Intelligent Agents

1d-CNNs LSTMs ELMo Transformers BERT GPT

Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Acknowledgements

- Some slides are based on
  - CS546: Machine Learning in NLP (Spring 2020)
    - *http://courses.engr.illinois.edu/cs546/*
    - Julia Hockenmaier http://juliahmr.cs.illinois.edu
    - RNNs, LSTMs, ELMo, Transformers
  - Machine Learning (Spring 2020)
    - http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html
    - 李宏毅 (Hung-yi Lee) http://speech.ee.ntu.edu.tw/~tlkagk/
    - ELMo, BERT: http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/BERT%20(v3).pdf

- Respective sources are indicated in the gray line at the bottom
- Slides have been modified
  - All errors are mine

Recap: Convolution

Input image

Convolution Kernel

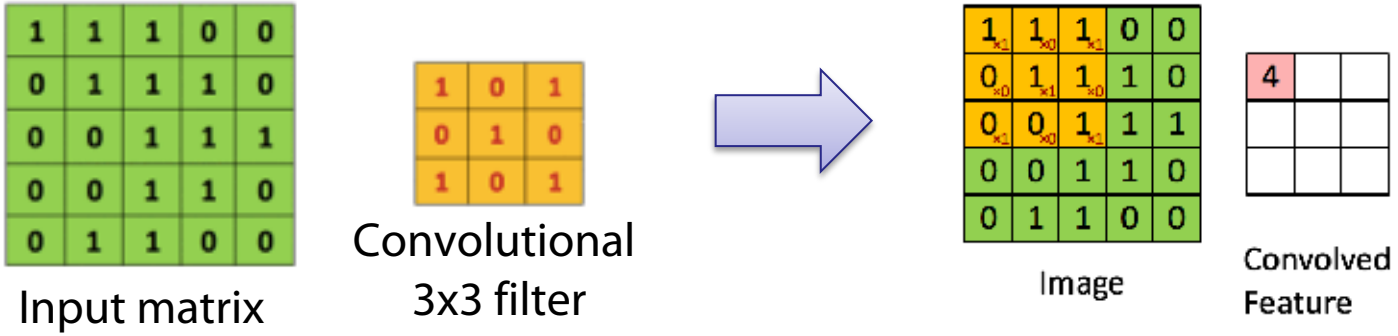
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

Recap: Convolutional Neural Networks (CNNs)

Main CNN idea for text:
Compute vectors for n-grams and group them afterwards

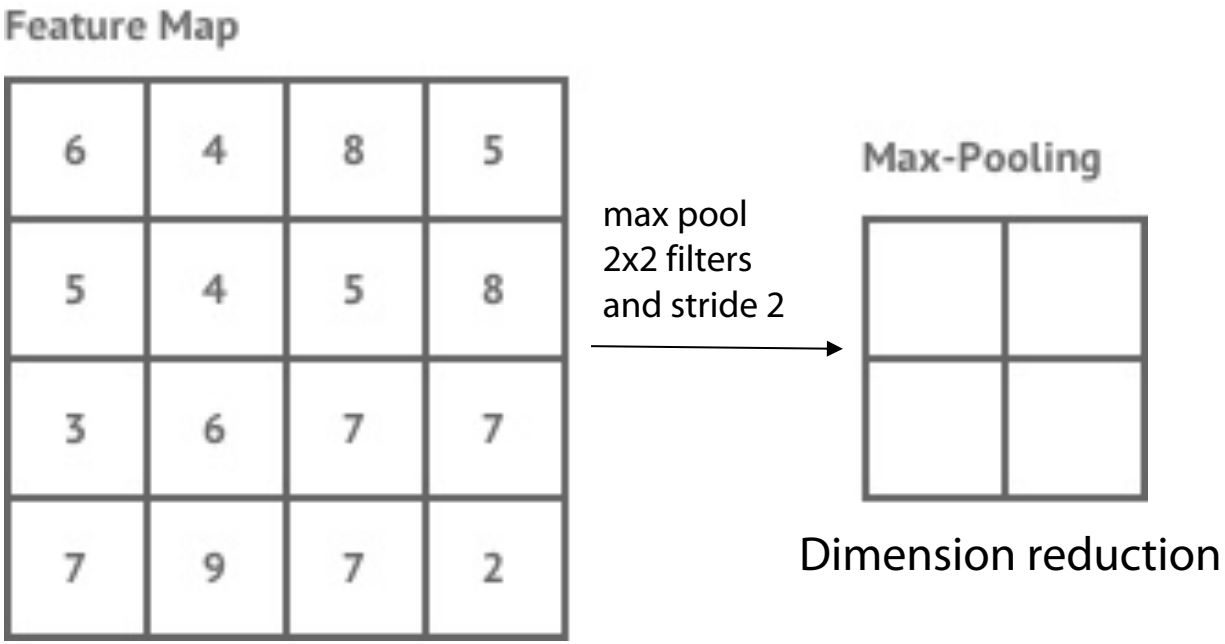
Example: “this takes too long” compute vectors for:
This takes, takes too, too long, this takes too, takes too long, this takes too long



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Recap: ConvNets (CNNs)

Main CNN idea for text:
Compute vectors for n-grams and **group them afterwards**



https://shafeentejani.github.io/assets/images/pooling.gif

1d-CNNs for text

Text is a (variable-length) sequence of words (word vectors)

We can use a 1d-CNN to slide a window of n tokens across:

- filter size n = 3, stride = 1, no padding

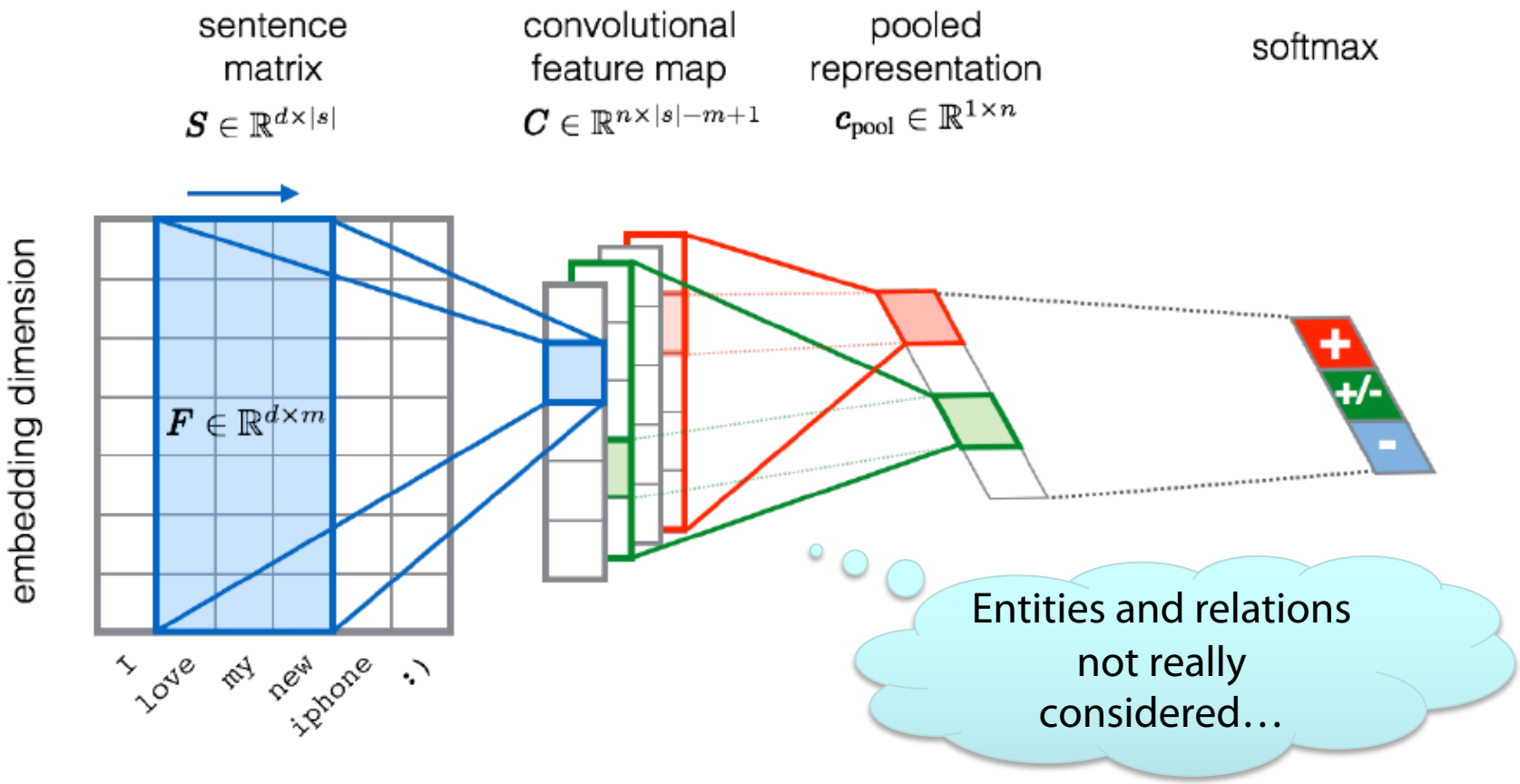
```
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
```

- filter size n = 2, stride = 2, no padding:

```
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
```

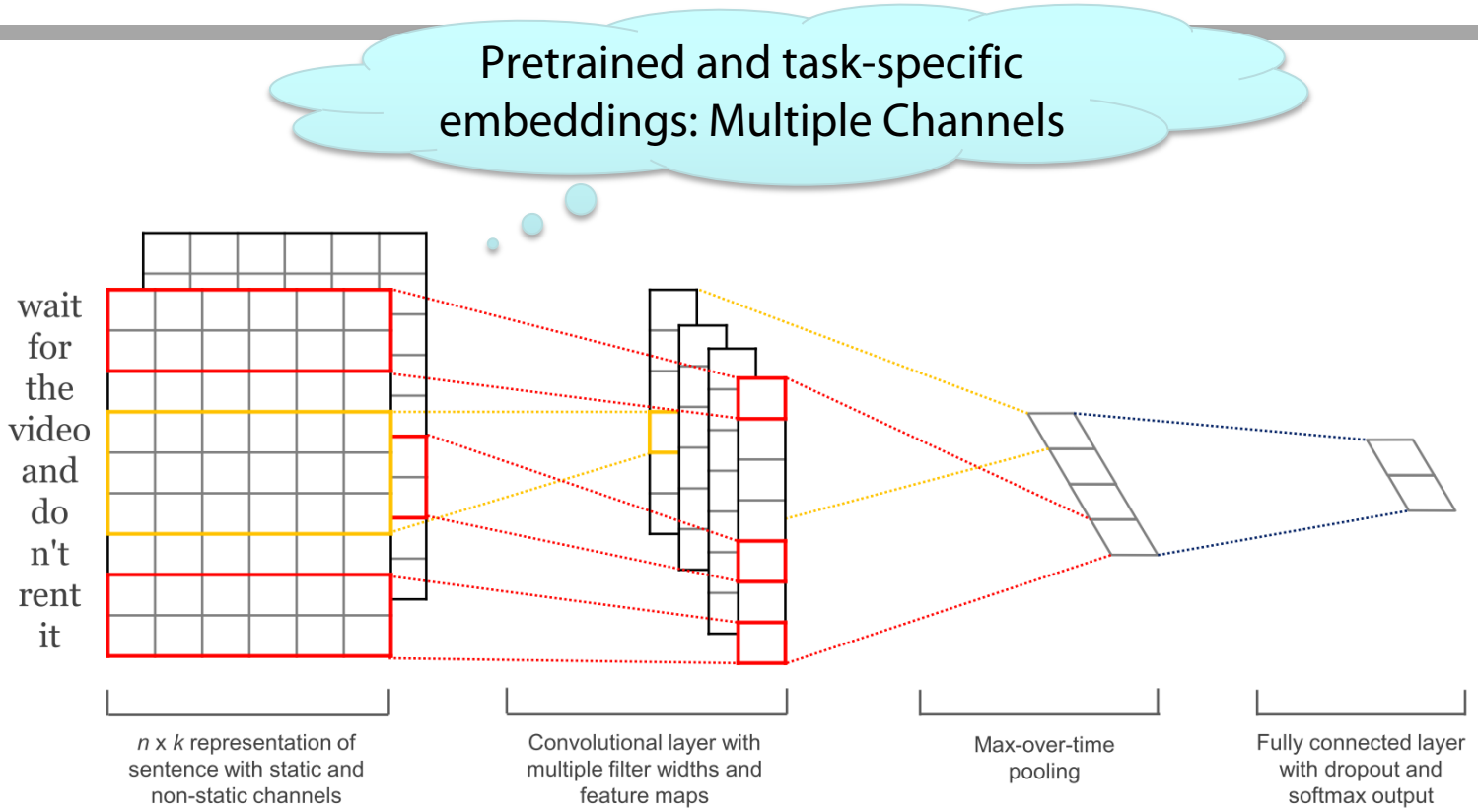
CNNs (w/ ReLU and maxpool) can be used for classifying (parts of) the text

CNNs for sentiment analysis



Severyn, Aliaksei, and Alessandro Moschitti. "UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification." *SemEval@ NAACL-HLT*. 2015.

CNNs for sentence/text classification



Kim, Y. "Convolutional Neural Networks for Sentence Classification", EMNLP (2014)

sliding over 3, 4 or 5 words at a time

Static = pre-trained, non-static = task-specific

Fasttext (https://fasttext.cc)

- Library for word embeddings and text classification
  - static word embeddings and ngram features
  - that get averaged together in one hidden layer
  - hierarchical softmax output over class labels

- Enriching word vectors with subword information
  - Skipgram model where each word is a sum of character ngram embeddings and its own embedding
  - Each word is deterministically mapped to ngrams

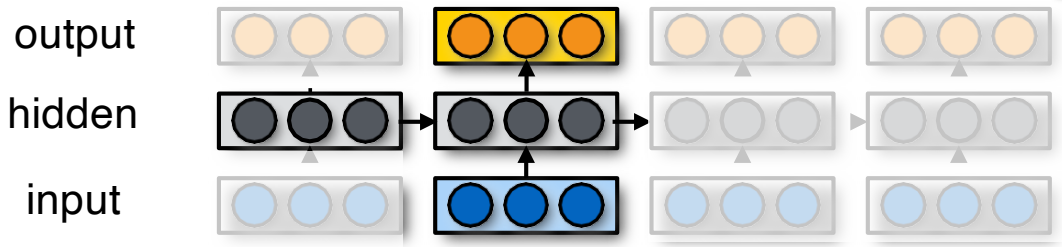
Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, Volume 5. 135-146. **2017.**

Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, Bag of Tricks for Efficient Text Classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. 427-431. **2017**.

Alon Jacovi, Oren Sar Shalom, Yoav Goldberg. Understanding Convolutional Neural Networks for Text Classification. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. **2018**.

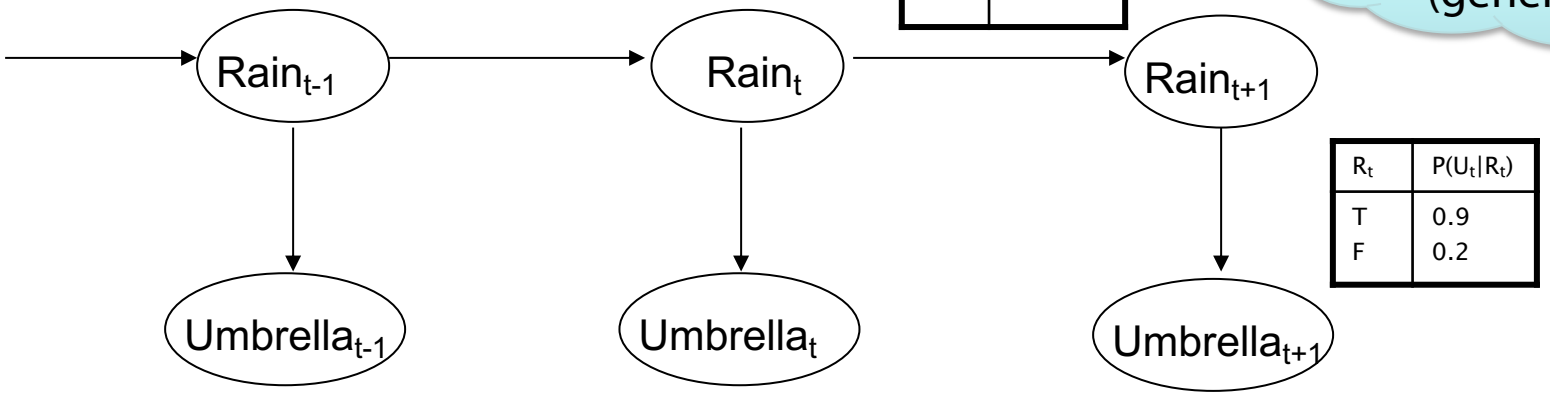
Recursive Networks – Or: Copying the Pattern

- Basic computational network copied per time slice
- Input: previous hidden state, output: next hidden state



Computational model

- Compare with HMM:



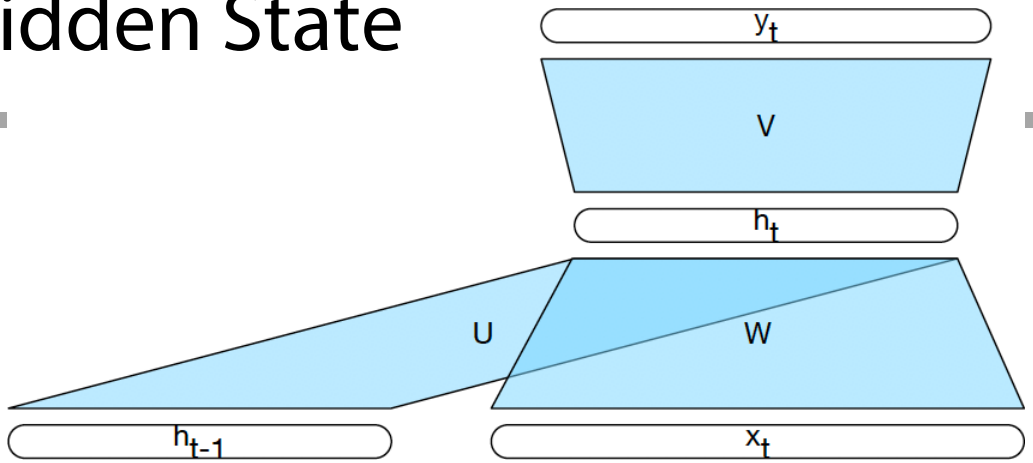
R_{t-1}	$P(R_t \mid R_{t-1})$
T	0.7
F	0.3

R_t	$P(U_t \mid R_t)$
T	0.9
F	0.2

Declarative model (generative)

Computing the Hidden State

- RNN



Computing the hidden state at time t: $\mathbf{h}^{(t)} = g(\mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{W}\mathbf{x}^{(t)})$

The i-the element of $\mathbf{h}_t$: $h_i^{(t)} = g\left(\sum_j U_{ji} h_j^{(t-1)} + \sum_k W_{ki} x_k^{(t)}\right)$

- HMM Filtering

$$P(X_{t+1} / e_{1:t+1}) = \alpha P(e_{t+1} / X_{t+1}) \sum_{X_t} P(X_{t+1} / x_t) P(x_t / e_{1:t})$$

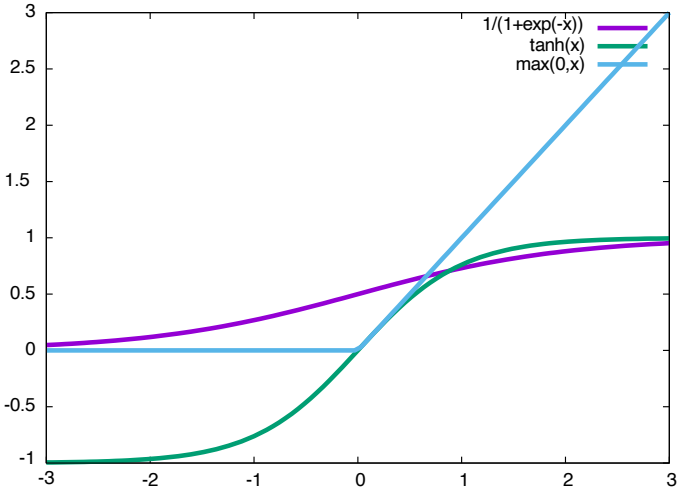
What about hindsight queries (aka smoothing)?

Recap: Activation Functions

Sigmoid (logistic function):

$\sigma(x) = 1/(1 + e^{-x})$

Returns values bound above and below in the $0,1$ range



Hyperbolic tangent:

$\tanh(x) = (e^{2x-1})/(e^{2x+1})$

Returns values bound above and below in the $-1, +1$ range

Rectified Linear Unit:

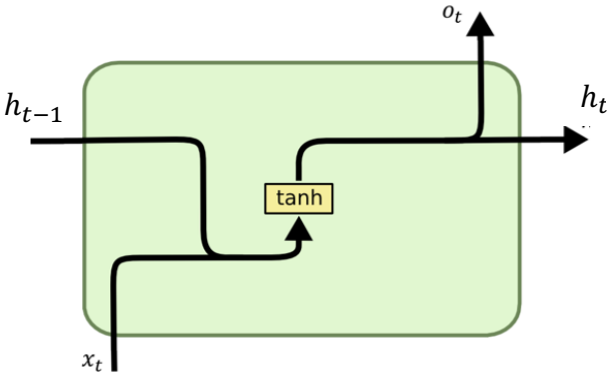
$\mathrm{ReLU}(x) = \max(0, x)$

Returns values bound below in the $0, +\infty$ range

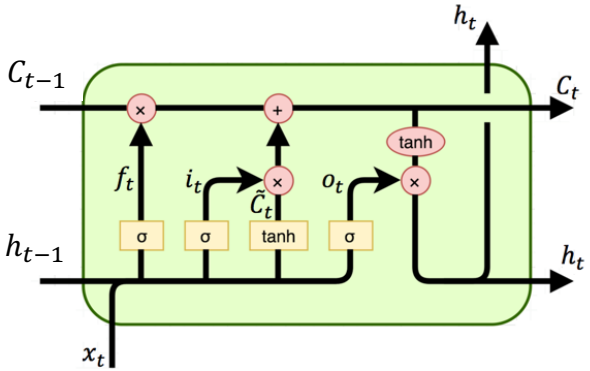
RNN Variants: LSTMs, GRUs

- Long Short Term Memory networks (LSTMs) are RNNs with a more complex architecture to combine the last hidden state with the current input.
- Gated Recurrent Units (GRUs) are a simplification of LSTMs
- Both contain "gates" to control how much of the input or past hidden state to forget or remember

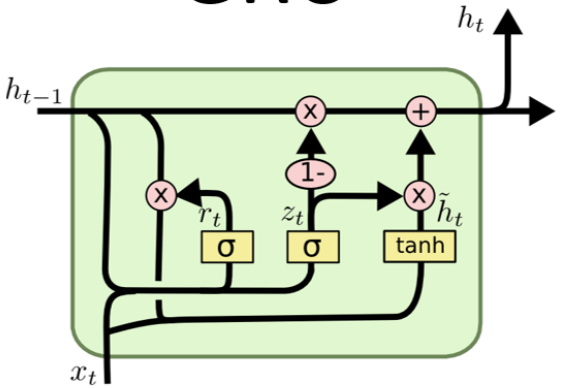
RNN



LSTM



GRU



Gates

- A gate performs element-wise multiplication of
  - the output of a *d*-dimensional sigmoid layer (all elements between 0 and 1), and
  - a *d*-dimensional input vector
- Result: a *d*-dimensional output vector which is like the input, except some dimensions have been (partially) "forgotten"

RNNs for Language Modeling

- If our vocabulary consists of V words, the output layer (at each time step) has V units, one for each word (one-hot encoding)
- The softmax gives a distribution over the V words for the next word
- To compute the probability of a string $w_0 w_1 \ldots w_n w_{n+1}$ (where $w_0 = \langle s \rangle$, and $w_{n+1} = \langle /s \rangle$), feed in w_i as input at time step i and compute

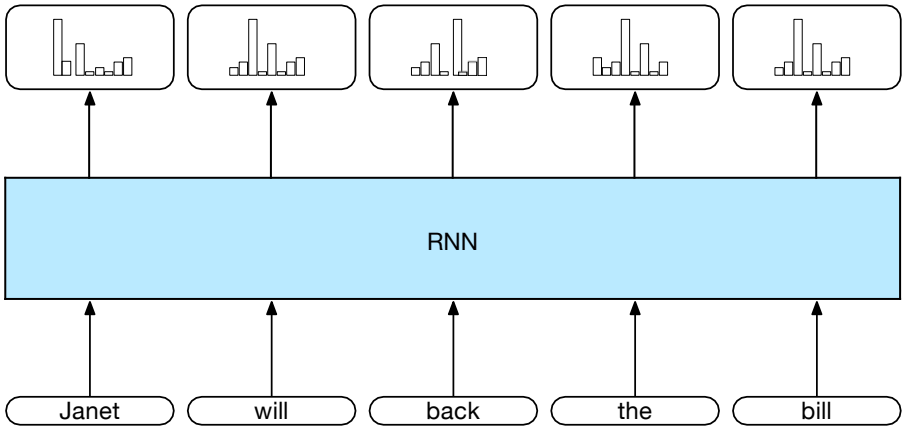
$$\prod_{i=1..n+1} P(w_i | w_0 \ldots w_{i-1})$$

RNNs for Sequence Labeling

- In sequence labeling, we want to assign a label or tag t_i to each word w_i
- Now the output layer gives a distribution over the T possible tags.
- The hidden layer contains information about the previous words and the previous tags.
- To compute the probability of a tag sequence $t_1 \ldots t_n$ for a given string $w_1 \ldots w_n$ feed in w_i (and possibly $t_{i\text{-}1}$) as input at time step i and compute $\mathrm{P}(t_i \mid w_1 \ldots w_{i\text{-}1}, t_1 \ldots t_{i\text{-}1})$

Basic RNNs for Sequence Labeling

Each time step has a distribution over output classes

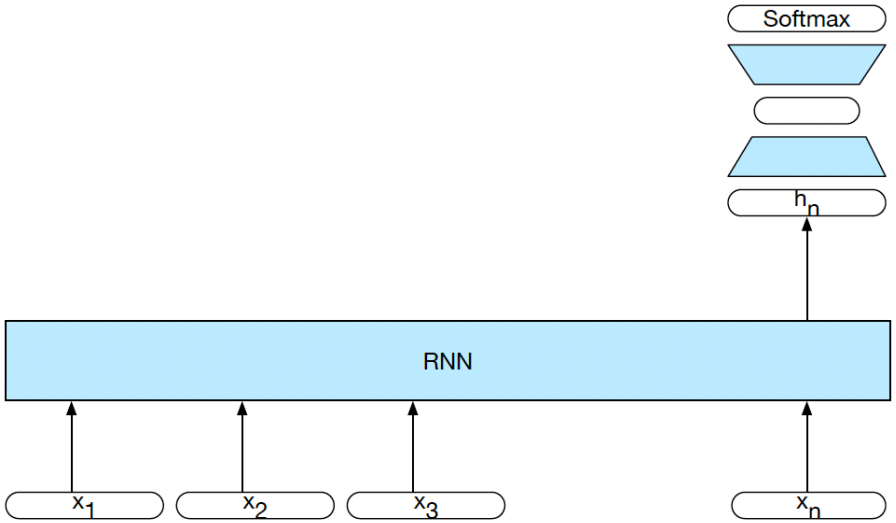


Extension: add a HMM/CRF layer to capture dependencies among labels of adjacent tokens.

RNNs for Sequence Classification

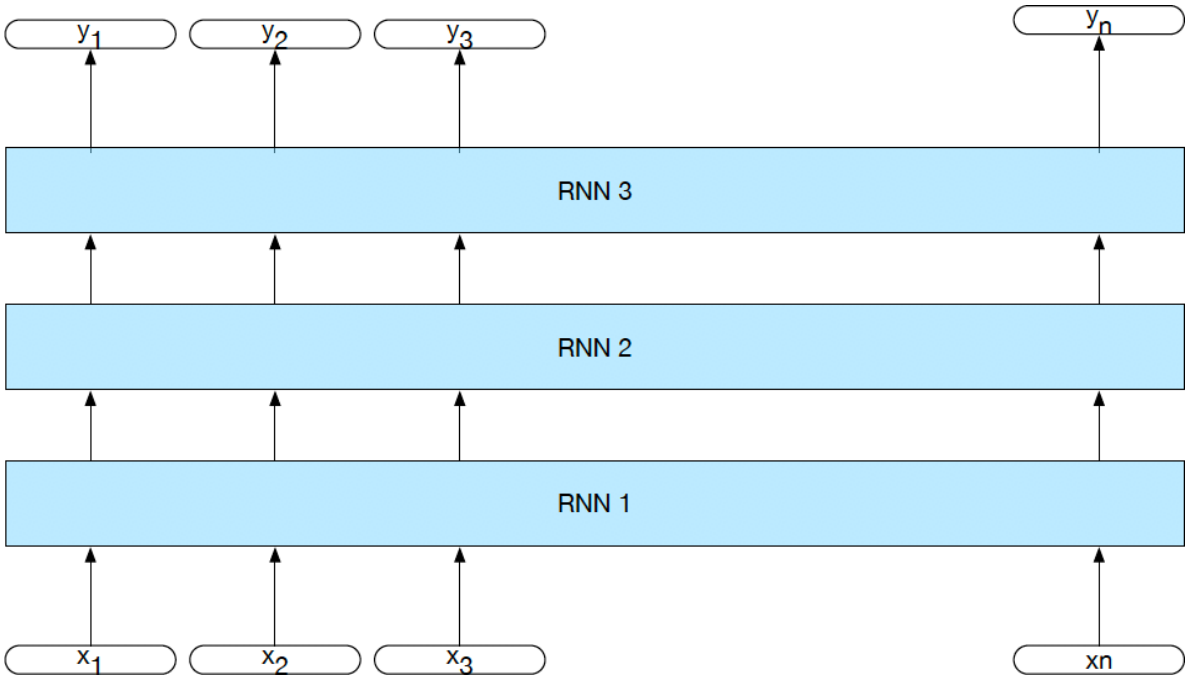
If we just want to assign a label to the entire sequence, we don't need to produce output at each time step, so we can use a simpler architecture.

We can use the hidden state of the last word in the sequence as input to a feedforward net:

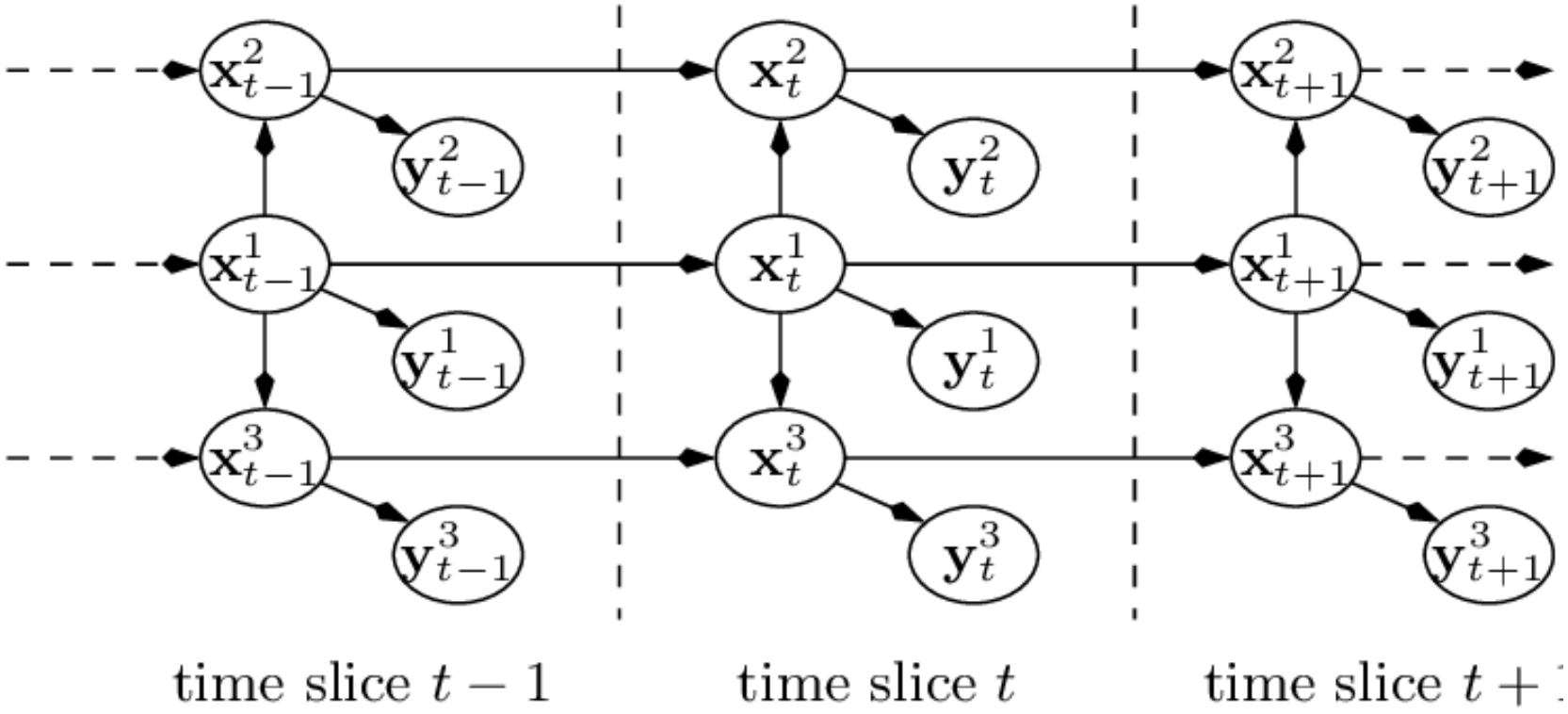


Stacked RNNs

We can create an RNN that has “vertical” depth (at each time step) by stacking multiple RNNs:



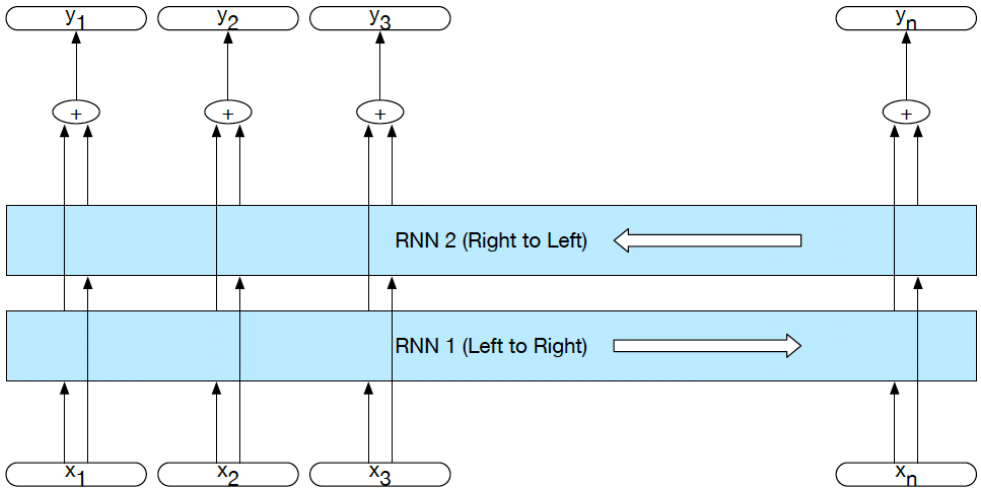
Comparison with Dynamic Baysian Networks



Bidirectional RNNs

Computational specification of smoothing?

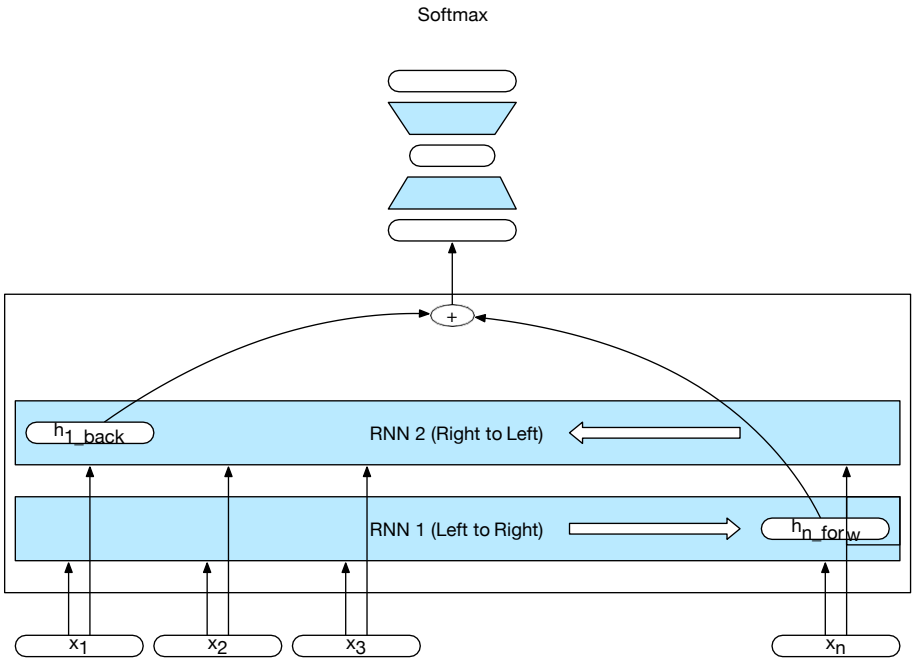
Unless we need to generate a sequence, we can run *two* RNNs over the input sequence — one in the forward direction, and one in the backward direction.
Their hidden states will capture different context information



Hidden state of biRNN: $\mathbf{h}_{\mathbf{bi}}^{(\mathbf{t})} = \mathbf{h}_{\mathbf{fw}}^{(\mathbf{t})} \oplus \mathbf{h}_{\mathbf{bw}}^{(\mathbf{t})}$ where $\oplus$ is typically concatenation (or element-wise addition, multiplication)

Bidirectional RNNs for sequence classification

Combine the hidden state of the last word of the forward RNN and the hidden state of the first word of the backward RNN into a single vector

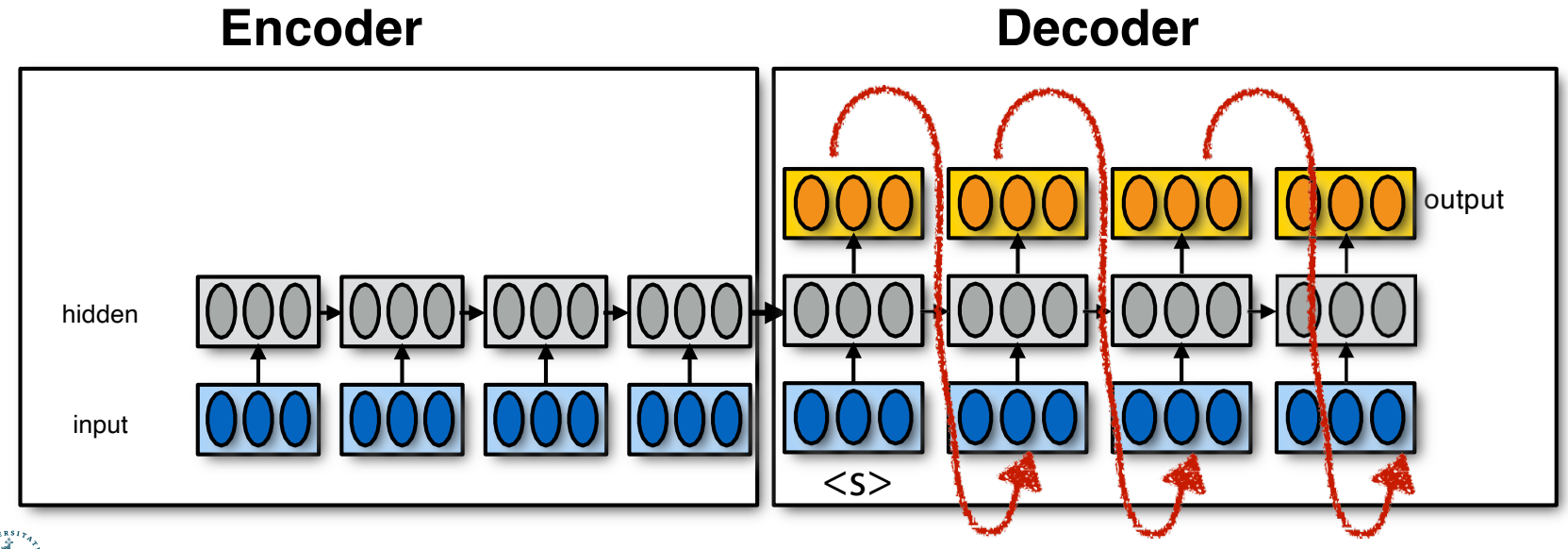


Encoder-Decoder (seq2seq) model

- Task: Read an input sequence and return an output sequence
  - Machine translation: translate source into target language
  - Dialog system/chatbot: generate a response

- Reading the input sequence: RNN Encoder

- Generating the output sequence: RNN Decoder



Encoder-Decoder (seq2seq) Model

Encoder RNN:

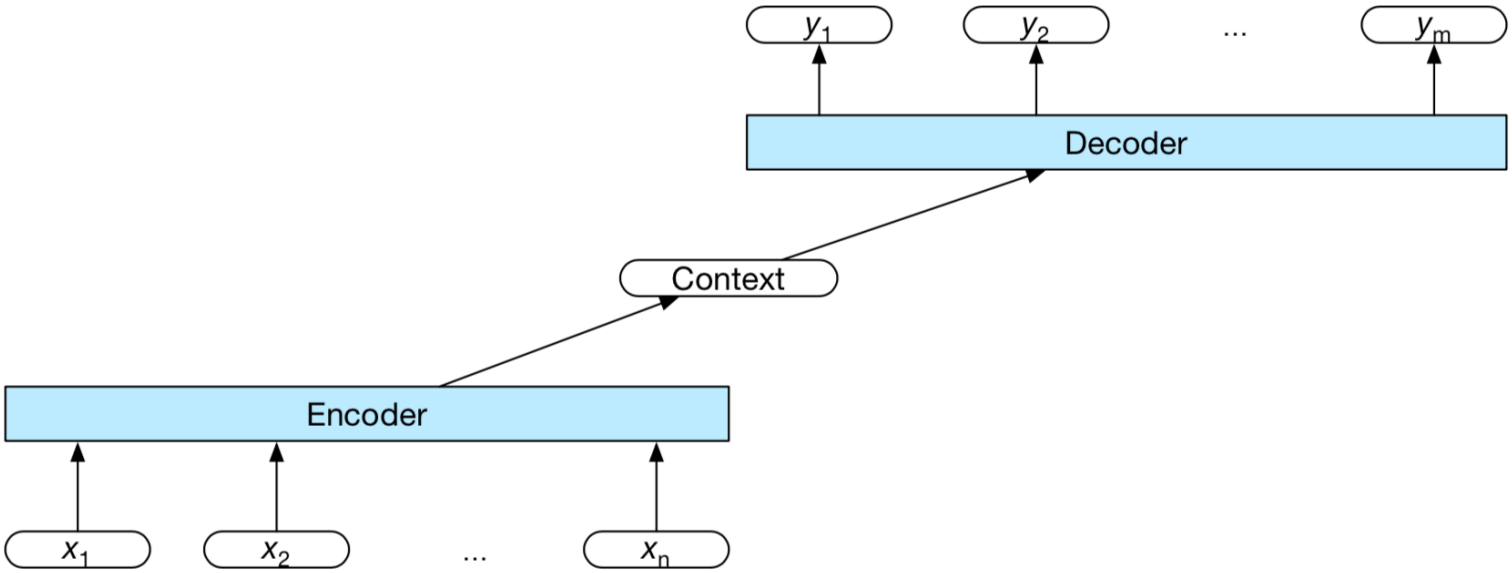
reads in the input sequence
passes its last hidden state to the initial hidden state of the decoder

Decoder RNN:

generates the output sequence
typically uses different parameters from the encoder
may also use different input embeddings

A More General View of seq2seq

In general, any function over the encoder's output can be used as a representation of the context we want to condition the decoder on.

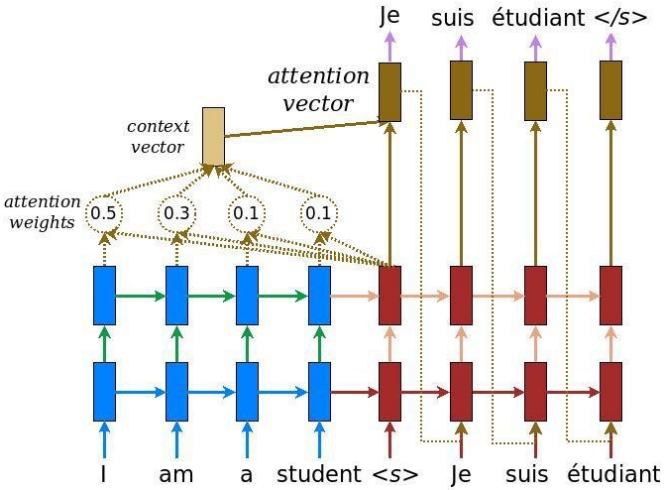


We can feed the context in at any time step during decoding (not just at the beginning).

Attention Mechanisms

Define a **distribution** $\alpha = (\alpha_{1t}, \ldots, \alpha_{St})$ **over the** ***S*** **elements of the input** sequence **that depends on the current output** element t (with $\sum_{s=1..S} \alpha_{st} = 1; \forall_{s \in 1 \ldots S}\, 0 \leq \alpha_{st} \leq 1$)

Use this distribution to compute a **weighted average of the input:** $\sum_{s=1..S} \alpha_{st} o_s$ and feed that into the decoder.



https://www.tensorflow.org/tutorials/text/nmt_with_attention

Remember? Dynamic Context Windows in word2vec

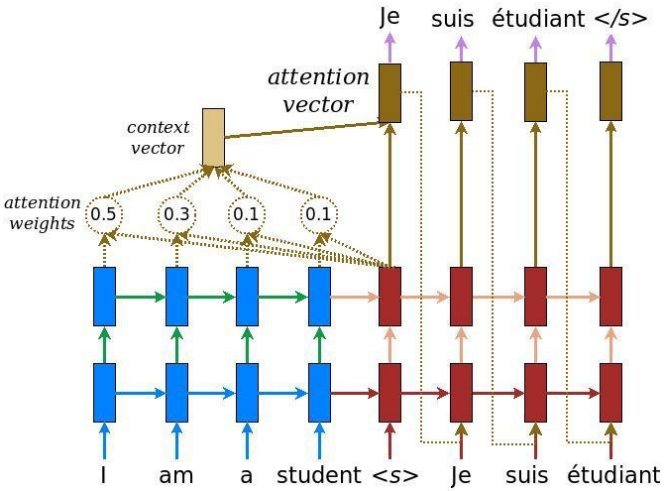
saw a furry little wampimuk hiding in the tree

Word2vec: $\frac{1}{4}$ $\frac{2}{4}$ $\frac{3}{4}$ $\frac{4}{4}$ $\frac{4}{4}$ $\frac{3}{4}$ $\frac{2}{4}$ $\frac{1}{4}$

GloVe: $\frac{1}{4}$ $\frac{1}{3}$ $\frac{1}{2}$ $\frac{1}{1}$ $\frac{1}{1}$ $\frac{1}{2}$ $\frac{1}{3}$ $\frac{1}{4}$

Aggressive: $\frac{1}{8}$ $\frac{1}{4}$ $\frac{1}{2}$ $\frac{1}{1}$ $\frac{1}{1}$ $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$

Attention Mechanisms



$\mathbf{h}_t$: current hidden state of decoder (target)
$\mathbf{h'}_s$: output of the encoder for word s (source)
Attention weights α_{ts}: distribution over $\mathbf{h'}_s$
α_{ts} depends on score($\mathbf{h}_t$, $\mathbf{h'}_s$)
Context vector $\mathbf{c}_t$: weighted average of $\mathbf{h'}_s$
Attention vector $\mathbf{a}_t$: computed by feedforward layer over $\mathbf{c}_t$ and $\mathbf{h}_t$

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

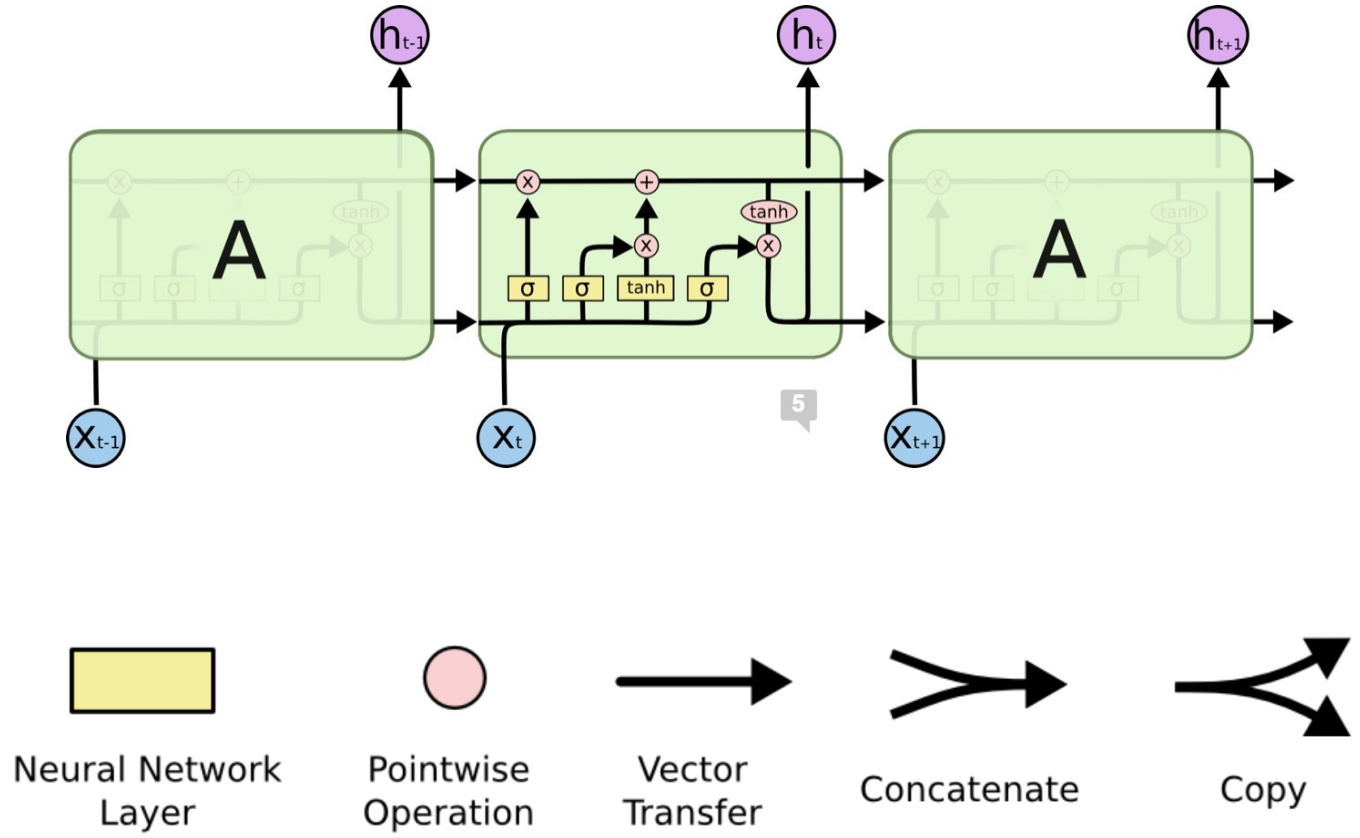
$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$

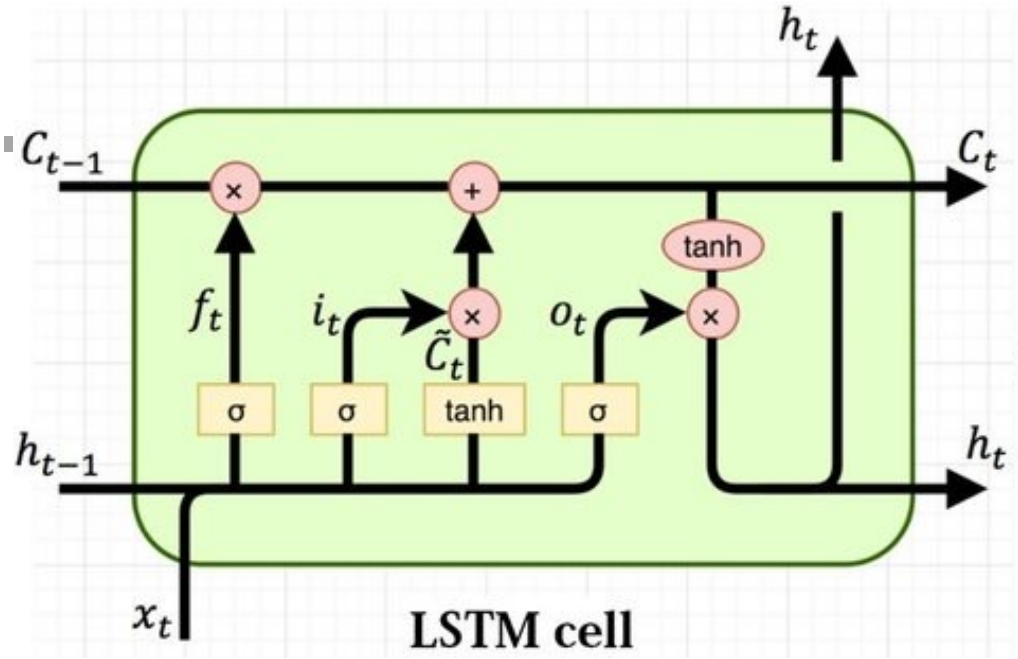
From RNNs to LSTMs

- In simple RNNs, hidden state depends on previous hidden state and on the input:
  - $\mathbf{h}_t = g(W_h[\mathbf{h}_{t-1}, \mathbf{x}_t] + b_h)$ with e.g. *g*=tanh
- Vanishing gradient problem
  - RNNs can't be trained effectively on long sequences
- LSTMs (Long Short-Term Memory networks) to the rescue
  - Additional cell state passed through the network and updated at each time step
  - LSTMs define four different layers (gates) that read in the previous hidden state and current input.

Long Short Term Memory Networks (LSTMs)



https://colah.github.io/posts/2015-08-Understanding-LSTMs/



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

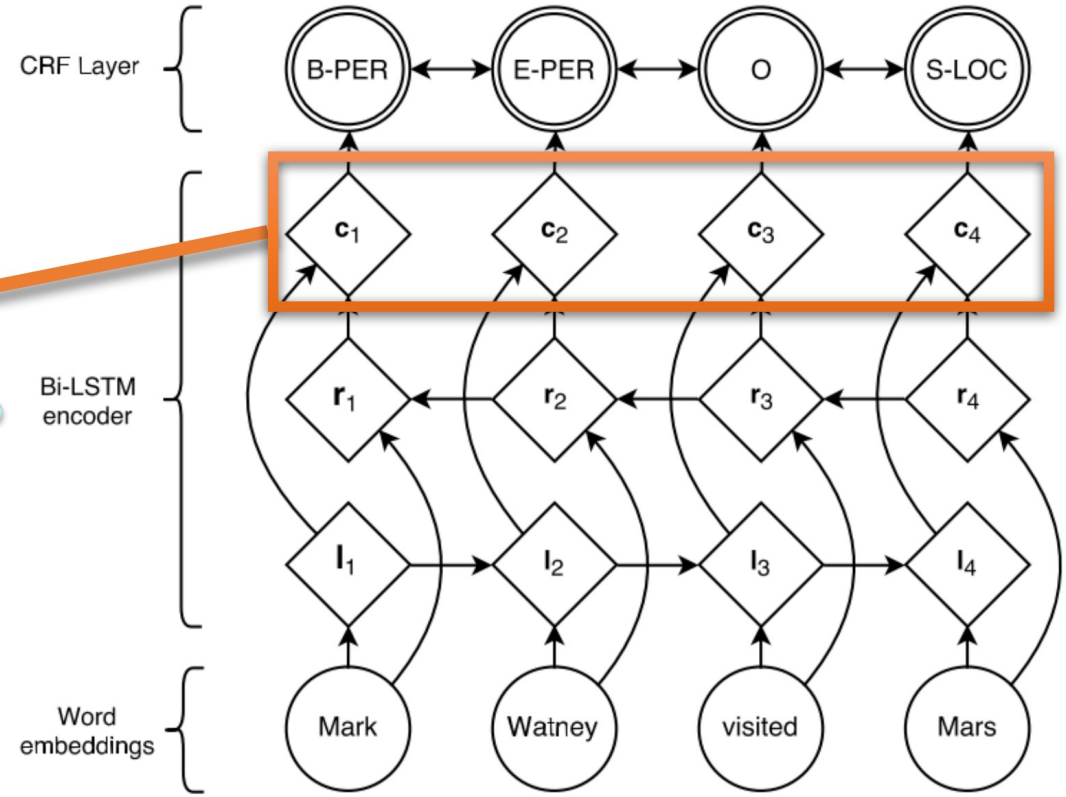
At time t, the LSTM cell reads in
— a *c*-dimensional previous cell state vector $\mathbf{c}_{t-1}$
— an *h*-dimensional previous hidden state vector $\mathbf{h}_{t-1}$
— a *d*-dimensional current input vector

At time t, the LSTM cell returns
— a *c*-dimensional previous cell state vector $\mathbf{c}_t$
— an *h*-dimensional previous hidden state vector $\mathbf{h}_t$
(which may also be passed to an output layer)

Bi-LSTM Encoder w/ HMM/CRF Layer

Contextualized Representations

High computational effort



Embeddings from Language Models

Replace static embeddings (lexicon lookup) with context-dependent embeddings (produced by a deep language model)

=> Each token's representation is a function of the entire input sentence, computed by a deep (multi-layer) bidirectional language model

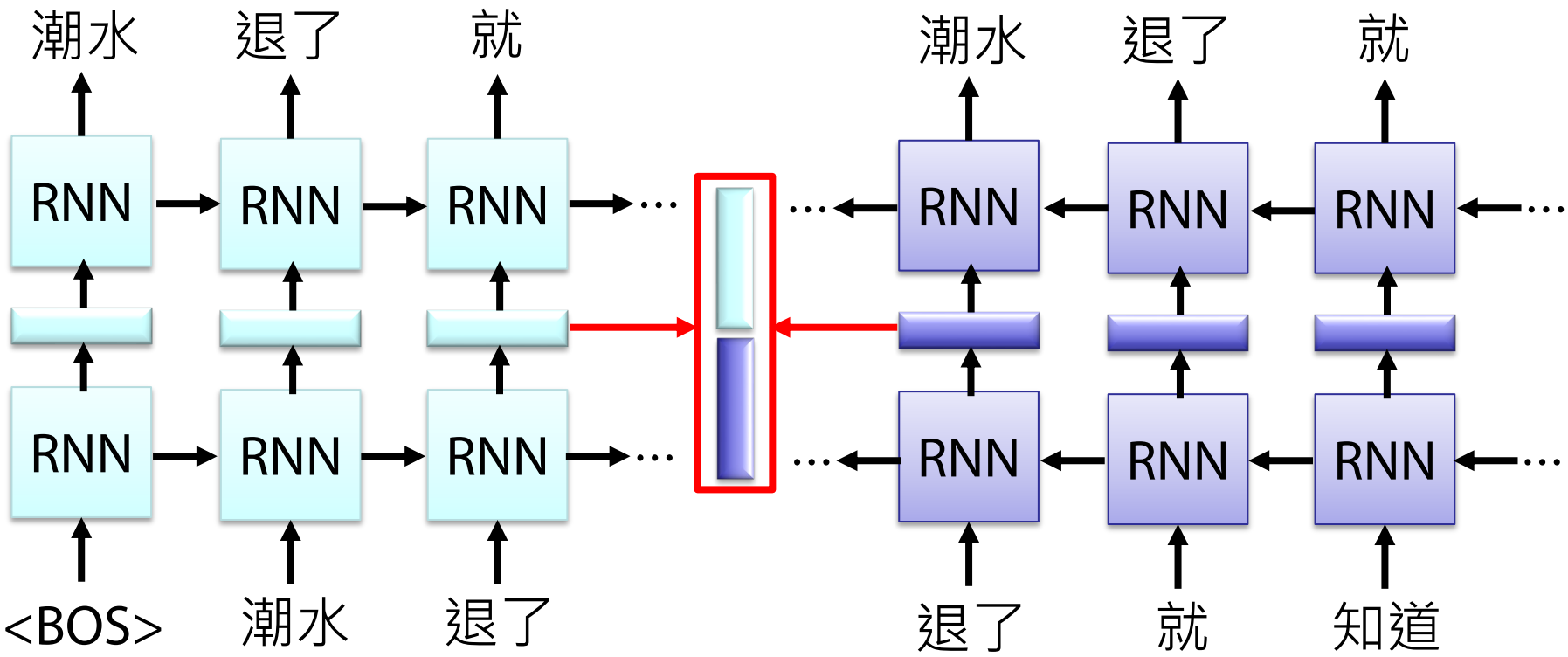
=> Return for each token a (task-dependent) linear combination of its representation across layers.

=> Different layers capture different information

Embeddings from Language Model (ELMO)

- RNN-based language models (trained from lots of sentences)

e.g., given “潮水 退了 就 知道 誰 沒穿 褲子”



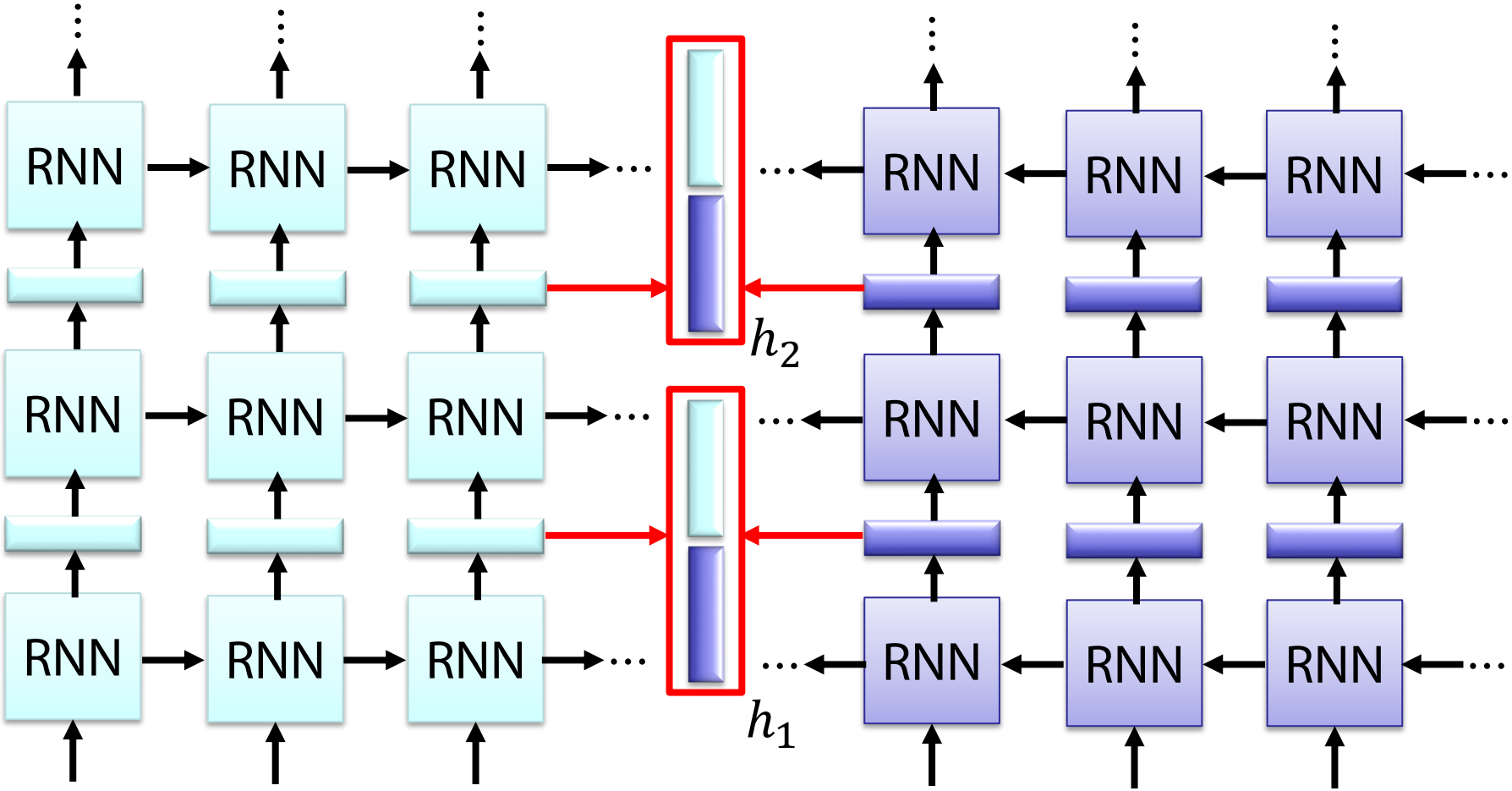
潮水 退了 就 知道 誰 沒穿 褲子 = When the tide goes out, you know who's not wearing pants.

https://arxiv.org/abs/1802.05365

ELMO

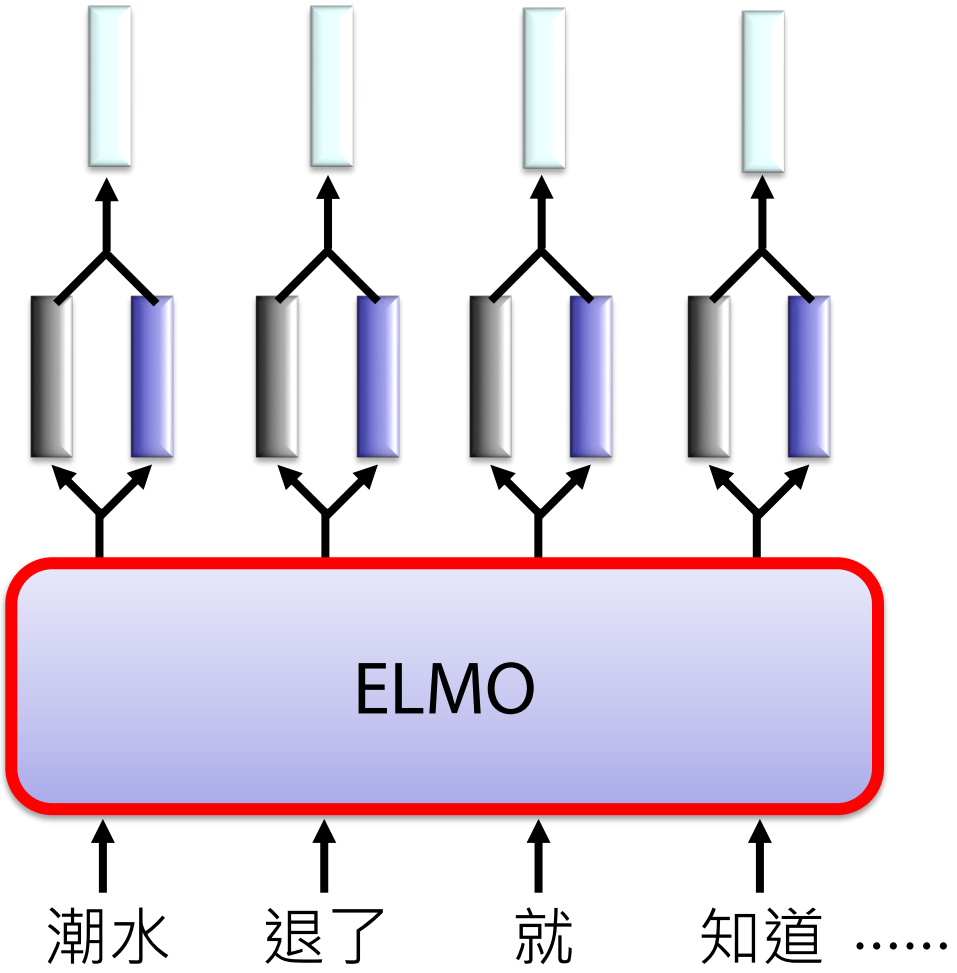
Each layer in deep LSTM can generate a latent representation.

Which one should we use???



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

ELMO



High computational effort, word2vec to the rescue?

$$= \alpha_1 + \alpha_2$$

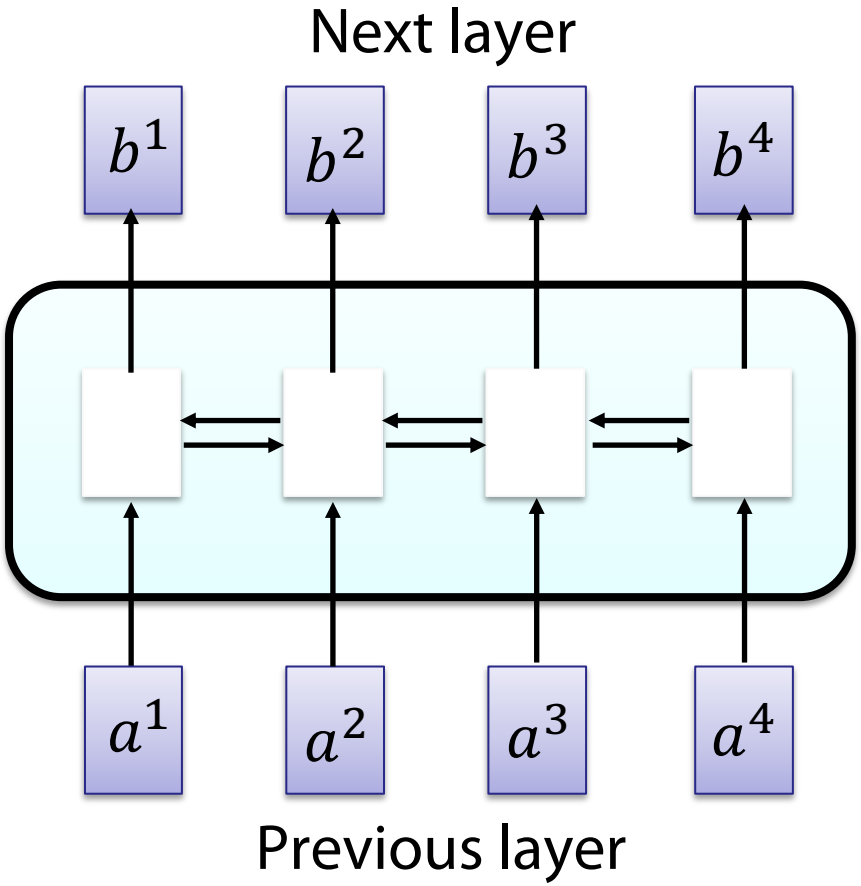
Learned with the down stream tasks

Transformers

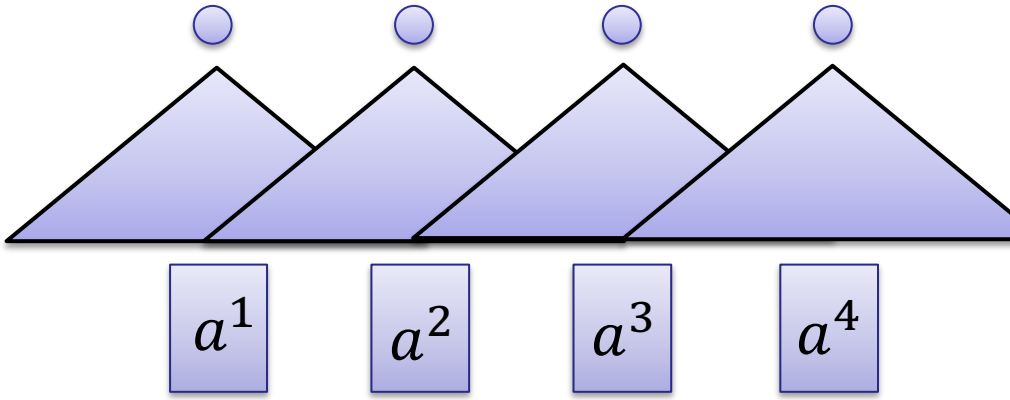
Sequence transduction model
(no convolutions or recurrence)

- Attention not only during decoding but also during encoding
  - Self-attention
- Captures more long-range dependencies than CNNs with fewer parameters
- Easier to parallelize than recurrent nets
- Faster to train than recurrent nets

Sequence

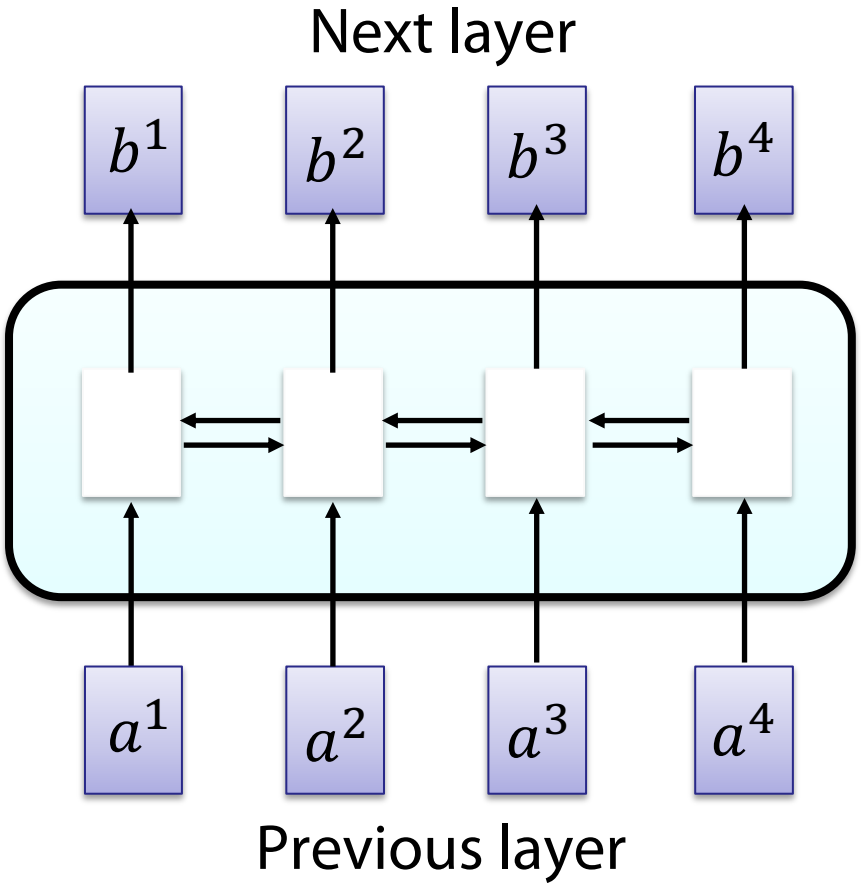


Hard to parallelize !



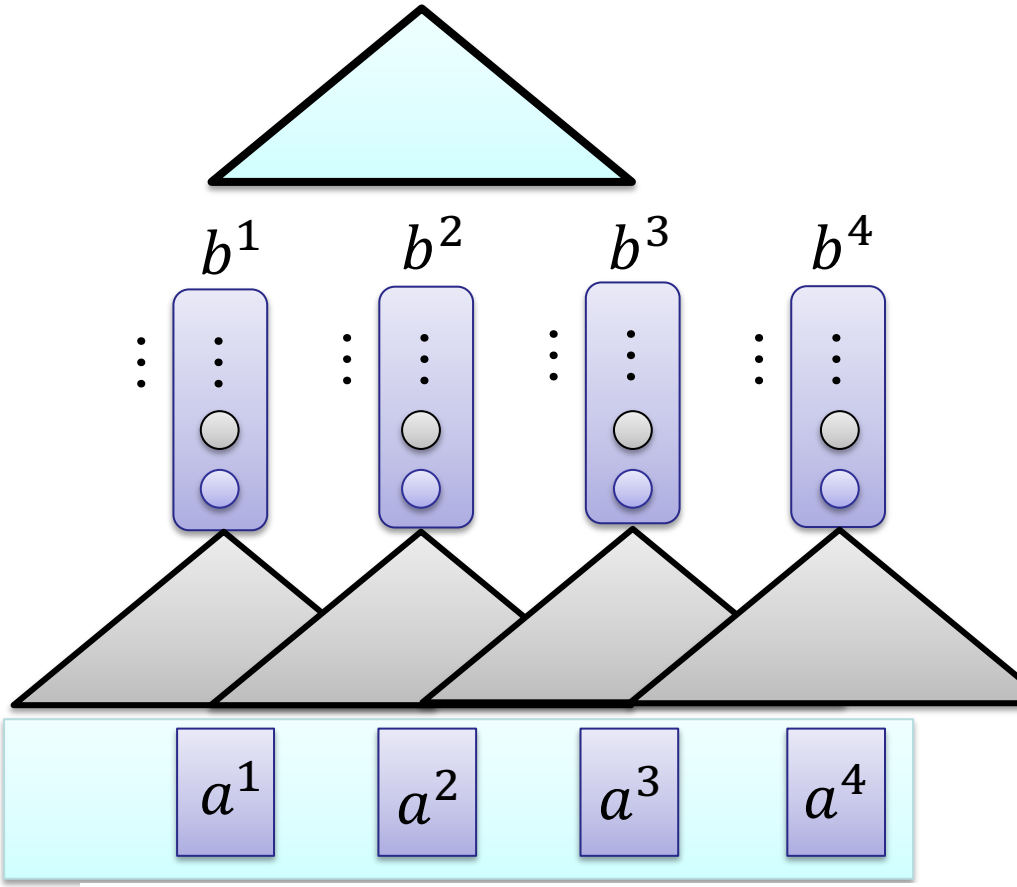
Using CNN to replace RNN

Sequence



Hard to parallelize !

Filters in higher layer can consider longer sequence

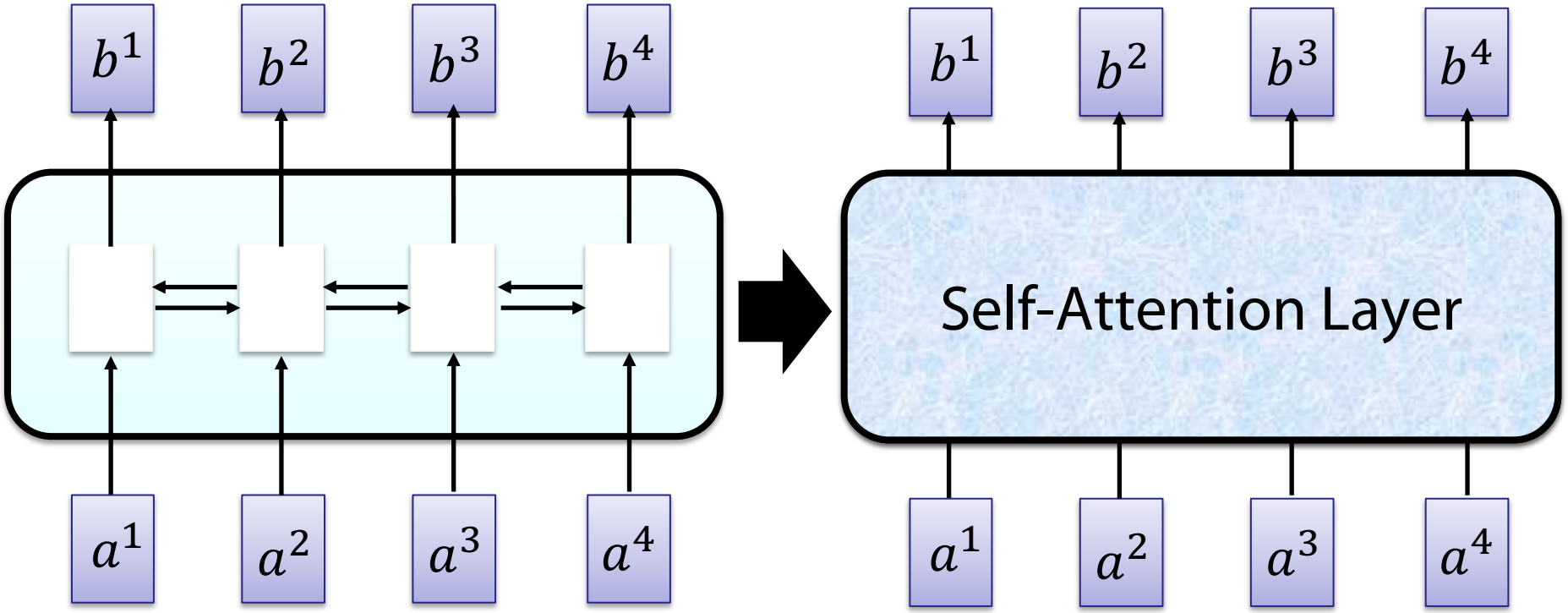


Using CNNs to replace RNN (CNNs can be parallelized)

Self-Attention

b^i is obtained based on the whole input sequence.

b^1, b^2, b^3, b^4 can be computed in a parallel way



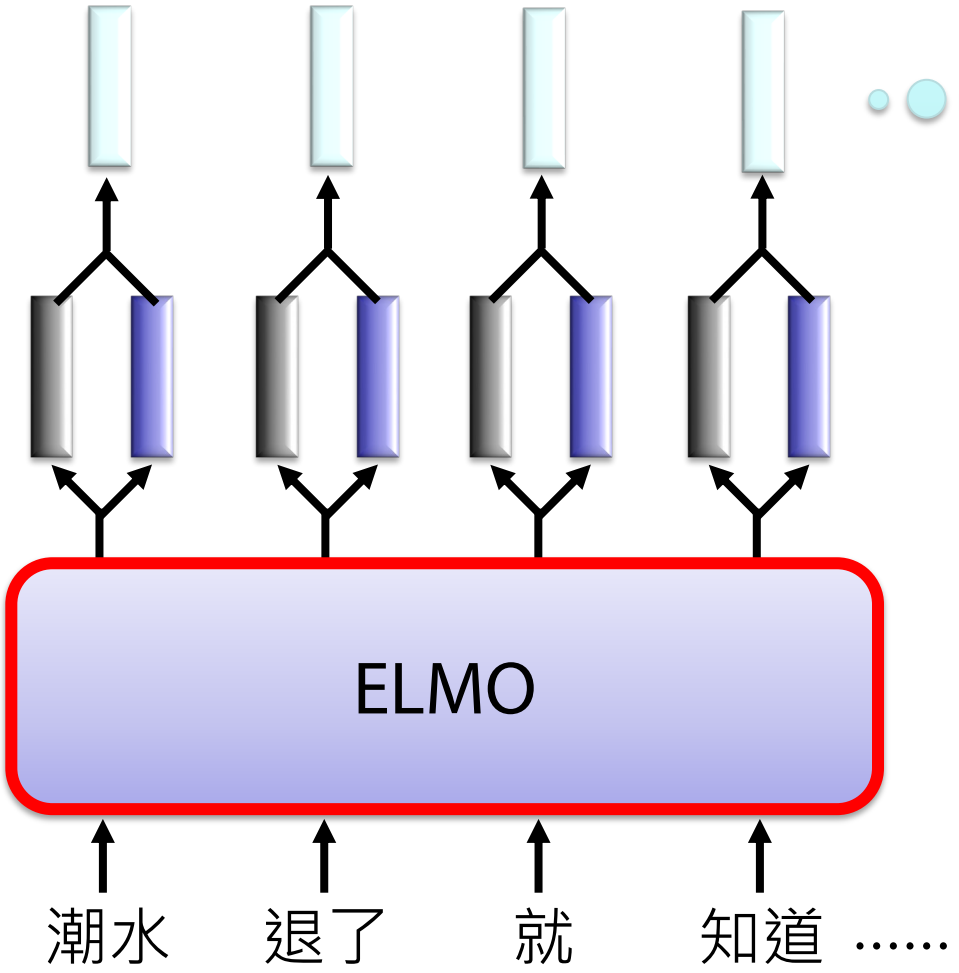
You can try to replace anything that has been done by RNN with self-attention.

Intelligent Agents

1d-CNNs LSTMs ELMo Transformers BERT GPT

Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Recap: Embeddings from Language MOdel (ELMO)



Sequence-based word embeddings (rather than bag-of-words based)

$$= \alpha_1 + \alpha_2$$

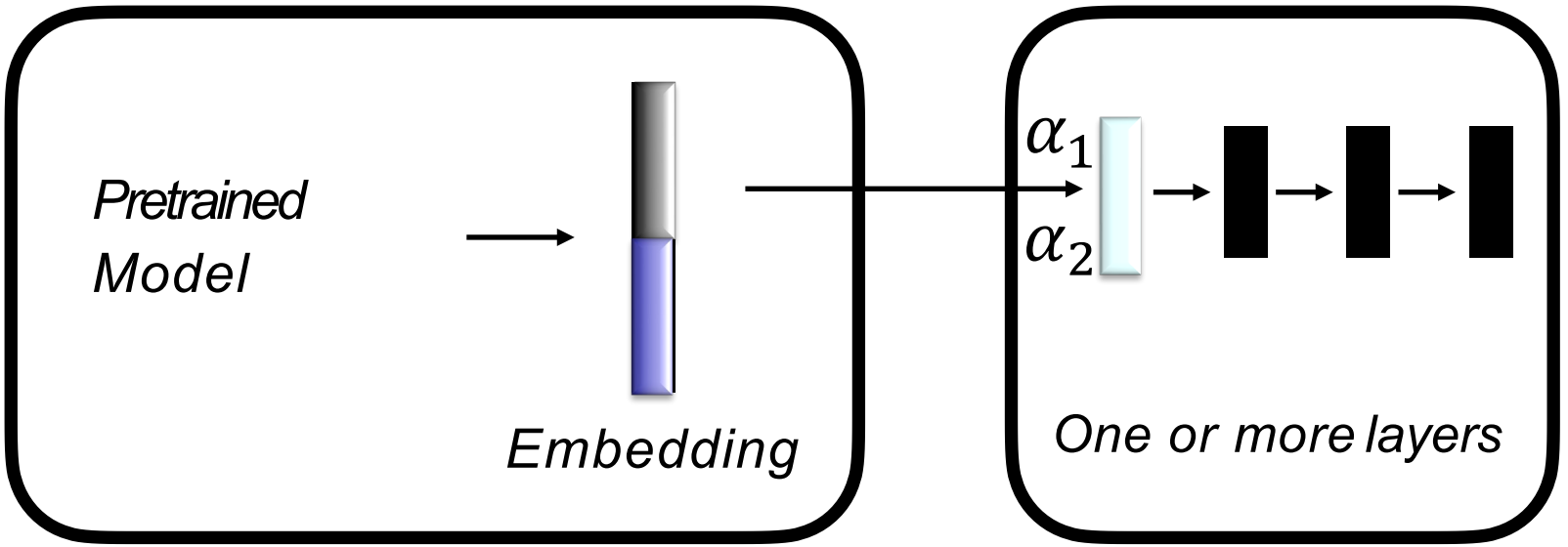
Learned with the downstream tasks

http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

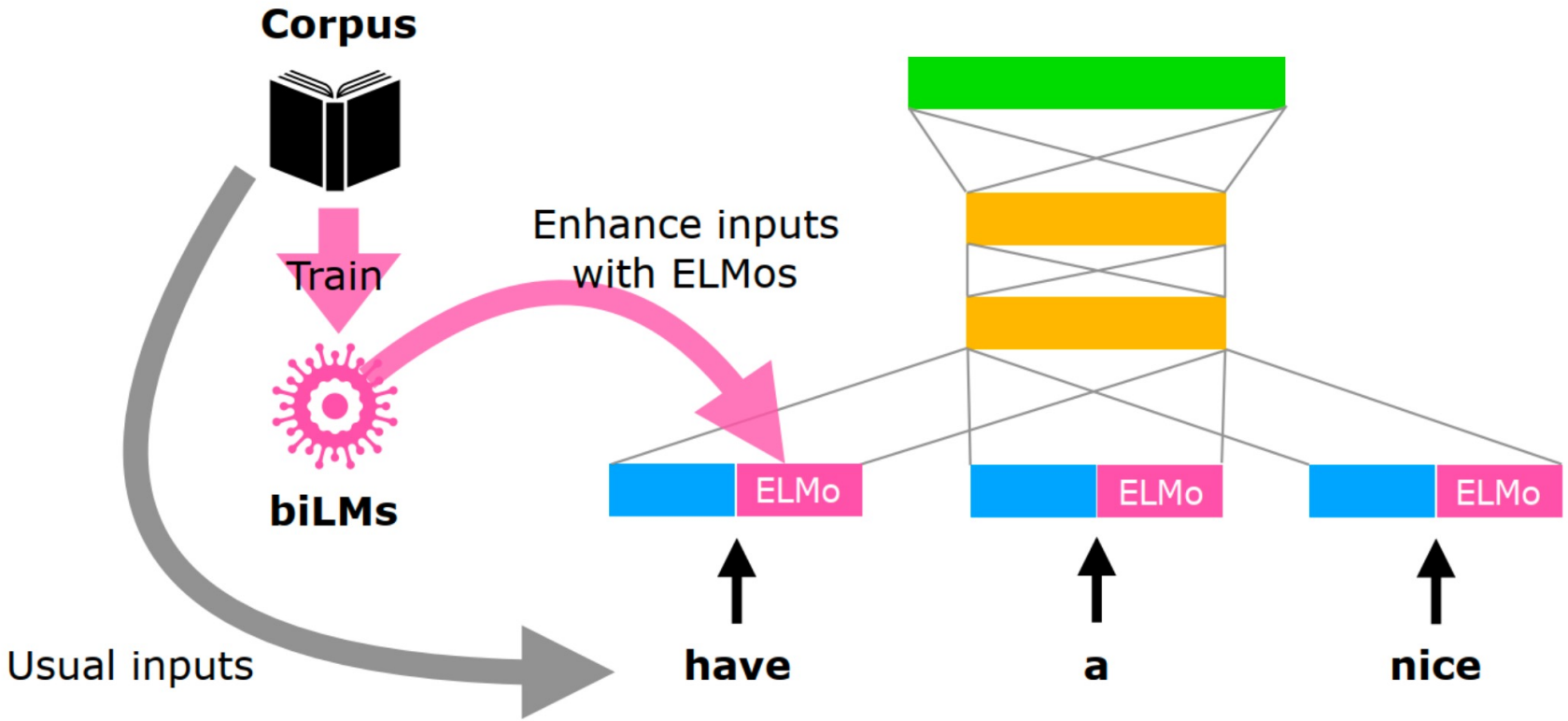
Pre-Training & Fine Tuning

ELMo
pre-trained on large corpus
(in self-supervised fashion)

2) Feature-based training ("fine-tuning")
on target/“downstream“ task
(supervised learning)

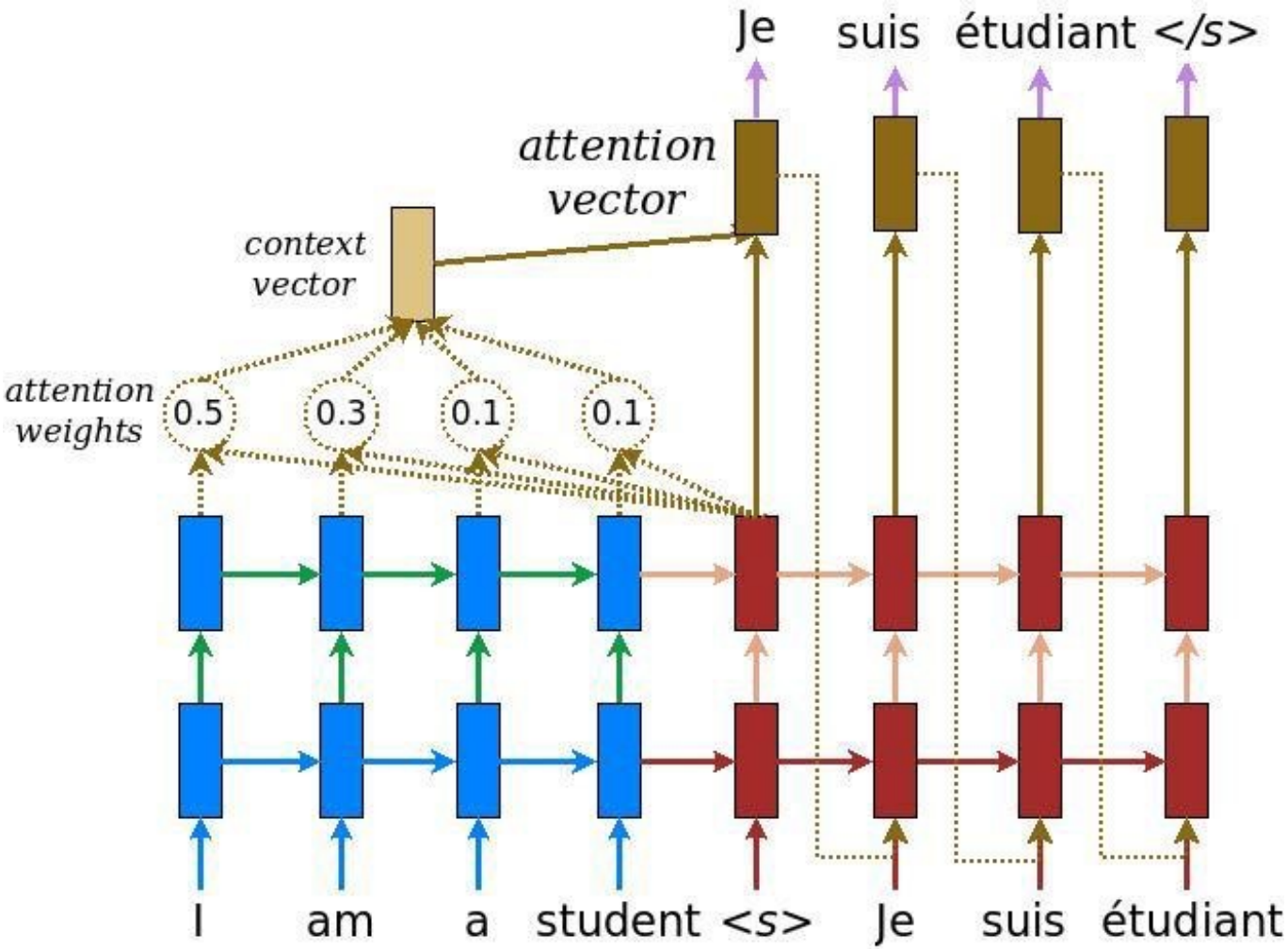


Integrate ELMos into other embeddings



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

Recap: Attention in Recurrent Encoding/Decoding



Recap: Transformers

Sequence transduction model
(no convolutions or recurrence)

- Attention not only during decoding but also during encoding
  - Self-attention
- Captures more long-range dependencies than recurrent architectures (and CNNs) with fewer parameters
- Easier to parallelize than recurrent nets
- Faster to train than recurrent nets

Self-attention

https://arxiv.org/abs/1706.03762

Attention is all you need.

q: query (to match others)

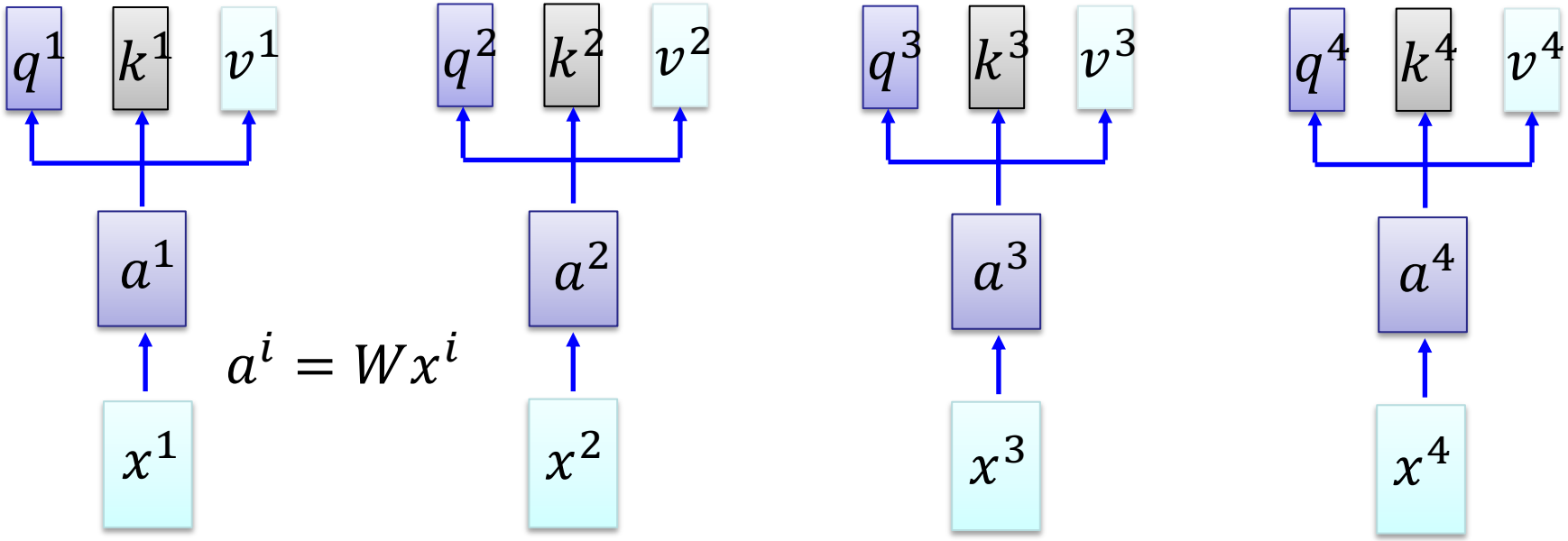
$$q^i = W^q a^i$$

k: key (to be matched)

$$k^i = W^k a^i$$

v: information to be extracted

$$v^i = W^v a^i$$

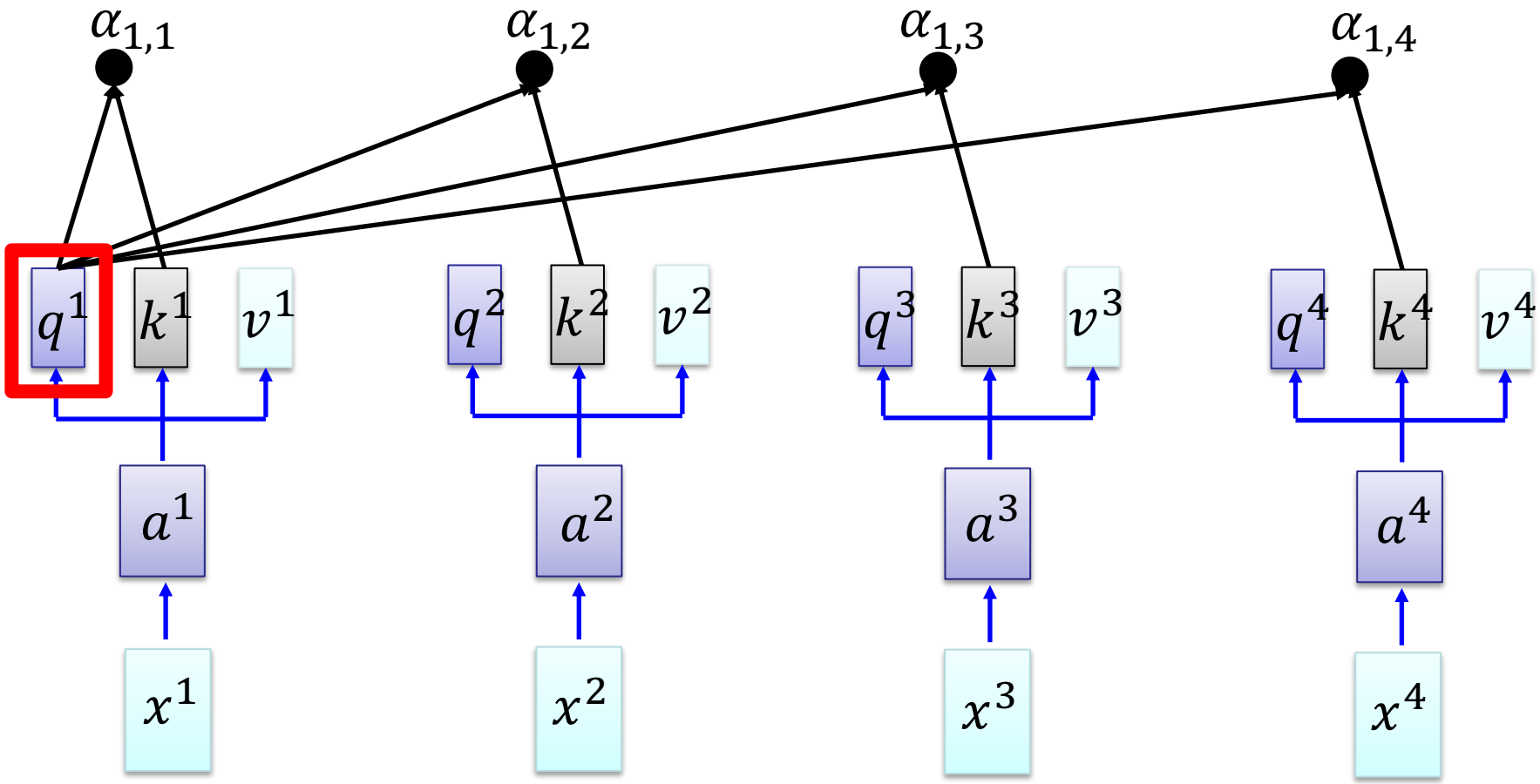


Self-attention

Take each query q, go to each key k, do attention

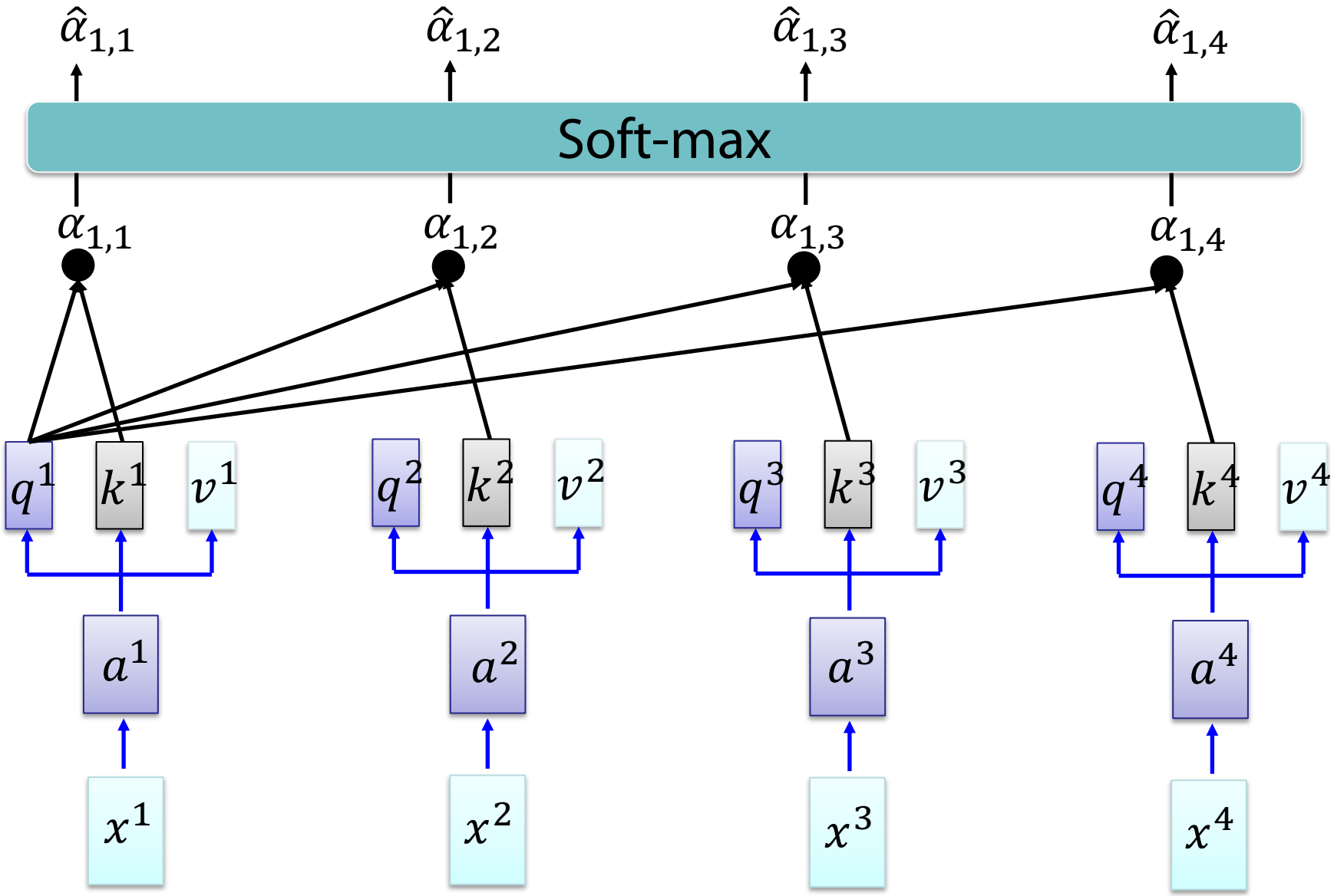
d is the dim of q and k

Scaled Dot-Product Attention: $\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{dot product}} / \sqrt{d}$



Self-attention

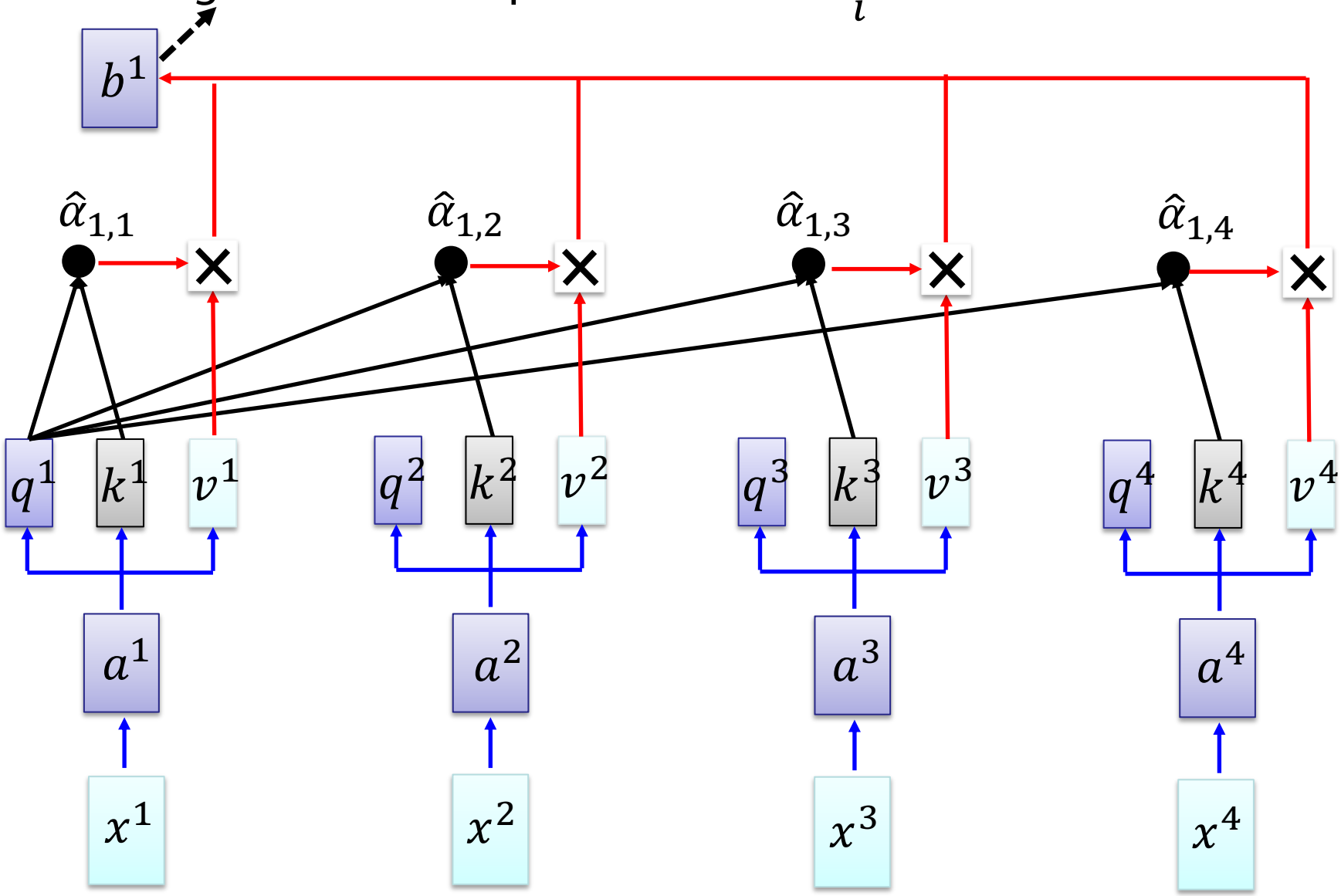
$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$



Self-attention

Considering the whole sequence

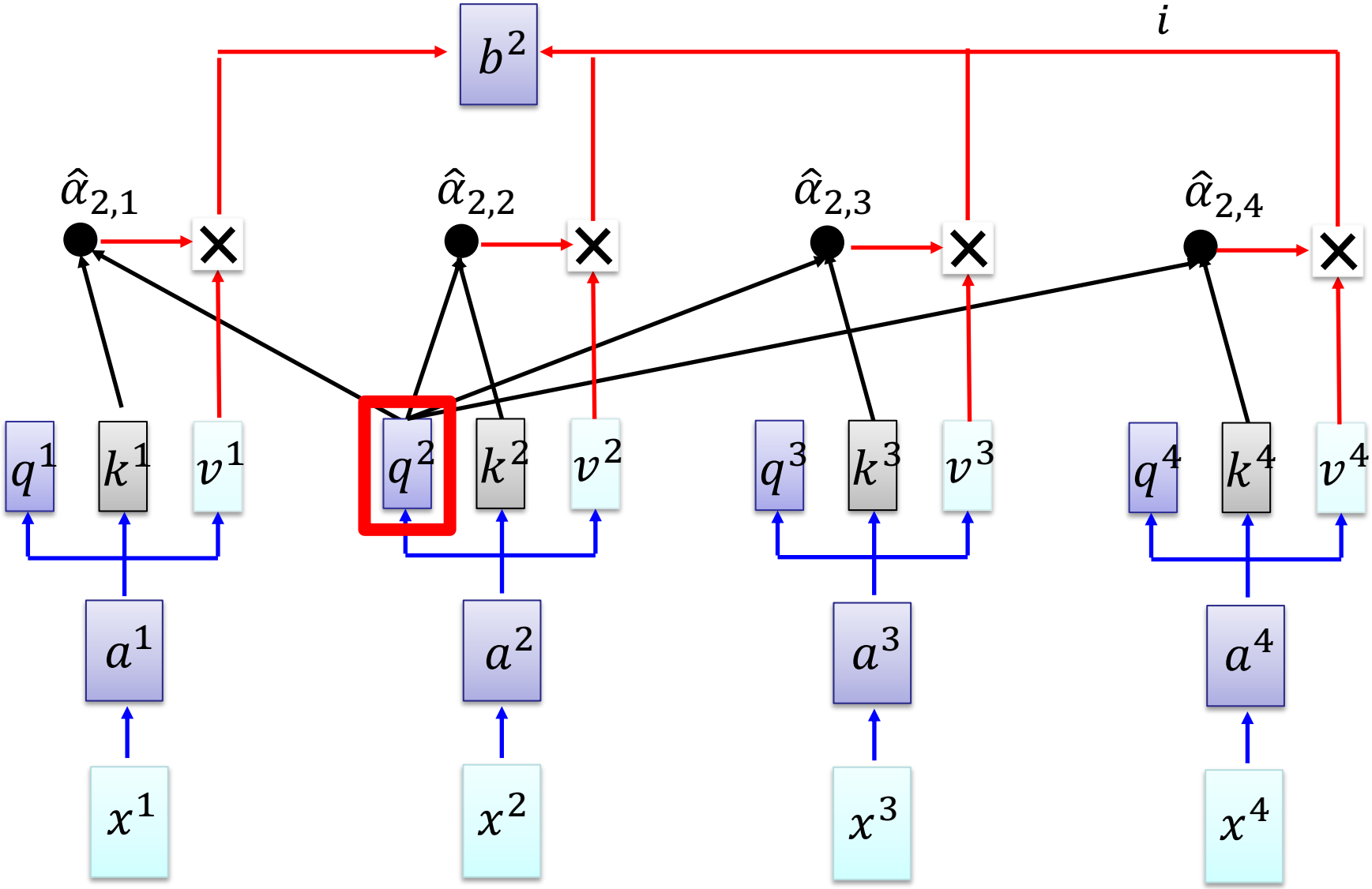
$$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$



Self-attention

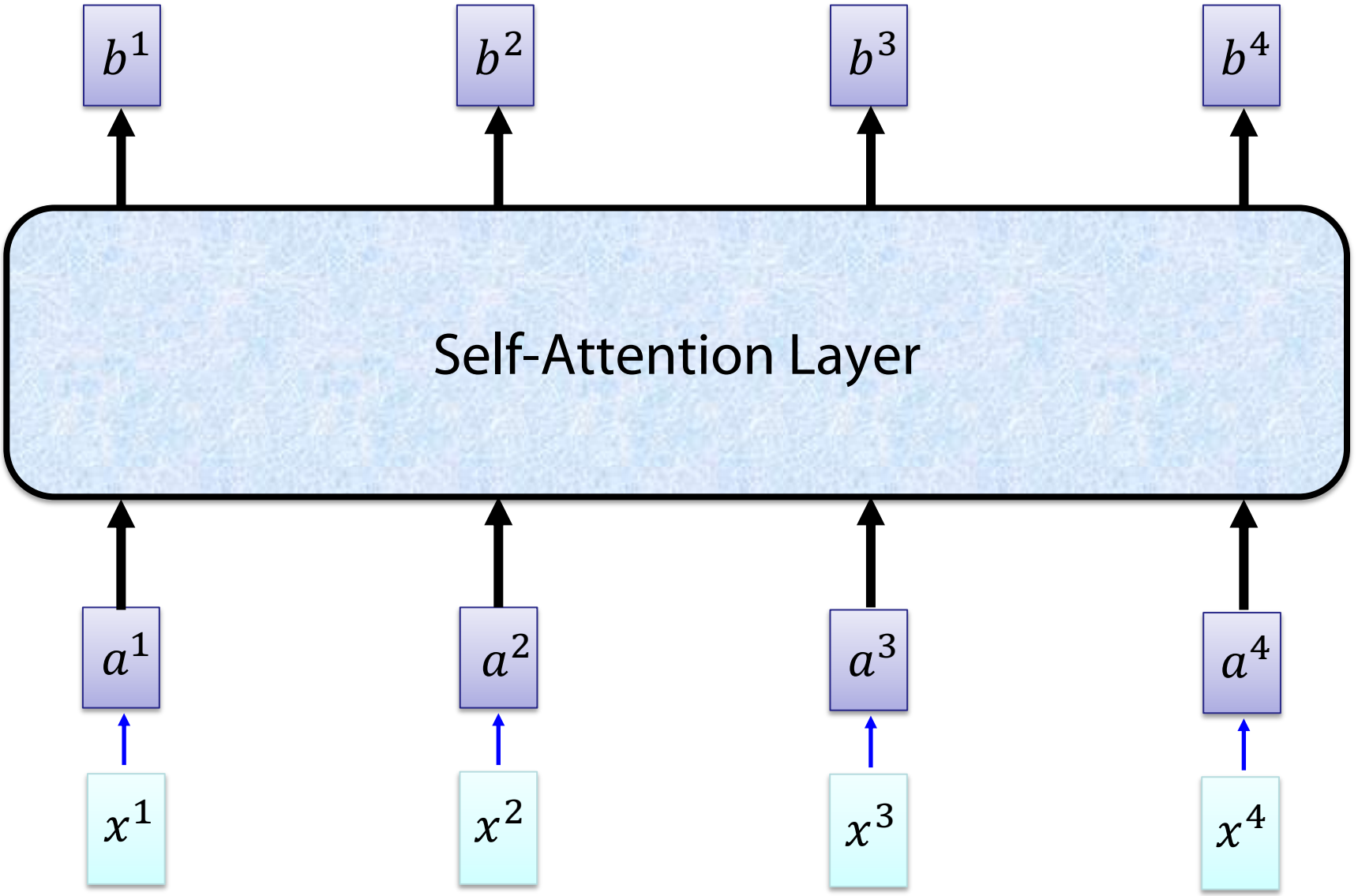
Take each query q, go to each key k, do attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



Self-attention

b^1, b^2, b^3, b^4 can be computed in parallel



Self-attention

$$q^i = W^q a^i$$

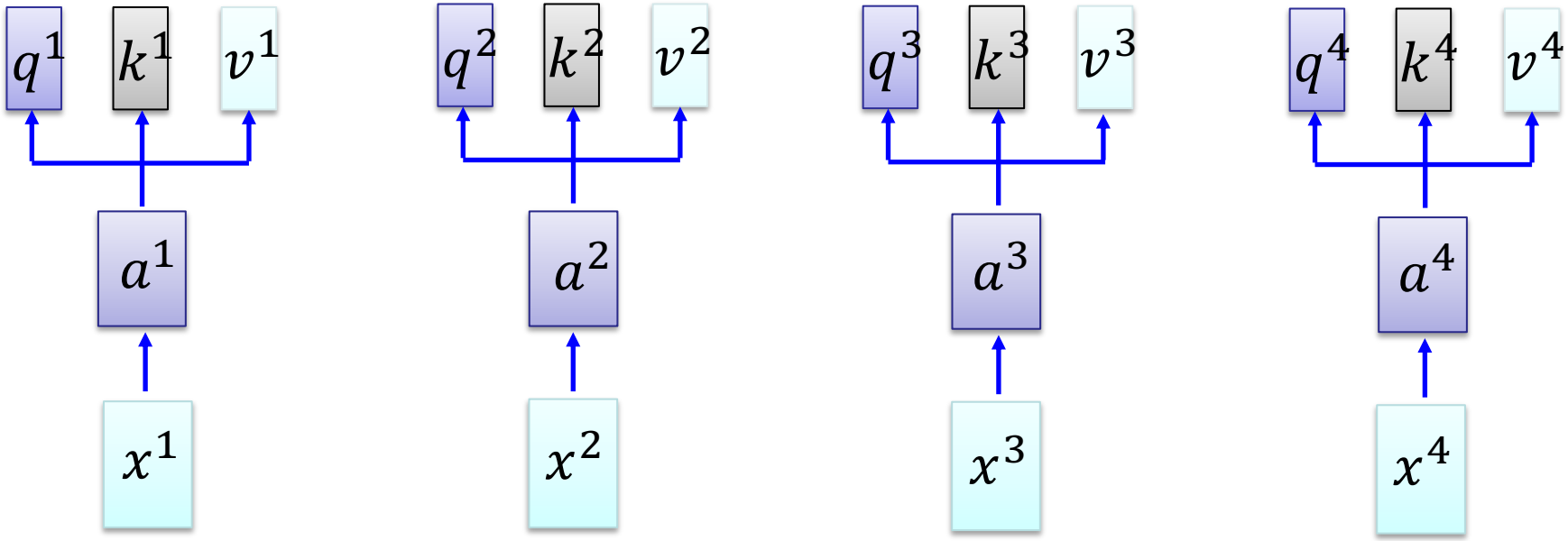
$$k^i = W^k a^i$$

$$v^i = W^v a^i$$

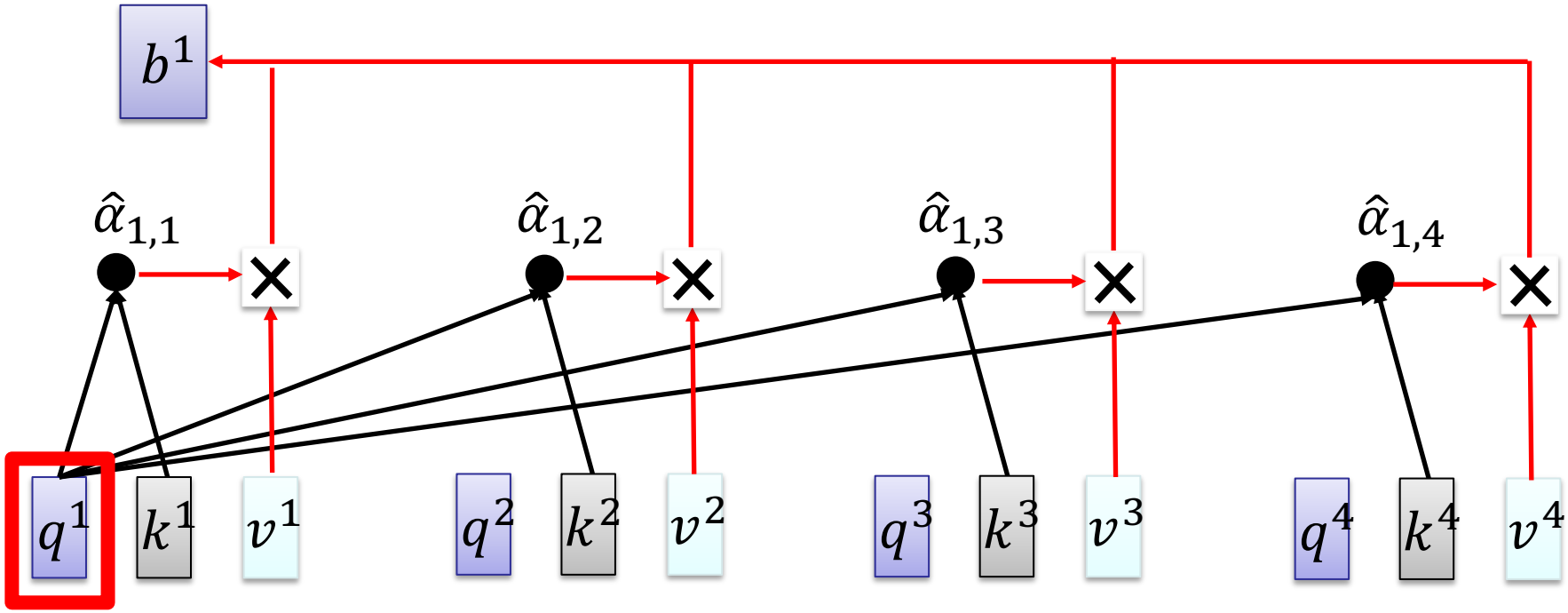
$$\underbrace{q^1\ q^2\ q^3\ q^4}_{Q} = W^q \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{k^1\ k^2\ k^3\ k^4}_{K} = W^k \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$

$$\underbrace{v^1\ v^2\ v^3\ v^4}_{V} = W^v \underbrace{a^1\ a^2\ a^3\ a^4}_{I}$$



Self-attention



$$\alpha_{1,1} = k^1 q^1 \qquad \alpha_{1,2} = k^2 q^1$$

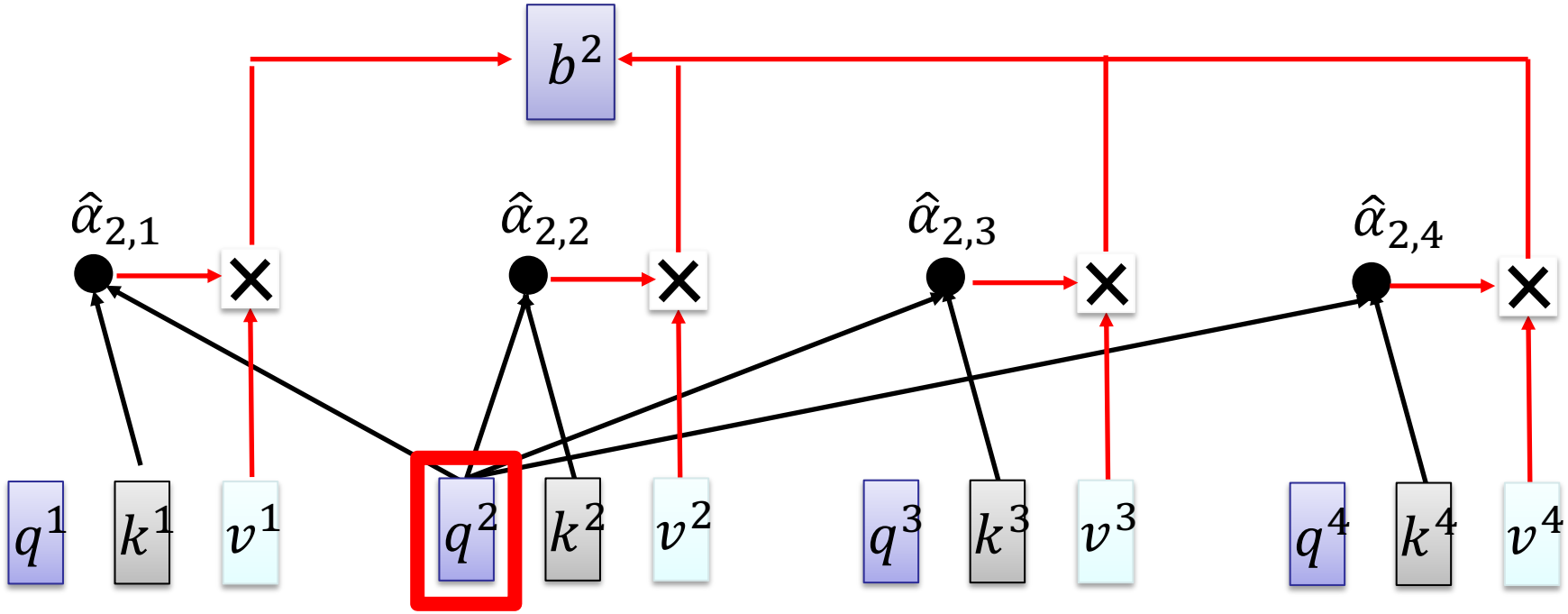
$$\alpha_{1,3} = k^3 q^1 \qquad \alpha_{1,4} = k^4 q^1$$

$$\begin{matrix}\alpha_{1,1}\\ \alpha_{1,2}\\ \alpha_{1,3}\\ \alpha_{1,4}\end{matrix} = \begin{matrix}k^1\\ k^2\\ k^3\\ k^4\end{matrix} q^1$$

(ignore $\sqrt{d}$ for simplicity)

Self-attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



$$\begin{bmatrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{bmatrix} \quad \hat{A}$$

softmax

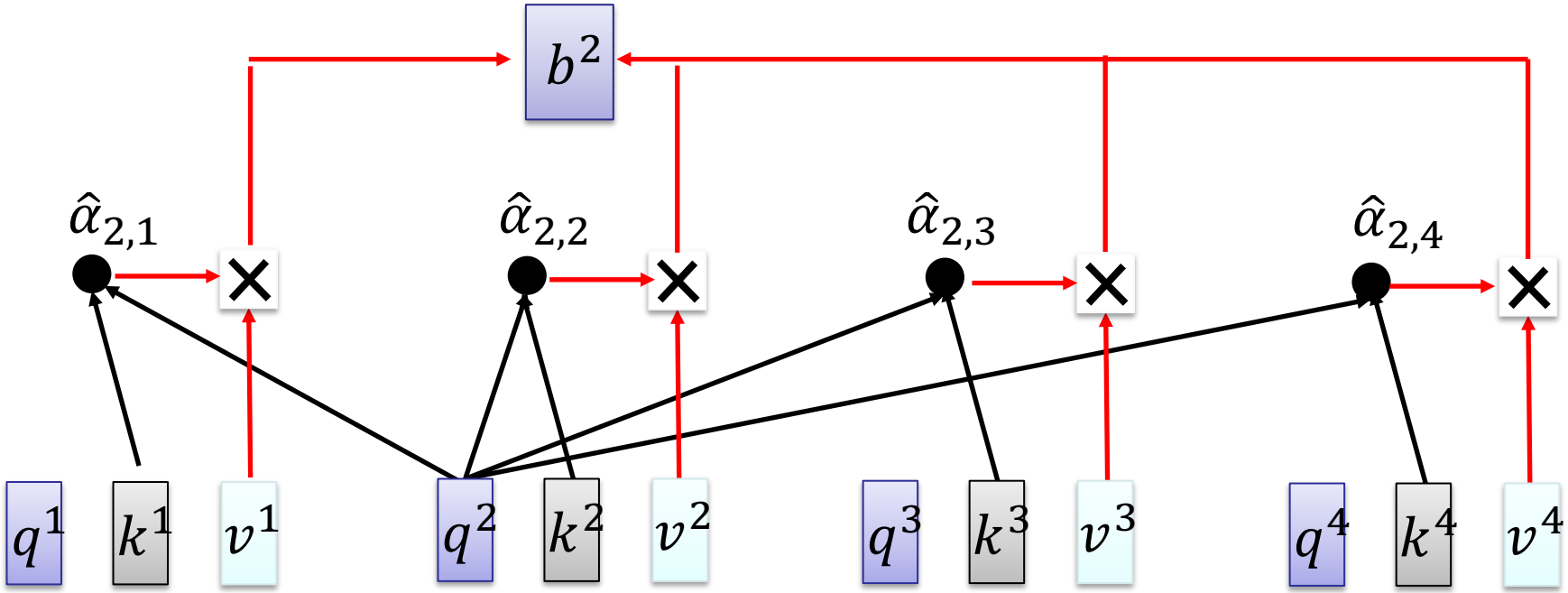
$$\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix} \quad A$$

$$= \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} \quad K^T$$

$$\begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} \quad Q$$

Self-attention

$$b^2 = \sum_i \hat{\alpha}_{2,i} v^i$$



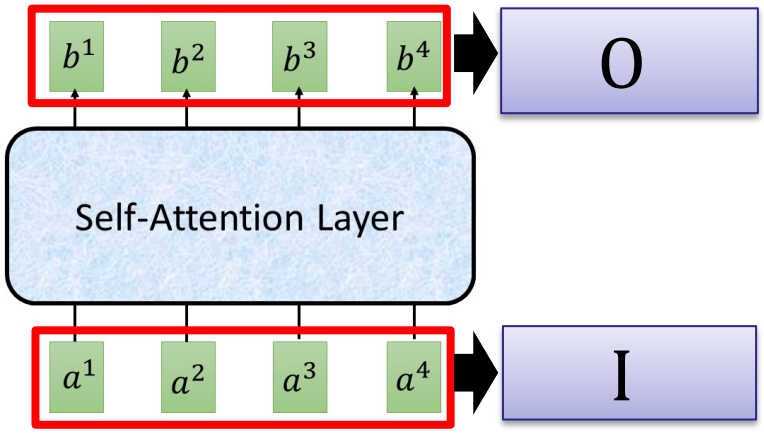
$$\begin{matrix} b^1 & b^2 & b^3 & b^4 \end{matrix} = \begin{matrix} v^1 & v^2 & v^3 & v^4 \end{matrix}$$

$$O \qquad V$$

$$\begin{matrix} \hat{\alpha}_{1,1} & \hat{\alpha}_{2,1} & \hat{\alpha}_{3,1} & \hat{\alpha}_{4,1} \\ \hat{\alpha}_{1,2} & \hat{\alpha}_{2,2} & \hat{\alpha}_{3,2} & \hat{\alpha}_{4,2} \\ \hat{\alpha}_{1,3} & \hat{\alpha}_{2,3} & \hat{\alpha}_{3,3} & \hat{\alpha}_{4,3} \\ \hat{\alpha}_{1,4} & \hat{\alpha}_{2,4} & \hat{\alpha}_{3,4} & \hat{\alpha}_{4,4} \end{matrix}$$

$$\hat{A}$$

Self-attention



$$O = Q = W^q I$$

$$K = W^k I$$

$$V = W^v I$$

$$A = K^T Q$$

$$\hat{A} = \text{softmax}(A)$$

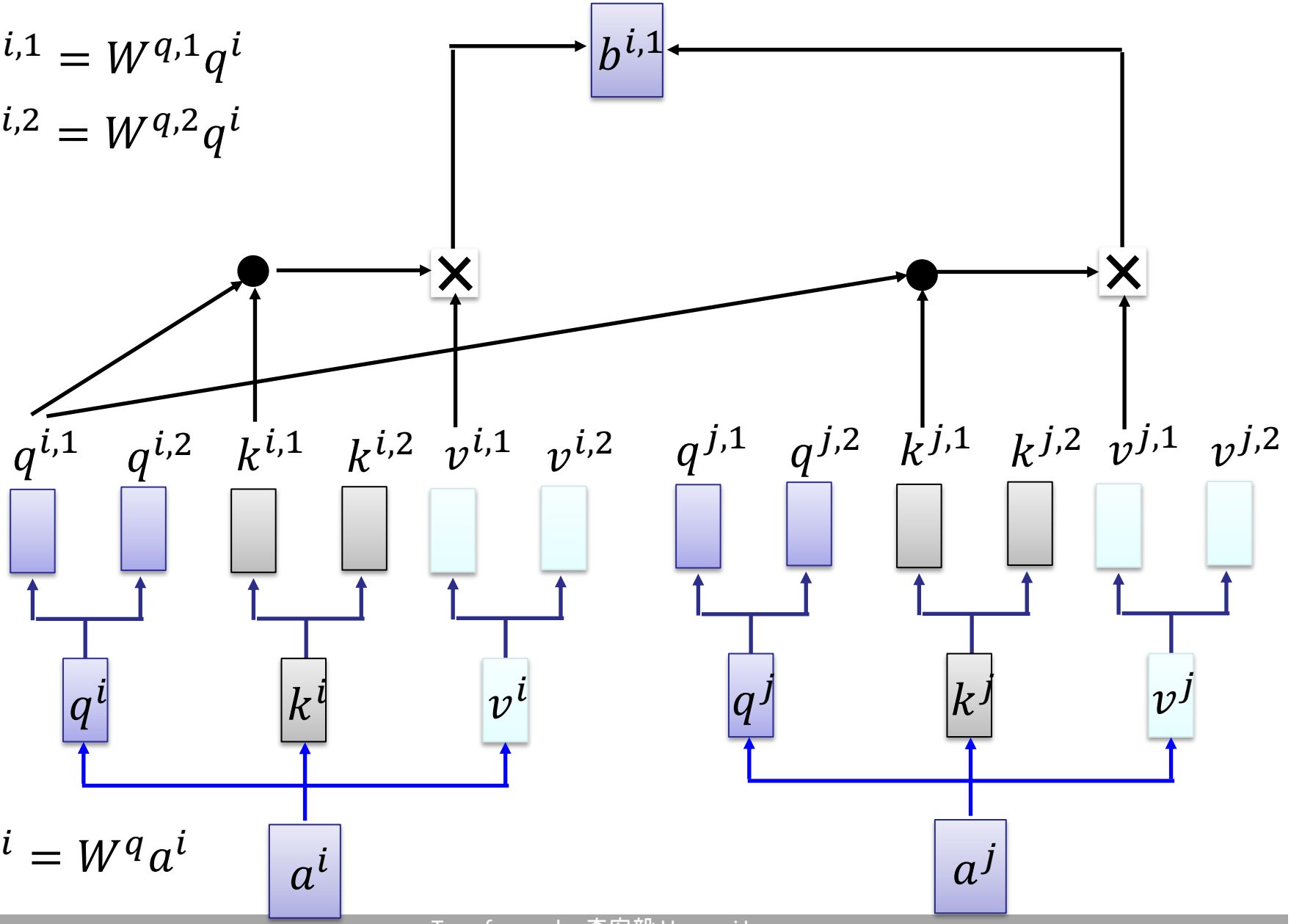
$$O = V \hat{A}$$

Can be optimized with GPUs

Multi-head Self-attention (2 heads as example)

$q^{i,1} = W^{q,1} q^i$

$q^{i,2} = W^{q,2} q^i$

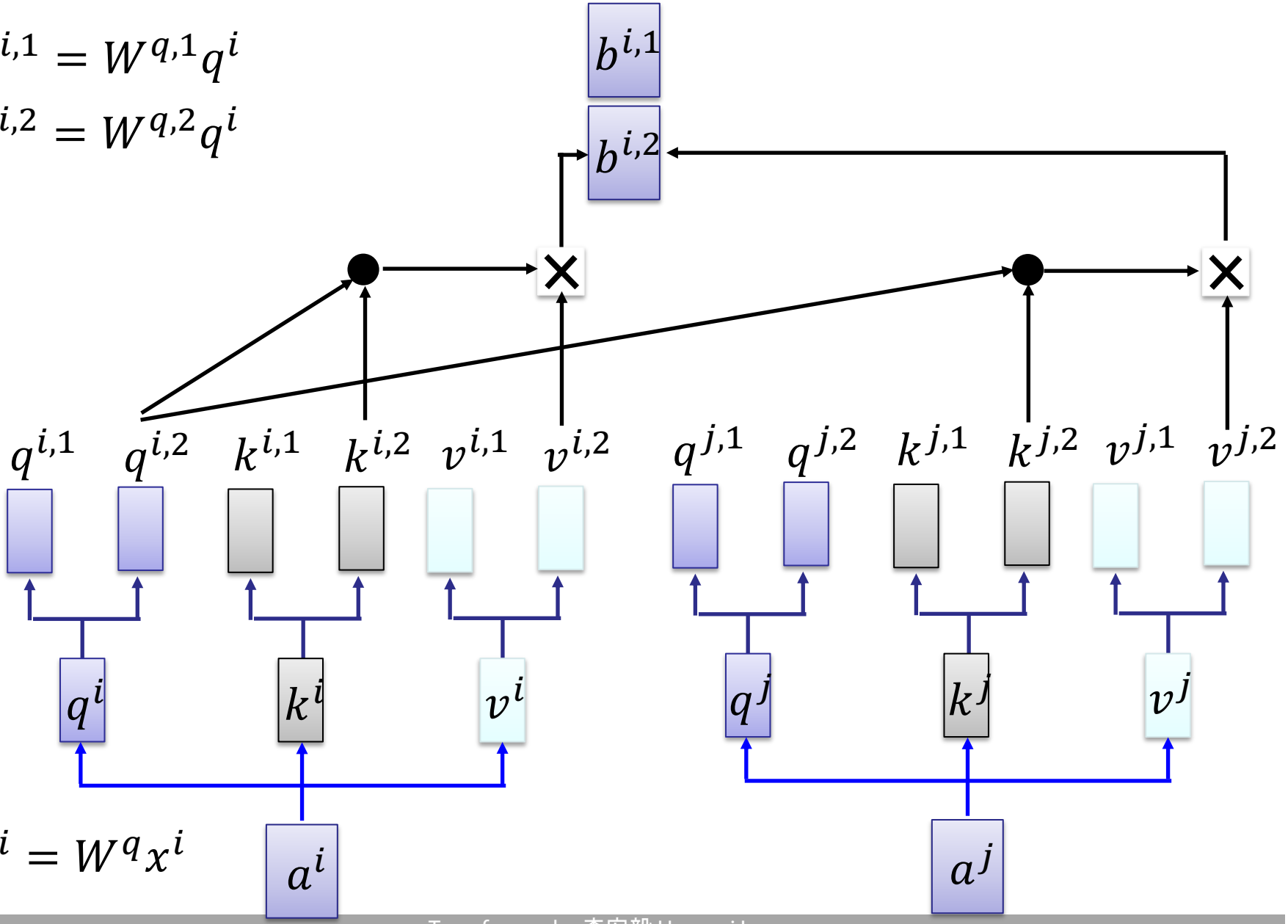


$q^i = W^q a^i$

Multi-head Self-attention (2 heads as example)

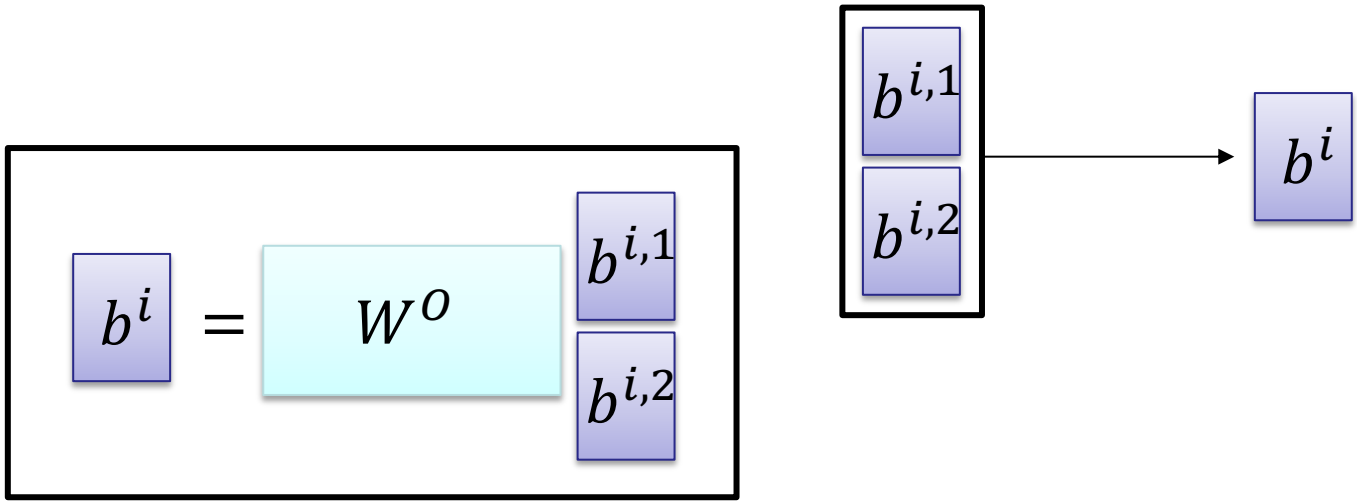
$q^{i,1} = W^{q,1} q^i$

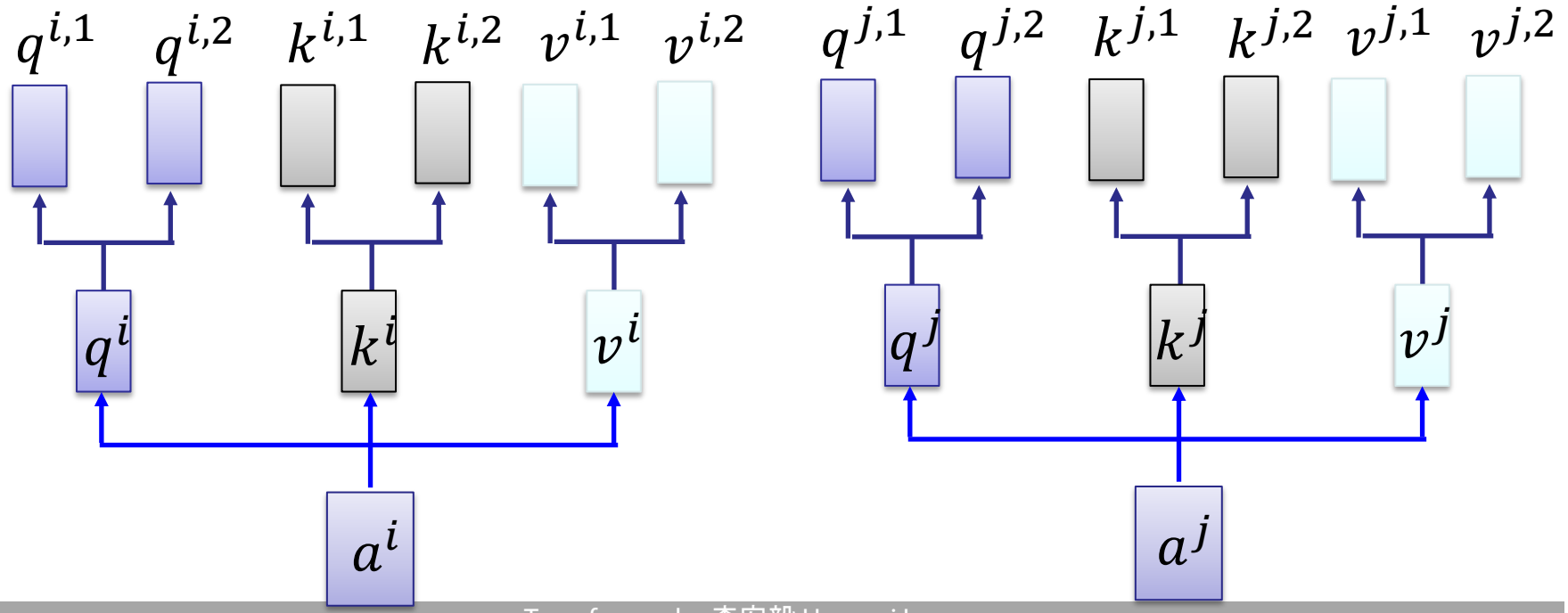
$q^{i,2} = W^{q,2} q^i$



$q^i = W^q x^i$

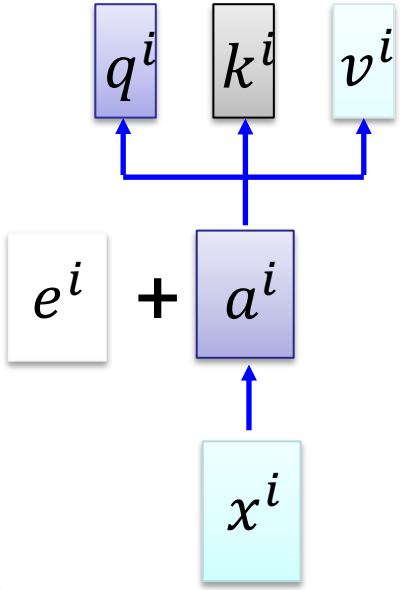
Multi-head Self-attention (2 heads as example)



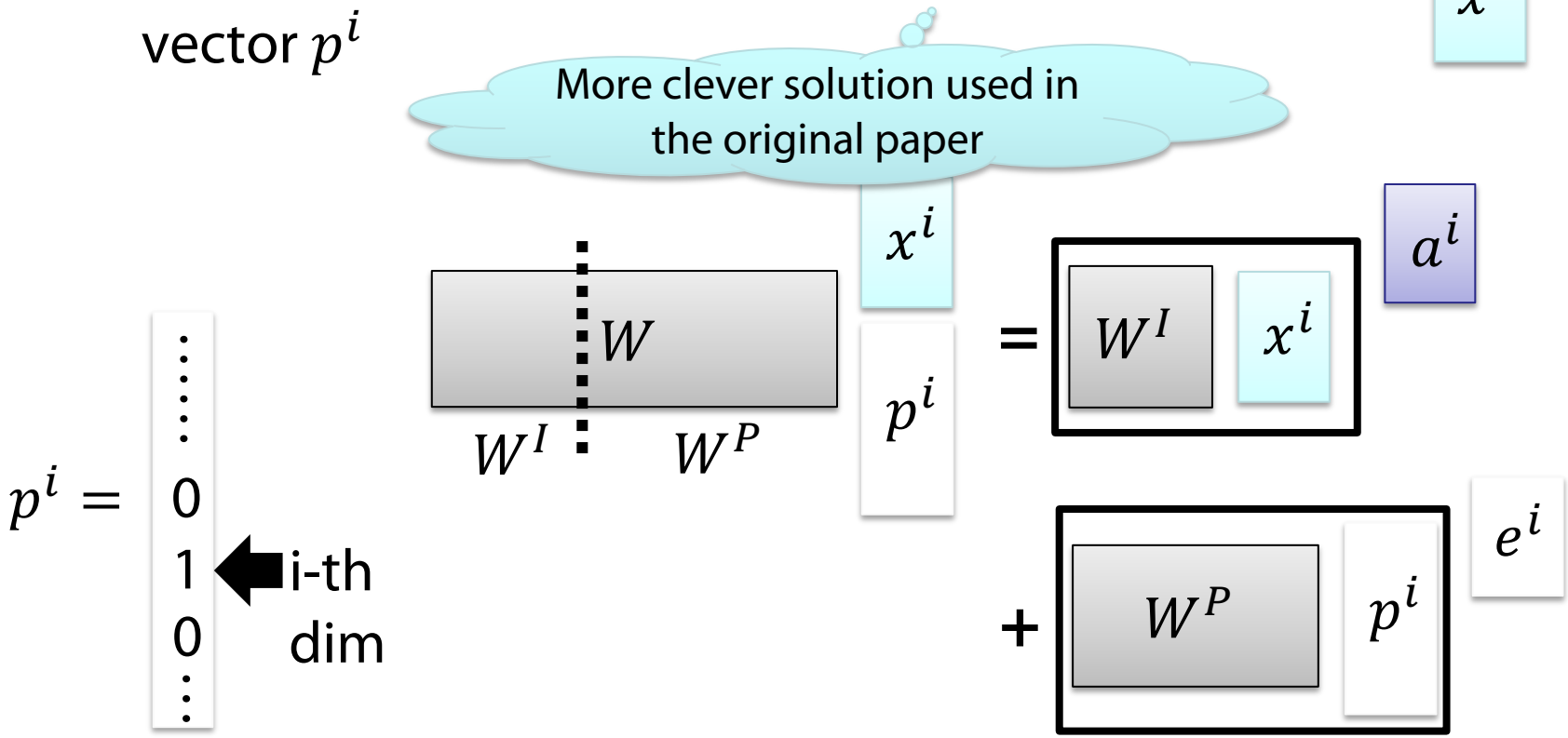


Positional Encoding

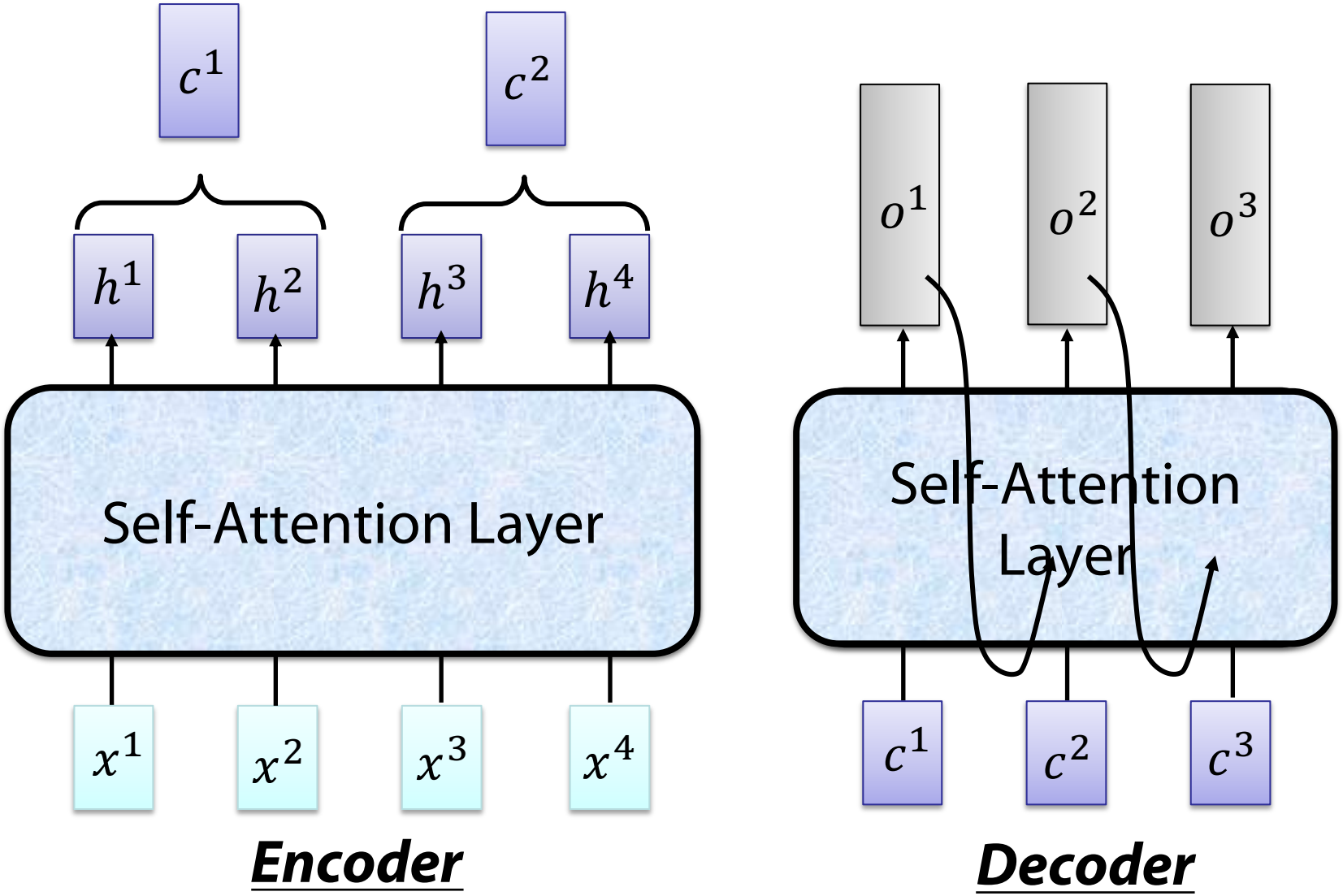
- No position information in self-attention.
- Each position has a unique positional vector e^i (not learned from data)
- Idea: Append each x^i is with a one-hot vector p^i



More clever solution used in the original paper

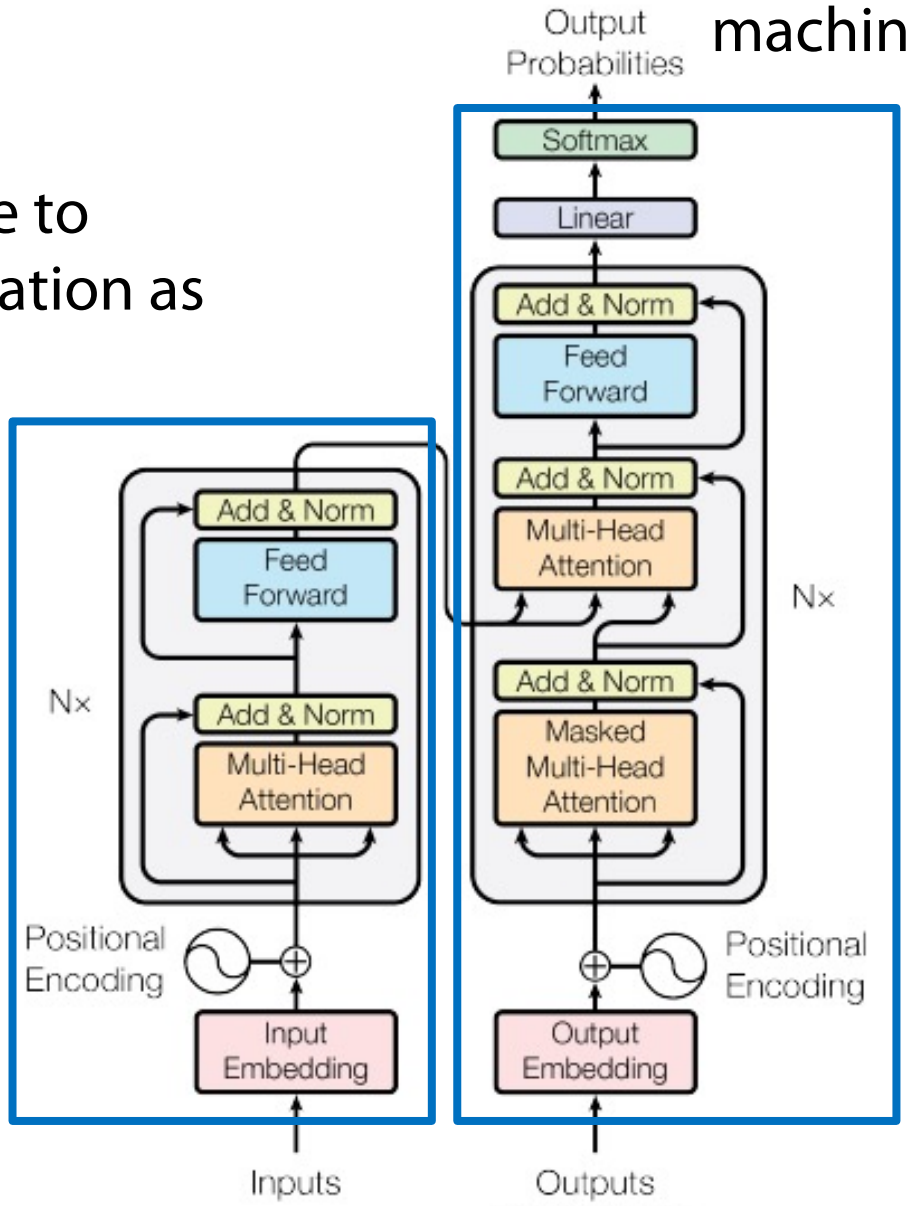


Seq2seq with Attention



Transformer

Using Chinese to English translation as example



Encoder

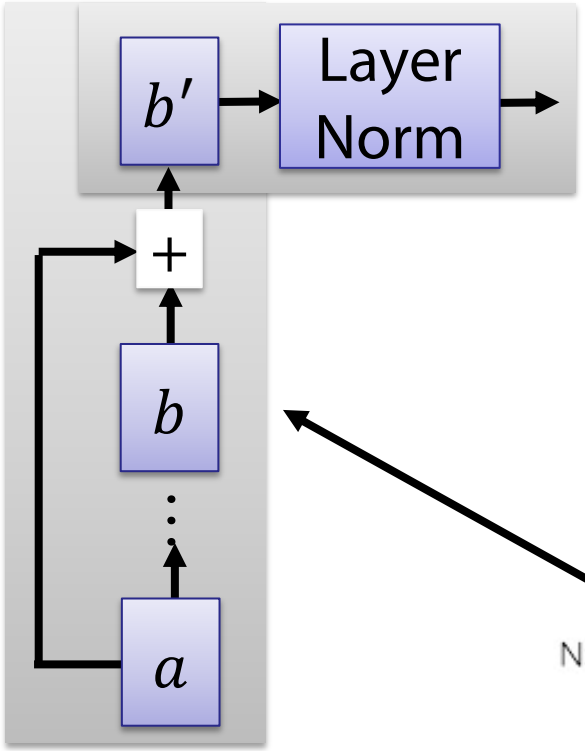
Decoder

機 器 學 習

<BOS> machine

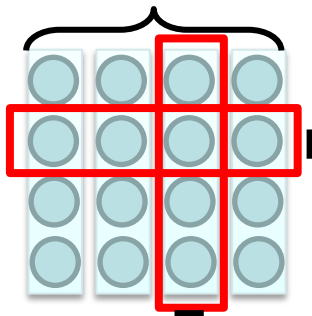
machine learning

Transformer



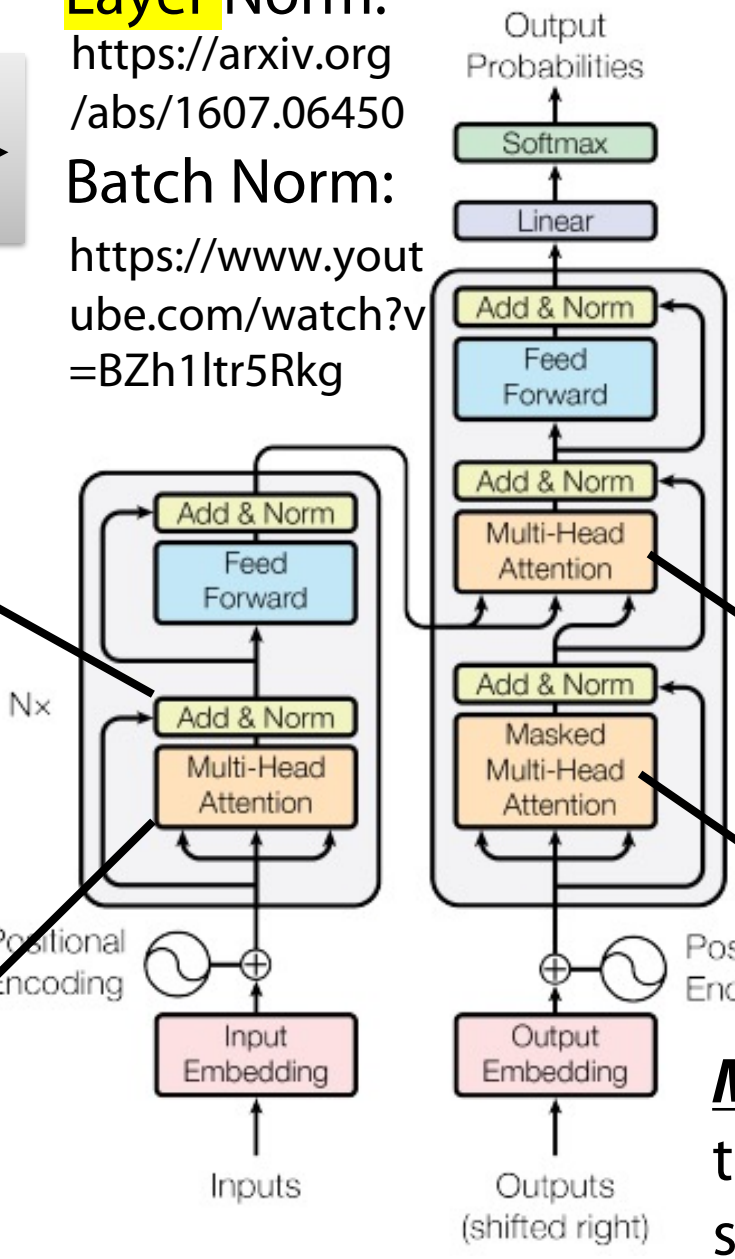
Layer Norm:
https://arxiv.org/abs/1607.06450

Batch Norm:
https://www.youtube.com/watch?v=BZh1ltr5Rkg

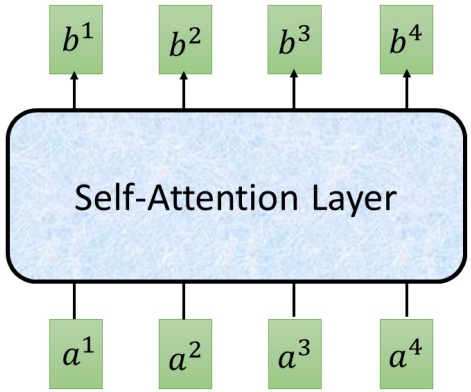


$\mu = 0, \sigma = 1$
Batch

$\mu = 0, \sigma = 1$
Layer



attend on the input sequence

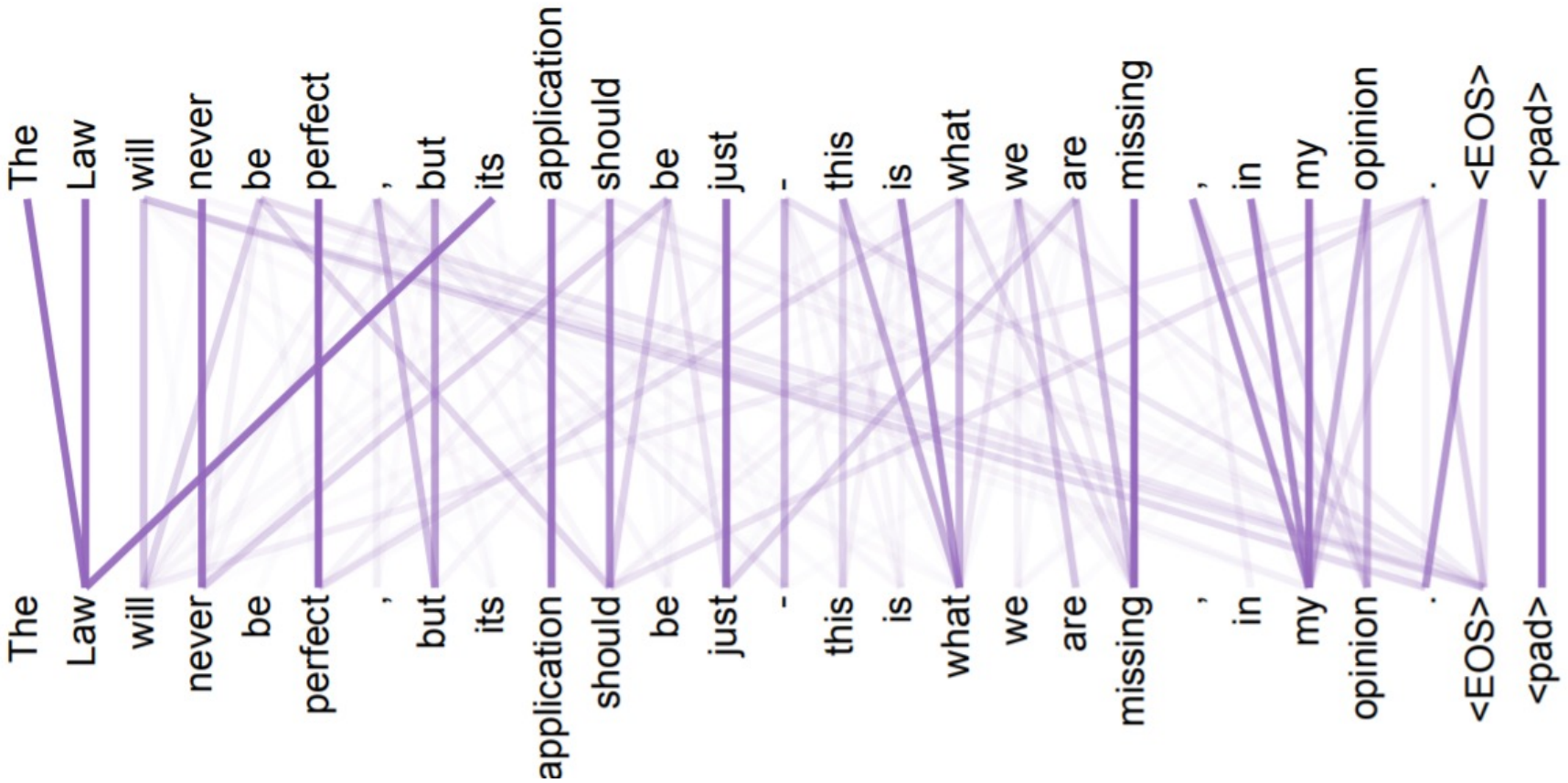


Masked: attend on the generated sequence [MASK]

Masked Multihead Attention

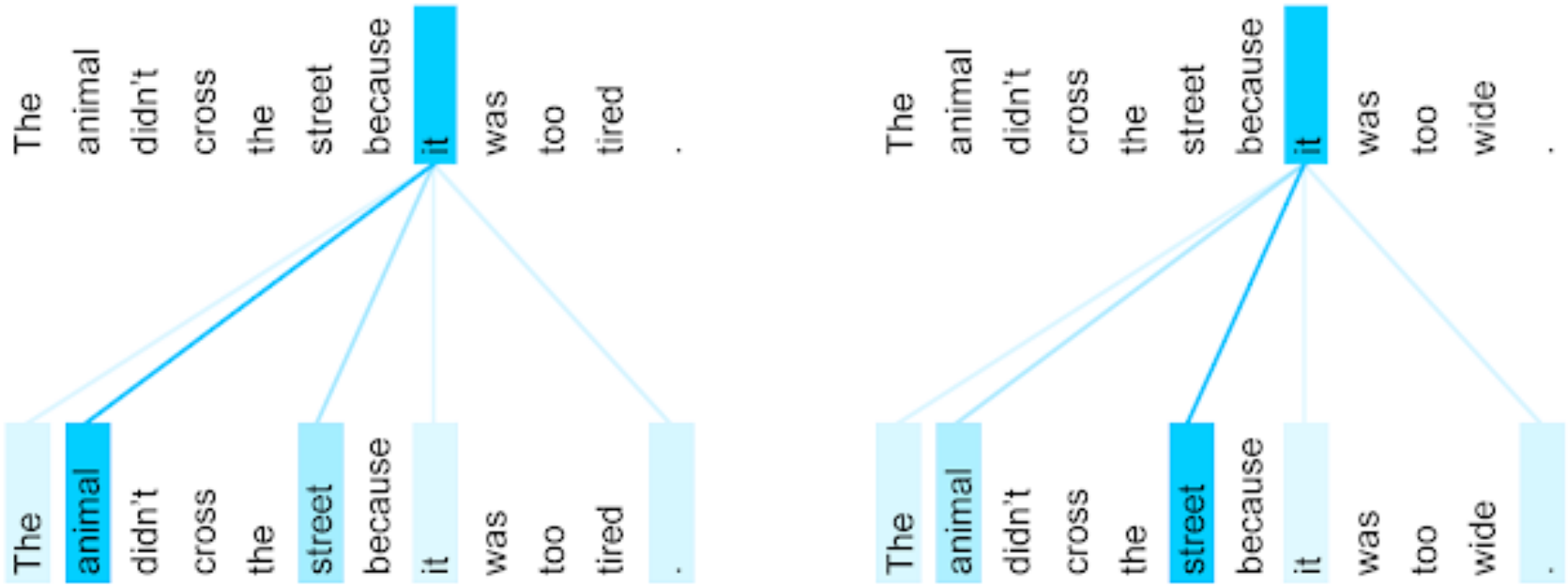
- Decoder should work in parallel as well
- During training all output tokens are known
- Copy output #token times
- For each position use [MASK] token in copies
- Attention becomes possible during training also for decoding
- Train decoder such that [MASK] is replaced correctly while paying attention to the overall output training data

Attention Visualization



https://arxiv.org/abs/1706.03762

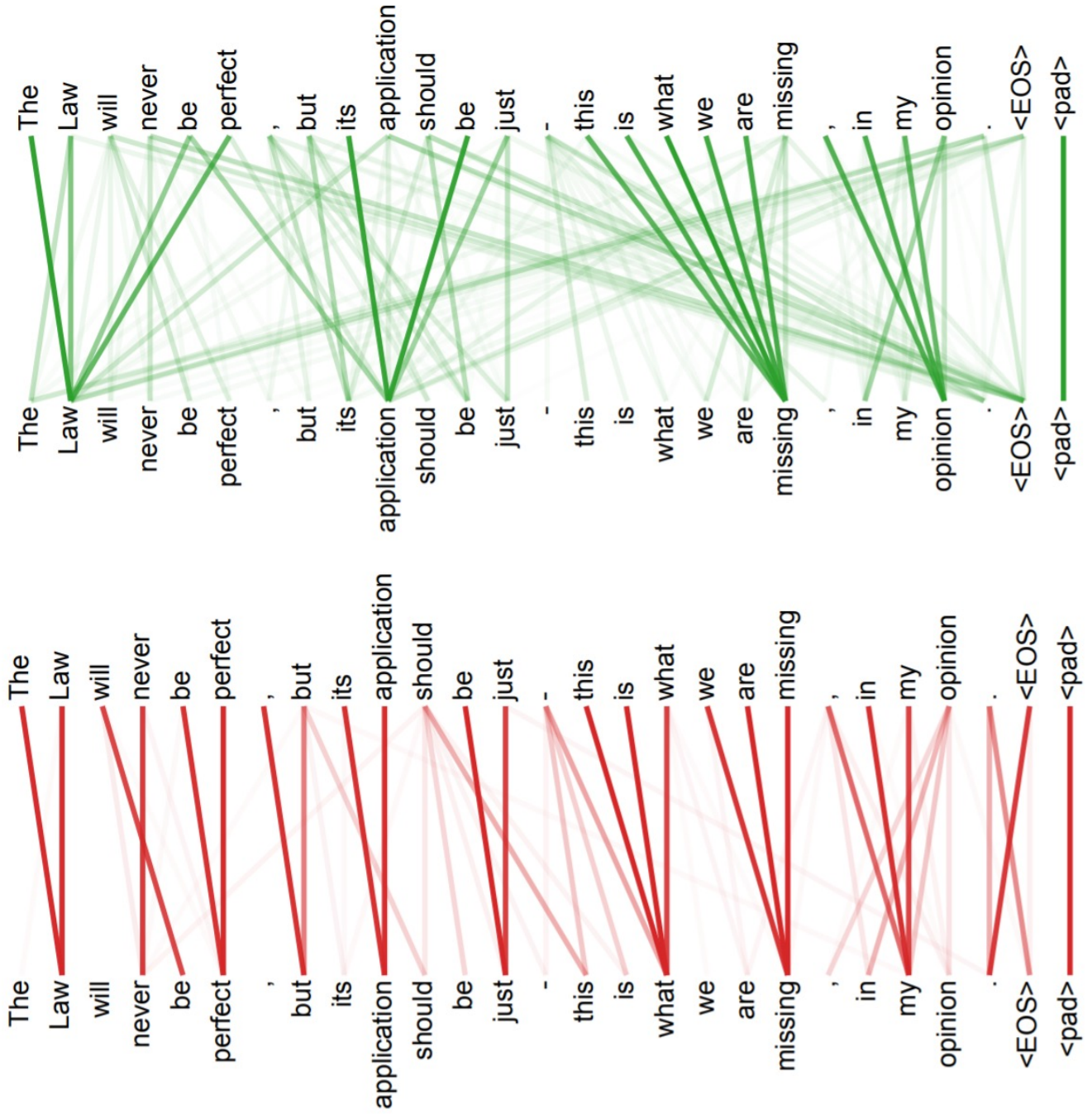
Attention Visualization



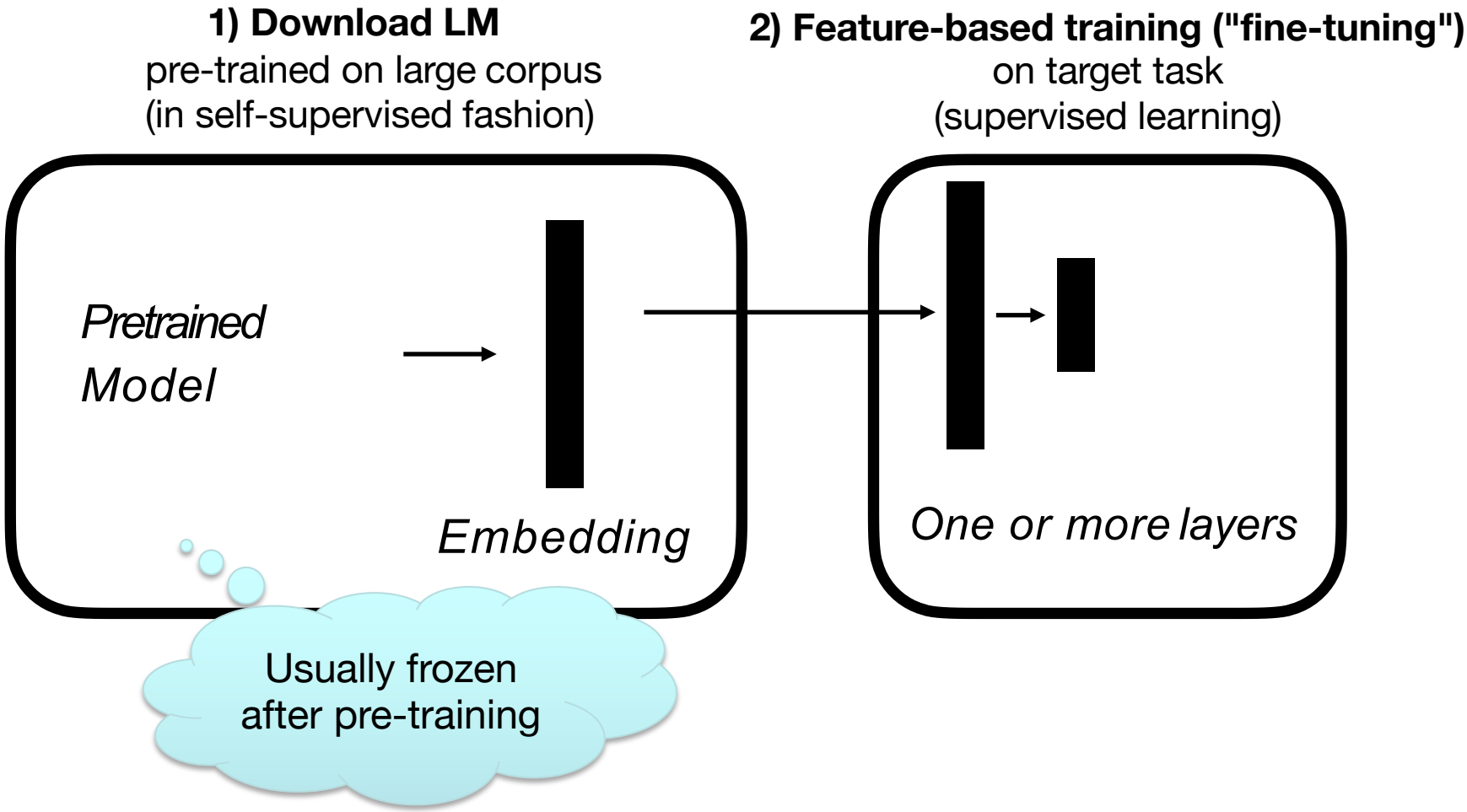
The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

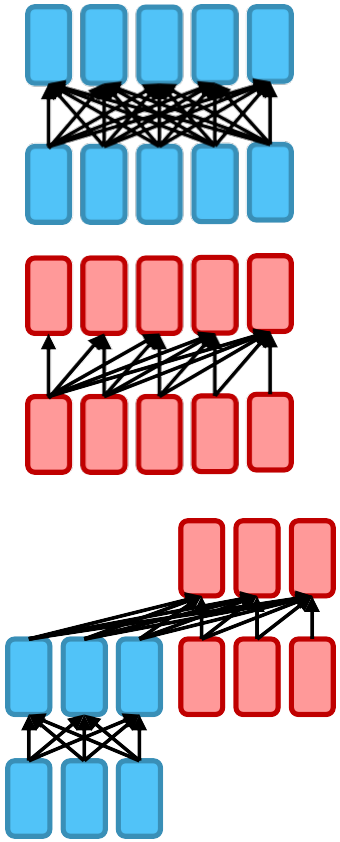
Multi-head Attention



Pre-Training & Fine Tuning

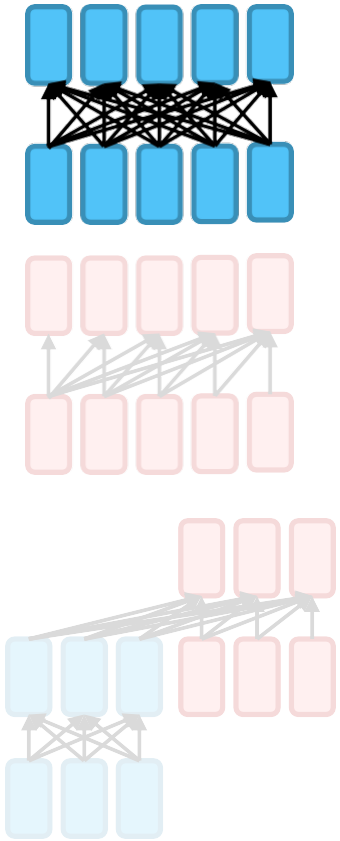


Model Pre-Training



- Encoder
  - Bidirectional context
  - Examples: BERT and its variants
- Decoder
  - Language modeling; better for generation
  - Example: GPT-2, GPT-3, LaMDA
- Encoder-Decoder
  - Sequence-to-sequence model
  - Examples: Transformer, BART, T5

Model Pre-Training

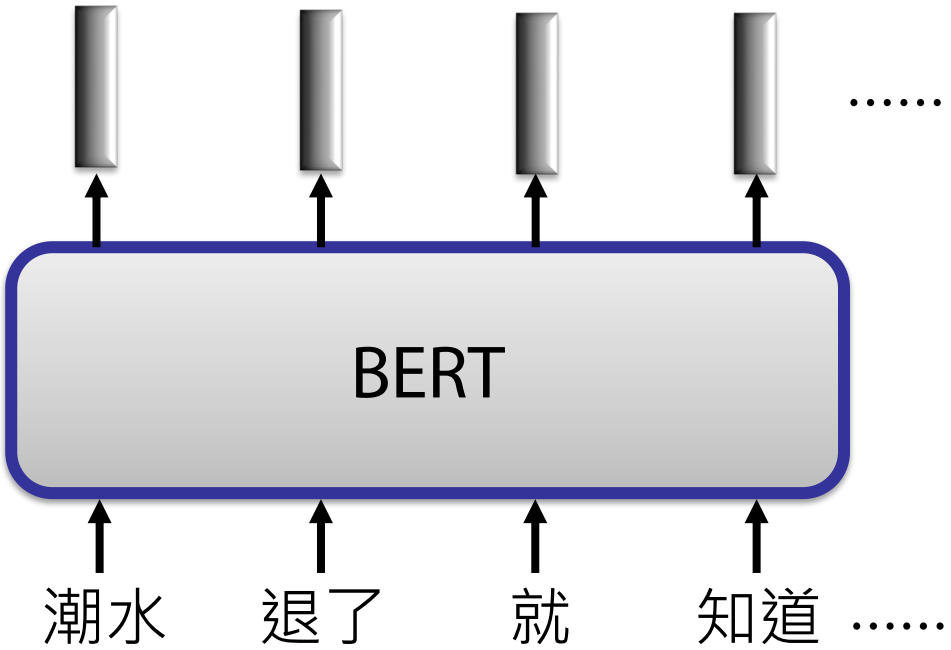


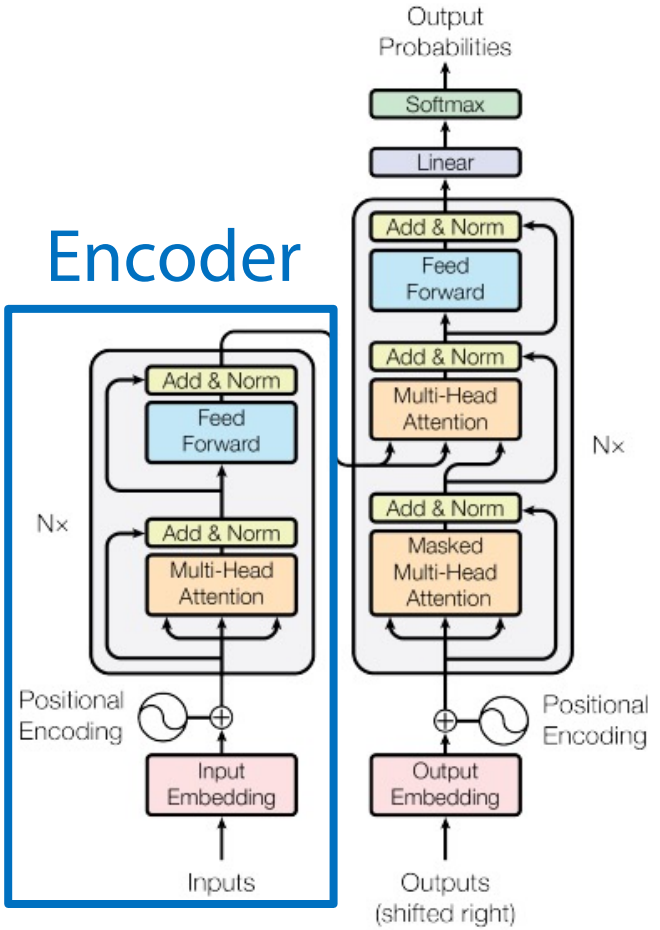
- Encoder
  - Bidirectional context
  - Examples: BERT and its variants
- Decoder
  - Language modeling; better for generation
  - Example: GPT-2, GPT-3, LaMDA
- Encoder-Decoder
  - Sequence-to-sequence model
  - Examples: Transformer, BART, T5

Bidirectional Encoder Representations from Transformers (BERT)

- BERT = Encoder of Transformer

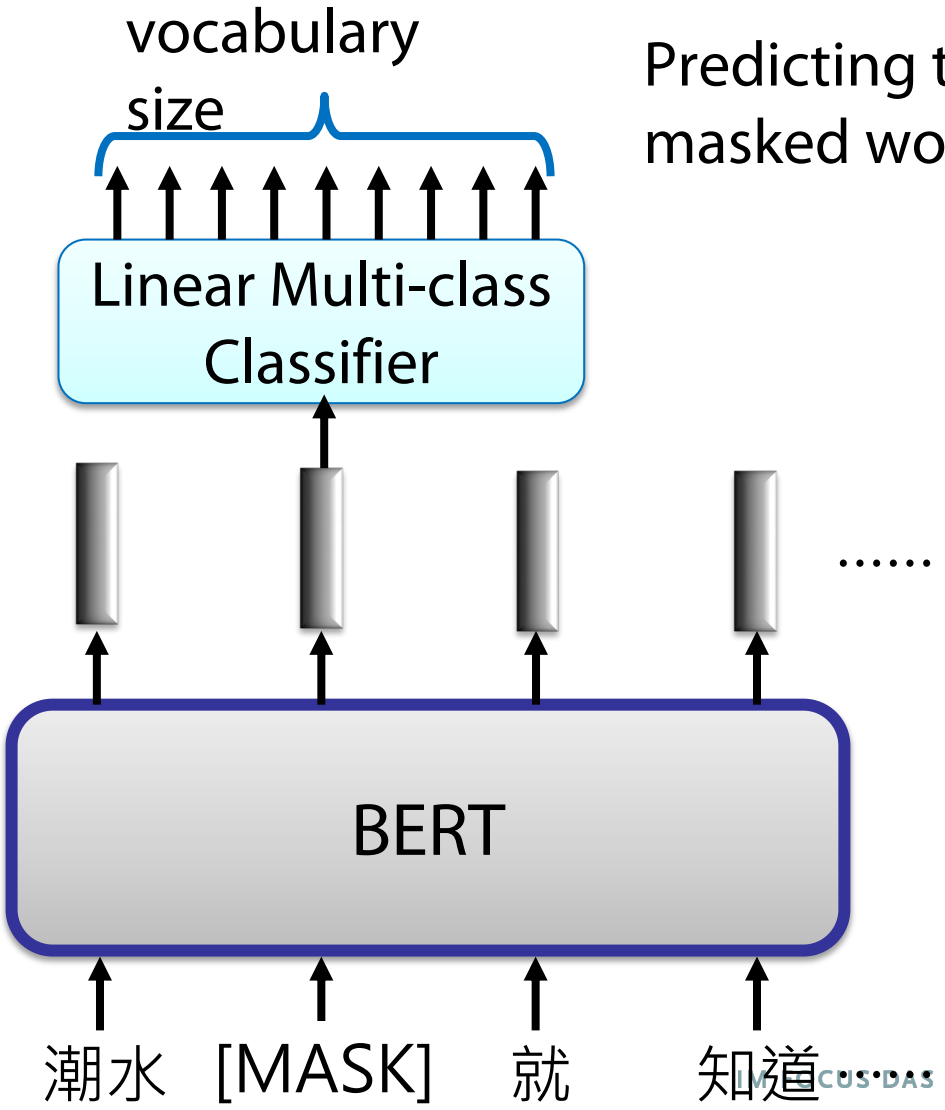
Learned from a large amount of text without annotation





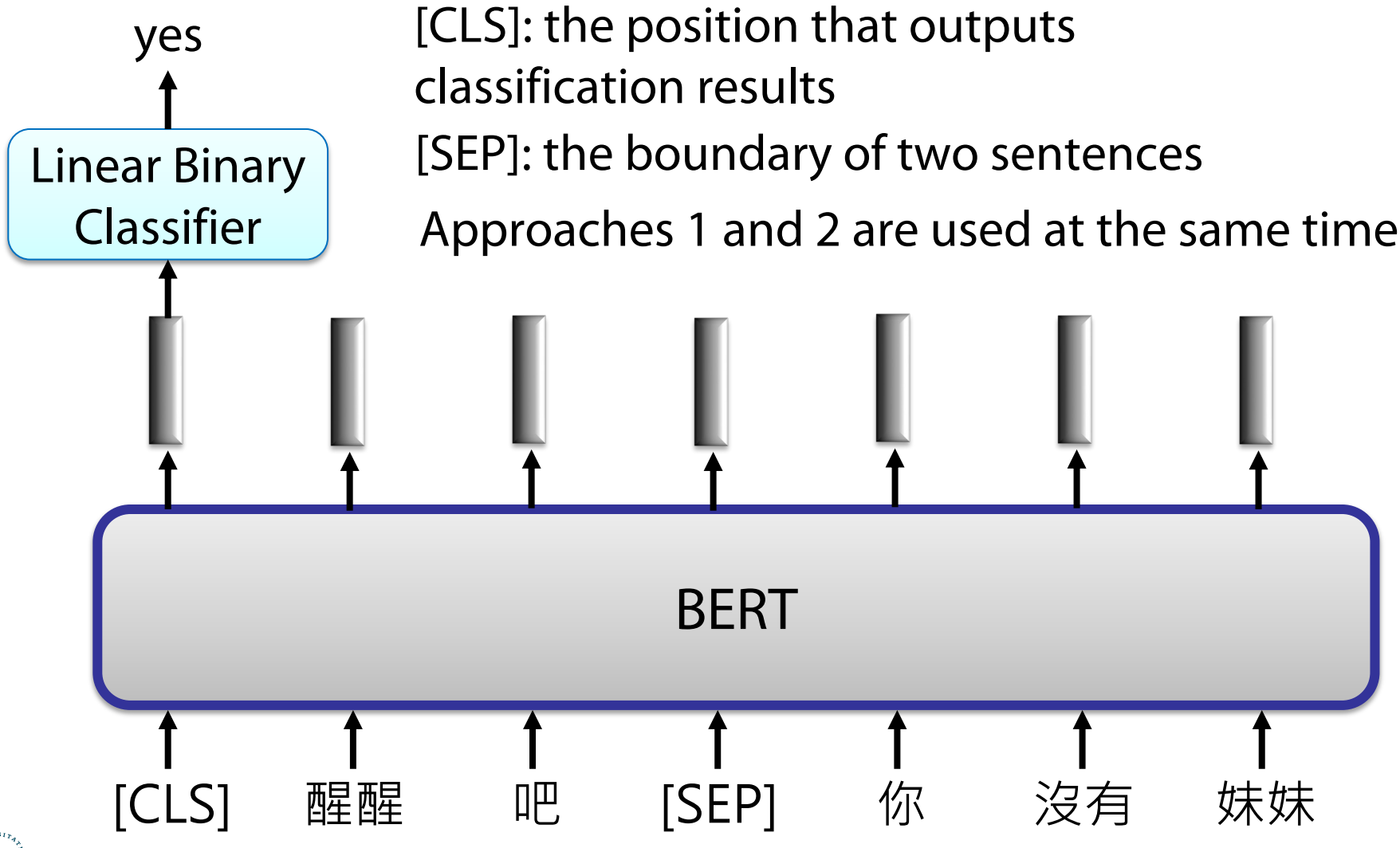
Training of BERT

- Approach 1: Masked LM



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

Training of BERT – Approach 2: Next Sentence Prediction

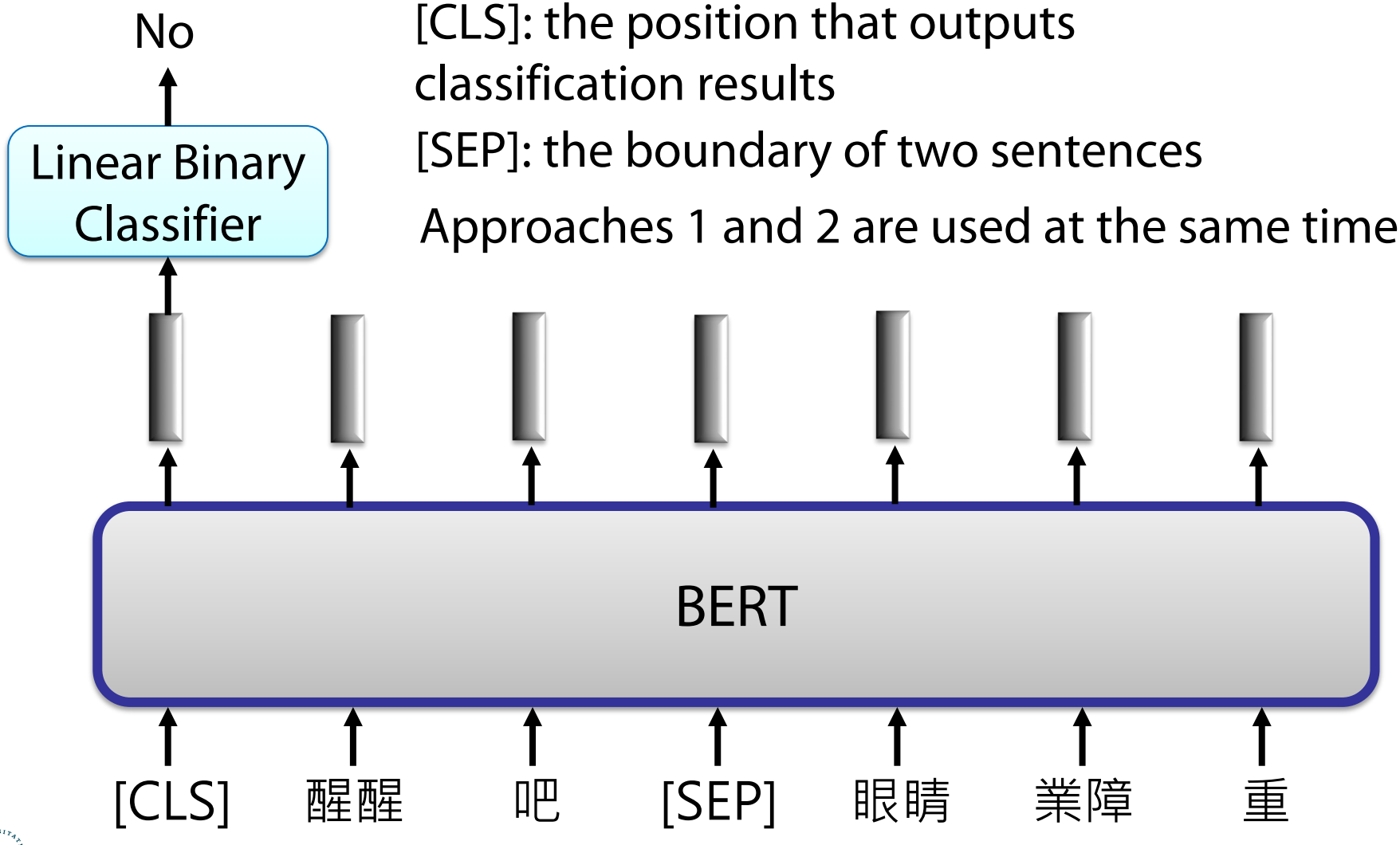


[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

Approaches 1 and 2 are used at the same time.

Training of BERT – Approach 2: Next Sentence Prediction

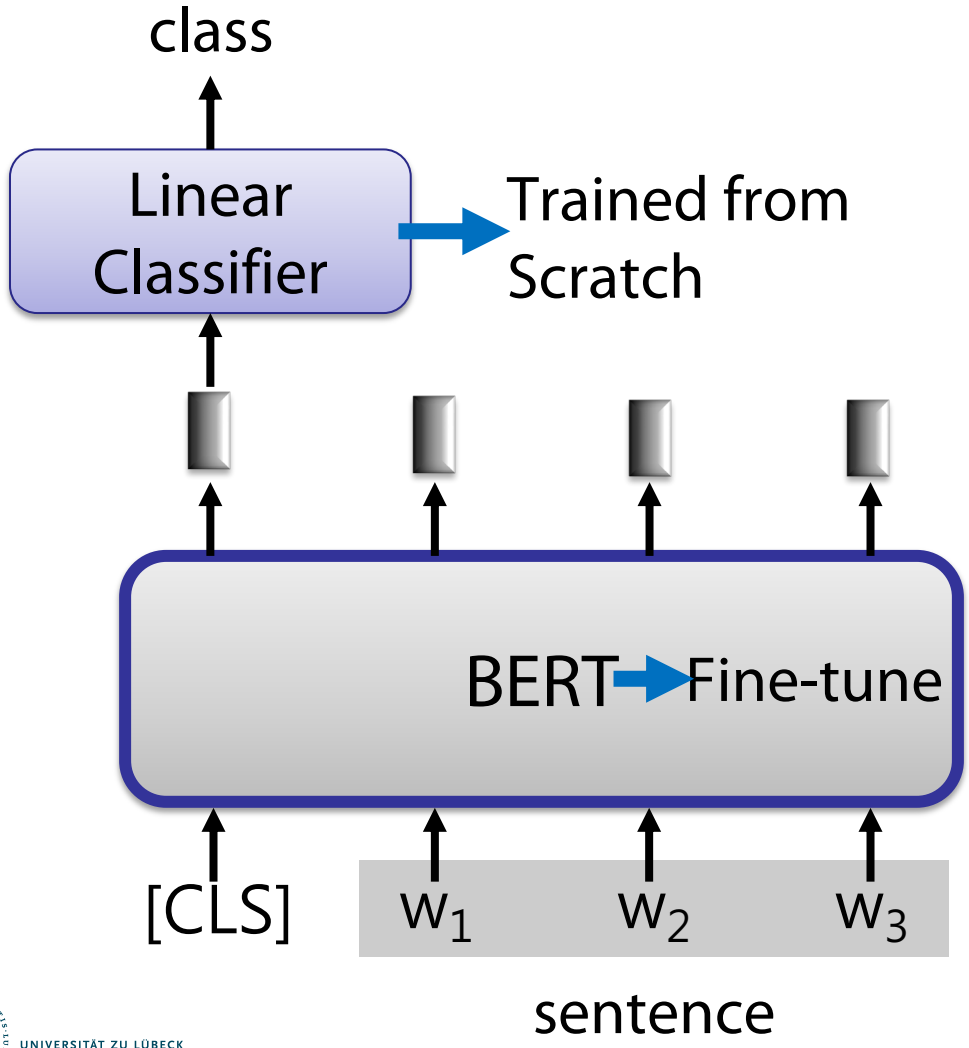


[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

Approaches 1 and 2 are used at the same time.

How to use BERT – Case 1

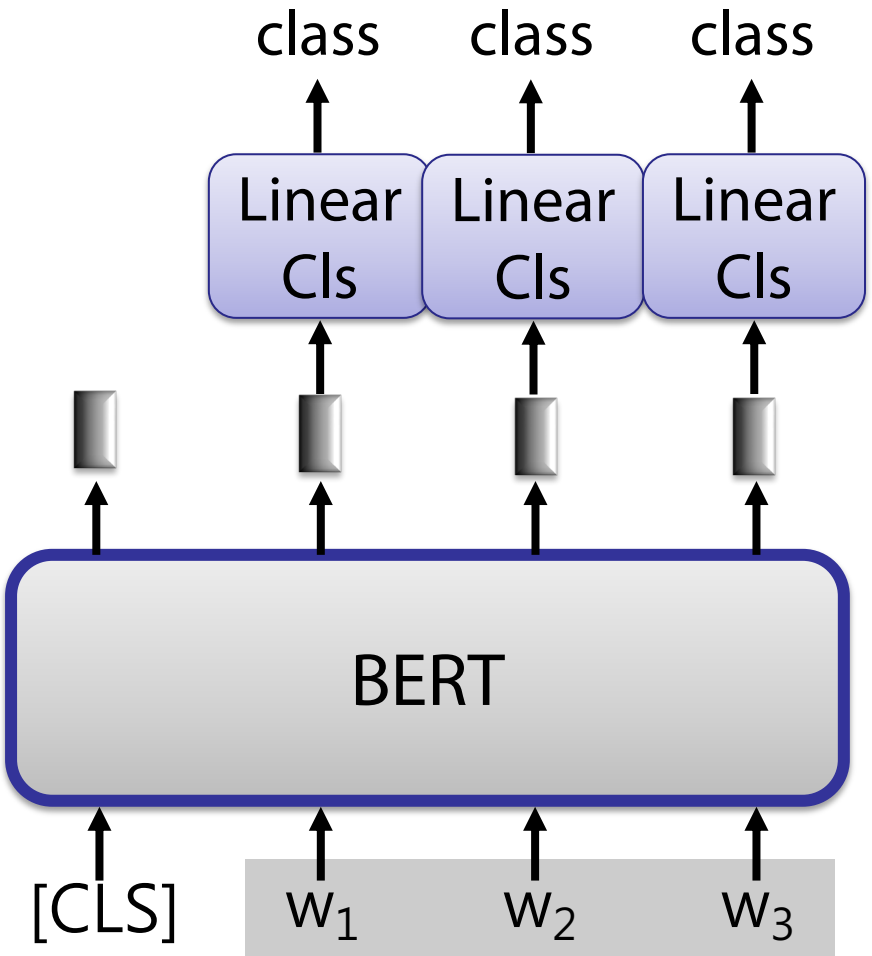


Input: single sentence, output: class

Example: Sentiment analysis, Document Classification

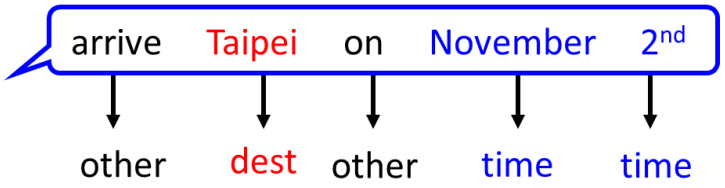
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

How to use BERT – Case 2



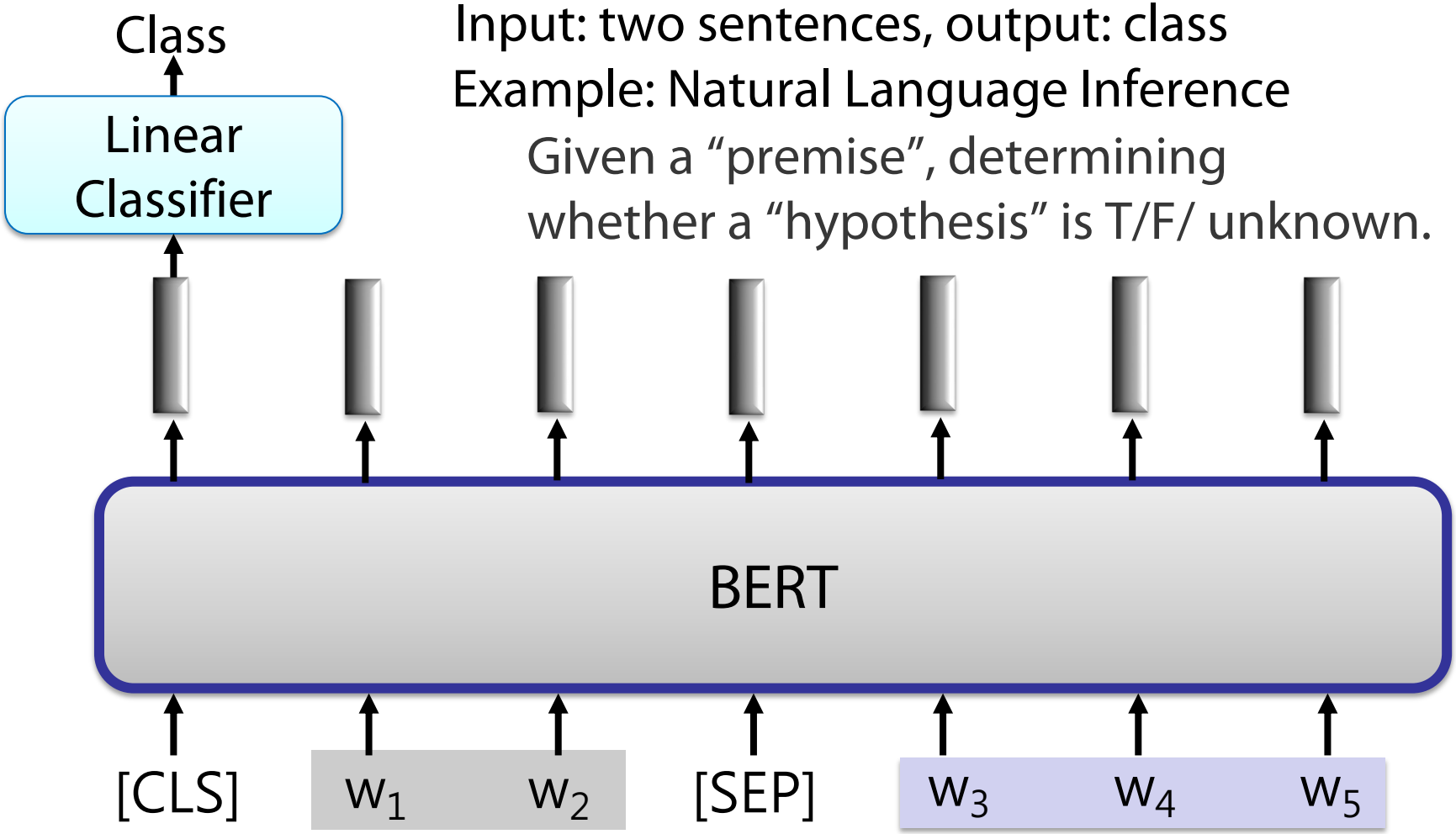
Input: single sentence, output: class of each word

Example: Semantic role labelling



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

How to use BERT – Case 3



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

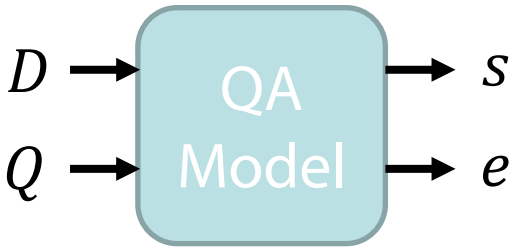
https://rajpurkar.github.io/SQuAD-explorer/

How to use BERT – Case 4

- Extraction-based Question Answering (QA) (E.g. SQuAD)

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$



output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

S=start, e=end

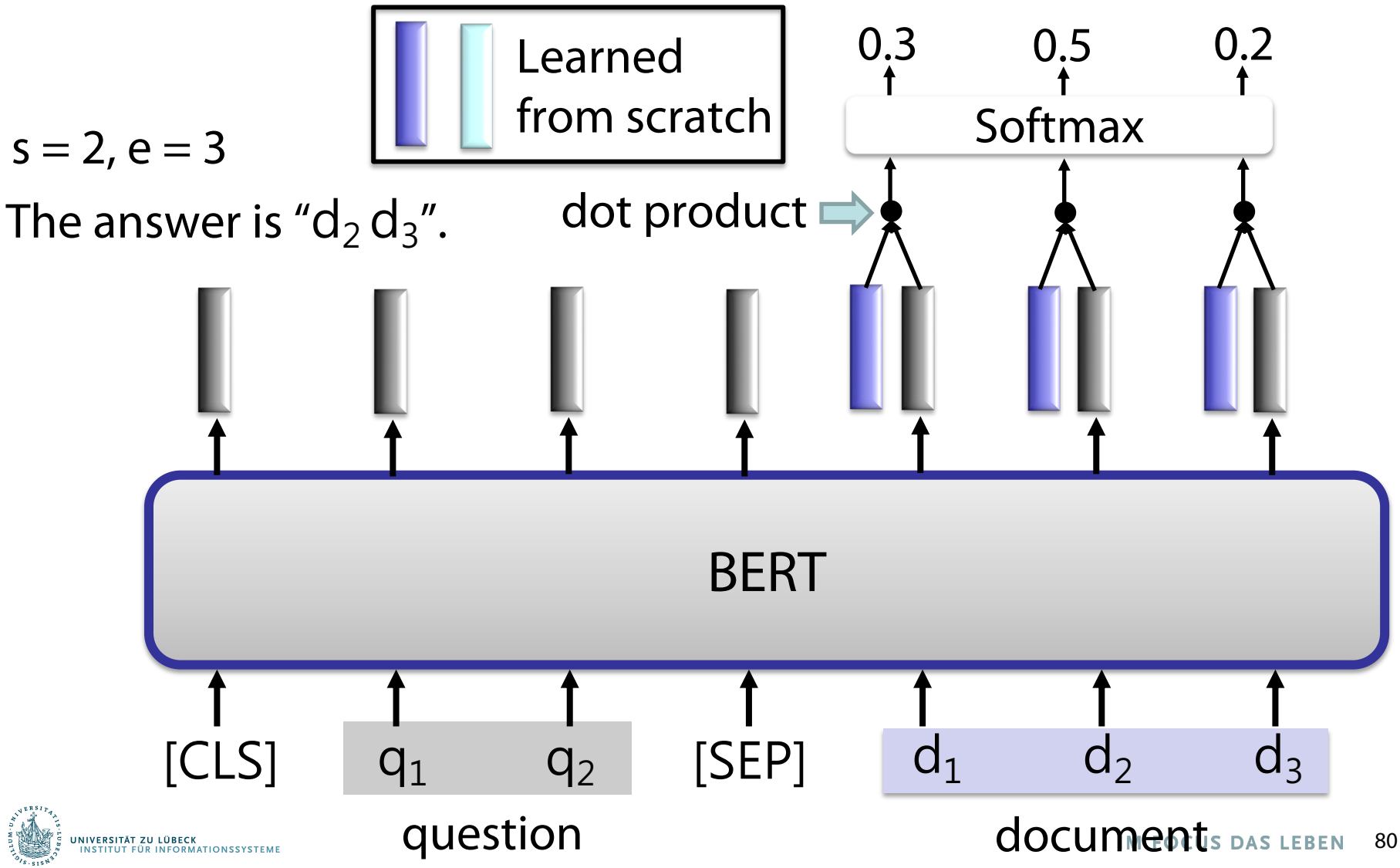
In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

What causes precipitation to fall?
gravity $s = 17, e = 17$

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

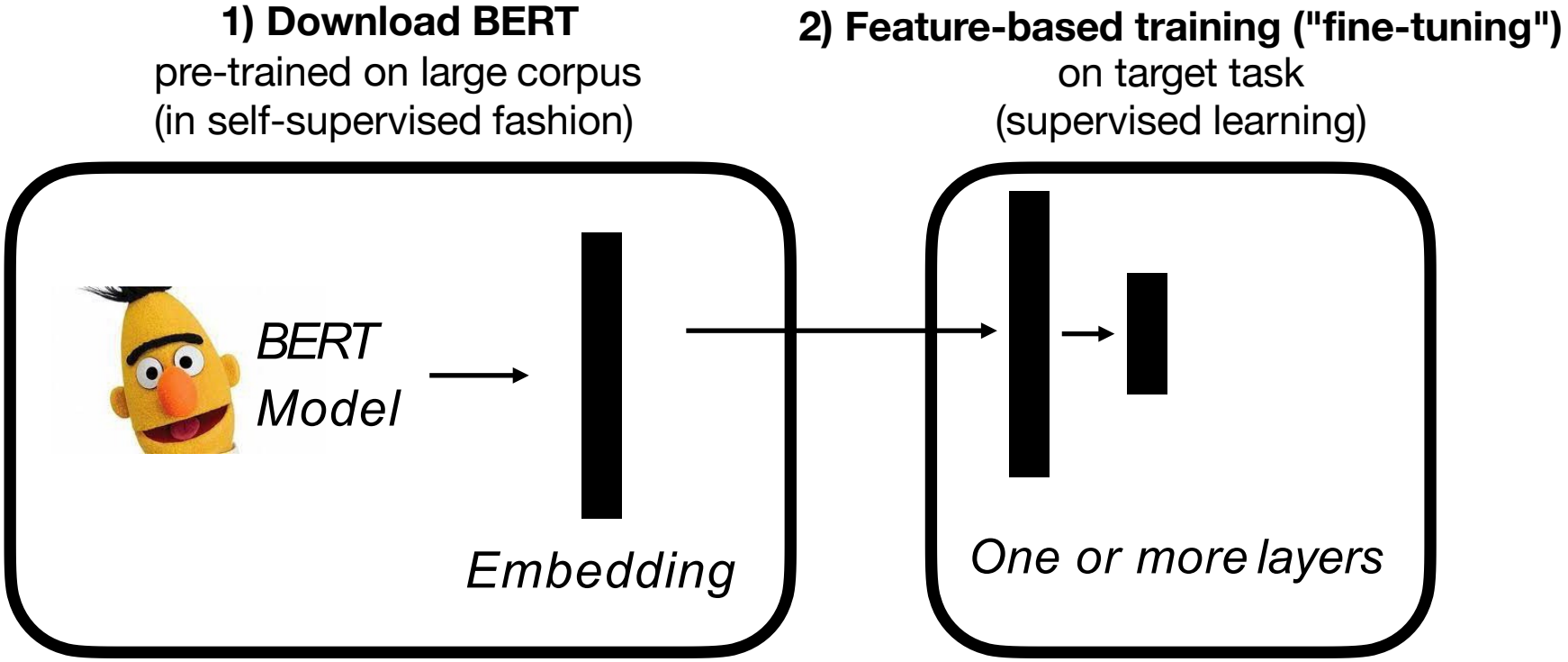
Where do water droplets collide with ice crystals to form precipitation?
within a cloud $s = 77, e = 79$

How to use BERT – Case 4

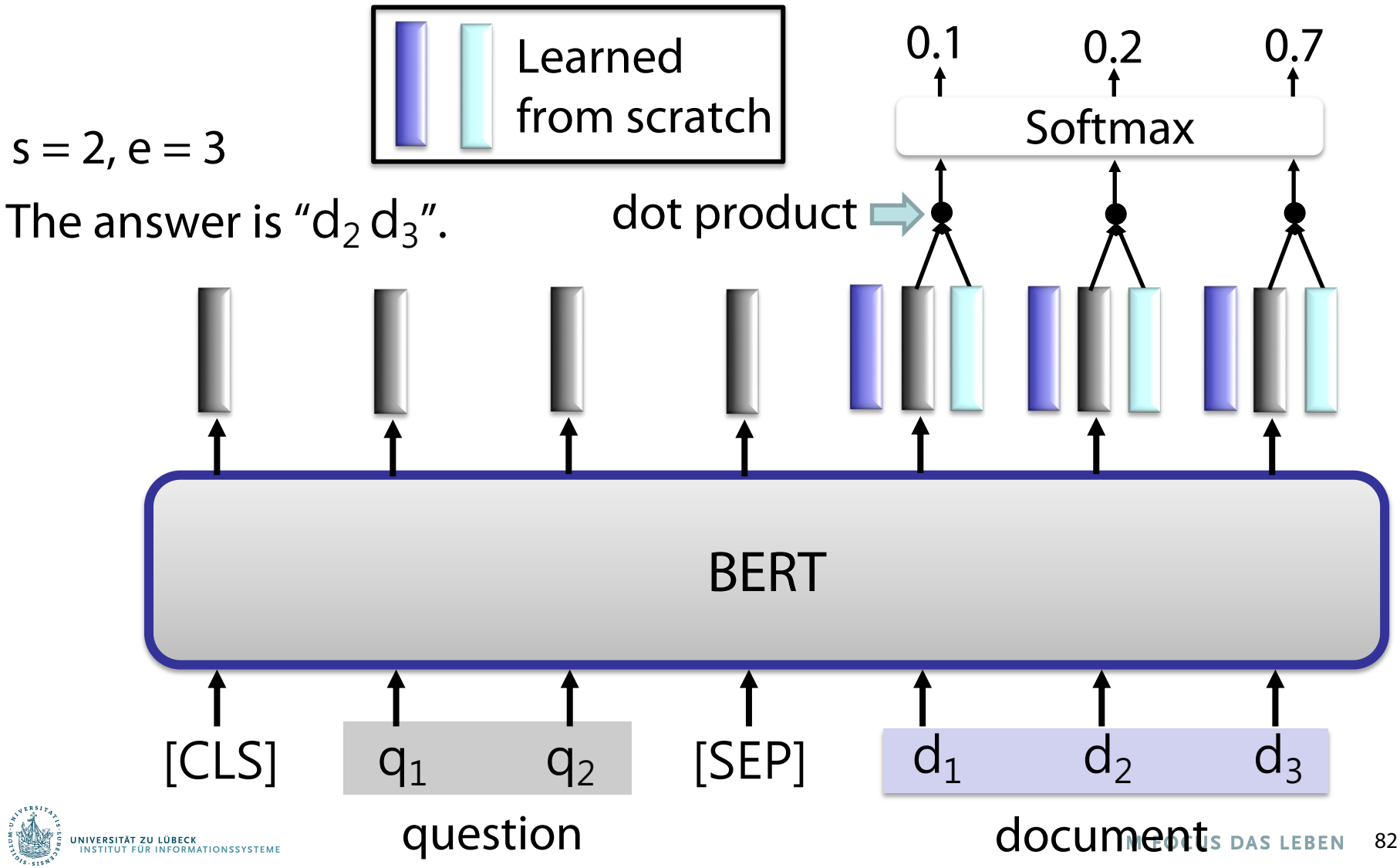


BERT Pre-Training & Fine Tuning

- Keep BERT frozen after pre-training
- Create BERT embeddings for labeled dataset for "downstream task" and train new model on these embeddings

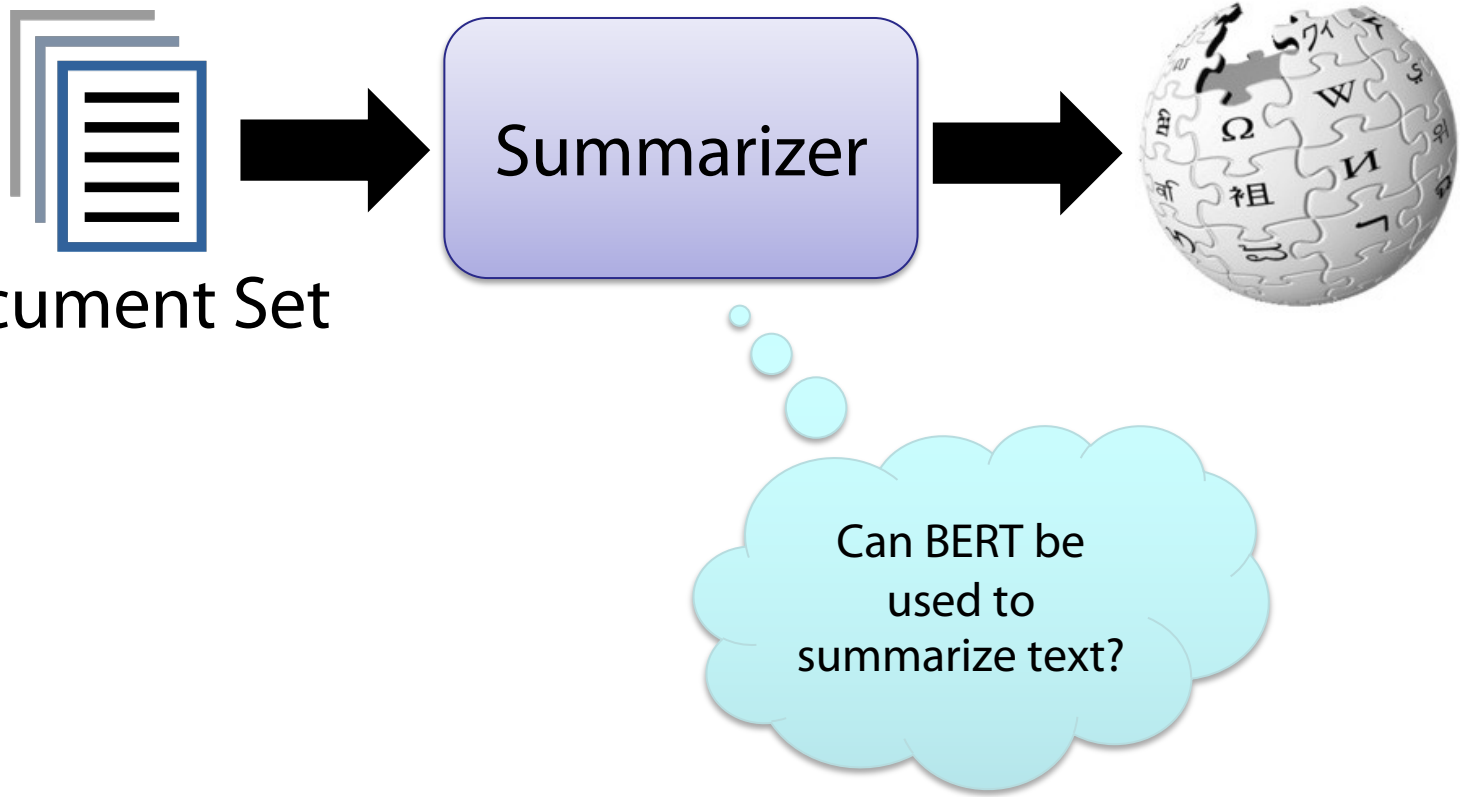


How to use BERT – Case 4



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

Example Application: Summarization



https://arxiv.org/abs/1801.10198

BERT as a Markov Random Field Language Model

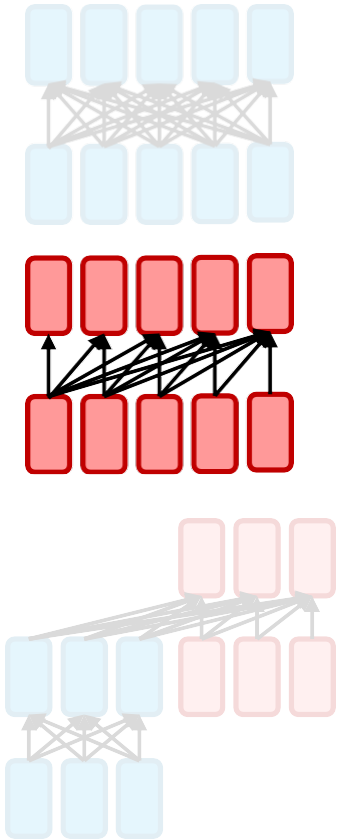
- Wang et al. show that BERT
  (as described by Devlin et al., 2018)
  is essentially a Markov random field language model

Alex Wang, Kyunghyun Cho. BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model. Volume:
In Proc. of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation, June **2019**.
https://arxiv.org/abs/1902.04094

Recap: From word2vec/ELMo via Transformers to BERT

- Language modeling is the "ultimate" NLP task
  - I.e., a perfect language model is also a perfect question answering/entailment/sentiment analysis model
  - Training a massive language model learns millions of latent features which are useful for these other NLP tasks
- E.g., for natural language inference
  - No internal "logical" representation
  - Use language directly to infer new propositions
    - ~~What kind of a thing is the meaning of a sentence?~~
    - What concrete phenomena do you have to deal with to understand a sentence?
- BERT was just a start – many extensions in the literature

Model Pre-Training



- Encoder
  - Bidirectional context
  - Examples: BERT and its variants
- Decoder
  - Language modeling; better for generation
  - Example: GPT-2, GPT-3, LaMDA
- Encoder-Decoder
  - Sequence-to-sequence model
  - Examples: Transformer, BART, T5

GPT (Generative Pre-trained Transformer)

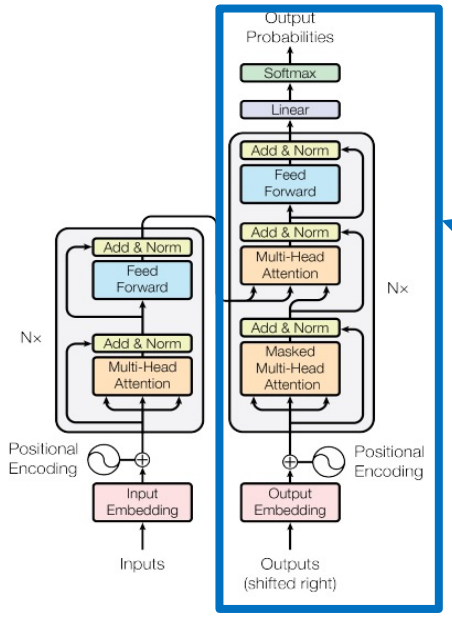
- Developed by OpenAI
- Unidirectional: trained to predict next word in a sentence

GPT (110 million parameters)
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

GPT-2 1.5 billion parameters)
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.
https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

GPT-3 (175 billion parameters)
Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165. https://arxiv.org/abs/2005.14165

Generative Pre-Training (GPT)



Transformer Decoder

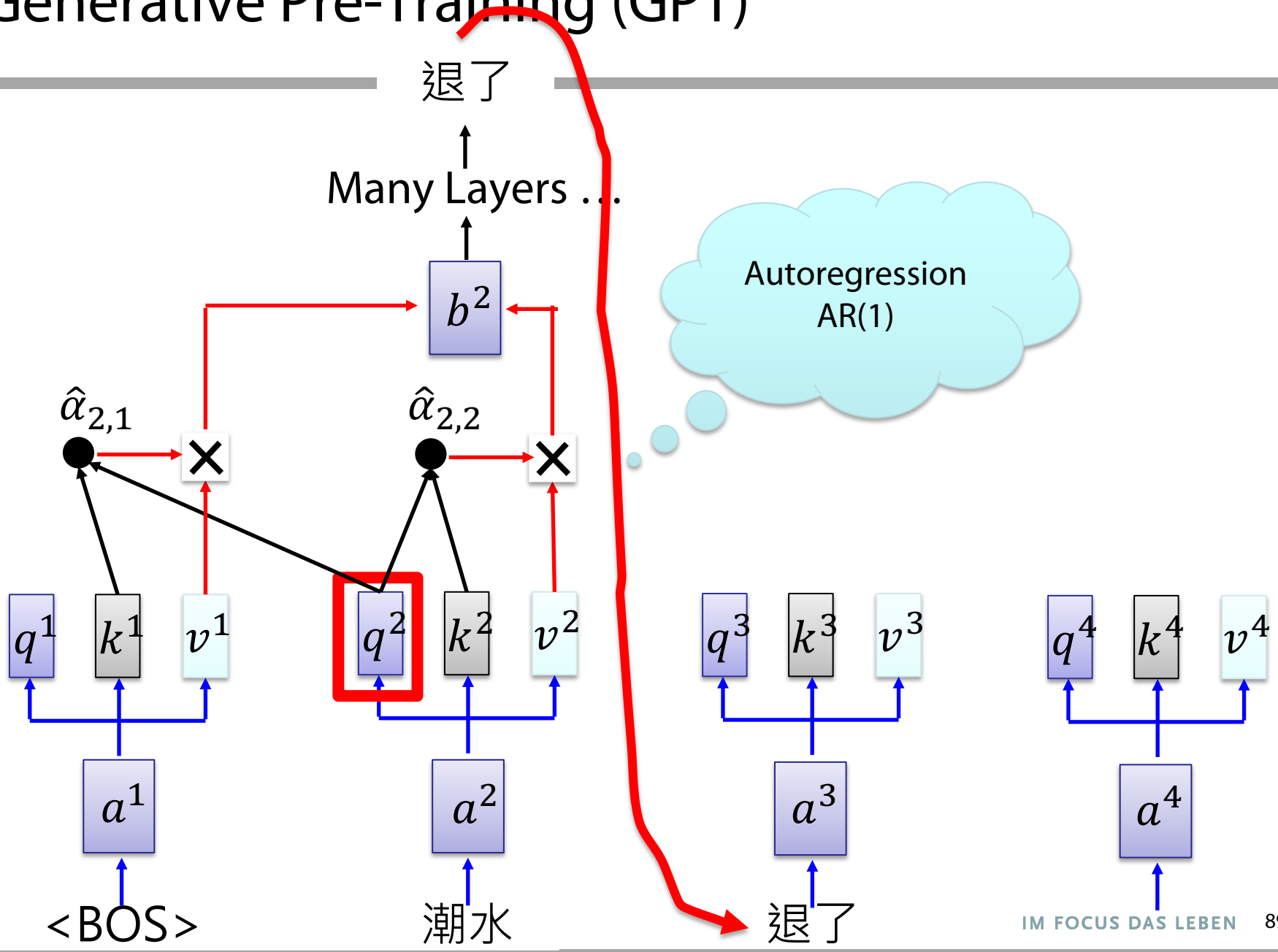
BERT (340M)

ELMO (94M)

GPT-2 (1.5B)

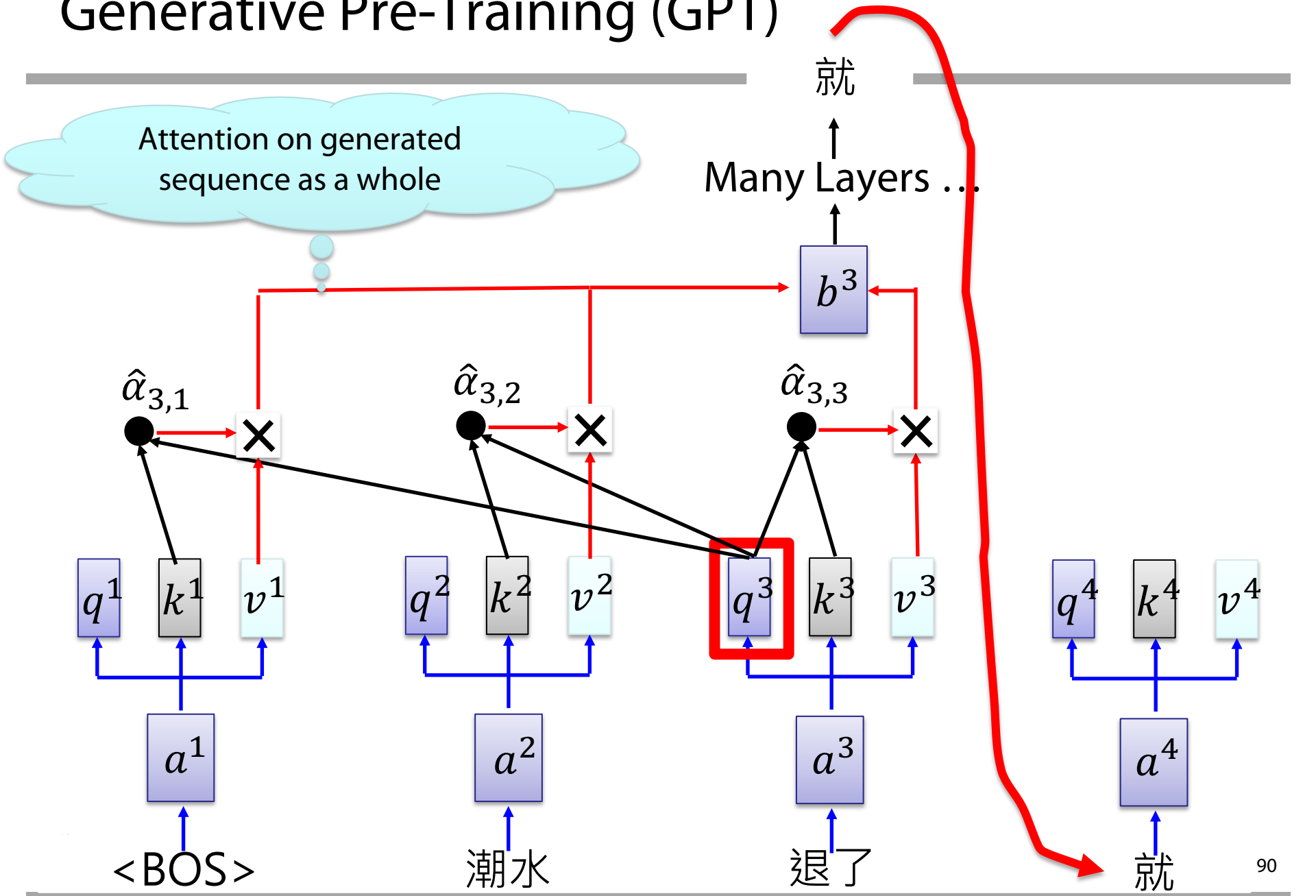
https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Generative Pre-Training (GPT)



http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

Generative Pre-Training (GPT)



Application: Summaries (Open AI)

ORIGINAL TEXT — 26,449 WORDS
SOURCE: PROJECT GUTENBERG↗

ALICE'S ADVENTURES IN WONDERLAND
Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 3.0

CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversations?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

thousand miles down, I think–' (fo
several things of this sort in her
and though this was not a VE
showing off her knowledge, as the
her, still it was good practice to say
the right distance–but then I
Longitude I've got to?' (Alice had
or Longitude either, but thought th
to say.)

Presently she began again. 'I
THROUGH the earth! How funn
among the people that walk with t
Antipathies, I think–' (she was rath
listening, this time, as it didn't sou
but I shall have to ask them what
you know. Please, Ma'am, is this N
(and she tried to curtsey as she spe
you're falling through the air! Do y
it?) 'And what an ignorant little gir
No, it'll never do to ask: perhaps
somewhere.'

Down, down, down. There was
soon began talking again. 'Dinah
night, I should think!' (Dinah wa
remember her saucer of milk at t
wish you were down here with me
air, I'm afraid, but you might catch
mouse, you know. But do cats eat
Alice began to get rather sleepy
herself, in a dreamy sort of way, 'D
bats?' and sometimes, 'Do ba

1 2 3 4 5 6 7 8 9 10 11 12 13 14

The original text is divided into sections, and each section is summarized.

66 SUMMARIES — 6,024 WORDS

ALICE is bored sitting by her sister on the bank, and she's thinking about making a daisy chain when a white rabbit with pink eyes runs by. She's surprised to see a rabbit with a waistcoat pocket and a watch, and she follows it down a rabbit hole. She falls down a deep well, and as she's falling she takes note of the shelves and jars she passes. She thinks about how brave she'll seem when she gets home, and how she'll never tell anyone about her fall. She wonders how deep the well is and how many miles she's fallen.

ALICE thinks she's fallen a thou
what latitude or longitude she
fall right through the earth and wha
she'll emerge in is. She starts to get
about walking hand in hand with h
she's ever eaten a bat. Suddenly she
and dry leaves, and she's not hurt.
in a moment.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Section summaries are summarized again into higher-level summaries.

6 SUMMARIES — 830 WORDS

Alice falls down a deep well and lands in a heap of sticks and leaves. She follows the White Rabbit down a long passage, but loses sight of it. She finds a bottle that says "Drink Me" on it and drinks it, shrinking to 10 inches tall. She grows to 9 feet tall and then shrinks again. She cries and grows larger again. A mouse appears and Alice asks it for help. The mouse agrees to tell Alice its story if she'll take it to the shore. Once there, the animals discuss how to get dry again. The Dodo suggests a Caucus-race to get dry. Everyone crowds around Alice asking for prizes. Alice pulls out a box of comfits and hands them out as prizes. The Mouse then asks Alice if she has a prize for herself, and she says she only has a thimble. The Dodo takes the thimble and presents it to Alice, saying they all beg her acceptance

Alice grows larger after drinki
table. She stops growing befo
then starts to miss being at home a
down the rabbit hole. She hears the
realizes she's now much bigger tha
longer afraid of it. Alice decides h
back to her normal size, and her se
way into the garden. She doesn't
those things, however. She sees
hookah on a mushroom and asks hi
to her normal size. The Caterpillar
the mushroom will make her grow
will make her shorter. She starts eat

1 2 3 4 5 6

The summarizing process continues
until a complete summary is achieved.

COMPLETE SUMMARY — 136 WORDS

Alice falls down a rabbit hole and grows to giant size after drinking a mysterious bottle. She decides to focus on growing back to her normal size and finding her way into the garden. She meets the Caterpillar who tells her that one side of a mushroom will make her grow taller, the other side shorter. She eats the mushroom and returns to her normal size. Alice attends a party with the Mad Hatter and the March Hare. The Queen arrives and orders the execution of the gardeners for making a mistake with the roses. Alice saves them by putting them in a flowerpot. The King and Queen of Hearts preside over a trial. The Queen gets angry and orders Alice to be sentenced to death. Alice wakes up to find her sister by her side.

LLMs of the GPT family

- Generate term papers
- Generate code
- Generate Powerpoint presentations
- Generate useful completions of texts in Word?
- …

- The successor to GPT-3, most likely called GPT-4, is expected to be unveiled in the near future, perhaps as soon as 2023
- Interim versions: GPT-3.5, GPT-3.6

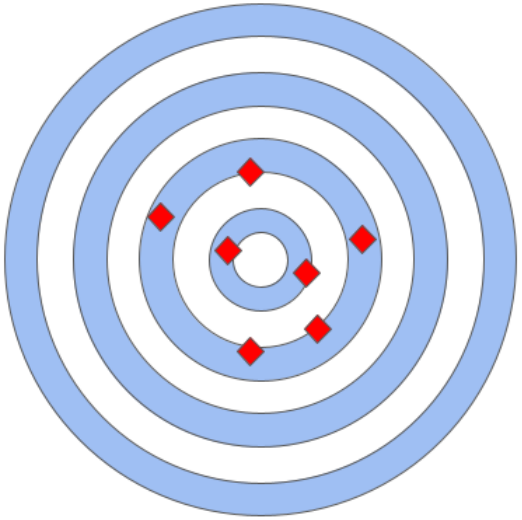
Compatibility vs. Alignment in LLMs

A model's *capability* is typically evaluated by **how well it is able to optimize its objective function**, the mathematical expression that defines the goal of the model

Alignment, on the other hand, is concerned with **what we actually want the model to do** versus what it is being trained to do

Models like the original GPT-3 are misaligned

High capability
Low alignment



Low capability
High alignment

Assessment GPT-3

- **Capable of generating human-like text**, but they may not always produce output that is consistent with human expectations or desirable values
- In practical applications, however, **these models are intended to perform some form of valuable cognitive work**,
  - and there is a clear divergence between the way these models are trained and the way we would like to use them
- Humans generate language by choosing text sequences that are best for the given situation,
  - using our background knowledge and common sense to guide this process
- Language model training strategies can produce misalignment (e.g., using [MASK])
  - Model which is only trained to predict the next word (or a masked word) in a text sequence, may not necessarily be learning some higher-level representations of its meaning
  - Model struggles to generalize for tasks or contexts that require a deeper understanding of language

Improving Language Model Behavior by Training on a Curated Dataset

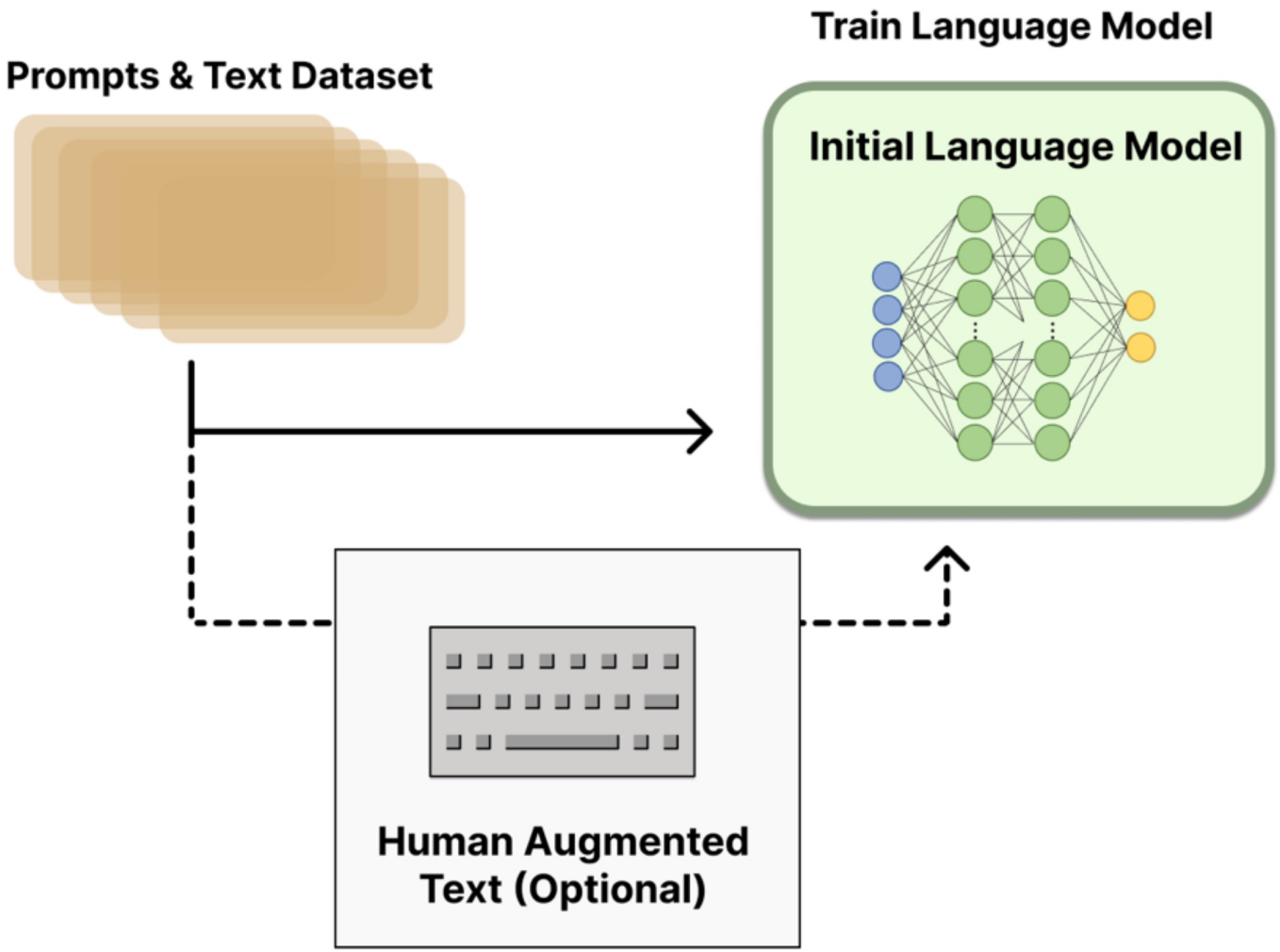
June 10, 2021
5 minute read

Our latest research finds we can improve language model behavior with respect to specific behavioral values by fine-tuning on a small, curated dataset.

READ PAPER

We've found we can improve language model behavior with respect to specific behavioral values by fine-tuning on a curated dataset of <100 examples of those values. We also found that this process becomes more effective as models get larger. While the technique is still nascent, we're looking for OpenAI API users who would like to try it out and are excited to find ways to use these and other techniques in production use cases.

Train on Curated Dataset



Aligning Language Models to Follow Instructions

January 27, 2022
16 minute read

We've trained language models that are much better at following user intentions than GPT-3 while also making them more truthful and less toxic, using techniques developed through our alignment research. These *InstructGPT* models, which are trained with humans in the loop, are now deployed as the default language models on our API.

READ PAPER VIEW MODEL CARD

InstructGPT is better than GPT-3 at following English instructions.

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

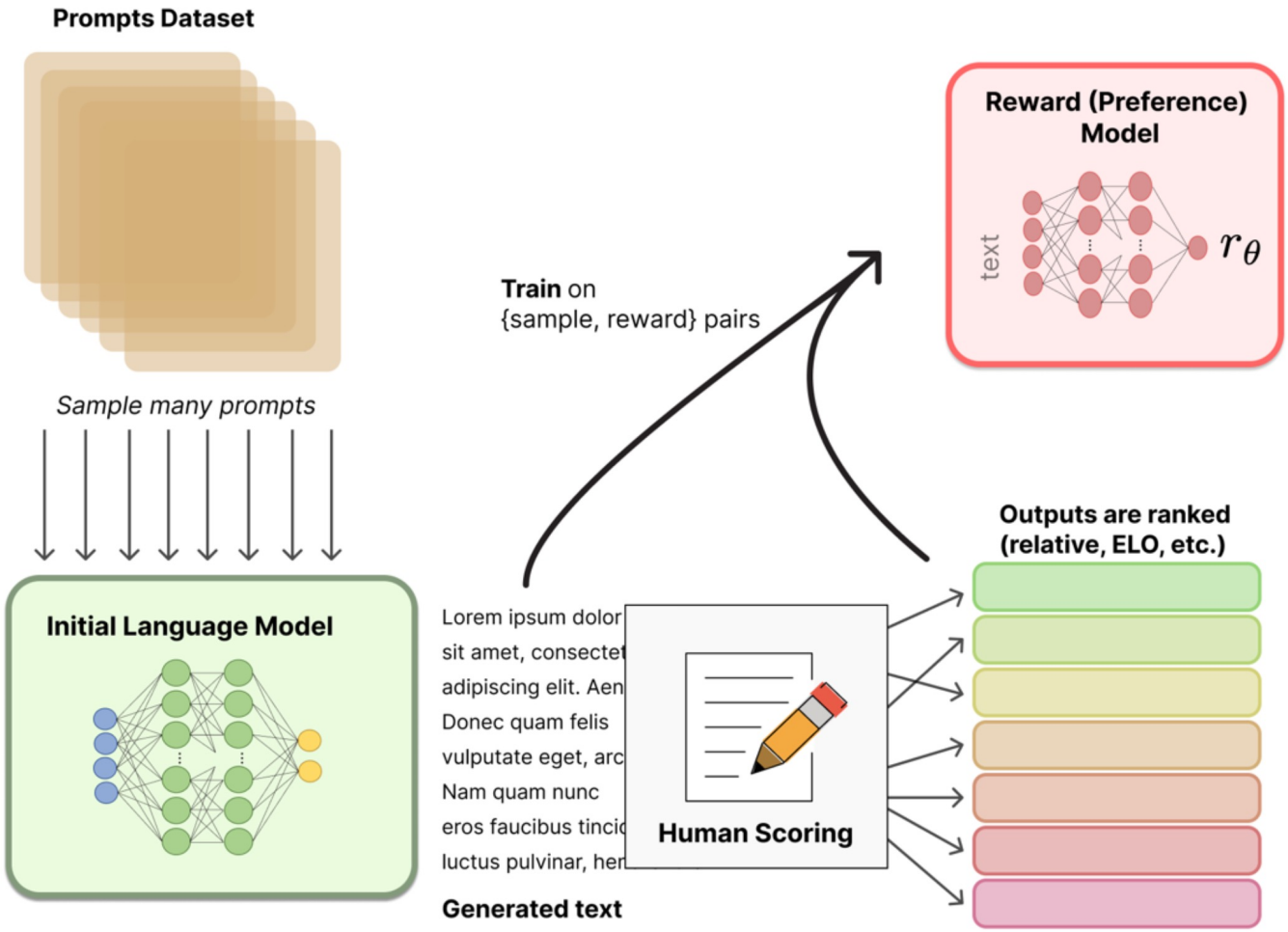
```
People went to the moon, and they took pictures of what they saw, and sent them
back to the earth so we could all see them.
```

InstructGPT: Reinforcement Learning from Human Feedback (RLHF)

1. Pretraining a language model (LM),
   - OpenAI used a smaller version of GPT-3 for its first popular RLHF model, InstructGPT
2. Gathering data and training a reward model (RM, aka preference model), and
   - Get a model that takes in a sequence of text, and returns a scalar reward which should numerically represent the human preference
   - The training dataset of prompt-generation pairs for the RM is generated by sampling a set of prompts from a predefined dataset
3. Fine-tuning the LM with reinforcement learning

https://huggingface.co/blog/rlhf

Train a Reward (Preference) Model



ELO rating: https://en.wikipedia.org/wiki/Elo_rating_system

Fine-Tuning

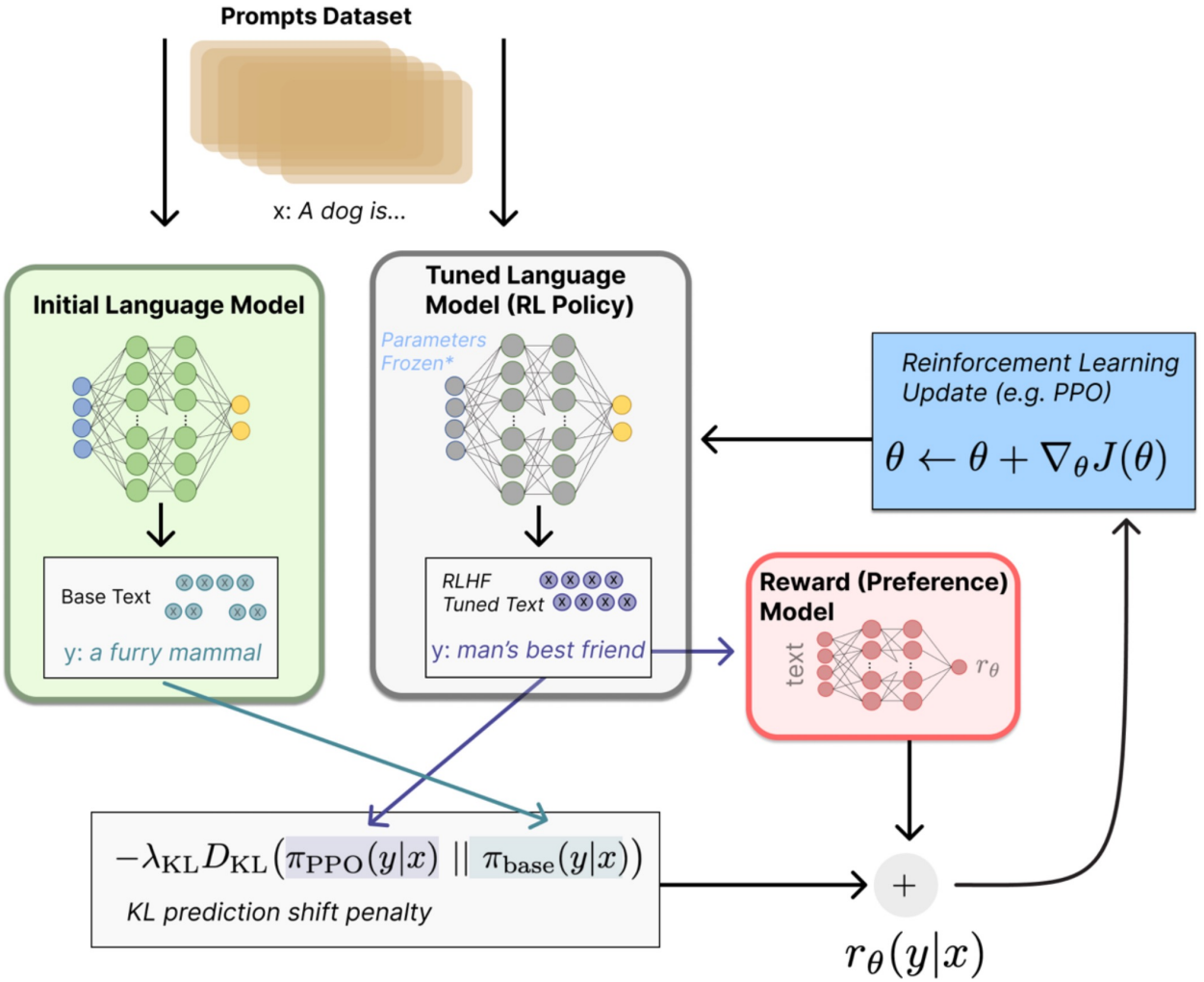
- Human annotators are used to rank the generated text outputs from the LM.
  - One may initially think that humans should apply a scalar score directly to each piece of text in order to generate a reward model, but this is difficult to do in practice.
  - The differing values of humans cause these scores to be uncalibrated and noisy. Instead, rankings are used to compare the outputs of multiple models and create a much better regularized dataset.
- There are multiple methods for ranking the text.
  - One method that has been successful is to have users compare generated text from two language models conditioned on the same prompt.
  - By comparing model outputs in head-to-head matchups, an Elo system can be used to generate a ranking of the models and outputs relative to each-other.
  - These different methods of ranking are normalized into a scalar reward signal for training
- At this point in the RLHF system, we have an initial language model that can be used to generate text and a preference model that takes in any text and assigns it a score of how well humans perceive it. Next, we use **reinforcement learning (RL)** to optimize the original language model with respect to the reward model.

Reinforcement Learning

- **Policy** is a language model that takes in a prompt and returns a sequence of text (or just probability distributions over text).
- The **action space** of this policy is all the tokens corresponding to the vocabulary of the language model (often on the order of 50k tokens) and
- the **observation space** is the possible input token sequences, which is also quite large (size of vocabulary ^ number of input tokens).
- The **reward function** is a combination of the preference model and a constraint on policy shift.
- Fine-tuning some or all of the parameters of a **copy of the initial LM** with a policy-gradient RL algorithm, Proximal Policy Optimization (PPO)
- Parameters of the LM are frozen because fine-tuning an entire 10B or 100B+ parameter model is prohibitively expensive (for more, see Low-Rank Adaptation (LoRA) for LMs or the Sparrow LM from DeepMind)

https://huggingface.co/blog/rlhf

Reinforcement Learning



https://huggingface.co/blog/rlhf

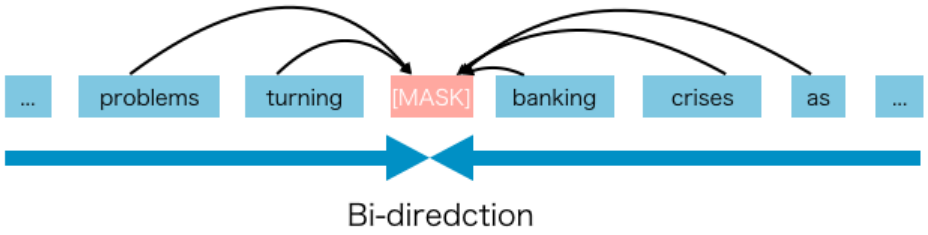
Intelligent Agents

1d-CNNs LSTMs ELMo Transformers BERT GPT

Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

AE and AR

Autoencoding(AE) Language Modeling:



Can be implemented with self-attention

The AE language model aims to reconstruct the original data from **corrupted input**.

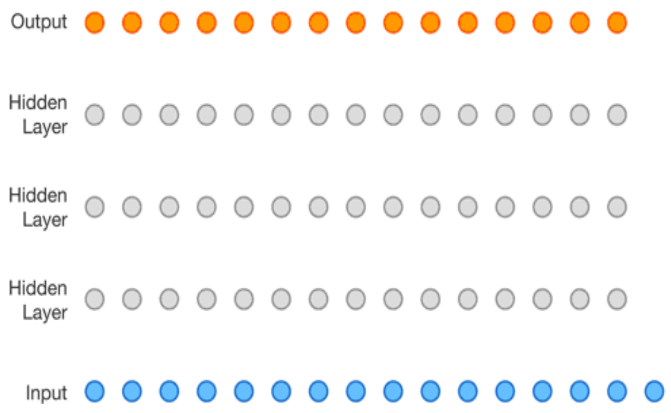
Corrupted input: The corrupted input means we use [MASK] to replace the original token

Example:

BERT

AE and AR

Autoregressive (AR) language modeling:



Can also be implemented with attention (GPT, not ELMO)

An autoregressive model's output h_t at time t depends on not just x_t, but also all x_s from previous time steps.

given a text sequence x = (x1, · · · , xT), AR language modeling factorizes the likelihood into a forward product. $p(x) = \prod p(xt \mid x<t)$

Examples:

GPT , ELMO

Summarization

- Extractive Text Summarization
  - The **traditional** method with the main objective to identify the significant sentences of the text and add them to the summary. Note that the summary obtained contains **exact sentences from the original text data.**
  - Can be done with encoder (e.g., BERT)
- Abstractive Text Summarization
  - The **advanced** method, with the approach to identify the important sections, interpret the context and reproduce the text in a new way. This ensures that the core information is conveyed through the shortest text possible. Note that here, the sentences, in summary, **are generated by the model, not just extracted from the original text data.**
  - Need Decoder (e.g., GPT-x, PEGASUS)

https://medium.com/analytics-vidhya/text-summarization-using-bert-gpt2-xlnet-5ee80608e961

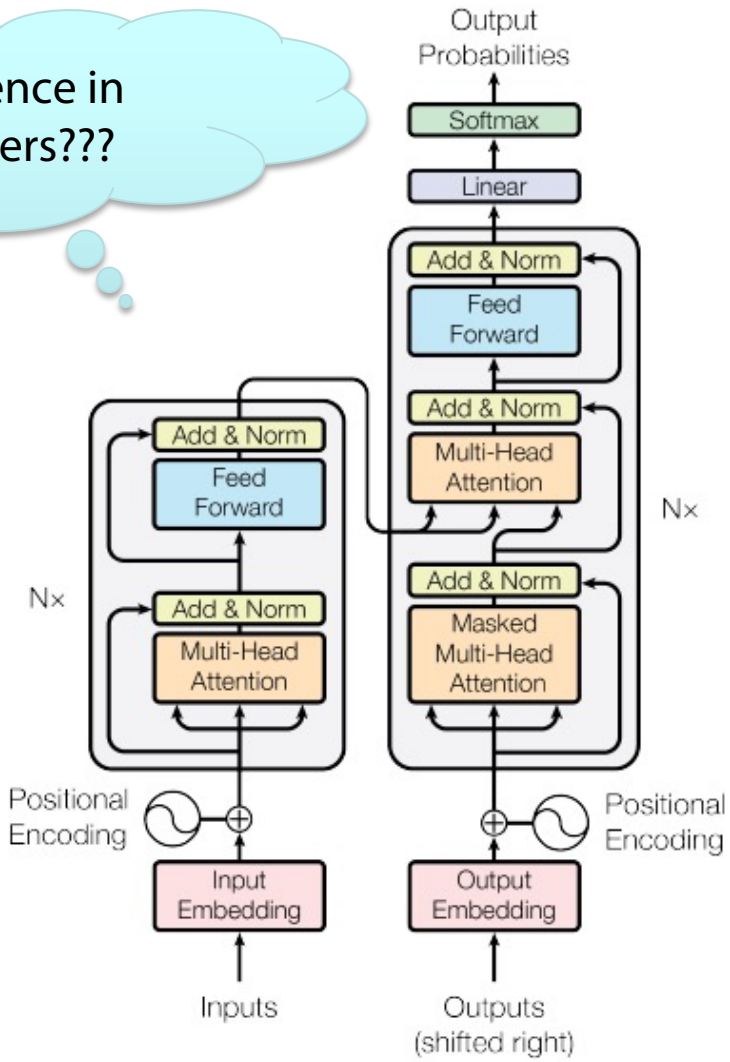
https://ai.googleblog.com/2020/06/pegasus-state-of-art-model-for.html

Summarization with attention-based AE + AR

Document Set

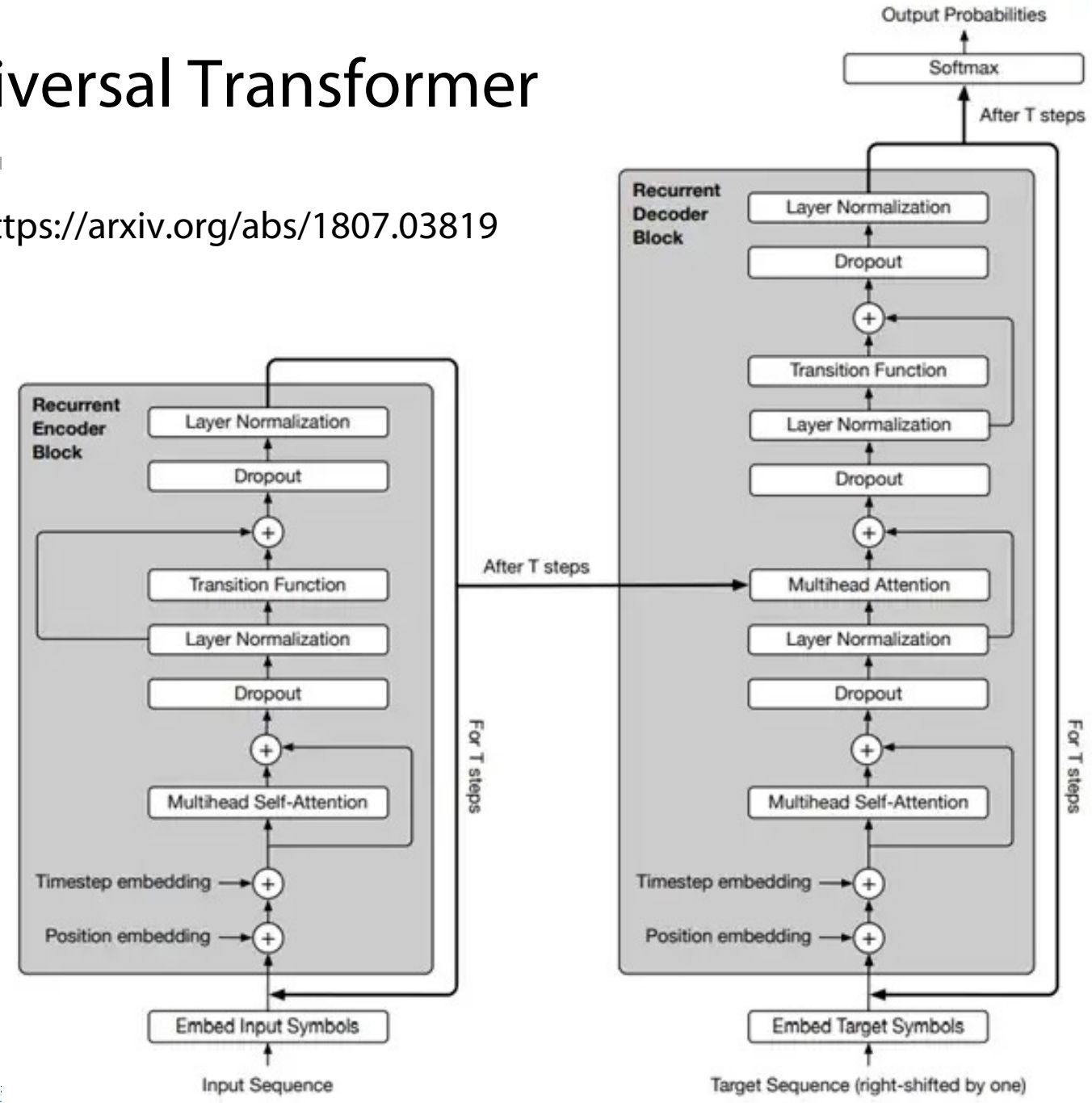
No recurrence in transformers???

Summarizer

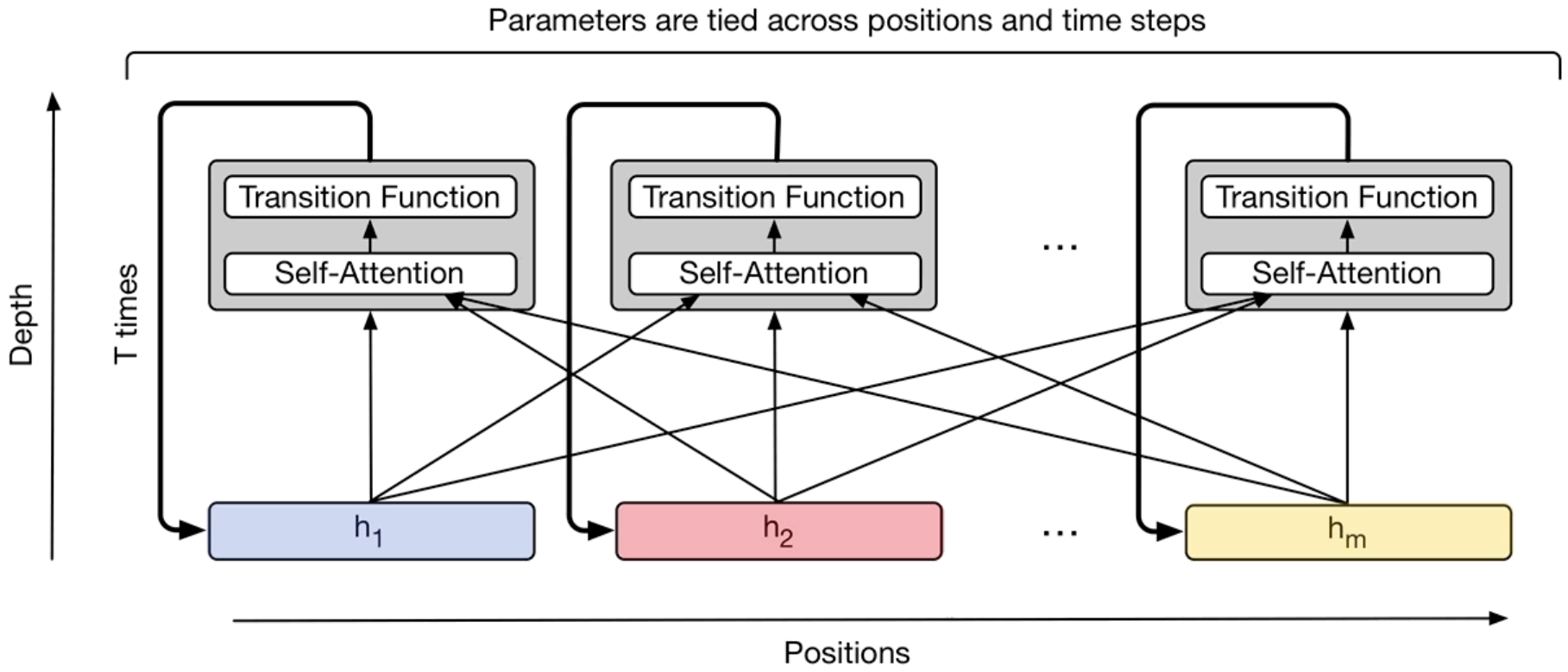


Universal Transformer

https://arxiv.org/abs/1807.03819



Universal Transformer



https://ai.googleblog.com/2018/08/moving-beyond-translation-with.html

THE COST OF TRAINING NLP MODELS
A CONCISE OVERVIEW

Or Sharir
AI21 Labs
ors@ai21.com

Barak Peleg
AI21 Labs
barakp@ai21.com

Yoav Shoham
AI21 Labs
yoavs@ai21.com

April 2020

http://arxiv.org/abs/2004.08900

Costs: Not for the faint hearted

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

Tricks: Subtoken Encoding

Byte Pair Encoding (BPE)

Word embedding sometimes is too high level, pure character embedding too low level. For example, if we have learned

old older oldest

We might also wish the computer to infer

smart smarter smartest

But at the whole word level, this might not be so direct. Thus, the idea is to break the words up into pieces like er, est, and embed frequent fragments of words.

GPT adapts this BPE scheme.

Tricks: Subtoken Encoding

Byte Pair Encoding (BPE)

GPT uses BPE scheme. The subwords are calculated by:

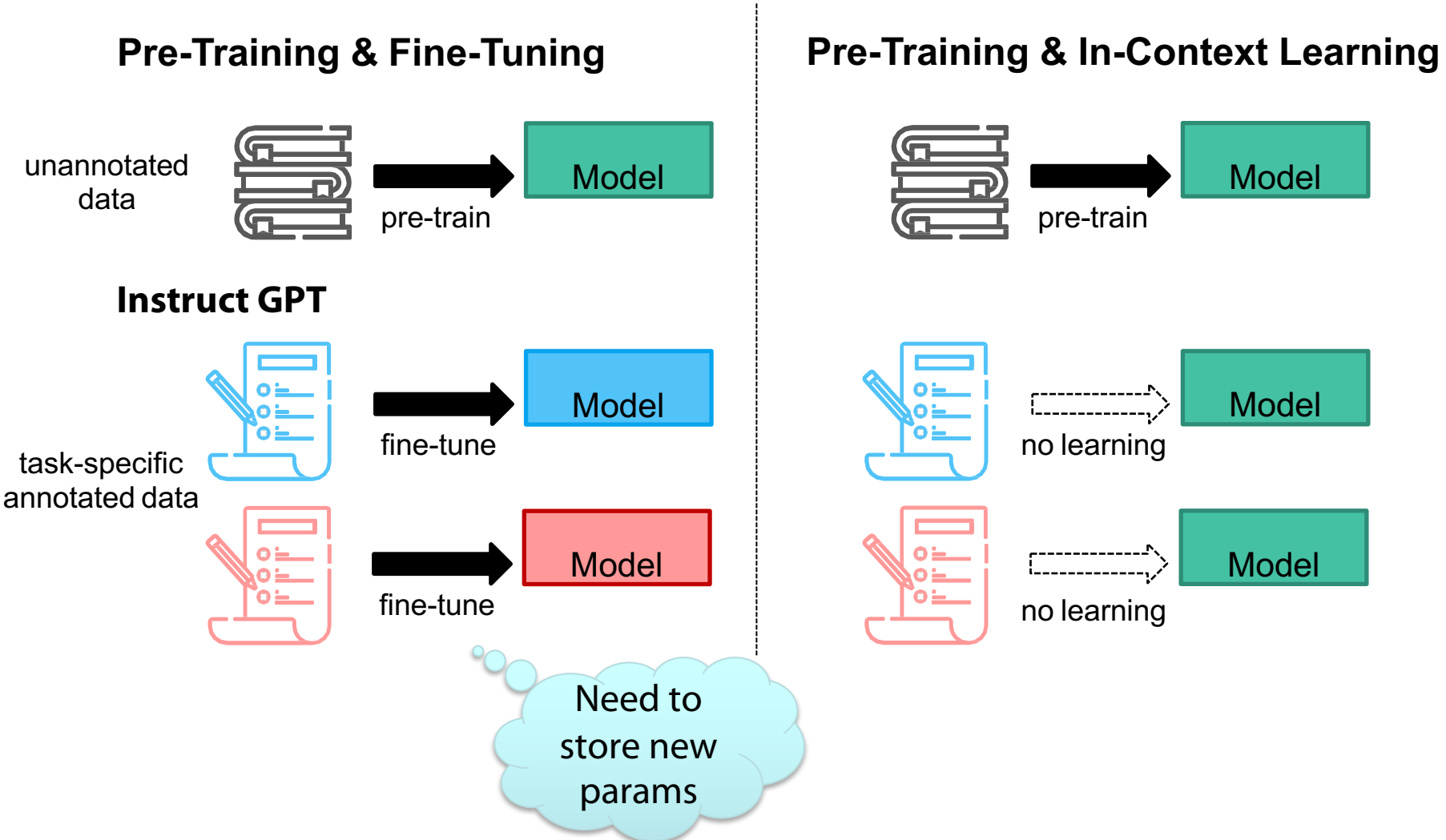
1. Split word to sequence of characters (add </w> char)
2. Joining the highest frequency pattern.
3. Keep doing step 2, until it hits the pre-defined maximum number of sub-words or iterations.

Example (5, 2, 6, 3 are number of occurrences)

{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3 }
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w es t </w>': 6, 'w i d es t </w>': 3 }
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w est </w>': 6, 'w i d est </w>': 3 } ("est" freq. 9)
{'lo w </w>': 5, 'lo w e r </w>': 2, 'n e w est</w>': 6, 'w i d est</w>': 3 } ("lo" freq 7)
…..

Pretraining and Fine-tuning

Powerful Pre-Trained Model – GPT 3



The Amazing Power of Large Language Models

- Large scale leads to a particularly interesting emergent behavior called in-context learning
- With in-context learning, the text given to the model is a written description (optional) plus some examples. The last example is left unfinished for the model to complete
  - Use language models to learn tasks given only a few examples
  - Give the LLM a prompt that consists of a list of input-output pairs that demonstrate a task
  - At the end of the prompt, we append a test input and allow the LM to make a prediction just by conditioning on the prompt and predicting the next tokens

https://ai.stanford.edu/blog/understanding-incontext/

Example

Circulation revenue has increased by 5% in Finland. // Positive

Panostaja did not disclose the purchase price. // Neutral

Paying off the national debt will be extremely painful. // Negative

The company anticipated its operating profit to improve. // _____

LM

Circulation revenue has increased by 5% in Finland. // Finance

They defeated … in the NFC Championship Game. // Sports

Apple … development of in-house chips. // Tech

The company anticipated its operating profit to improve. // _____

LM

- In-context learning is competitive with models trained with much more labeled data and is state-of-the-art on LAMBADA (commonsense sentence completion) and TriviaQA (question answering)
- Other examples: Writing code from natural language descriptions, helping with app design mockups, and generalizing spreadsheet functions

https://ai.stanford.edu/blog/understanding-incontext/

GPT-3 “In-Context” Learning

- Zero-Shot

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

- One-Shot

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

- Few-Shot

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

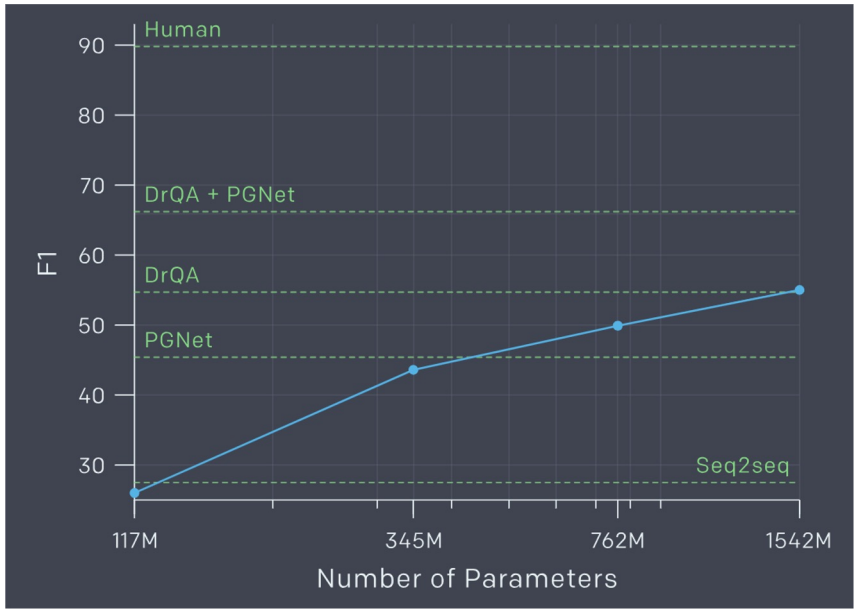
- Traditional Fine-Tuning

One-shot or Few-shot Learning? GPT-3 175B

- Reading Comprehension

$d_1, d_2, \cdots, d_N,$
"Q:", $q_1, q_2, \cdots, q_N,$
"A:"

CoQA



- Summarization $d_1, d_2, \cdots, d_N$,"TL;DR:"

TL;DR=Too long; didn't read.

- Translation

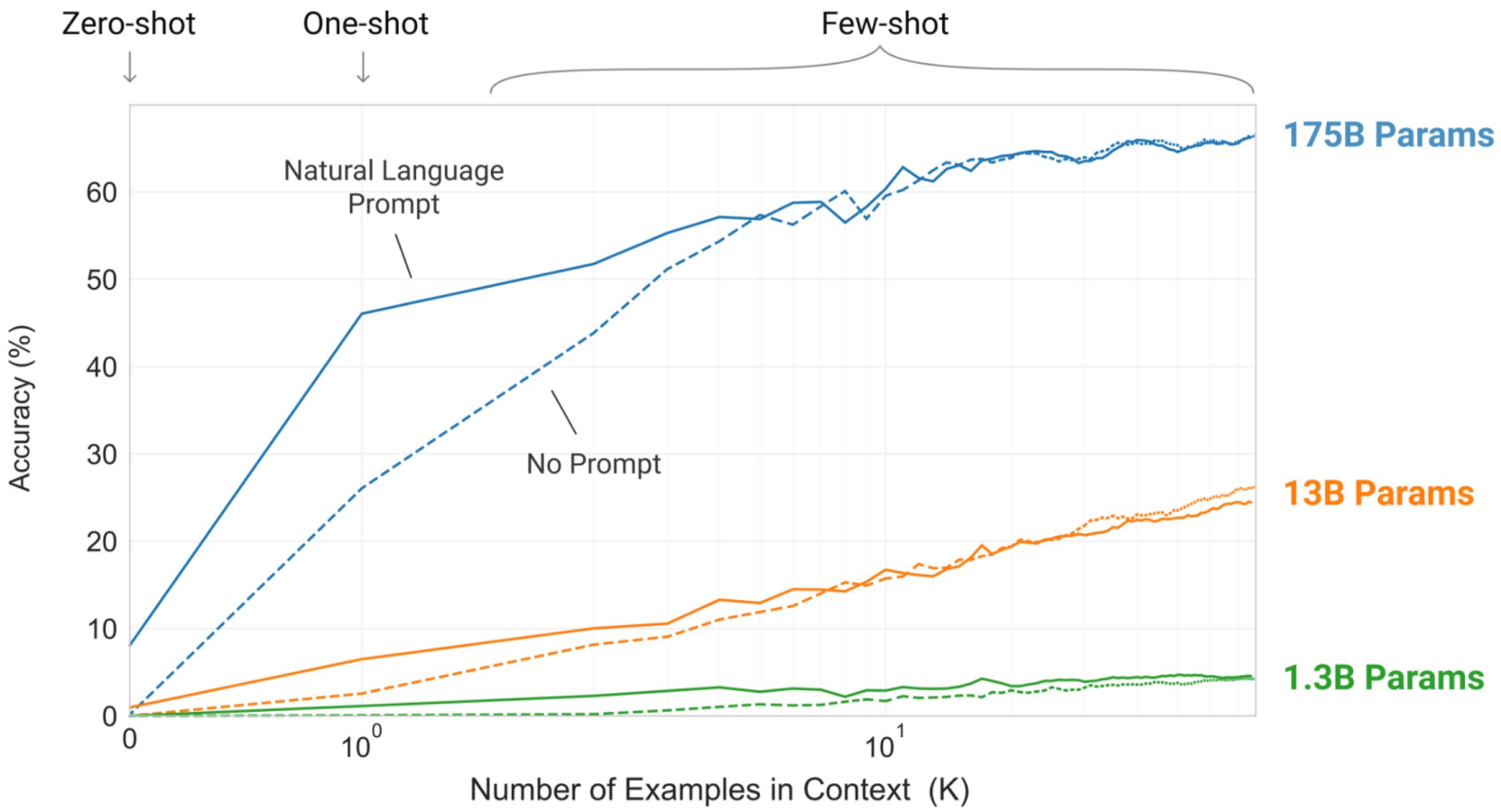
English sentence 1	=	French sentence 1
English sentence 2	=	French sentence 2
English sentence 3	=	

Tom B. Brown et al.
Language Models are Few-Shot Learners.
https://arxiv.org/abs/2005.14165.
2020.

In-context learning: Analysis

- In-context learning describes a different paradigm of “learning”
- where the model is fed input normally as if it were a black box,
- and the input to the model describes a new task with some possible examples
- while the resulting output of the model reflects that new task as if the model had “learned”

Evaluation


Zero-shot
One-shot
Few-shot
Natural Language Prompt
No Prompt
175B Params
13B Params
1.3B Params
Accuracy (%)
0
10
20
30
40
50
60
0
10^0
10^1
Number of Examples in Context (K)

Understanding context learning

- An Explanation of In-context Learning as Implicit Bayesian Inference
  - https://arxiv.org/abs/2111.02080
- Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?
  - https://arxiv.org/abs/2202.12837

- Can in-context learning help us with dialogues?

ChatGPT

- Interactive, conversational model
- Part of GPT-3.5 family
  - fine-tuned mostly on programming code
- ChatGPT
  - is a sibling model to InstructGPT
    - ChatGPT is similar but not identical
    - slight differences in the data collection setup
  - a fine-tuned version of GPT-3.5 that's essentially a general-purpose chatbot
- Dialogue format of ChatGPT makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests

Similar Approaches

- OPT (Meta, LLM)
  - https://ai.facebook.com/blog/democratizing-access-to-large-scale-language-models-with-opt-175b/
- Galactica (Meta, LLM for science)
  - https://galactica.org/
- Pegasus (Google, LLM, text summarization)
  - https://ai.googleblog.com/2020/06/pegasus-state-of-art-model-for.html

Back to Agents

- Why is few-shot learning important?
- Agents can use pretrained model
- Few-shot learning for specifying dedicated tasks!

- From language models to general intelligence?

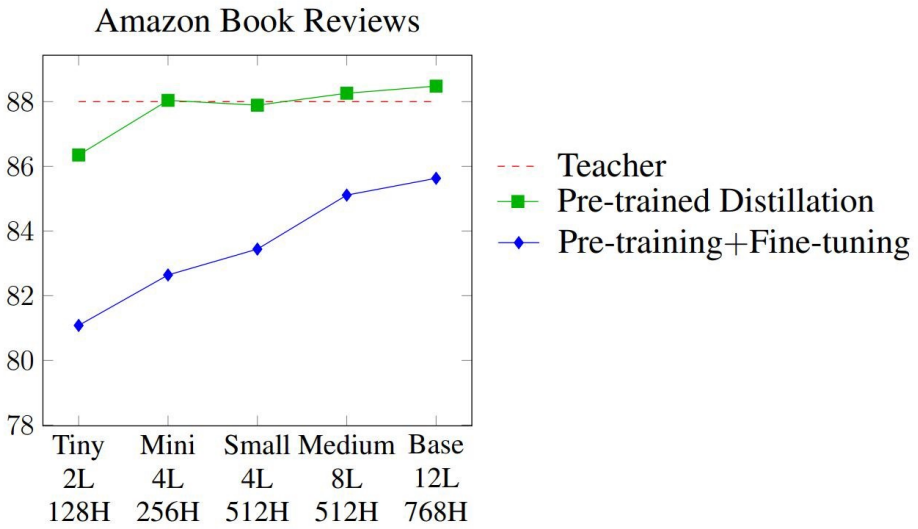
- Implementing agents might be difficult
- Is there any hope that ordinary people can get access to GPT-3 / ChatGPT for fine-tuning?

Distillation

- A.k.a., model compression
- Idea has been around for a long time:
  - *Model Compression* (Bucila et al, 2006)
  - *Distilling the Knowledge in a Neural Network* (Hinton et al, 2015)
- Simple technique:
  - Train "Teacher": Use SOTA pre-training + fine-tuning technique to train model with maximum accuracy
  - Label a large amount of unlabeled input examples with Teacher
  - Train "Student": Much smaller model (e.g., 50x smaller) which is trained to mimic Teacher output
  - Student objective is typically Mean Square Error or Cross Entropy

Distillation

- Example distillation results
  - 50k labeled examples, 8M unlabeled examples



Well-Read Students Learn Better: On the Importance of Pre-training Compact Models (Turc et al, 2020)

- Distillation works *much* better than pre-training + fine-tuning with smaller model

Why does distillation work so well?

A hypothesis:

- Finetuning mostly just picks up and tweaks existing latent features
- This requires an oversized model, because only a subset of the features are useful for any given task
- Distillation allows the model to only focus on those features
- Supporting evidence: Simple self-distillation of a small model (e.g., distilling a smaller BERT model) doesn't work very well

GPT-J

- GPT-3 was released by OpenAI, has 175 billion parameters and is not openly available.
- GPT-J is a 6 billion parameter model released by Eleuther AI. The goal of the group is to democratize huge language models, so they released GPT-J and it is currently publicly available.

https://github.com/TheProtaganist/gpt-j

Summary: Large Language models

- Translating one language into another,
- Summarizing a long document into a brief highlight,
- Answering information-seeking questions
- Open-domain dialog
  - converse about any topic
  - dialog models should adhere to Responsible AI practices
  - avoid making factual statements that are not supported by external information sources.

https://ai.googleblog.com/2022/01/lamda-towards-safe-grounded-and-high.html