Contextualized embeddings

Prof. Luca Cagliero
Dipartimento di Automatica e Informatica
Politecnico di Torino

Politecnico
di Torino

Lecture goal

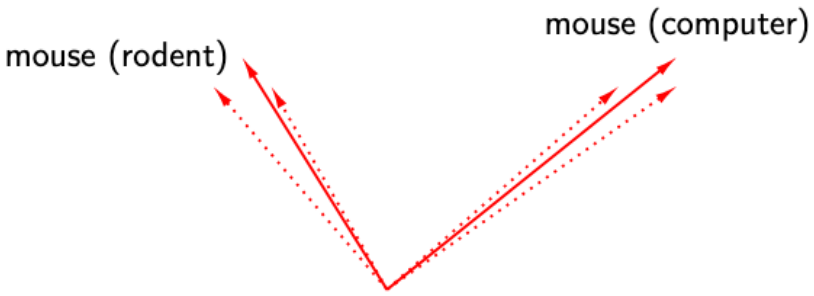
- Context matters
- Language models
- Deep contextualized word embeddings
- Seq2Seq model learning
- Directional embeddings
- Attention mechanism
- Transformer architectures

From word- to sequence-level

- More comprehensive text representation
  - Better capture the underlying semantic meaning

Context matters

- Examples of use of word *mouse*
  - The *mouse* is not working anymore
  - The cat ate the *mouse*
- Examples of use of word *cell*
  - the *cells* of a honeycomb
  - mobile *cell*
  - prison *cell*

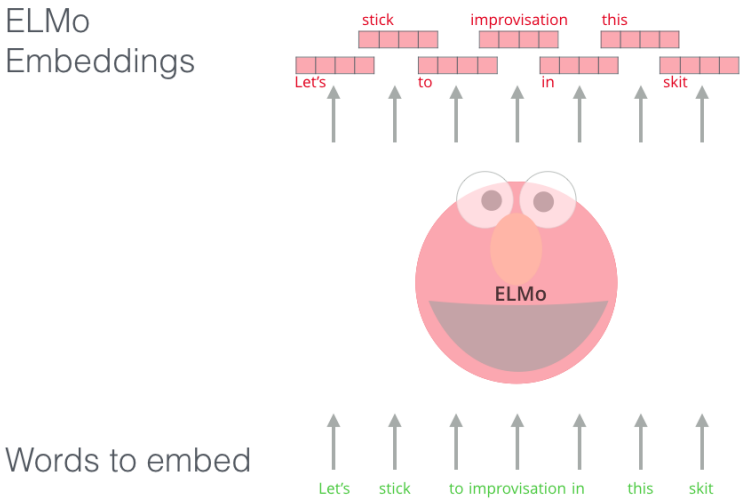


Limitations of word-level embeddings

- They ignore the role of context in triggering specific meanings of words
- They are unable to capture higher order semantic phenomena
  - E.g., compositionality and long-term dependencies
- They ignore positional information
  - E.g.,
    - I'm going to study Math instead of English
    - I'm going to study English instead of Math

Deep contextualized word representations

- Produce word representations from large text corpus
- Capture the word meaning in that context as well as other (potentially relevant) contextual information



Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

ELMo: Deep Contextualized Word representation

http://jalammar.github.io/illustrated-bert/

Deep contextualized word representations

- Model complex characteristics of word use
  - E.g., syntax and semantics
- Capture how these uses vary across linguistic contexts
  - E.g., to model polysemy
- The word vectors are learned functions of the internal states of a deep bidirectional language model
  - pretrained on a large text corpus

Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

Language models

- Forward language model
  - Given a sequence of N tokens (t_1, t_2, ..., t_N) compute the probability of token t_k given (t_1, t_2, ..., t_{k-1})

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$$

- Backward language model
  - Given a sequence of N tokens (t_{k+1}, t_{k+2}, ..., t_N) compute the probability of token t_k given (t_{k+1}, t_{k+2}, ..., t_N)

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N)$$

Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

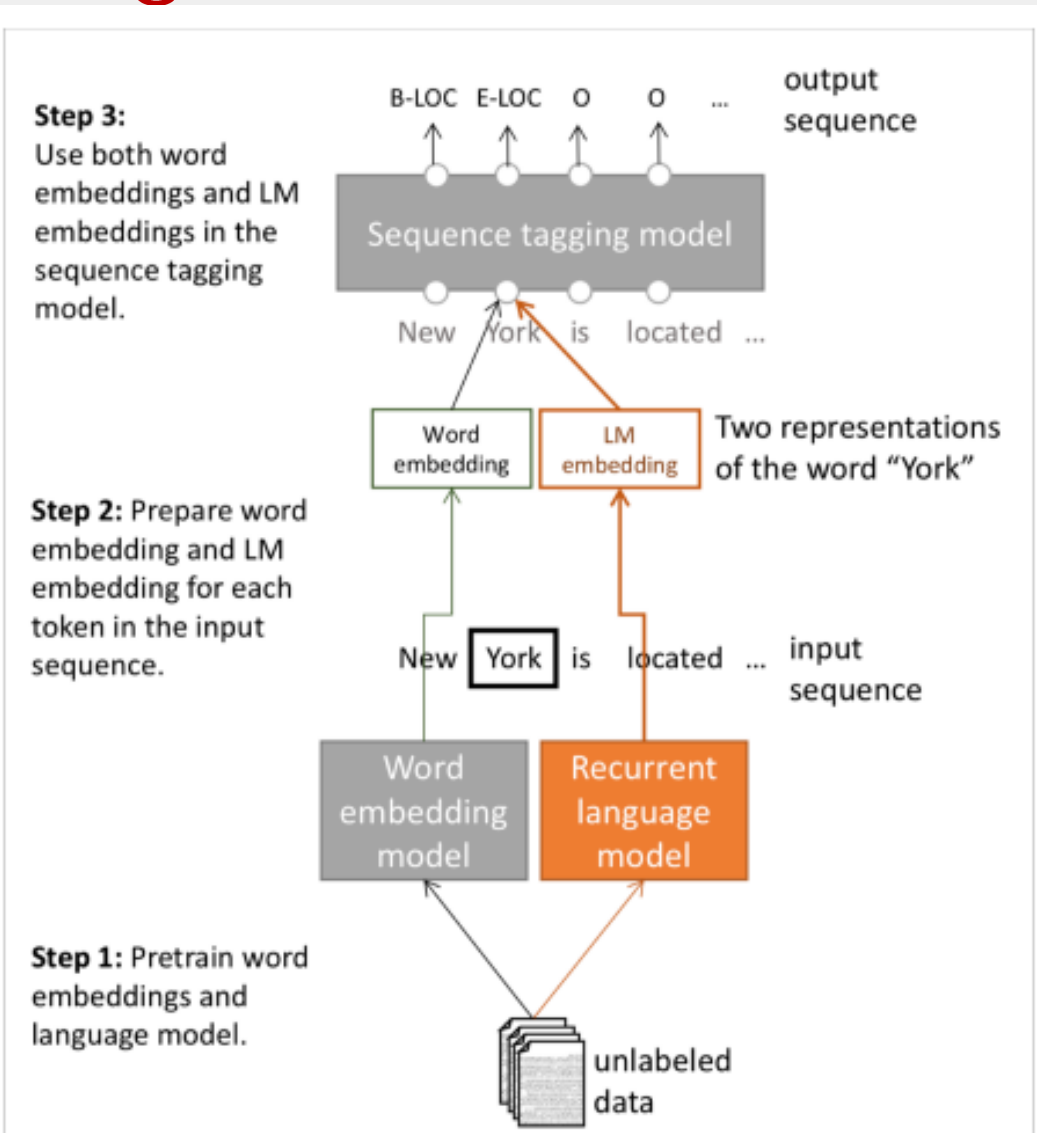
Language models

- Bidirectional model
- Train the forward and backward LMs jointly
  - with the same parameters for the token representations

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

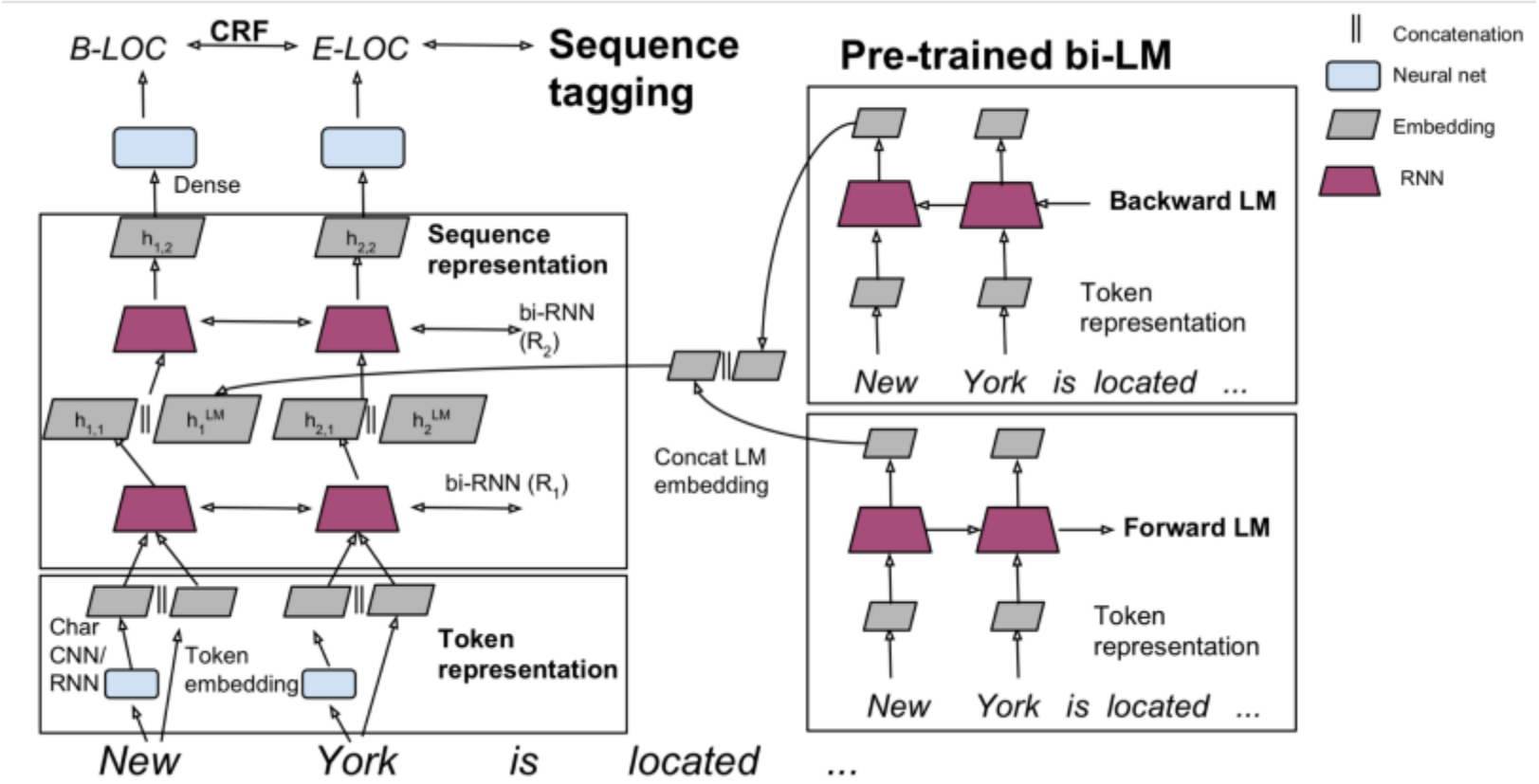
Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

Precursor: Tag LM



Semi-supervised sequence tagging with bidirectional language models. Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power. 2017

Precursor: TagLM

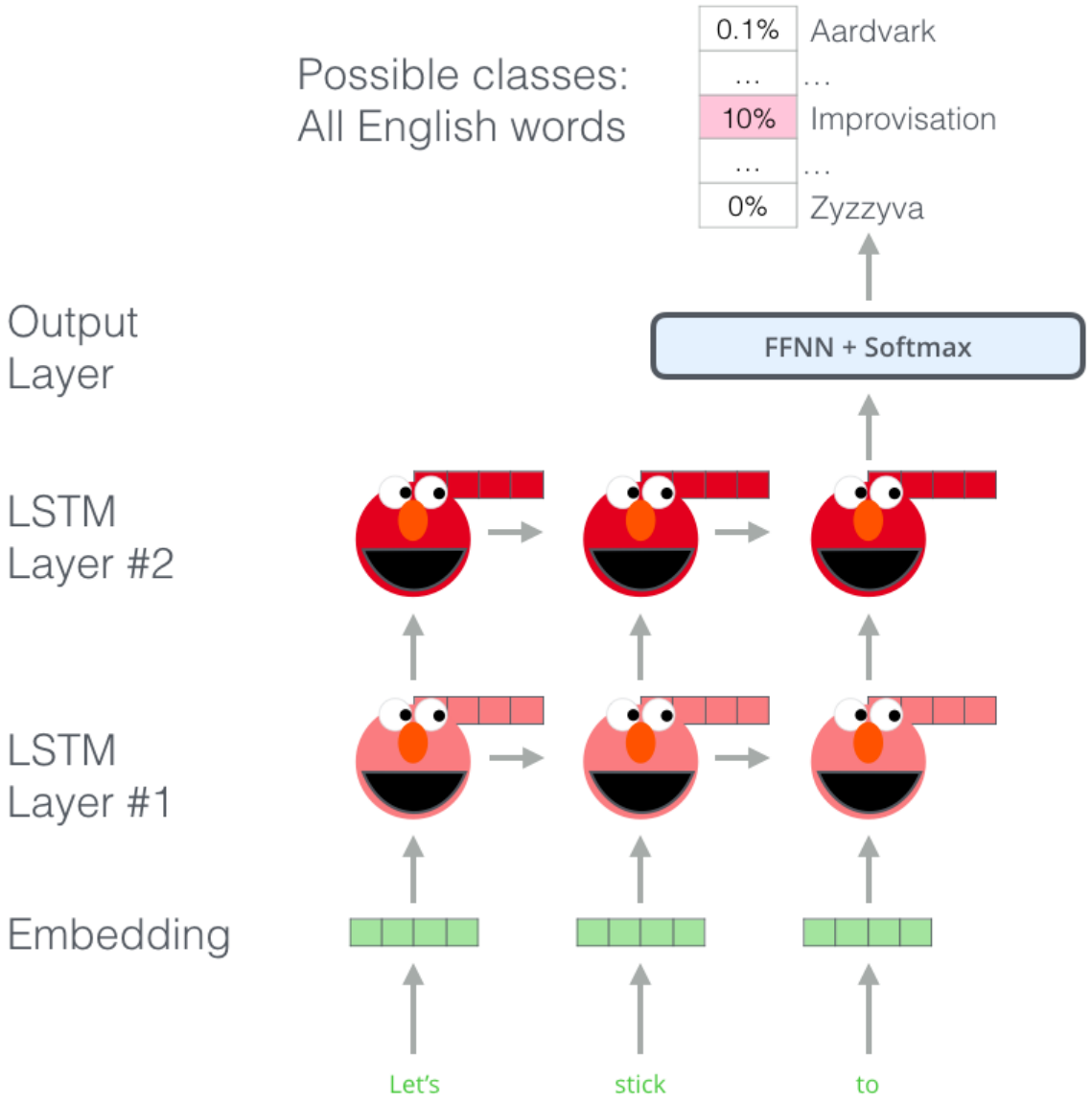


$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}].$$

Semi-supervised sequence tagging with bidirectional language models. Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, Russell Power. 2017

ELMo

- Input text
  - *Let's stick to*
- Bi-LSTM
  - Have the sense of both the next and the previous words



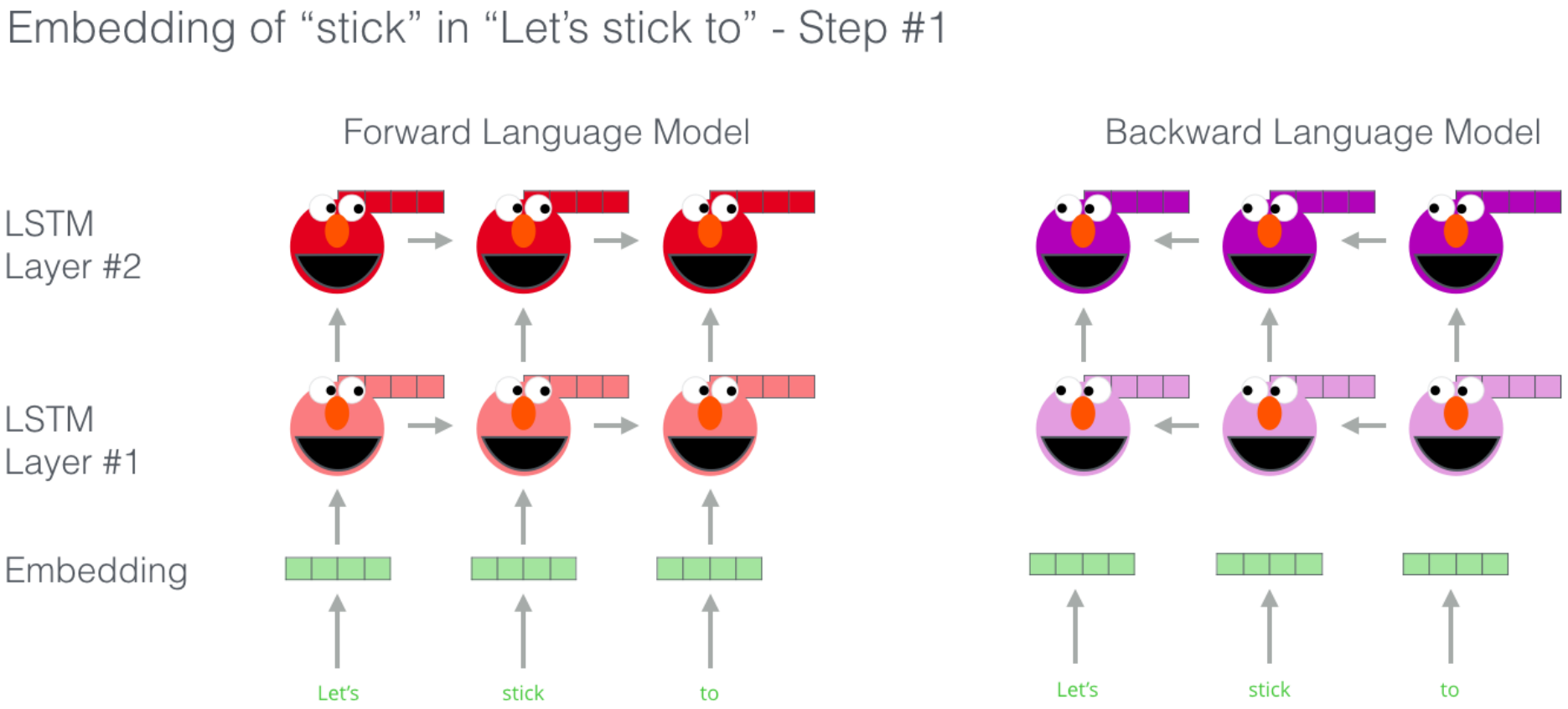
http://jalammar.github.io/illustrated-bert/ (latest access: June 2021)

ELMo

- Embeddings from Language Models (ELMo)
- Replace static embeddings (lexicon lookup) with context-dependent embeddings
  - produced by a deep neural language model
- Each token's representation is a function of the entire input sentence
  - computed by a deep (multi-layer) bidirectional language model
- Return for each token a (task-dependent) linear combination of its representation across layers
  - Different layers capture different information

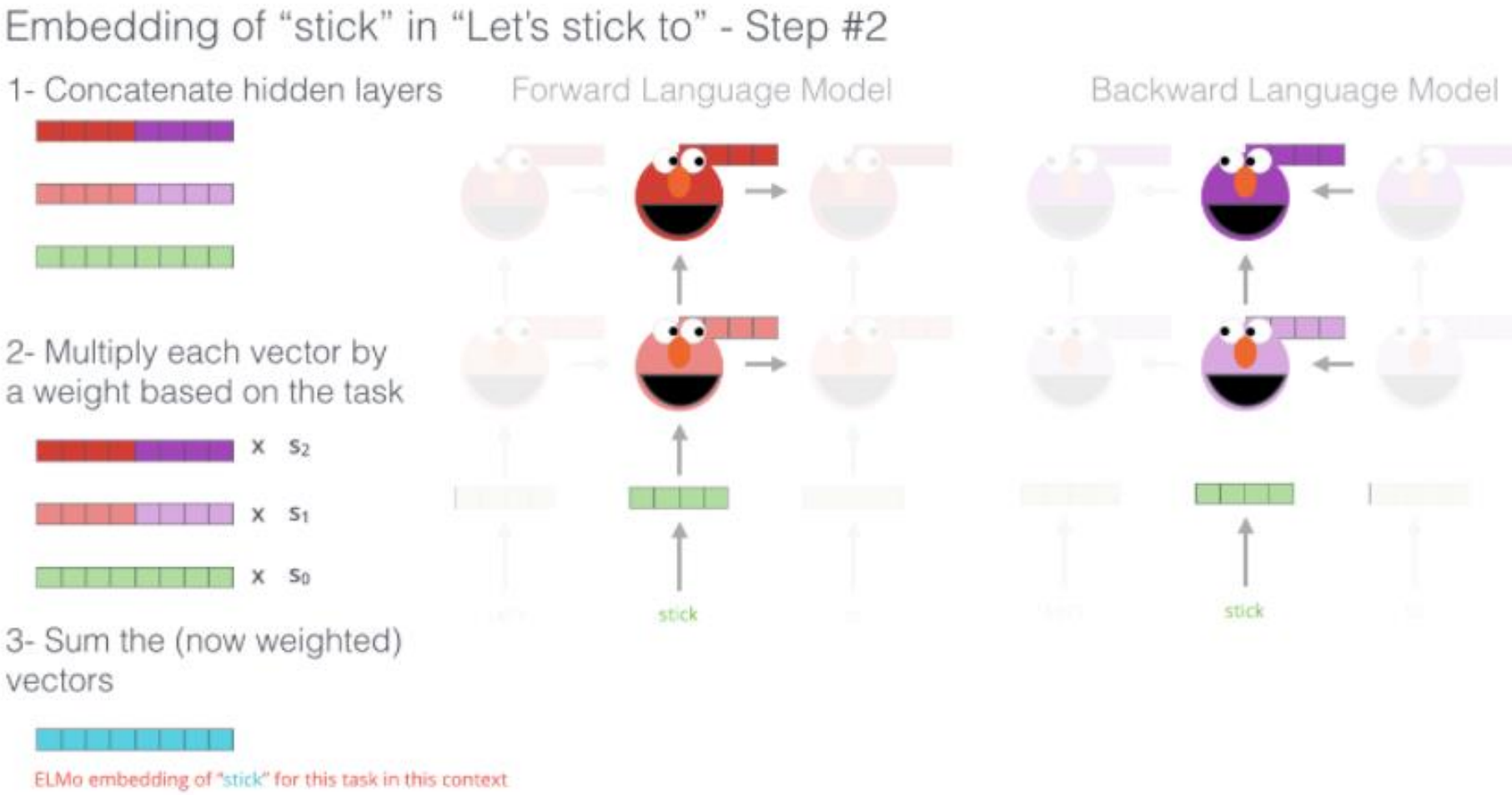
ELMo

- Bidirectional LSTM
  - Have the sense of both the next and the previous words



http://jalammar.github.io/illustrated-bert/ (latest access: June 2021)

ELMo



http://jalammar.github.io/illustrated-bert/

Deep contextualized word representations

1. Run biLM to get representations for each word sensing previous and next words in the sentence
2. Let (whatever) end-task model use them
3. Freeze weights of ELMo for purposes of supervised model
4. Concatenate ELMo weights into task-specific model (details depend on task)
   - Train task-dependent softmax weights to combine the layer-wise representations into a single vector

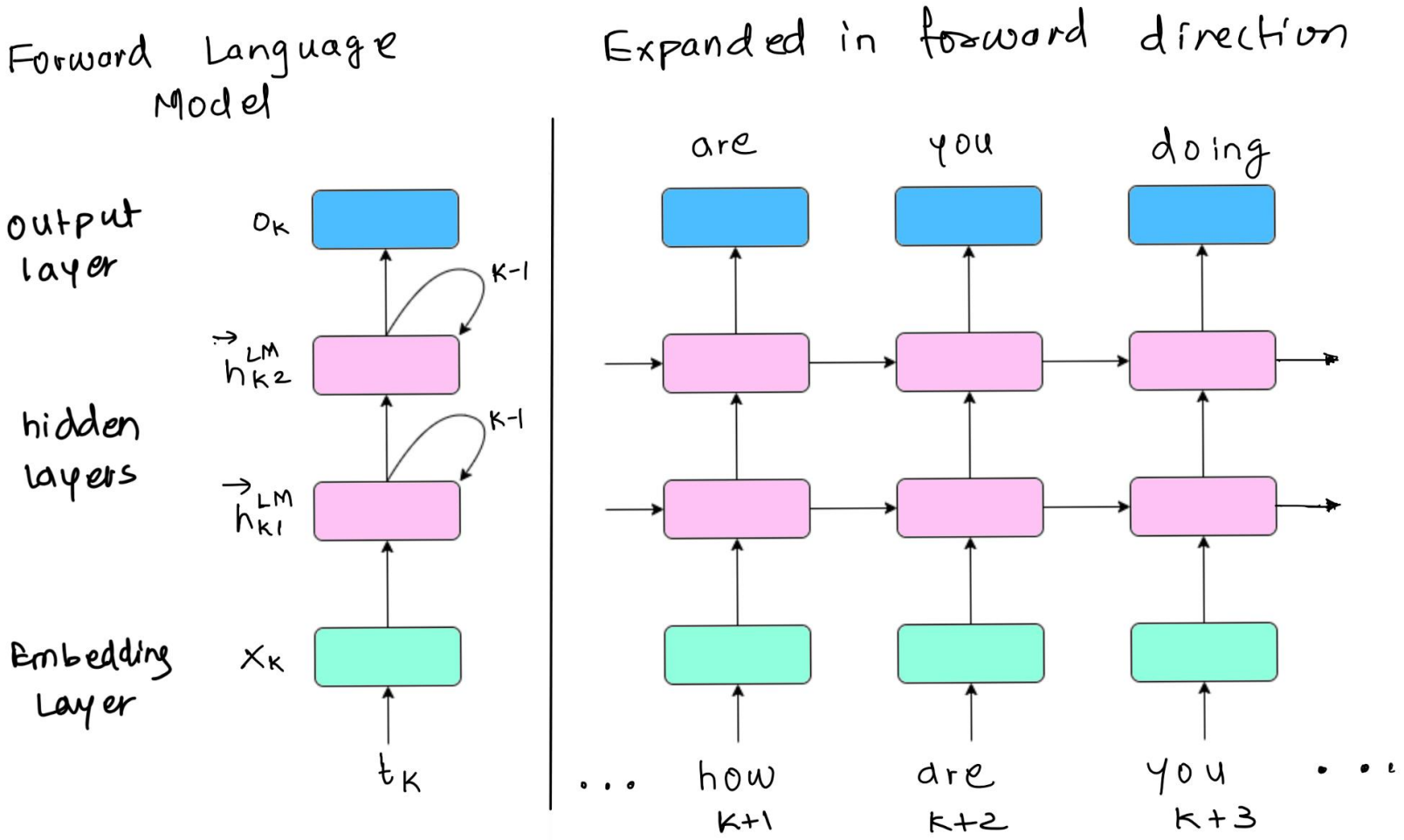
Deep contextualized word representations

- Once pretrained, the biLM can compute representations for any task
- Fine tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance

Step 1: ELMo pretraining

- Train the model to predict the next/previous word given a sequence words
  - Forward and backward language modeling
  - Each hidden layer is a bidirectional LSTM
  - The hidden states have access to both previous and next words
- Once the forward and backward language models have been trained, ELMo concatenates the hidden layer weights together into a single embedding
- No need for explicit labels as needed in other supervised learning tasks

Step 1: ELMo pretraining



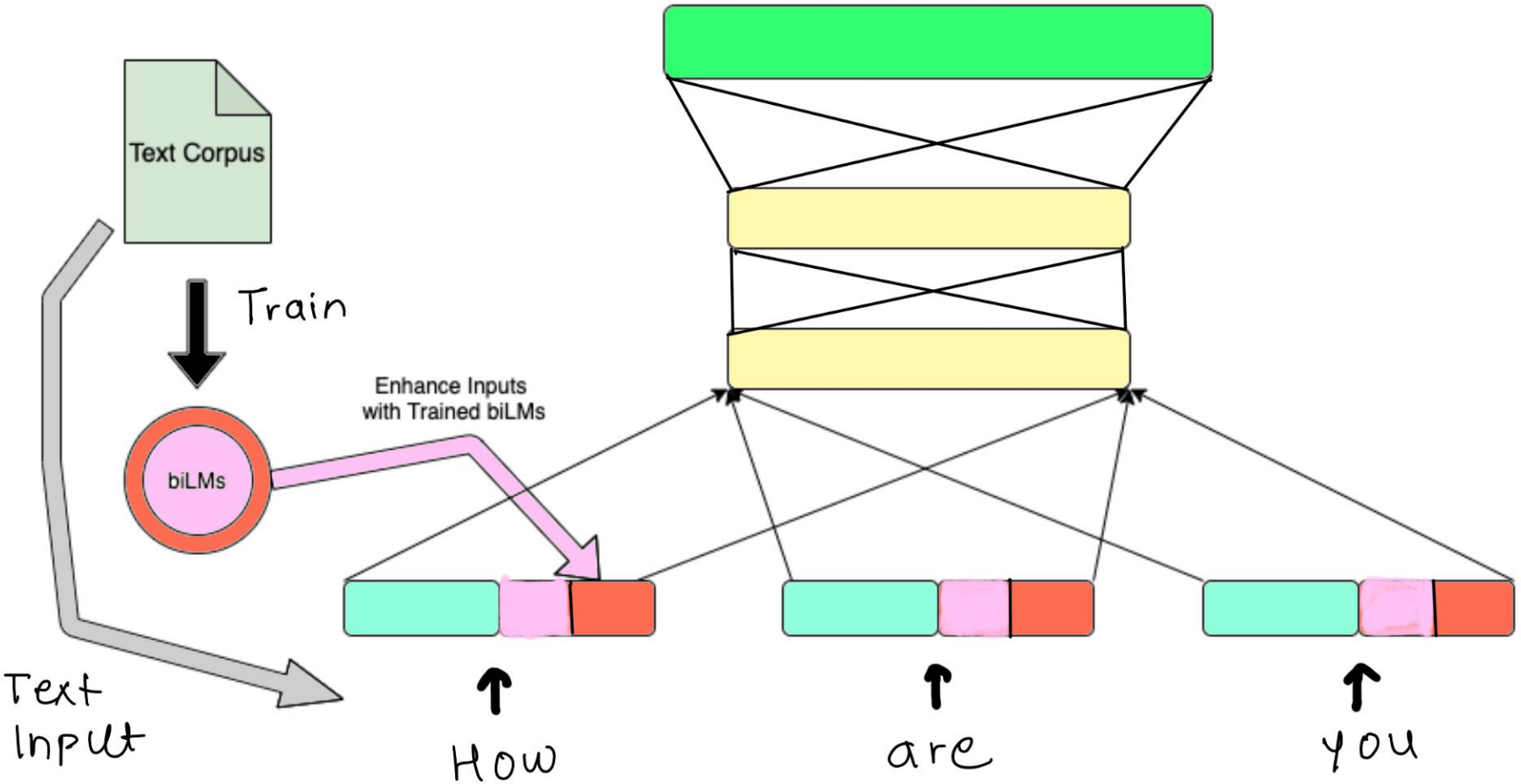
Unrolled Forward Language Model used in ELMo. Mandar Deshpande.

Steps 2-4: ELMo fine-tuning

- Specialize the word embeddings to the task under analysis
  - Question answering
  - Sentiment analysis
  - Named entity extraction
  - Semantic role labeling
  - Conflict resolution

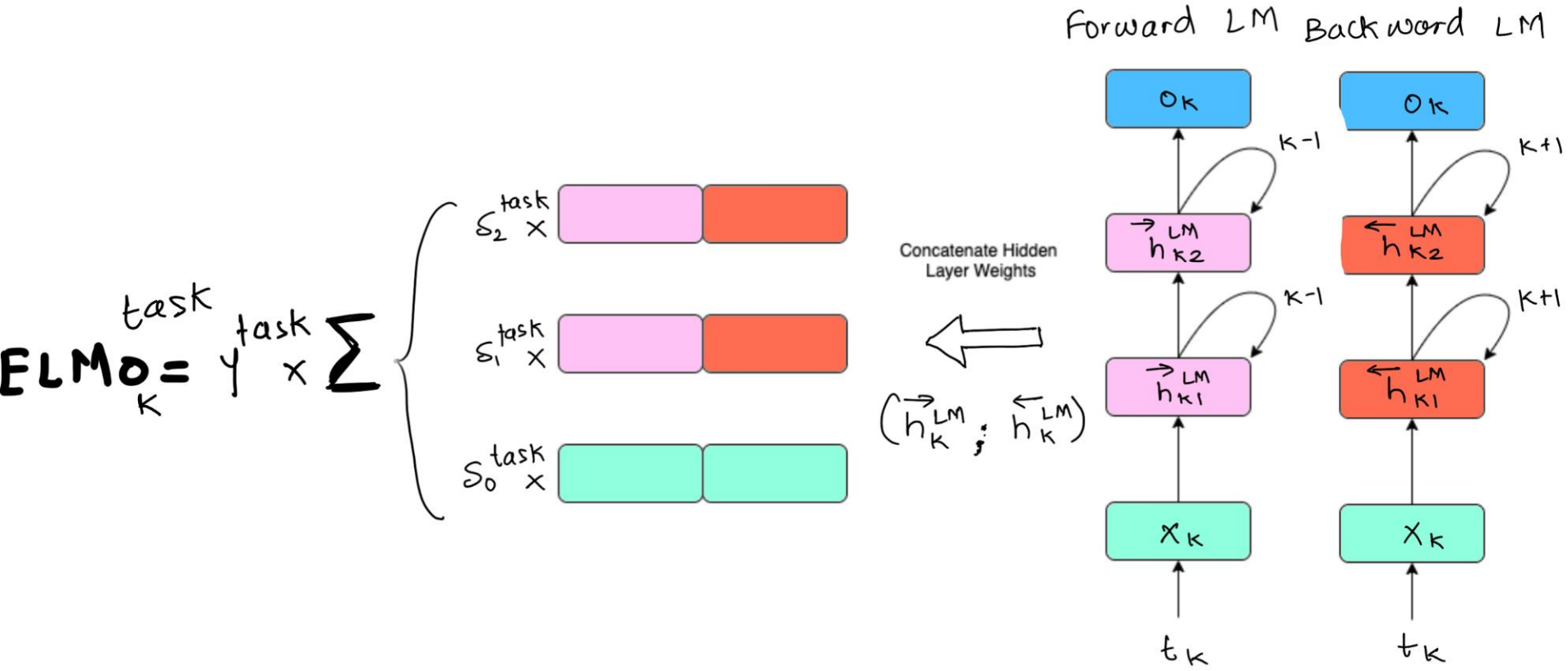
  (More details can be found at https://arxiv.org/pdf/1802.05365.pdf)
- Weight concatenation is multiplied with a weight based on the task being solved
- ELMo represents a token t_k as a linear combination of corresponding hidden layers (including its embedding)

ELMO: steps 2-4



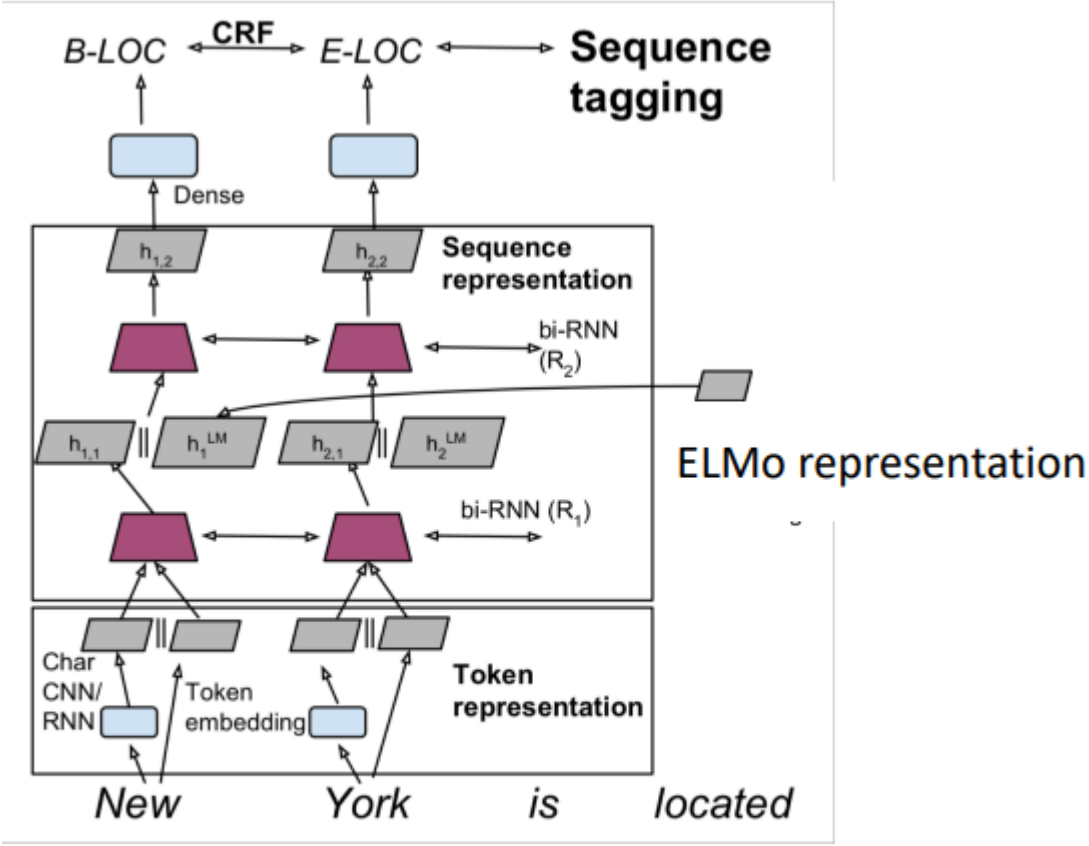
Unrolled Forward Language Model used in ELMo. Mandar Deshpande.

ELMO pretraining: step 4



Unrolled Forward Language Model used in ELMo. Mandar Deshpande.

Deep contextualized word representations



$$\mathbf{h}_{k,1} = [\overrightarrow{\mathbf{h}}_{k,1}; \overleftarrow{\mathbf{h}}_{k,1}; \mathbf{h}_k^{LM}]$$

Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

Deep contextualized word representations

- BiLSTM
  - Maximize log likelihood of the combination of backward and forward LM

$$\sum_{k=1}^{N} (\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s))$$

Where

- Θx : context-independent token encoding
- Θs : softmax layer

Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

Deep contextualized word representations

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

- L: layers
- γ^{task}: scaling factor
- S^{task} : softmax-normalized mixture model weights

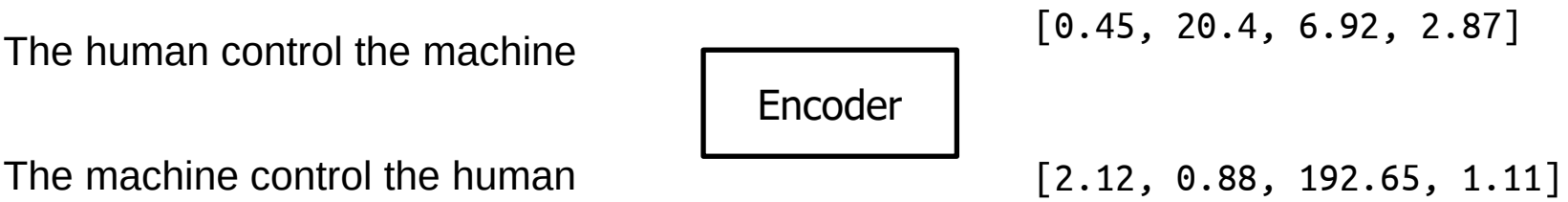
Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018

Additional reading on ELMo

- Deep contextualized word representations. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. NAACL 2018
- Download and read the paper: https://arxiv.org/pdf/1802.05365.pdf

Sentence encoding

- ELMo produces word-level text representations
- How can we extend the idea to sentence representations?
  - Sentence encoders
- Sentence encoders are in charge of directly mapping an arbitrary sequence of textual units to a fixed-size vector
  - The output is no longer a word-level representation



Sentence encoders

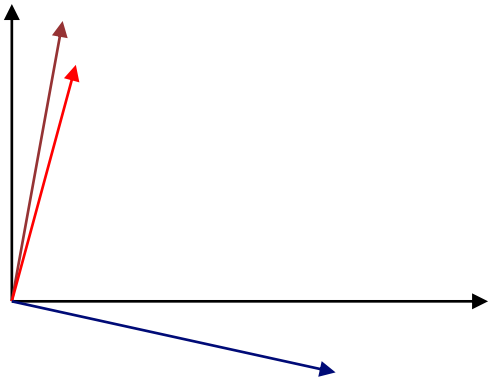
- Sentence encoders are in charge of mapping an arbitrary sequence of textual units to a fixed-size vector
- They provide us with semantically sensitive representations

The pen is on the table

The pencil is on the chair

Biden won the USA election

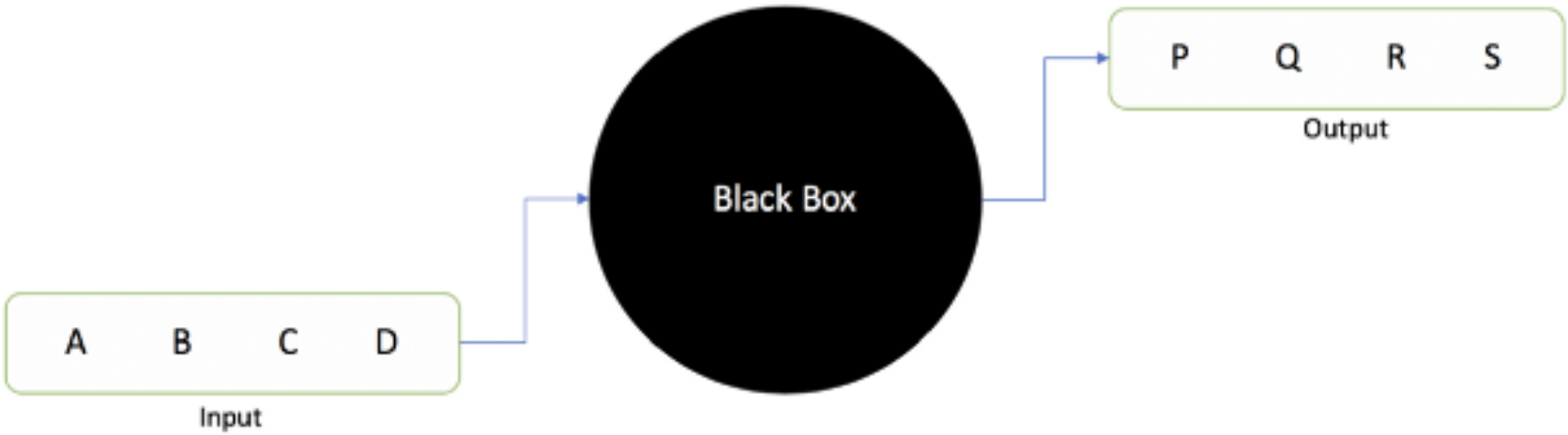
Encoder



Sequence-to-Sequence models (Seq2Seq)

- Seq2Seq learning aims at training models to convert sequences from one domain to sequences in another domain
- Common application domains
  - machine translation
    - translate a sentences from English to Italian
  - Question answering
    - generate a natural language answer given a natural language question
  - Summarization
    - generate a summary of a long piece of text
  - Text paraphrasing
    - Reformulate a piece of text (e.g., changing the style from informal to formal, removing offensive expressions, inverting the polarity from negative to positive)

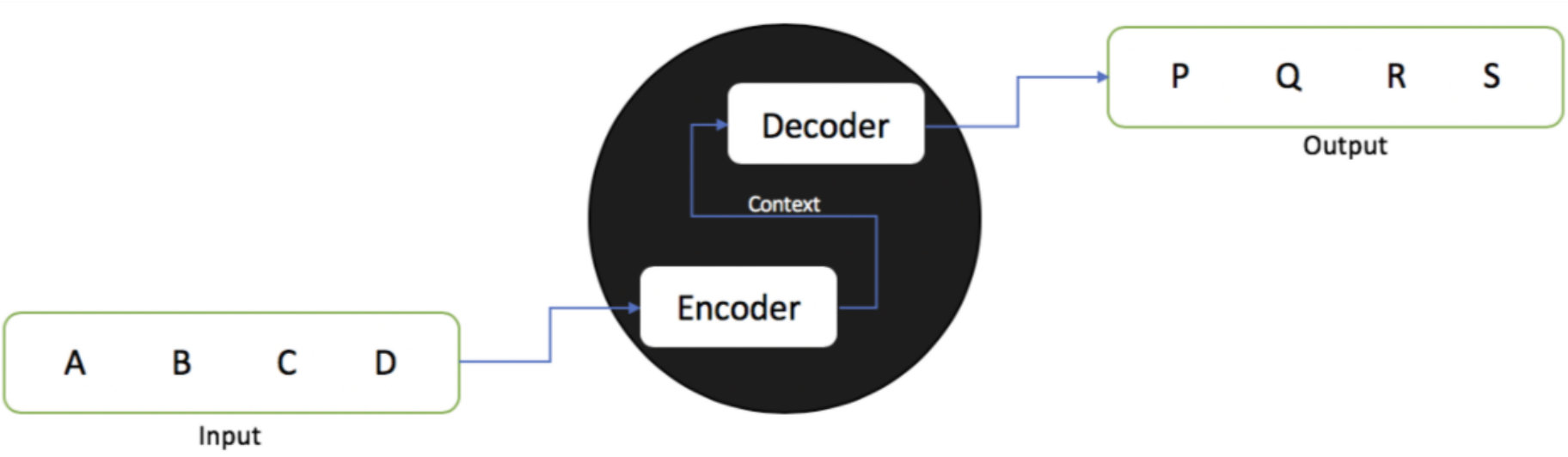
Seq2Seq: open the black box!



https://towardsdatascience.com/ (latest acces: June 2021)

Seq2Seq: open the black box!

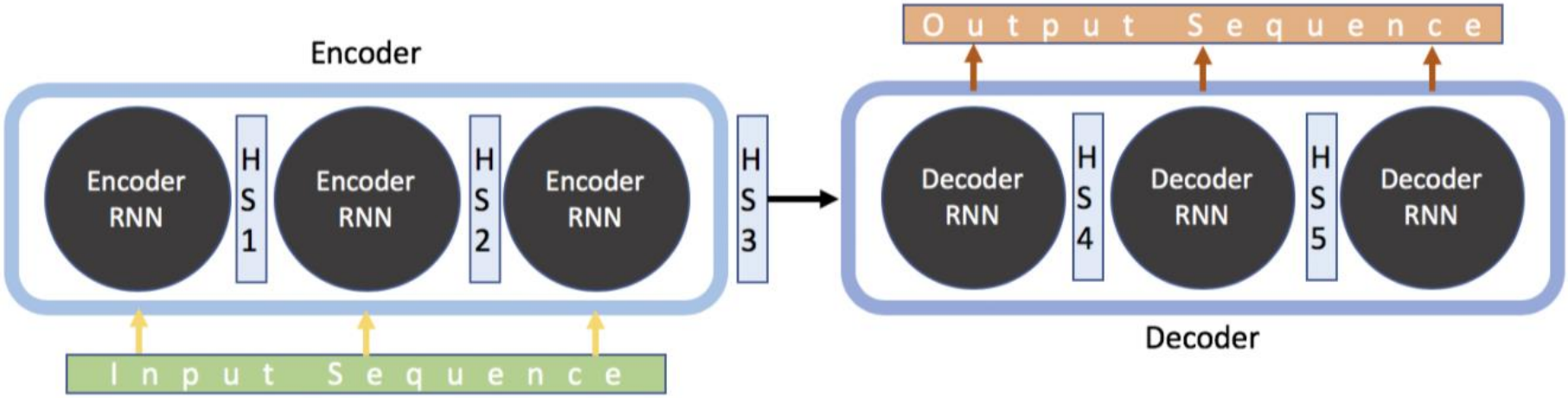
- Encoder
  - captures the context of the input sequence in the form of a hidden state vector and sends it to the decoder
- Decoder
  - produces the output sequence
- Both encoders and decoders consist of either RNNs, LSTMs, or GRUs



https://towardsdatascience.com

Seq2Seq: open the black box!

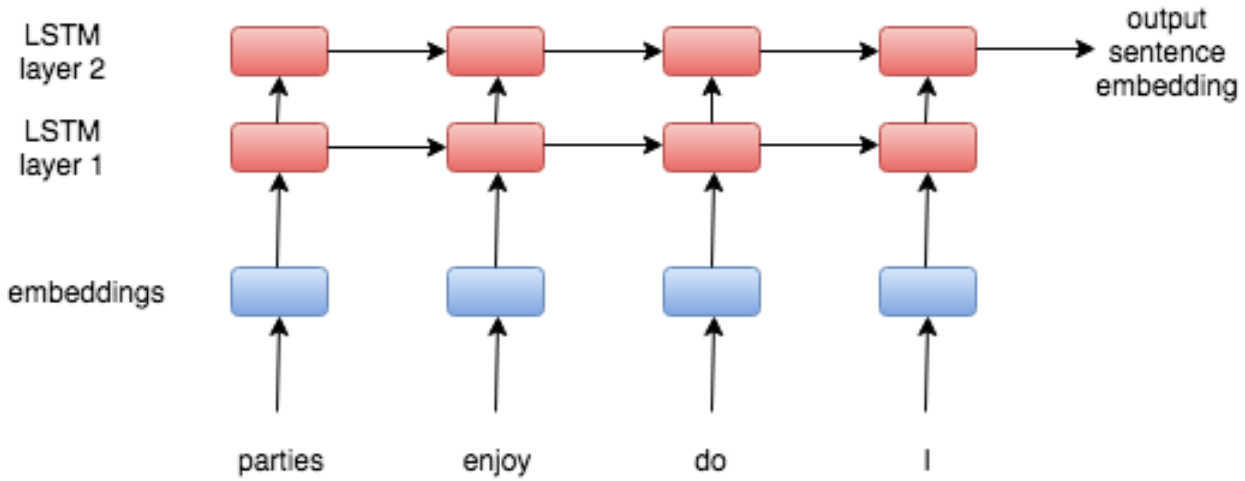
- Each RNN takes as input the current sample and a representation of the previous inputs
- The sequential information is stored into a hidden state and used at the next instance



https://towardsdatascience.com/

Directional embeddings

- LSTM networks take into account variable sequence lengths
  - Similar to a human readers they process the input from left to right (or vice versa, e.g., for arabic languages)
- They provide contextualized word-level representation that could be easily combined to create sentence embeddings



Directional embeddings

- The preceding context is taken into account to create sequence embedding
- Intermediate, word-level representation are also contextualized

A **dividend** is a distribution of cash or stock to a class of shareholders in a company.

In division, the amount or number to be divided is called the **dividend**.

dividend != **dividend**

Directional embeddings: drawbacks

- Sentences must be processed word by word
  - This hinders parallel training
- In directional sequence-to-sequence models each state (word) is assumed to be dependent only on the previously seen states
  - This does not allow contextual information to be effectively retained for long sequences
- Words are processed in a single direction

What's beyond directional embeddings? Leverage the attention mechanism!

- Goal
  - Mimic the retrieval of a value v_i for a query q based on a key k_i in a database

$$\text{Attention}(q, k_i, v_i) = \sum_i \text{similarity}(q, k_i) \times v_i$$

The attention mechanism: the key steps

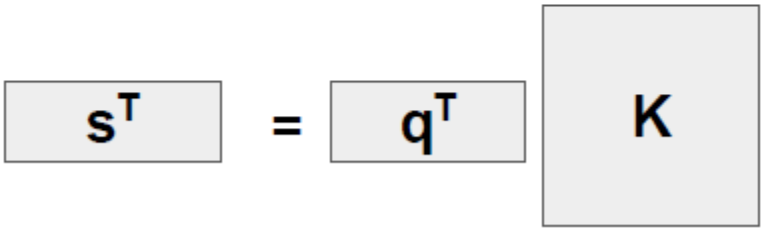
1. Given a query vector **q**, a key vector $\mathbf{k_i}$ representing value $\mathbf{v_i}$
   - S_i : similarity score between **q** and $\mathbf{k_i}$
2. Normalize the similarity scores to sum to 1
   - P_i = softmax(S_i)
3. Compute z as the weighted sum of the value vectors $\mathbf{v_i}$ weighted by their scores p_i

$$z = \sum_{i=1}^{L} p_i v_i$$

Attention types

- Given a query vector **q**, a key vector $\mathbf{k_i}$ representing value $\mathbf{v_i}$
  - S_i : similarity score between **q** and $\mathbf{k_i}$
- Additive attention
  - $Si = w_3 \tanh(\mathbf{w_2}^T\mathbf{q} + \mathbf{w_1}^T\mathbf{k_i})$
- Dot product attention
  - $S_i = \mathbf{q}^T\mathbf{k_i}$
- Scaled product attention
  - $S_i = \mathbf{q}^T\mathbf{k_i} / \mathbf{d_k}^{1/2}$

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

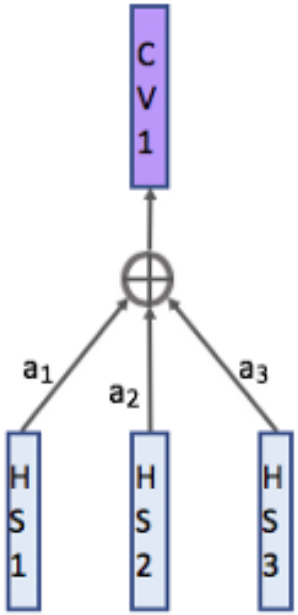


The attention mechanism: goals

- Performance improvement
  - No recurrence -> facilitate parallelization (per layer)
  - Focus on specific text portions
  - Few training steps
- Overcome the vanishing gradient and explosion
  - Shortcut to faraway states
  - Facilitate long range dependencies
- Model explainability
  - Attention distribution indicates on which sentence portion the decoder is focusing on

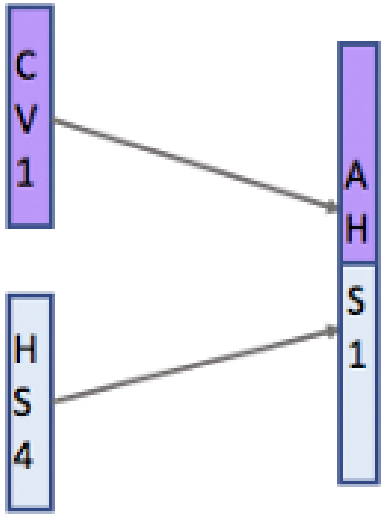
Attention mechanism: the context vector

- It is computed as the weighted sum of the input hidden state vectors



https://towardsdatascience.com/ (latest acces: June 2021)

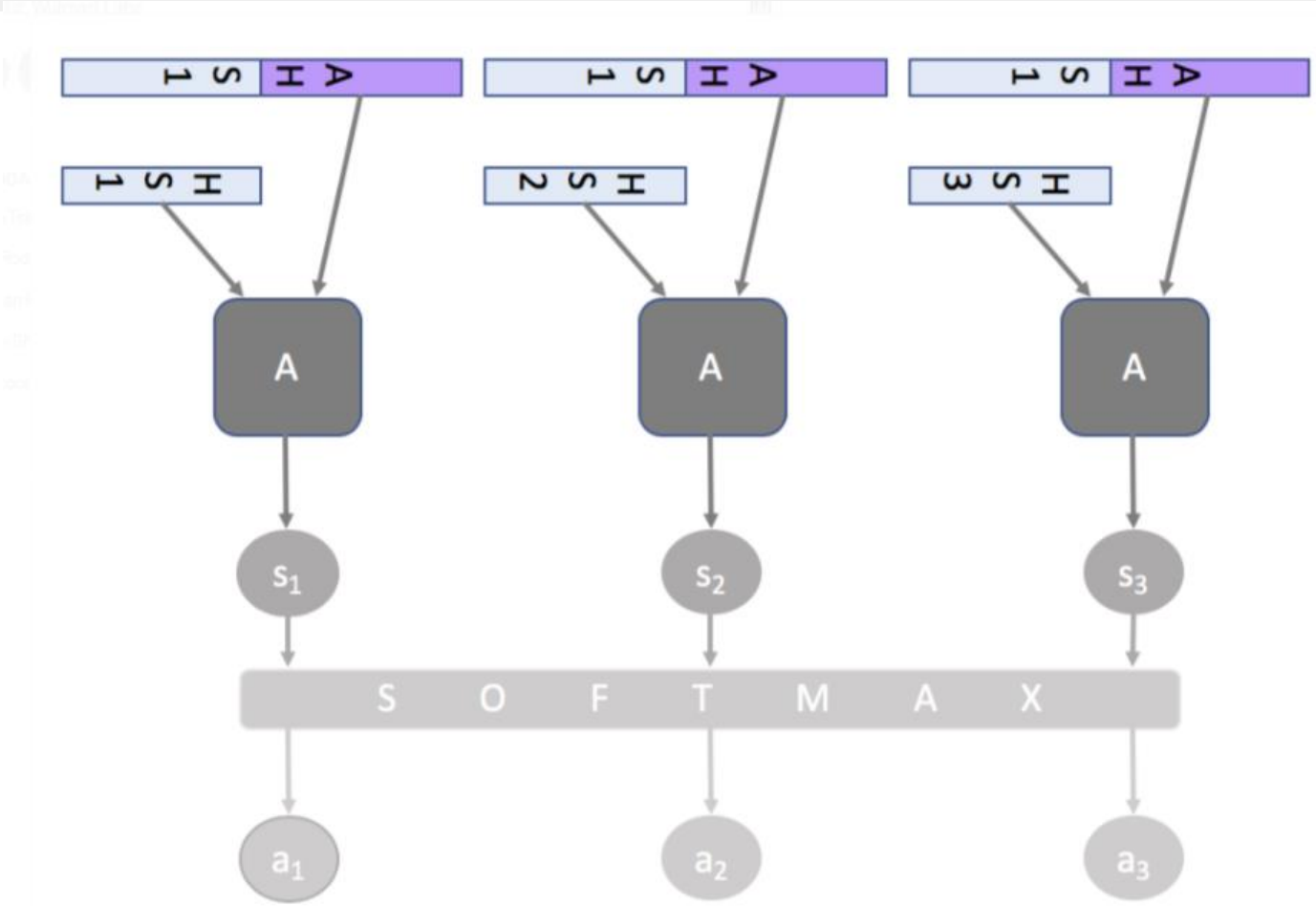
Attention mechanism: the attention hidden vector



- The generated context vector is combined with the hidden state vector by concatenation to form a *attention hidden vector*
- The attention hidden vector is used for predicting the output at that time instance

https://towardsdatascience.com/ (latest acces: June 2021)

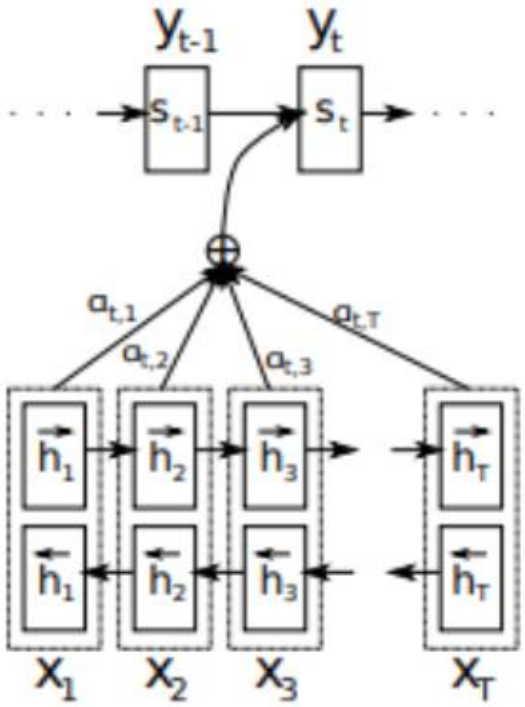
Attention mechanism: the alignment model



https://towardsdatascience.com/ (latest acces: June 2021)

The alignment model

- Align source and target sequences
  - Instead of compressing the key information about the source sentence into a fixed-size vector
- Attend to a specific part of the source sentence
- Minimize information loss from long sentences
  - Avoid missing long-term dependencies

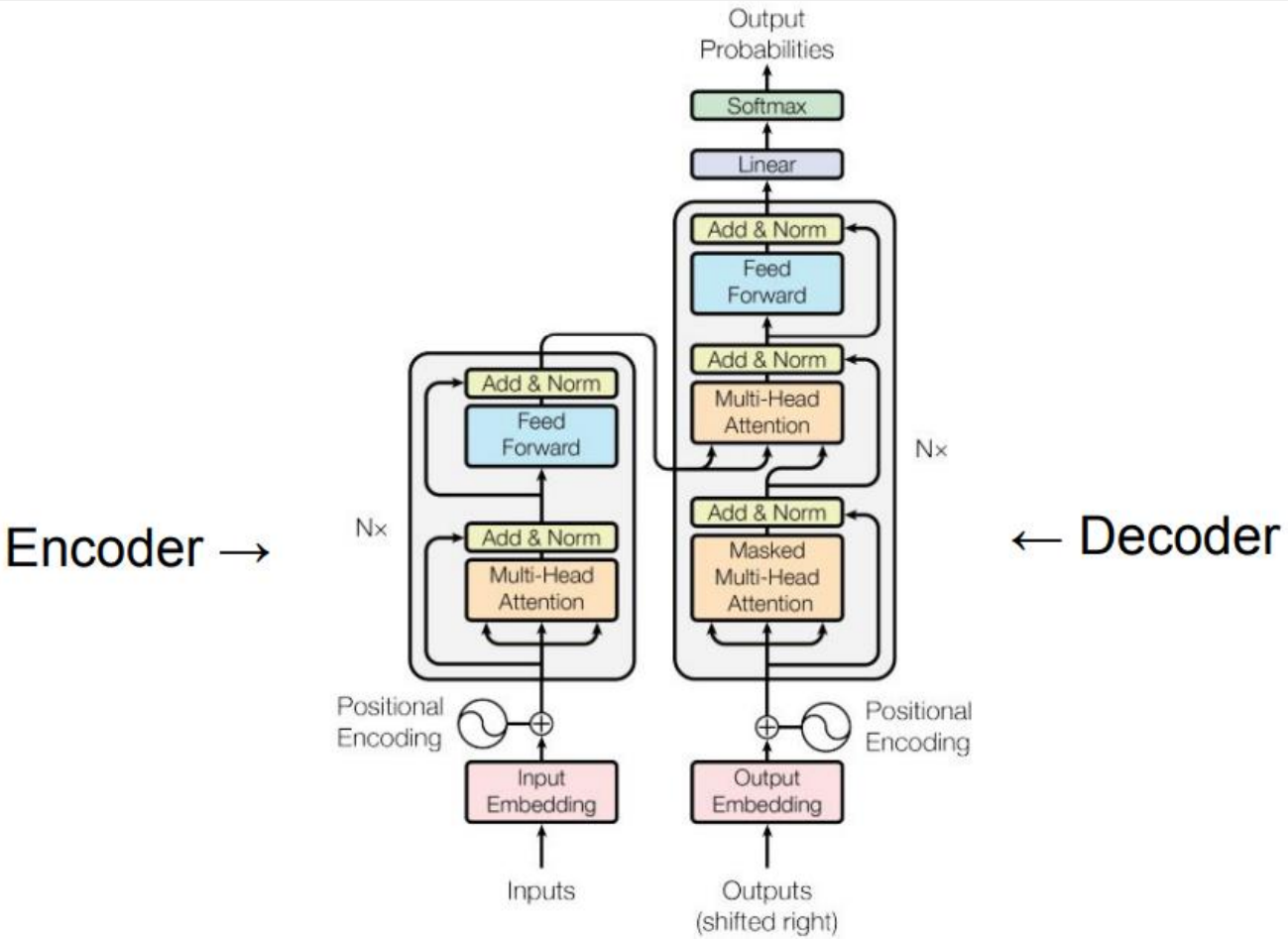


The alignment model

- The alignment model is trained jointly with the seq2seq model initially
- It scores how well an input (represented by its hidden state) matches with the previous output (represented by attention hidden state) and does this matching for every input with the previous output
- Then a softmax is taken over all these scores and the resulting number is the attention score for each input

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

The Transformer architecture



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer

- Goal: minimize recurrence and maximize parallelization
  - Optimization is too long
  - Too many sequential operations
- All you need is attention!
  - According to the pioneering work by Vaswani et al. (2017)
- Major shift in the thinking of sequential data

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture

- Encoder-decoder based on attention
  - No recurrence
- Key properties
  - Fed the entire sequence of words in input to the encoder
  - Compute the corresponding embeddings
  - Process the whole input text in parallel
    - No sequential processing
  - With positional encoding it considers the relative word positioning
    - Otherwise, you would get a BOW-like representation (as in occurrence-based representations)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture

- Every word pair is processed in parallel
  - Great for GPU-based computation
- Complexity
  - Quadratic in the input sequence length n
  - Linear with the representation dimension (fixed)
- Self-attention layers are faster than recurrent layers when the sequence length n is smaller than the representation dimensionality d
  - Common case for sentence representations in several NLP tasks
- While coping with very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

Computational complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Self-attention layer connects positions with a constant number of executed operations, whereas RNN layers require O(n) operations

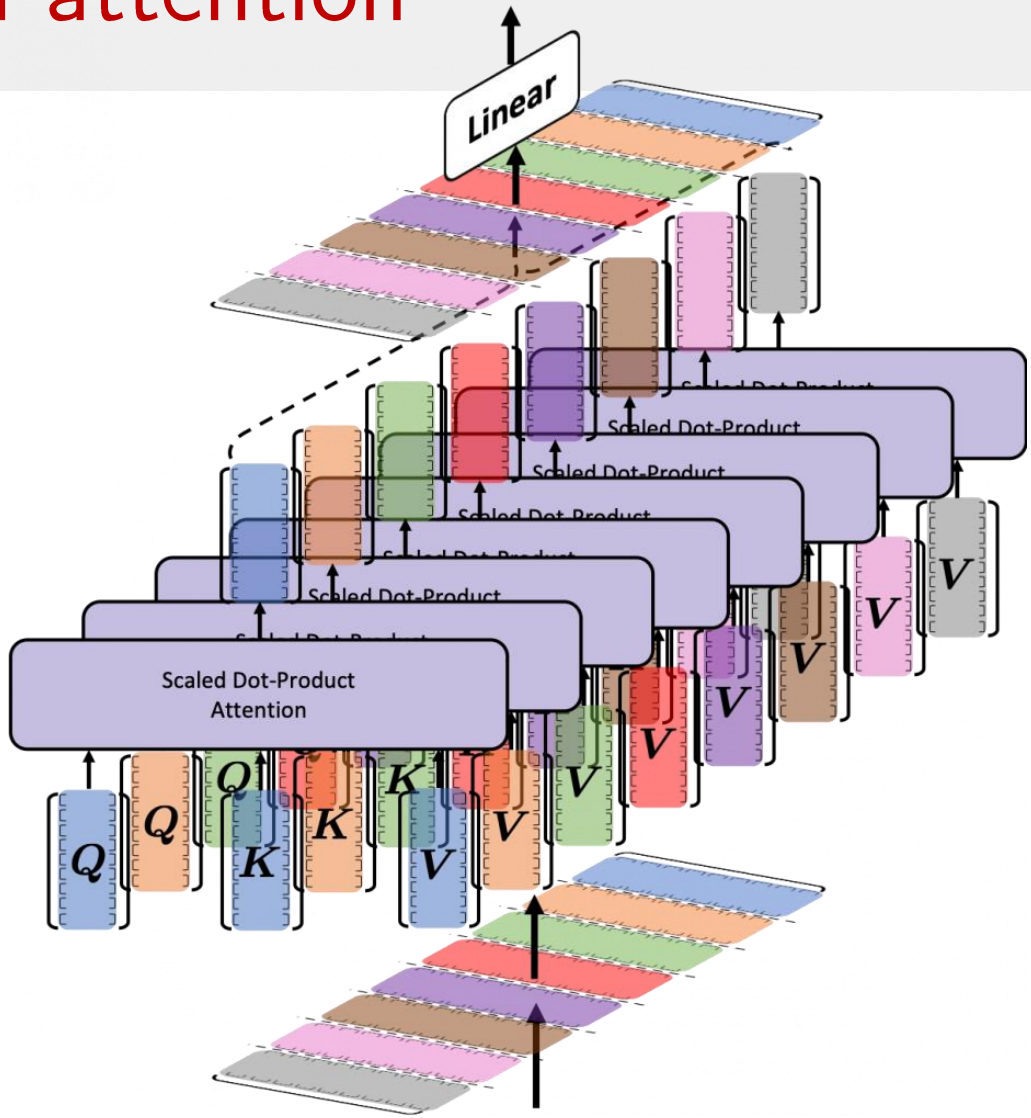
Distance in hops between any two input and output positions in the network (1 for Fully Connected Networks)

Trasnformer: the encoder

- Multi-head attention
  - Fed a vector
    - Subvectors of all words in our input sentence
  - Compute the attention between every position and every other position
    - Treat each word as a query
    - Retrieve the keys of other words in the sequence
    - Take a convex combination of the corresponding values (the same as the key)
    - Take a dot product of that
  - Merge together information from pairs of words
- We repeat the process multiple times
  - The first block merges word pairs
  - The second one merges pairs of pairs
  - ...

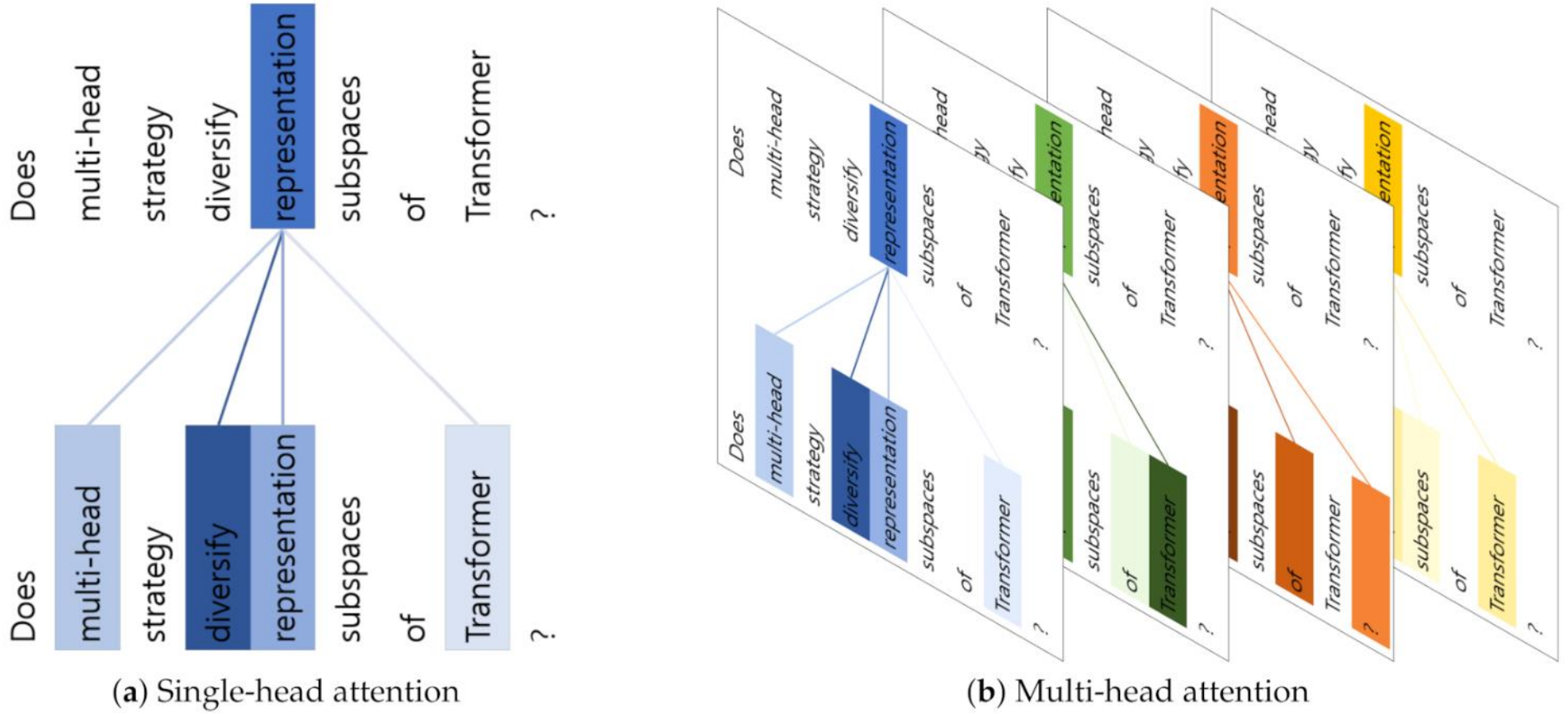
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

Multi-head attention



Multi-head attention mechanism: “queries”, “keys”, and “values,” over and over again. in Artificial Intelligence, Data Mining, Data Science, Data Science Hack, Deep Learning, Machine Learning, Main Category, Predictive Analytics, Python, Tutorial, Use Case /by Yasuto Tamura

Single vs. multi-head attention



(a) Single-head attention

(b) Multi-head attention

Yun, H.; Kang, T.; Jung, K. Analyzing and Controlling Inter-Head Diversity in Multi-Head Attention. *Appl. Sci.* **2021**, *11*, 1548. https://doi.org/10.3390/app11041548

Multi-head attention

- Multi-head attention
  - Multiple attentions produced by different projections
    - Like feature maps in CNNs
  - Linear combinations of h heads of attention
    - h times with different learned linear projections

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Masked multi-head attention

- Masked multi-head attention
  - Remove links to future words
    - Words that have never been generated yet

The transformers: the encoder

- Layer normalization
  - reduce the *covariance shift*
    - Avoid a large number of weight adjustments in gradient descent
  - Reduce the Gradient dependencies between each layer
  - Fasten the convergence of the network
  - All outputs have the same scale
    - Normalize values in each layer to have zero mean and one variance
- The output of the encoder is an encoding of the input sentence

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

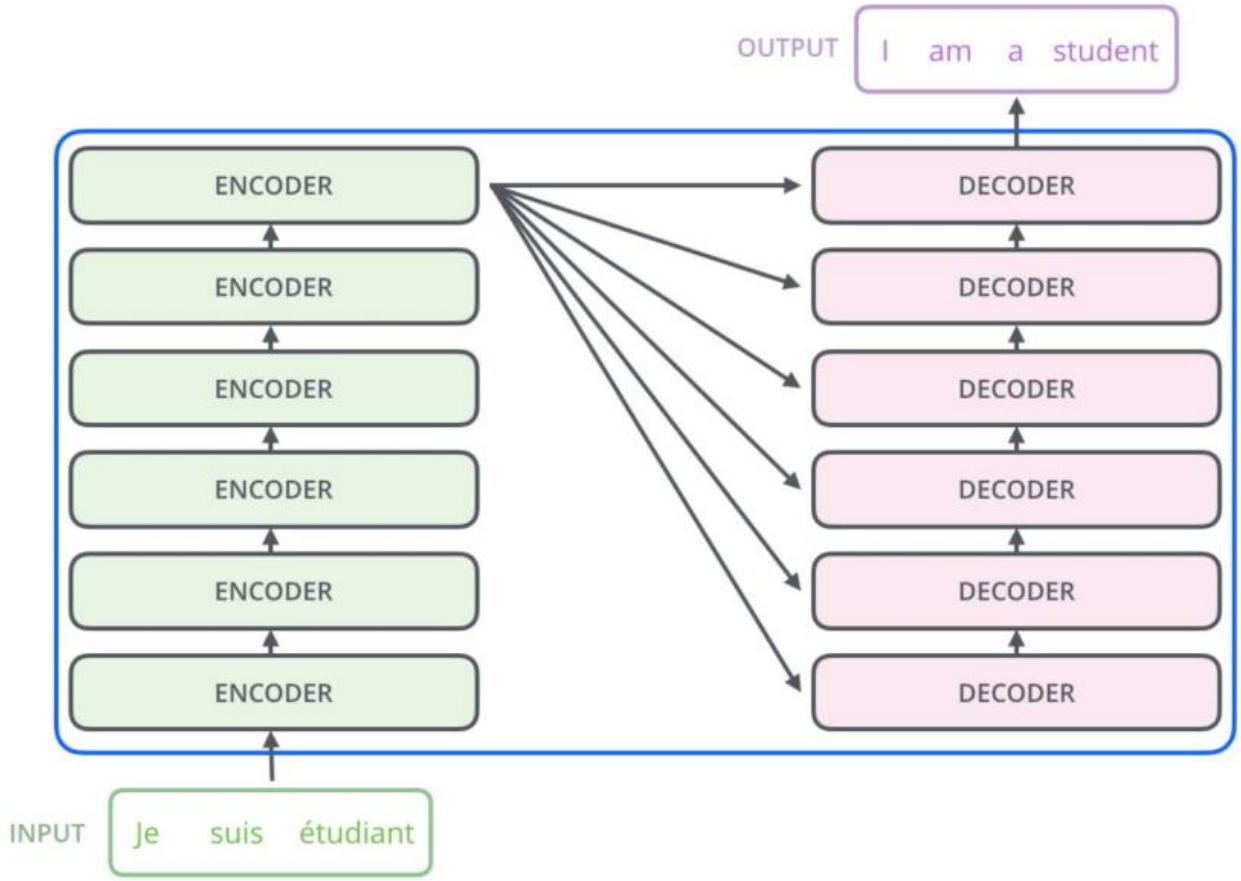
The transformers: the decoder

- Softmax produces a probability distribution as output
  - Labels associated with each sentence position
  - We get distributions over the words in the dictionary

- Two layers of attention (repeated n times again)
  - Self-attention: combine output words with previous output words
    - We can only attend previous words
      - We mask future words
  - Combine output words with input words

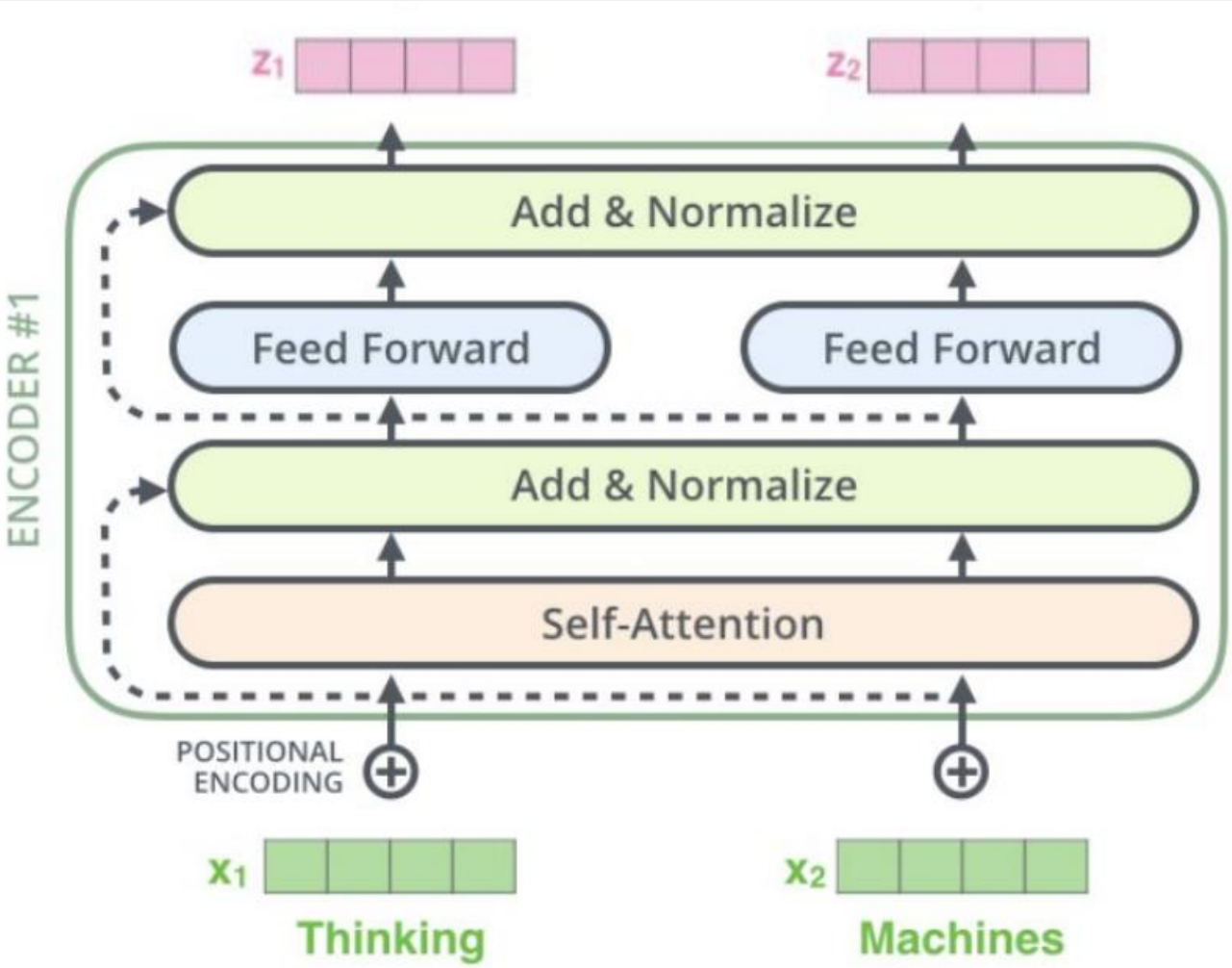
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture



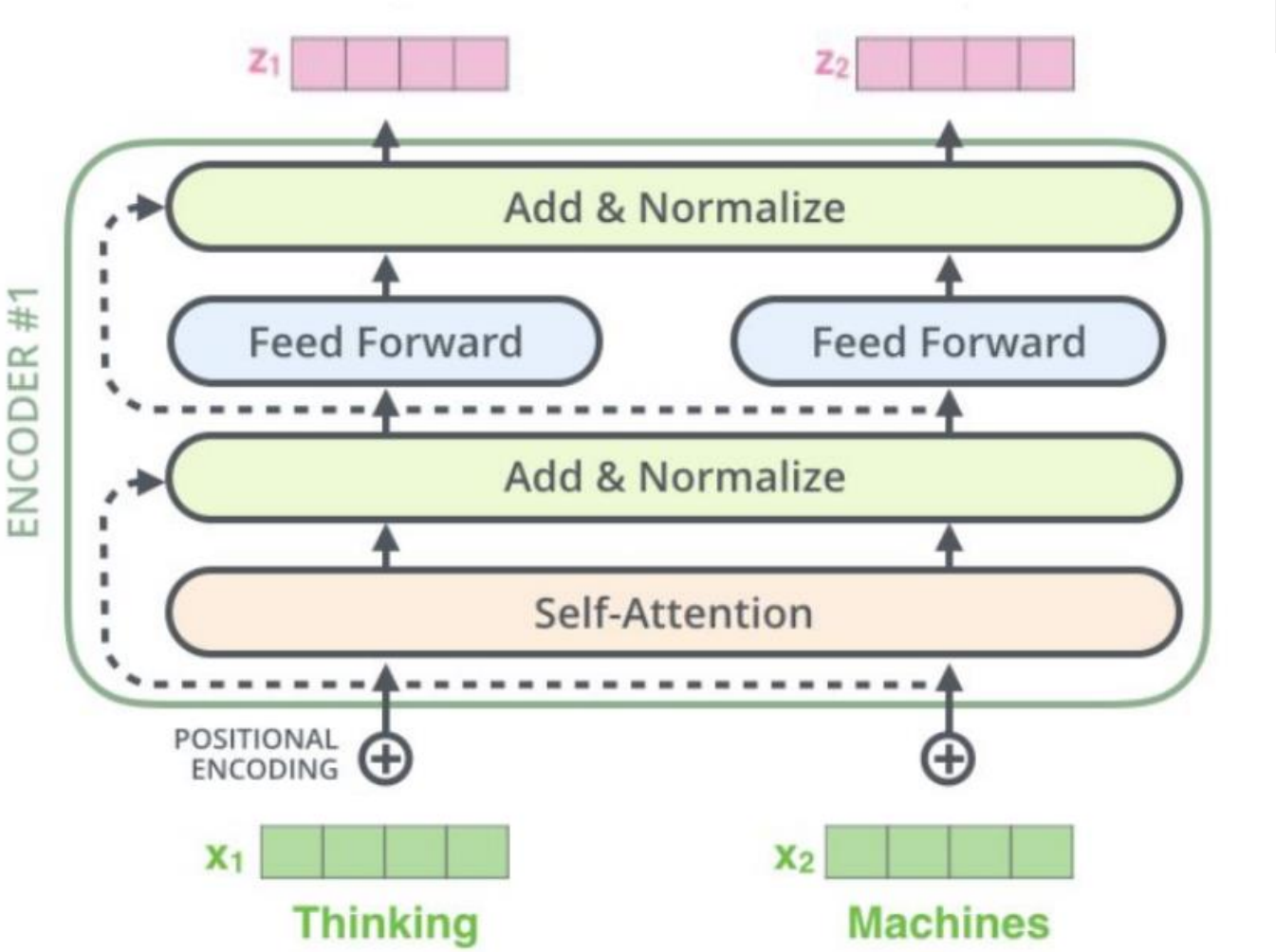
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: single block



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

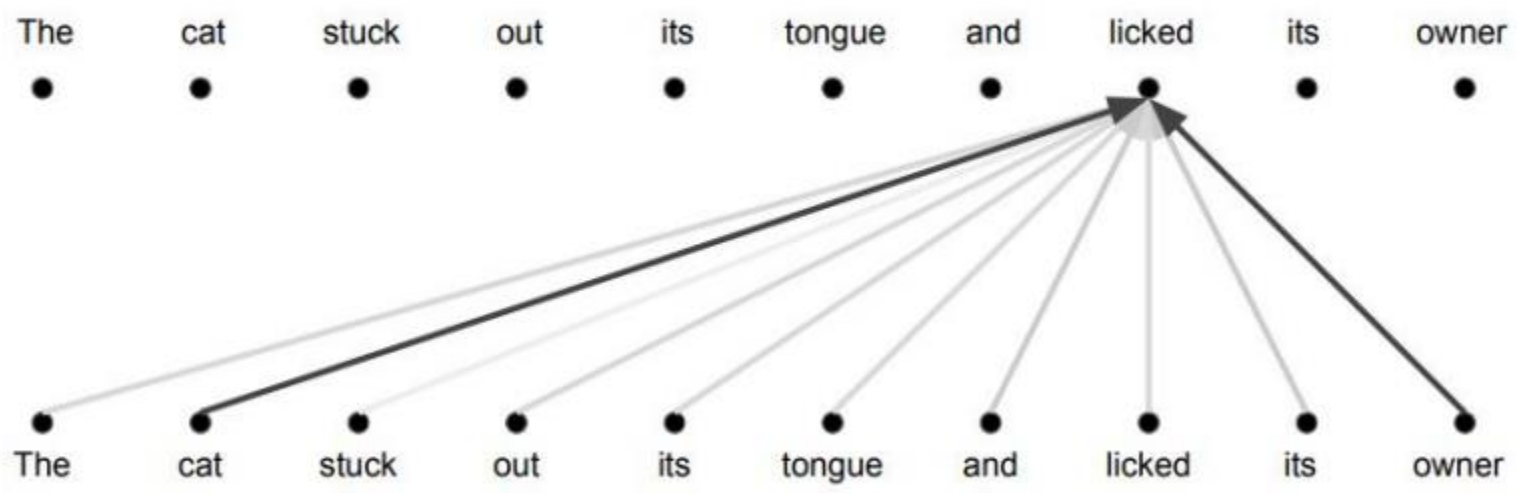
The Transformer architecture: single block



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

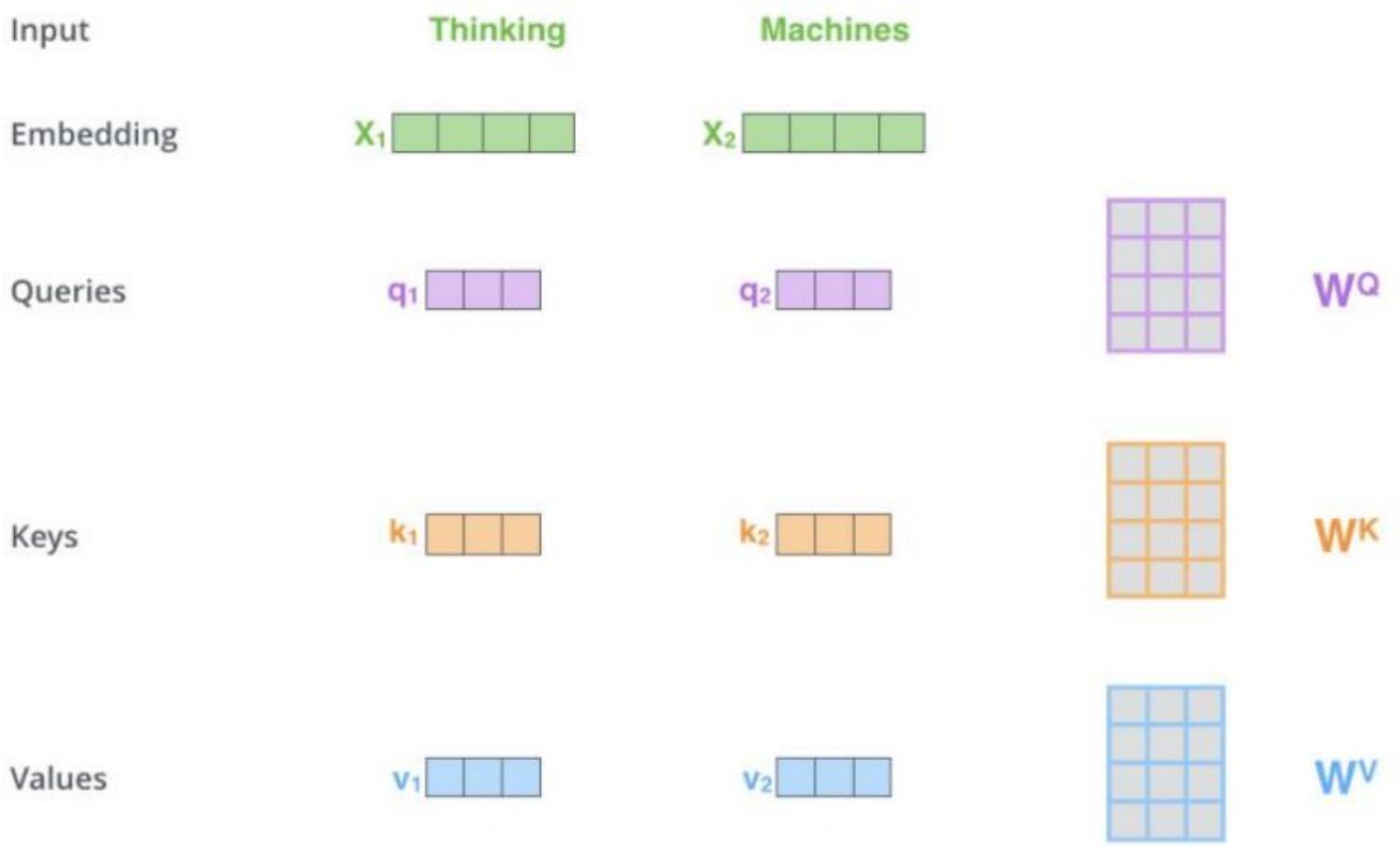
The Transformer architecture: self-attention

- Trivial to parallelize
  - Per layer



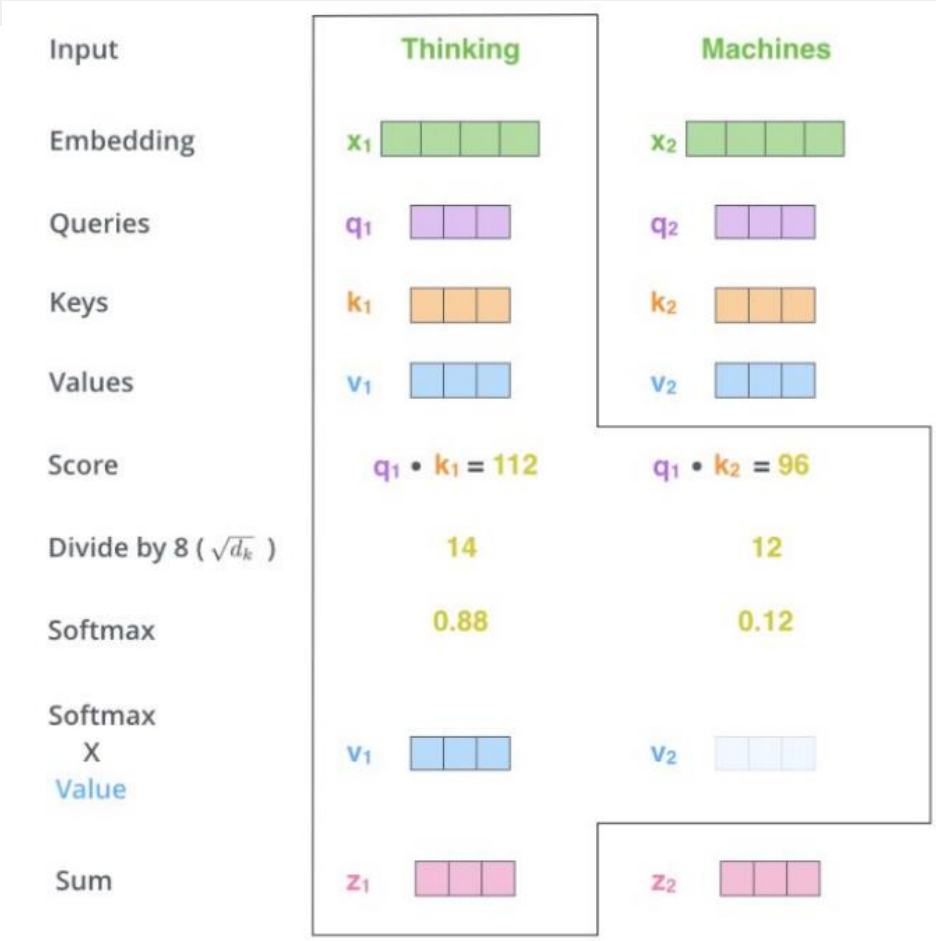
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: self-attention



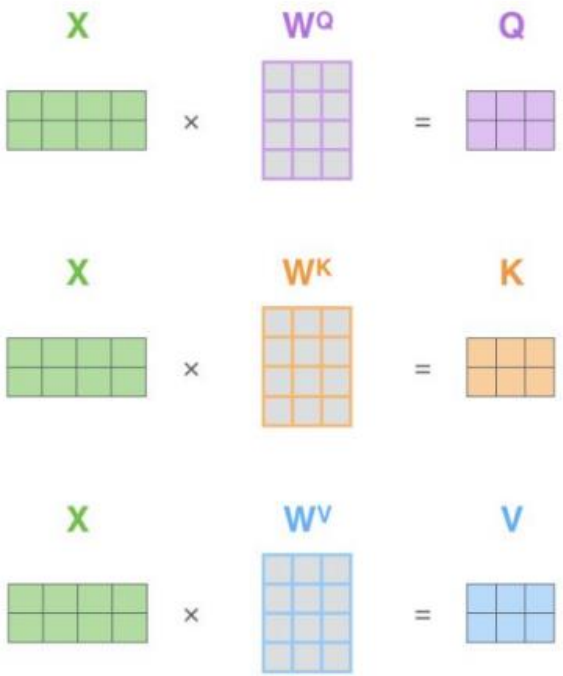
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: self-attention

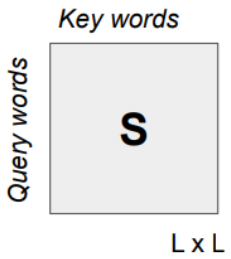


Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: self-attention

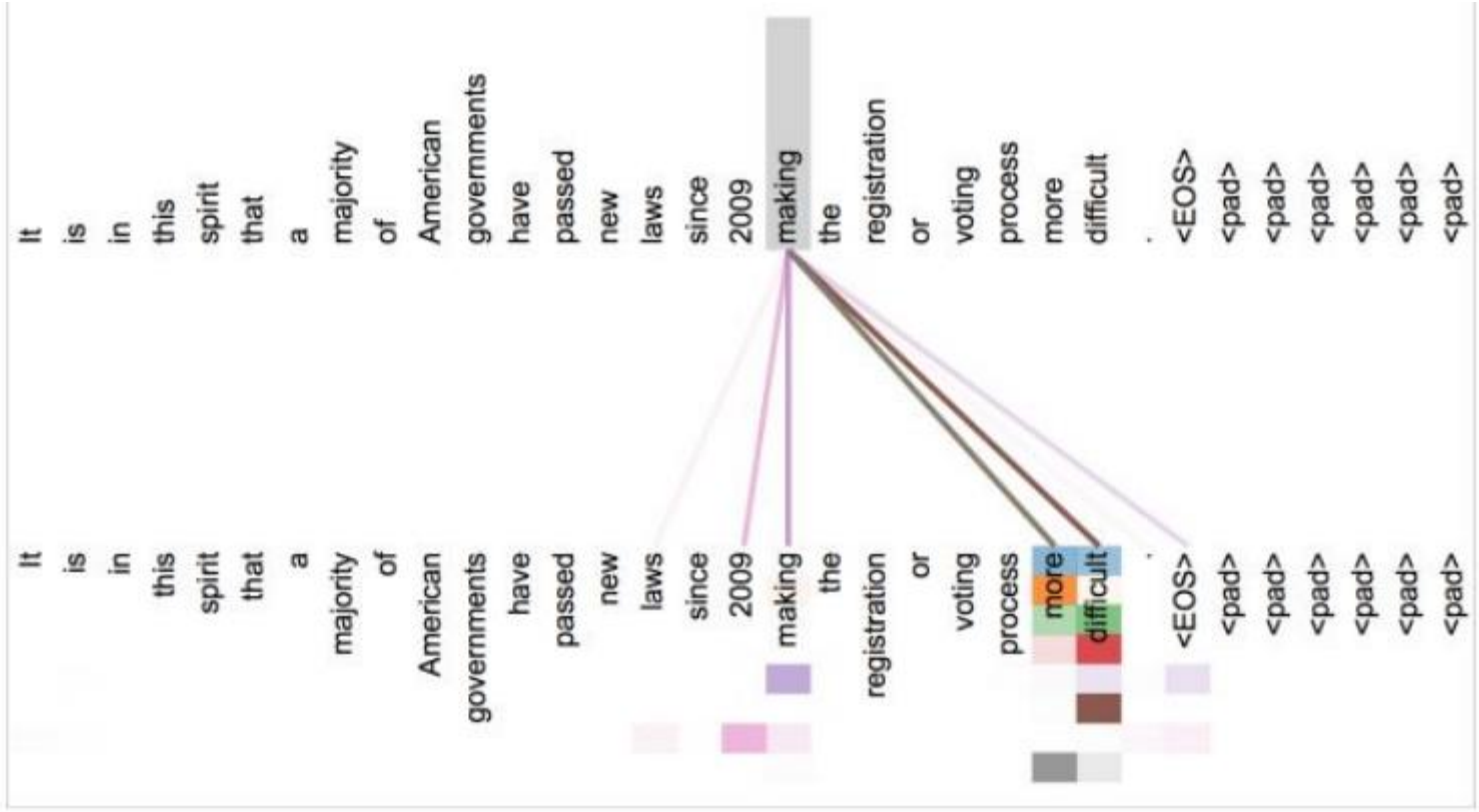


$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})$$



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

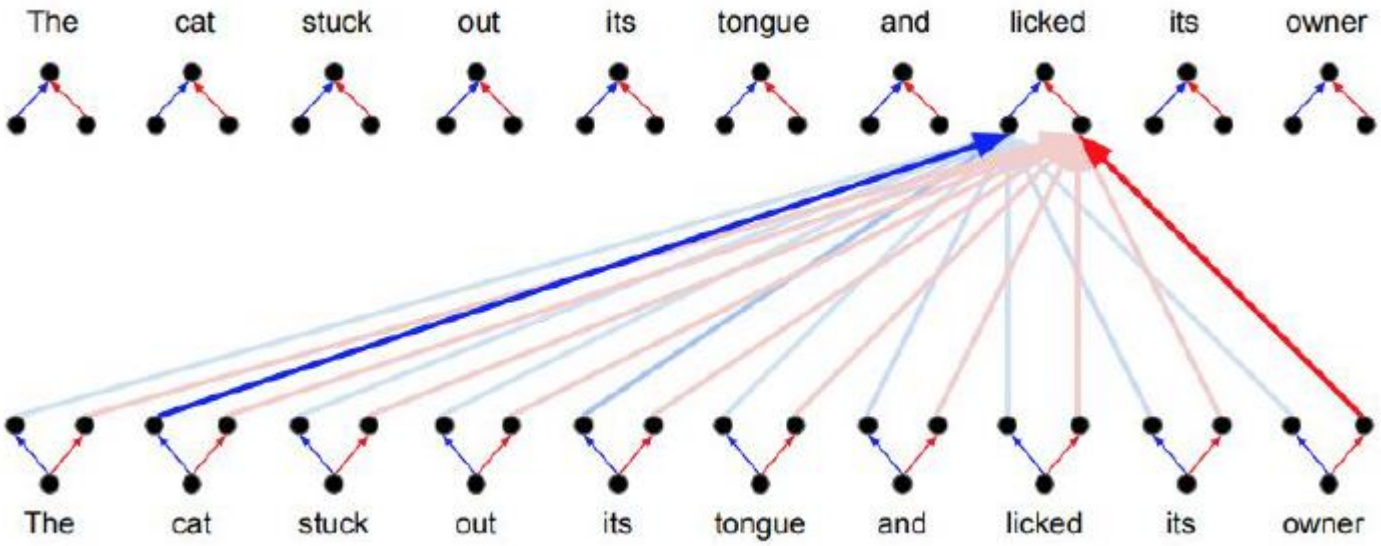
The Transformer architecture: self-attention



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

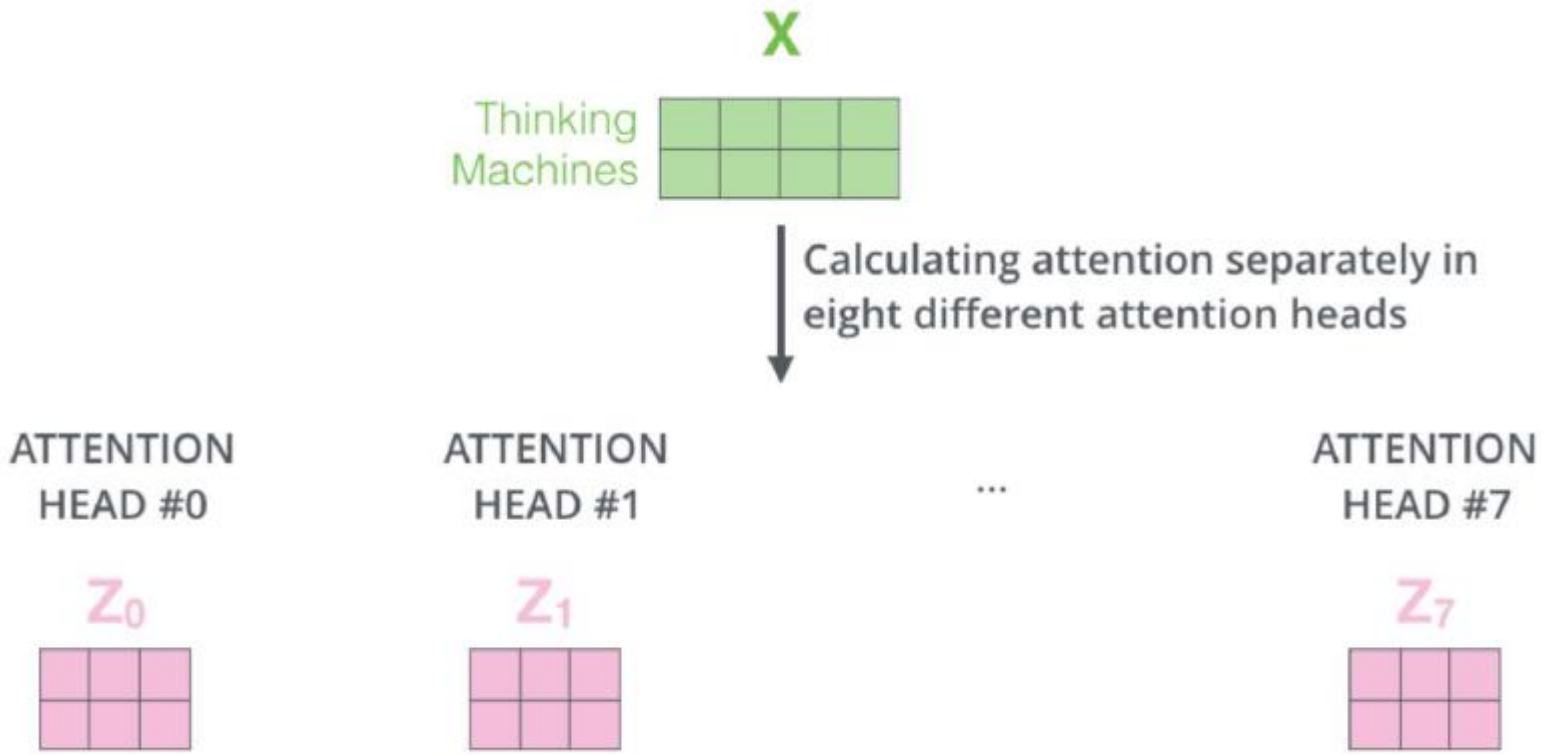
The Transformer architecture: multi-head attention

- Parallel attention layers with different linear transformations on input and output



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

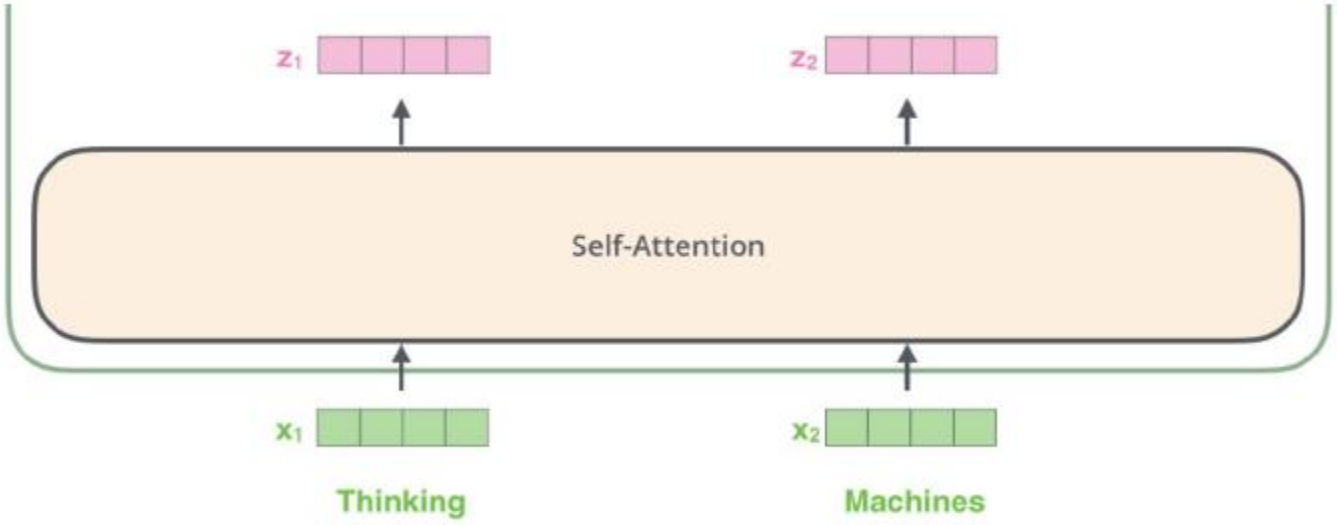
The Transformer architecture: multi-head attention



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

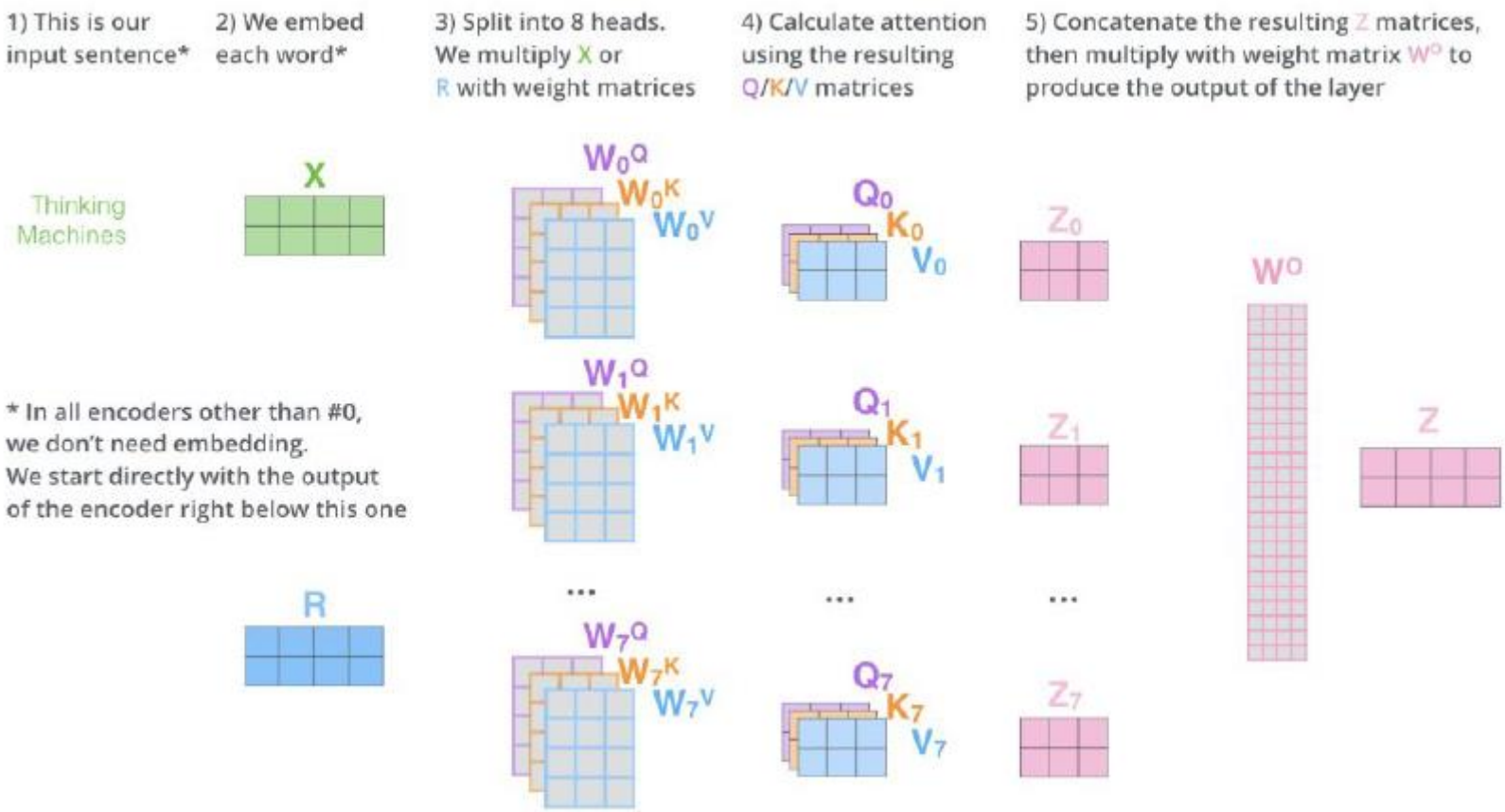
The Transformer architecture: multi-head attention

- Retain the size of the hidden state



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: Multi-head attention



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: positional encoding

- The position of a word in the sentence matters
- RNN models process text in a particular order
- The Transformer architecture does not
  - The process is fully parallelized
- The Transformer allows the injection of temporal information so that the model knows the word relationships based on their context
  - Distance and order

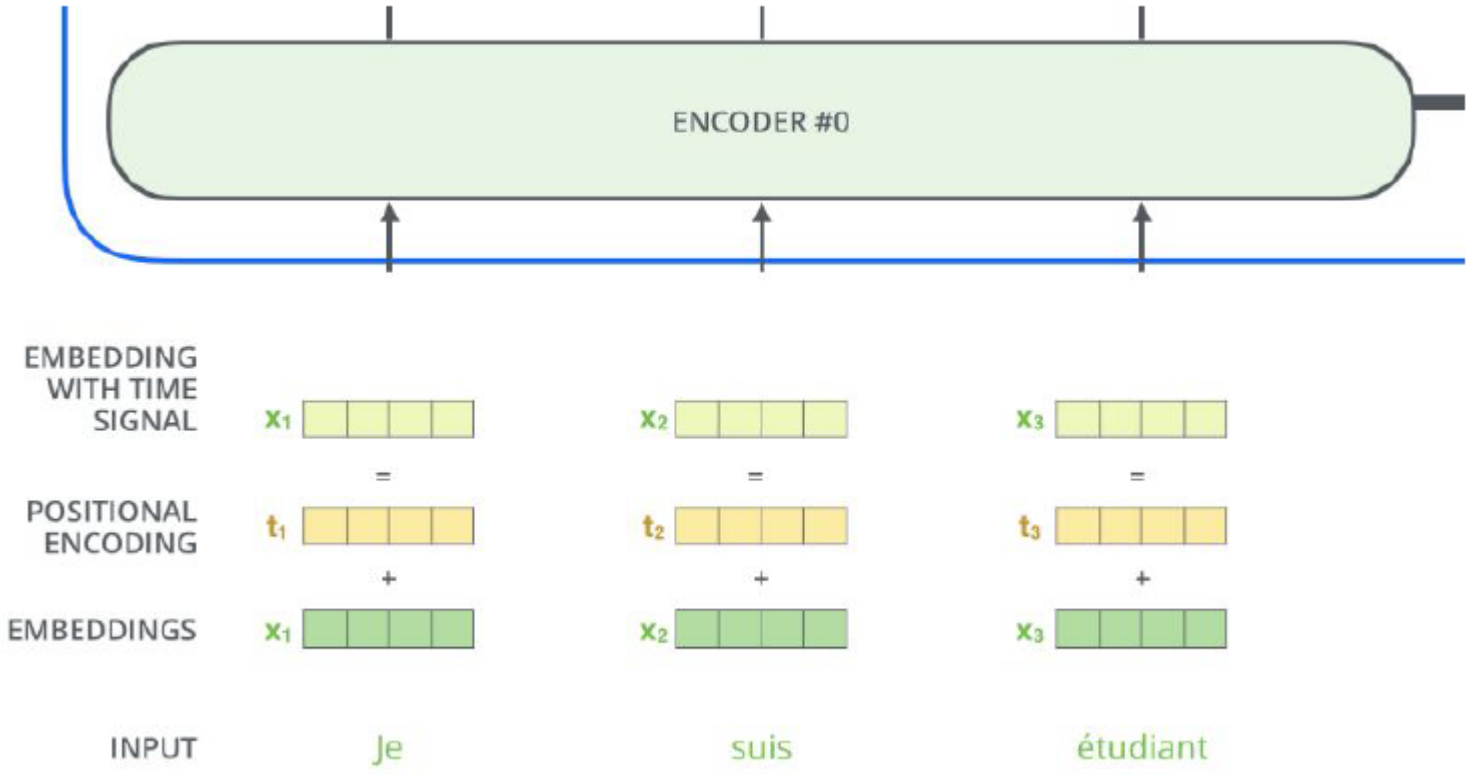
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

Positional embedding

- Embedding to distinguish each position using a sinusoid:
  - $PE_{position,2i} = \sin(position/10000^{2i/d})$
  - $PE_{position,2i+1} = \cos(position/10000^{2i/d})$
- Idea: allow the model to easily learn to attend by relative positions
  - for any fixed offset k $PE_{position+k}$ can be represented as a linear function of PE_{pos}
- Same dimension of the embeddings
  - Can be summed up

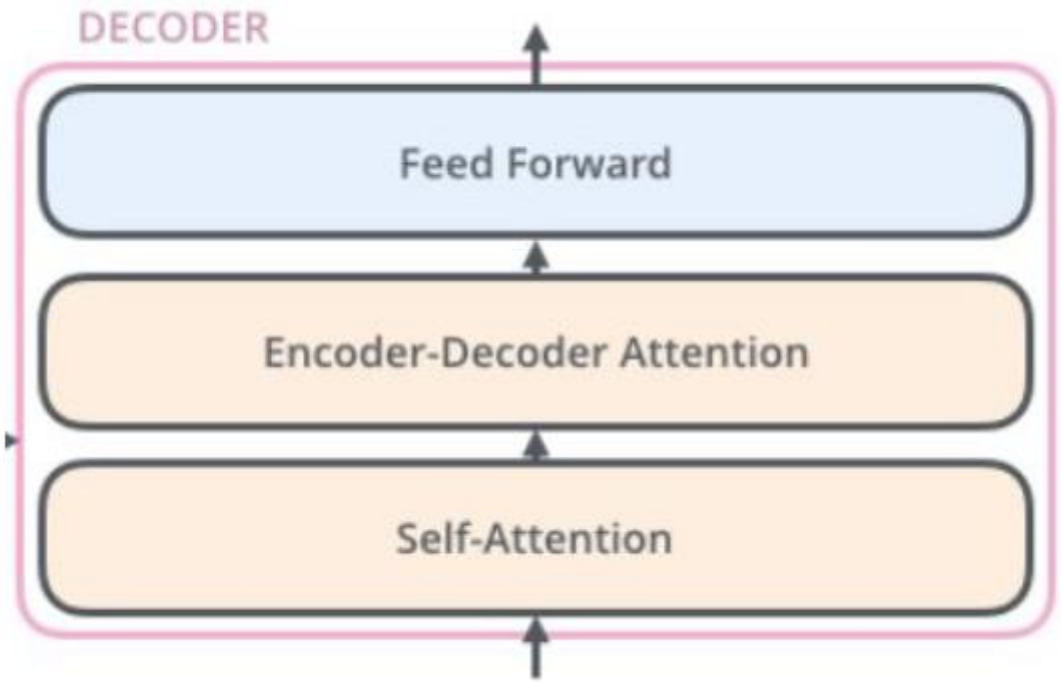
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: positional encoding



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

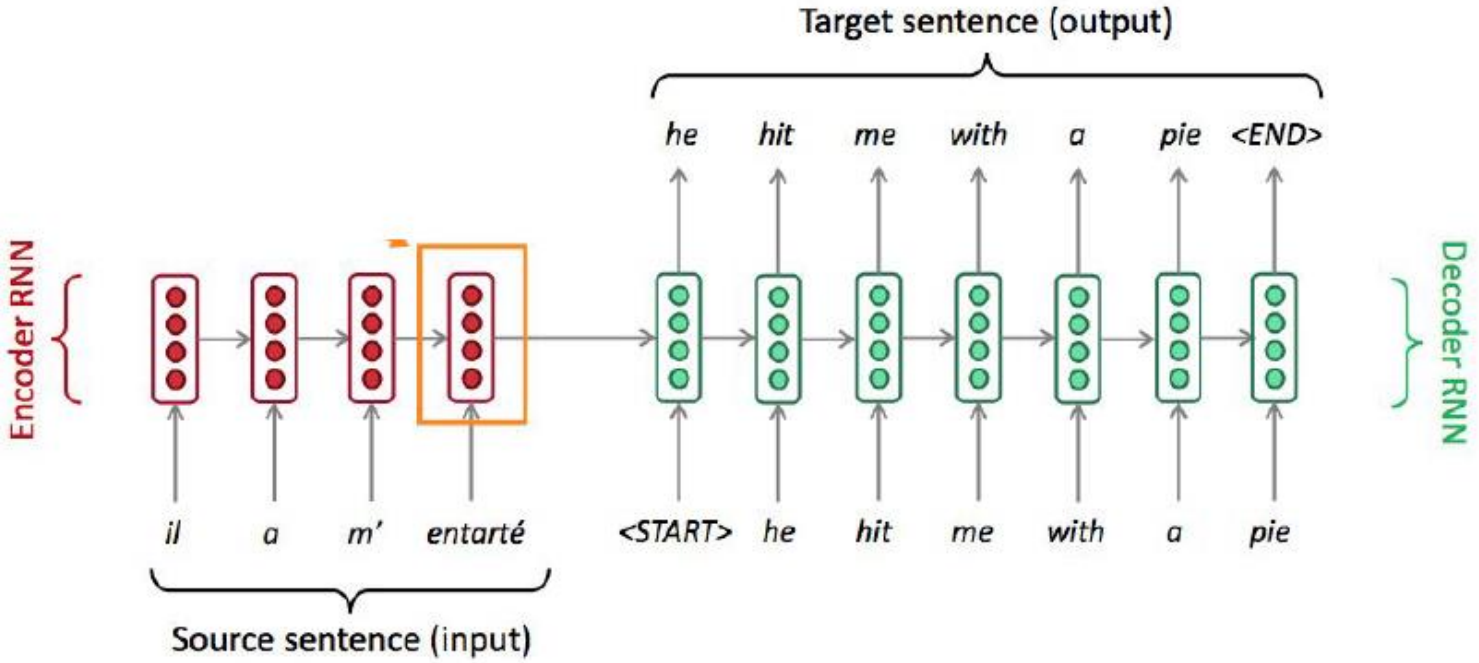
The Transformer architecture: The decoder block



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

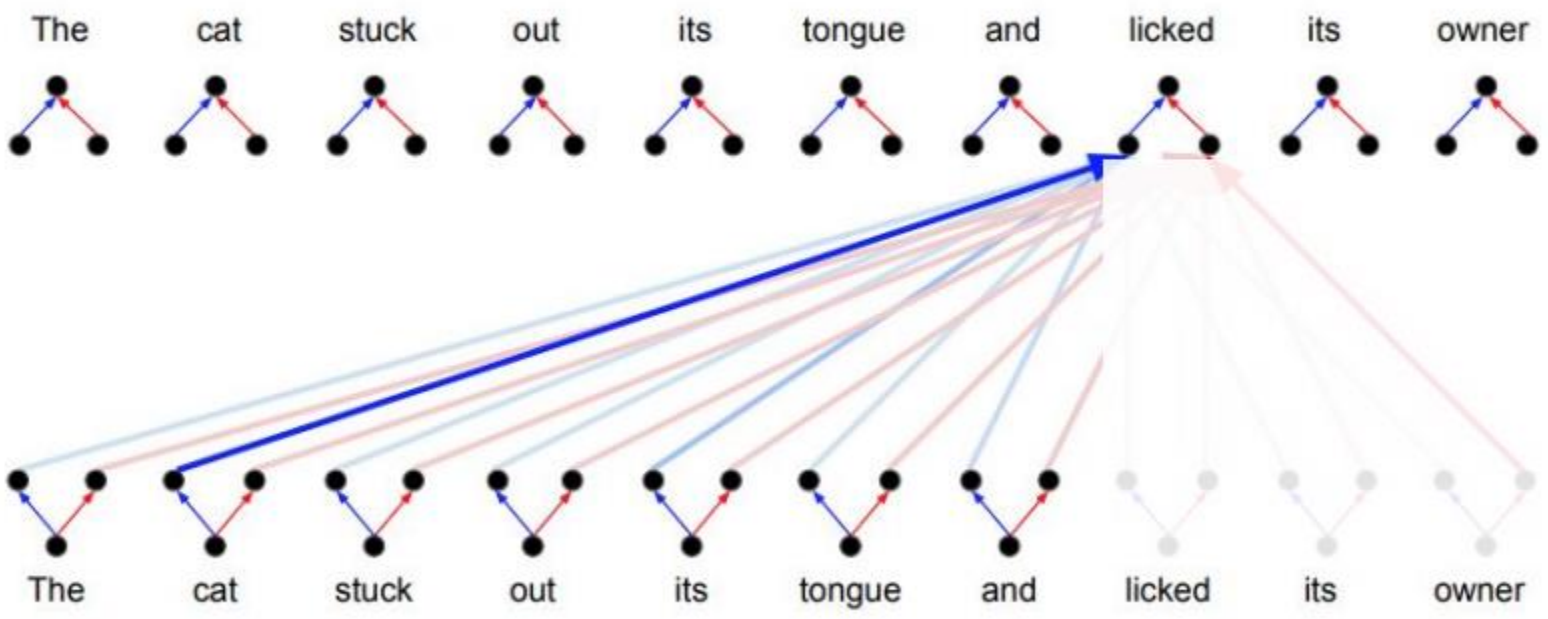
The Transformer architecture: The decoder block

P("he hit me") =
P("he")P("hit"|"he")P("me"|"he hit")



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

The Transformer architecture: Masked multi-head attention



Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

Results on Machine Translation

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Just slight accurate performance improvements... So what?

Conclusions

- Significantly more efficient still achieving state-of-the-art performance
  - 2/3 orders of magnitude better!
- Starting point of a new class of Deep NLP architectures
  - GPT-*
  - BERT, DistilBERT
  - ROBERTa
  - XLNet
  - ...

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017

Additional reading on Transformer

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. NIPS 2017
- Download and read the paper: https://arxiv.org/abs/1706.03762

Acknowlegdements and copyright license

- Copyright licence
  - Attribution + Noncommercial + NoDerivatives

- Acknowledgements
  - I would like to thank Dr. Moreno La Quatra, who collaborated to the writing and revision of the teaching content
- Affiliation
  - The author and his staff are currently members of the Database and Data Mining Group at Dipartimento di Automatica e Informatica (Politecnico di Torino) and of the SmartData interdepartmental centre
    - https://dbdmg.polito.it
    - https://smartdata.polito.it

Thank you!