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
- Sequence-To-Sequence Transformers
- Language Model Transformers
- Graph NNs vs Attention NNs
- Conclusion

Outline

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

Neural Networks

- Main goal of neural networks is to learn the best possible data representation, which can be used to solve any arbitrary task (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).

Review of Neural Networks

- MLP/FC :
  - Good for (quasi-)linear data.
- ConvNets :
  - Great for grid-structured data like 2D/3D images and videos.
  - Revolution in Computer Vision since 2012.
- RNNs :
  - Good for ordered sequential data like text and sound/speech.
  - Great progresses but not breakthrough in NLP.

Network Structure

- MLP/FC :
  - Pattern matching (fixed size)
- ConvNets :
  - Translation invariance on grids (stationarity)
  - Hierarchical representation (multi-scale)
  - Compositionality learns with hierarchy and stationarity (linear #parameters to parametrize exponential functions)
- RNNs :
  - Time/ordered translation invariance (weights are shared across time)
  - Causality (ordered sequences)
  - Memory/summary of sequences (recurrence formula)
  - Forget/remember mechanism (gates)

MLP/FC Neural Networks

- Input/output data sizes are fixed.
- Compositional functions :

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

- No data structure used.
  - Restricted to linear data.

ConvNets

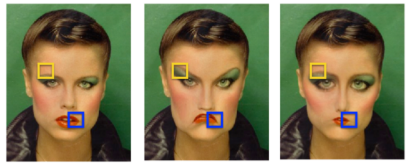
- Input data has a grid structure.
- Input size can change !
  - Convolution is invariant to input size, although images are usually resized to the same shapes (for batch computational efficiency/GPU)
  - ConvNet layer :

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

- Convolution is a linear operation like MLP but specialized to template matching with sliding window (translation).
- A_l is the filter/kernel (size is fixed and small/compact support)

ConvNets

- Pooling introduces :
  - Global translation invariance on the grid at the cost of lower resolution.
  - Local deformation invariance on the grid.
    - Essential for intra-class variations and recognition robustness.



- ConvNets have been very successful in Computer Vision :
  - First model to solve the fundamental image recognition task started in 1960s.
  - First model that captures appropriate image invariances.
  - Great at classification rate and transfer learning (generalization).

RNNs

- Input is an ordered 1D grid structure.
- Input length and output length are variable :

$$y_i = f(x_0, x_1, ..., x_i)$$

  - Output y_i is a function of all inputs up to now.
- How to learn representation of sequences with different length/size ?
  - Fixed memory size (upper size limit and zero padding), too limited.
  - Recurrence mechanism

RNNs

- Recurrence mechanism / memory module :
  - Apply the same equation to input data at all time steps :

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

    - This process is independent of the sequence length !
  - Depth (#layers) of RNNs is the length of the sequence (can be billions layers).
  - Gating/multiplicative mechanism is essential :

$$\sigma \odot h$$

    - Forgets/remembers relevant information for NLP tasks.
    - Prevents blowing up.
  - Dominant NNs (for learning variable length sentence representation) in NLP up to 2018.
    - Used for MT, QA, summarization, etc

RNNs

- Input sequences : $x_0, x_1, ..., x_n, ...$
- Hidden states : $h_0, h_1, ..., h_n, ...$ $\quad h_{i+1} = f(h_i, x_{i+1})$
- Output sequences : $y_0, y_1, ..., y_n, ...$ $\quad y_{i+1} = g(h_{i+1})$

f, g are independent of time.

f, g can be LSTM/GRU layers.

- RNNs are non-Markovian chains :
  - Markov chain :

$$x_{i+1} = f(x_i) = Px_i$$

  - RNNs keep hidden states h (memory/summary of sequence) :

$$y_{i+1} = (g \circ f)(h_i, x_{i+1})$$

RNNs

- Major limitations of RNNs :
  - They cannot remember/summarize long sequences (no more than 50 steps)
    - Because any small perturbation in non-linear systems will either be :
      - Amplified and blows up after 50 iterations
      - Decreased and vanishes after 50 iterations
  - Consequently, they cannot learn long-term dependencies.
  - They cannot stack more than a few layers (usually 2-3) along the representation axis.
  - They are slow because they are sequential processes (vs parallelizable ConvNets).
    - Important limitation in the era of Big Data.

Input Features (Reminder)

- Images in CV :
  - Inputs of ConvNets is a continuous tensor :

$$x_i \in \mathbb{R}^{n_x \times n_y}$$

- Text in NLP :
  - Inputs of RNNs is a discrete variable, indicating the index of the world in the dictionary :

$$w_i \in \{0, 1, ..., |D| - 1\}$$

  - Backpropagation does not work on discrete variables (gradient is not defined).

$$\nabla_w L(f(w)) \quad ?$$

    - Discrete variables must be embedded in a continuous space :

$$f(w_i) = x_i, \quad i^{th} \text{ row of } X \in \mathbb{R}^{n \times d}$$

    - Discrete embedding is a simple indexing/slicing operation of a matrix X (look-up table).
    - Embedding matrix X can be learned by backpropagation.

RNNs for NLP

- Discrete input sequence (text corpus) :

$$w_0, w_1, ..., w_n, ...$$

- Continuous input sequence (embedding) :

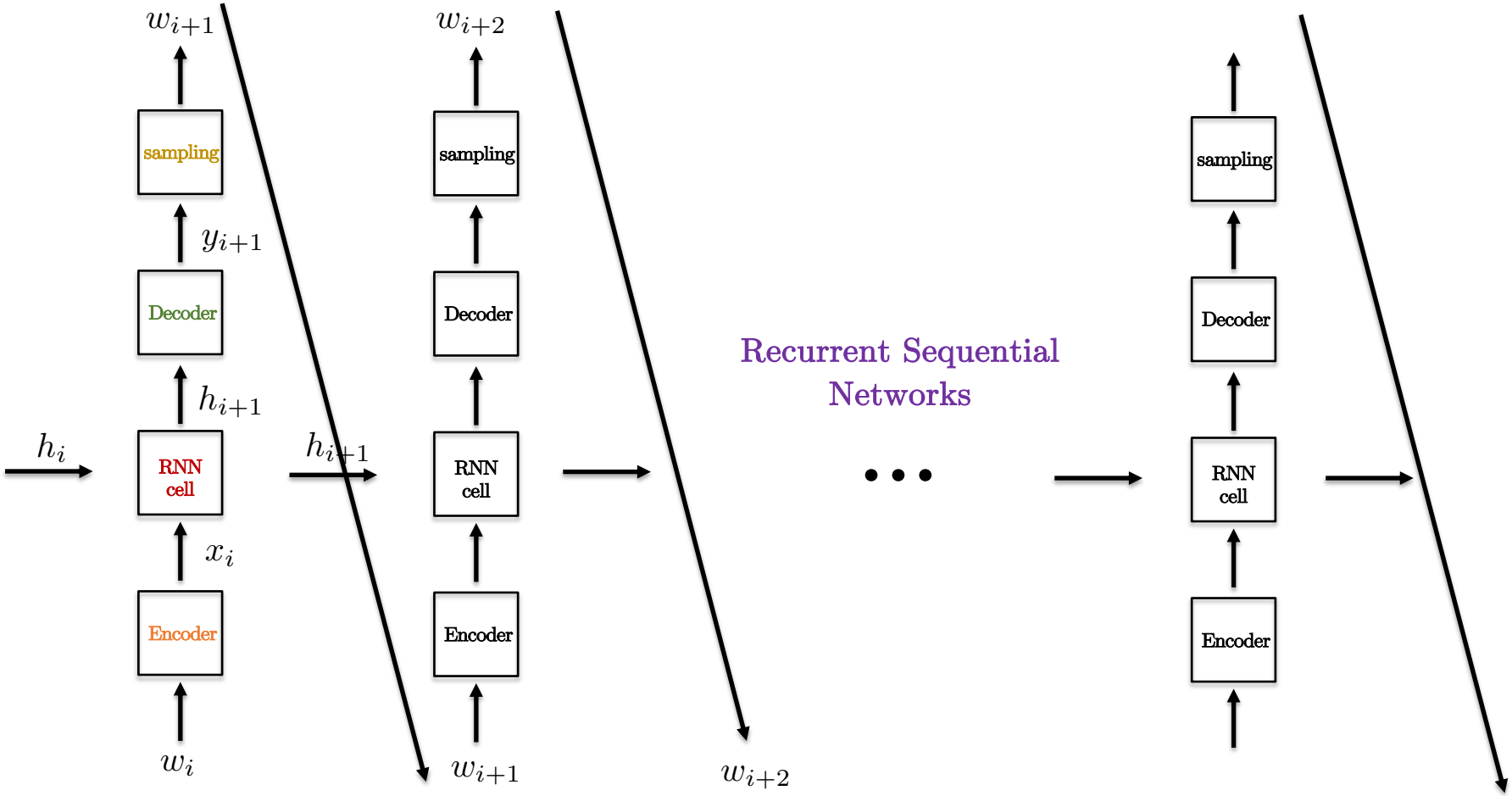
$$x_0, x_1, ..., x_n, ...$$

- Predictive task :
  - Given word w_0 , predict word w_{i+1}.

- Network updates :

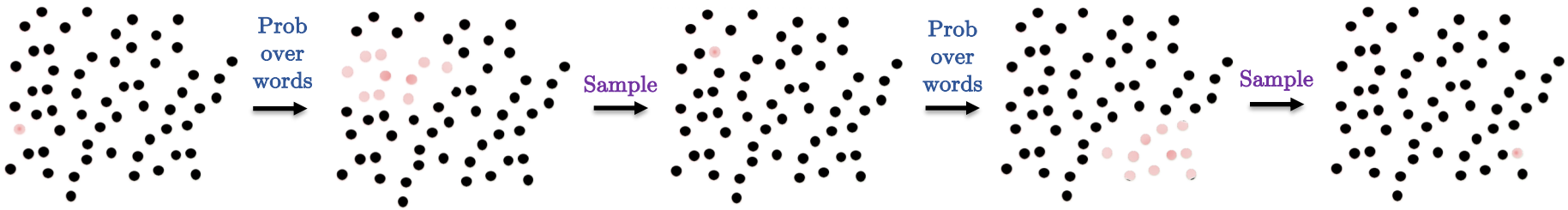
$$h_{i+1} = f_{\text{NN}}(h_i, x_{i+1})$$
$$y_{i+1} = g_{\text{NN}}(h_{i+1})$$

RNNs for NLP


w_{i+1}
w_{i+2}
sampling
sampling
sampling
y_{i+1}
Decoder
Decoder
Decoder
h_{i+1}
h_i
h_{i+1}
RNN cell
RNN cell
RNN cell
x_i
Encoder
Encoder
Encoder
w_i
w_{i+1}
w_{i+2}
Recurrent Sequential Networks

Text generation

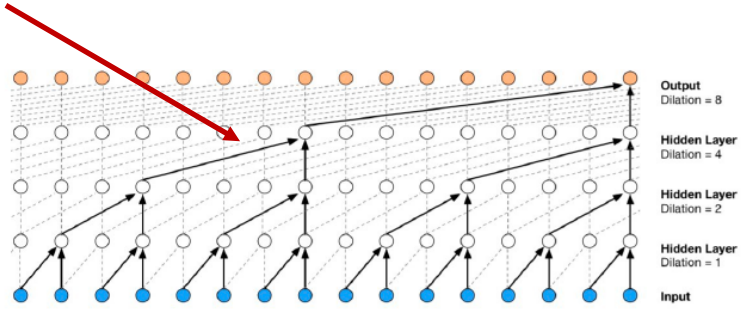
- By-product :
  - It is easy to generate new text sequences :
    - Given a seed w_0, roll-out.
- Start at word w_0,
- Iterate :
  - Update network and get distribution probability over words to select the next word,
  - Sample next word w_{i+1}.



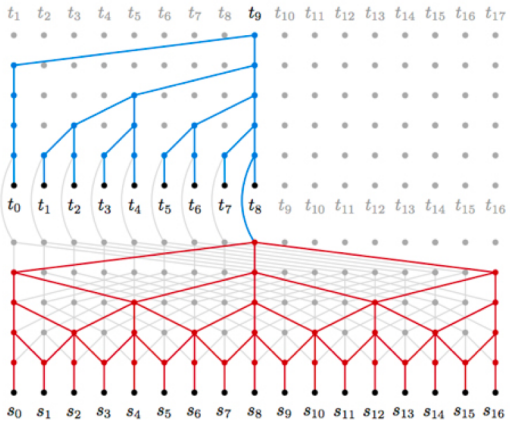
ConvNets for NLP

- Can we learn representation of sequences with different length/size with ConvNets ?
  - Yes ! Convolution is independent of the sequence length.
- WaveNets/ByteNets are examples of ConvNets architectures for NLP.

Visualization of a stack of dilated causal convolutional 1D layers



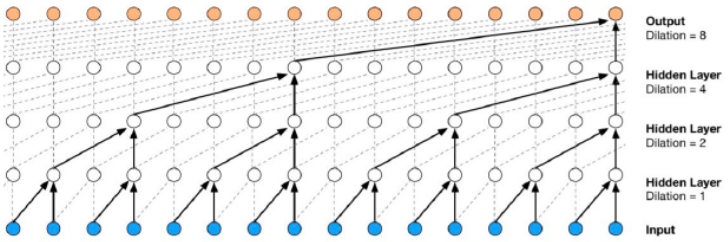
WaveNets : Generative model for raw audio
Blog : https://deepmind.com/blog/article/wavenet-generative-model-raw-audio



ByteNets : Machine Translation

ConvNets for NLP

- Advantages :
  - Easy to parallelize
  - Learn local dependencies (local reception fields/compact kernels)
- Limitations :
  - Difficult to learn long-term dependencies :
    - Require to stack many layers (increase the dependence domain) or
    - Use special multi-resolution/hierarchical architecture.
      - WaveNets/ByteNets force arbitrary dependencies due to dilated convolutions.
      - For example, word 4 and word 8 require 3 hops to connect, but word 5 and word 9 require 4 hops.



Outline

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

Data Domain

- ConvNets :
  - Data on multi-dimensional grids
- RNNs :
  - Data on ordered 1D grids
- Graph NNs :
  - Data on graphs/non-regular grids (later discussed)
- What about data not supported by any grid ?
  - Inputs are data with no graph structure, a.k.a. sets/bags (of features).
  - How to define neural networks for sets ?

Neural Networks for Sets

- How to design neural networks for sets ?
  - What invariances are necessary for processing sets ?
    - Permutation invariance (item indexing does not matter)
    - Size invariance (representation independent of set size)
  - Simplest solutions / pooling operations :
    - Function of the mean/sum/max operator for sets of vectorial data [Zaheer, Deep Sets'17]:

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

Mean

where vector 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.

Deep Sets with Attention

- Attention-based deep sets [Ilse-et.al.'18]:
  - Focus on the most relevant data.
    - Weighted mean/sum :

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

  - Attention $a_i = a(x_i, X \setminus x_i)$ is a probability distribution over all data X, which can be static or change dynamically as a function of data and state of the system.
  - Weight a_i can be binary (hard attention) or continuous (soft attention).

Attention over Sets

- Weighted sets/bags :

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

- Soft attention :

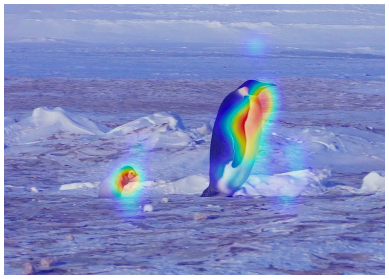
$$\begin{aligned} a_i &= g(x_0, ..., x_i, ..., x_{n-1}, h), \ x_i \in \mathbb{R}^{1\times d}, \ h \in \mathbb{R}^{1\times d} \\ &= \mathrm{softmax}(x_i h^T) \\ &= \frac{e^{x_i h^T}}{\sum_j e^{x_j h^T}} \end{aligned}$$

- Hard attention :

$$\begin{aligned} a_i &= \delta(x_i, h) \\ &= \begin{cases} 1 & \text{if } h = x_i \\ 0 \end{cases} \end{aligned}$$

History of Attention

- Attention in CV :
  - Attention over grid



Visual attention

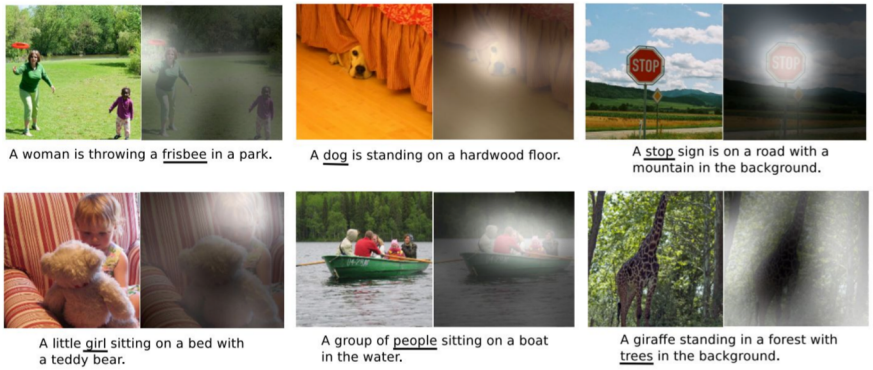
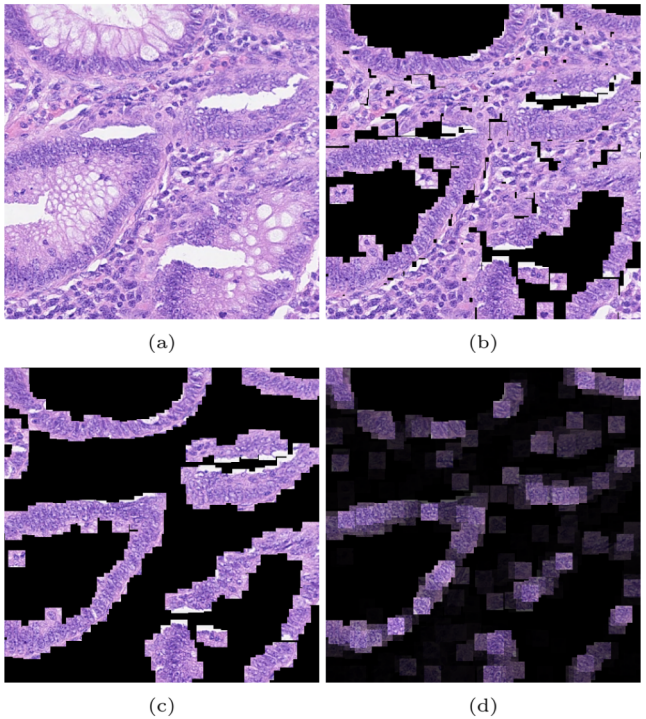


Image captioning

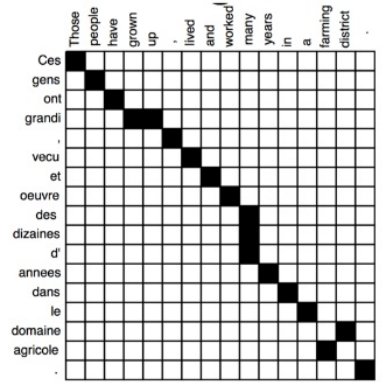


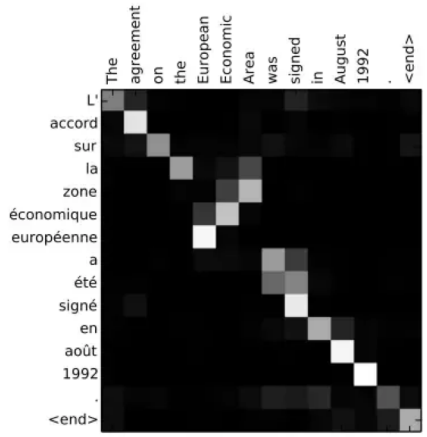
Ground truth patches that belong to the class epithelial.

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

History of Attention

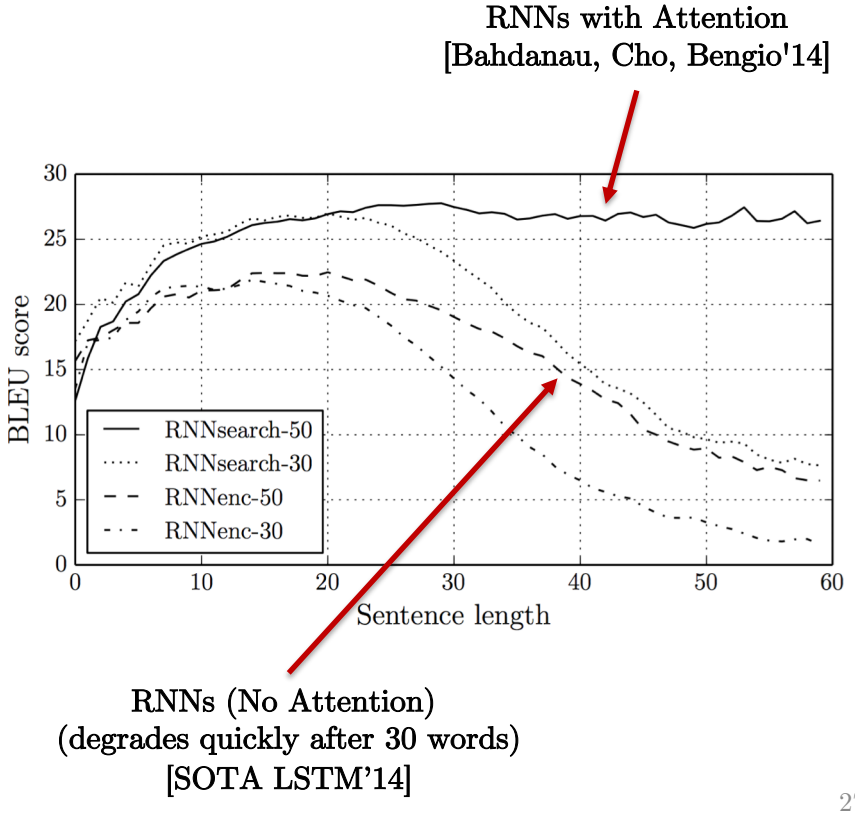
- Attention in NLP :
  - Alignment in Machine Translation (MT) :
    - Hard attention :
      - (Binary) alignment is a (difficult) combinatorial optimization problem.
    - Soft attention :
      - For each word in target sequence, get a probability over words in the source sequence [Brown-et-al'93].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





Attention in NLP

- Attention in NLP :
  - First paper to do soft alignment with attention in MT is [Bahdanau, Cho, Bengio'14], followed by [Luong, Pham, Manning'15].
    - Single hop attention.
    - Improved SOTA in MT.
    - Precursor of Transformers [Vaswani-et-al'17].
    - Start competing RNNs/LSTM.
- Breakthrough idea in NLP !
  - As revolutionary as ConvNets for CV.

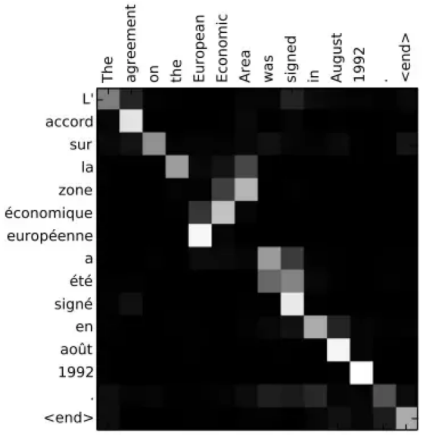


Attention in NLP

- Why casting MT as a soft alignment/attention problem is a good 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 nice but) RNN architectures of a memory module 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 (any word in the target sequence communicates with any word 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 in 2019.

Attention in NLP

- Illustrations of attention :



  - Example 1 : MT
    - 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
- Sequence-To-Sequence Transformers
- Language Model Transformers
- Graph NNs vs Attention NNs
- Conclusion

Memory Networks

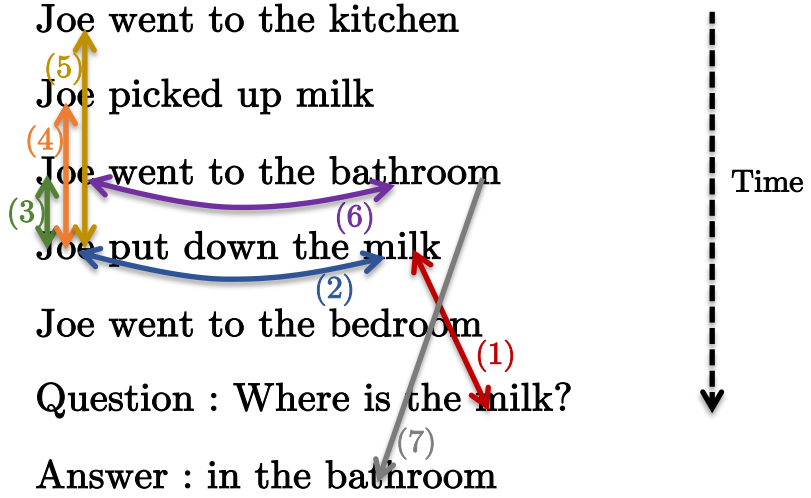
- [Bahdanau, Cho, Bengio'14] has a single-hop attention mechanism (1 layer of attention).
- Memory networks :
  - Differentiable Memory Computers.
  - Precursors of Attention Networks.
  - General idea is that intelligence requires an adaptive long-term memory.
  - [Weston, Chopra, Bordes, Memory networks'14], [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

- Example w/ 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 : ----

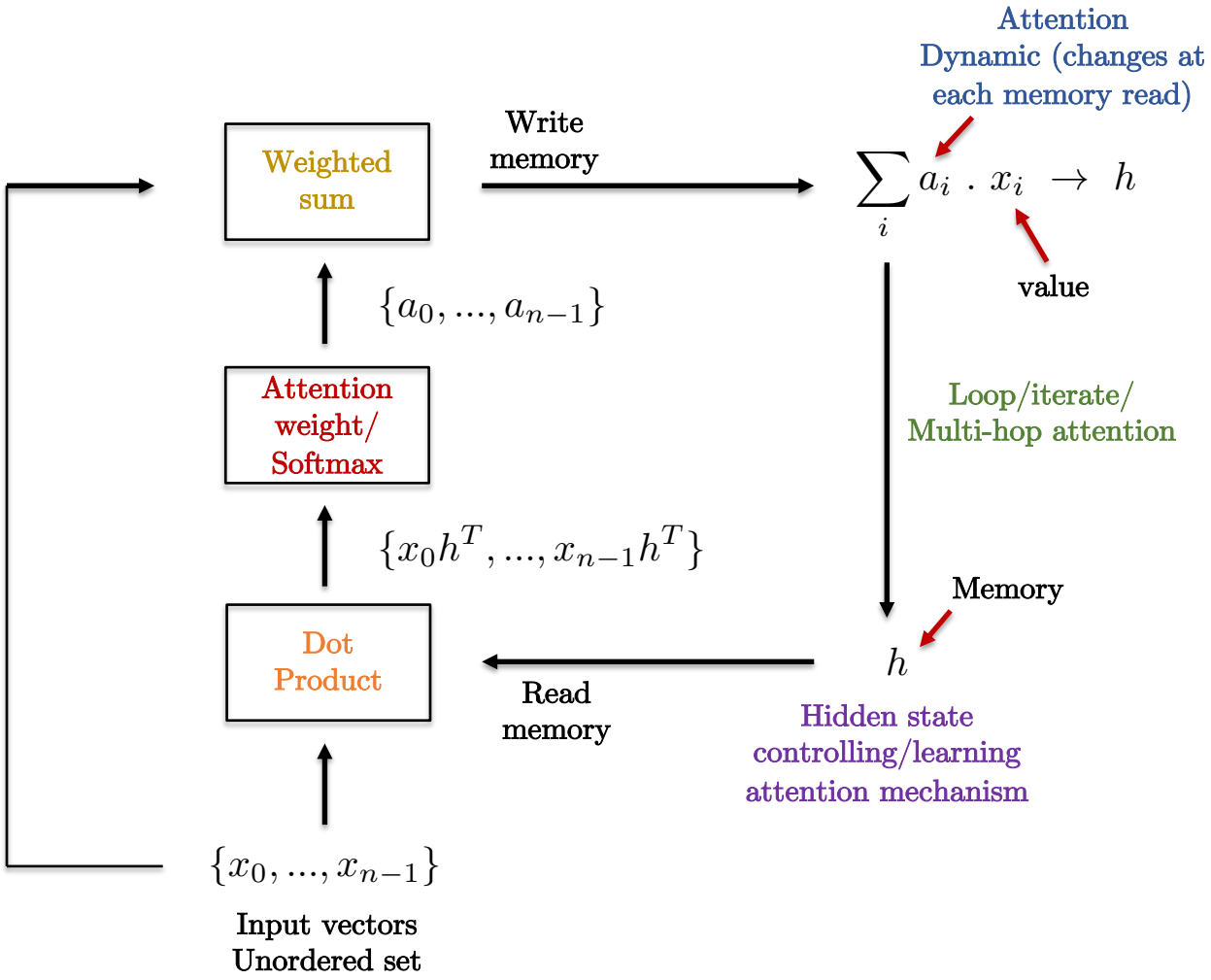
Memory Networks

- Attention networks will solve the problem with words 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 !
- [Weston, Chopra, Bordes, Memory networks'14] : Not fully end-to-end backpropagation.
- [Sukhbaatar, Szlam, Weston, Fergus, End-to-end memory networks'15]: Fully end-to-end backpropagation.

Memory Networks/Multi-Hop Attention



Memory Networks/Multi-Hop Attention

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

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

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

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

- Repeat K hops :

Softmax

$$a \leftarrow \sigma(Qh^T) \in \mathbb{R}^{n \times 1}$$
$$h \leftarrow a^T V \in \mathbb{R}^{1 \times d}$$

Outline

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

Memory Networks Limitations

- Memory network performances 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 sets in general.
  - Proposed improvements over memory networks :
    - Multiple hidden states (one per word)
    - Multi-head attention (more learning capacity)
    - Residual blocks (select specific attention layers, better backpropagation)

Transformer Layer

- Multiple hidden states :
  - Memory nets have one single hidden state h for all inputs x_i .
  - Transformers have one hidden state h_i for any single input x_i .
    - Each hidden state h_i is interacting with all other hidden states h_j .
  - This is a more powerful representation than memory networks as the inputs representation x_i are not fixed but can change for a better representation h_i that makes easier matching operations.

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 :

Softmax

$$A \leftarrow \sigma(HH^T) \in \mathbb{R}^{n \times n}$$
$$H \leftarrow AH = \sigma(HH^T)H \in \mathbb{R}^{n \times d}$$

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 :

Softmax

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

with

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

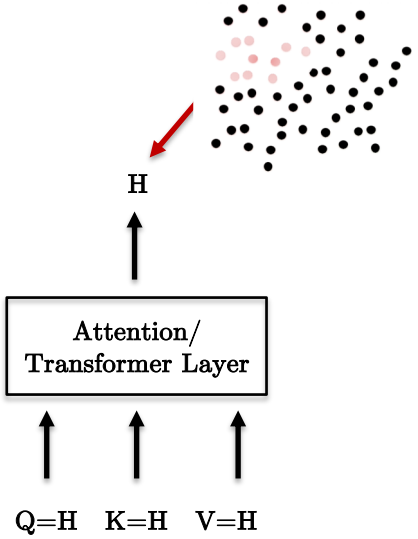
Differentiable dictionary
Dict is a standard structure in CS
dict=(query,key,value)

Context-To-Word Representation

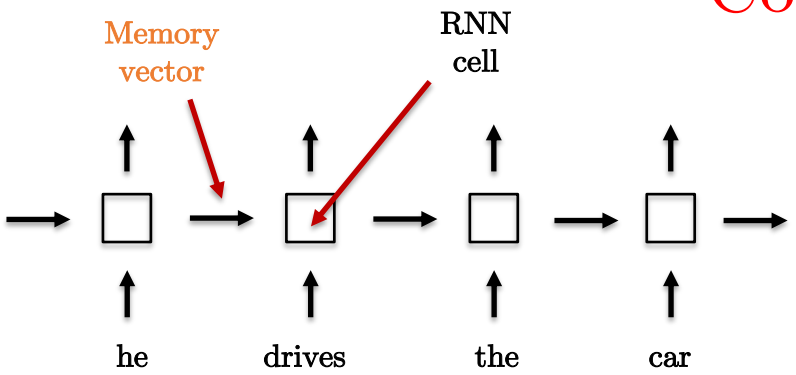
- From Word representation to Context-to-Word representation :

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

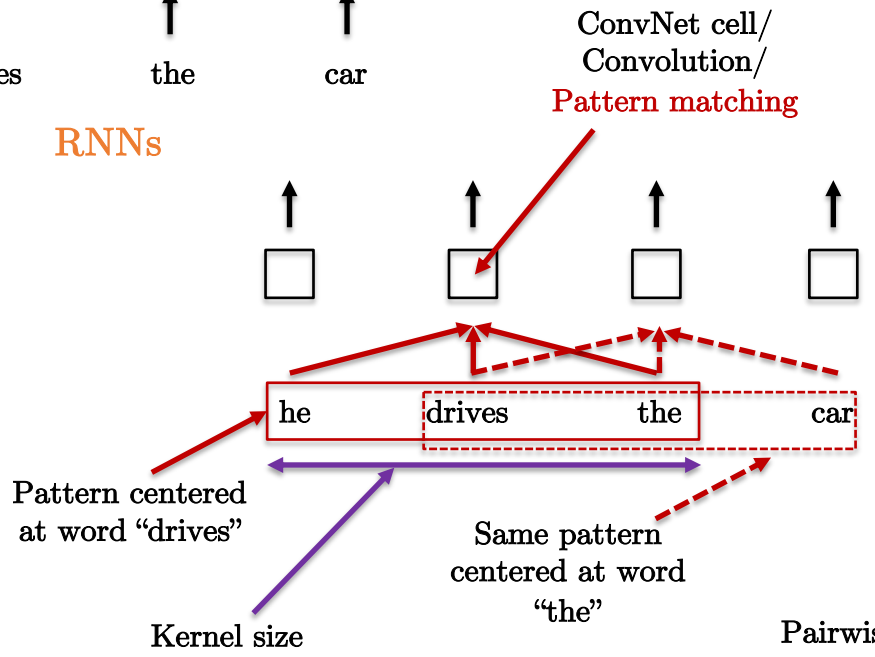
- The new representation of data is a sum of all input data representations weighted by the pairwise matching scores.
- The subset of selected data by attention forms the context of the new data.
- Attention mechanism allows to dynamically change the word representation and meaning according to its context (context-to-word representation).
- Context-to-Word is a powerful idea in NLP because a word may have a lot of different meanings, that can only be clarified in a specific 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.



Comparing Layers

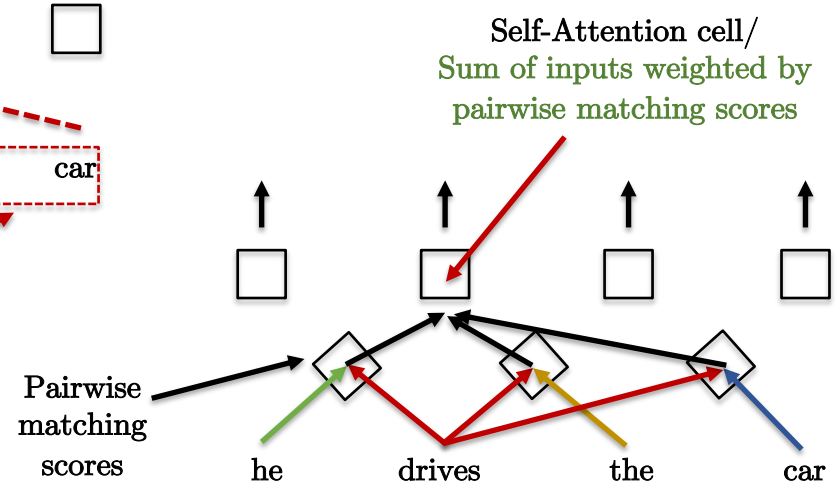


RNNs



ConvNets

All these layers can learn representation of sequences of different length/size and different context.



ANNs

Computational Cost

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

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

- Attention networks have actually less parameters as long as $d \geq n$!
  - 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: $O(10^9)$

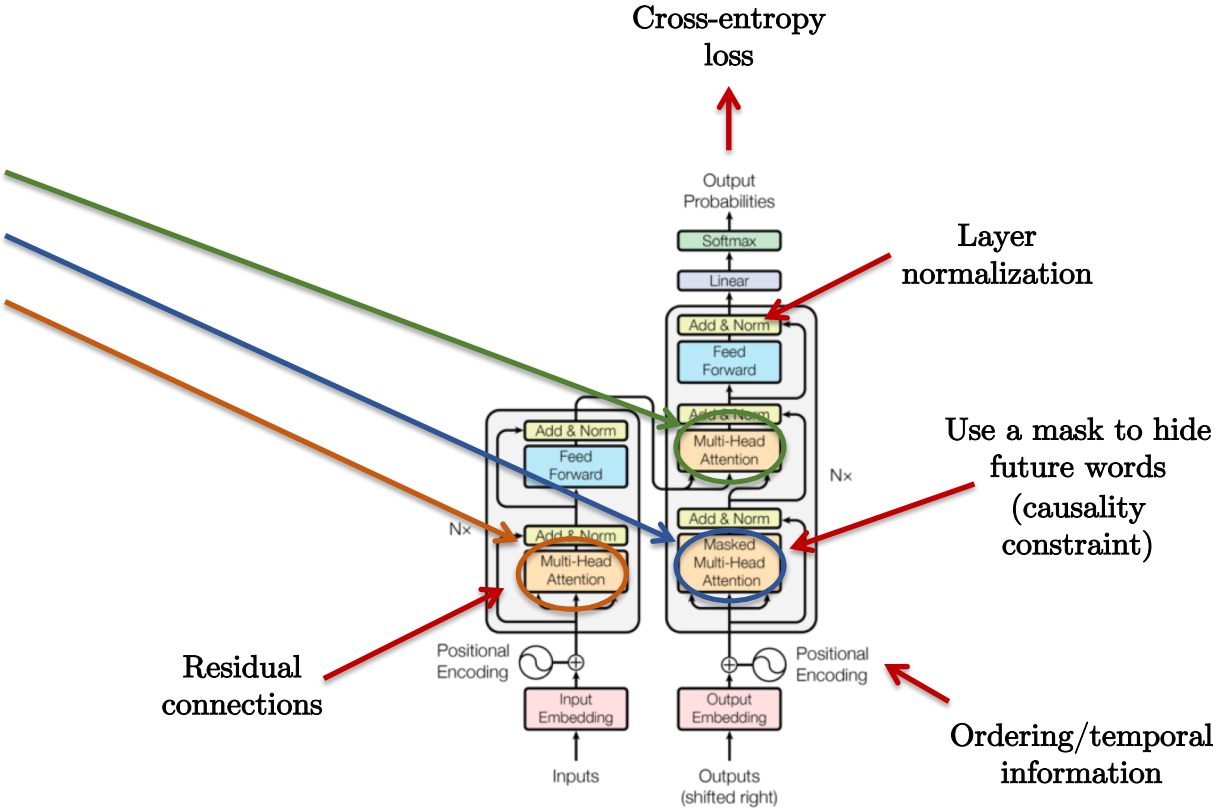
Outline

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

Seq2Seq Transformer

- Three types of transformer layers :
  - Transformer Encoder-Decoder
  - Self-Transformer Decoder
  - Self-Transformer Encoder

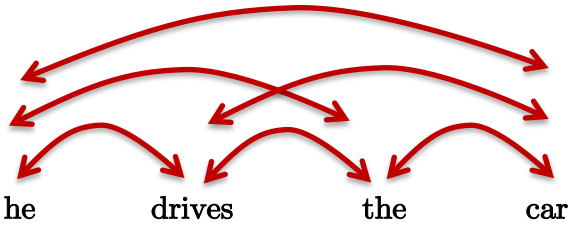
- A transformer layer is modular.
  - Not only used for seq2seq task.



Seq2Seq Transformer Architecture
Vaswani-et.al, Attention is all you need, 2017

Seq2Seq Transformer

- Self-Transformer Encoder :
  - Make attention to every pairs of words.
  - Soft-attention equations :

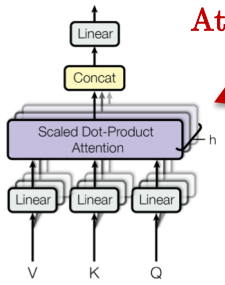


Multi-Head Attention

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{Head}_1,...,\text{Head}_h)W^O$$

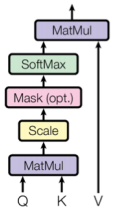
$$\text{where Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{and Attention}(Q,K,V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V \in \mathbb{R}^{n\times d}$$



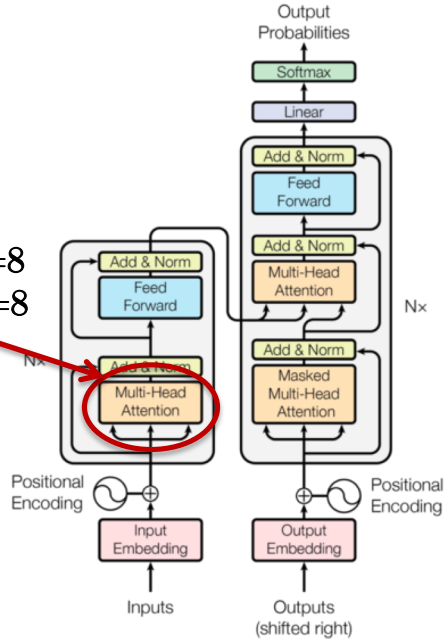
(Single) Head Attention

Normalization constant



Self-Transformer Encoder

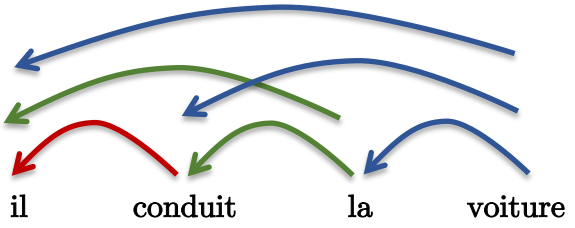
h=8
N=8

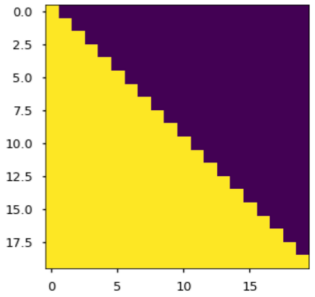


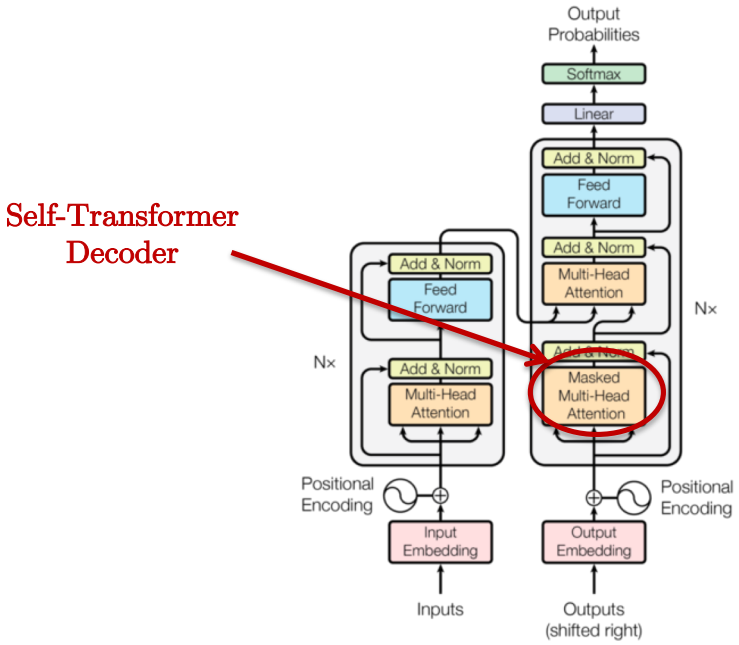
- Properties :
  - (Much) simpler layer than RNNs (matrix-matrix multiplications)
  - Parallelizable (no recurrence/sequential process)

Seq2Seq Transformer

- Self-Transformer Decoder :
  - Only make attention of words before the current world (causality constraint)
  - Use a mask to impose causality.
    - Value of matching scores QK^T for the mask is -10^9
      ⇒ Softmax will produce a zero value for these matching pairs.

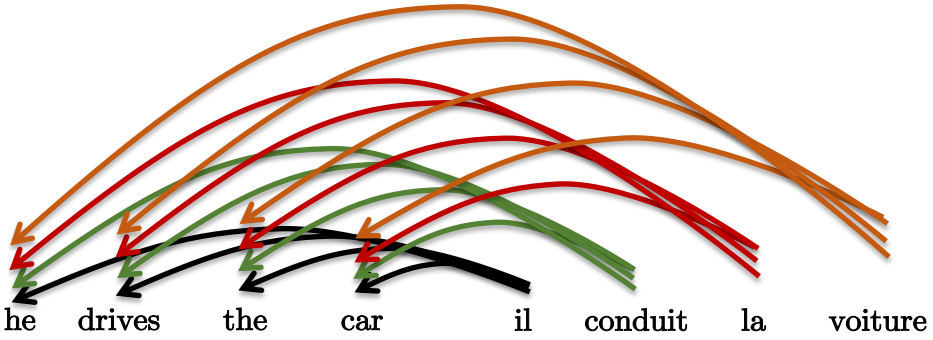






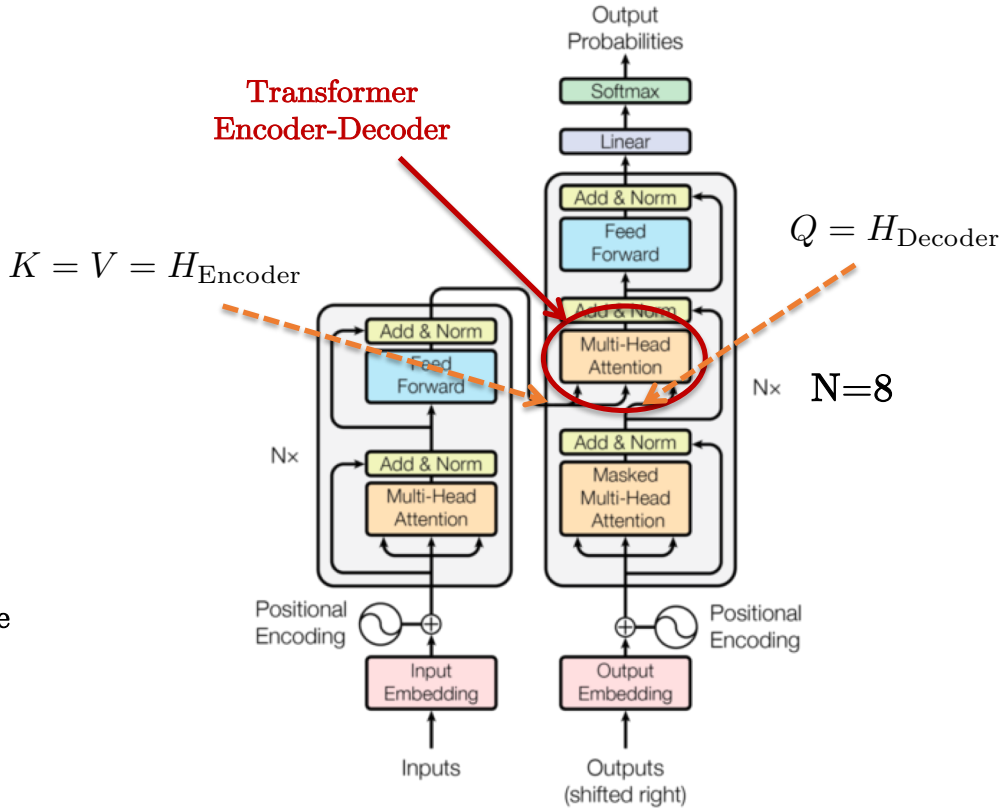
Seq2Seq Transformer

- Transformer Encoder-Decoder :
  - Make attention to pair of words between input/source sequence and output/target sequence.



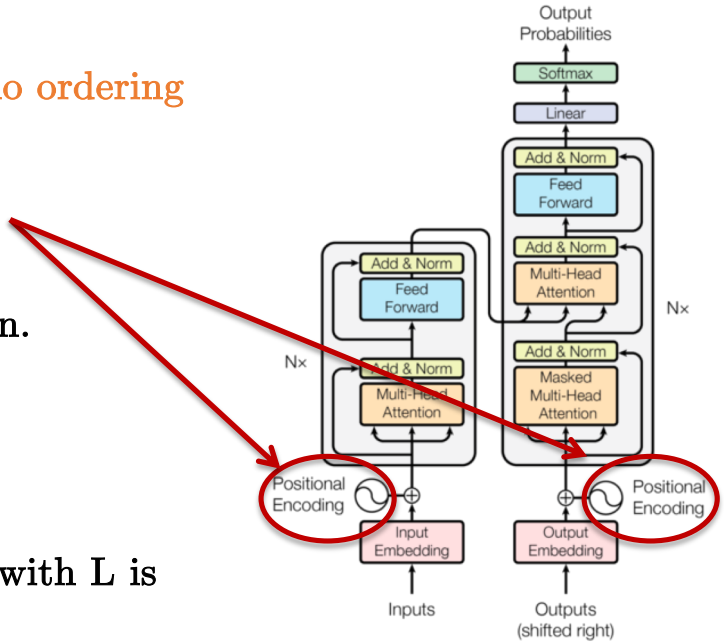
Source sequence
H_{Encoder}

Target sequence
$H_{Decoder}$



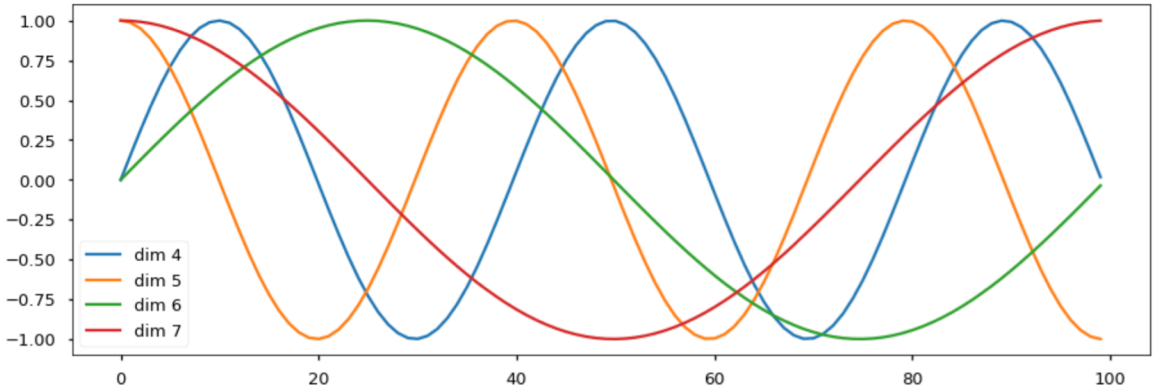
Positional/Temporal Encoding

- ANNs like Transformers are designed to process sets which have no ordering information about data.
- This is an issue for NLP.
  - An additional ordering feature is needed.
  - Positional features are introduced to add temporal information.
  - Two ways to inject positional information :
    - Learning vs non-learning positional features
    - Learning positional features :
      - 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/Temporal Encoding

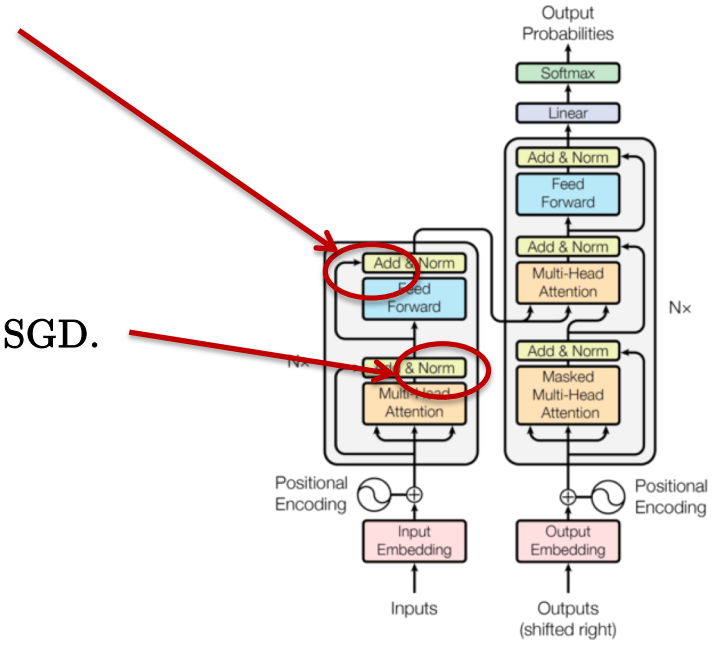
- Non-learning positional features :
  - Continuous ordering with sin and cos functions.
  - Advantages :
    - No training necessary
    - Experiments show learning and non-learning positional features have same performances.
    - No need to know the maximum length in train set.
    - Test sequences may have lengths not present in the train set.



Residual Connections and Z-Scoring

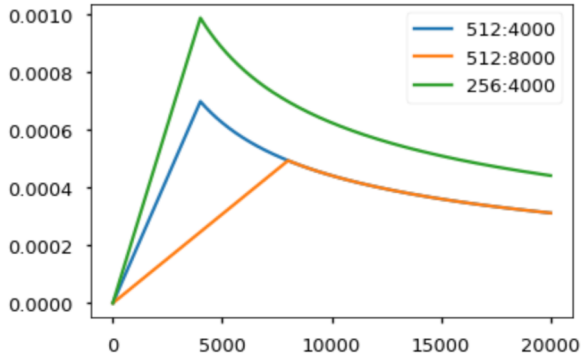
- Importance of skip connections/residual blocks :
  - Generic to NNs :
    - Good for backpropagation.
    - Allows to skip a few steps of reasoning in the k-hop attention mechanism if necessary.
  - Specific to Transformers :
    - Carry positional information to next layers.

- Normalization layer :
  - Simple z-scoring.
  - Optimization trick to flatten the loss landscape and speed-up SGD.



Training Tricks

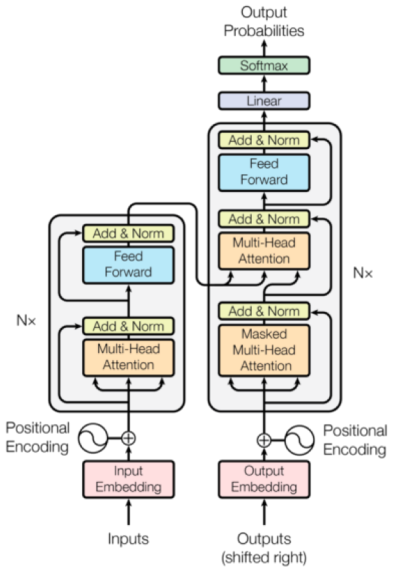
- Adams with special warming learning rate strategy.
- Regularizations :
  - Dropout
  - Label smoothing
- Decoding with beam search and length penalty.
- Ensemble by average.

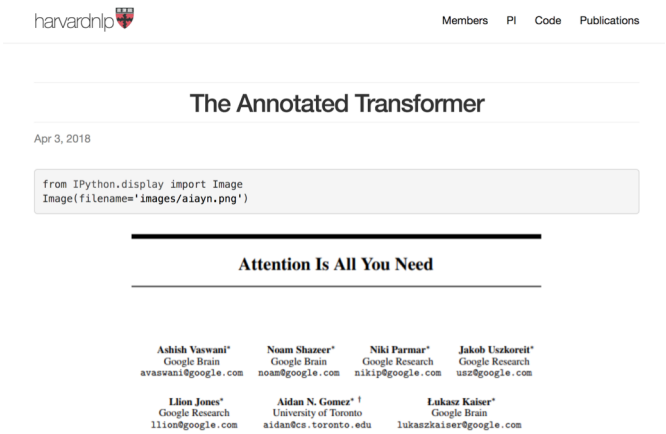


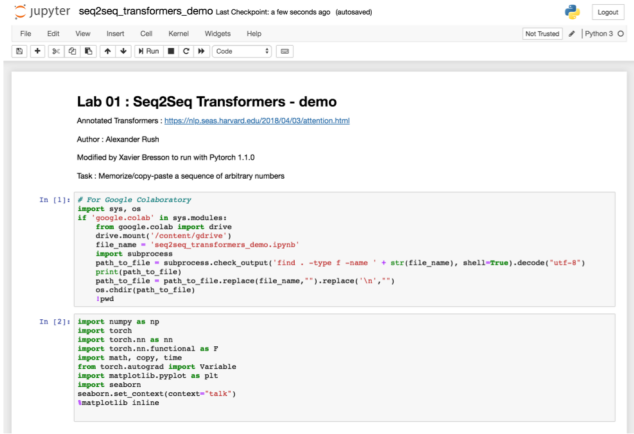
Lab 01

- PyTorch implementation of Seq2Seq Transformers.
  - The Annotated Transformers, Alexander Rush, April 2018

    https://nlp.seas.harvard.edu/2018/04/03/attention.html







Results

- 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
Transformer	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.

Human Level Translation

- Human 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.

Outline

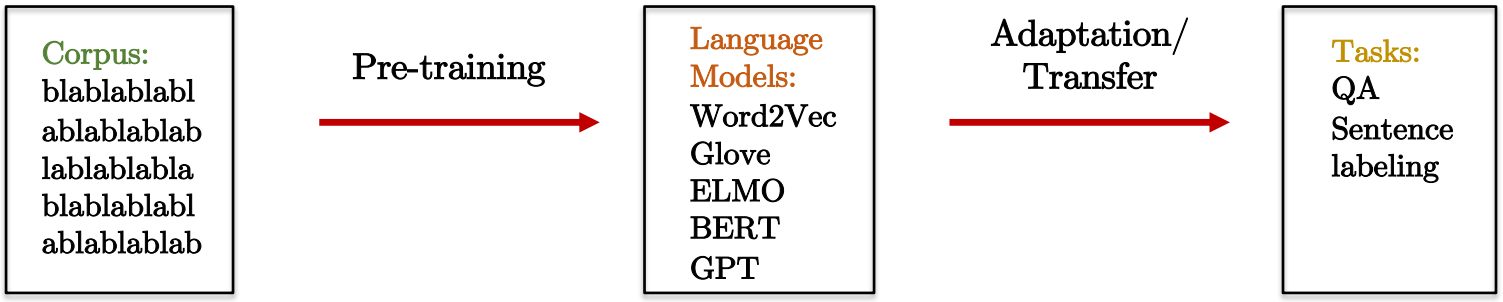
- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Sequence-To-Sequence Transformers
- **Language Model Transformers**
- Graph NNs vs Attention NNs
- Conclusion

Language Modeling

- Given a sequence of words, predict the next word in the sequence.
- Basic problem in NLP
  - Requires a world representation that can be flexible/change to different contexts.
    - From word representation to word-in-context representation.
  - To fully succeed, the model must understand languages !
    - Syntax, semantic, reasoning, common sense understanding of the world, etc.
- Attractive task because unsupervised/self-supervised task.
  - No need for labeled data.
  - Consequence : Big datasets for training are available.
    - Train with billions of words from Wikipedia, Reddit, etc.

Language Modeling

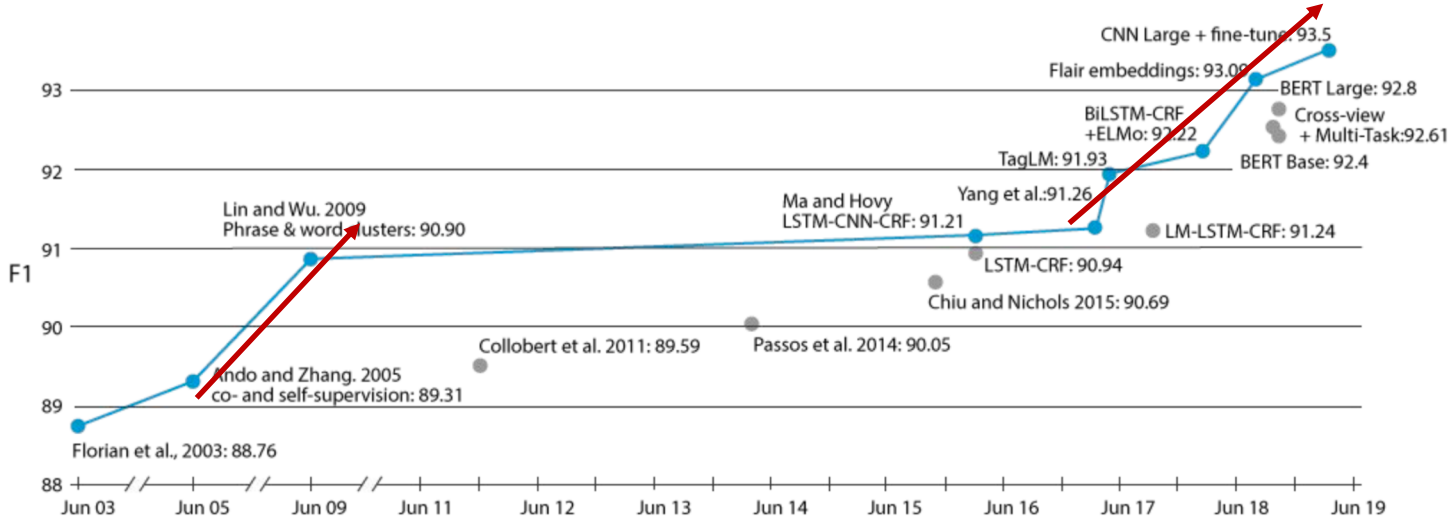
- A major theme in NLP'19 is sequential transfer learning :
  - Pre-trained language models on large-scale corpus (capture language prior) and
  - Transfer to new tasks s.a. document classification, Q&A, named-entity recognition, etc by fine-tuning layers of the pre-trained network.
  - Best performance for NLP tasks.



Sequential Transfer Learning

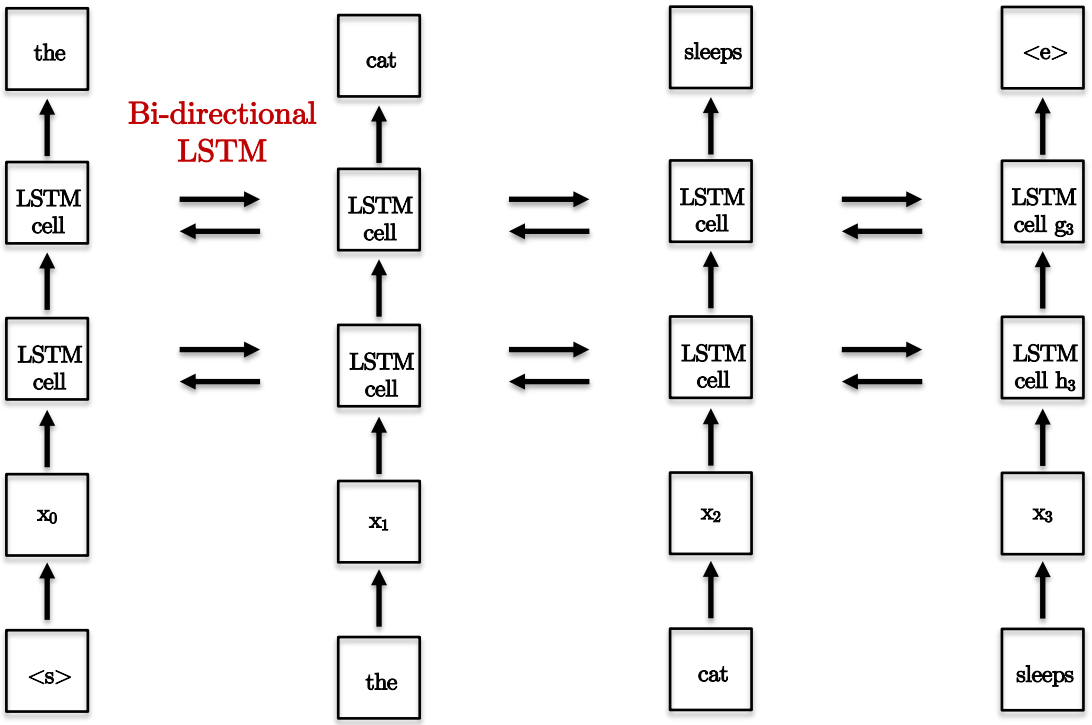
Language Modeling

- Usefulness of sequential transfer learning :
  - Many NLP tasks do not have labeled data.
    - Example : English-to-isiZulu
  - Transfer learning has offered performance progress.
    - Example : Named-Entity Recognition



ELMO

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



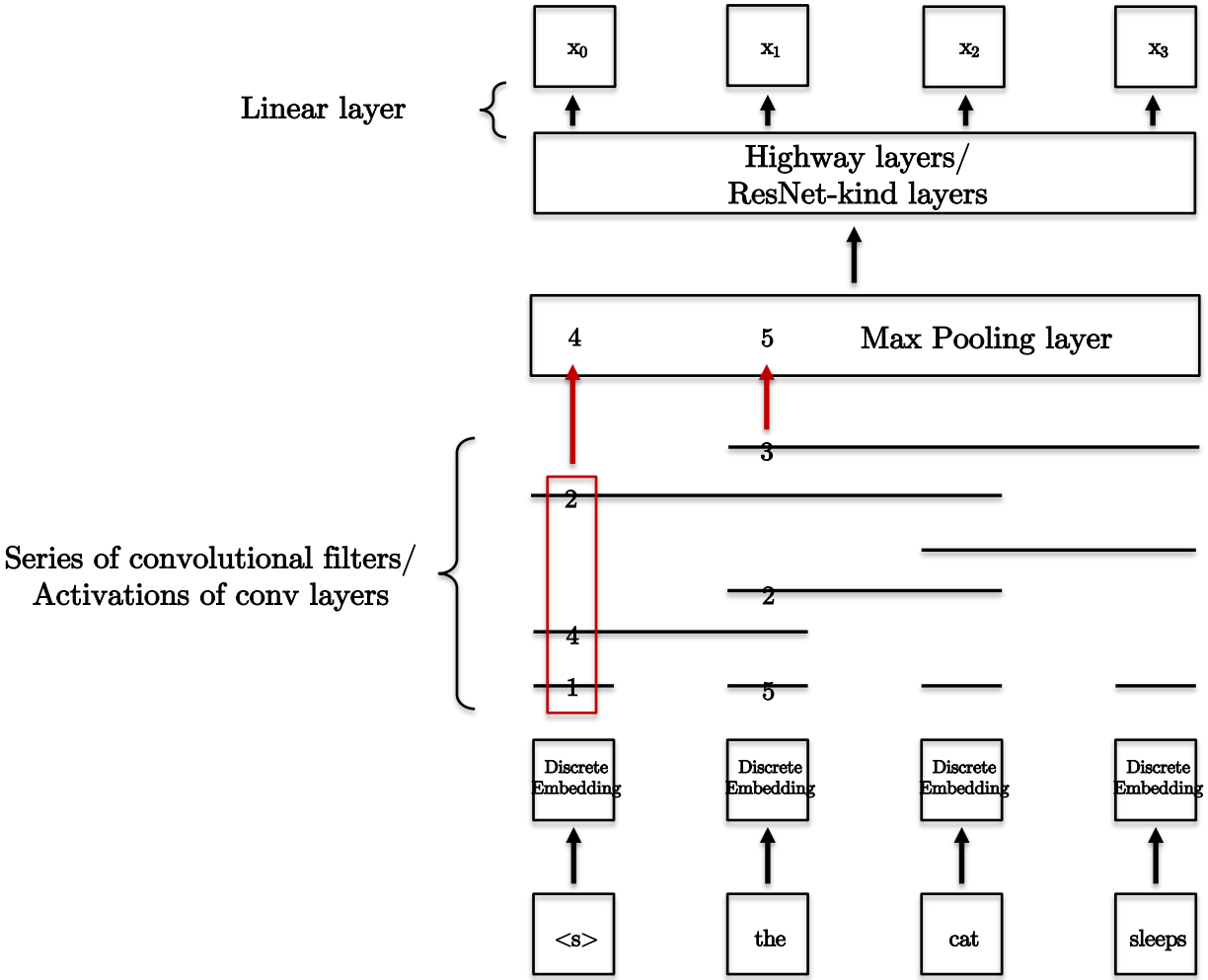
Transfer learning :
Represent word "sleeps" as a linear combination of x_3, h_3, g_3 :

$$\text{sleeps} = W \begin{bmatrix} x_3 \\ h_3 \\ g_3 \end{bmatrix}$$

Transfer weight (learned by backpropagation)

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

ConvNet World Embedding



ELMO

- ELMo showed to be very effective at transfer learning.
  - Demonstrates the importance of context-to-word representation.

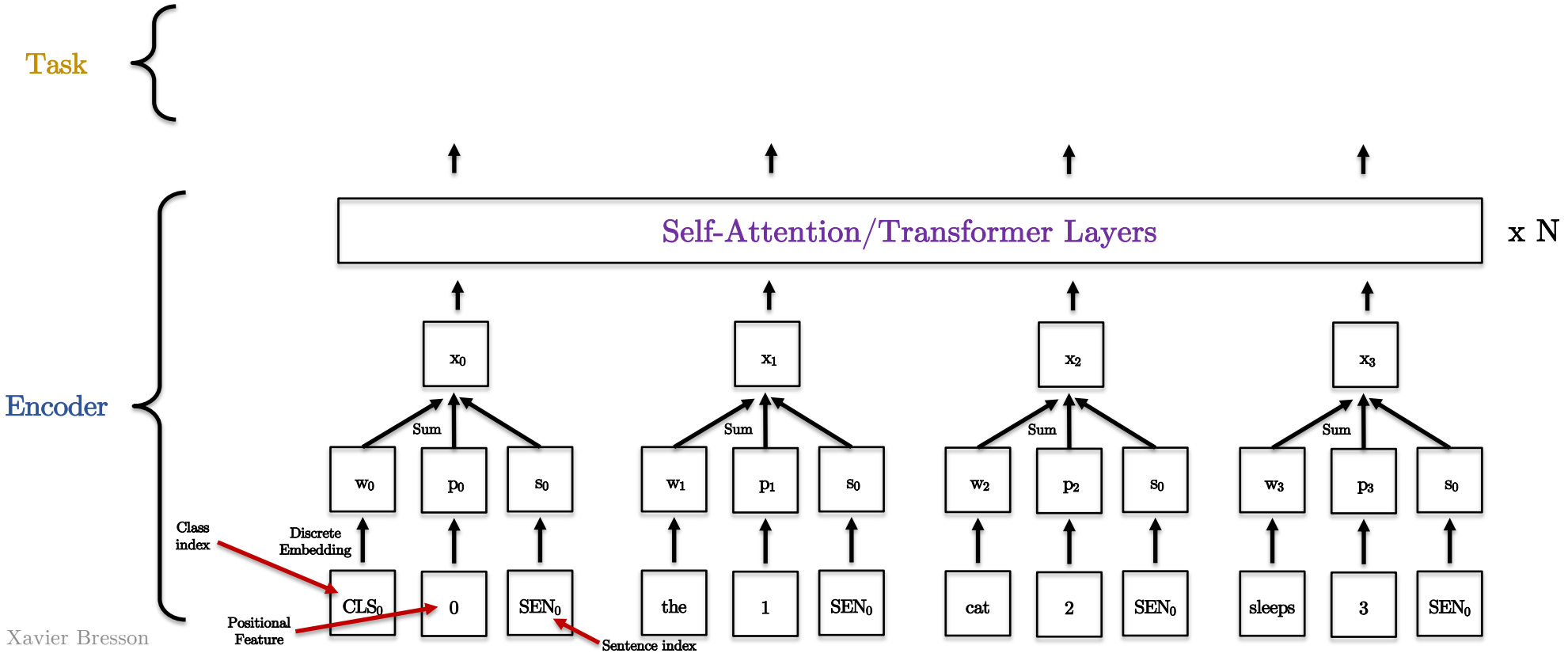
Use pre-trained context-to-word representation

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%

- ELMo is limited by LSTM/RNNs which cannot be parallelized (sequential technique) and cannot stack more than 2-3 layers.

BERT

- Bi-directional Encoder Representation for Transformers, Devlin'19, Google
  - Add class index and sentence index for training.

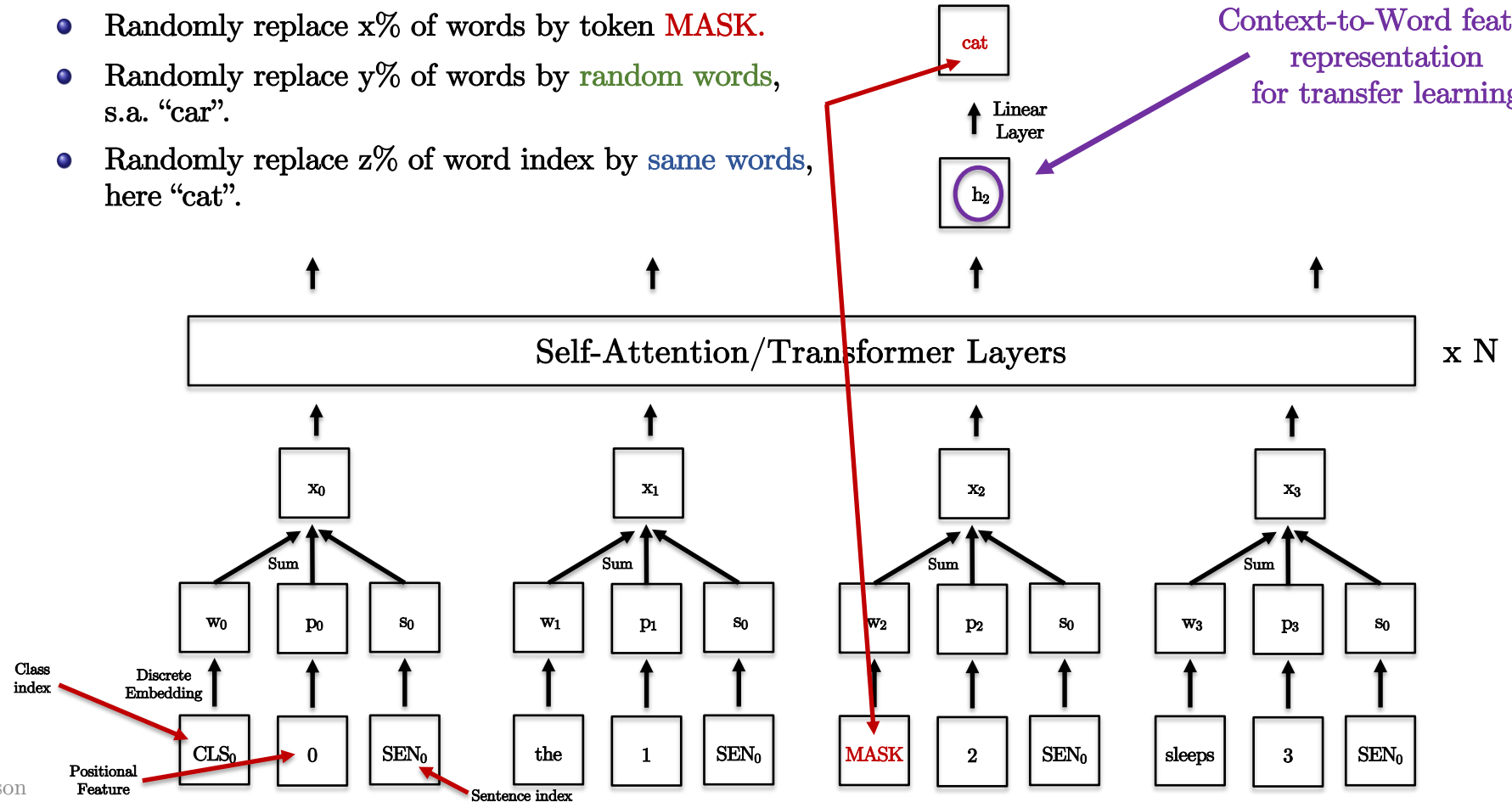


BERT

- Language modeling is a not an auto-regressive task unlike Machine Translation where the generation of the sequence is carried out one word at a time.
  - So, unlike MT, LM can look at the "future" (or any time) of the sequence to learn the word representation.
  - LM training is done differently than MT.
    - Word representation will be learned by masking words and use the context to predict the current word.
    - Similar idea to Word2Vec and Glove.
  - LM training has used two levels of hierarchical context :
    - Local dependencies with word prediction.
    - Global dependencies with sequence prediction.

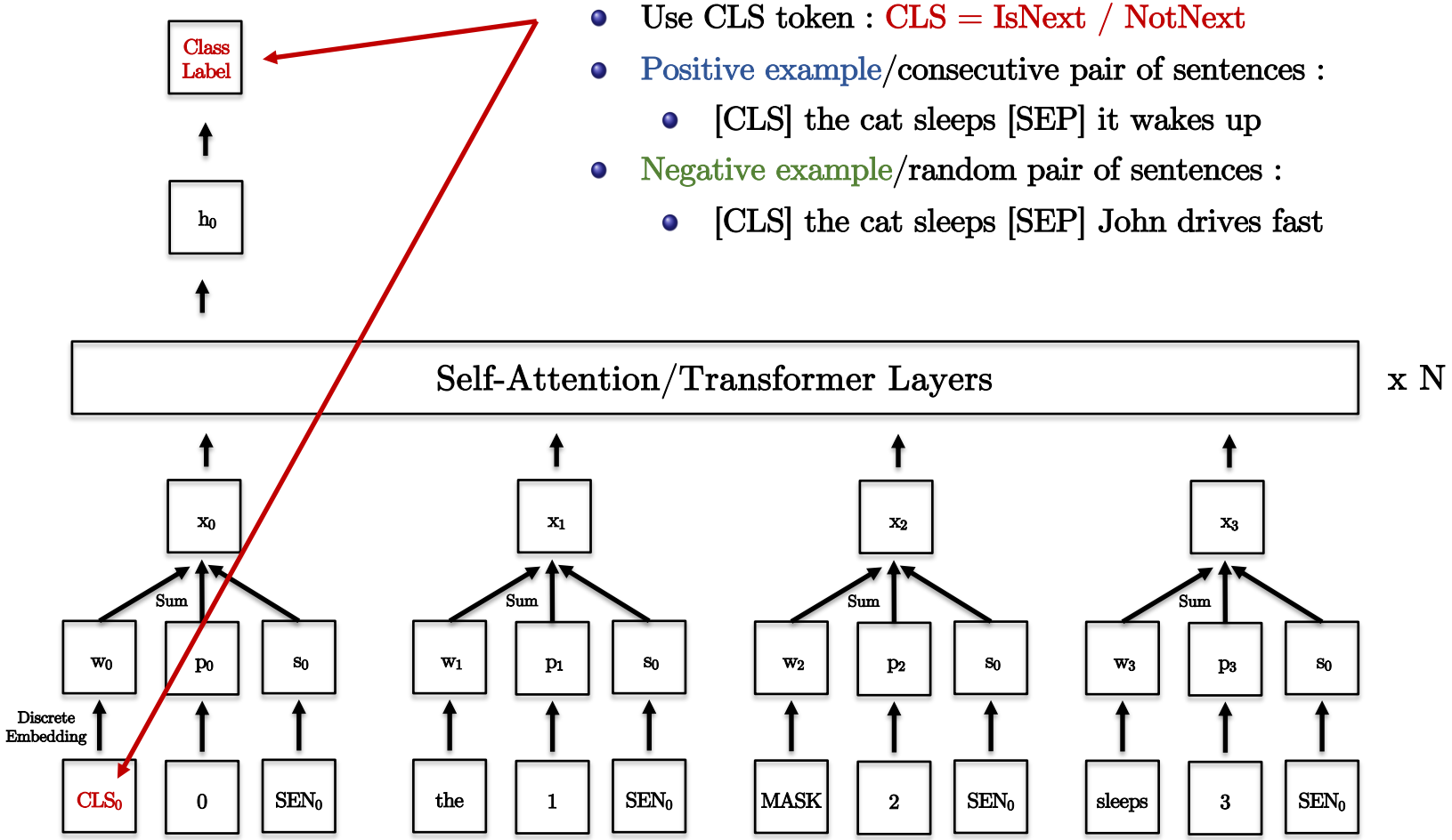
Masked Language Modeling – Local Context

- Train by predicting masked 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”.



Masked Language Modeling – Global Context

- Train by predicting 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



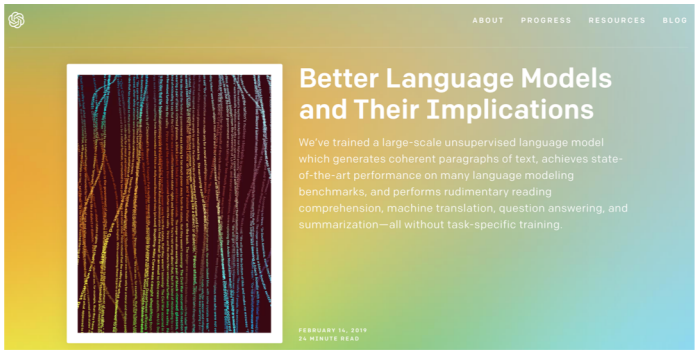
Training

- 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'19, 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 TPUs



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. ...

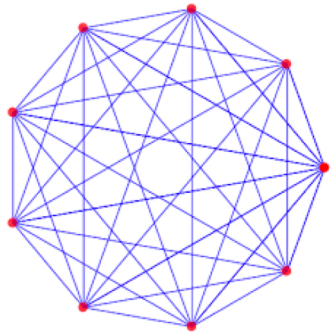
Outline

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

ANNs vs GNNs

- Connections with GNNs :
  - Attention encodes a graph.
    - A graph represents the pairwise relationships between data.
    - Attention precisely searches for these relationships.

- GNNs = ANNs when
  - Each data is connected to each other data.
    - Fully connected graph
  - Or, attention is specifically selected to a subset of data.
    - Sparse/Graph Transformers



Outline

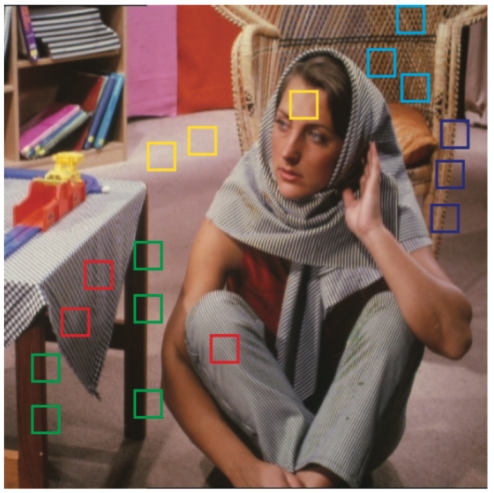
- Neural Networks
- Neural Networks for Sets
- Memory Networks
- Transformers
- Sequence-To-Sequence Transformers
- Language Model Transformers
- Graph NNs vs Attention NNs
- **Conclusion**

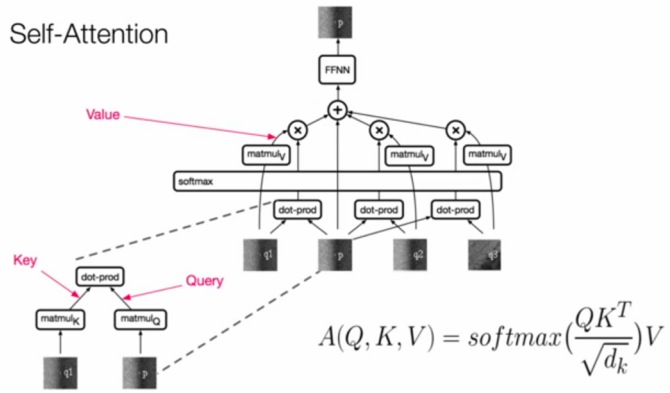
ANNs

- 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 data set, i.e. sets.
  - Good but also too generic !
    - It requires specialization to data/task.
      - For example, in Machine Translation, use a mask to hide future words and positional encoding for data ordering.
- Open questions:
  - Performance analysis
  - How to regularize attention mechanism?

Attention for CV

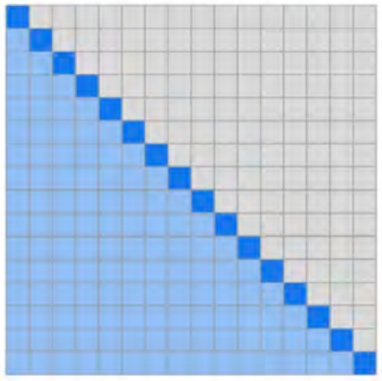
- Attention is "eating" deep learning :
  - Applications to NLP very successful.
  - Preliminary ideas of visual attention (image self-similarity) in Computer Vision :
    - Efros-Leung'99 for image generation.
    - Buades-Morel'05 for image denoising.
  - Issue of long sequences/image, here 3,072 pixels.
  - Recent papers on Image Transformer.



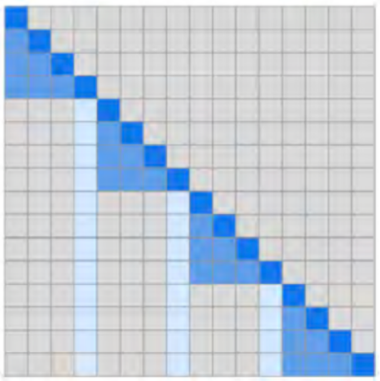


Current Limitations

- Highly computational cost for :
  - Transformer layers are functions of linear transformations.
  - Long sequence issue with $O(n^2d)$, n being sequence length and d hidden dimension.
  - Structured/Sparse Transformers Child-et.al'19



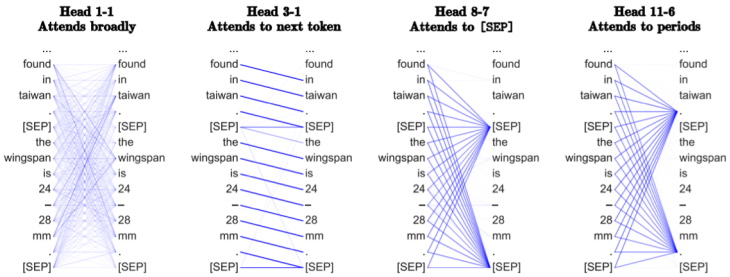
Original
Transformers



Sparse
Transformers

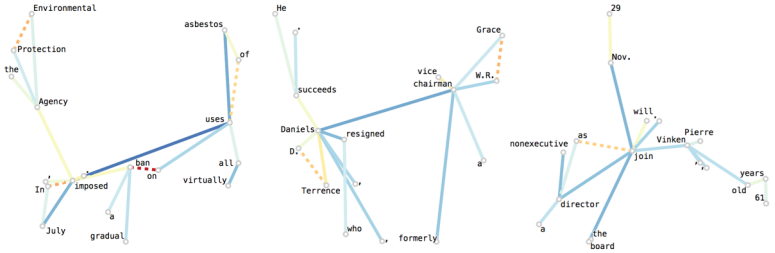
Transformer Visualization

- 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 :

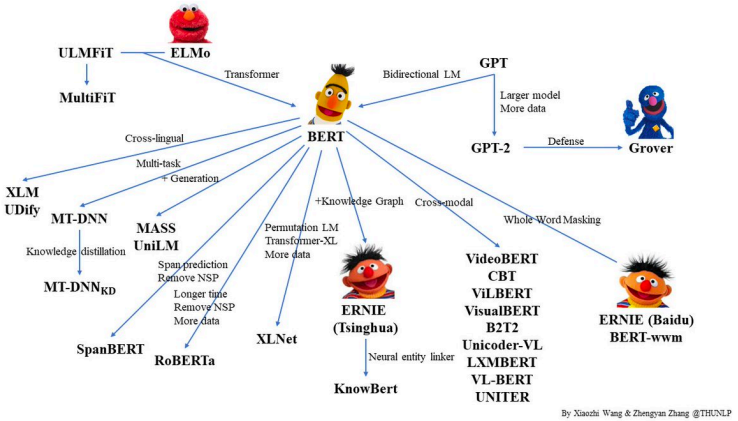
Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture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

- Sharing pre-trained models :
  - Hubs : TensorFlow, PyTorch
  - Codes + checkpoints : BERT, ELMO
  - Third party libraries : HuggingFace, Allen Institute, etc



- 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?