AdvNLE Lecture 8

Pre-training Large Language Models

Dr Julie Weeds, Spring 2024

Previously

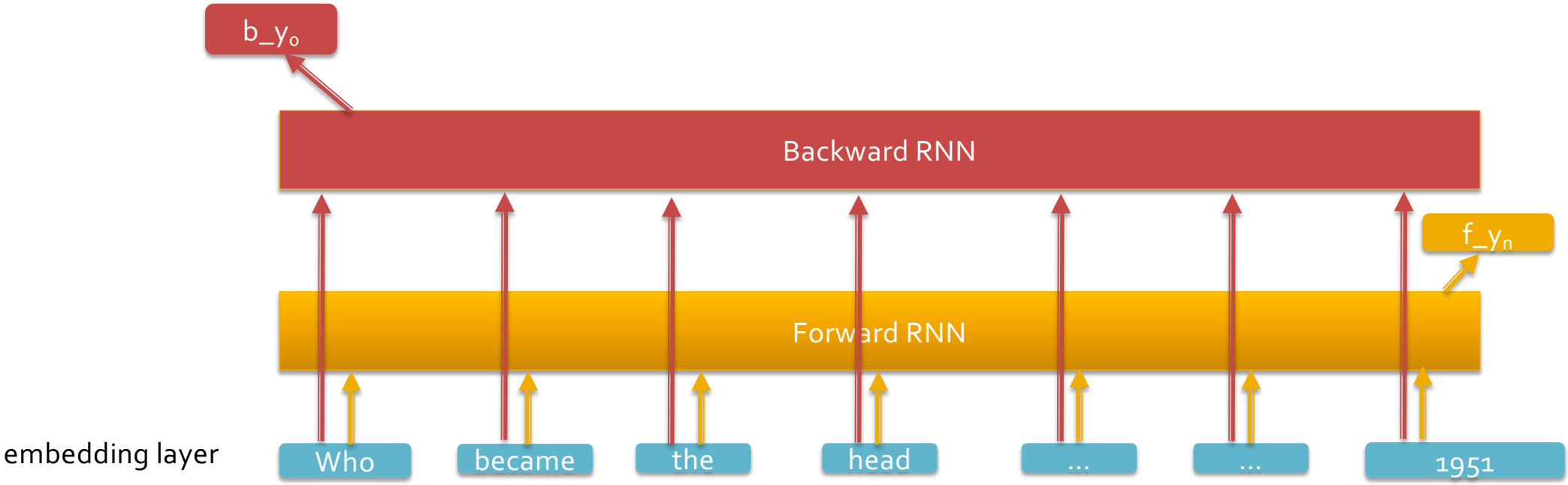
- Distributional models of word meaning
  - how similar are two words based on how they are used in text?
- Language models
  - how likely is a sequence of words in a language?
- Neural language models
- Tasks
  - Sequence labelling
  - Sequence classification
  - Sequence generation

This week

- Large language models
  - Contextualised word embeddings (ELMo)
  - Transformer architecture and attention
  - BERT (bidirectional encoder representation from transformers)
  - Pre-training:
    - Masked language modelling
    - Autoregressive language modelling
    - Next sentence prediction
  - Sentence representations / paraphrasing (Seminar)

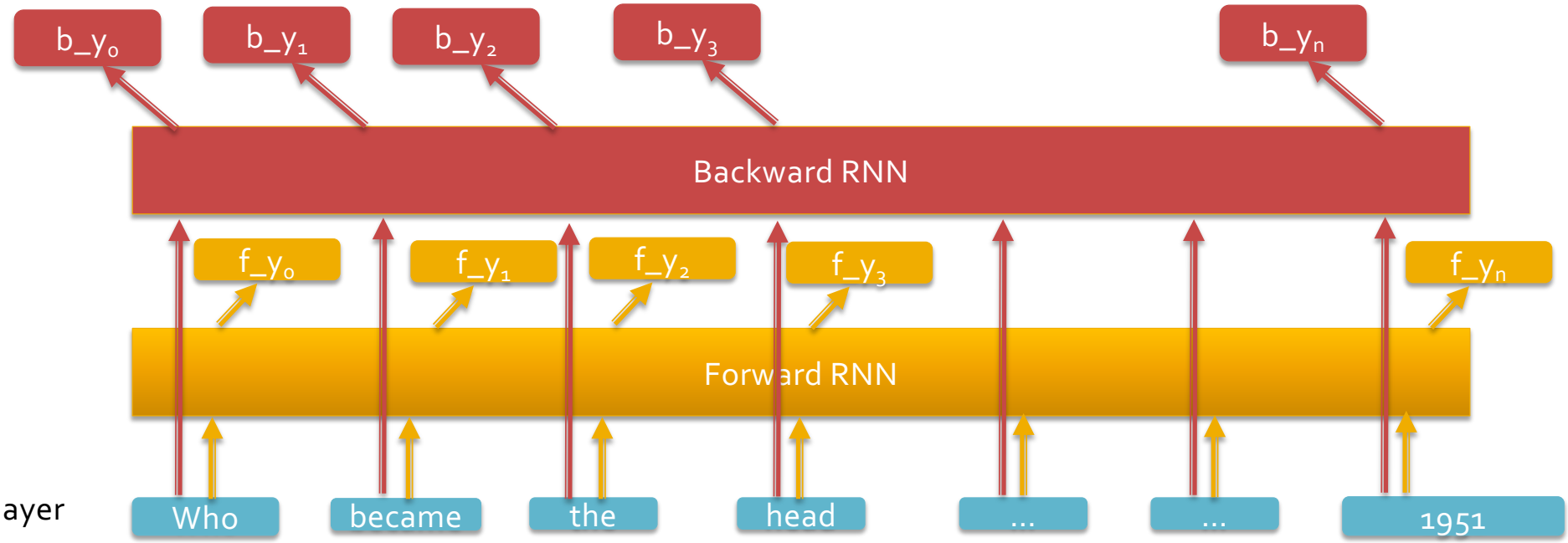
Language modelling for Sentence Representations

- Represent a sequence by what is predicted to the left and right of it
- e.g., concatenation of [b_y_o,f_y_n]



Contextualised word embeddings

- Represent each word based on its context
- e.g., concatenation of [b_y_t,f_y_t]
- compose contextualised word embeddings as before



ELMo (Peters et al. 2018)

- Embeddings from Language Models
- representations are functions of the entire word sequence
- computed
  - on top of two-layer biLMS with character convolutions
  - as a linear function of internal network states
- multiple representations of the same word:
  - higher-level layers capture context dependent aspects of word embeddings
  - lower-level layers capture aspects of syntax
- semi-supervised learning –
  - biLMs can be pre-trained at a large scale
  - final layers fine-tuned on smaller / task-specific data-set

ELMo Base Model

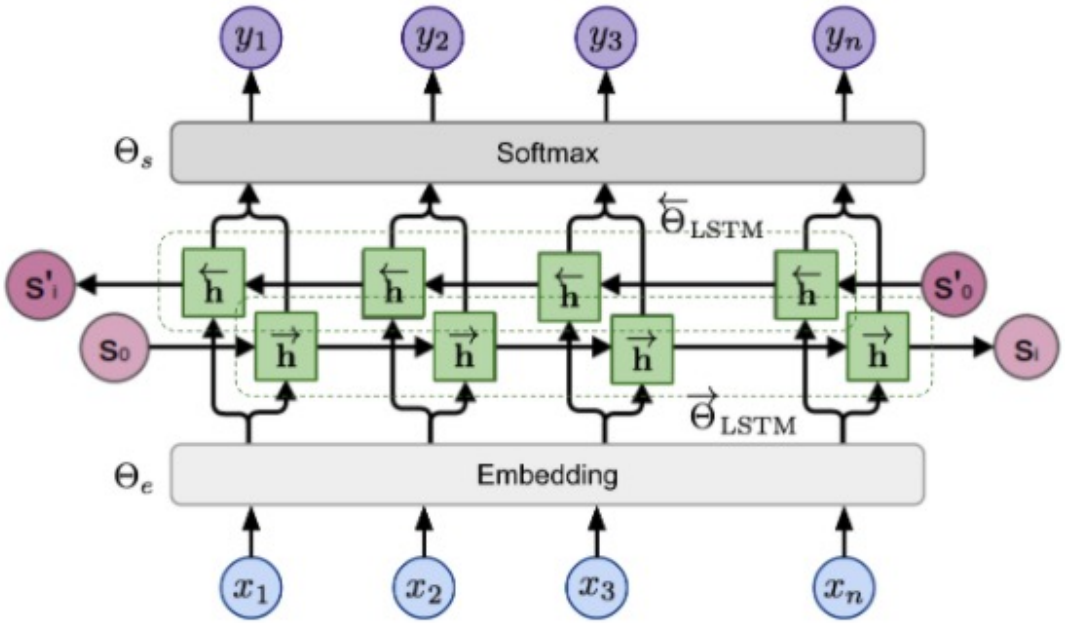
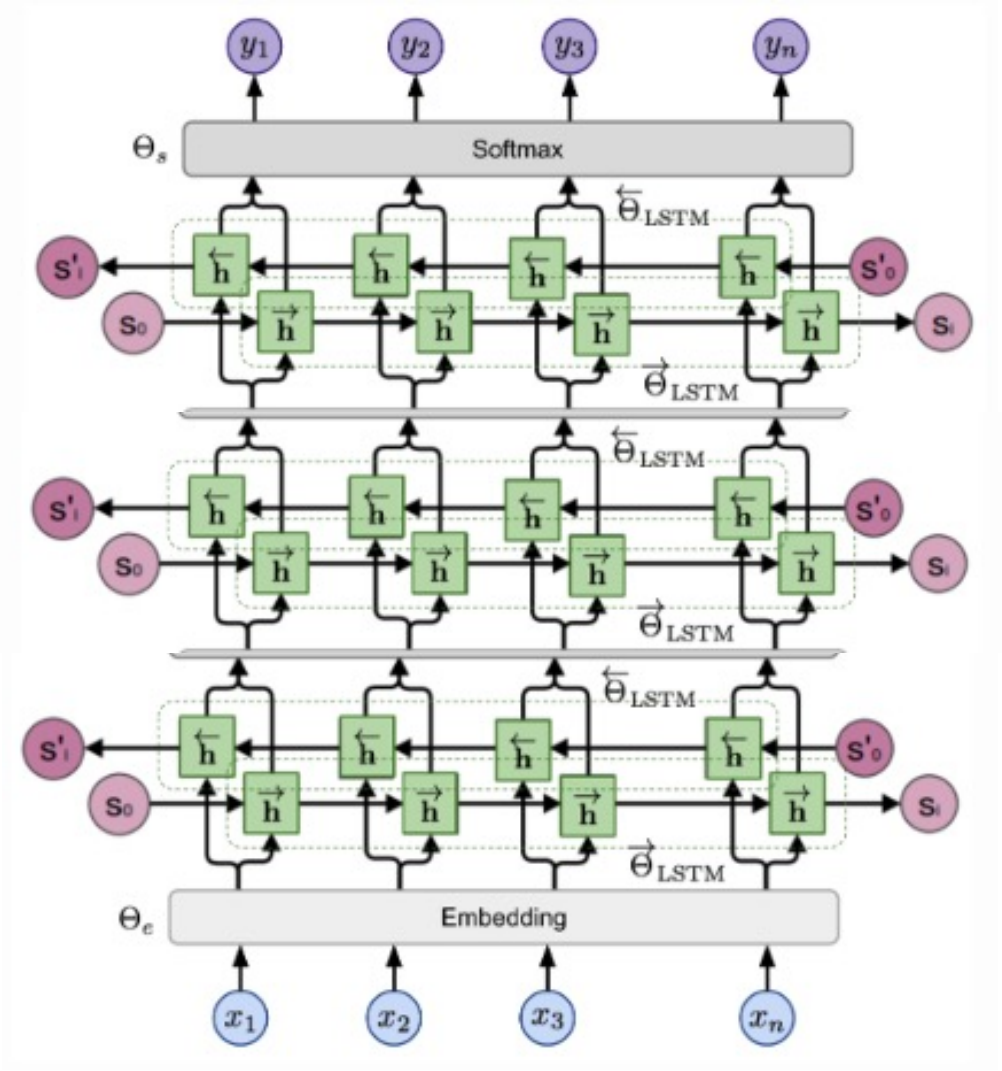


Fig. 3. The biLSTM base model of ELMo. (Image source: recreated based on the figure in "Neural Networks, Types, and Functional Programming" by Christopher Olah.)

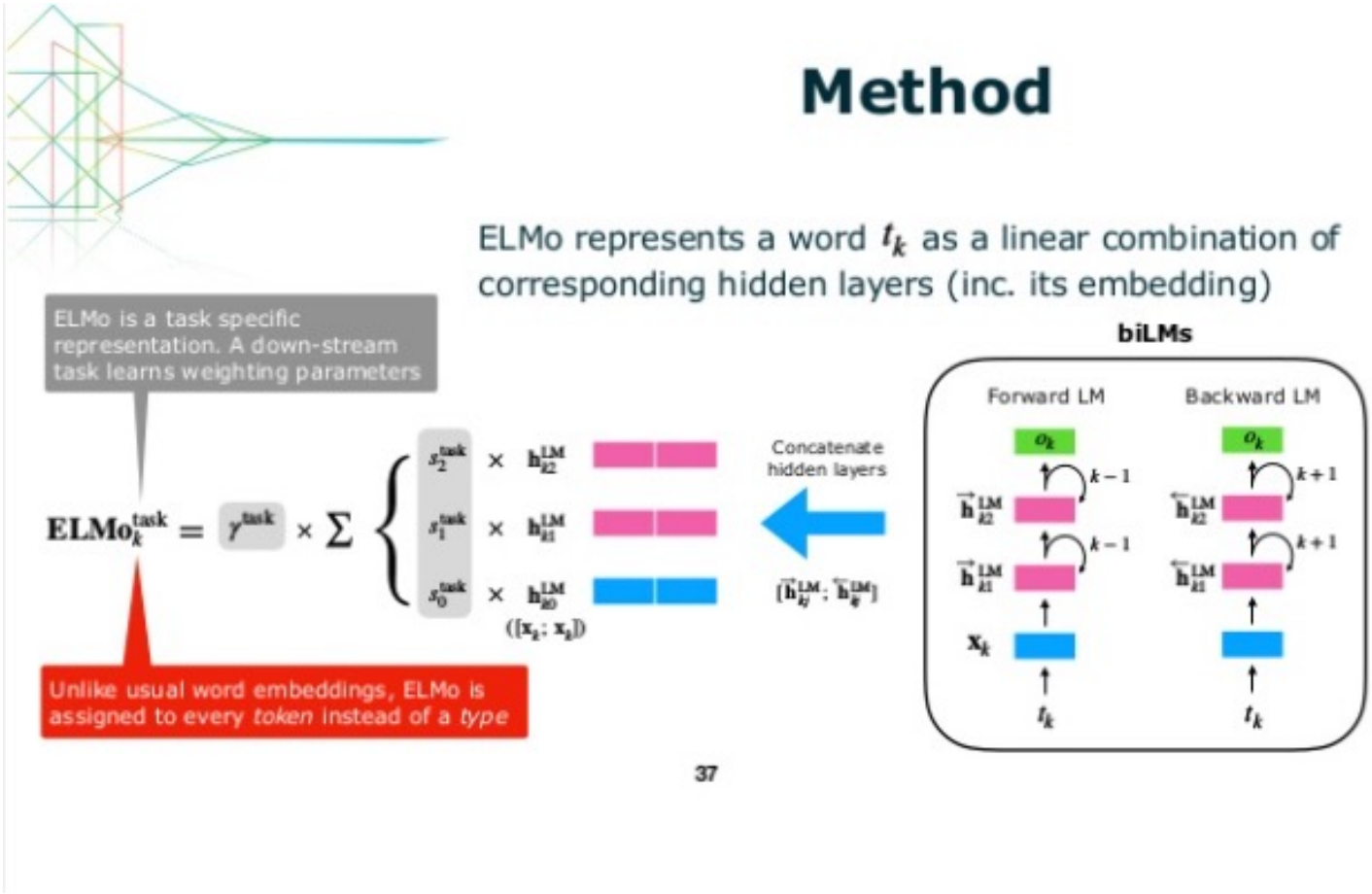
ELMo Deep Model

- Stack L biLSTMs on top of each other
- inputs to second layer are contextualised by the words either side of a target word
- inputs to third layer are contextualised by 2 words either side of target word
- higher layers better at capturing longer-range semantic dependencies



ELMo Embeddings

- Each layer learns its own vector representation for every token
- These are combined and a final softmax layer is trained to collapse these vectors into a single vector and a task-specific scalar
- Trained ELMo word embeddings often used as input to another neural model!



Slide from: https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018

ELMo Fine Tuning Tasks

- Prediction of word-level tags
  - NER, semantic role labelling (SRL)
- Prediction of sentence-level labels
  - sentiment analysis (SST-5)
- Prediction of sentence-pair labels
  - textual entailment / natural language inference (SNLI), question-answering (SQuAD)

ELMo Results

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%

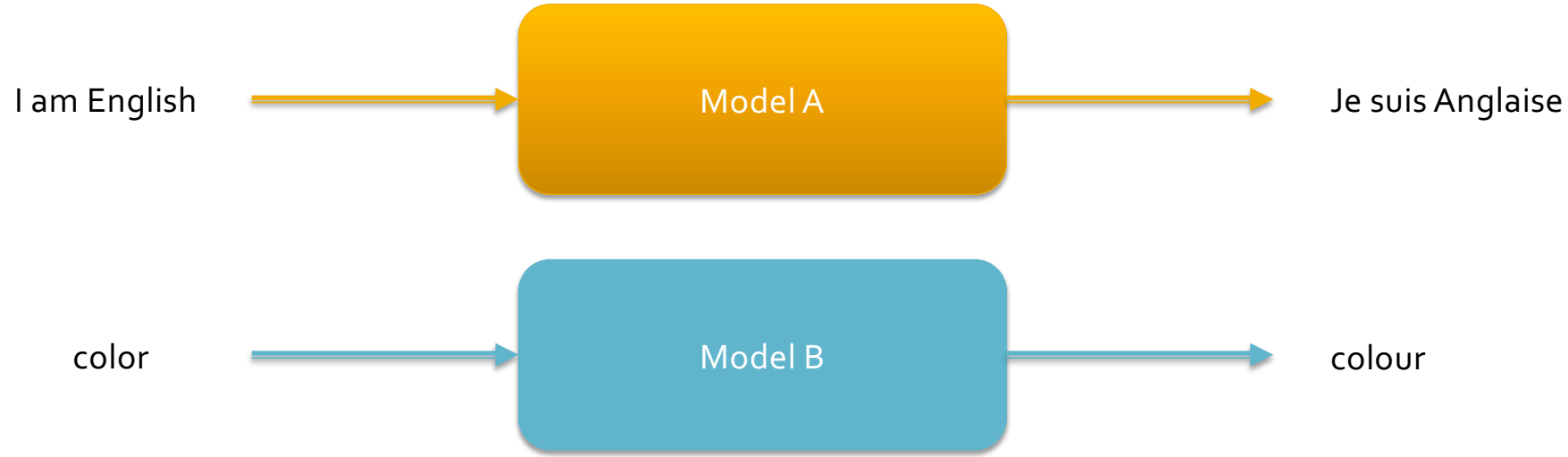
Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

Transformers (Vashwani et al. 2017)

- Sequence transduction model using stacked self-attention
  - no convolutions or recurrence
  - easier to parallelize than RNNs
  - faster to train than RNNs
  - captures more long-range dependencies than CNNs with fewer parameters
- What's a sequence transduction model?
- What's stacked self-attention?

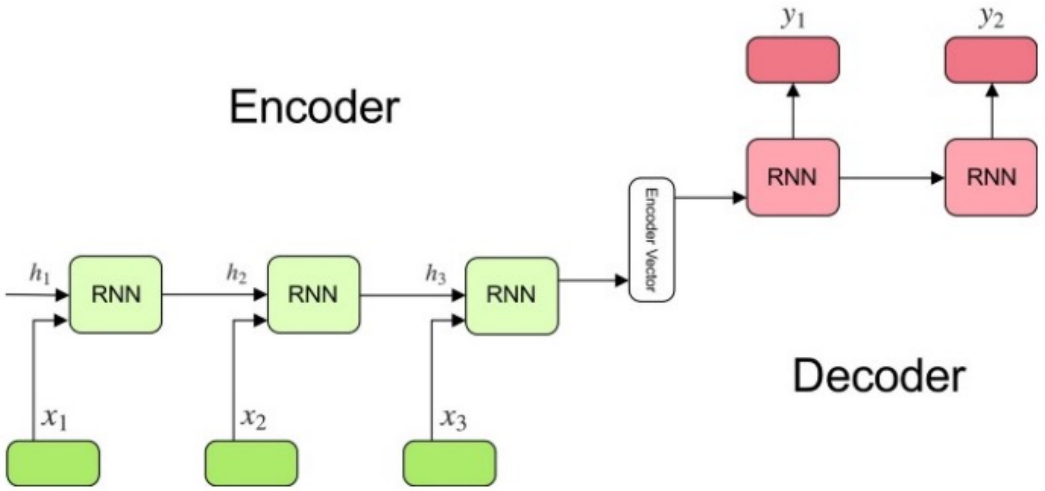
Sequence transduction

- Sequence to sequence models
  - Input: Sequence A
  - Output: Sequence B
- learning to convert one string into another



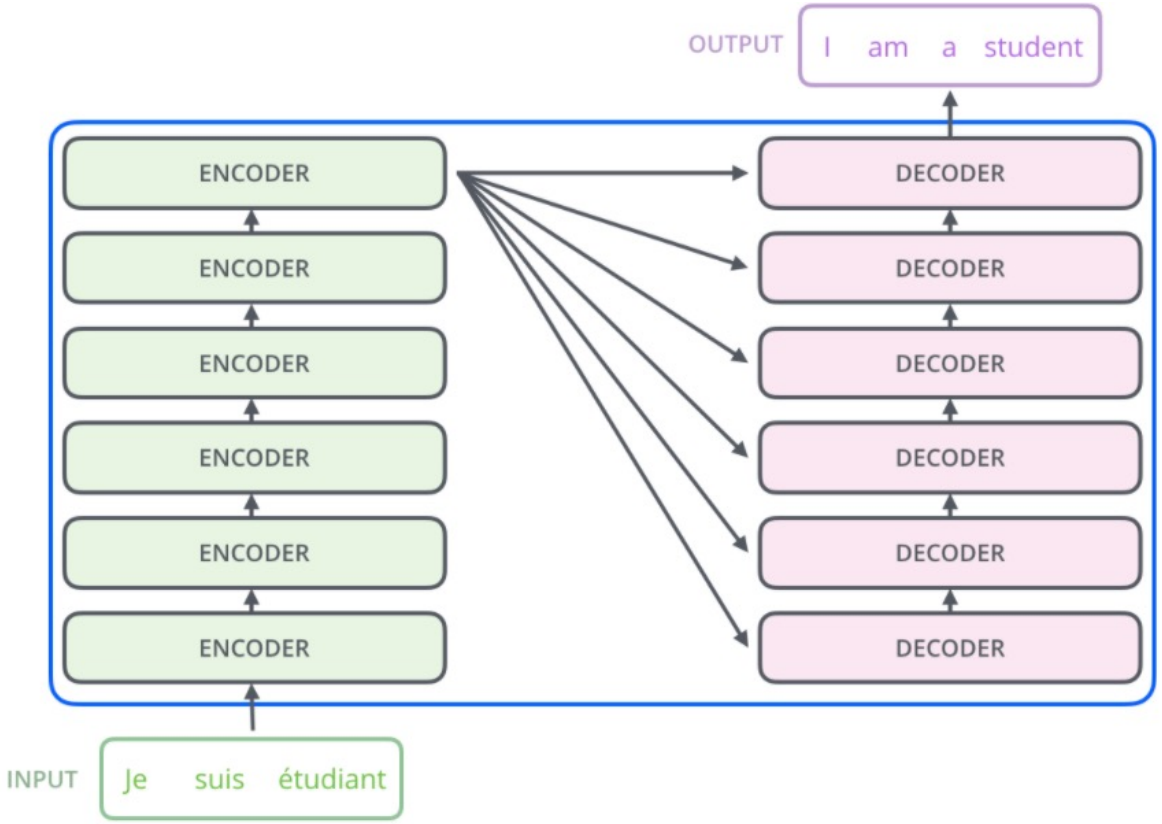
Encoder – Decoder Networks

- **Encode** the input sequence into a point in some latent space (*encoder vector*)
- **Decode** the point in latent space (*encoder vector*) into the output sequence
- The encoders and decoders can be any kind of neural networks, typically RNNs or attention-based



Encoder-decoder sequence to sequence model

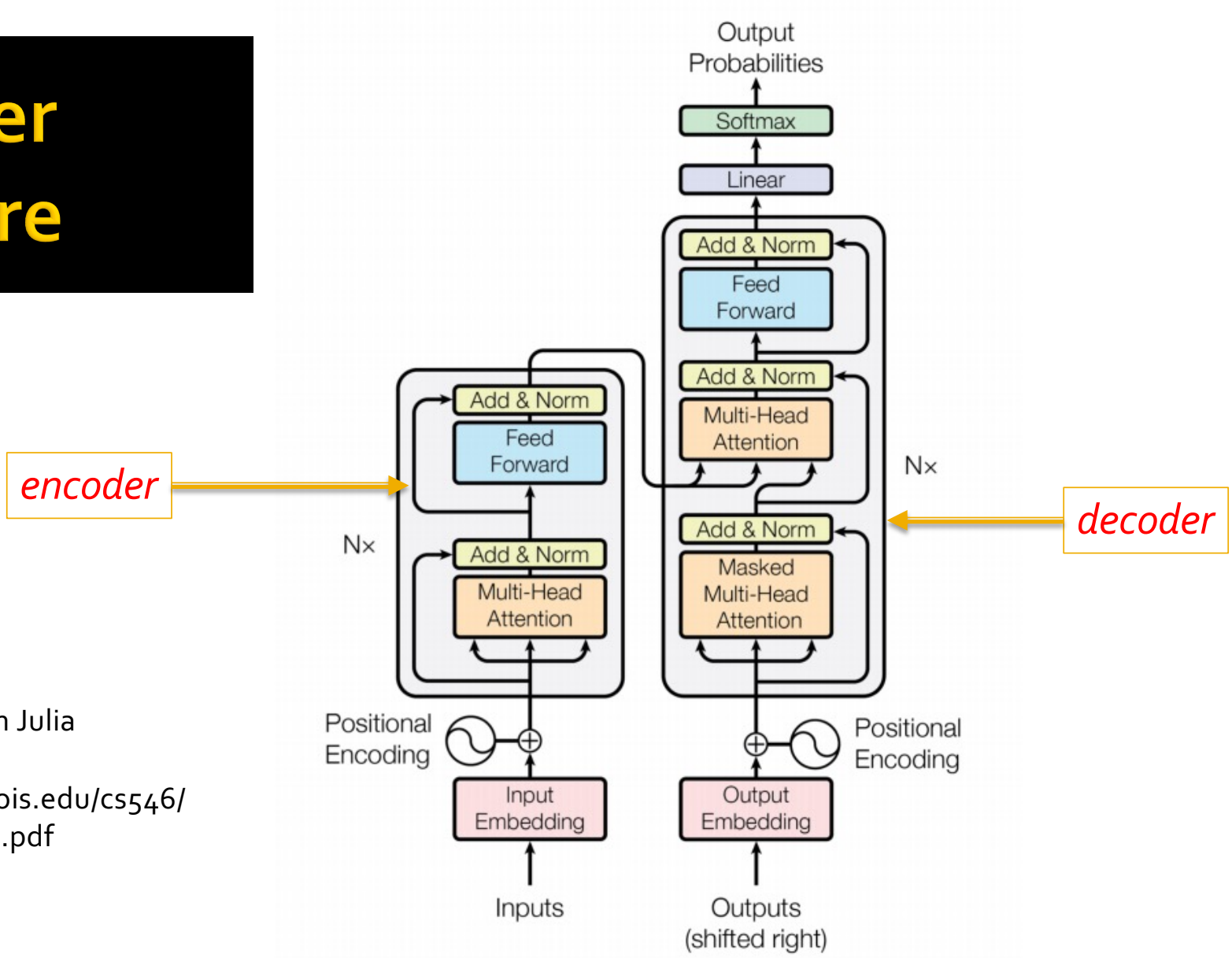
Stacked Encoder and Decoder Networks



typically:

- same number of decoders as encoders
- typically 6 of each
- each decoder as access to the output of the last encoder (the *encoder vector*) as well as the previous decoder in the stack

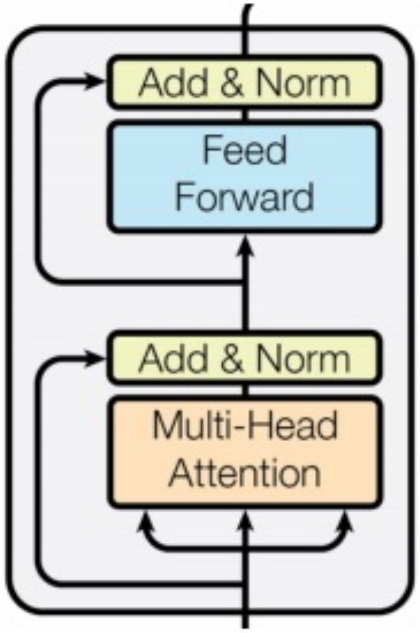
Transformer Architecture



Diagrams borrowed from Julia Hockenmaier: https://courses.engr.illinois.edu/cs546/sp2020/Slides/Lecture09.pdf

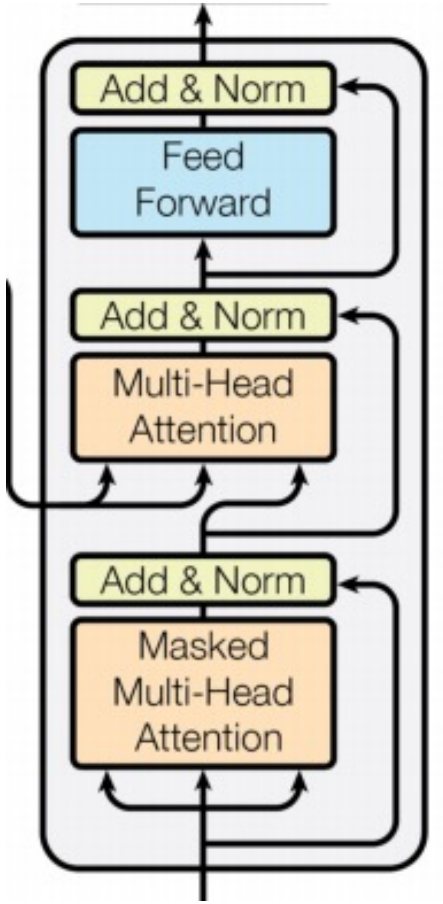
Encoder

- a stack of N=6 identical layers
- all layers and sublayers are 512-dimensional
- Each layer consists of **2** sublayers
  - one multi-headed self-attention layer
  - one position-wise fully connected layer
- Each sublayer has a residual connection and is normalised:
  - LayerNorm(x+Sublayer(x))



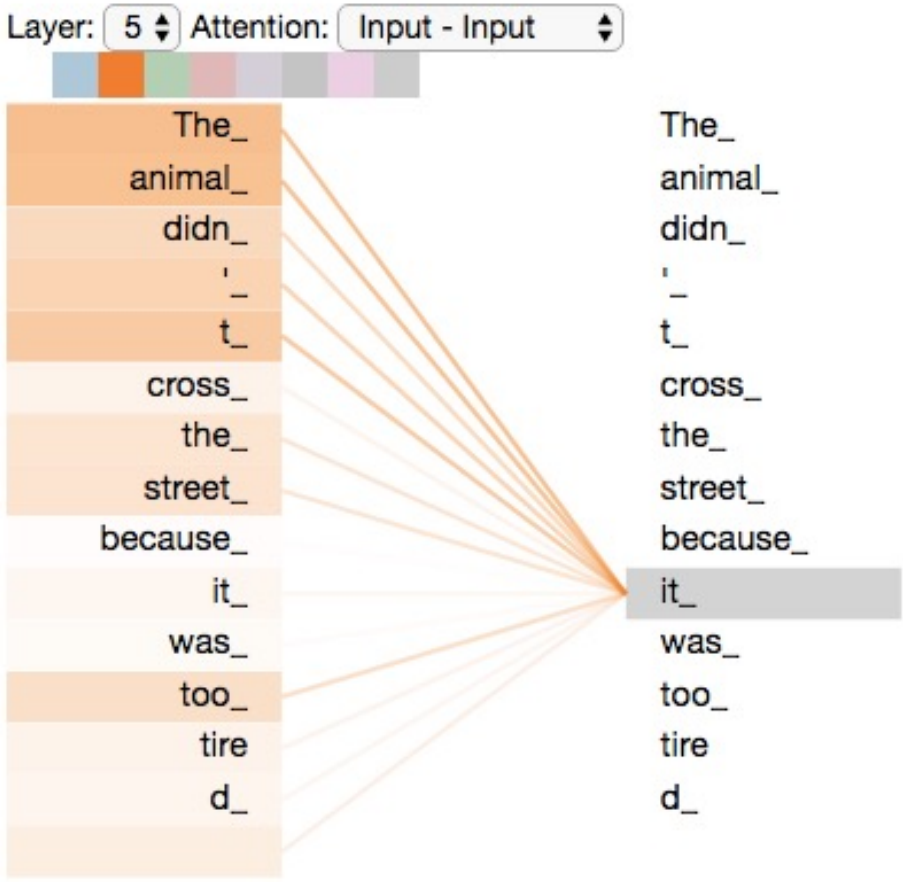
Decoder

- stack of N=6 identical layers
- all layers and sublayers are 512-dimensional
- Each layer consists of **3** sublayers:
  - one multi-headed self-attention layer over decoder output (ignore future tokens)
  - one multi-headed attention layer over encoder output
  - one position-wised fully connected layer
- each sublayer has a residual connection and is normalised
  - LayerNorm(x+Sublayer(x))



Self-Attention (High level)

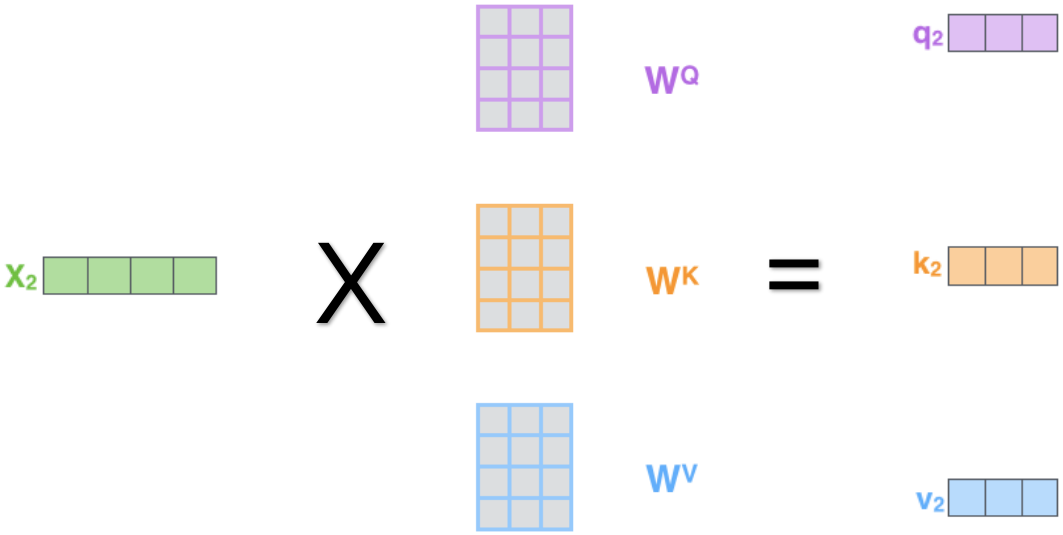
- What to pay attention to when encoding or decoding
  - **Attention**: in previous layer
  - **Self-attention**: in the same layer
- When encoding the word ***it*** in the sentence "*The animal didn't cross the street because* ***it*** *was too tired*", self-attention should allow the model to associate ***it*** with ***animal***



Example from http://jalammar.github.io/illustrated-transformer/

Self-attention - Step 1

- Step 1: from the encoder's input vector (e.g., word embedding), create 3 vectors: **Q**uery, **K**ey and **V**alue
- Q,K, and V embeddings usually smaller in number of dimensions e.g.,
  - input 512
  - output 64



Self-attention score

- Step 2: To work out how much the word in position$_i$ (e.g., "*it*") should pay attention to a word in position$_j$ (e.g., "*animal*") words in the sentence, we take the **dot product** of the Q_i (the **query** vector for *it*) with the K_j (the **key** vector for *animal*)

$$SelfAtt(w_i, w_j) = Q_i . K_j$$

- Do this for every word in the sentence with every other word in the sentence

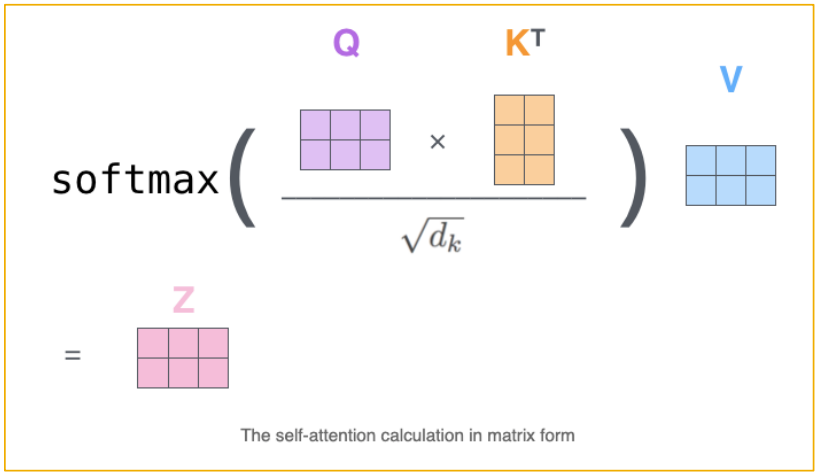
Normalise

- Step 3: Divide all of the scores by 8
  - the square root of the dimension of the key vectors which is 64 here
  - this is the default and seems to lead to more stable gradients
- Step 4: Apply a softmax to the scores
  - this results in them all being positive
  - and summing to 1
  - so can be thought of a as a probability / proportion – "what proportion of my attention should I pay to word j?"

Weight sum of the value vectors

- Step 5: For each word w_i, multiply the value vector (V_j) of each other word w_j by its softmax score with w_i
- Step 6: Sum to produce output embedding (at this layer) for w_i

$$Z_i = \sum_j softmax\_score(w_i, w_j) \times V_j$$



Self-attention summary

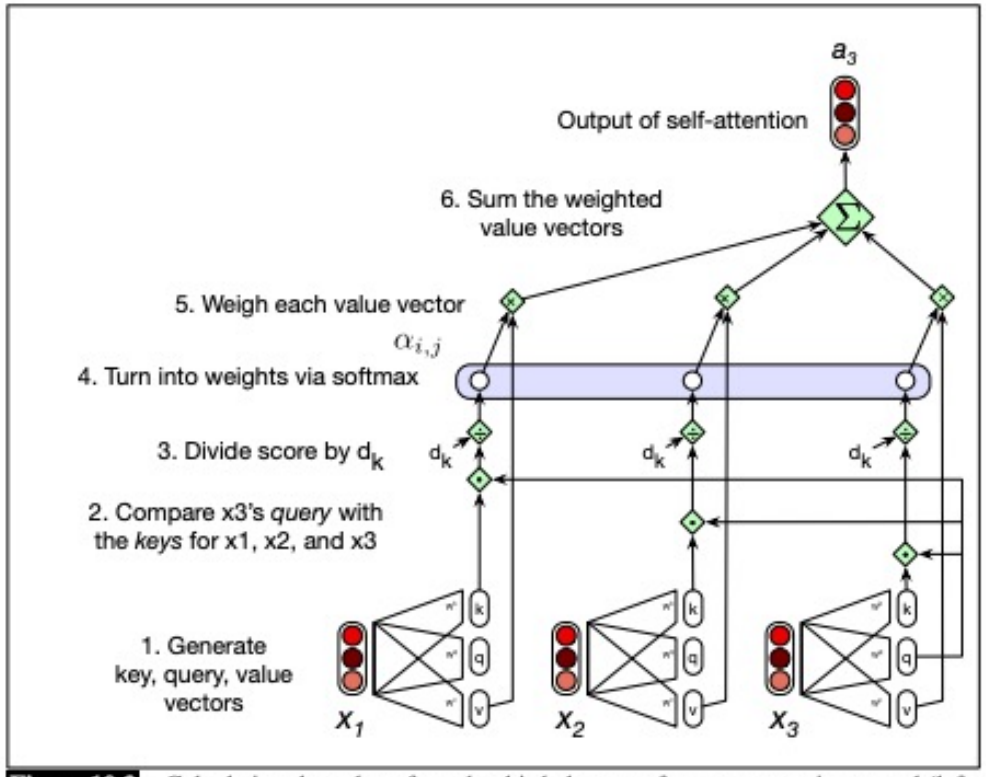


Figure 10.3 Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

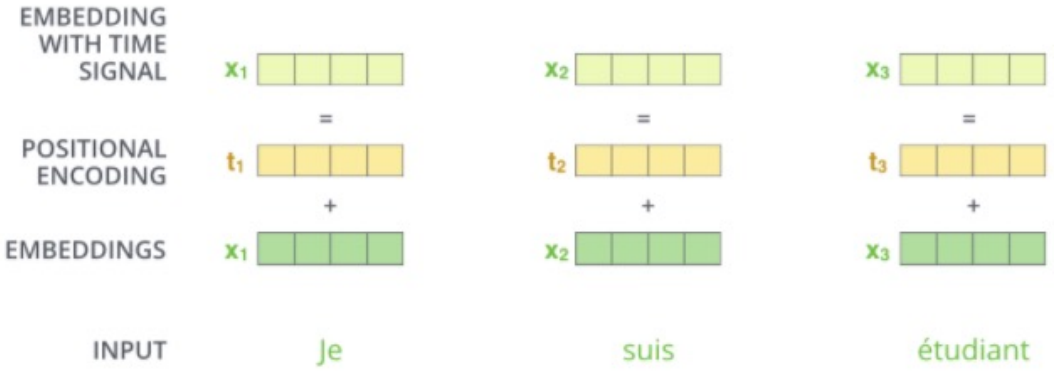
- Image taken from Chapter 10, Jurafsky and Martin
- This is actually for uni-directional self-attention
- However, it would be the same for the a_3 in the bidirectional case assuming an input sequence which is 3 tokens long!

Multi-head attention

- Paying attention to multiple things at the same time!
- Multiple "representation subspaces"
- Multiple sets of Query/Key/Value weight matrices
- Typical transformer uses 8 attention heads per self-attention layer
  - that's 8 sets of Q,K,V matrices for each encoder and 16 for each decoder
- All randomly initialised
- Embeddings produced by multiple heads are concatenated and then multiplied by an additional weights matrix before passing to the feed-forward layer

Positional Encoding

- How does the model account for word order?
- Adds a positional encoding vector to each input embedding
- Positional encoding follows a fixed pattern
  - enables the model to determine the position of each word and distance between them

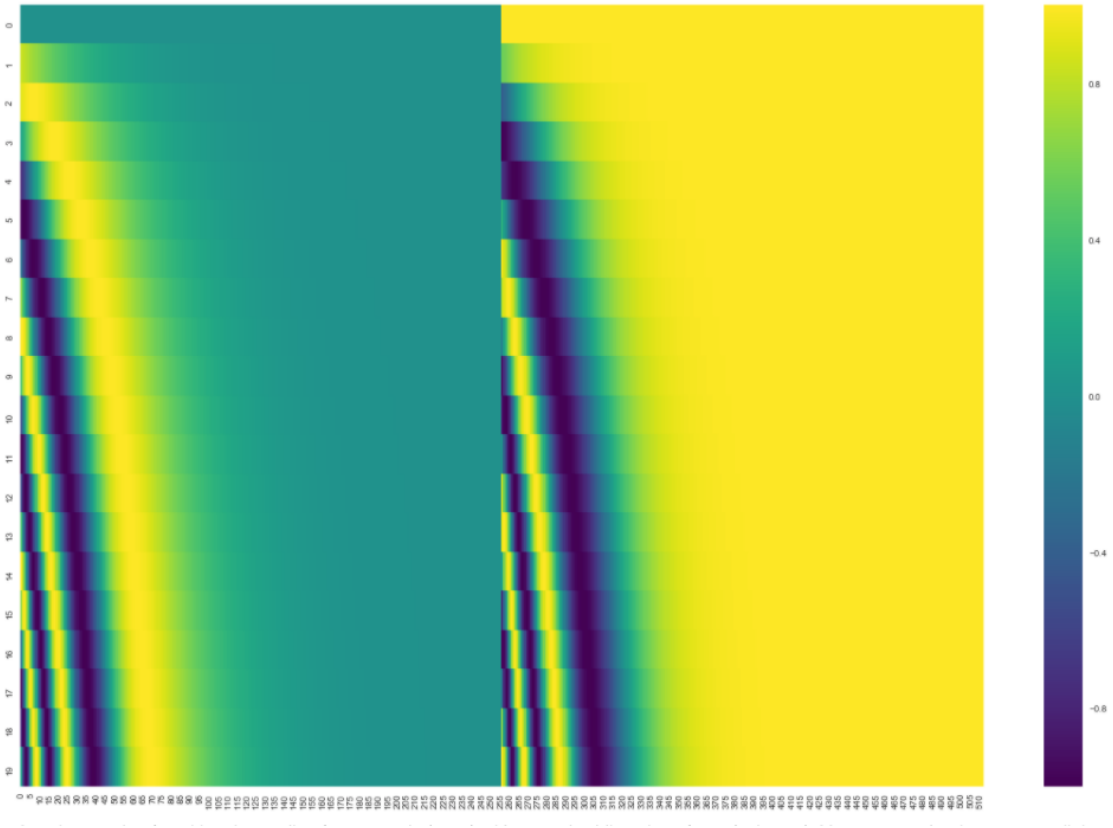


Absolute position

- Start with randomly initialized embeddings corresponding to each possible input position up to some maximum length
- We have embeddings for
  - words e.g., student
  - Positions e.g., position 3
- Positional embeddings are learned / updated during training
- Alternatives to absolute position include
  - Static function maps integer inputs to real-valued vectors

    Original transformers used combinations of sines and cosines
  - relative position

Positional encoding visualisation

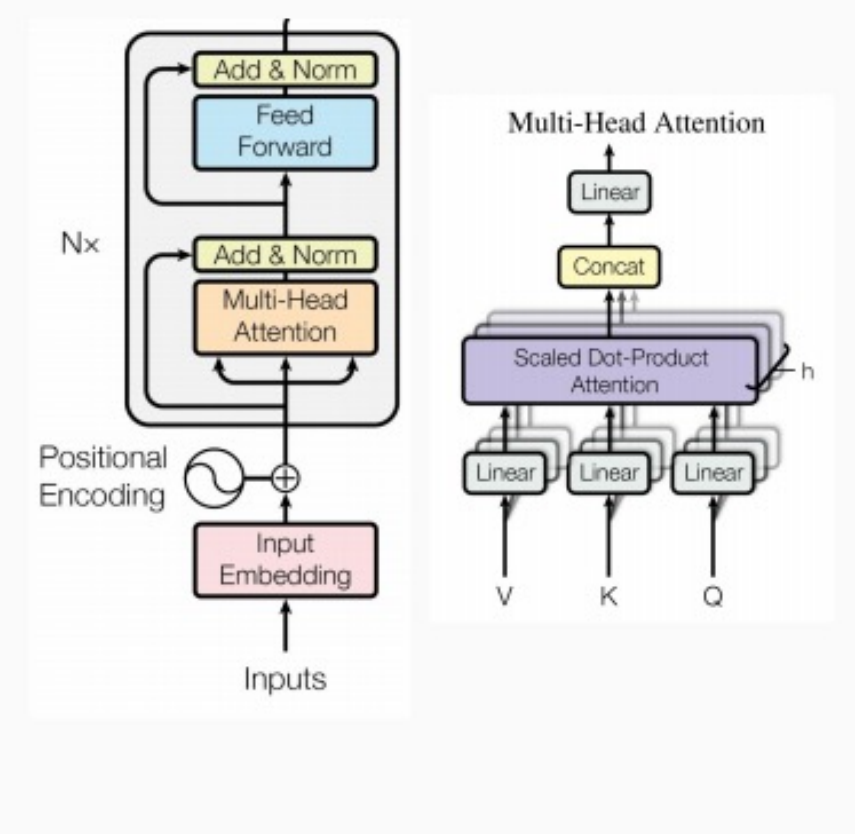


BERT (Devlin et al. 2019)

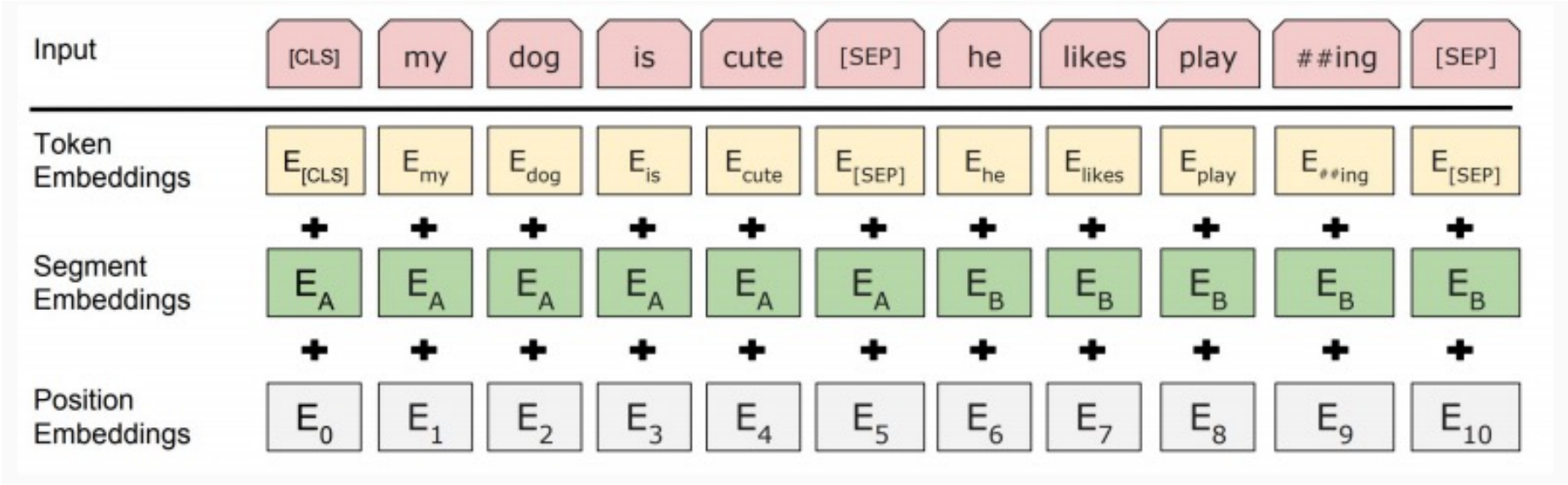
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - just uses the encoder portion of the transformer architecture
  - uses left and right context of a target word to build representation
- Pre-trained on general language modelling tasks
  - masked language modelling task
  - next sentence prediction
- fine-tuned on task-specific training data
- Pre-trained models available from Google:
  - https://github.com/google-research/bert
- And from huggingface!

Encoder Representation

- Multi-headed self-attention
  - models context
- Feed-forward layers
  - non-linear hierarchical features
- Layer norm and residuals
  - makes training deep networks healthy
- Positional embeddings
  - learn relative positioning

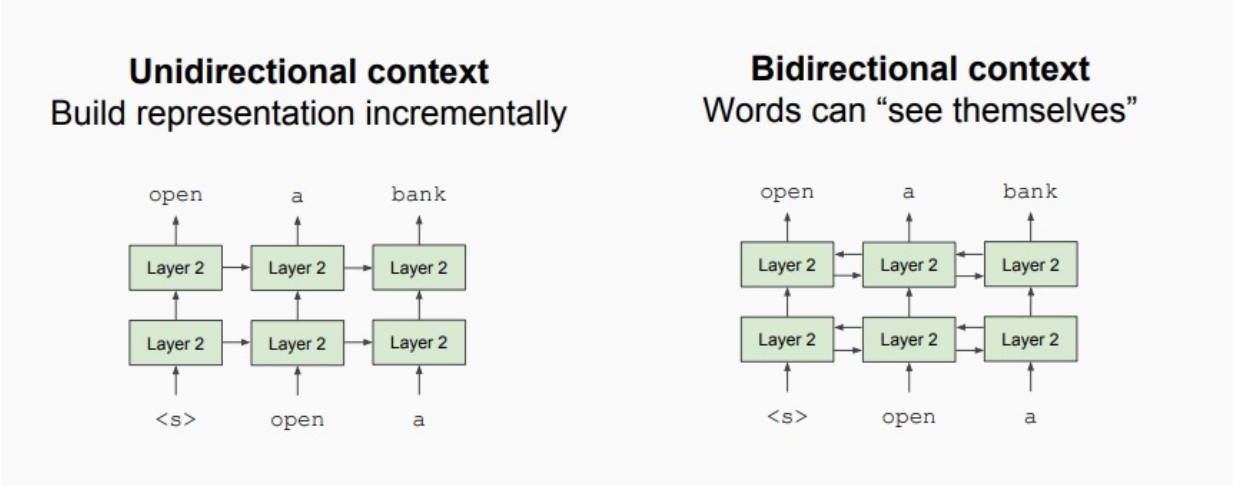


Input Representation



- Each token is sum of three embeddings
- 30,000 WordPiece vocabulary → morphology → better representations for rare words
- sentences are separated by a special "SEP" token
- a special "CLS" token is added at the beginning of the chunk
  - often used in classification

Unidirectional vs Bidirectional models

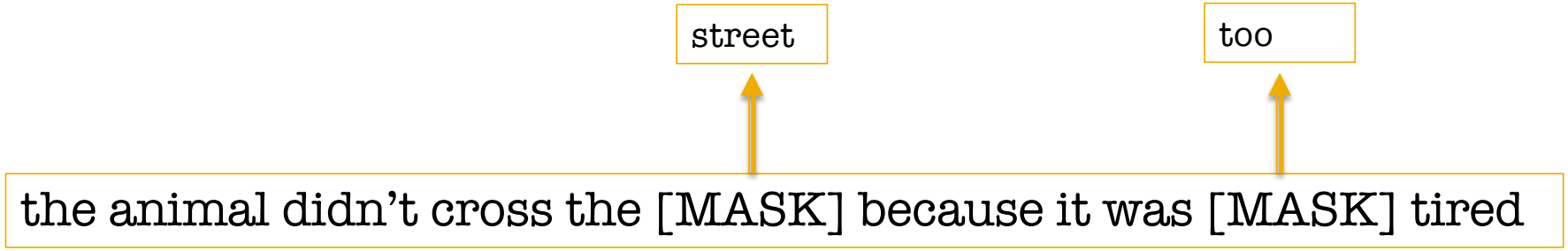


from https://nlp.stanford.edu/seminar/details/jdevlin.pdf

- Most language models trained to predict the next word in sequence
- Known as **auto-regressive** or **unidirectional**
- BERT uses the whole of the sentence to predict missing or masked word
- masked language model is **bidirectional**

Masked Language Model

- Mask out k% of the input words, and predict the masked words
  - if k is too small → expensive to train
  - if k is too large → not enough context
  - Google use k=15



Next Sentence Prediction

- To learn relationships between sentences, predict whether sentence B is actual sentence that follows sentence A or is a random sentence

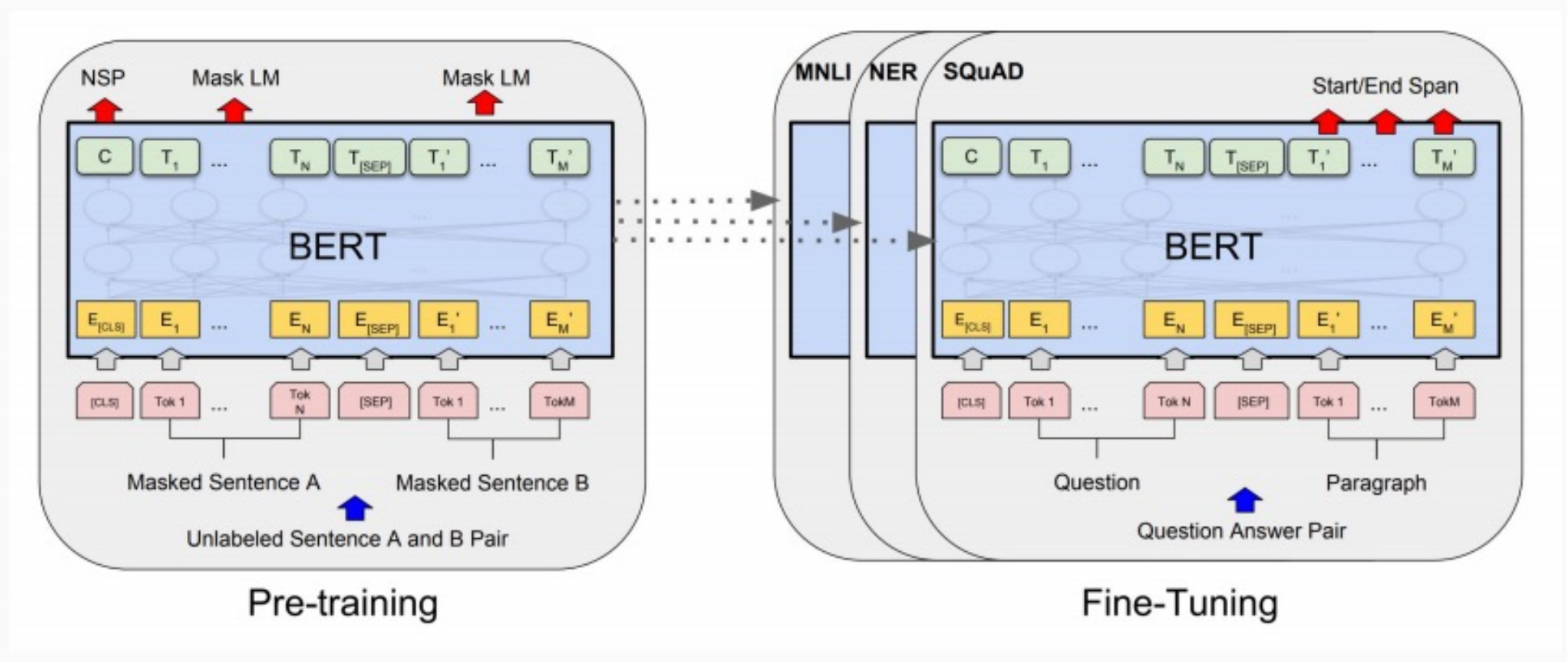
sentence A = The man went to the store
sentence B = He bought a gallon of milk
Label = IsNextSentence

sentence A = The man went to the store
sentence B = Penguins are flightless
Label = NotNextSentence

Pre-trained model details

- Data: Wikipedia (2.5B words) + BookCorpus (800M words)
- Batch Size: 131,072 words (1024 sequences * 128 length or 256 sequences * 512 length)
- Training Time: 1M steps (~40 epochs)
- Optimizer: AdamW, 1e-4 learning rate, linear decay
- `BERT-Base`: 12-layer, 768-hidden, 12-head
- `BERT-Large`: 24-layer, 1024-hidden, 16-head
- Trained on 4x4 or 8x8 TPU slice for 4 days

Fine-tuning Procedure



GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

Sentence: The car honked down the road.
Label: Unacceptable

Further reading

- Devlin et al. (2019): Pre-training of Deep Bidirectional Transformers for Language Understanding in NAACL 2019, https://www.aclweb.org/anthology/N19-1423/
- Peters et al. (2018): Deep contextualised word representations in Proceedings of NAACL 2018 https://arxiv.org/pdf/1802.05365.pdf
- Peters et al. (2018): Dissecting Contextual Word Embeddings: Architecture and Representation in Proceedings of EMNLP 2018 https://www.aclweb.org/anthology/D18-1179.pdf
- Reimers et al. (2019):
- Vashwani et al. (2017): Attention is all you need in Proceedings of NIPS 2017
- Yang et al. (2020) :

(https://arxiv.org/pdf/2004.12297.pdf)