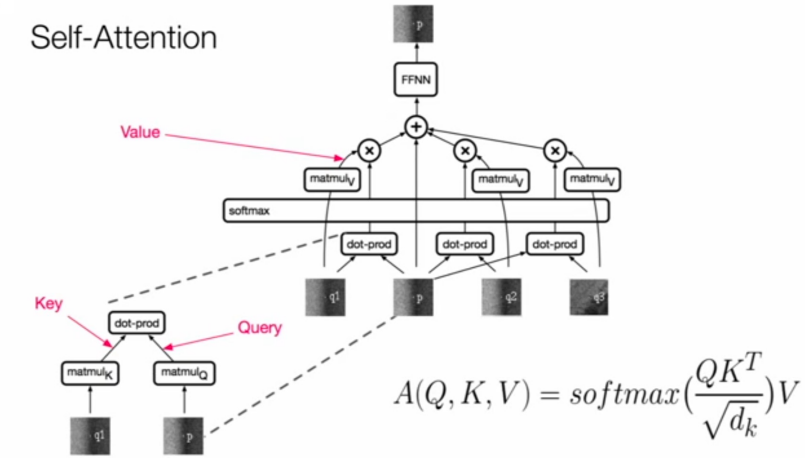
Attention Networks

- Human attention mechanism allows to focus biological resources on a small set of important things (visual, sound, cognitive signals) to make decisions.
- ANNs are a generic/universal architecture to process any unstructured datasets, a.k.a. sets.
  - Good but also too generic !
    - It requires specialization to data/task.
      - For example in NLP, use positional encoding for data ordering.
- Open questions
  - Performance analysis
  - How to train ANNs (generically, without tricks)?
  - How to regularize effectively attention mechanism?

Attention in Computer Vision

- Attention is "eating" deep learning :
  - Applications to NLP very successful.
  - Preliminary ideas of visual attention (image self-similarity) in Computer Vision :
    - Efros-Leung'99 for image generation.
    - Buades-Morel'05 for image denoising.
  - Issue of long sequences, here image with 3,072 pixels.
    - Use image 16 x 16 patches.
  - Many recent papers have investigated Visual Transformer architectures.

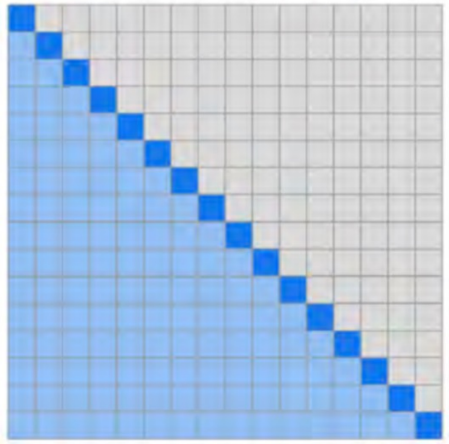




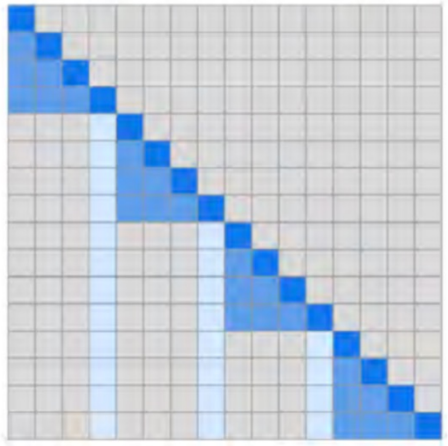
$$A(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Improving Quadratic Complexity

- Long sequence issue with $O(n^2d)$, n being sequence length and d hidden dimension.
  - Structured/Sparse Transformers Child et-al'19.
  - Linear approximations of the attention mechanism.



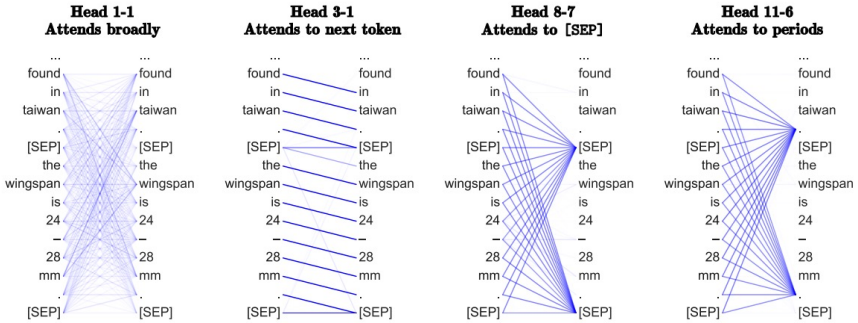
Original
Transformers



Sparse
Transformers

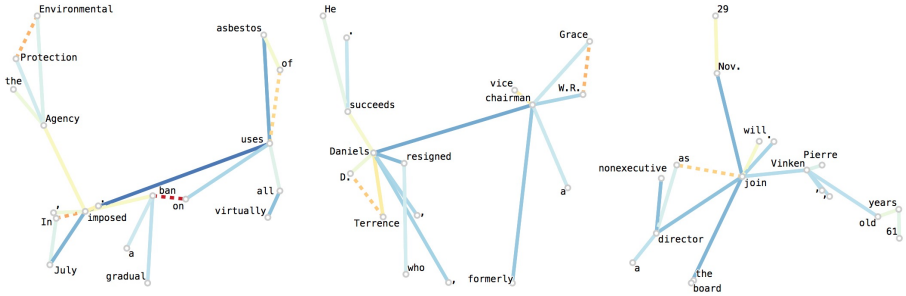
Interpretability

- What Does BERT Look At? An Analysis of BERT's Attention, Clark, Manning-et.al'19, Stanford



- Visualizing and Measuring the Geometry of BERT, Coenen-et-al.'19, Google

  https://pair-code.github.io/interpretability/bert-tree



Open Source Transformers

- Pre-training large-scale models is costly in terms of time, money, CO_2 :

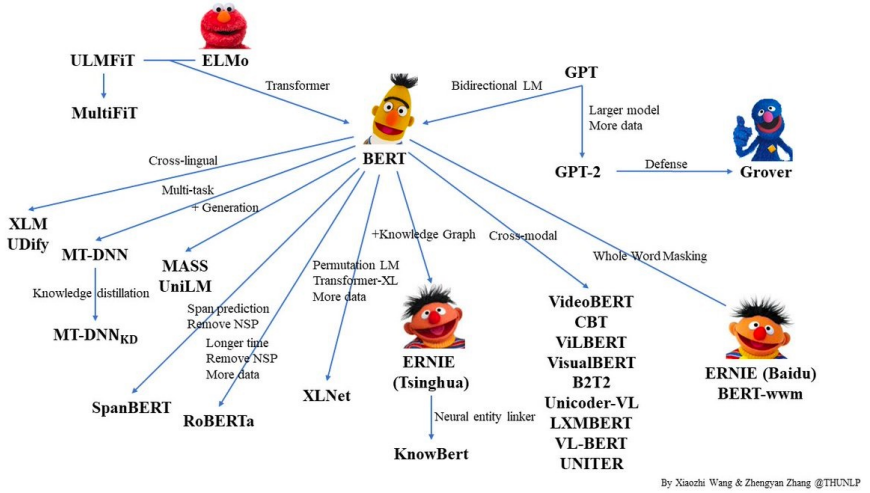
Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$\text{Transformer}_{base}$	P100x8	1415.78	12	27	26	$41–$140
Transformer_{big}	P100x8	1515.43	84	201	192	$289–$981
ELMo	P100x3	517.66	336	275	262	$433–$1472
BERT_{base}	V100x64	12,041.51	79	1507	1438	$3751–$12,571
BERT_{base}	TPUv2x16	—	96	—	—	$2074–$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	$942,973–$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055–$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902–$43,008

Table 3: Estimated cost of training a model in terms of CO_2 emissions (lbs) and cloud compute cost (USD).[7] Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

- Available pre-trained models :
  - Hubs : PyTorch, TensorFlow
  - Codes + checkpoints : BERT, ELMo
  - Third party libraries : HuggingFace, Allen Institute



- Material :
  - Arthur Szlam, Tutorial on “Multi-Hop Attention” at IPAM-UCLA, Feb 2019
  - Alex Smola and Aston Zhang, Tutorial on “Attention in Deep Learning”, ICML, Jun 2019
  - Ashish Vaswani, Talk on Self-Attention For Generative Models, CS224N, Mar 2019
  - Łukasz Kaiser, Talk on Tensor2Tensor Transformers, Oct 2017
  - Sebastian Ruder, Talk on Transfer Learning in Open-Source Natural Language Processing, spaCy IRL, Jul 2019
  - Christopher Potts, Contextual Vectors, Stanford CS224U NLP, Lecture 14, Jul 2019
  - Christopher Manning, Contextual Word Embeddings, Stanford CS224N, Lecture 13, Mar 2019

Questions?