CS 307 Modeling and Learning in Data Science

Today’s goal

- Recall:
  - Image autoencoder
  - Simple word embeddings
  - Encoder decoder for word embeddings
  - Attention
  - Transformers
- BERT
- GPT

Sequence-to-sequence is everywhere!

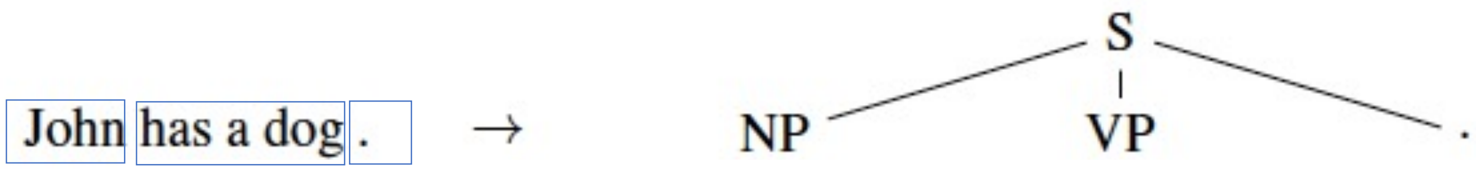
- Sequence-to-sequence is useful for *more than just MT*
- Many NLP tasks can be phrased as sequence-to-sequence:
  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)

Grammar as a Foreign Language

- Vinyals et al., 2015

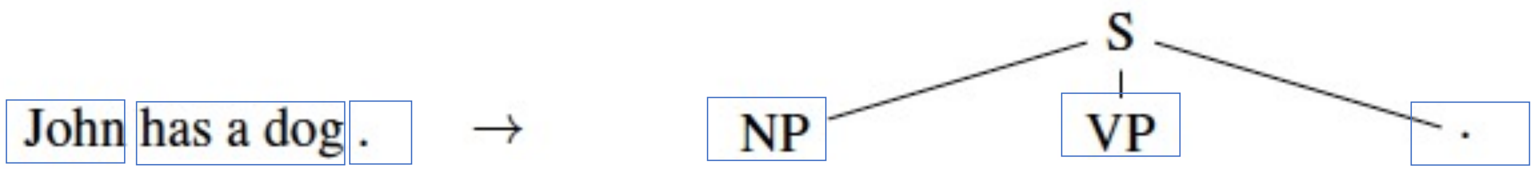
Grammar as a Foreign Language

Parsing tree



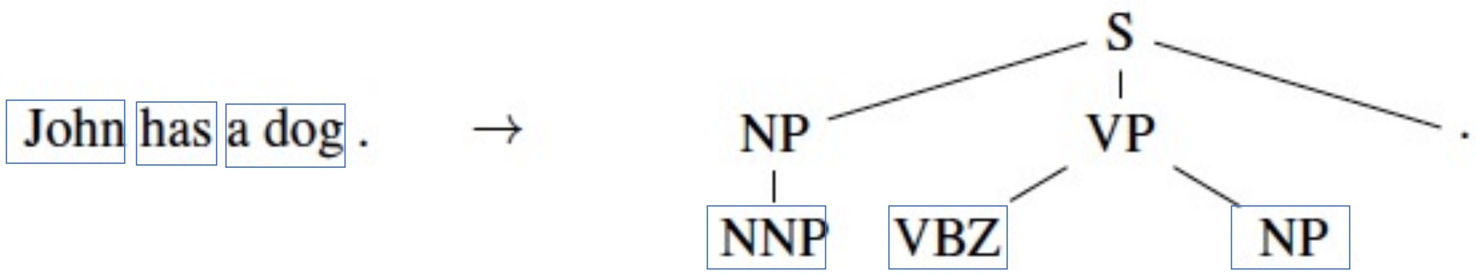
Grammar as a Foreign Language

Parsing tree



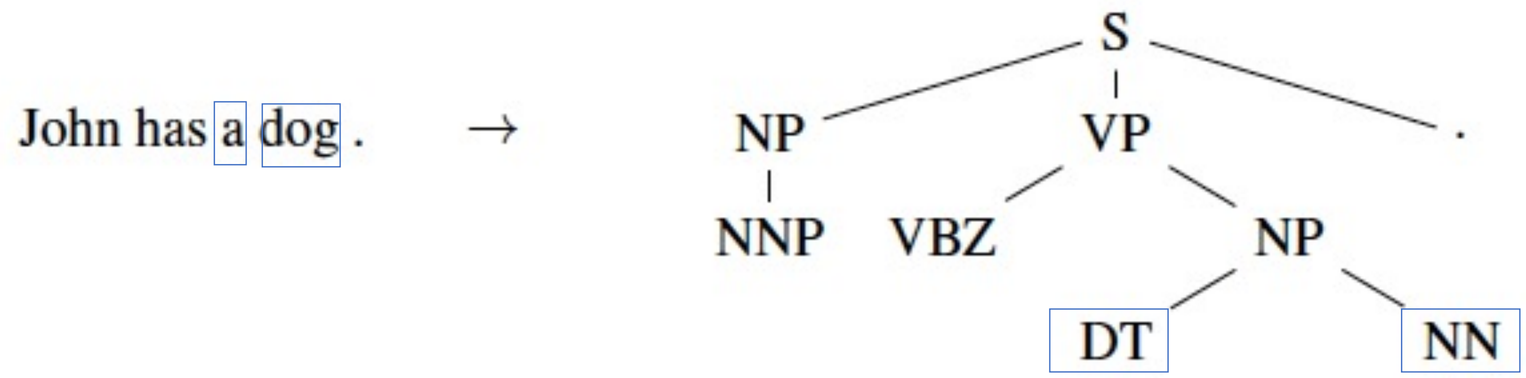
Grammar as a Foreign Language

Parsing tree



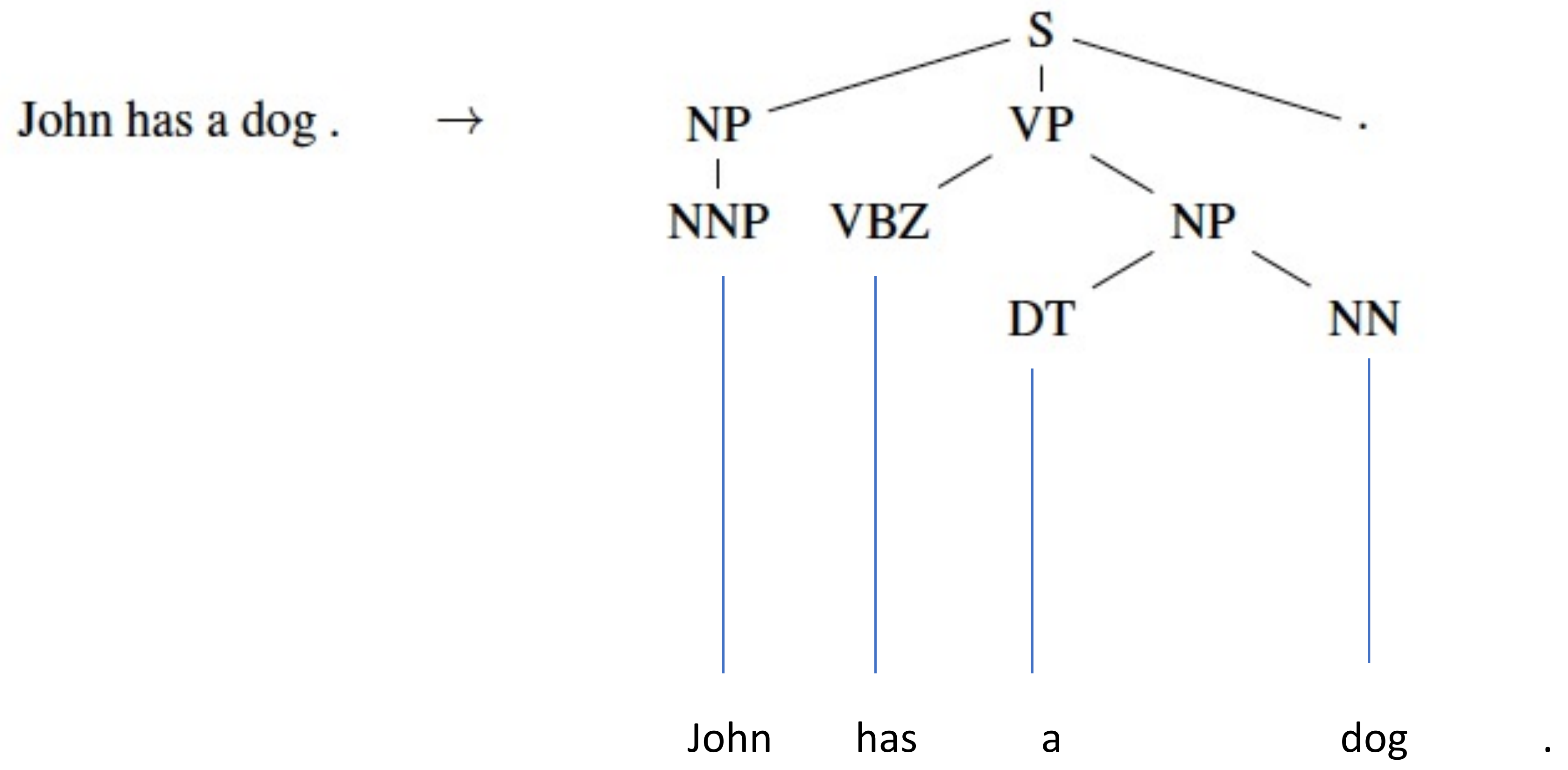
Grammar as a Foreign Language

Parsing tree



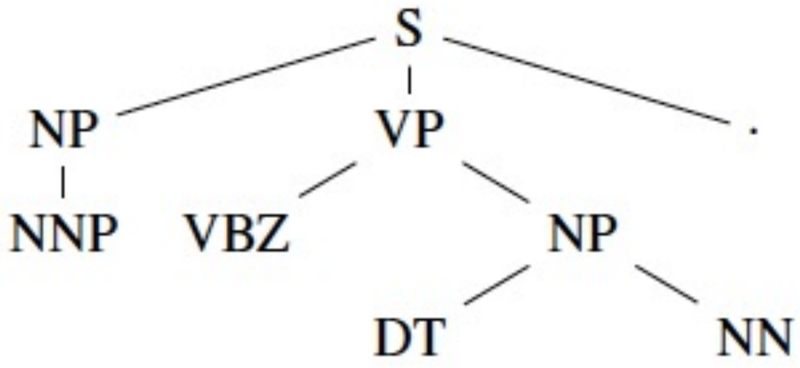
Grammar as a Foreign Language

Parsing tree



Grammar as a Foreign Language

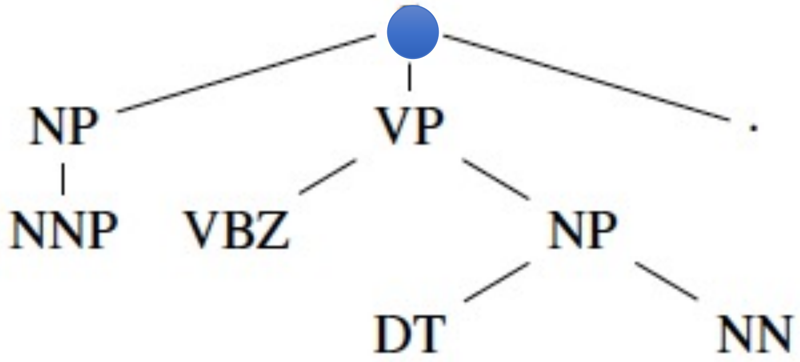
Converting tree to sequence



(S (NP NNP $)_{NP}$ (VP VBZ (NP DT NN $)_{NP}$ $)_{VP}$. $)_{S}$

Grammar as a Foreign Language

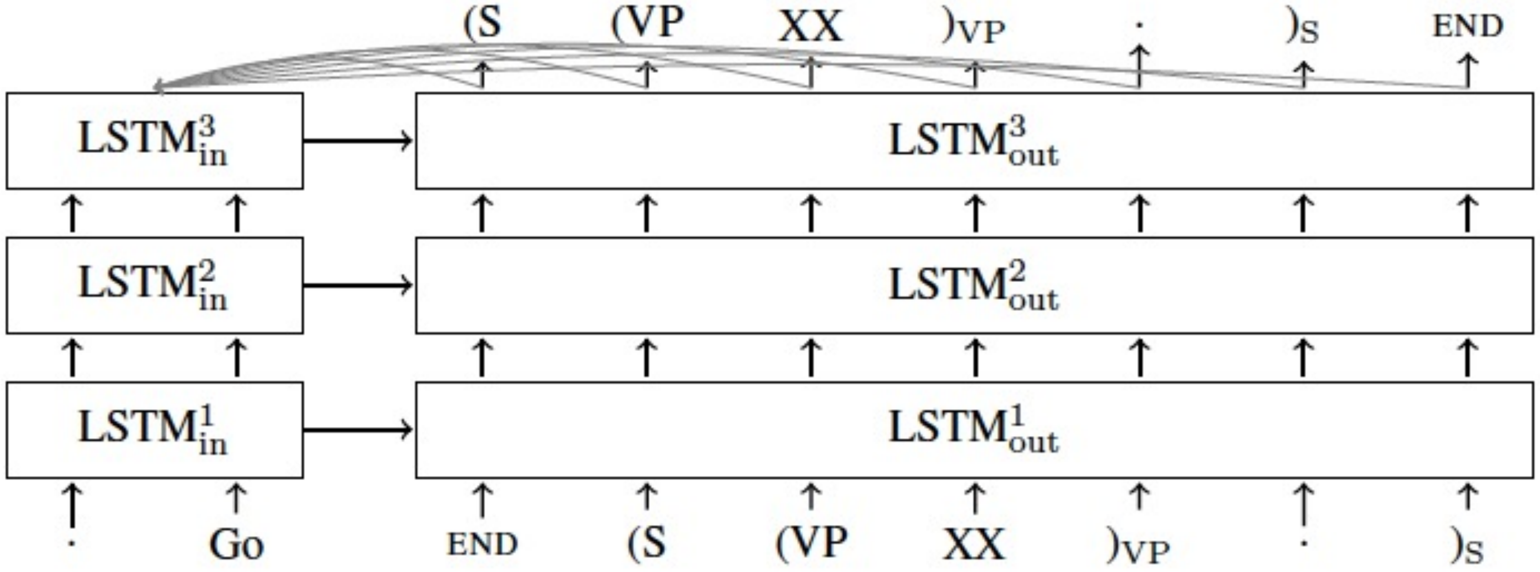
Converting tree to sequence



$($NP NNP $)_{NP}$ $($VP VBZ $($NP DT NN $)_{NP}$ $)_{VP}$. $)_{S}$

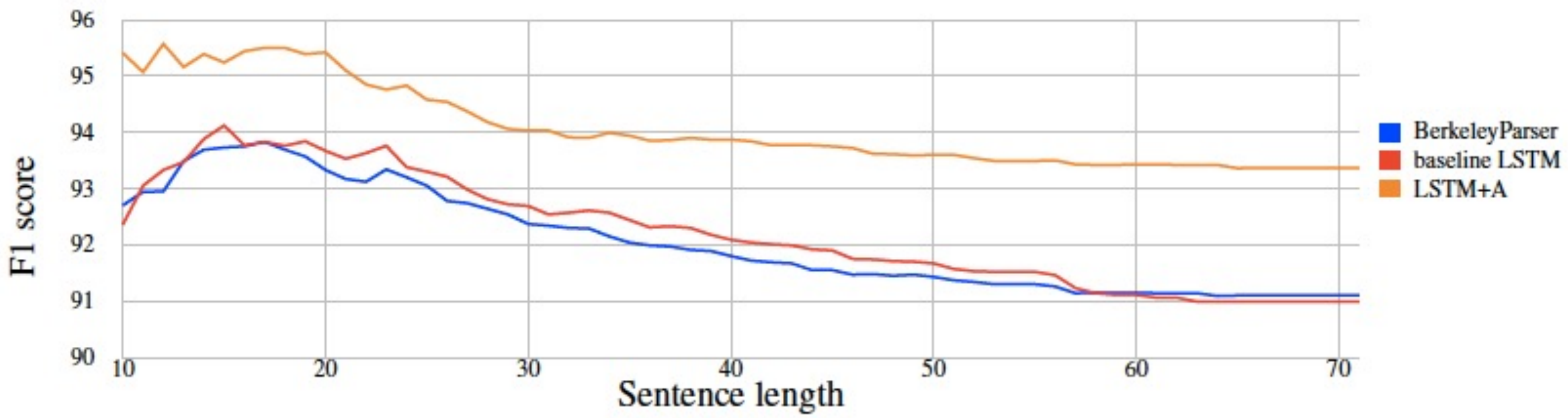
Grammar as a Foreign Language

Model



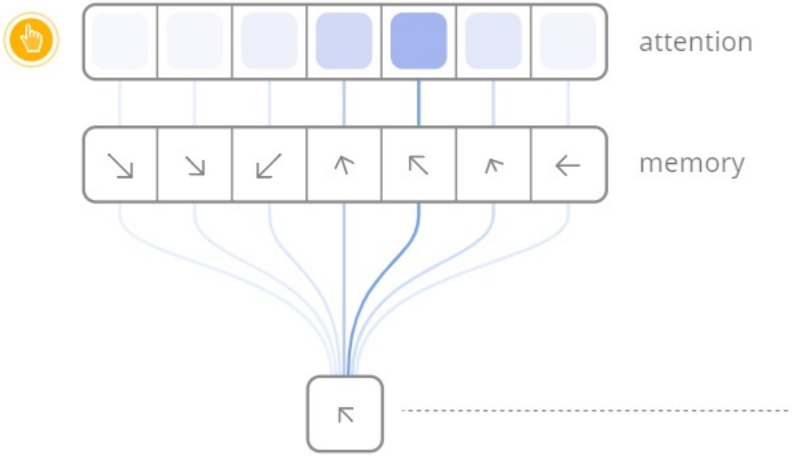
Grammar as a Foreign Language

Results

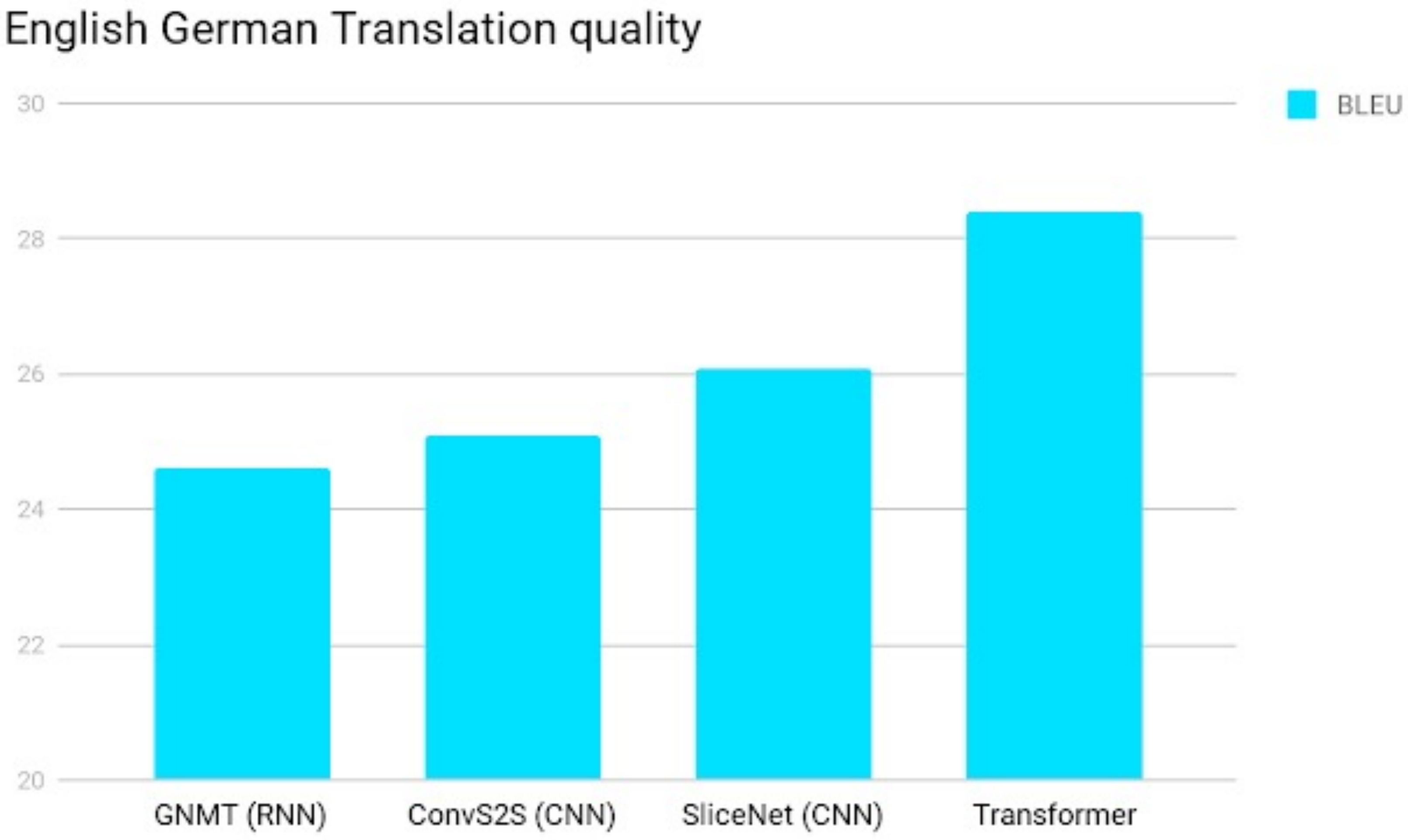


Attention is a *general* Deep Learning technique

- More general definition of attention
  - Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.
  - We sometimes say that the query *attends to* the values
  - For example, in the seq2seq + attention model, each decoder hidden state *attends to* the encoder hidden states



Attention is all you need? – Transformer Networks



Problems with RNNs = Motivation for Transformers

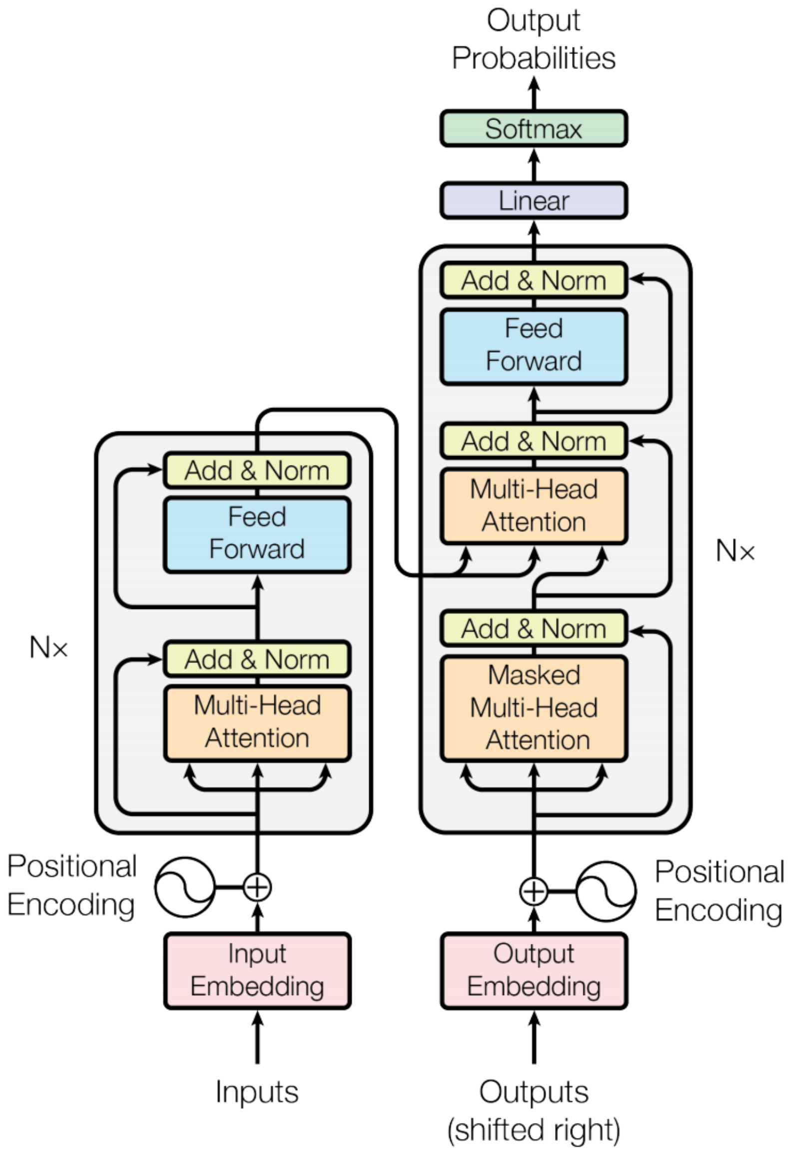
- Recurrent models typically factor computation along the symbol positions of the input and output sequences
  - Sequential computation prevents parallelization
  - Critical at longer sequence lengths, as memory constraints limit batching across examples

- Despite advanced RNNs like LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for codependent computation between states grows with sequence

- But if attention gives us access to any state… maybe we don’t need the RNN?

Transformer Overview Recall

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



Transformer Basics

- Let’s define the basic building blocks of transformer networks first: new attention layers!

Dot-Product Attention (Extending our previous def.)

- Inputs: a query q and a set of key-value (k-v) pairs to an output
  - Query, keys, values, and output are all vectors
- Output is weighted sum of values, where
  - Weight of each value is computed by an inner product of query and corresponding key
  - Queries and keys have same dimensionality

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

Dot-Product Attention – Matrix notation

- When we have multiple queries q, we stack them in a matrix Q:

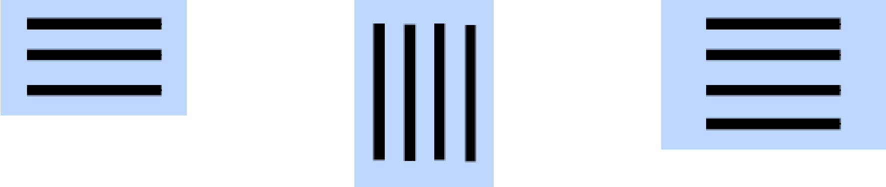
$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- Becomes:

$$A(Q, K, V) = softmax(QK^T)V$$

$[|Q| \text{ x } d_k] \text{ x } [d_k \text{ x } |K|] \text{ x } [|K| \text{ x } d_v]$

softmax
row-wise



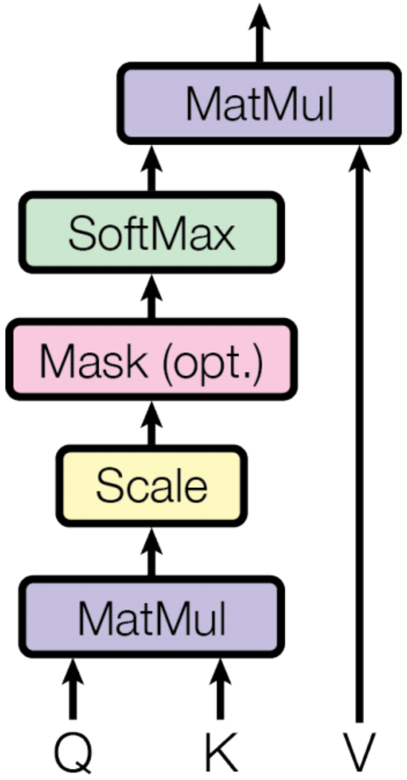
$= [|Q| \text{ x } d_v]$

Scaled Dot-Product Attention

- Problem: As d_k gets large, the variance of $q^T k$ increases → some values inside the softmax get large → the softmax gets very peaked → hence its gradient gets smaller.

- Solution: Scale by length of query/key vectors:

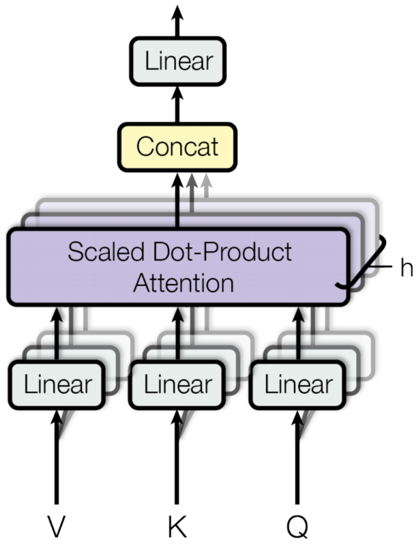
$$A(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



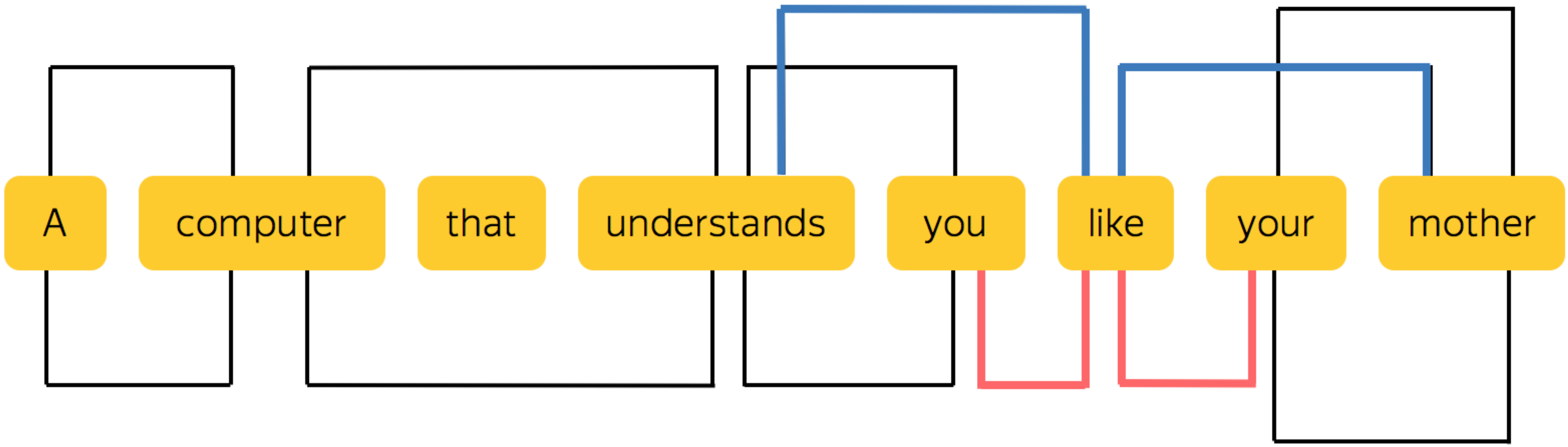
Self-attention and Multi-head attention

- The input word vectors could be the queries, keys and values
  - In other words: the word vectors themselves select each other
  - Word vector stack = Q = K = V
- Problem: Only one way for words to interact with one-another
- Solution: Multi-head attention
  - First map Q, K, V into h many lower dimensional spaces via W matrices
  - Then apply attention, then concatenate outputs and pipe through linear layer

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

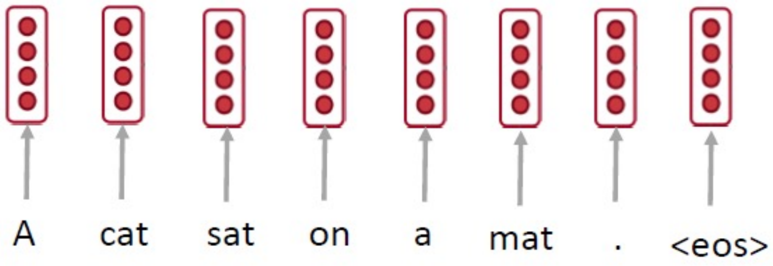


Self-Attention



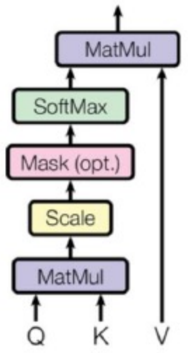
(example and picture from David Talbot)

Self-attention: A Running Example

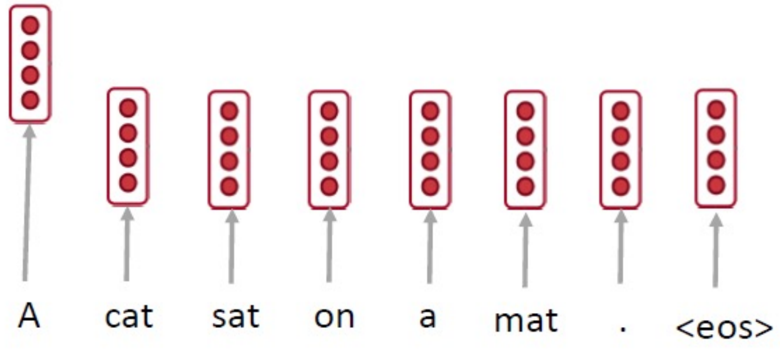


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention

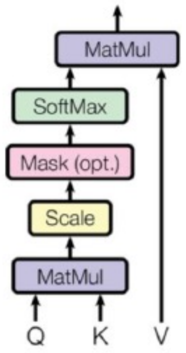


Self-attention: A Running Example



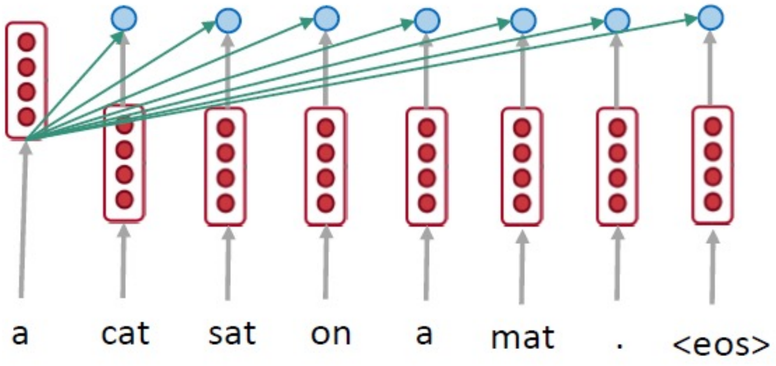
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

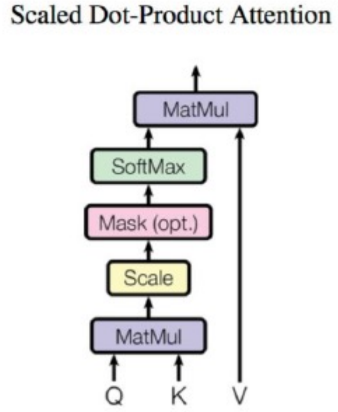
Scaled Dot-Product Attention



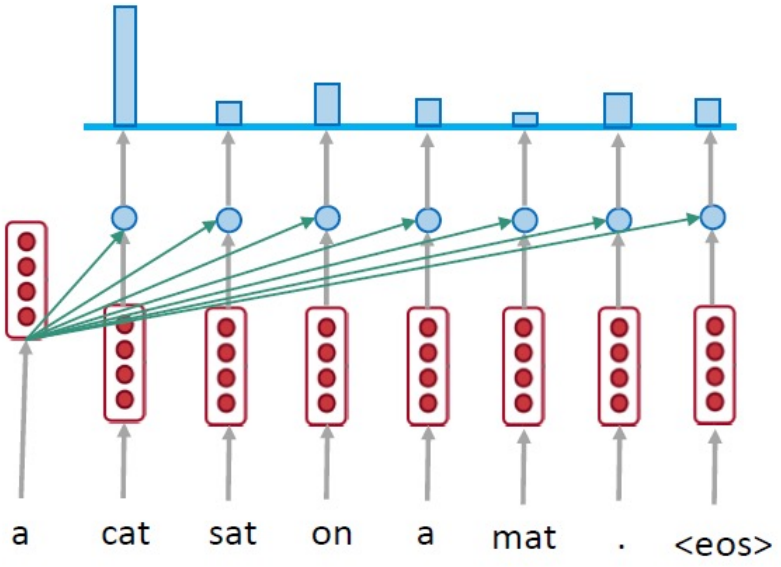
Self-attention: A Running Example

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



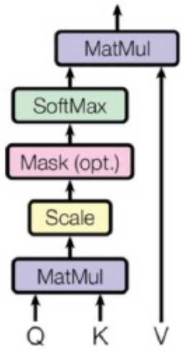


Self-attention: A Running Example

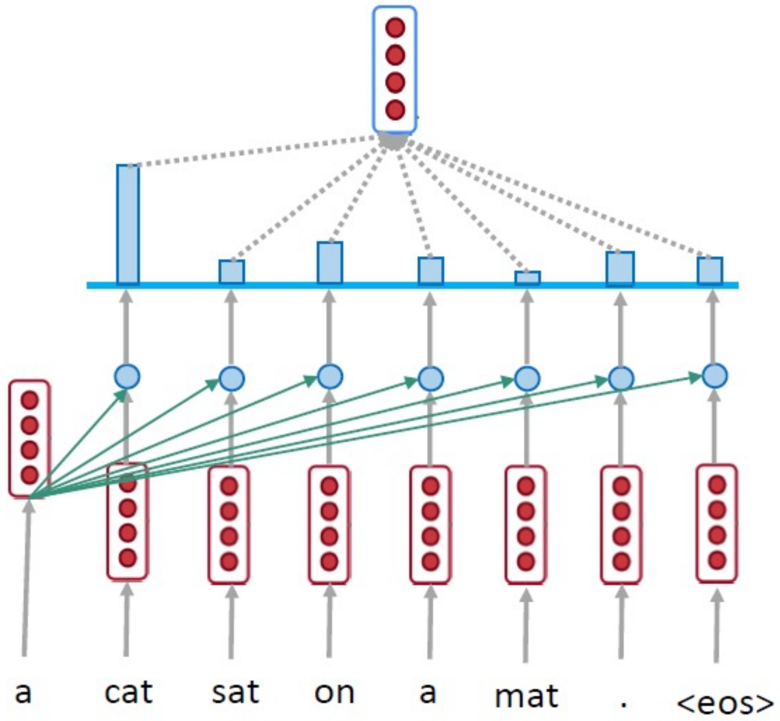


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention

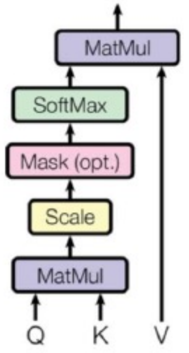


Self-attention: A Running Example

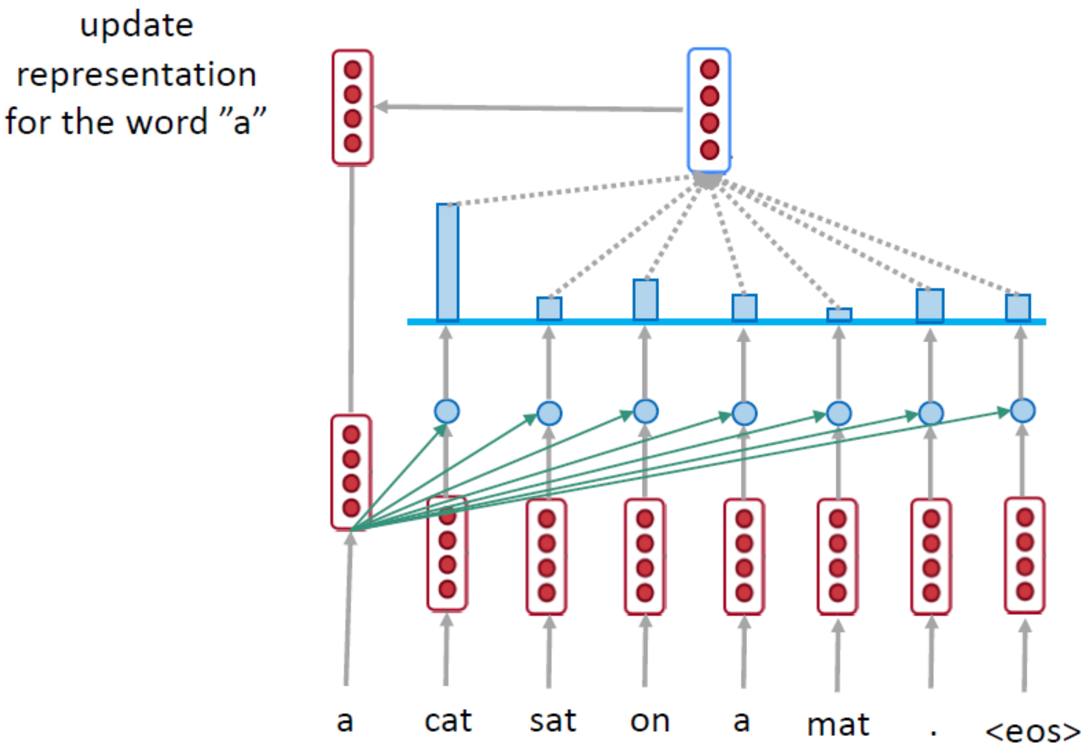


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention

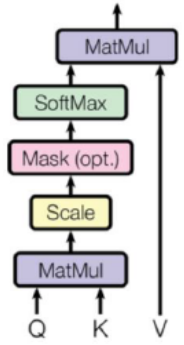


Self-attention: A Running Example

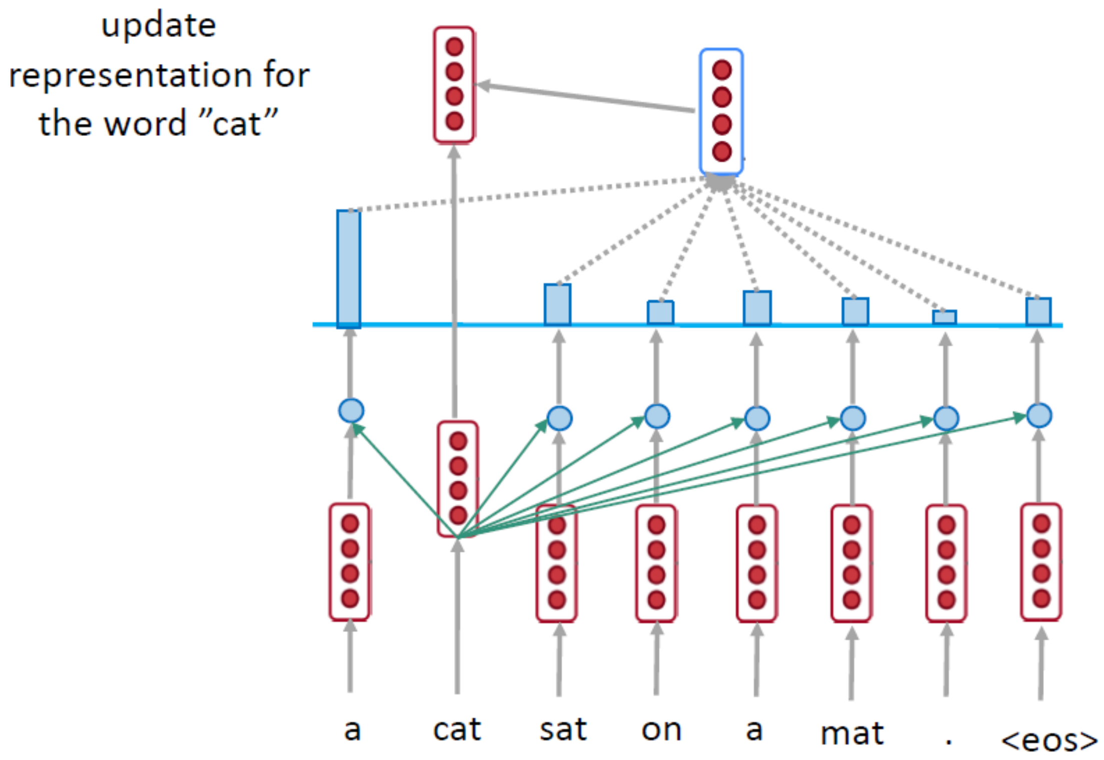


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

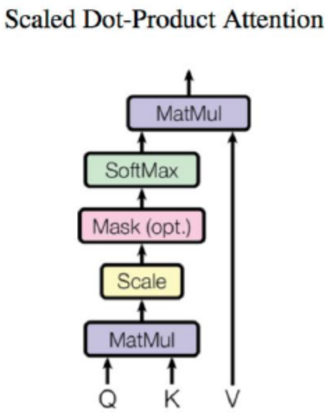
Scaled Dot-Product Attention



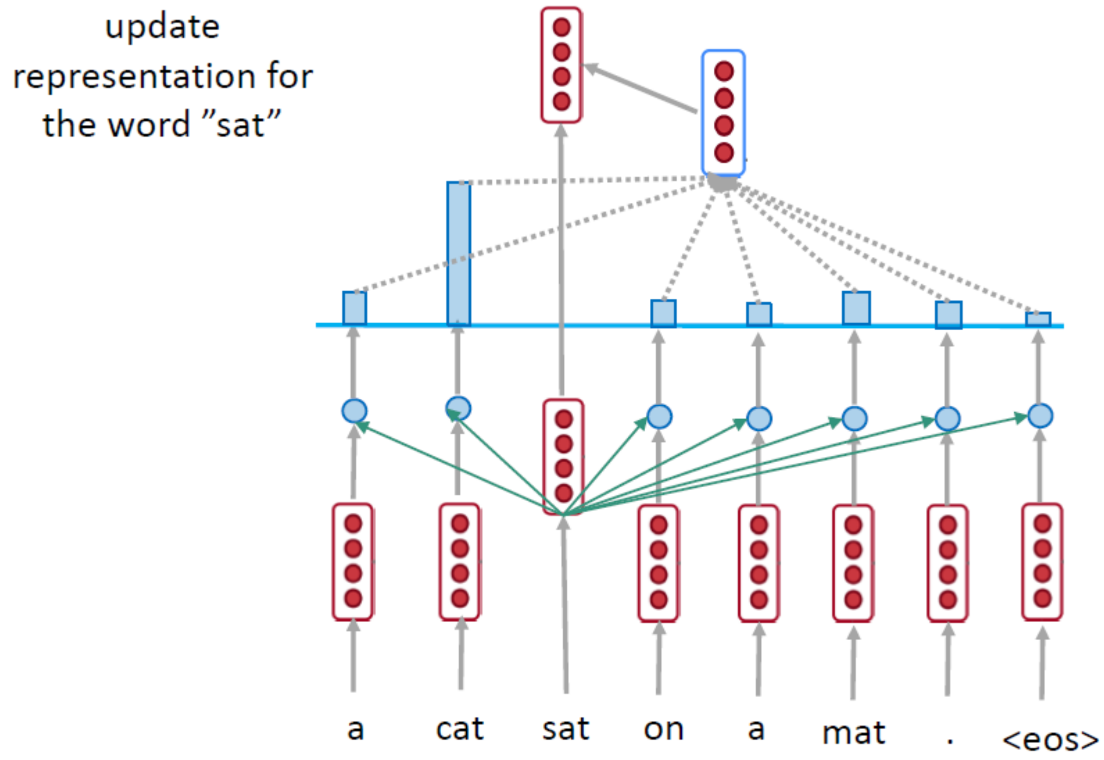
Self-attention: A Running Example



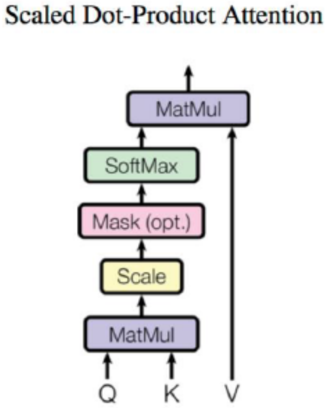
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Self-attention: A Running Example

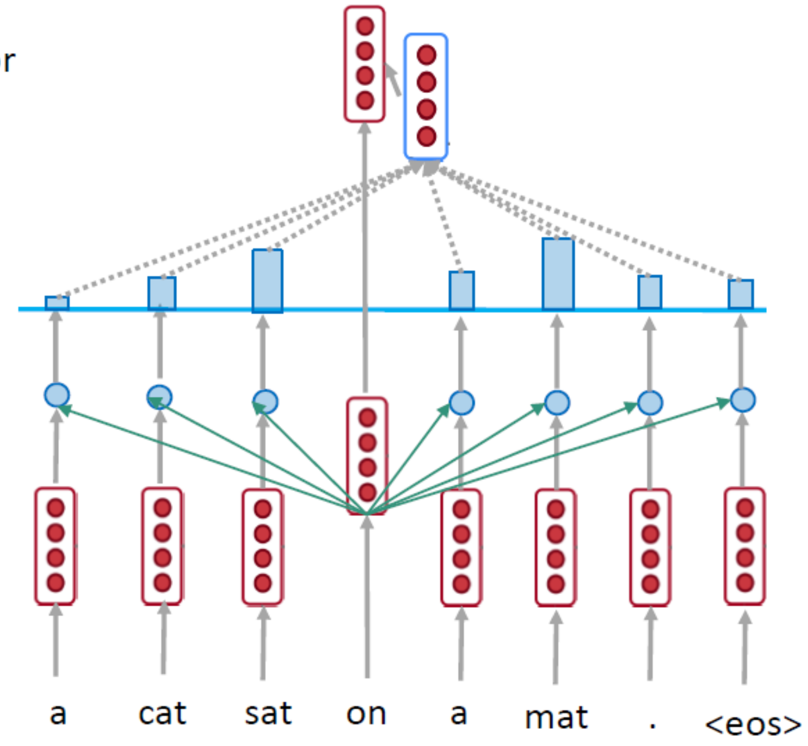


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



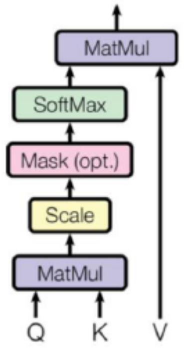
Self-attention: A Running Example

update
representation for
the word ”on”



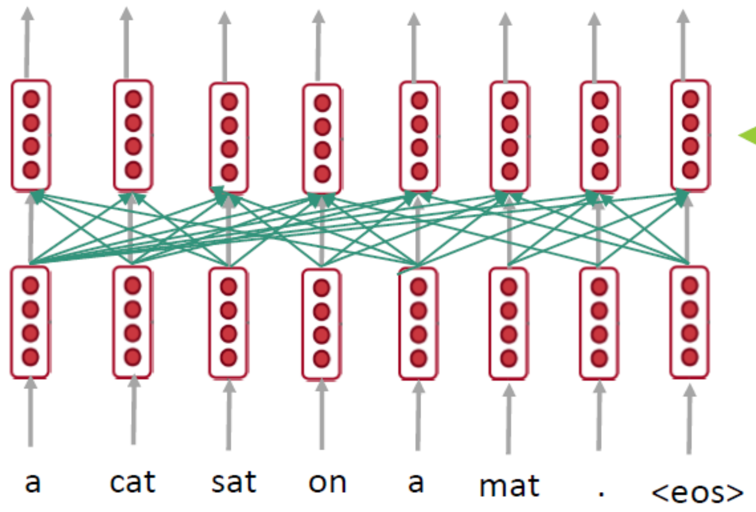
$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention



Self-attention: A Running Example

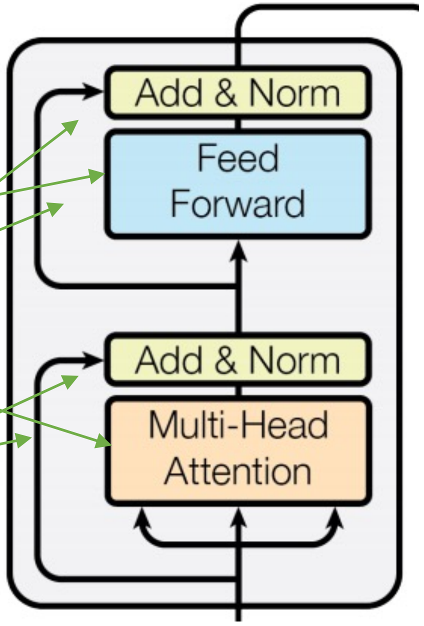
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Updated representations
for each word
(one layer)

Complete transformer block

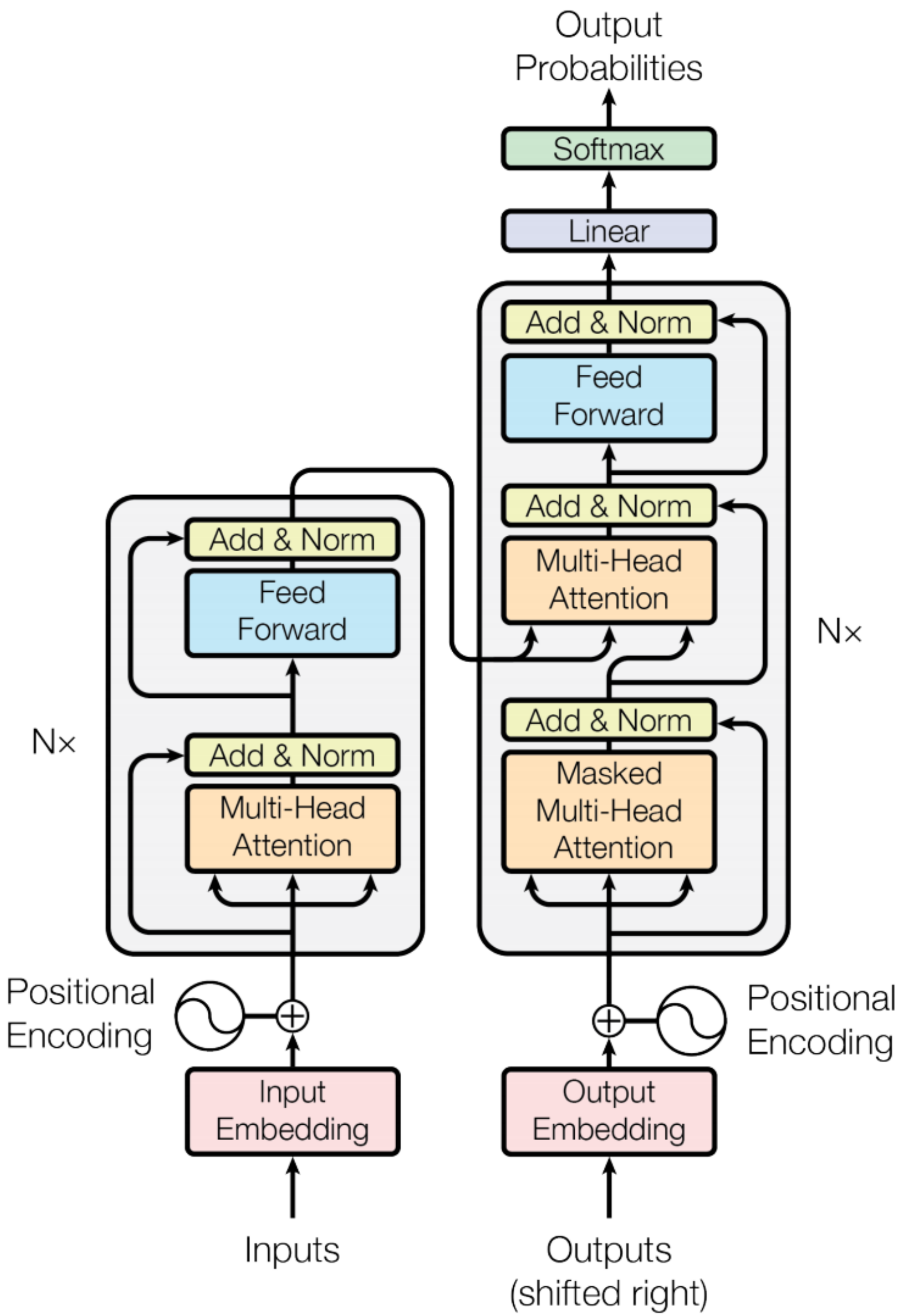
- Each block has two "sublayers"
  - 1. Multihead attention
  - 2. 2 layer feed-forward Nnet (with relu)



- Each of these two steps also has:
  - Residual (short-circuit) connection and LayerNorm:
  - LayerNorm(x + Sublayer(x))
    - Layernorm changes input to have mean 0 and variance 1, per layer and per training point (and adds two more parameters)

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

Blocks are repeated N=6 times

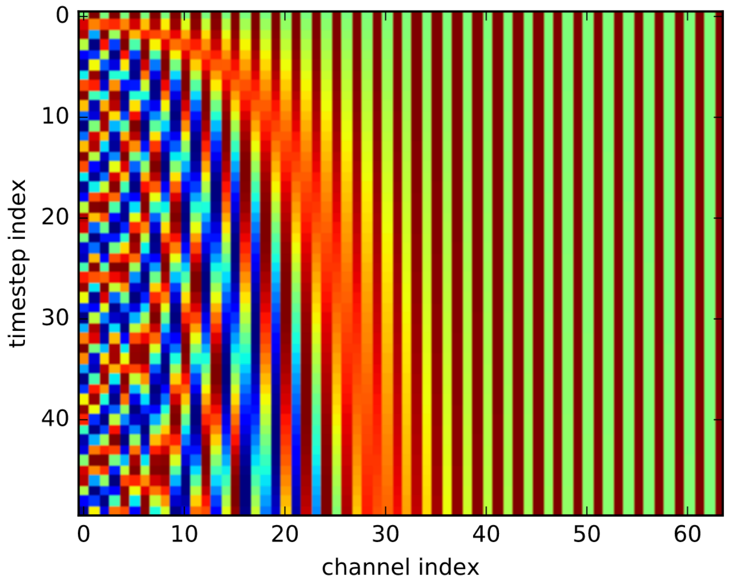


Encoder Input

- Actual word representations are byte-pair encodings
  - Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016.
- Added is a positional encoding so same words at different locations have different overall representations:

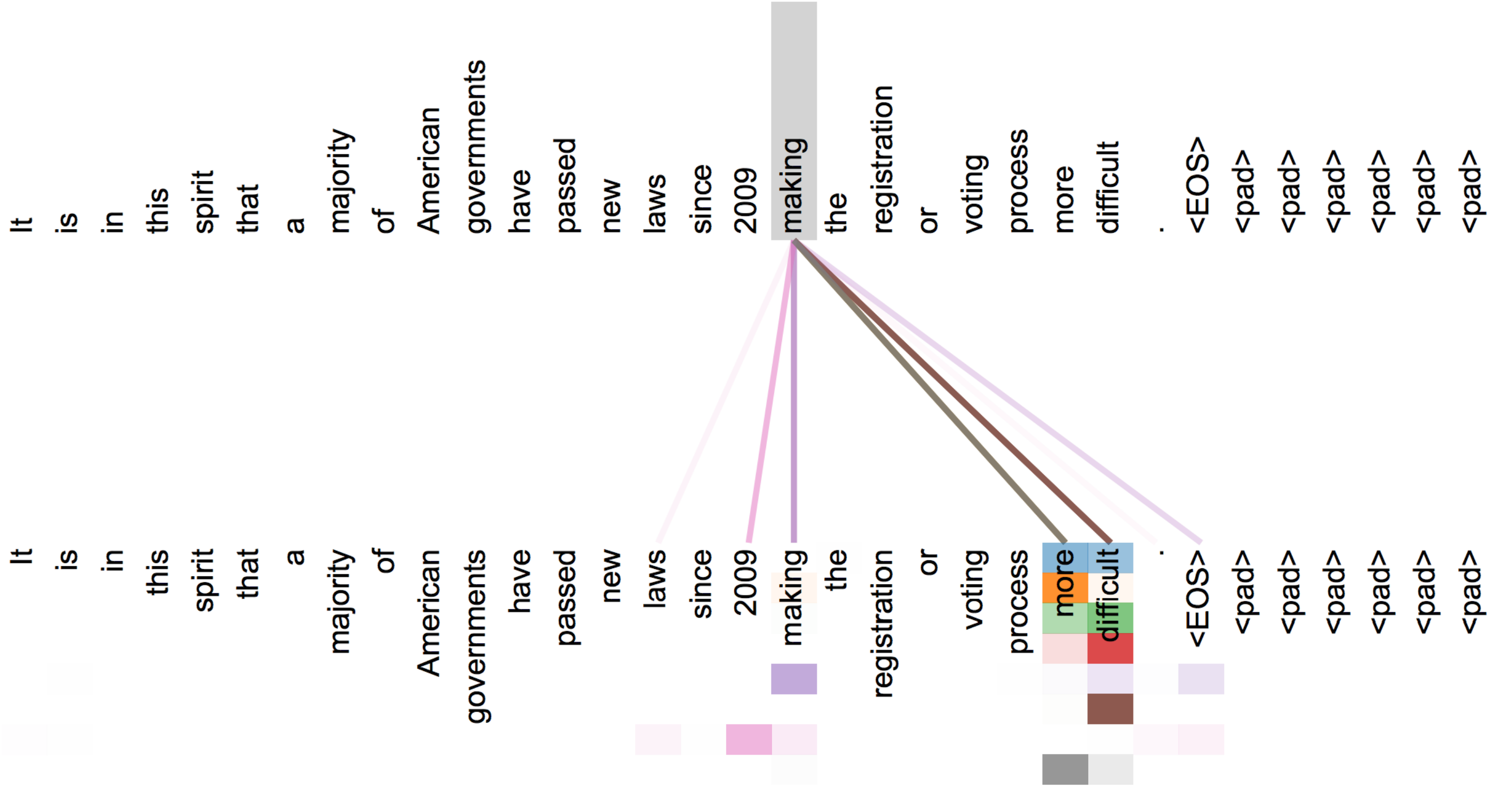
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

pos is the position of a word
i is the dimension index



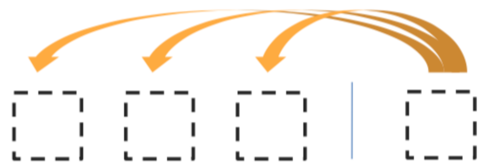
Self Attention Visualization in Layer 5

- Words start to pay attention to other words in sensible ways



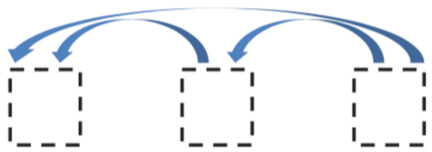
Transformer Decoder

- 2 sublayer changes in decoder
- Masked decoder self-attention
  - Only depends on previous words
- Encoder-Decoder Attention
  - Queries come from previous decoder layer and keys and values come from output of encoder

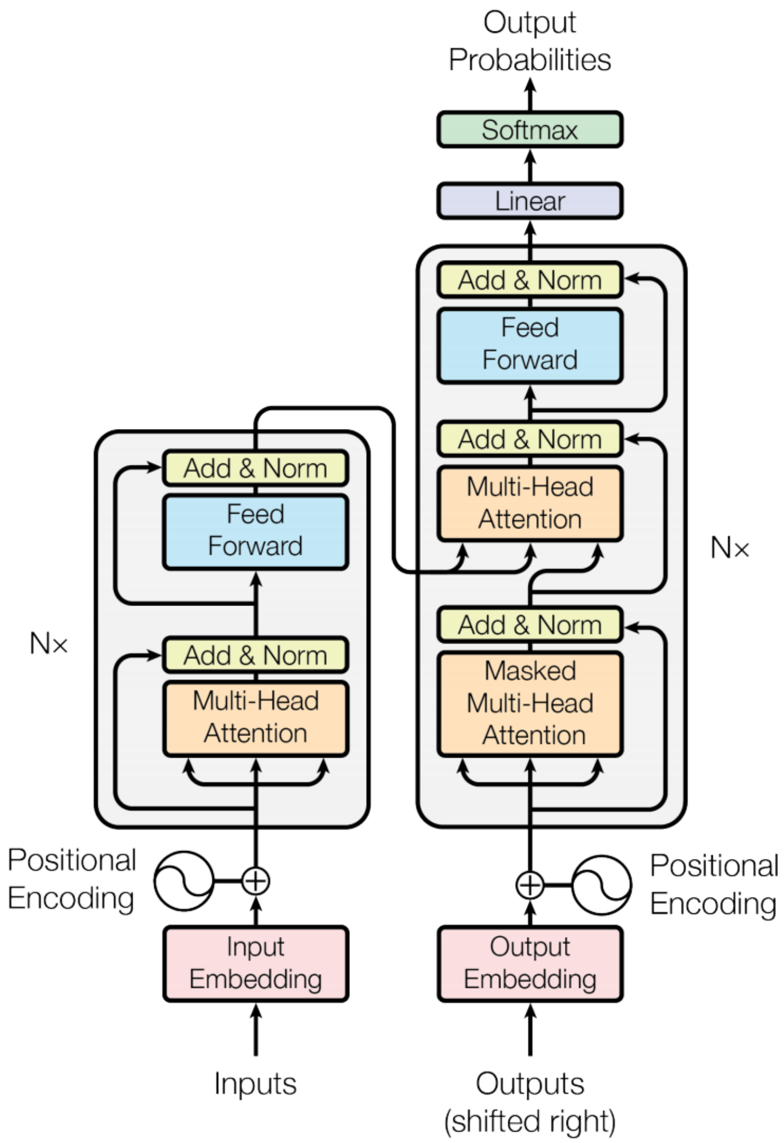


Encoder-Decoder Attention

Encoder Self-Attention



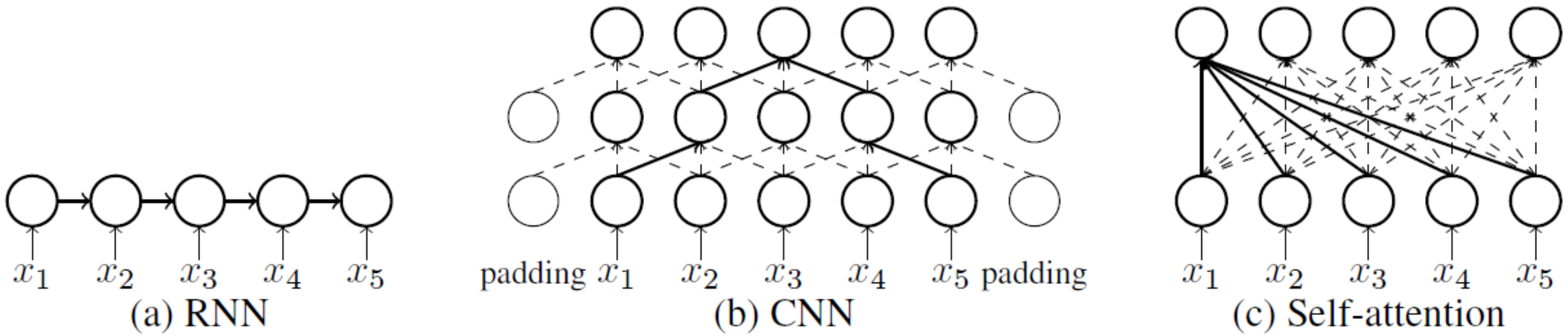
MaskedDecoder Self-Attention



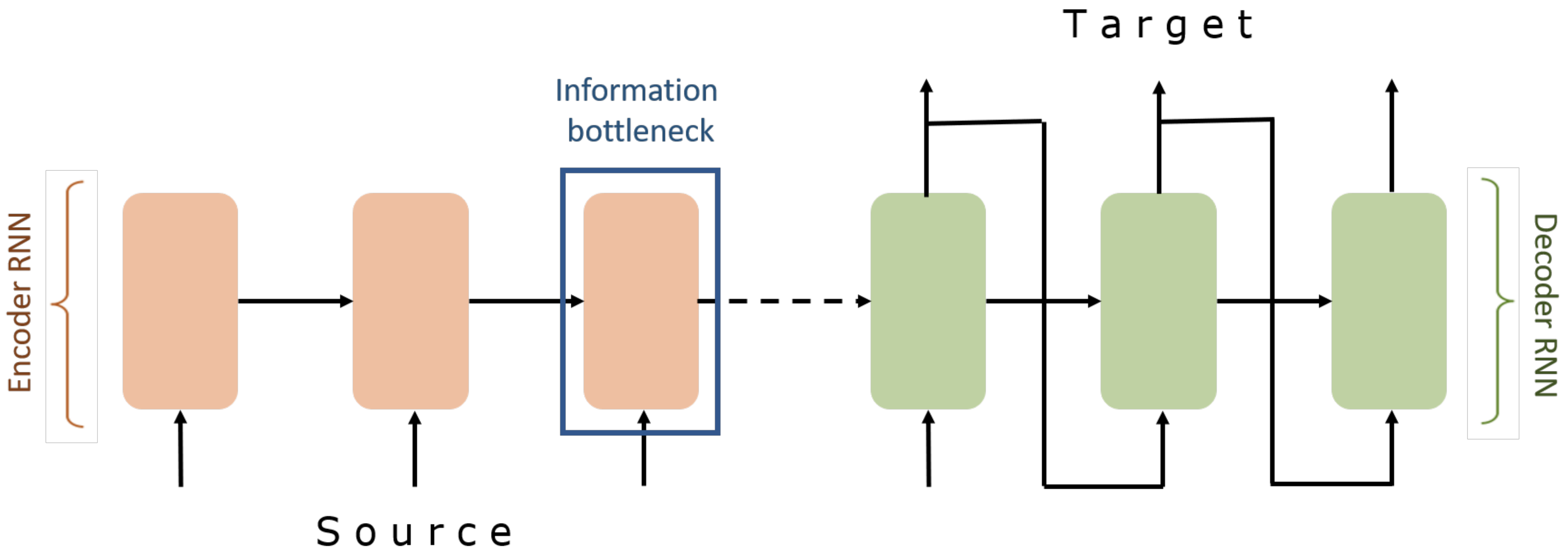
Advantages

- No recurrence: parallel encoding
- Fast training: both encoder and decoder are parallel
- No long range problem: O(1) for all tokens direct connections
- Three attentions: the model does not have to remember too much
- Multi-head attention allows to pay attention to different aspects

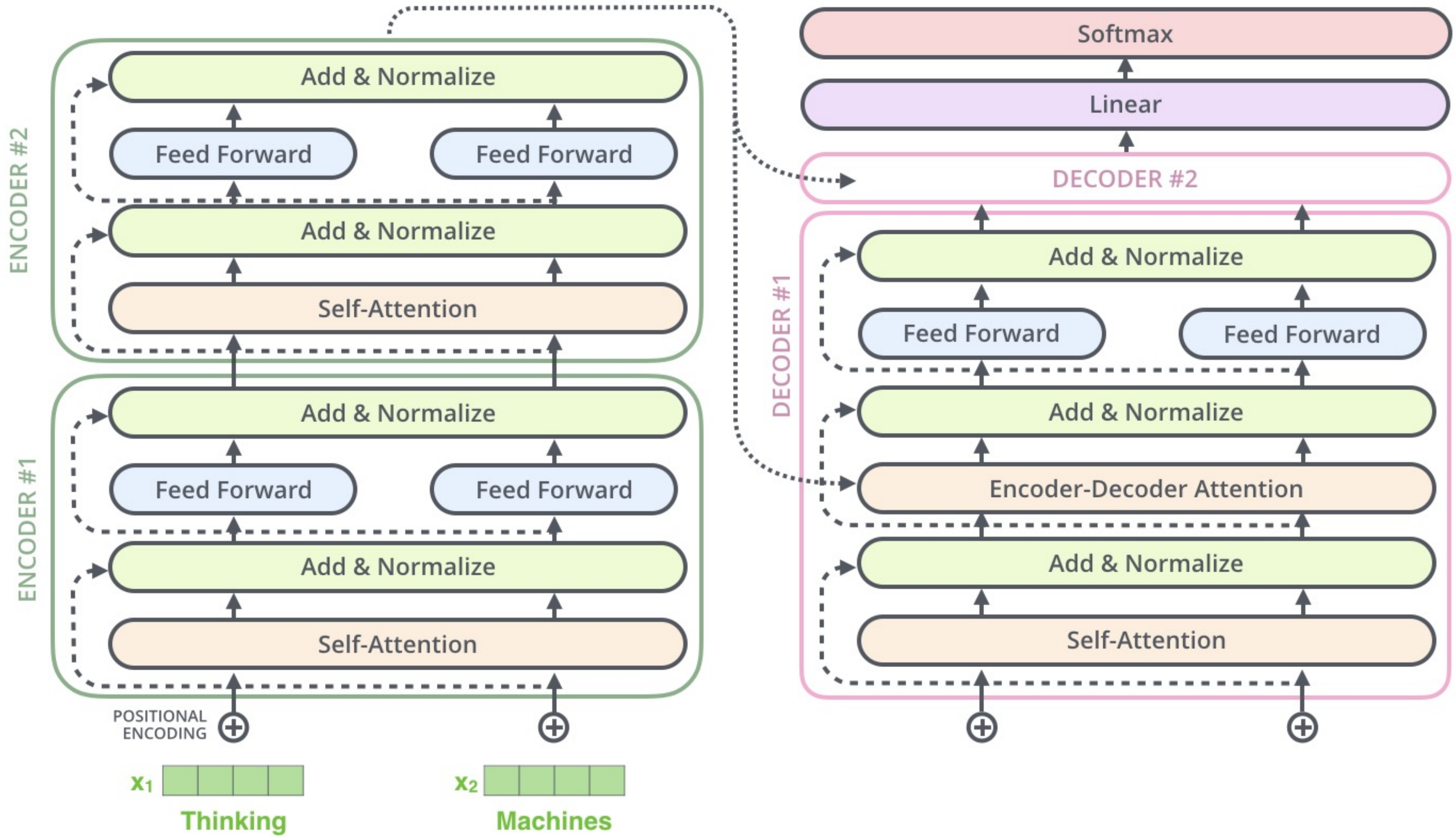
- Why self-attention and CNN is better than RNN on NMT is still under investigation



Avoid Information Bottleneck

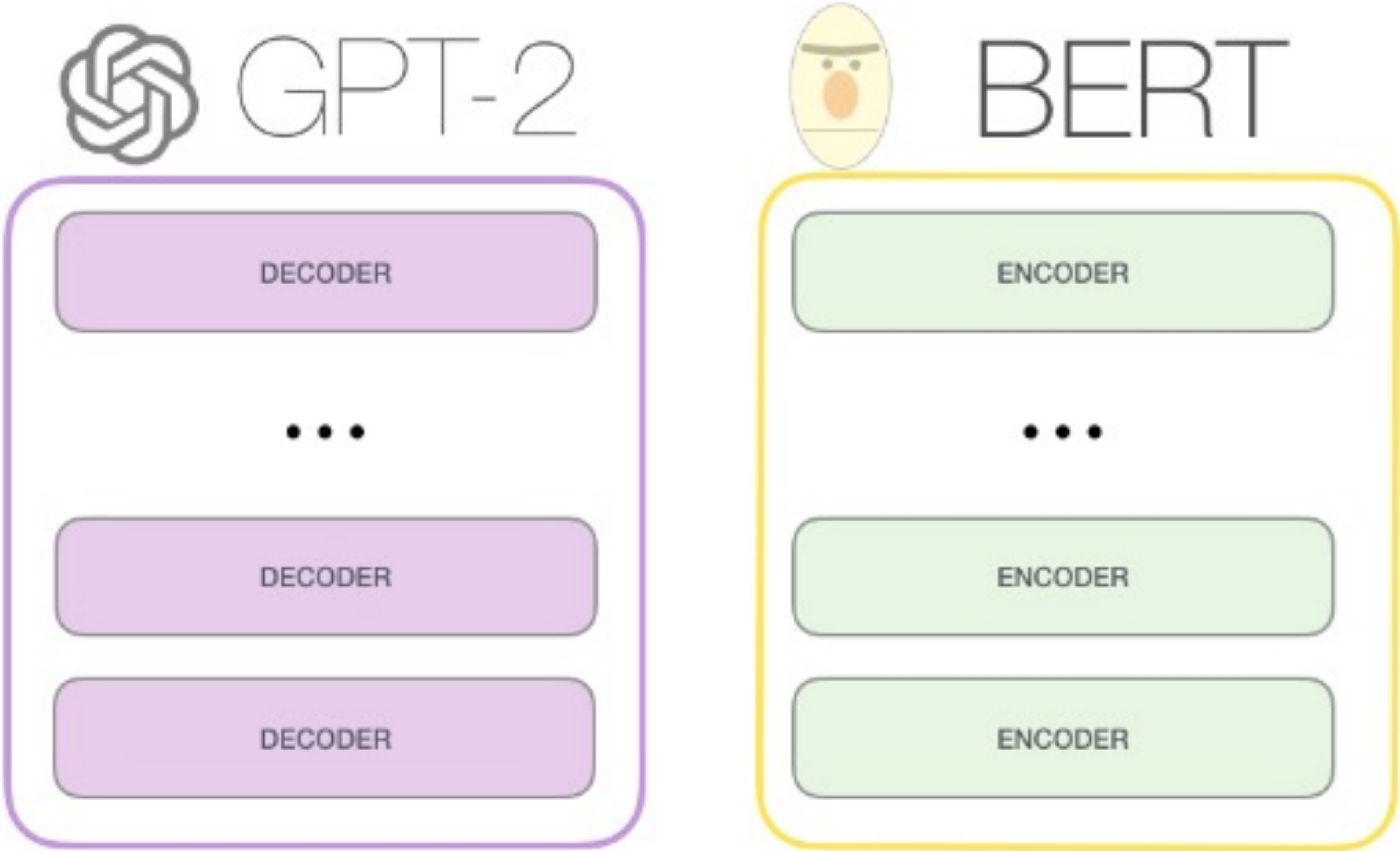


Last time we introduced transformer

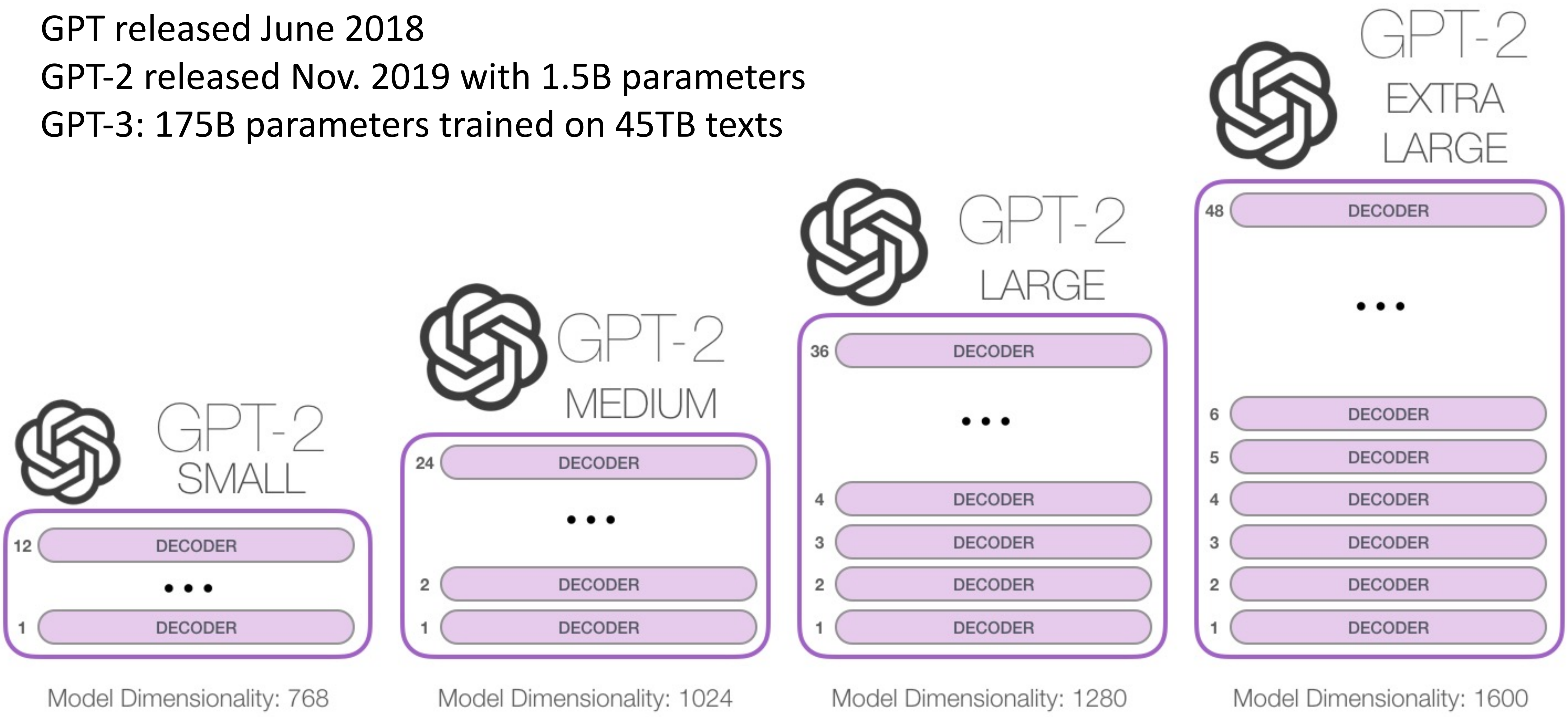


Transformers, GPT-2, and BERT

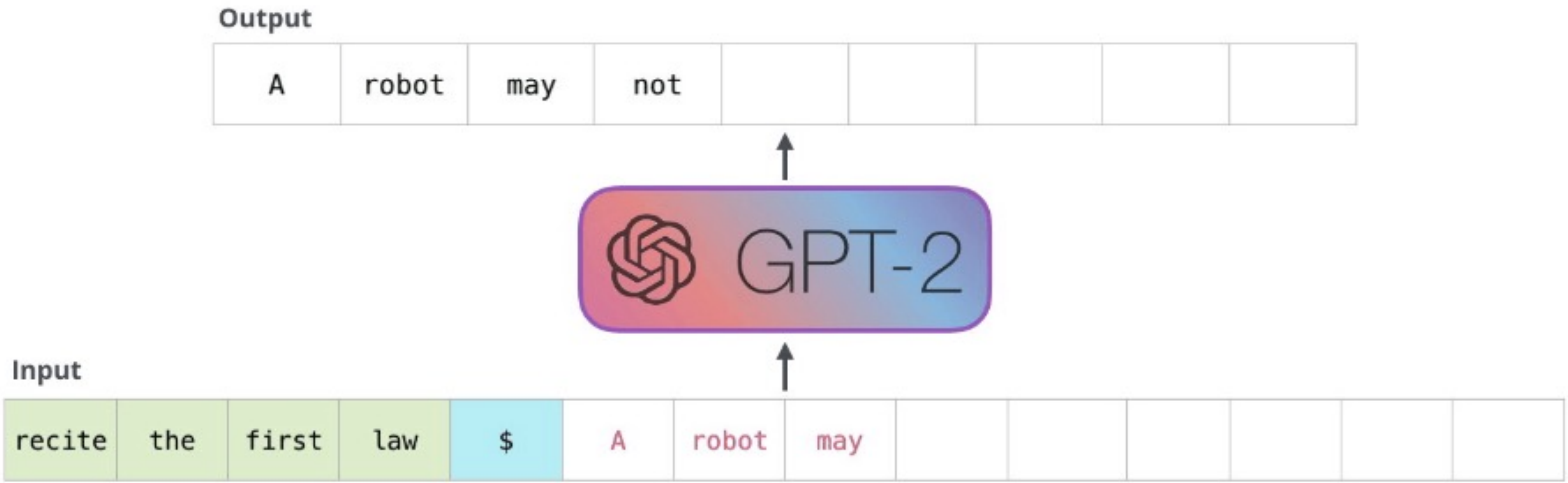
1. A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
2. But if we do not have input, we just want to model the "next word", we can get rid of the Encoder side of a transformer and output "next word" one by one. This gives us GPT.
3. If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.


GPT-2
BERT
DECODER
DECODER
DECODER
ENCODER
ENCODER
ENCODER

GPT released June 2018
GPT-2 released Nov. 2019 with 1.5B parameters
GPT-3: 175B parameters trained on 45TB texts



GPT-2 in action



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.

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 }

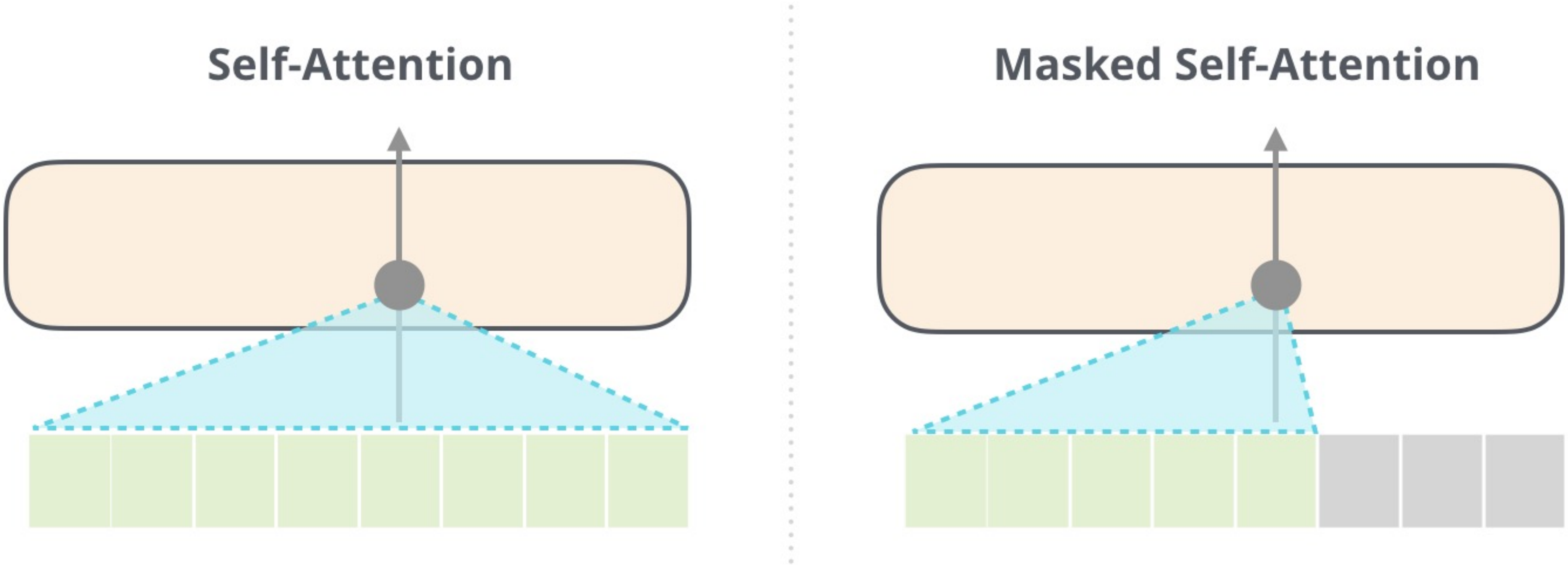
{‘l o w </w>’: 5, ‘l o w e r </w>’: 2, ‘n e w es t </w>’: 6, ‘w i d es t </w>’: 3 }

{‘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)

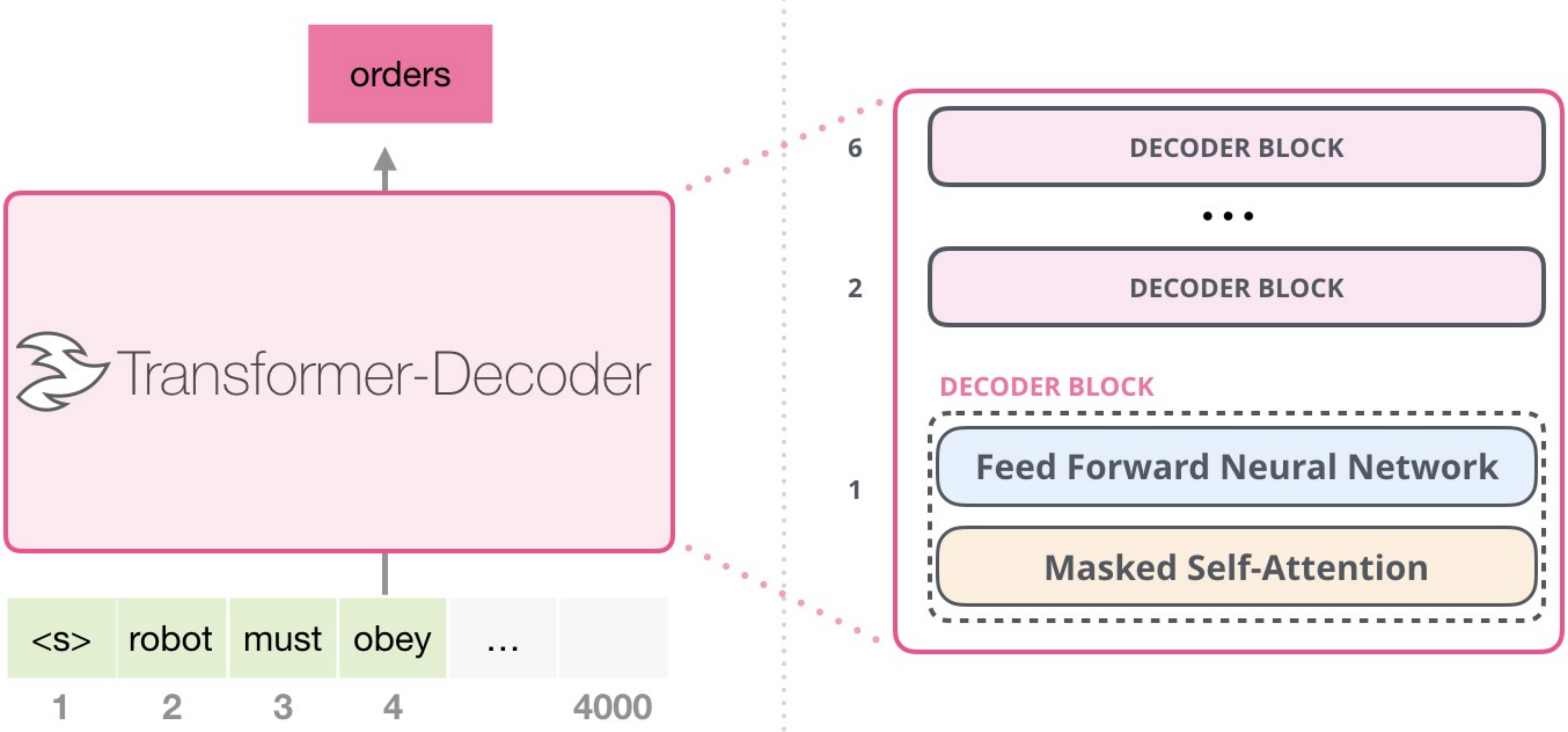
{‘lo w </w>’: 5, ‘lo w e r </w>’: 2, ‘n e w est</w>’: 6, ‘w i d est</w>’: 3 } (lo freq 7)

…..

Masked Self-Attention (to compute more efficiently)



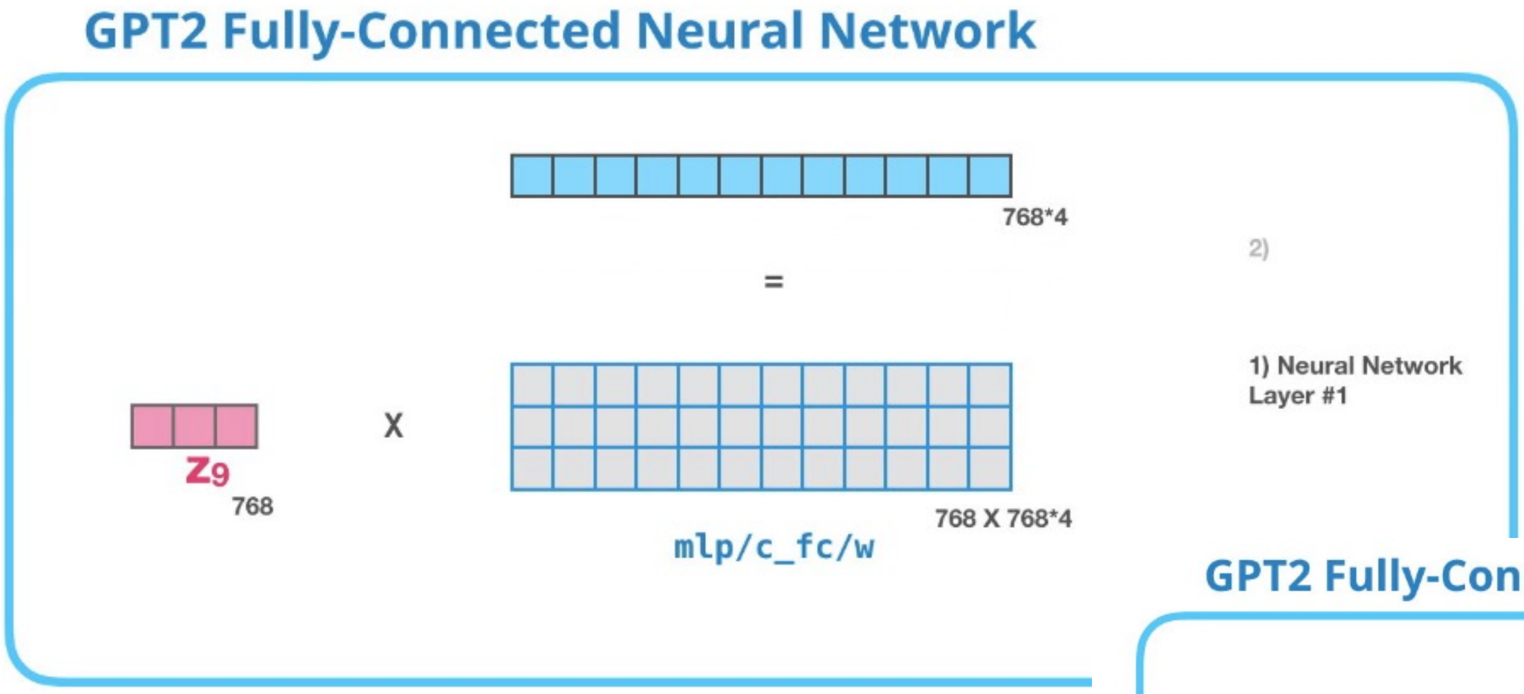
Masked Self-Attention

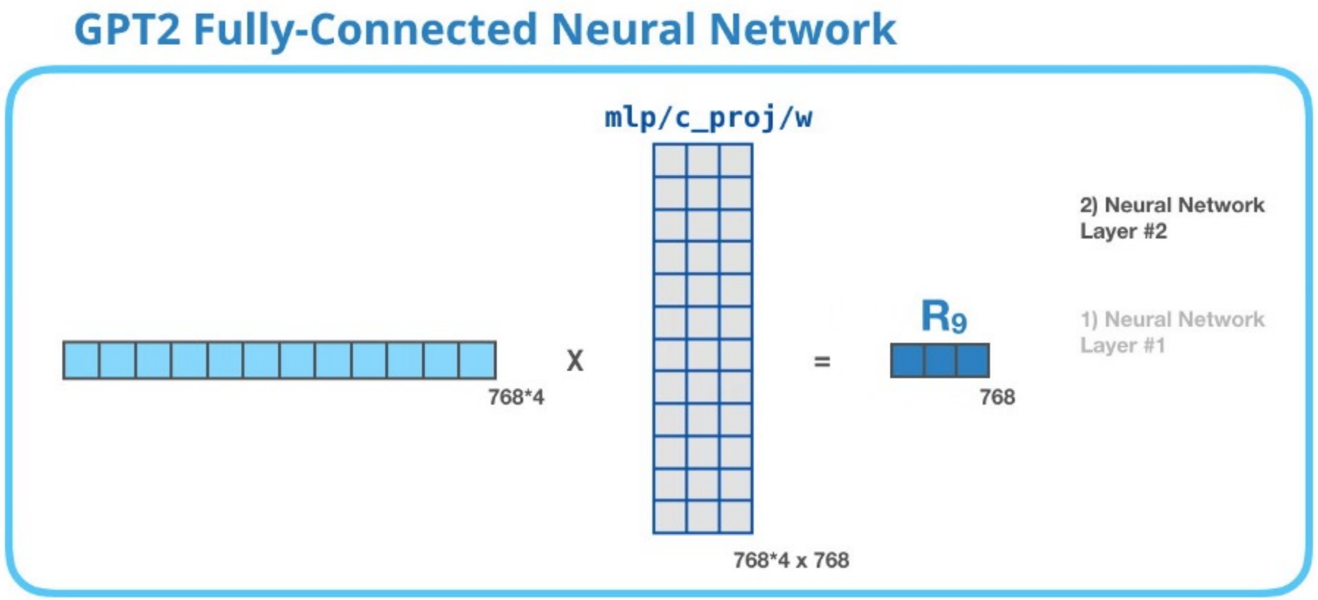


Masked Self-Attention Calculation

Re-use previous computation results: at any step, only need to results of q, k , v related to the new output word, no need to re-compute the others. Additional computation is linear, instead of quadratic.

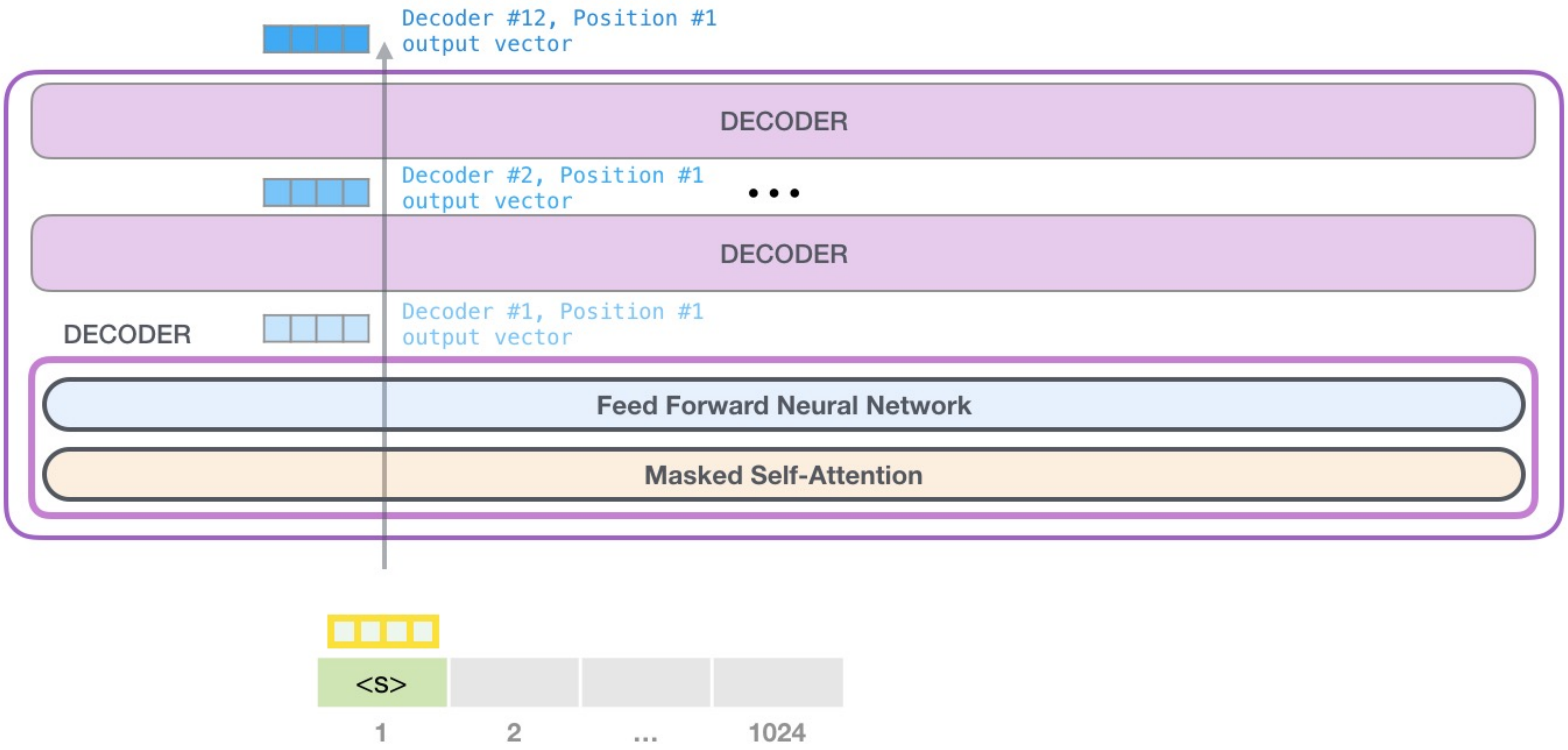
GPT-2 fully connected network has two layers (Example for GPT-2 small)





768 is small model size

GPT-2 has a parameter top-k, so that we sample words from top k (highest probability from softmax) words for each output



This top-k parameter, if k=1, we would have output like:

The first time I saw the new version of the game, I was so excited. I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of the game, I was so excited to see the new version of

GPT Training

GPT-2 uses unsupervised learning approach to training the language model.

There is no custom training for GPT-2, no separation of pre-training and fine-tuning like BERT.

A story generated by GPT-2

“The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

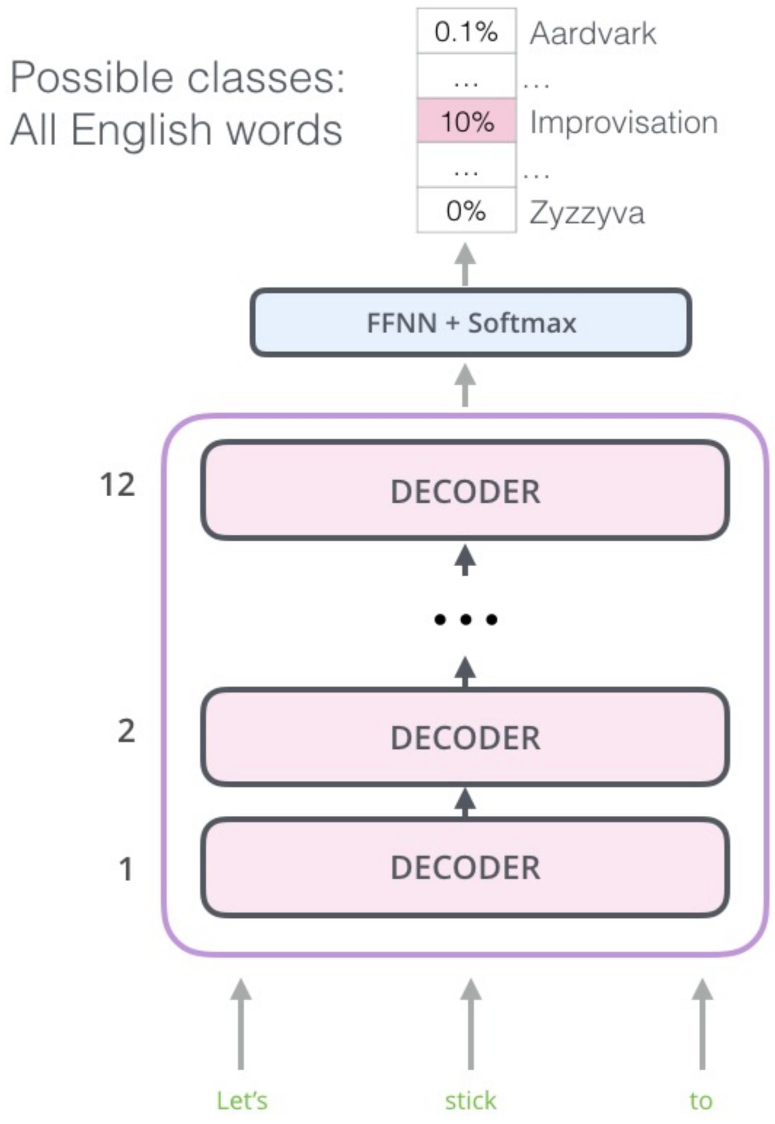
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. `By the time we reached the top of one peak, the water looked blue, with some crystals on top,’ said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns."

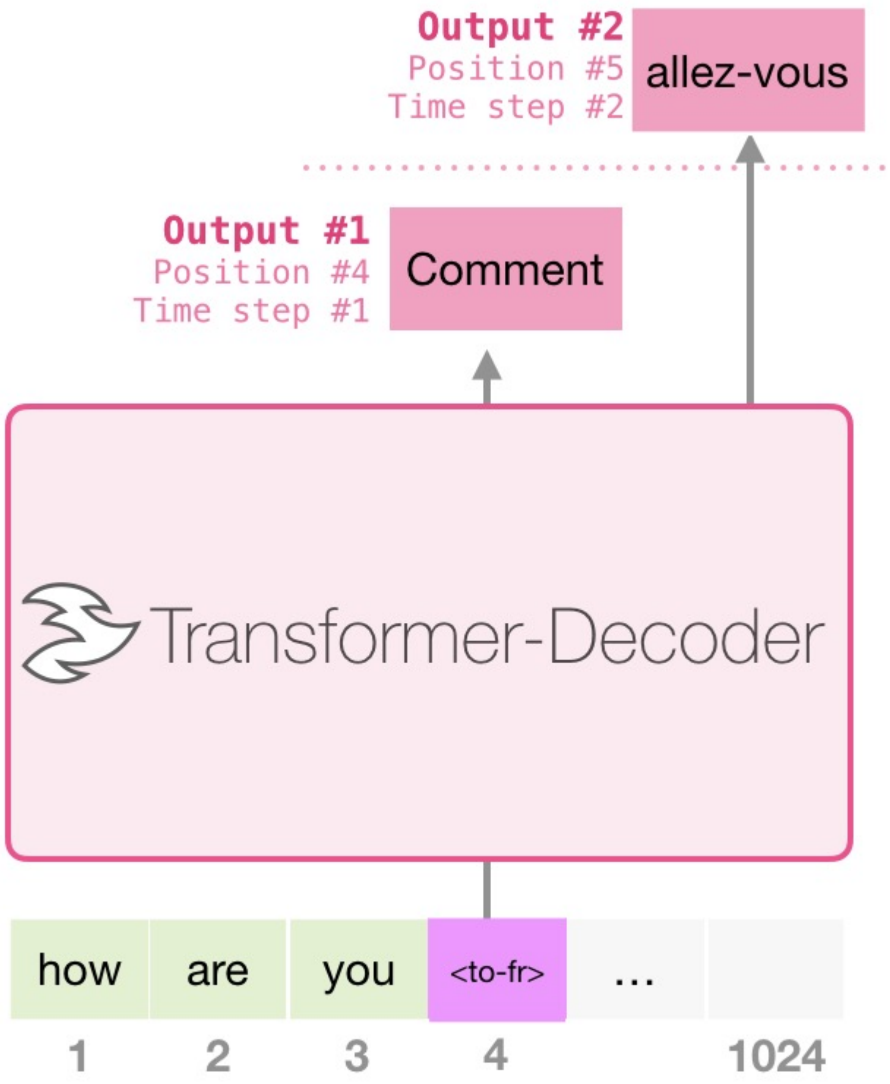
Transformer / GPT prediction



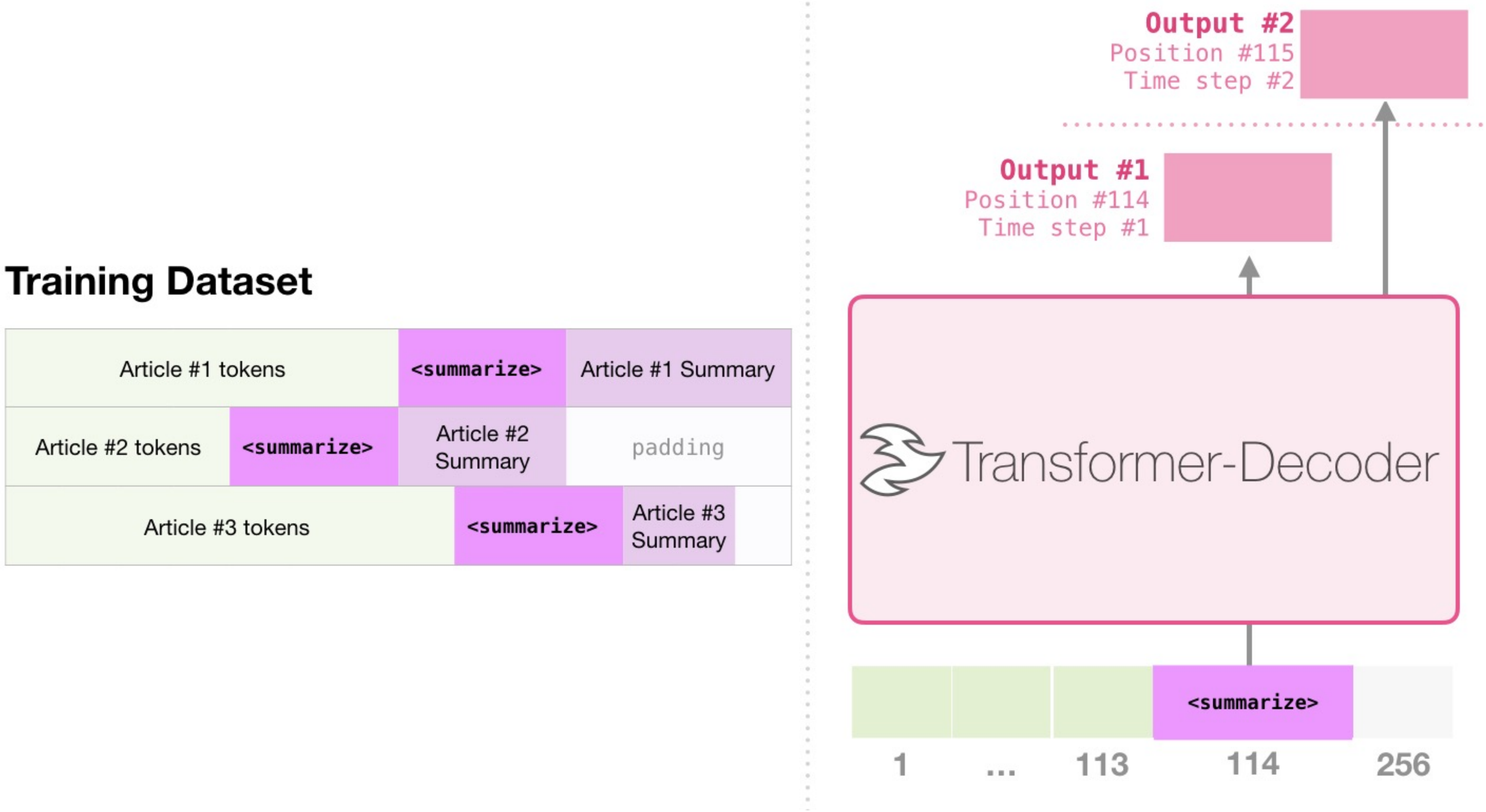
GPT-2 Application: Translation

Training Dataset

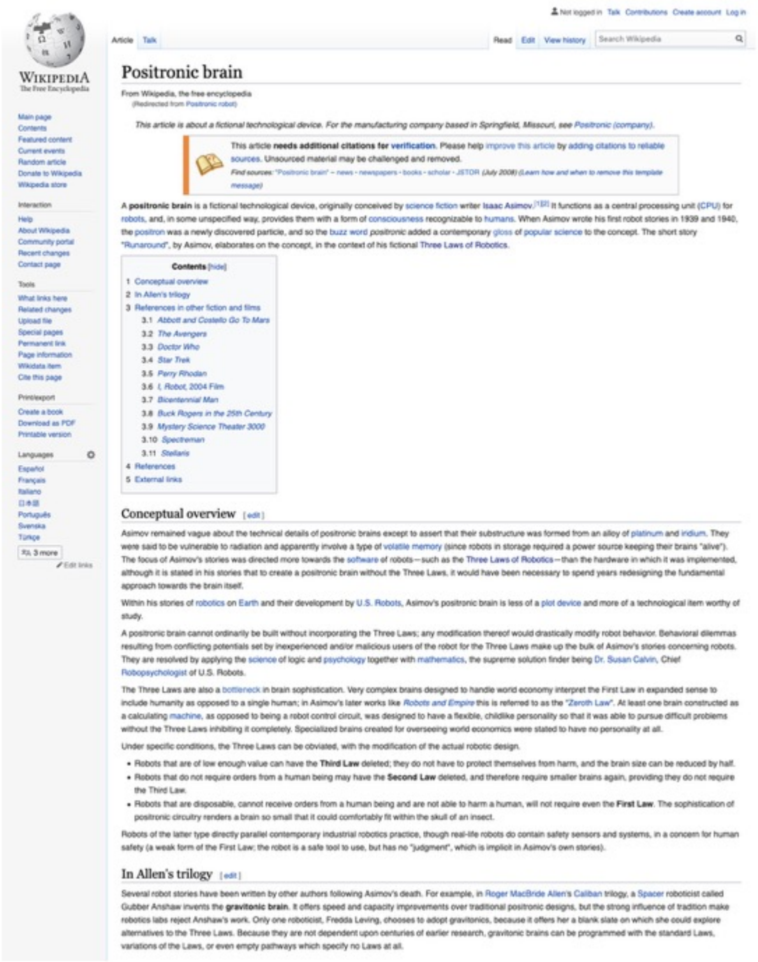
I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



GPT-2 Application: Summarization

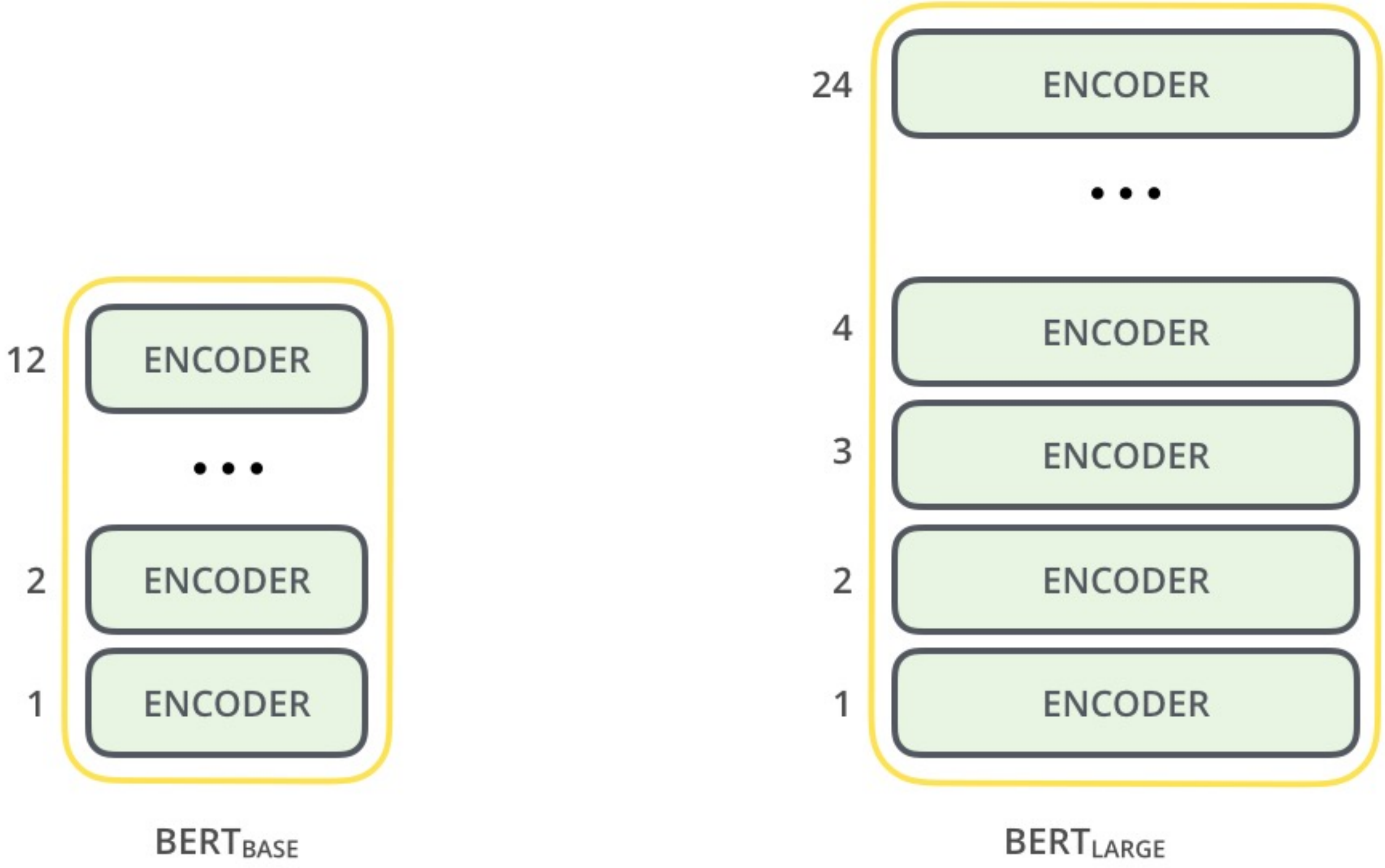


Using wikipedia data



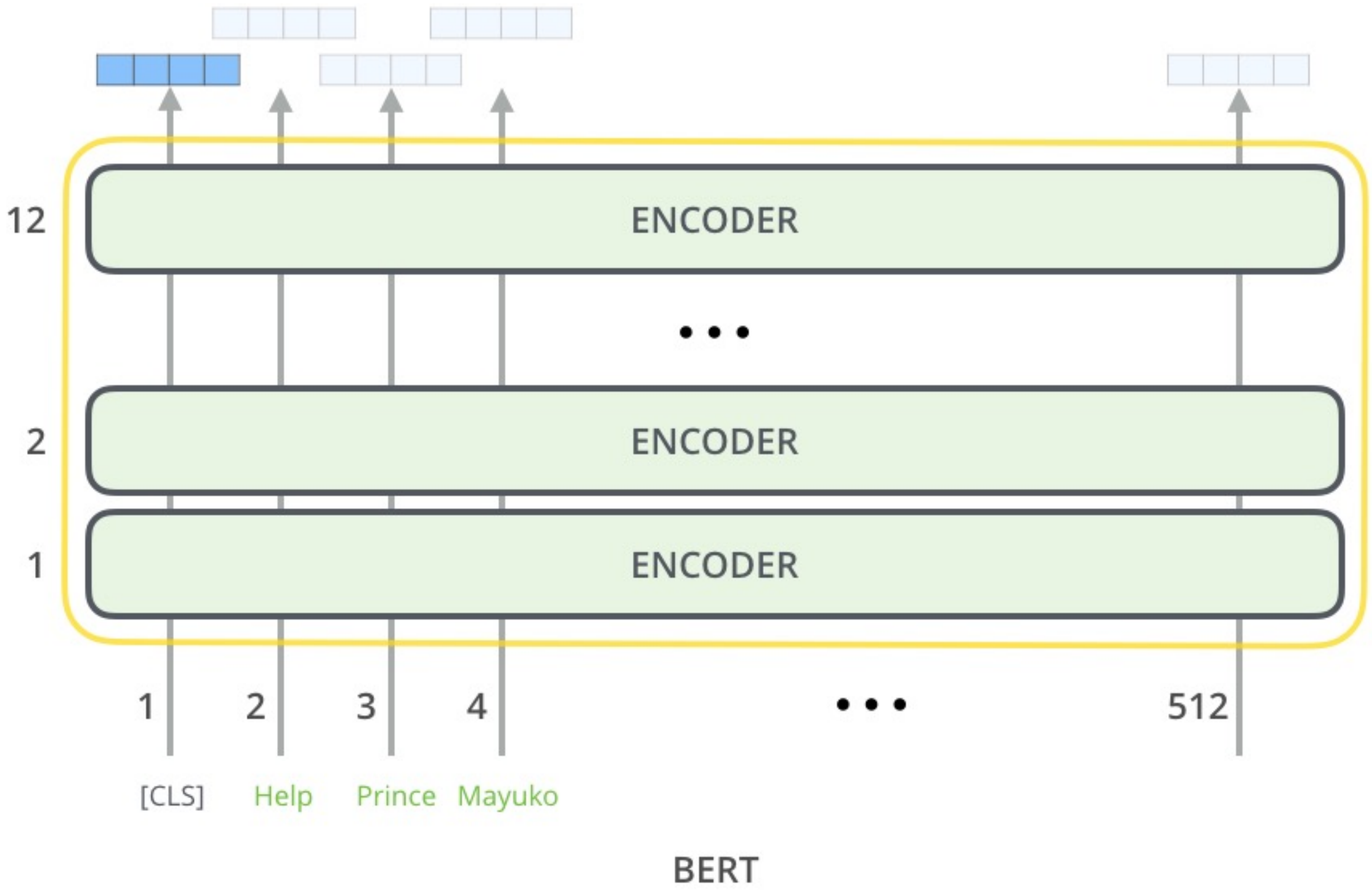
How about representation/word embedding?

BERT (Bidirectional Encoder Representation from Transformers)



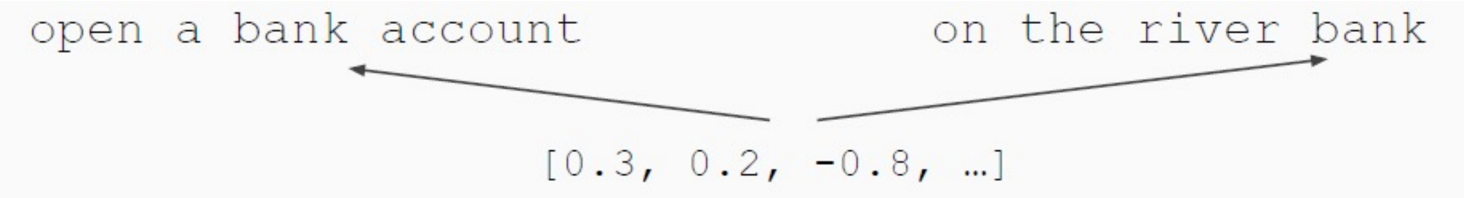
Model input dimension 512

Input and output vector size



Pre-training in NLP

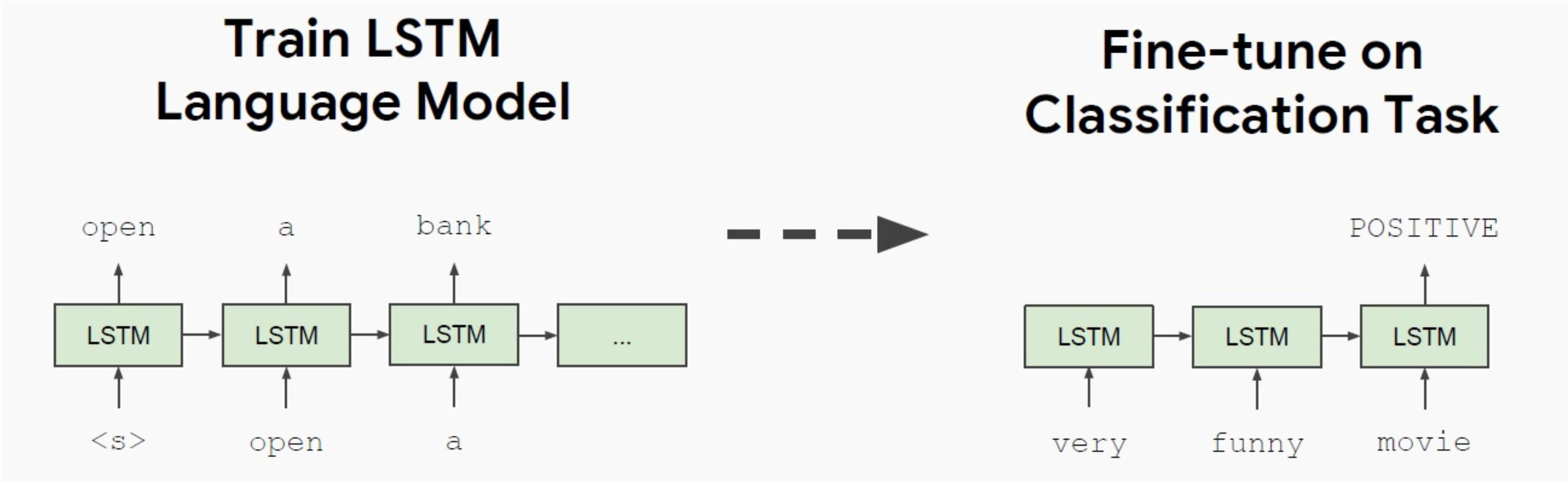
- Word embeddings are the basis of deep learning for NLP
- Word embeddings (word2vec, GloVe) are often *pre-trained* on text corpus from co-occurrence statistics
- **Problem**: Word embeddings are applied in a context free manner



- **Solution**: Train *contextual* representations on text corpus

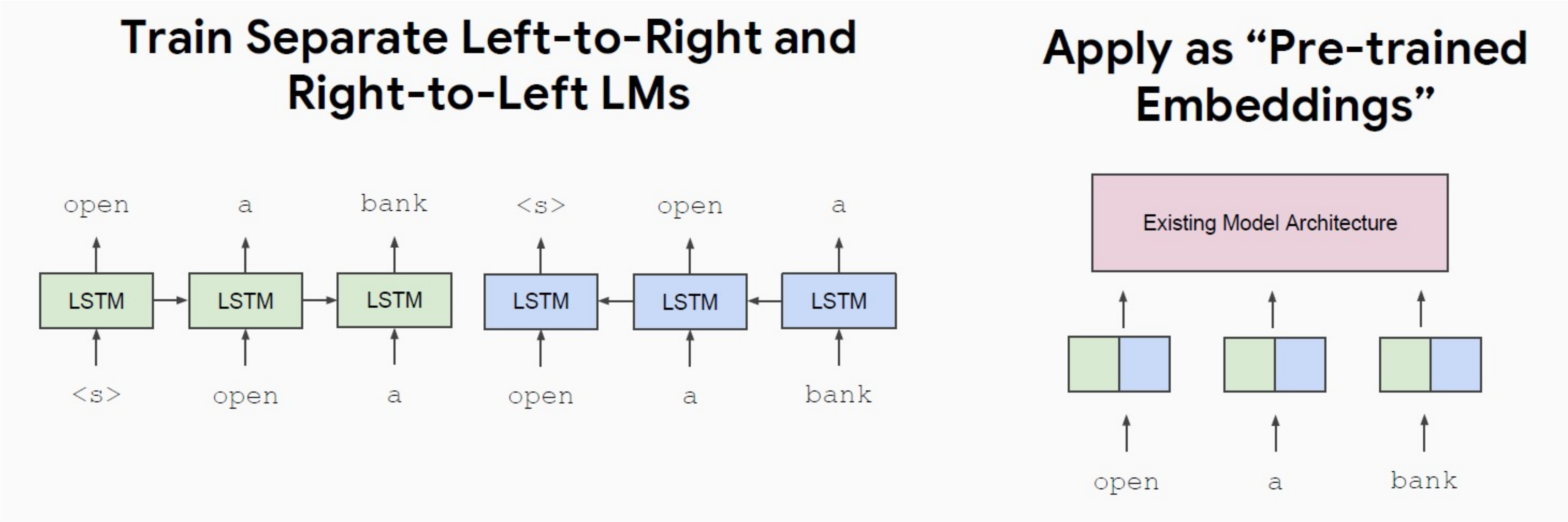
History of Contextual Representations

- *Semi-Supervised Sequence Learning*, Google, 2015

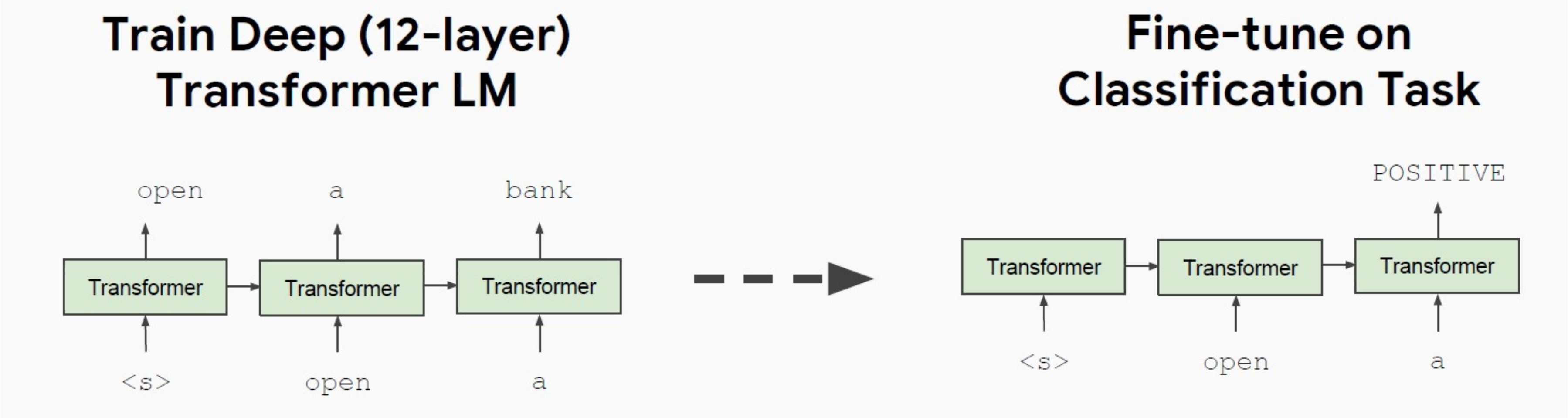


History of Contextual Representations

- *ELMo: Deep Contextual Word Embeddings*, AI2 & University of Washington, 2017



- *Improving Language Understanding by Generative Pre-Training*, OpenAI, 2018 (GPT)

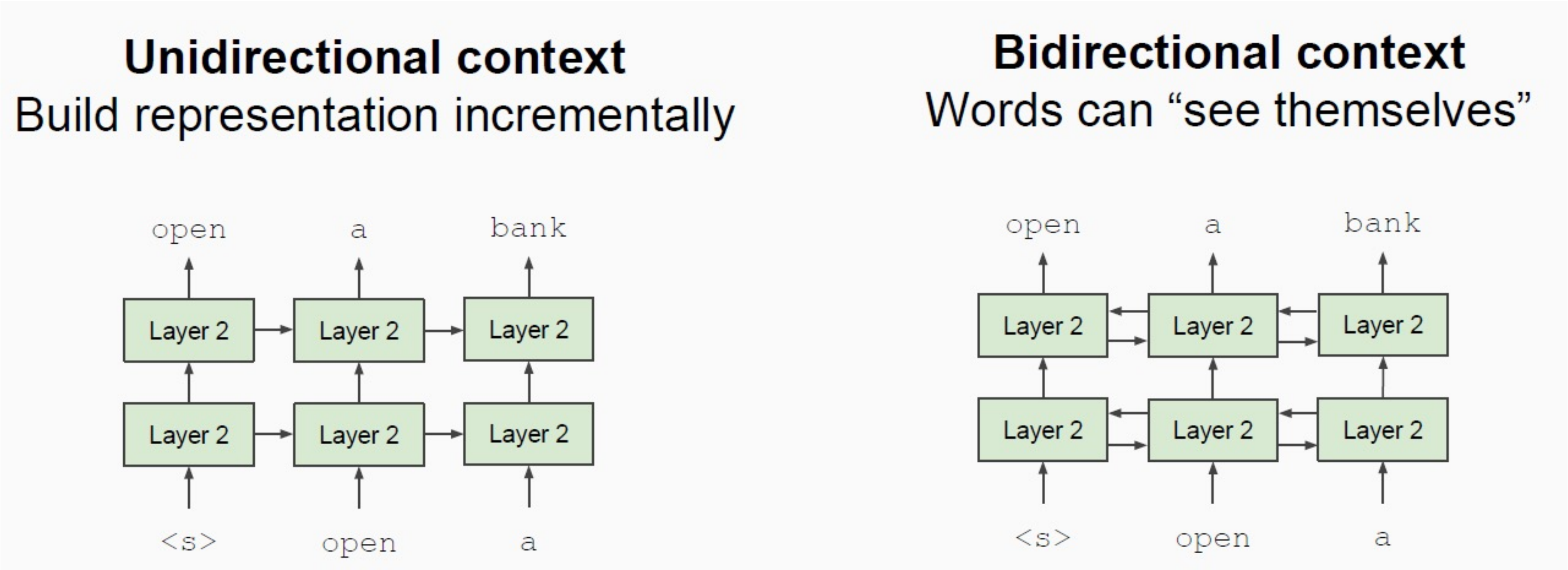


Problem with Previous Methods

- **Problem**: Language models only use left context *or* right context, but language understanding is bidirectional.

- Why are LMs unidirectional?
  - Reason 1: Directionality is needed to generate a well-formed probability distribution.
    - We don't care about this.
  - Reason 2: Words can "see themselves" in a bidirectional encoder.

Unidirectional vs. Bidirectional Models



Masked LM

- **Solution**: Mask out *k*% of the input words, and then predict the masked words
  - We always use *k* = 15%

- Too little masking: Too expensive to train
- Too much masking: Not enough context

Masked LM

- Problem: Mask token never seen at fine-tuning

- Solution: 15% of the words to predict, but don't replace with [MASK] 100% of the time. Instead:
  - 80% of the time, replace with [MASK]
    - went to the store → went to the [MASK]
  - 10% of the time, replace random word
    - went to the store → went to the running
  - 10% of the time, keep same
    - went to the store → went to the store

Next Sentence Prediction

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

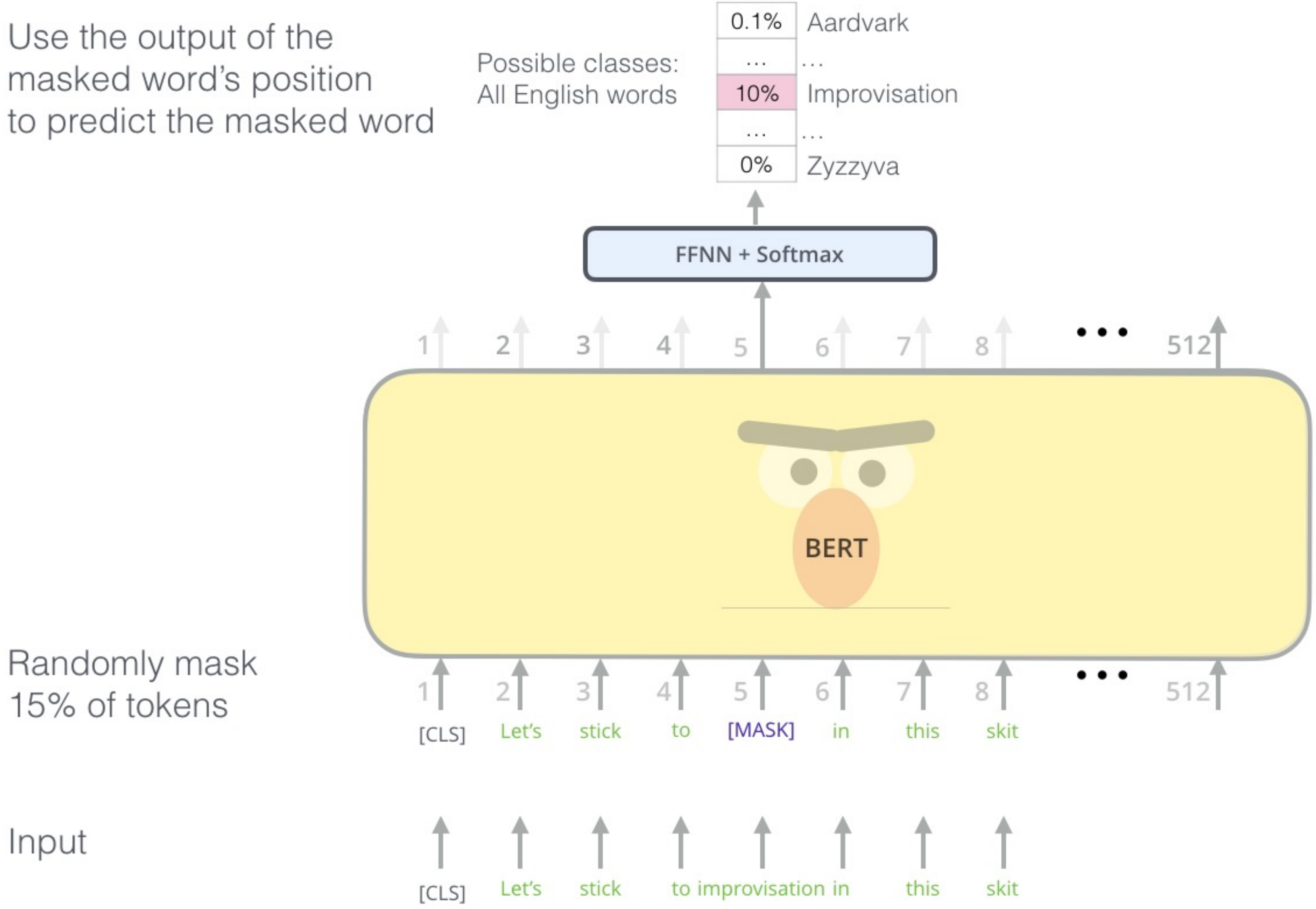
Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

BERT pretraining

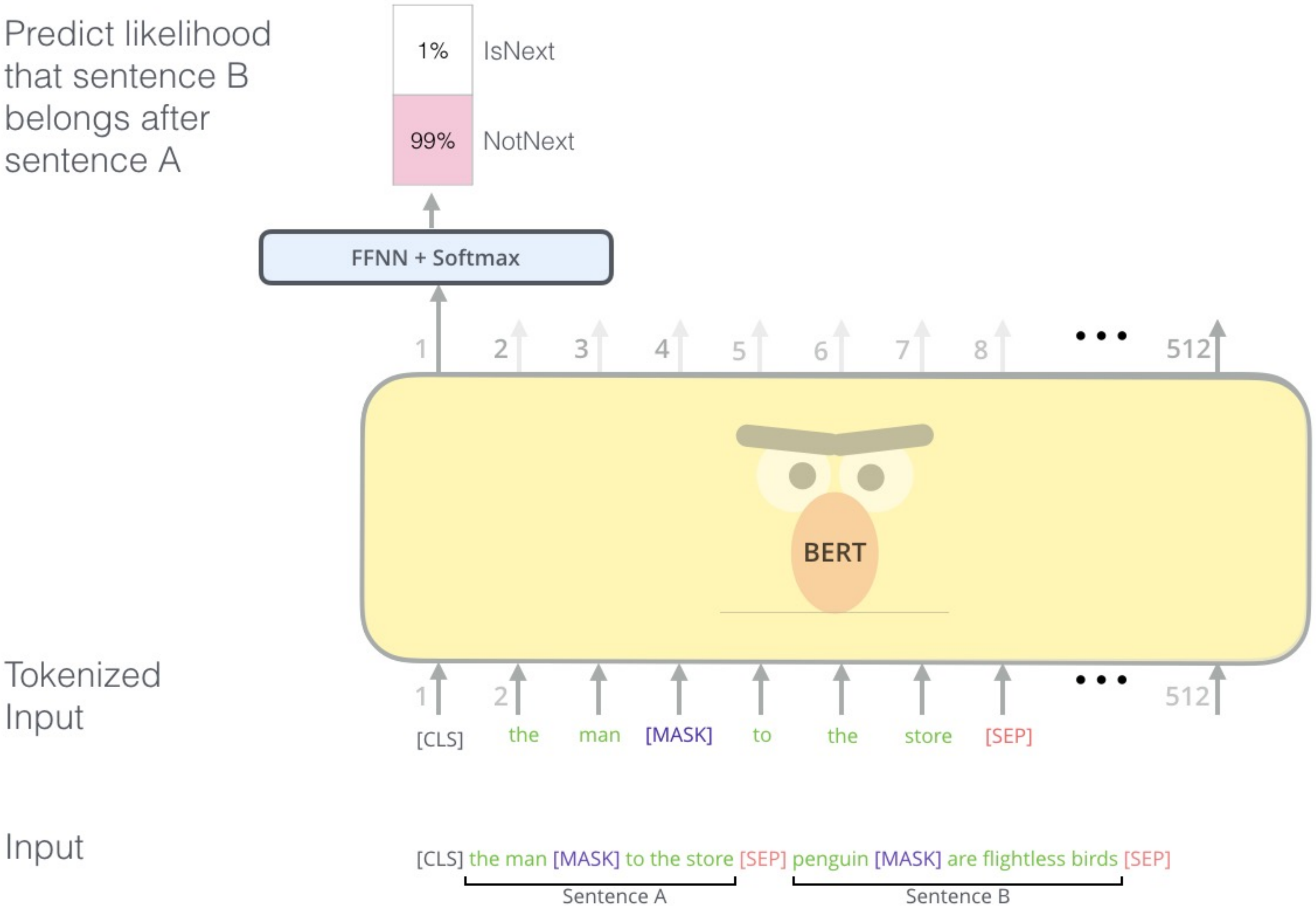
ULM-FiT (2018): Pre-training ideas, transfer learning in NLP.
ELMo: Bidirectional training (LSTM)
Transformer: Although used things from left, but still missing from the right.
GPT: Use Transformer Decoder half.
BERT: Switches from Decoder to Encoder, so that it can use both sides in training and invented corresponding training tasks: masked language model

BERT Pretraining Task 1: masked words

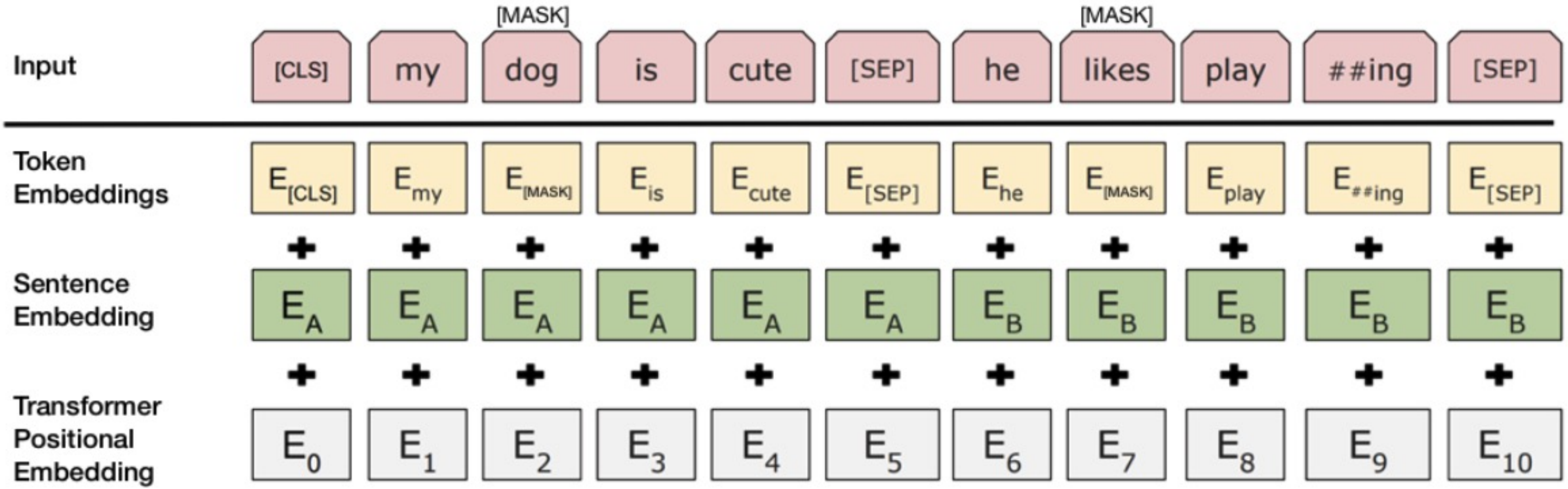


Out of this 15%,
80% are [Mask],
10% random words
10% original words

BERT Pretraining Task 2: two sentences



BERT Pretraining Task 2: two sentences

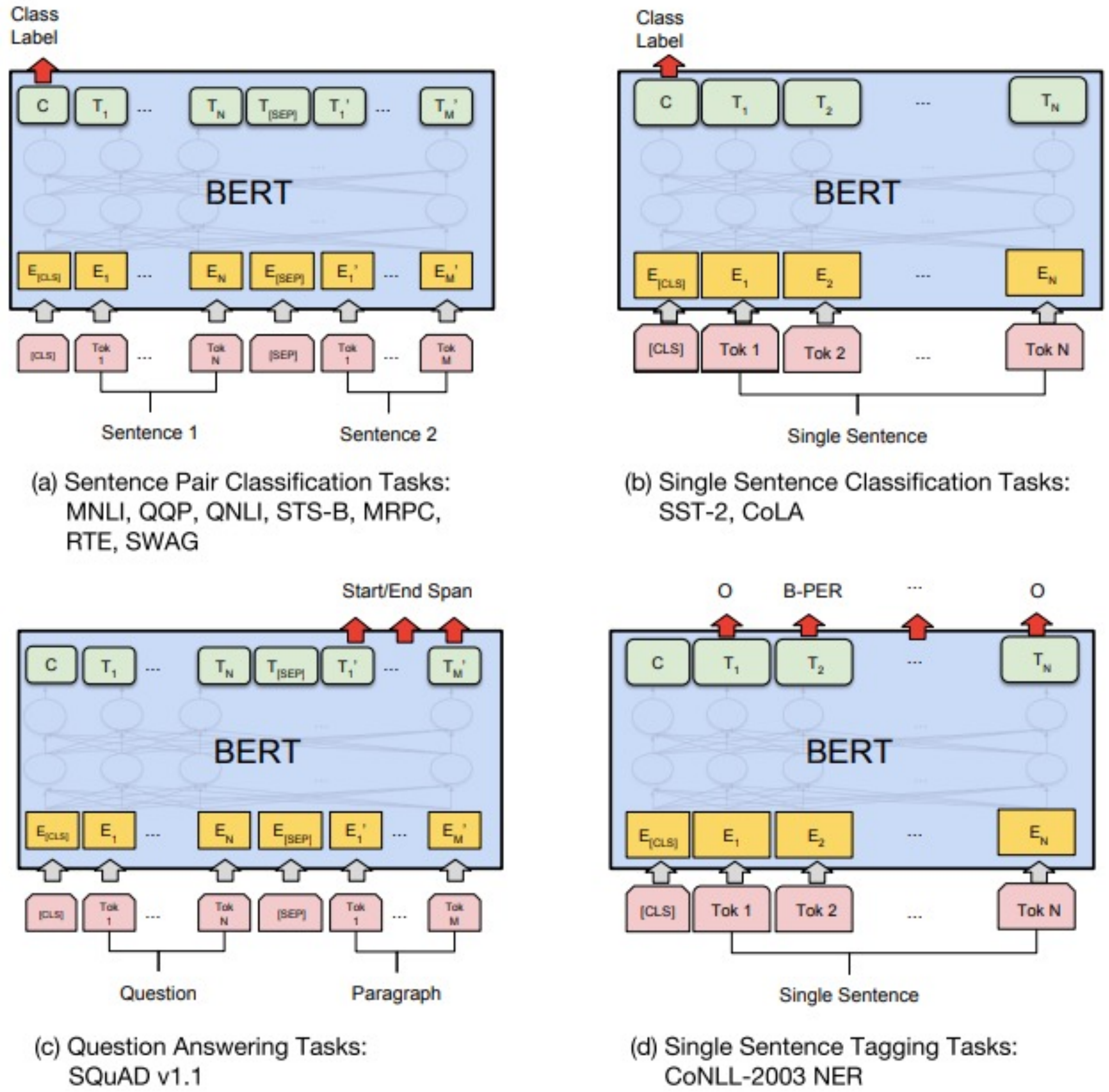


50% true second sentences

50% random second sentences

Fine-tuning BERT for other specific tasks

MNLI
QQP (Quaro Question Pairs)
 Semantic equivalence)
QNLI (NL inference dataset)
STS-B (texture similarity)
MRPC (paraphrase, Microsoft)
RTE (textual entailment)
SWAG (commonsense inference)
SST-2 (sentiment)
CoLA (linguistic acceptability
SQuAD (question and answer)



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

SST (Stanford sentiment treebank): 215k phrases with fine-grained sentiment labels in the parse trees of 11k sentences.

NLP Tasks: Multi-Genre Natural Lang. Inference

MNLI: 433k pairs of examples, labeled by entailment, neutral or contraction

Met my first girlfriend that way.	FACE-TO-FACE **contradiction** C C N C	I didn't meet my first girlfriend until later.
8 million in relief in the form of emergency housing.	GOVERNMENT **neutral** N N N N	The 8 million dollars for emergency housing was still not enough to solve the problem.
Now, as children tend their gardens, they have a new appreciation of their relationship to the land, their cultural heritage, and their community.	LETTERS **neutral** N N N N	All of the children love working in their gardens.
At 8:34, the Boston Center controller received a third transmission from American 11	9/11 **entailment** E E E E	The Boston Center controller got a third transmission from American 11.
I am a lacto-vegetarian.	SLATE **neutral** N N E N	I enjoy eating cheese too much to abstain from dairy.
someone else noticed it and i said well i guess that's true and it was somewhat melodious in other words it wasn't just you know it was really funny	TELEPHONE **contradiction** C C C C	No one noticed and it wasn't funny at all.

Table 1: Randomly chosen examples from the development set of our new corpus, shown with their genre labels, their selected gold labels, and the validation labels (abbreviated E, N, C) assigned by individual annotators.

NLP Tasks (SQuAD -- Stanford Question Answering Dataset):

Sample: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Which NFL team represented the AFC at Super Bowl 50?

Ground Truth Answers: Denver Broncos

Which NFL team represented the NFC at Super Bowl 50?

Ground Truth Answers: Carolina Panthers

SegaTron/SegaBERT

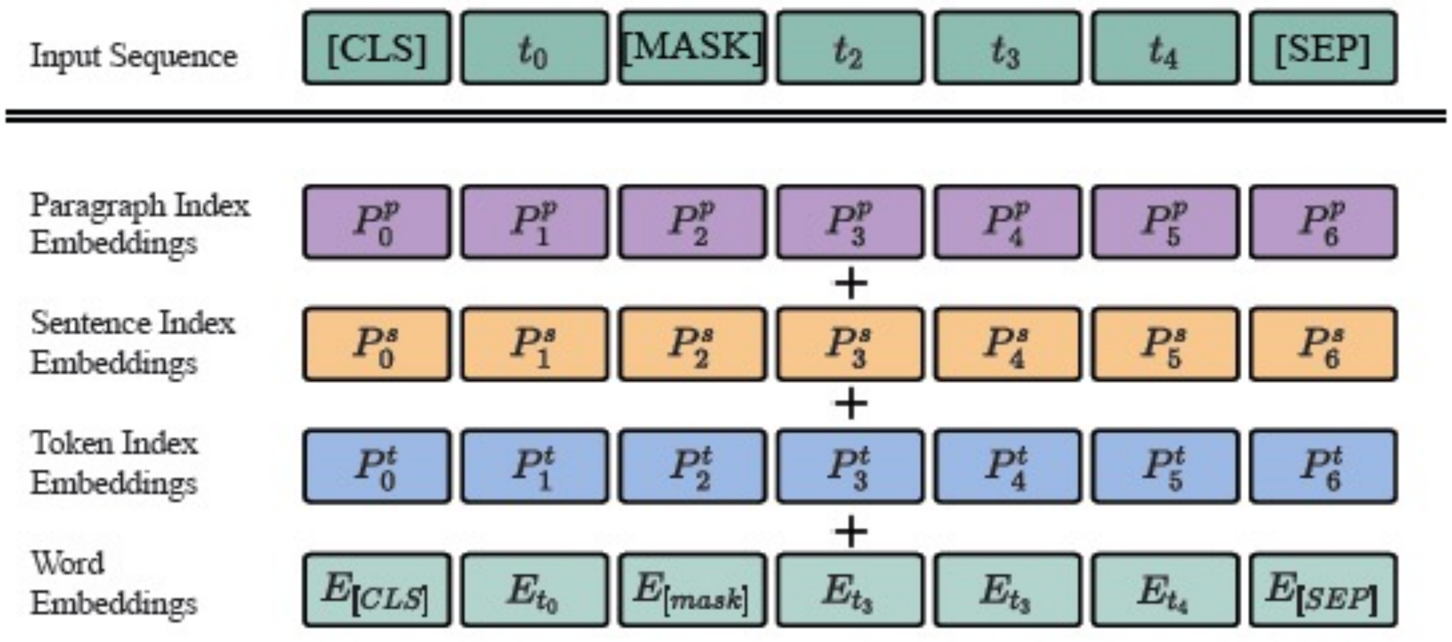
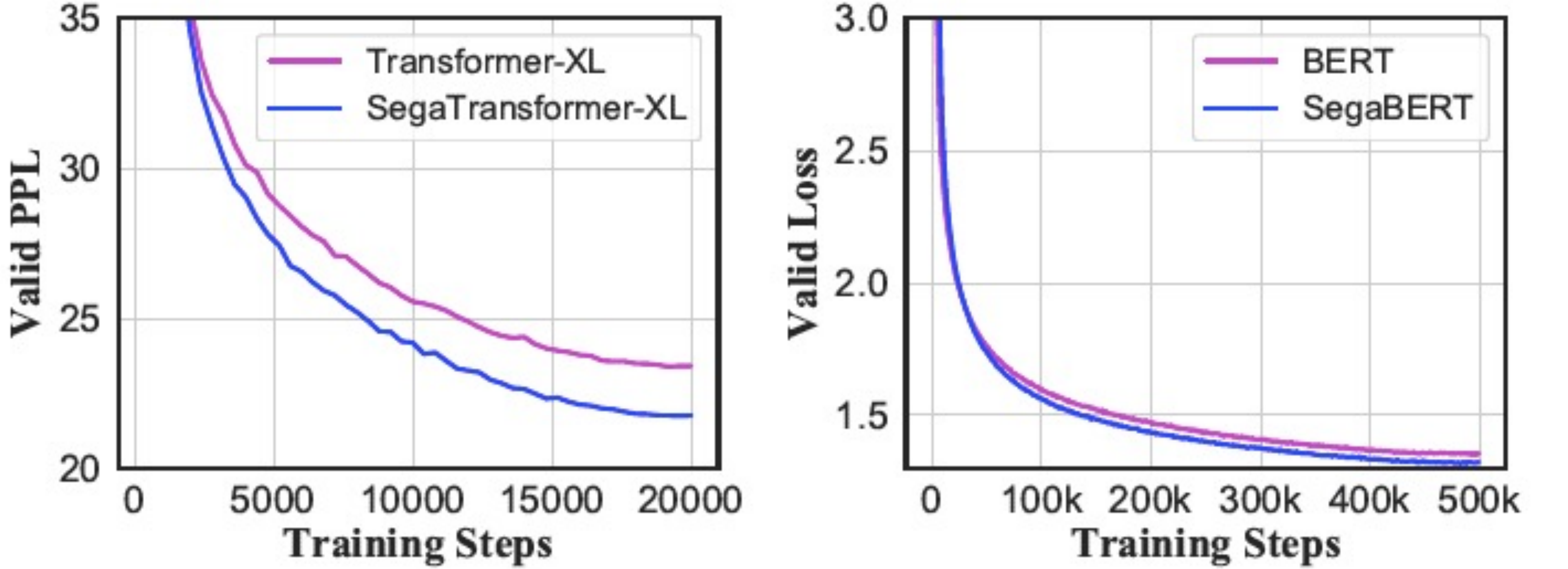


Figure 1: Input Representation of SegaBERT

H. Bai, S. Peng, J. Lin, L. Tan, K. Xiong, W. Gao, M. Li: SgaTron: Segment-aware transformer for language modeling and understanding. AAAI'2021



(a) Language modeling perplexities

(b) Pre-training losses

Figure 2: Valid perplexities and losses during the training processes of language modeling and pre-training.

Task(Metrics)	BASE model(wikipedia 500K steps)				LARGE model(wikibooks 1000K steps)			
	dev		test		dev		test	
	BERT	SegaBERT	BERT	SegaBERT	BERT	SegaBERT	BERT	SegaBERT
CoLA (Matthew Corr.)	**55.0**	54.7	43.5	**50.7**	60.6	**65.3**	60.5	**62.6**
SST-2 (Acc.)	91.3	**92.1**	91.2	**91.5**	93.2	**94.7**	**94.9**	94.8
MRPC (F1)	**92.6**	92.4	88.9	**89.3**	-	92.3	89.3	**89.7**
STS-B (Spearman Corr.)	88.9	**89.0**	83.9	**84.6**	-	90.3	86.5	**88.6**
QQP (F1)	86.5	**87.0**	70.8	**71.4**	-	89.1	72.1	**72.5**
MNLI-m (Acc.)	83.2	**83.8**	82.9	**83.5**	86.6	**87.6**	86.7	**87.9**
MNLI-mm (Acc.)	83.4	**84.1**	82.8	**83.2**	-	87.5	85.9	**87.7**
QNLI (Acc.)	90.4	**91.5**	90.1	**90.8**	92.3	**93.6**	92.7	**94.0**
RTE (Acc.)	68.3	**71.8**	65.4	**68.1**	70.4	**78.3**	70.1	**71.6**
Average	82.2	**82.9**	77.7	**79.2**	-	86.5	82.1	**83.3**

Table 2: The results on GLUE benchmark. All base models are pre-trained by this work. Every result of the dev set is the average score of 4 times finetuning with different random seeds. Scores of BERT large dev are from (Sun et al., 2019) and scores of BERT large test are from (Devlin et al., 2018).

System	Dev	
	EM	F1
BERT base (Single)	80.8	88.5
BERT large (Single	84.1	90.9
BERT large (Single+DA)	84.2	91.1
KT-NET	85.2	91.7
StructBERT Large (Single)	85.2	92.0
SegaBERT base (Single)	83.2	90.2
SegaBERT large (Single)	85.3	92.4

Table 3: Evaluation results of SQUAD v1.1.

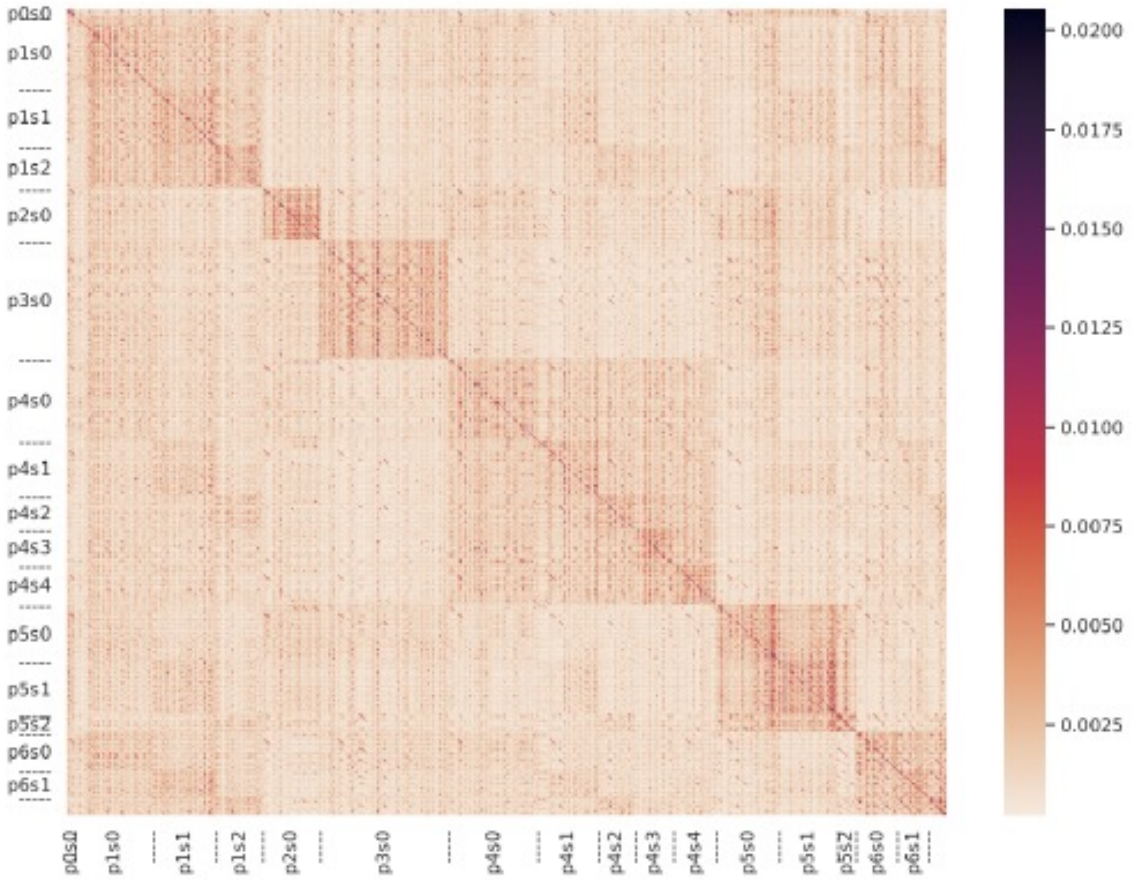
System	Dev	
	EM	F1
BERT base	72.3	75.6
BERT base (ours)	75.4	78.2
SegaBERT base	76.3	79.2
BERT large	78.7	81.9
BERT large wwm	80.6	83.4
SegaBERT large	81.8	85.2

Table 4: Evaluation results of SQUAD v2.0.

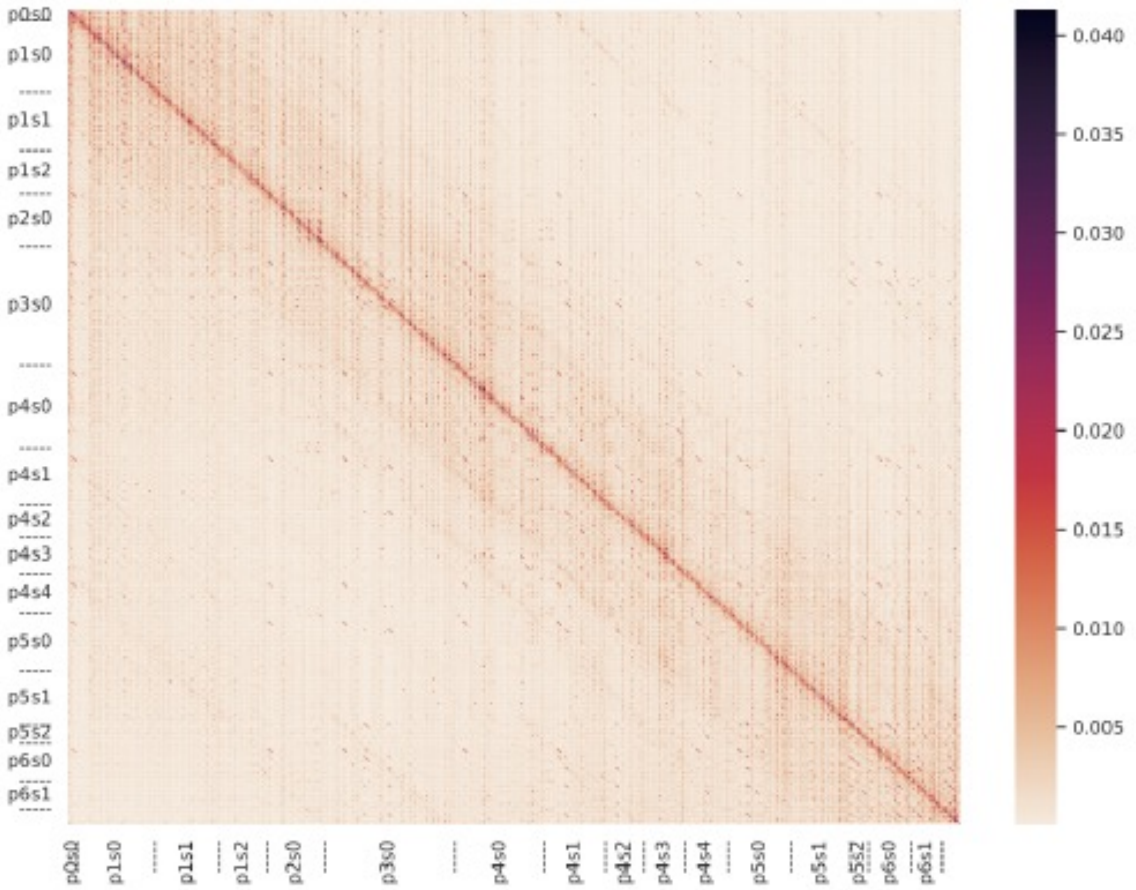
F1 = 2 (P*R) / (P+R), P is precision, R is recall, all in percentage, EM – exact match

Model	#Param.	PPL
LSTM+Neural cache (Grave et al., 2017)	-	40.8
Hebbian+Cache (Rae et al., 2018)	-	29.9
Transformer-XL base, M=150 (Dai et al., 2019)	151M	24.0
Transformer-XL base, M=150 (ours)	151M	24.4
SegaTransformer-XL base, M=150	151M	**22.5**
Adaptive Input (Baevski and Auli, 2019)	247M	18.7
Transformer-XL large, M=384 (Dai et al., 2019)	257M	18.3
Compressive Transformer, M=1024 (Rae et al., 2020)	257M	17.1
SegaTransformer-XL large, M=384	257M	**17.1**

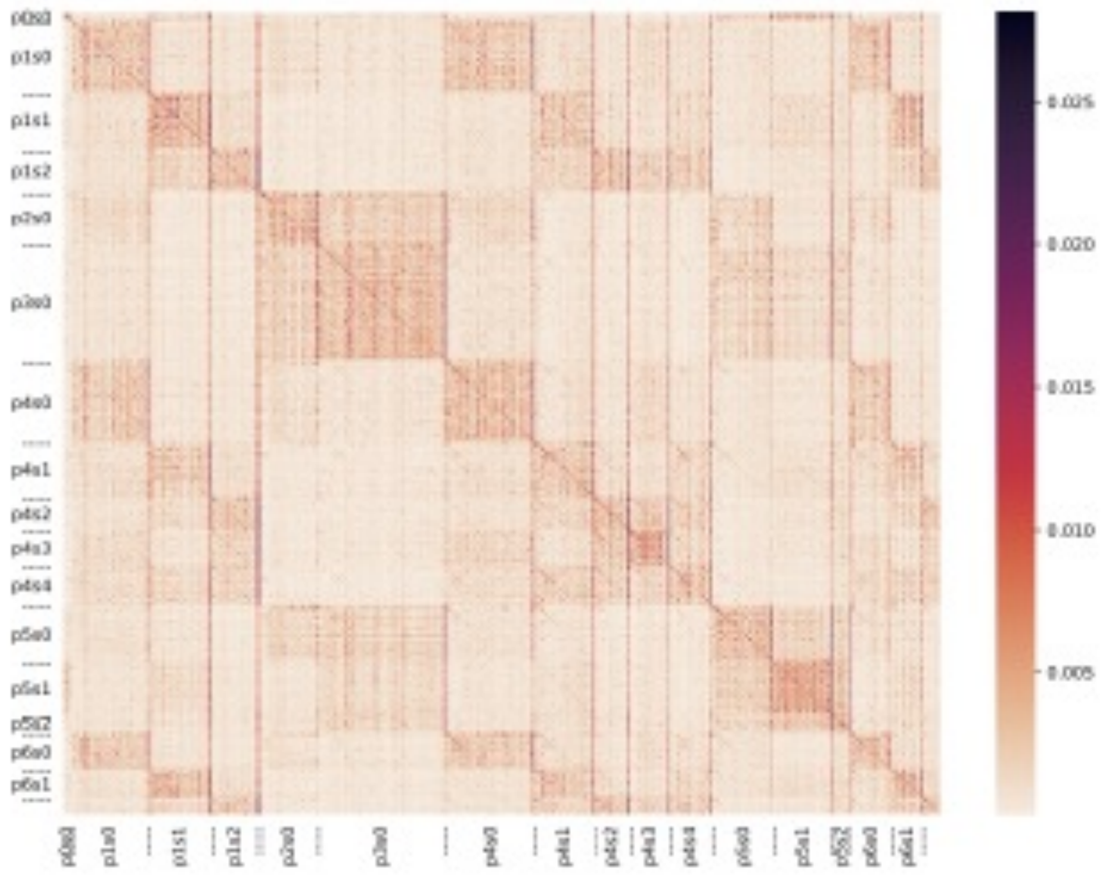
Table 1: Comparison with Transformer-XL and competitive baseline results on WikiText-103.



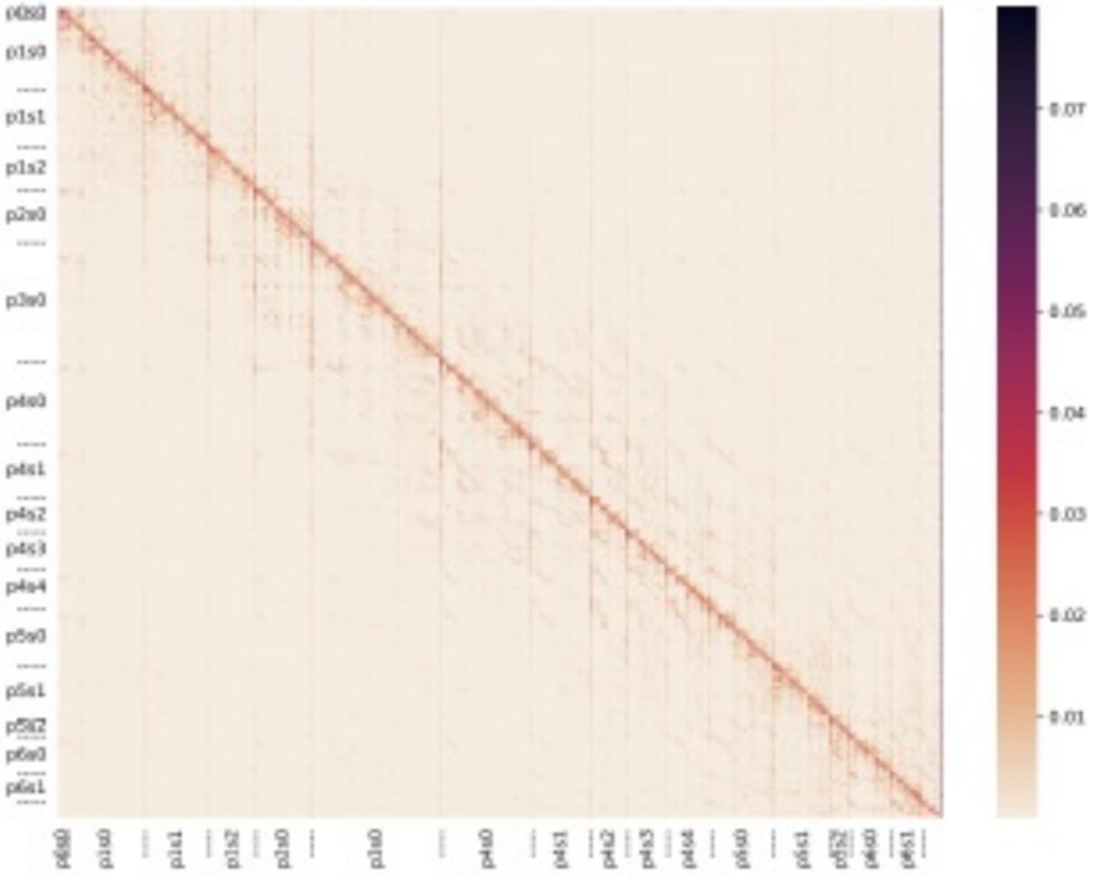
(a) SegaBERT-Layer 1



(b) BERT-Layer 1

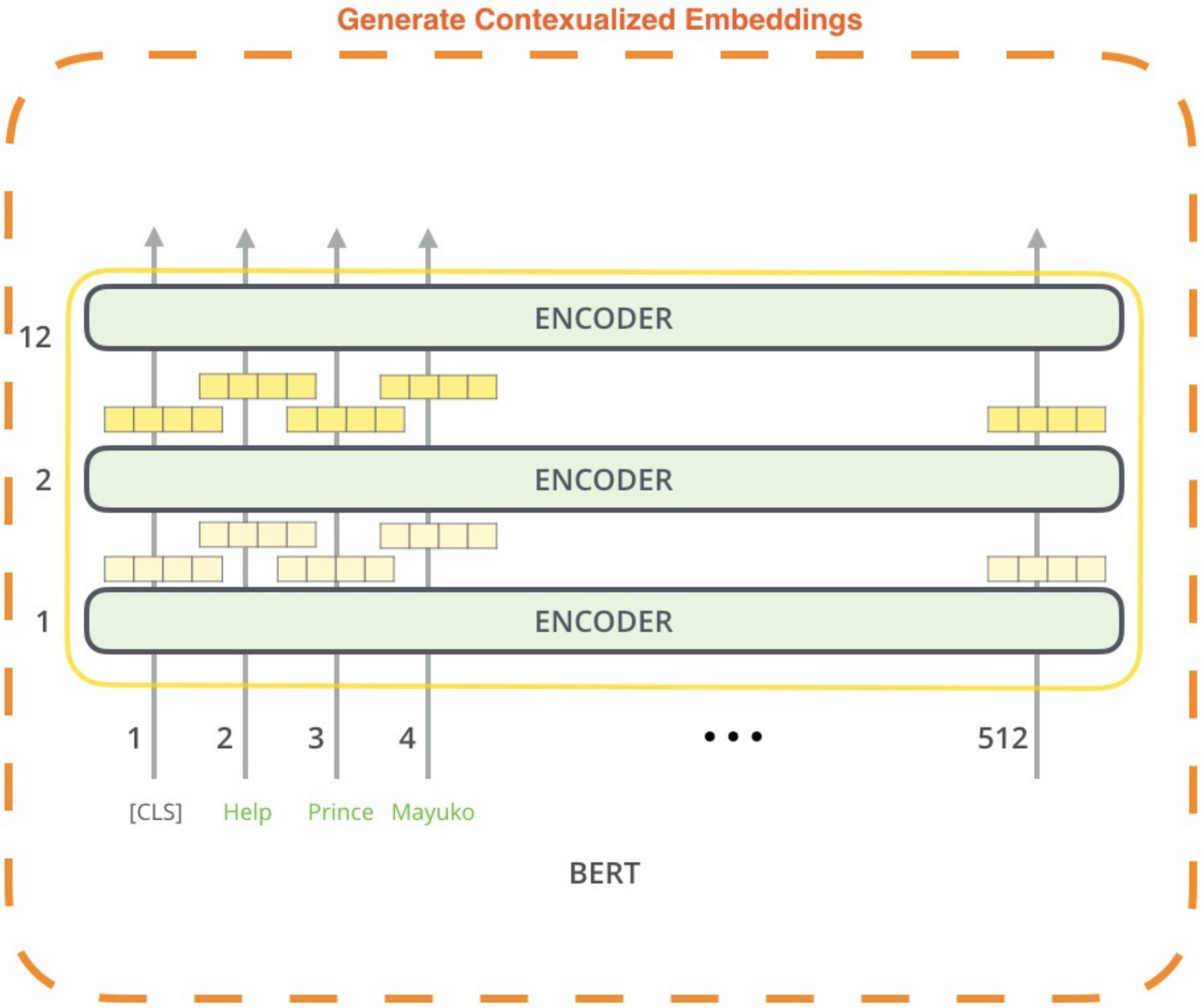


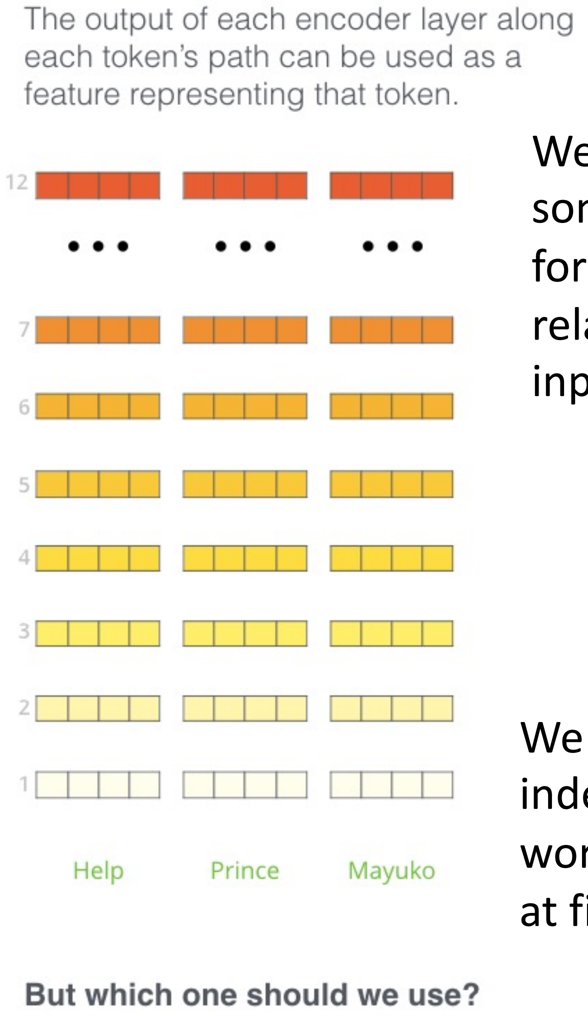
(a) SegaBERT-Layer 6



(b) BERT-Layer 6

Feature Extraction

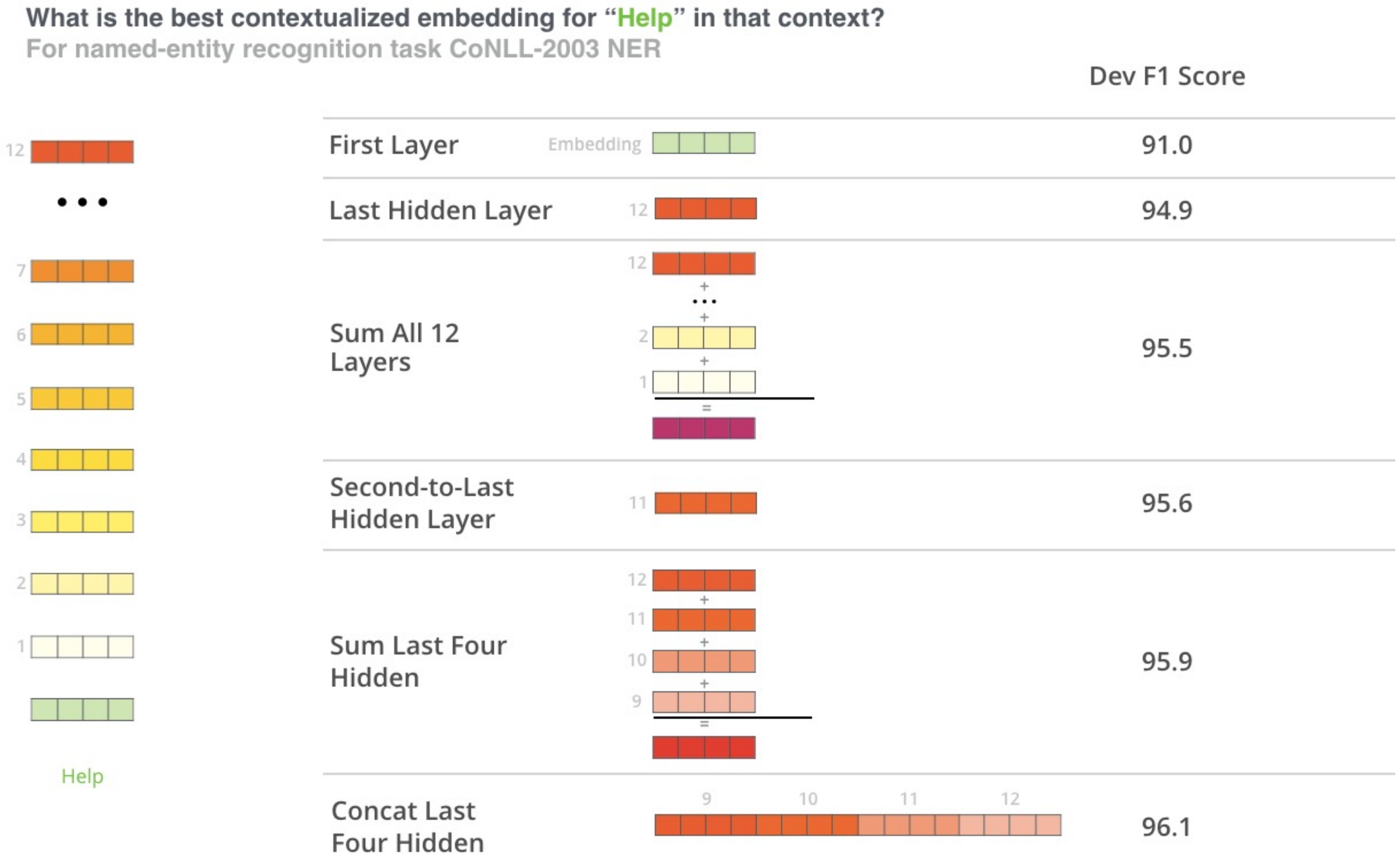




We end up with some embedding for each word related to current input

We start with independent word embedding at first level

Feature Extraction, which embedding to use?



What we have learned

1. Model size matters (345 million parameters is better than 110 million parameters).

2. With enough training data, more training steps imply higher accuracy

3. Key innovation: Learning from unannotated data.

Literature & Resources for Transformers

Resources:

OpenAI GPT-2 implementation: https://github.com/openai/gpt-2

BERT paper: J. Devlin et al, BERT, pretraining of deep bidirectional transformers for language understanding. Oct. 2018.

ELMo paper: M. Peters, et al, Deep contextualized word representation, 2018

ULM-FiT paper: Universal language model fine-tuning for text classification. J. Howeard, S. Ruder., 2018

Jay Alammar, The illustrated GPT-2, https://jalammar.github.io/illustrated-gpt2/

Labs

- T-SNE - Worked Example 19.2 T-SNE on MNIST Data

- Generate word embeddings with BOW for spam classification