Natural Language Processing
Lecture 09

Qun Liu, Valentin Malykh
Huawei Noah's Ark Lab

Spring 2022
A course delivered at KFU, Kazan

Content

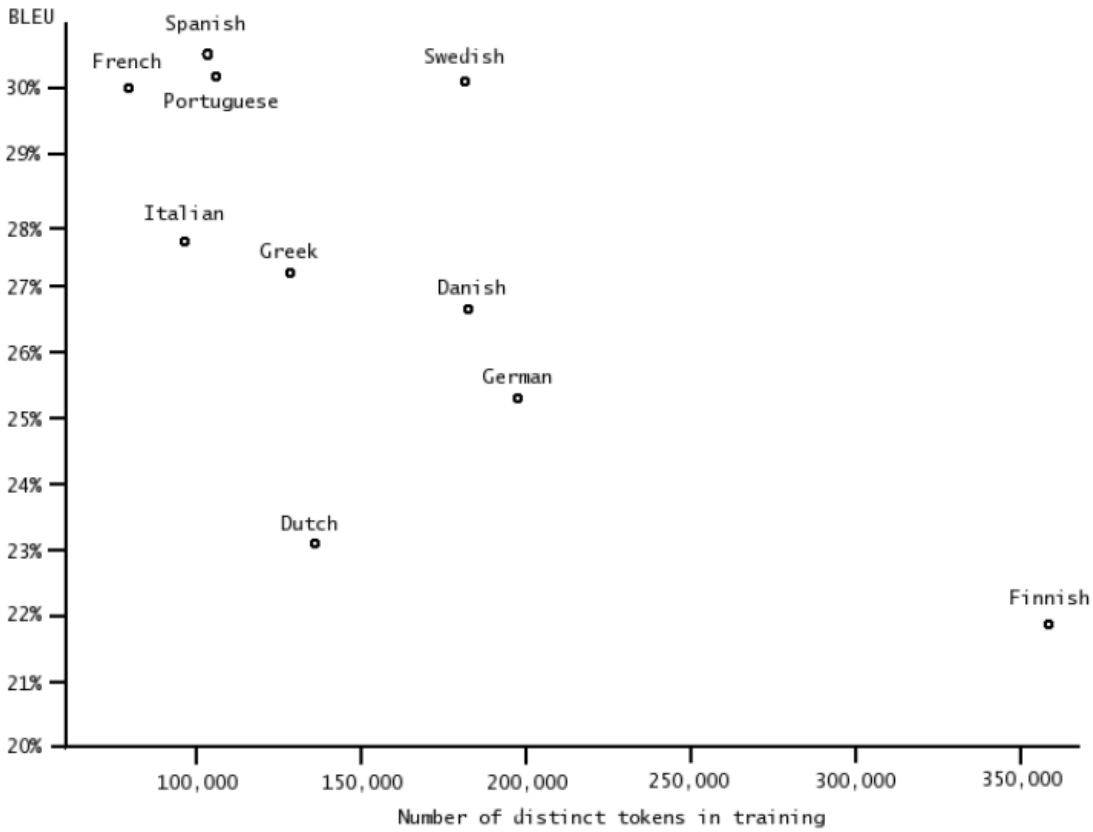
Content

Content

Open-vocabulary problem for NMT

- In NMT, due to the use of softmax in decoding, the output vocabulary is a closed set.
- Furthermore, a large size will make the decoding very slow because the softmax operation consumes huge computing resources.
- However, the vocabulary of a natural language is always an open set because new words emerge every day.
- Moreover, morphologically-rich languages such like finnish, Turkish, Arabic, etc. have very large vocabulary size compared with other languages.

Vocabulary Size vs. MT Performance



Vocabulary size vs. BLEU score when translating into English (which has about 65,000 distinct word forms) for SMT
Philipp Koehn, Europarl: A Parallel Corpus for Statistical Machine Translation, MT Summit 2005

Replace OOV words with the UNK symbol

- Practically, the vocabulary size of a NMT system is around 500k.
- In early NMT systems, all the out-of-vocabulary (OOV) words are replaced with a UNK symble (means UNKNOWN):
  - UNKs in source sentences will make the system not differentiat the rare words;
  - UNKs in the target sentences will make the system unreadable.

Content

Solutions to open-vocabulary problem

- Subword-level models
  - Byte-Pair Encoding (BPE)
  - Word Piece or Sentence Piece
- Character-level models

Bype-Pair Encoding (BPE)

- Originated from a compression algorithm
  - Replace a most frequent byte pair with a new byte
- BPE for NMT was proposed in 2016 and become very popular, not only in NMT but also in many other NN-based NLP tasks

> Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016.

Bype-Pair Encoding (BPE)

- Initialize **vocabulary** of subwords with all the basic characters
- Maintain a **dictionary** with all words and their frequencies calculated from the **corpus**
- Segment all the words in the **dictionary** into characters
- Repeat until the **vocabulary** size reaches a predefined limit:
  - Select the subword bigram with the highest frequency
  - Merge that bigram into a new subword and add it to the **vocabulary**
  - Update the segmented **dictionary** by replacing all the occurrence of that bigram with the newly added subword

BPE: build the vocabulary

- The dictionary:

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

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

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

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d **es**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

l o w</w>	5
l o w e r</w>	2
n e w **est**</w>	6
w i d **est**</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d es **est</w>**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

BPE: build the vocabulary

- The dictionary:

lo w</w>	5
lo w e r</w>	2
n e w est</w>	6
w i d est</w>	3

- The vocabulary:

l o w</w> w e r</w> n s t</w> i d es est</w> **lo**

Note: 'w</w>','r</w>' and 't</w>' are single characters which are different from 'w', 'r' and 't'.

More about BPE

- Do deterministic longest piece segmentation of words
- Segmentation is only within words after tokenization
- Advantages:
  - Automatically build vocabulary from corpus
  - Language agnostic
  - Unsupervised
  - Trade off between text length and vocabulary size
- A trick:
  - treat 't' and 't</w>' as different characters to keep the word boundary information

WordPiece and SentencePiece

- An alternative solution for subword segmentation
- Proposed by Google in 2012:

> Schuster, Mike, and Kaisuke Nakajima. Japanese and korean voice search. ICASSP 2012.

- Instead of merging the bigram with highest frequence in BPE, WordPiece merge the bigram to maximizing the language model log likelihood of the corpus.
- SentencePiece is similar with WordPiece but applied on raw texts without tokenization, while whitespace is treated as a special character (_).

Subword-level NMT

- Subword-level NMT is almost the same as word-level NMT except for the preprocessing process for subword segmentation and the postprocessing process for subword combination.
- The subword combination is simple because the word boundary information is kept in the words.
- Subword-level NMT solves the open-vocabulary problem very well.
- Subword-level NMT outperforms word-level NMT significantly.
- Subword segmentation and combination has become a standard technique for NMT.

Content

RNN-based NMT Recap

- RNN-based NMT obtained great success:
  - Sequence-to-sequence model
  - RNN encoder and RNN decoder
  - Attention between target and source
  - Subword or character level encoding

Advantages of NMT

NMT provided a brand new paradigm for machine translation, which demonstrated huge advantages over previous approaches:

- NMT is a single model which is trained as a whole (end-to-end training), while an SMT system has many components each of which is trained against a separate object function.
- The translation quality of NMT are much better than that of SMT, especially in terms of fluency.
- NMT is good at learning from huge amount of data.
- Subword or character level NMT provides an elegant mechanism to deal with morphologically rich languages.

Improvement and extension of NMT

Research of NMT is exploding in recent years and great progress has been made:

- Transformer-based NMT
- Convolution-based NMT
- Multilingual NMT
- Multimodal NMT
- Unsupervised NMT

Transformer-based NMT has replaced RNN-based NMT and become a new state-of-the-art approach.

Transformer-based NMT

- Proposed by Google in 2017:

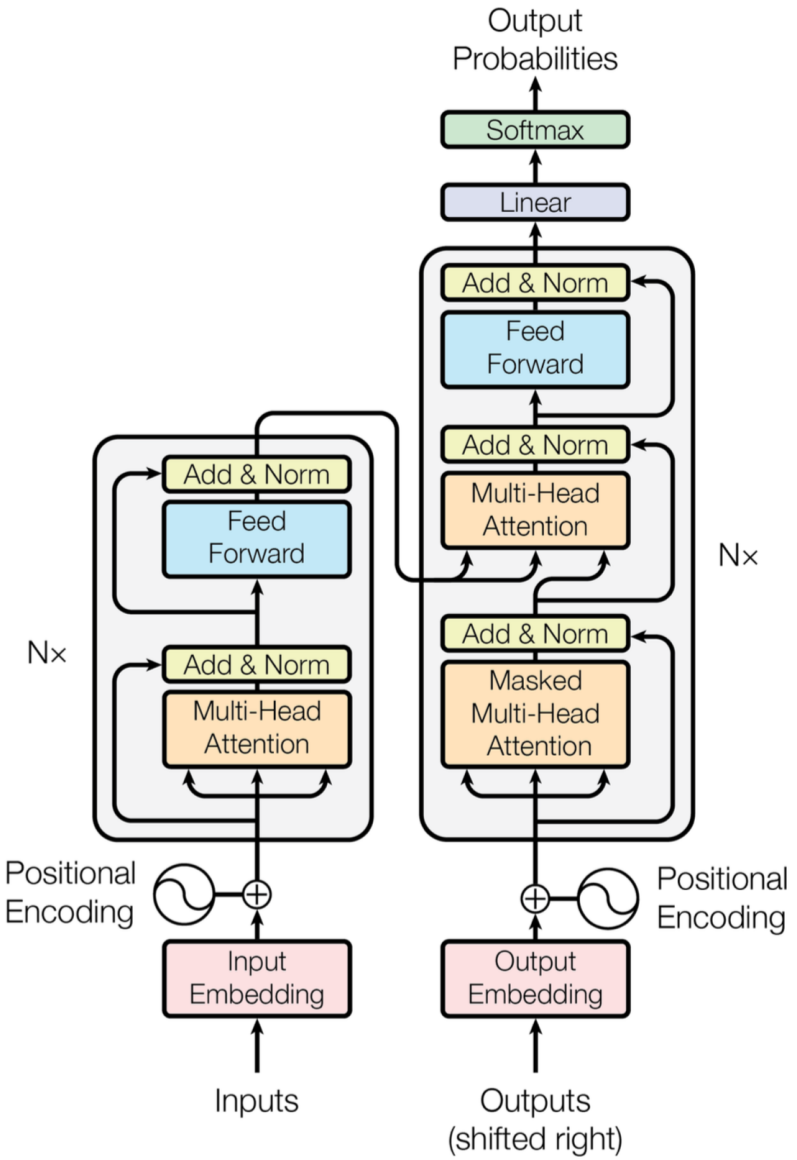
> Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NIPS 2017.

- Transformer is a new neural network architecture based soly on an attention mechanism, dispensing with recurrence and convolutions entirely.
- Transformer-based NMT is the state-of-the-art of MT technologies.

Transformer NMT architecture

- Transformer NMT adopts a sequence-to-sequenct architecture.
- In this section, we use figures from Jay Alammar's blog to illustrate the transformer NMT.

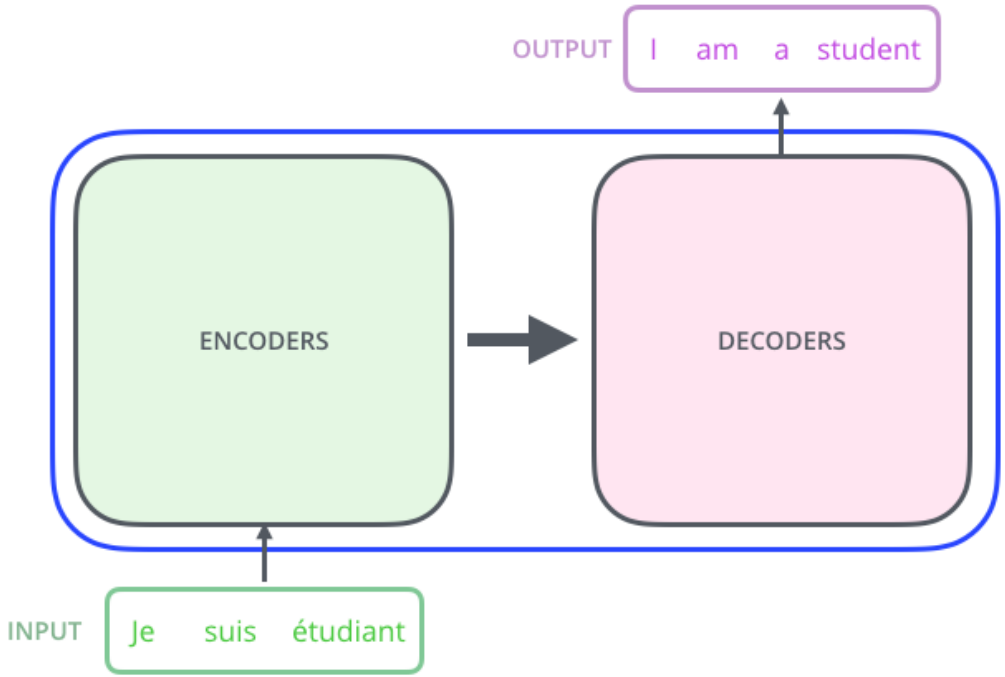
  `http://jalammar.github.io/illustrated-transformer/`



Content

A High-Level Look

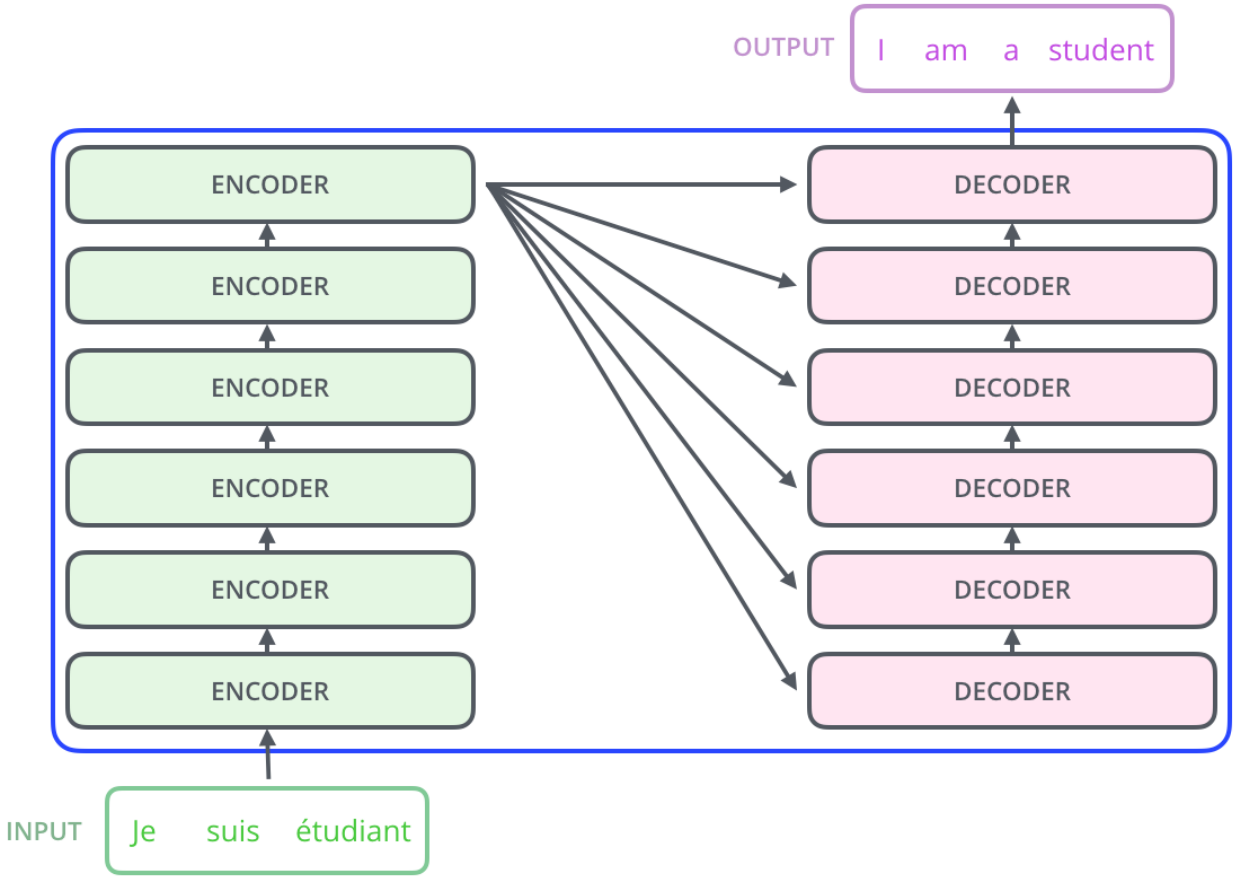
Transformer NMT is a seq2seq model.



A High-Level Look

The encoding component is a stack of encoders (6 in this paper).

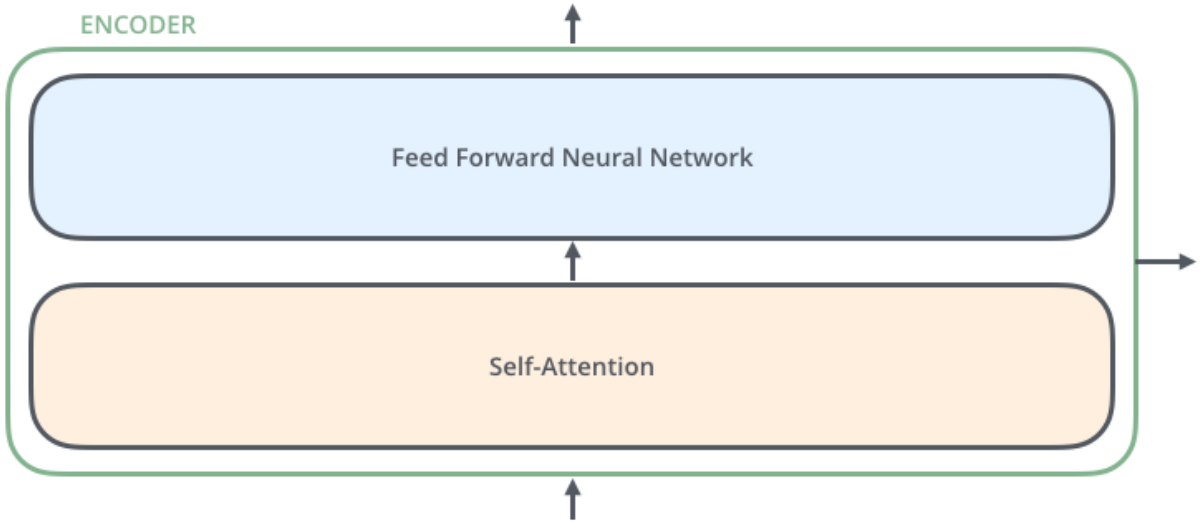
The decoding component is also a stack of decoders of the same number.

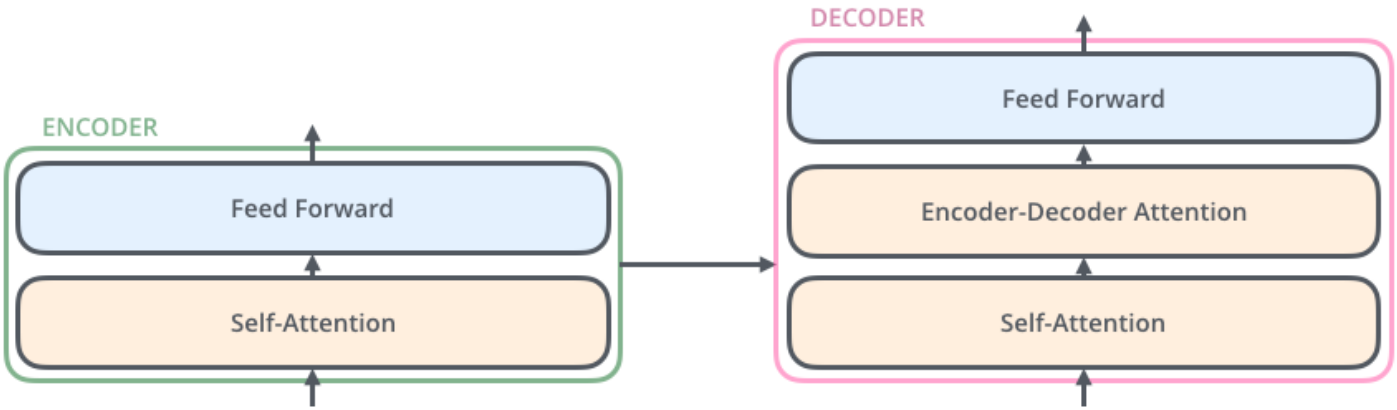


A High-Level Look

An encoder has two parts: FFNN and Self-Attention.

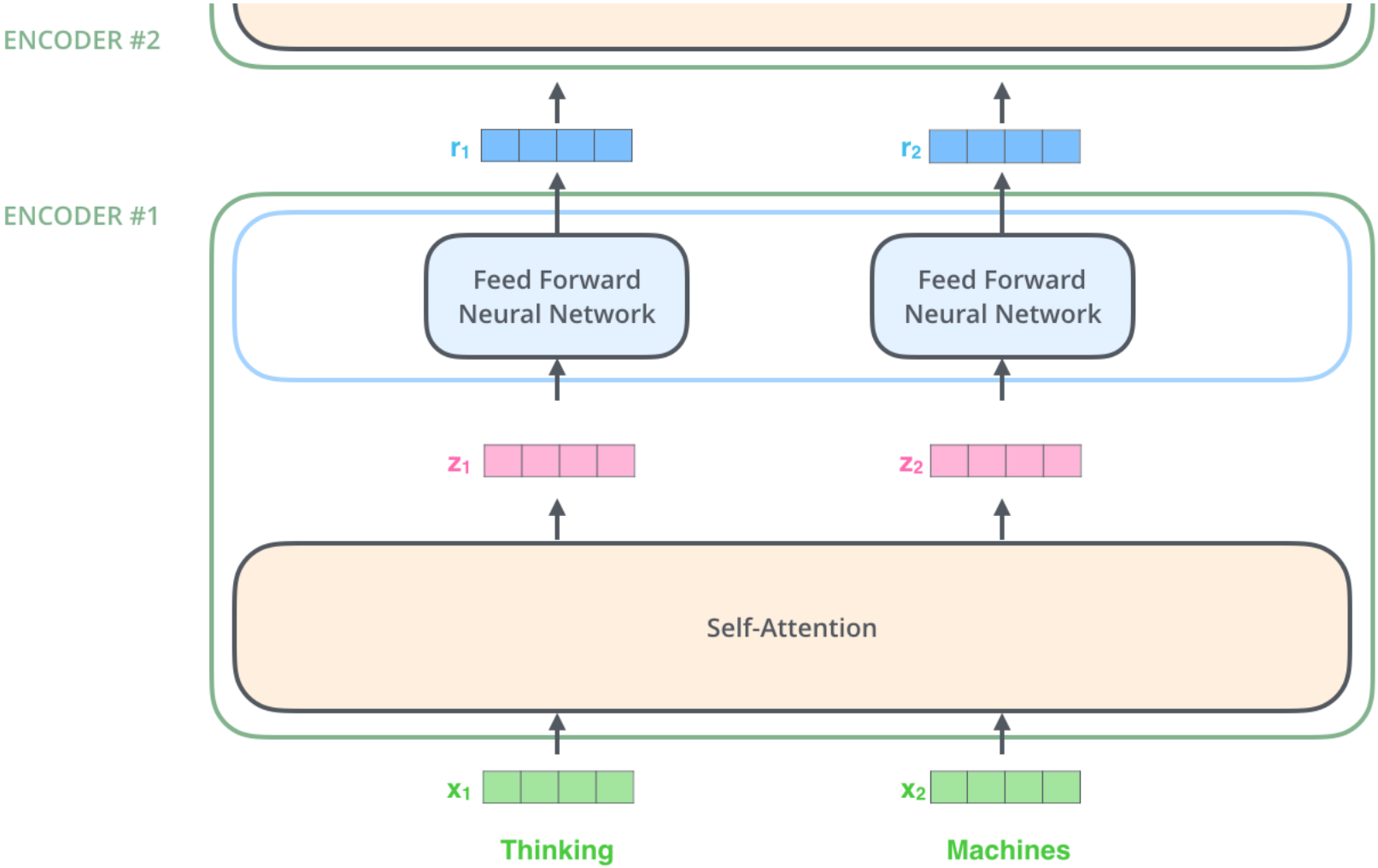
An decoder has one more part: Enc-Dec Attention.





Content

Self-Attention

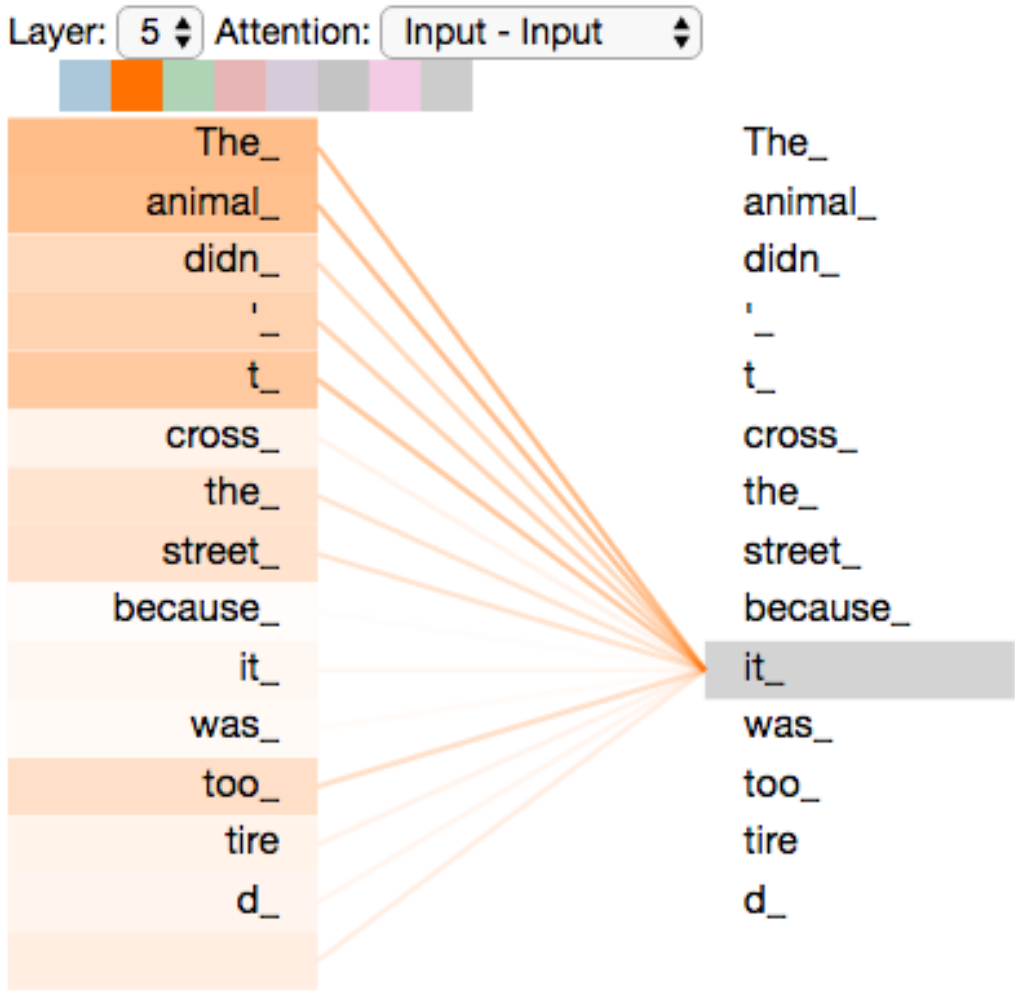


Self-attention: Visualization

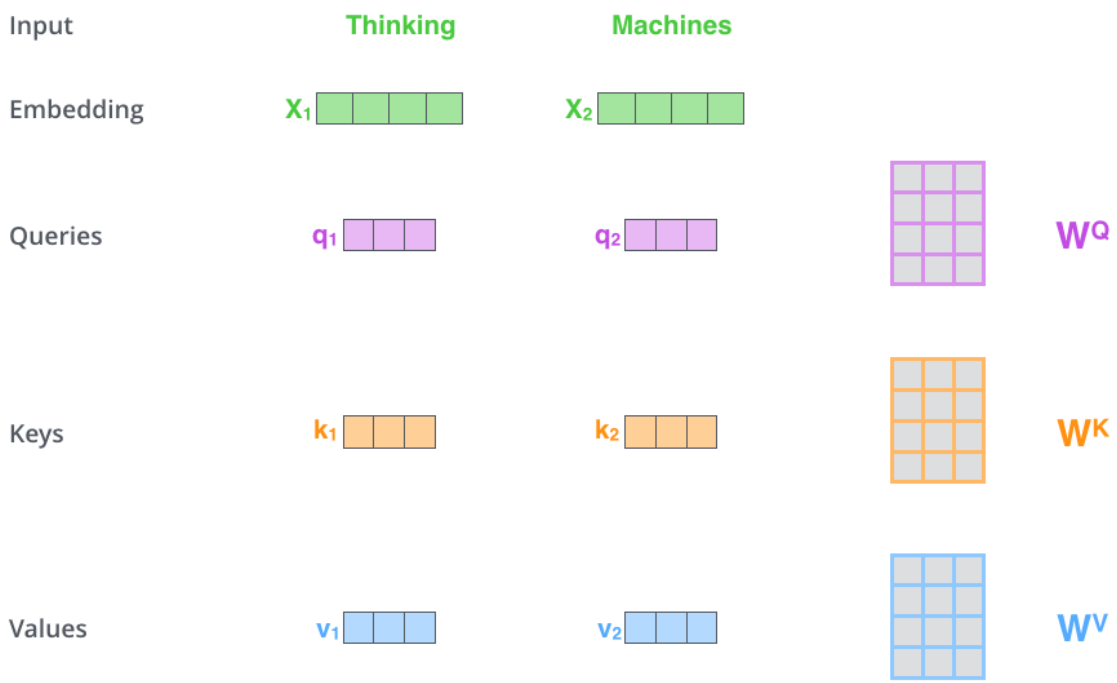
Example: `The animal didn't cross the street because it was too tired`

Associate `it` with `animal`

Look for clues when encoding

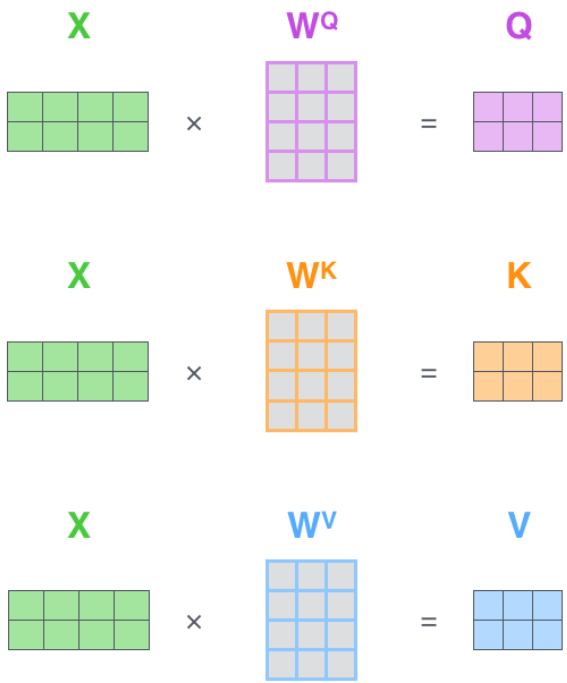


Self-attention: step 1 (create K,Q,V vectors)



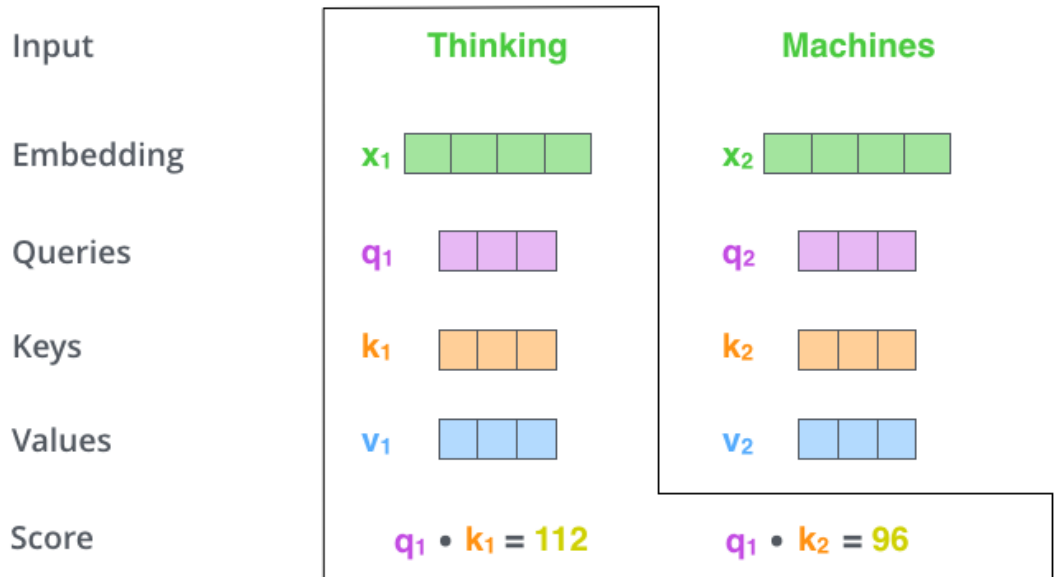
Multiplying x_1 by the W^Q weight matrix produces q_1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Self-attention: the calculation of K,Q,V vectors

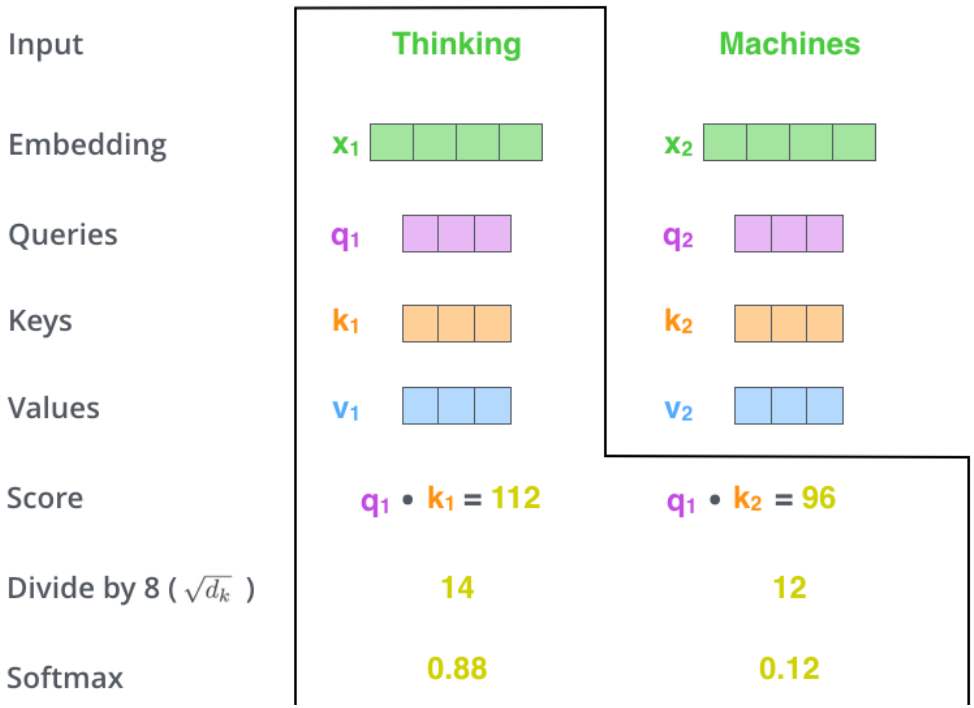


Every row in the X matrix corresponds to a word in the input sentence. We can see the difference in size of the embedding vector (4 boxes in the figure), and the $q/k/v$ vectors (3 boxes in the figure)

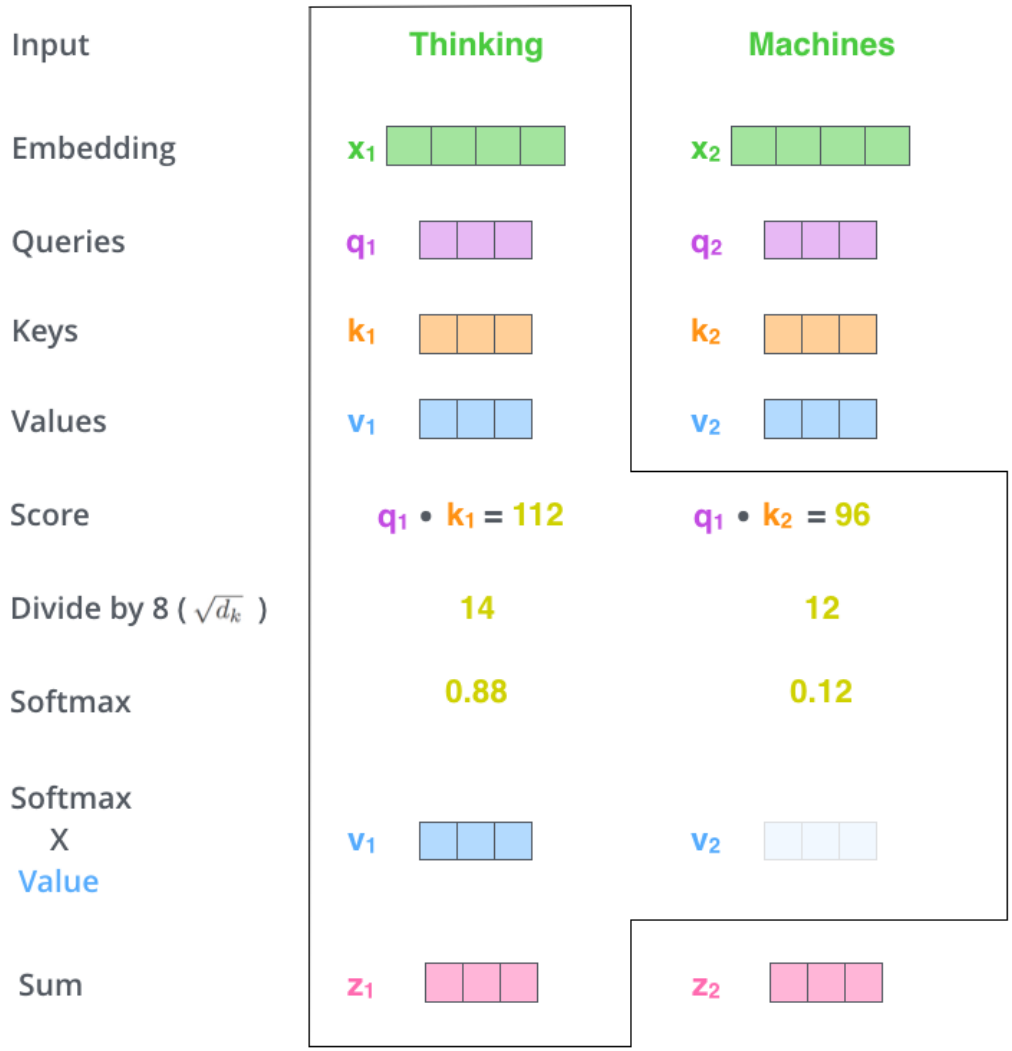
Self-attention: step 2 (calculate self-attention scores)



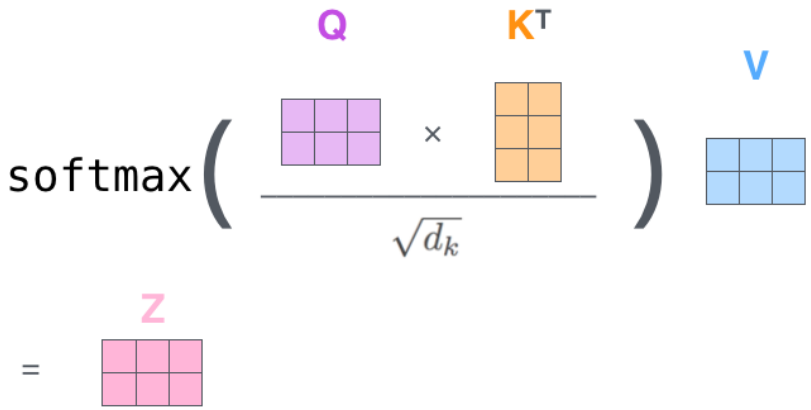
Self-attention: step 3 & 4 (normalize self-attention scores with softmax)



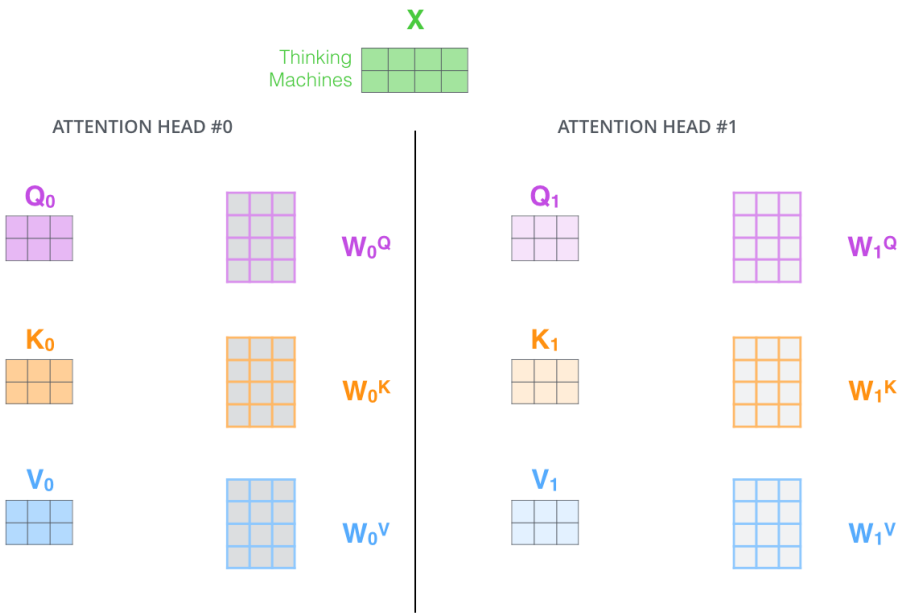
Self-attention: step 2 - 6 (weighted-sum of values from attented words)



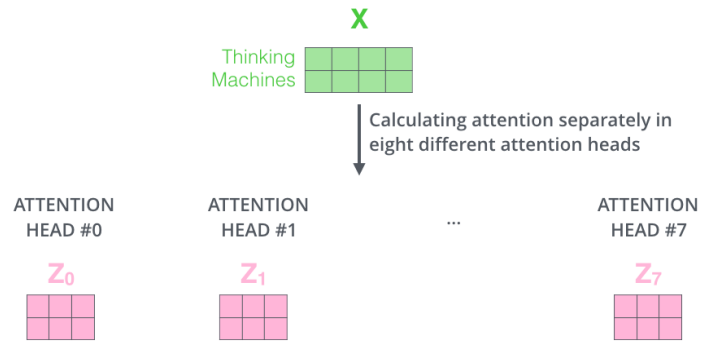
Self-attention: The self-attention calculation in matrix form



Multi-head attention



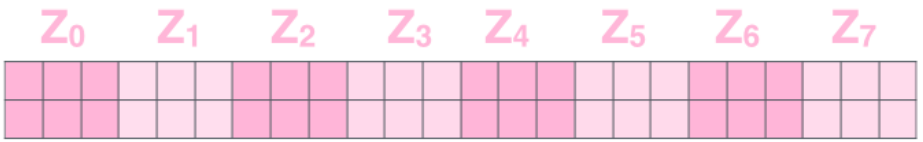
With multi-headed attention, we maintain separate $Q/K/V$ weight matrices for each head resulting in different $Q/K/V$ matrices.



We do the same self-attention calculation 8 times with different weight matrices, then we get 8 different Z matrices.

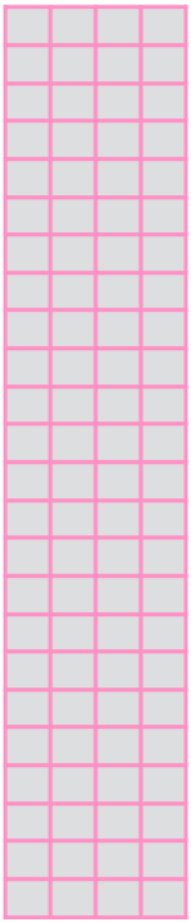
Multi-head attention

1) Concatenate all the attention heads

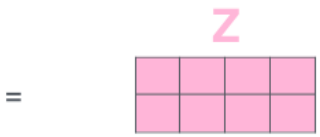


2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Overall picture of multi-head self-attention

1) This is our input sentence*

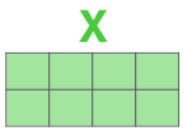
2) We embed each word*

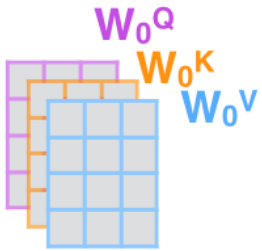
3) Split into 8 heads. We multiply X or R with weight matrices

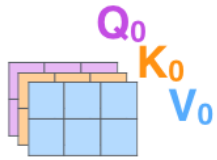
4) Calculate attention using the resulting Q/K/V matrices

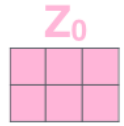
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

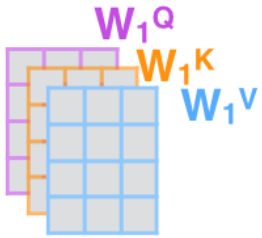
Thinking Machines

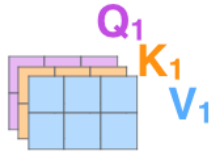


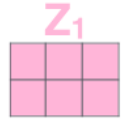




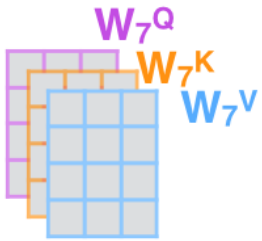


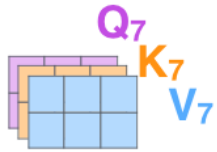


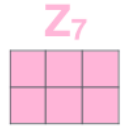


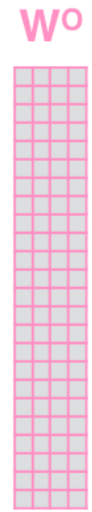


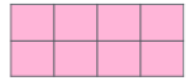
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one





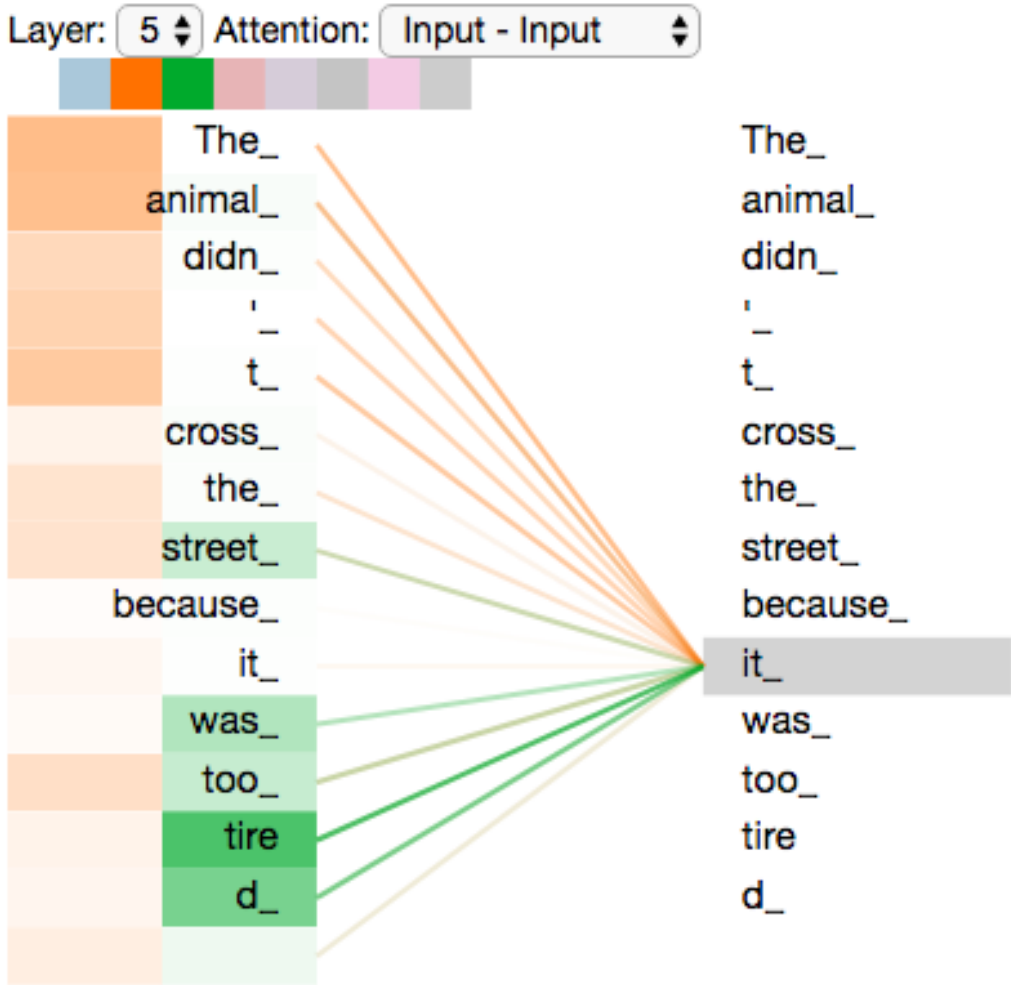






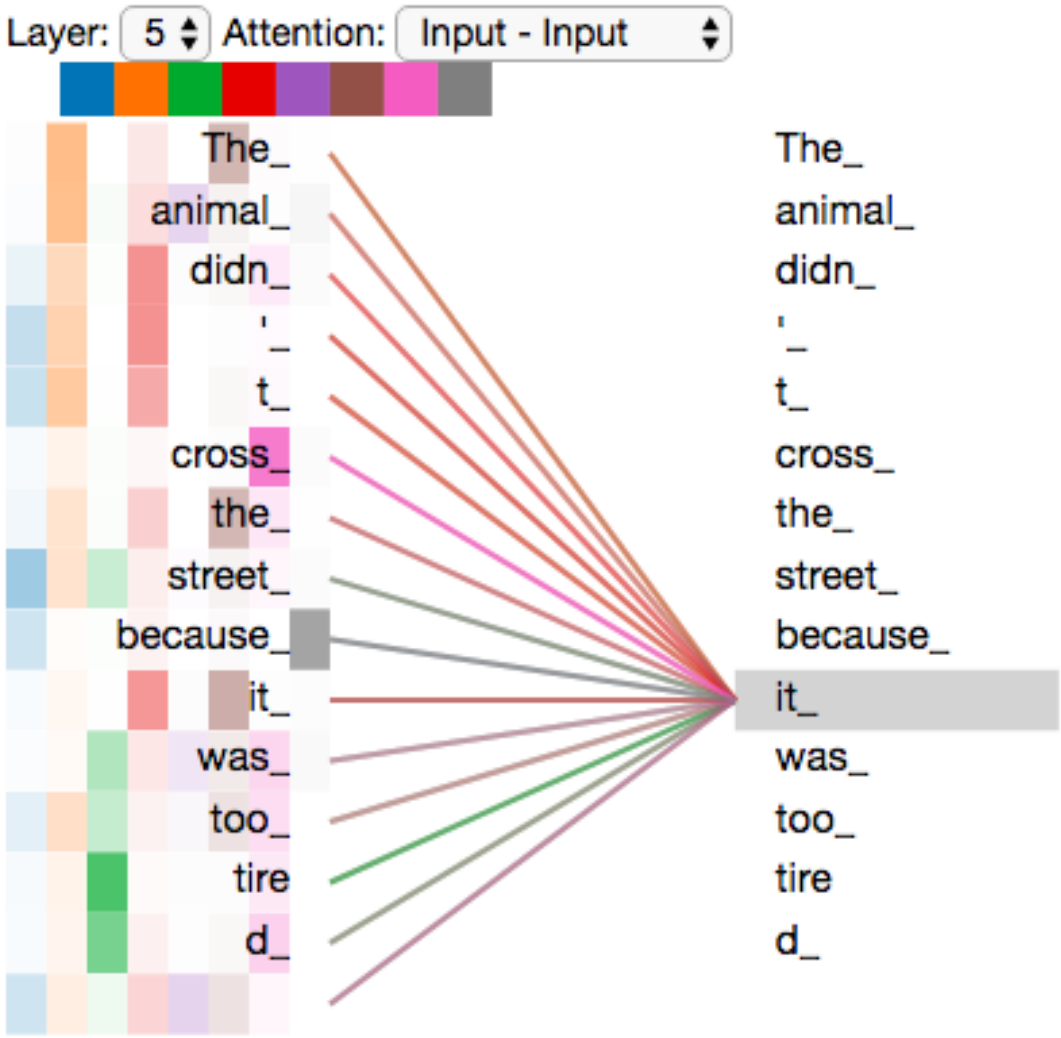
Self-attention: Visualization (Revisit with 2 heads)

As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" – in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

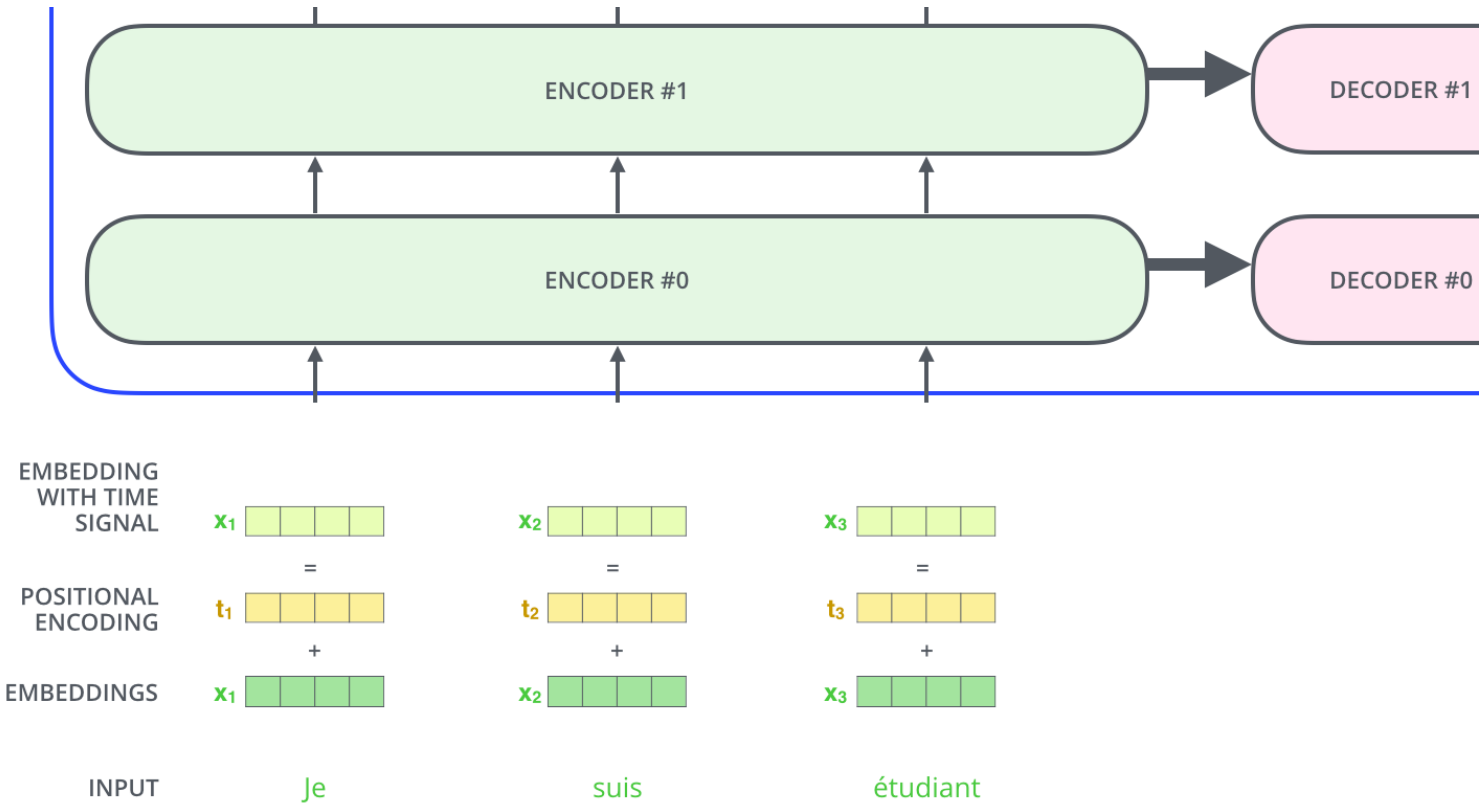


Self-attention: Visualization (Revisit with 8 heads)

If we add all the attention heads to the picture, however, things can be harder to interpret.



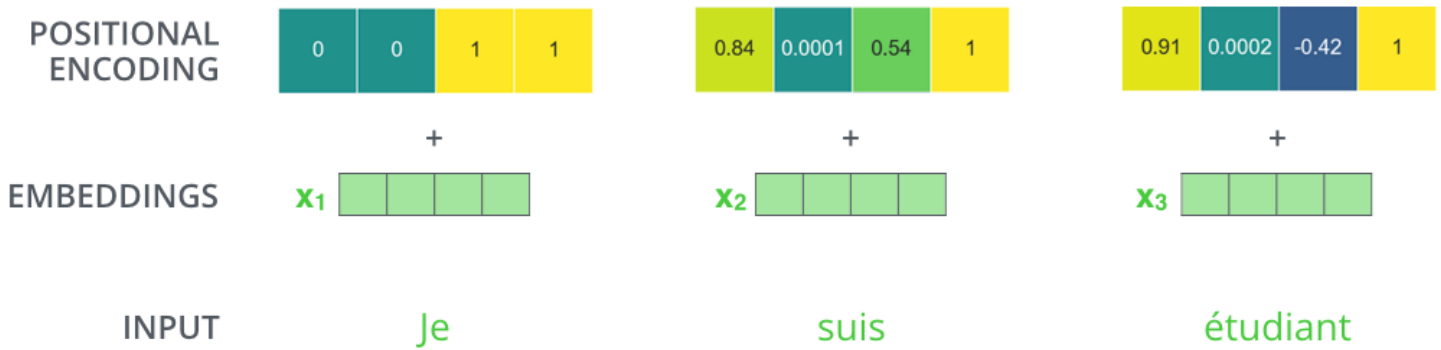
Positional Encoding: Representing Sequence Order



To give the model a sense of the order of the words, we add positional encoding vectors – the values of which follow a specific pattern.

Positional Encoding: Representing Sequence Order

If we assumed the embedding has a dimensionality of 4, the actual positional encodings would look like this:



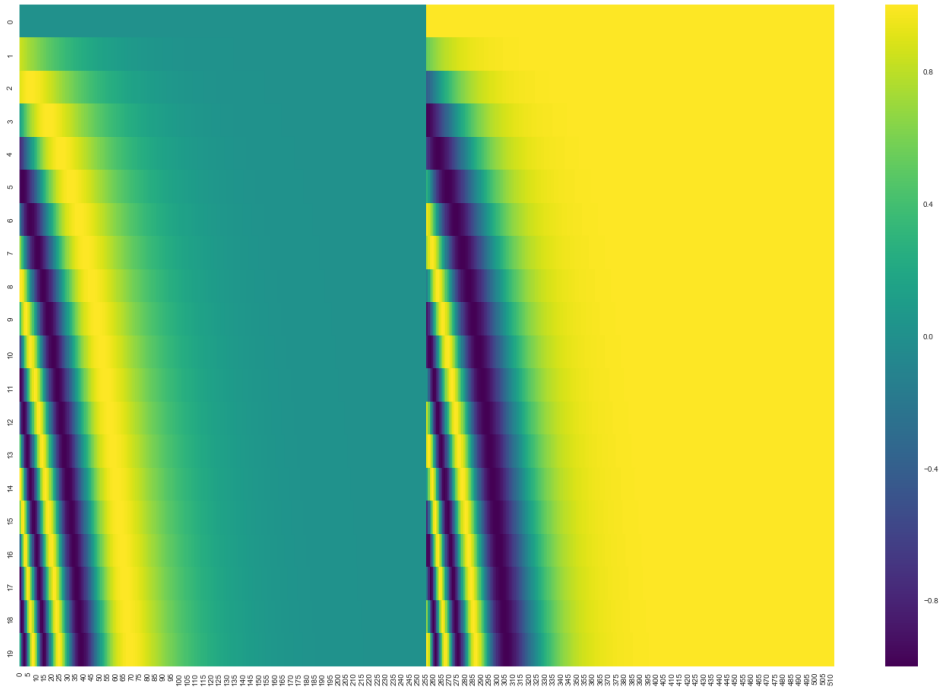
The formula for positional encoding is:

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

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

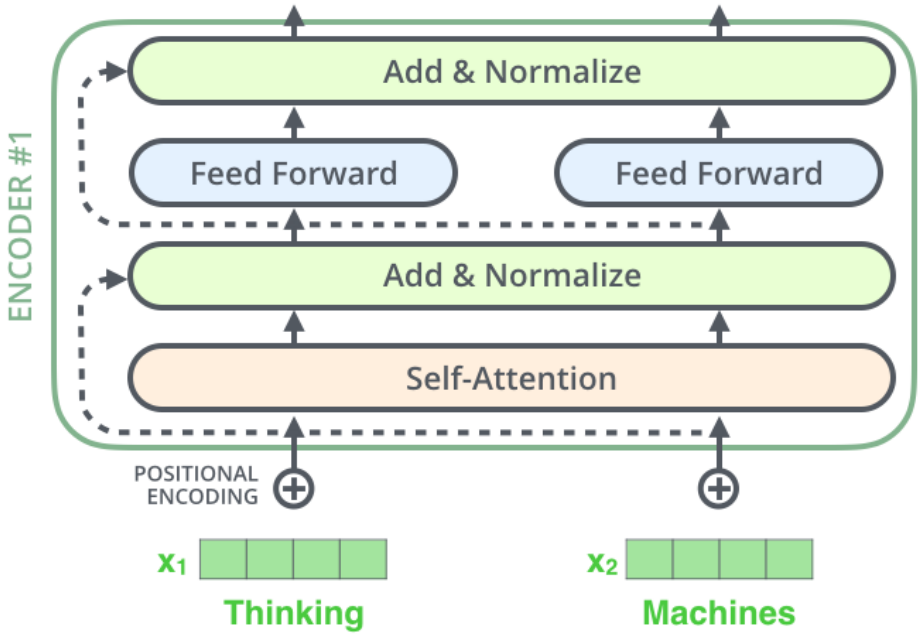
where *pos* is the position of the word in the input sequence, and *i* is the index of the dimension of the positional encoding vector.

Positional Encoding: Representing Sequence Order



A real example of positional encoding for 20 words (rows) with an embedding size of 512 (columns).

The residuals and layer normalization

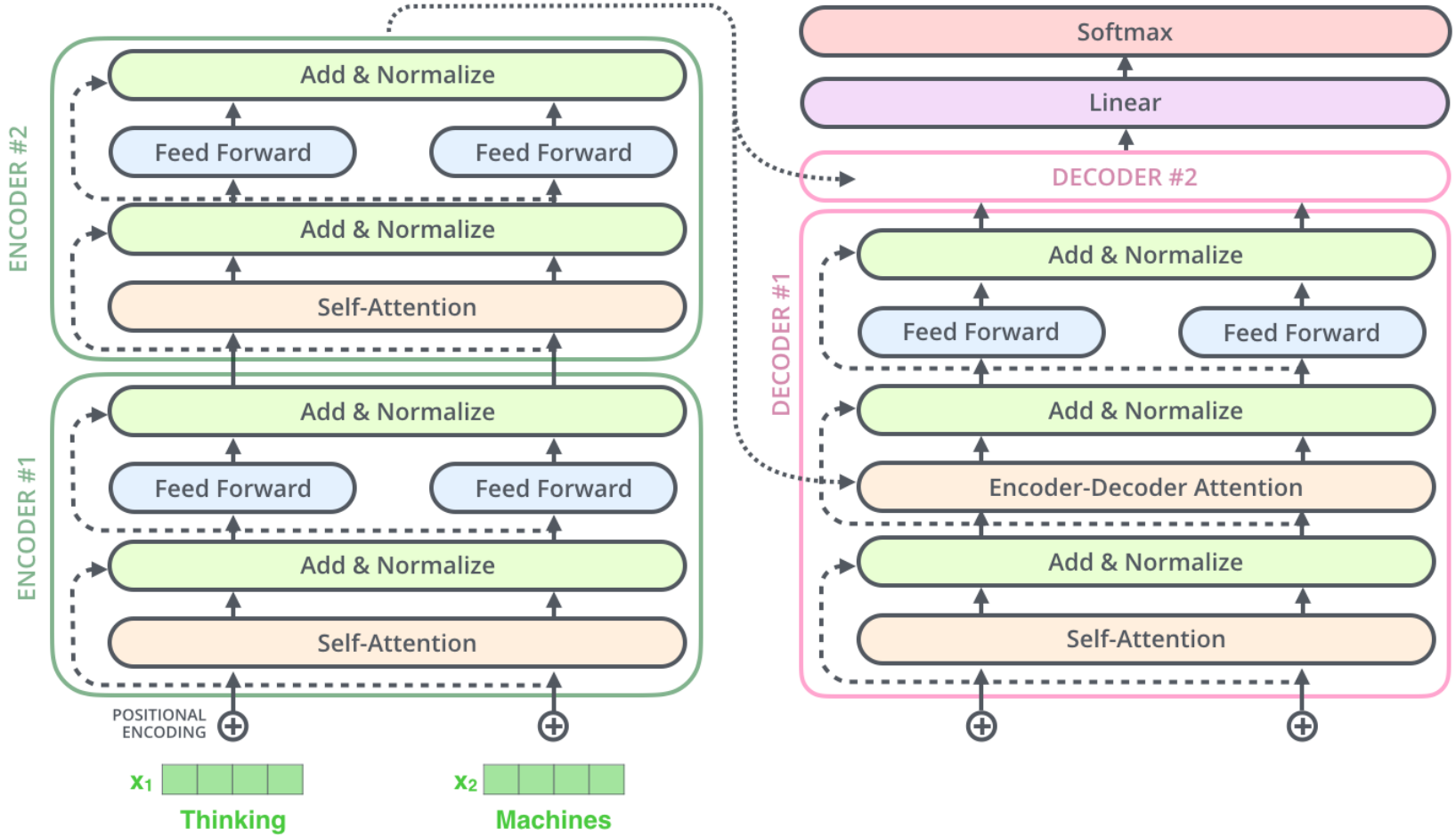


Each sub-layer (self-attention, ffnn) in each encoder has a residual connection around it, and is followed by a layer-normalization step.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Formula for the Feed Forward Network layer.

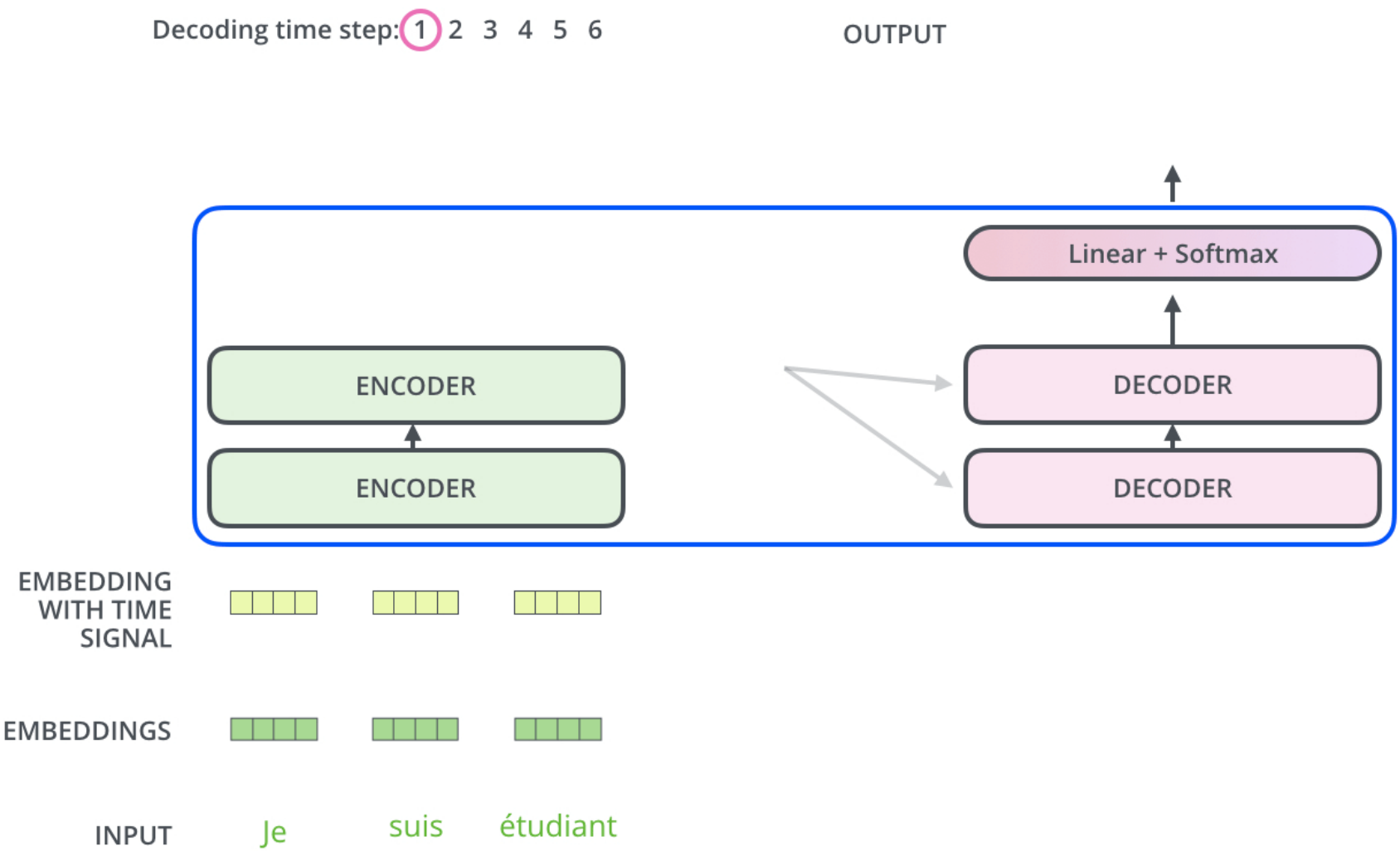
The residuals and layer normalizations



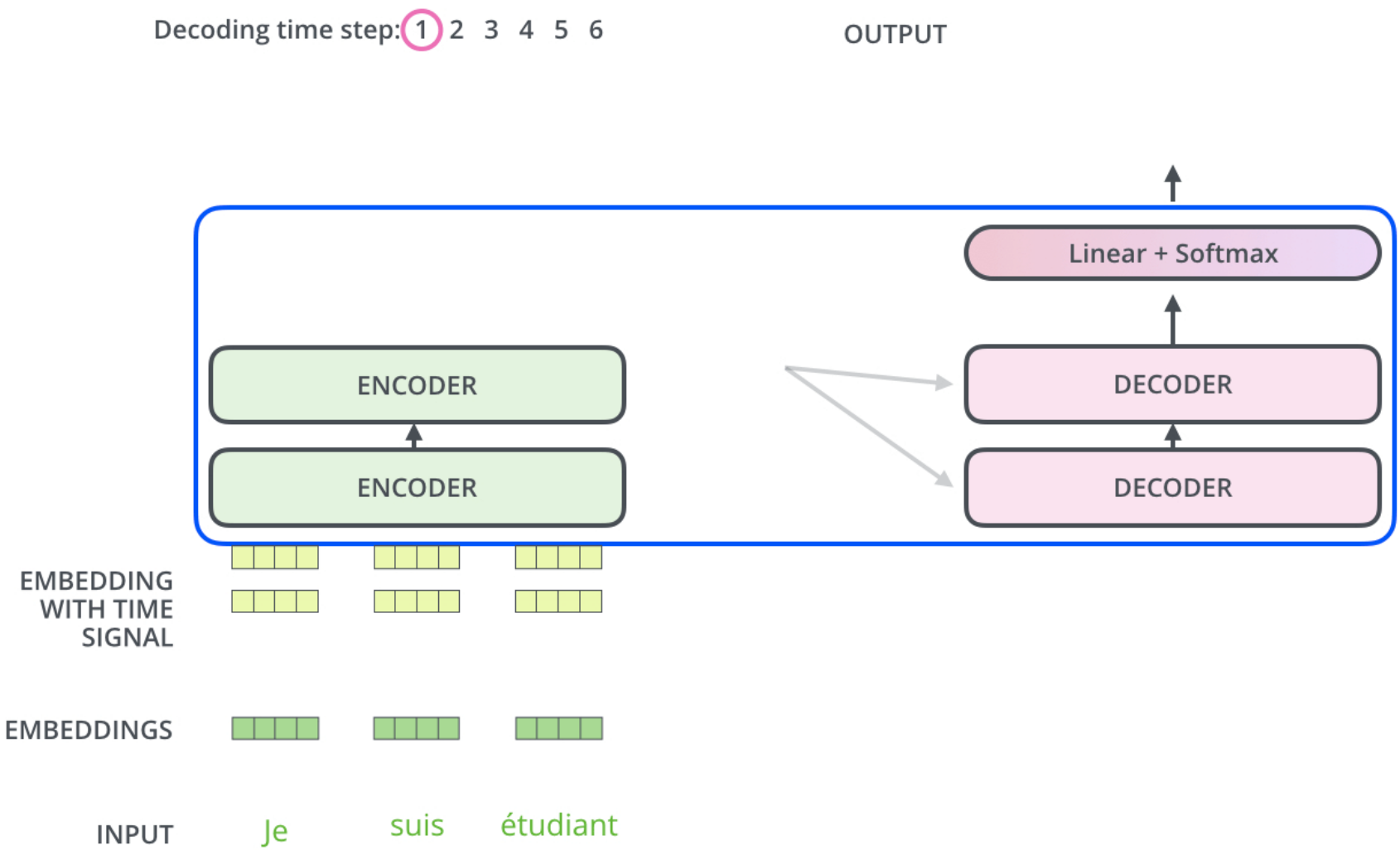
There are residual connections and layer normalizations for the sub-layers of the decoder as well.

Content

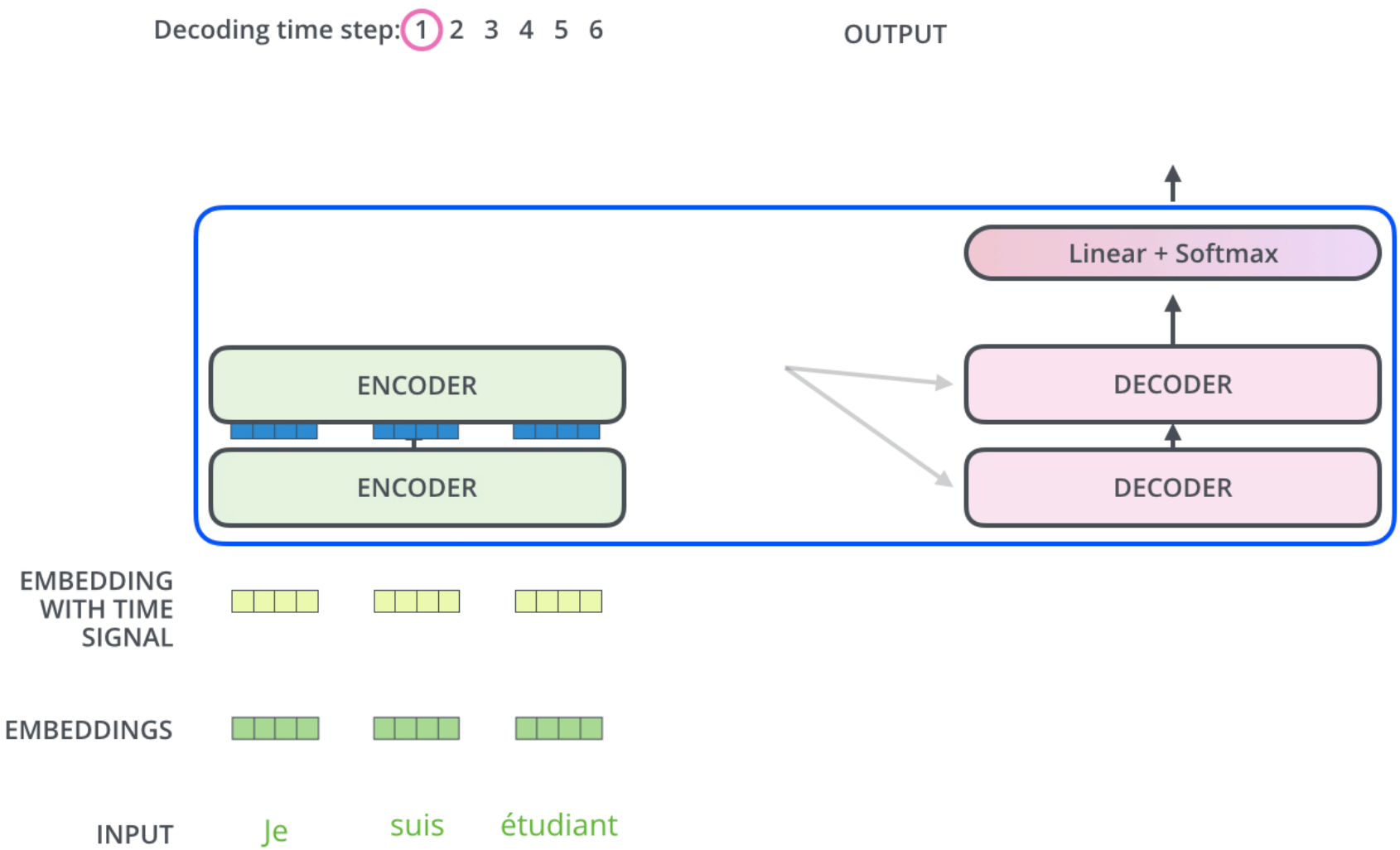
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

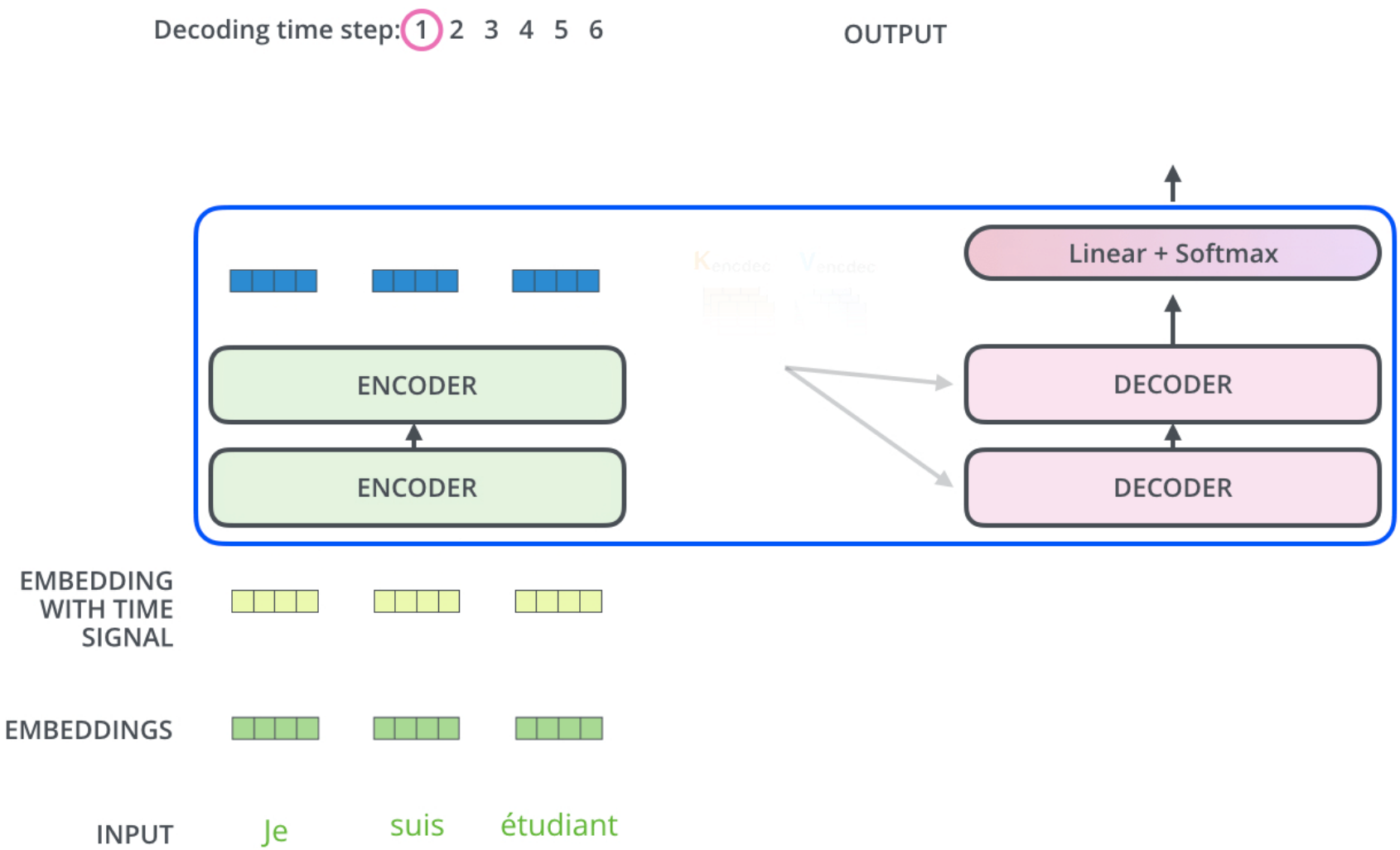
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

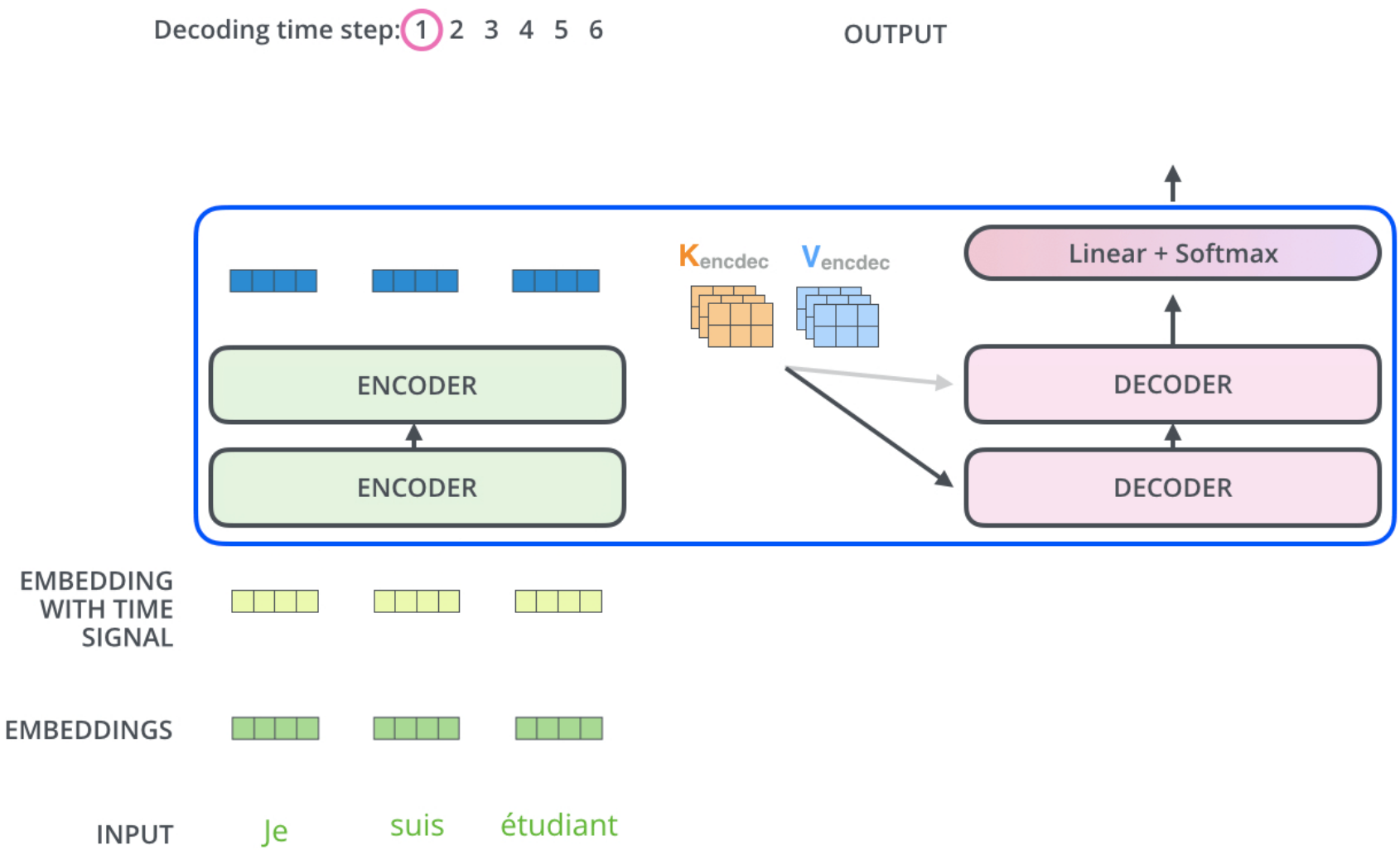
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

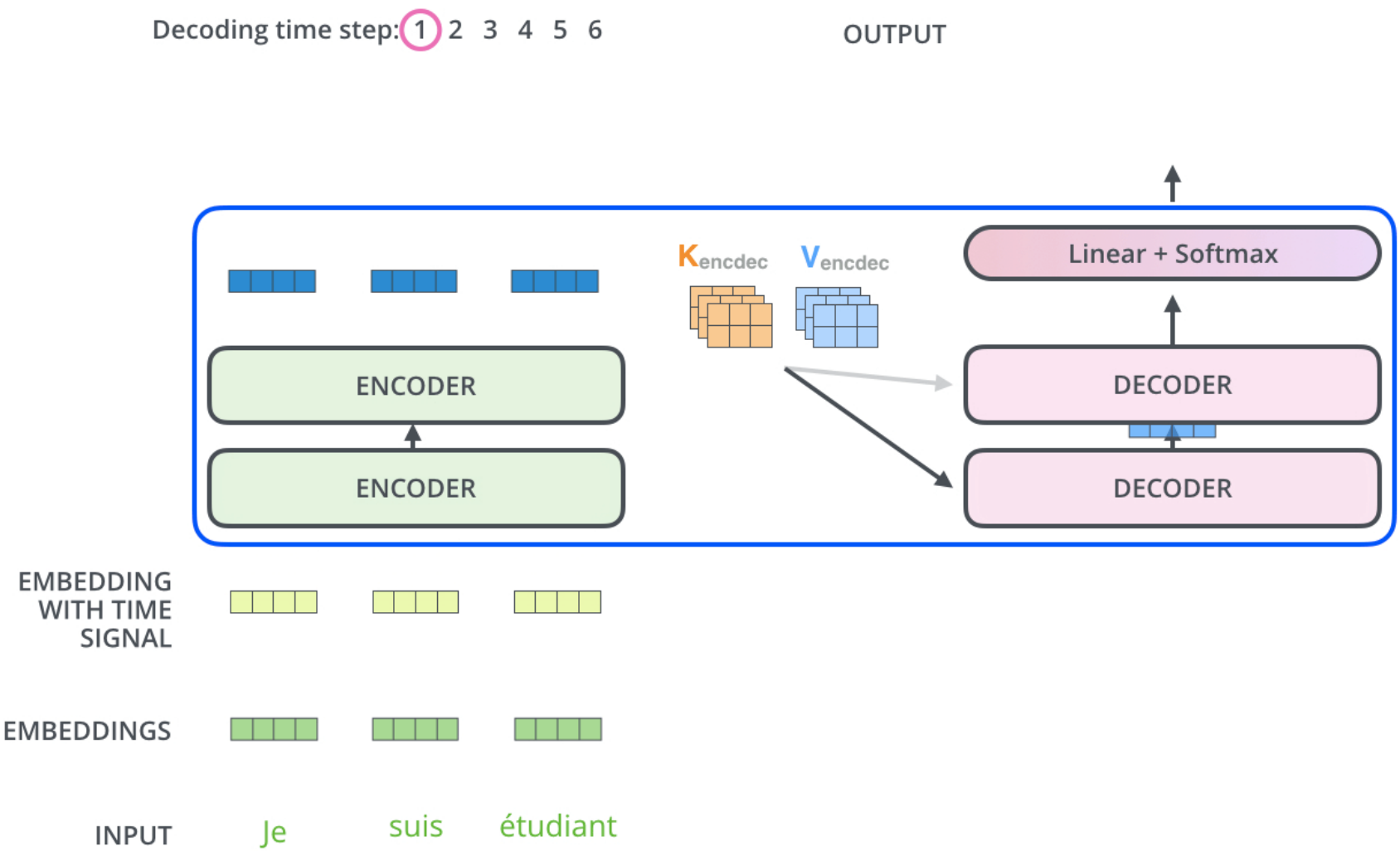
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

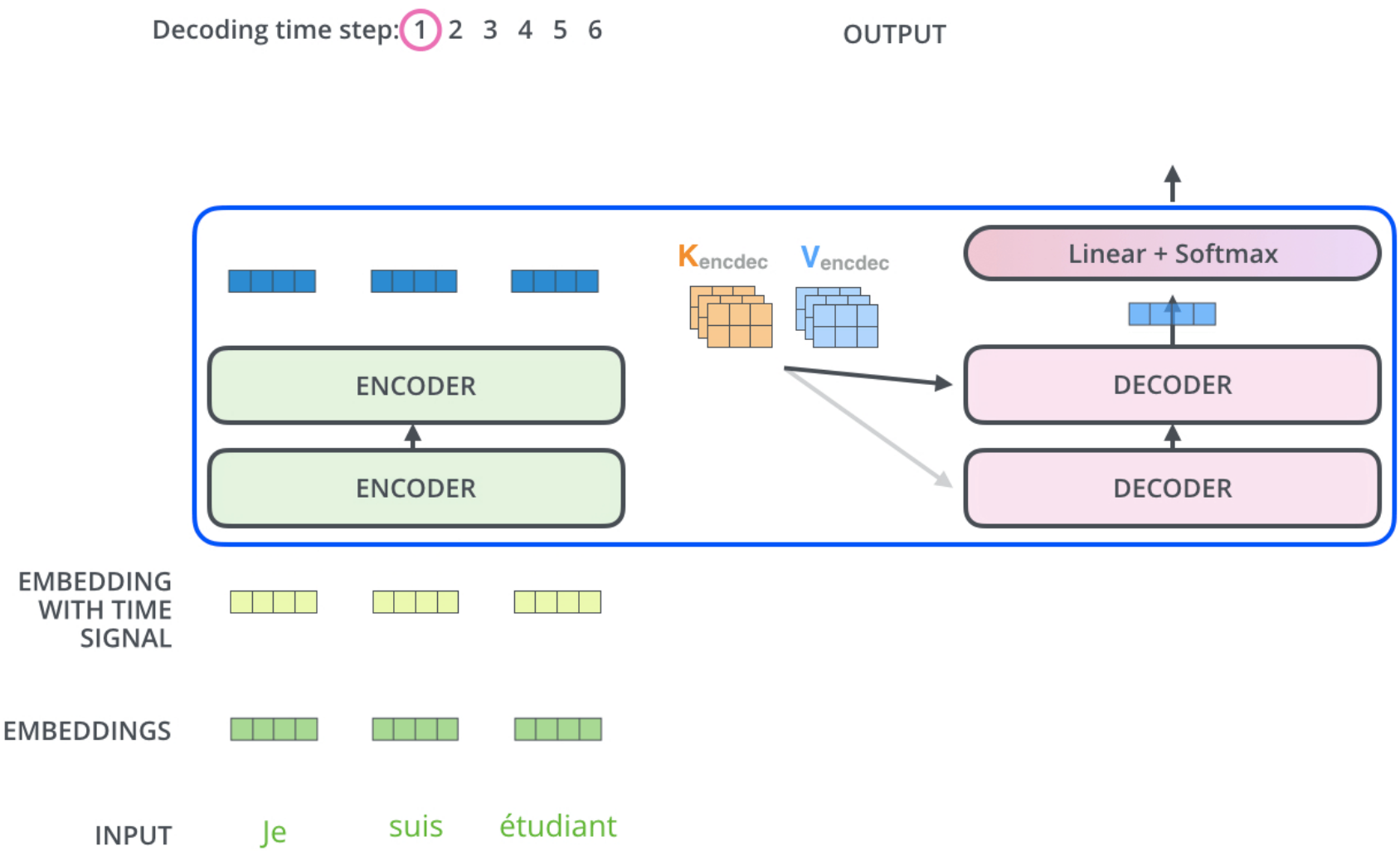
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

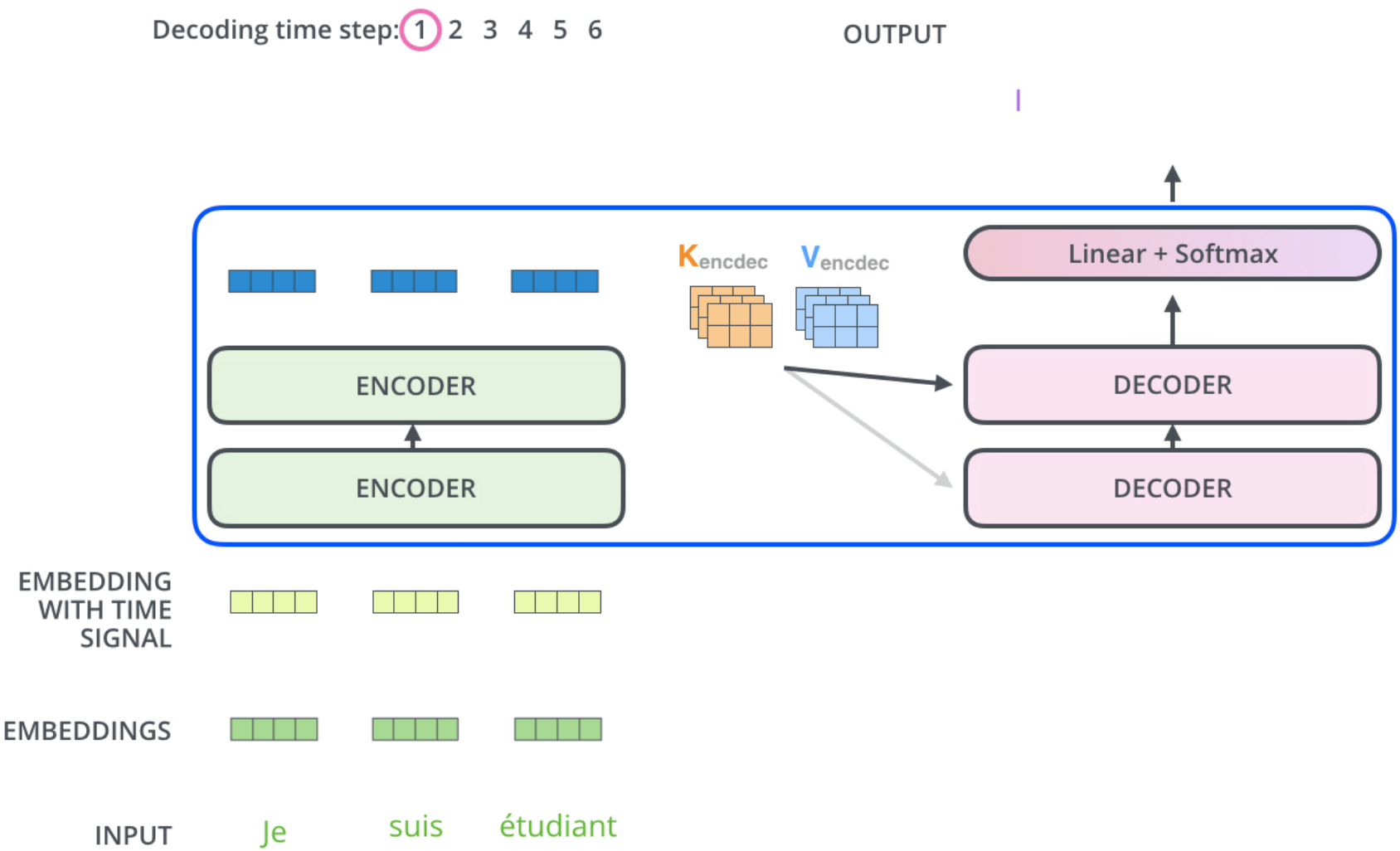
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

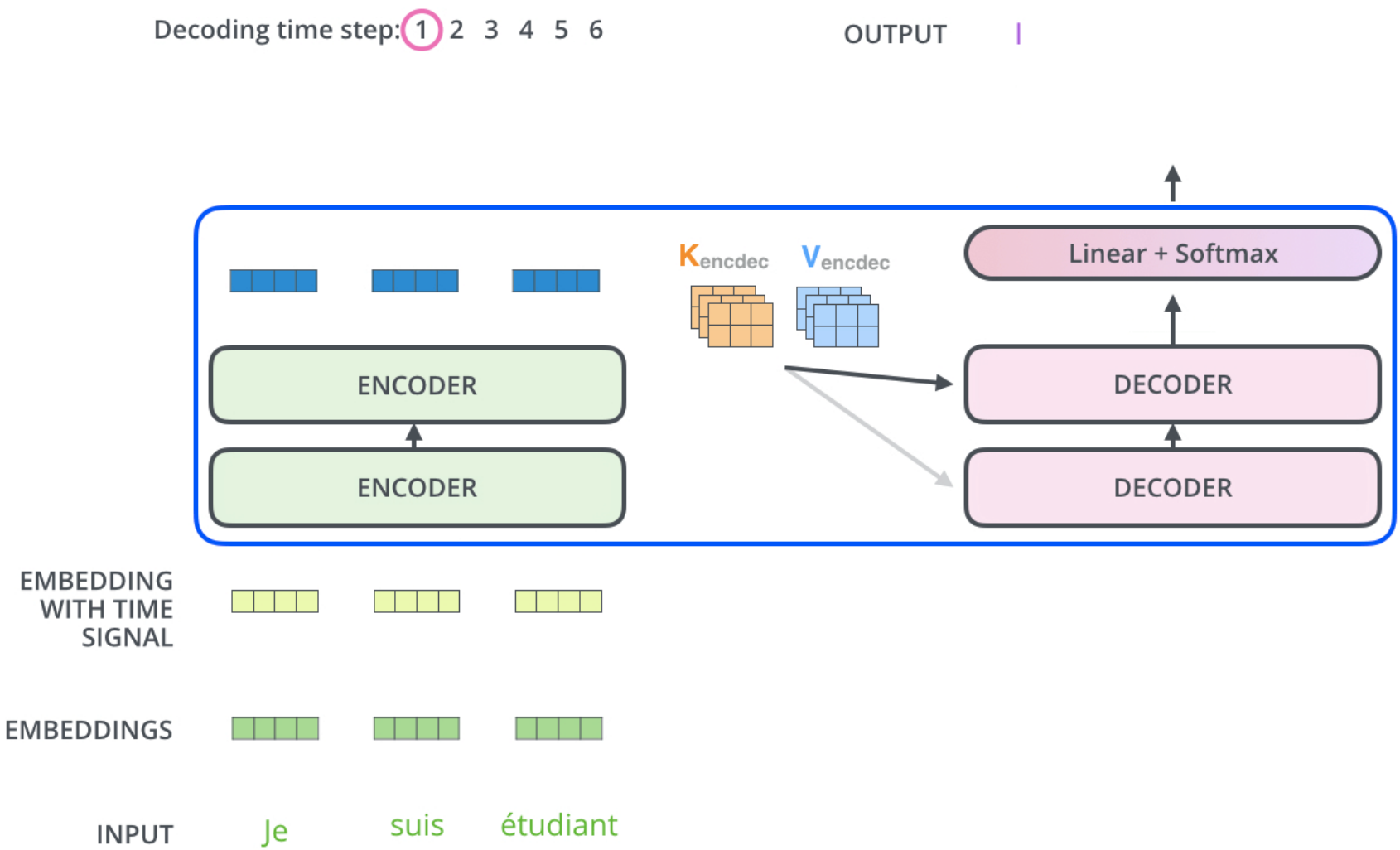
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

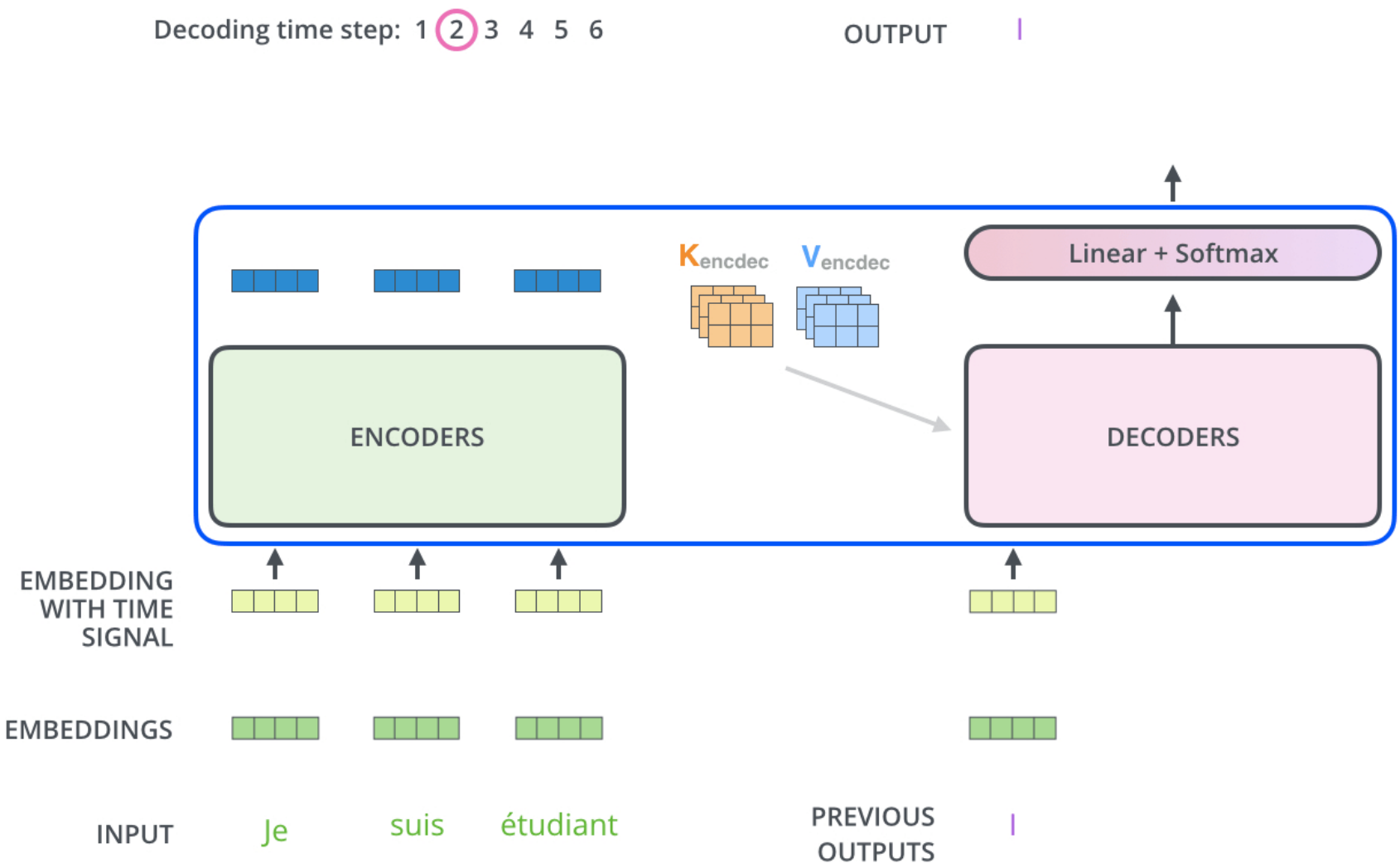
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
DECODER
DECODER
ENCODER
ENCODER
Kencdec
Vencdec
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

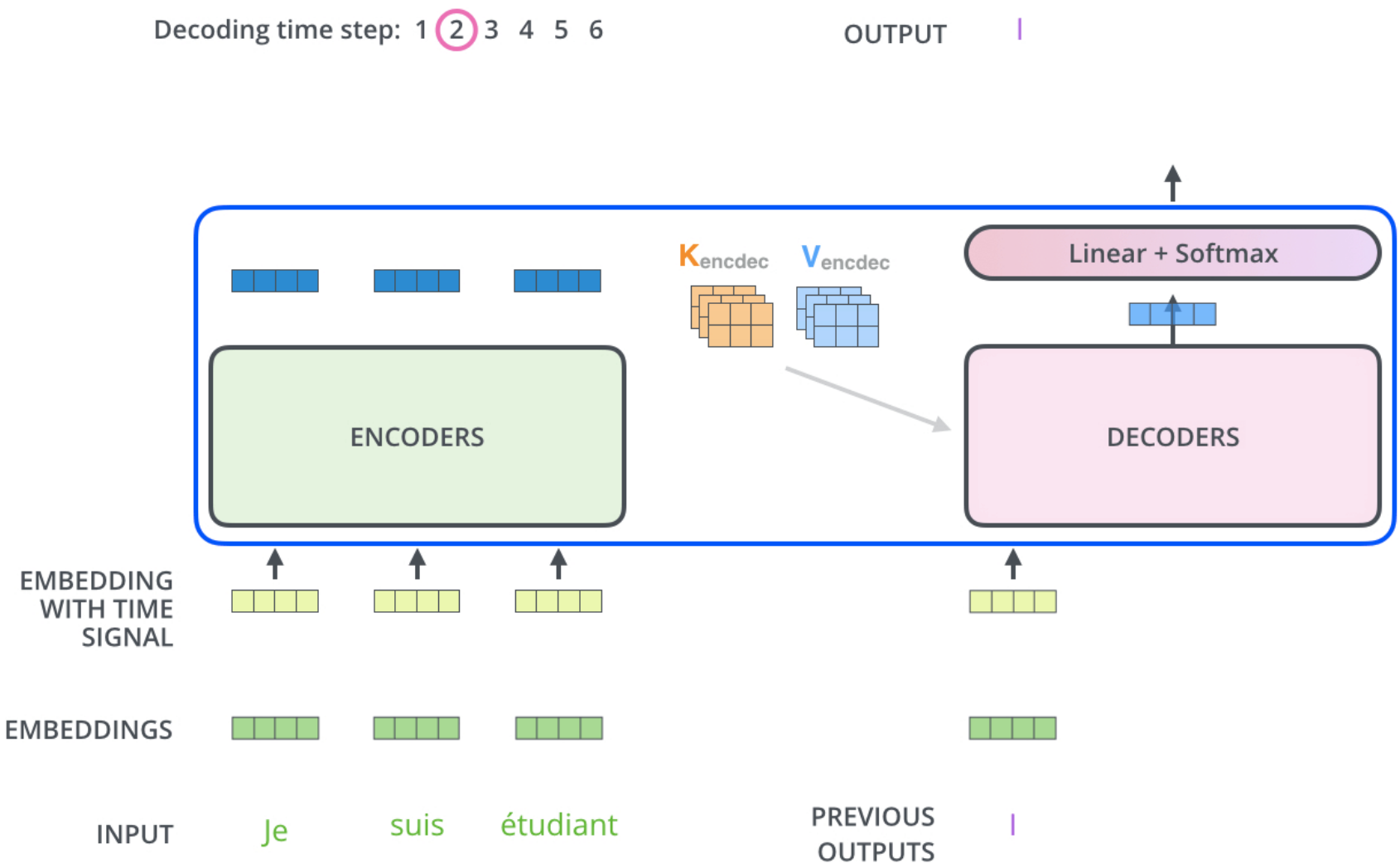
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

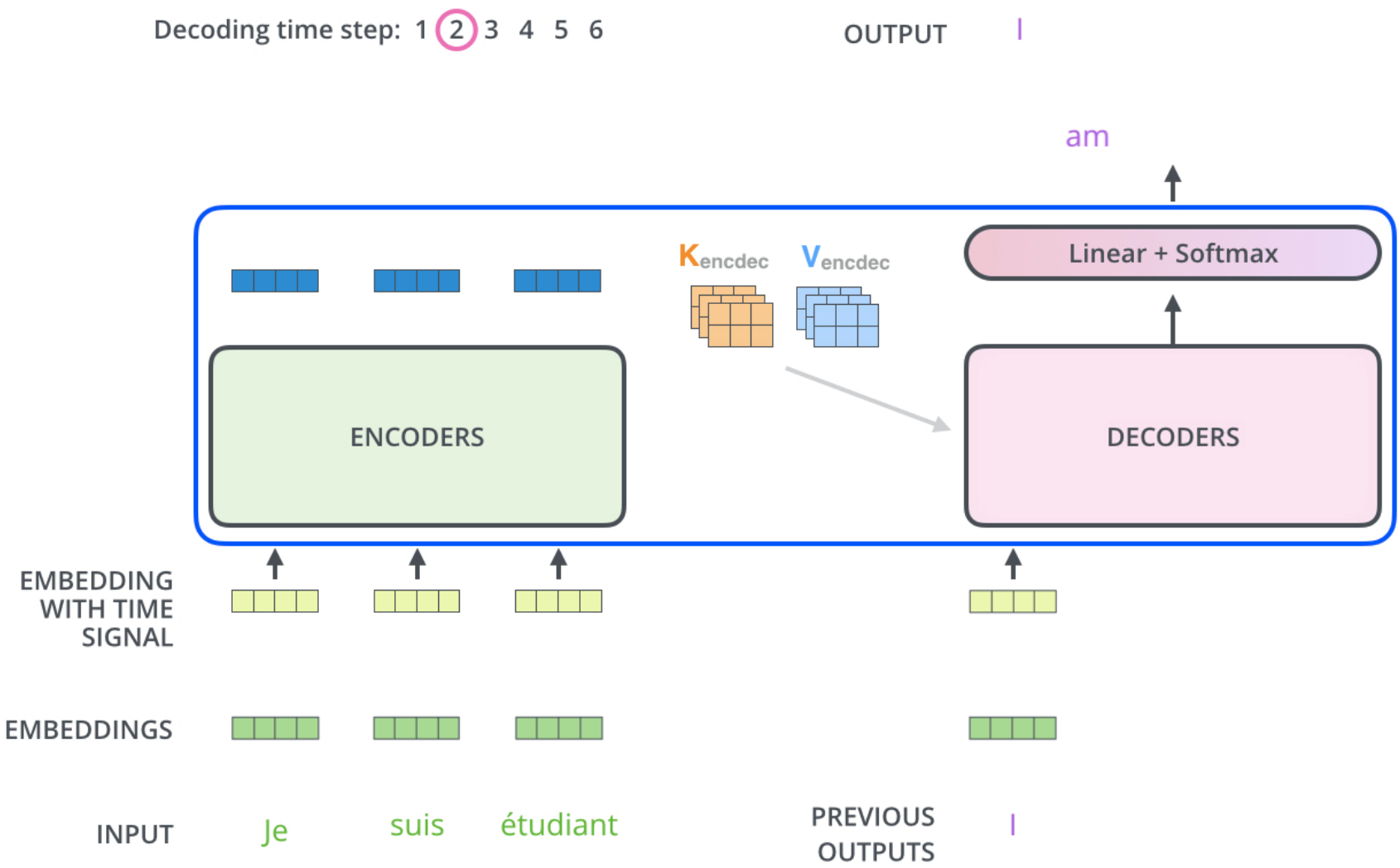
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

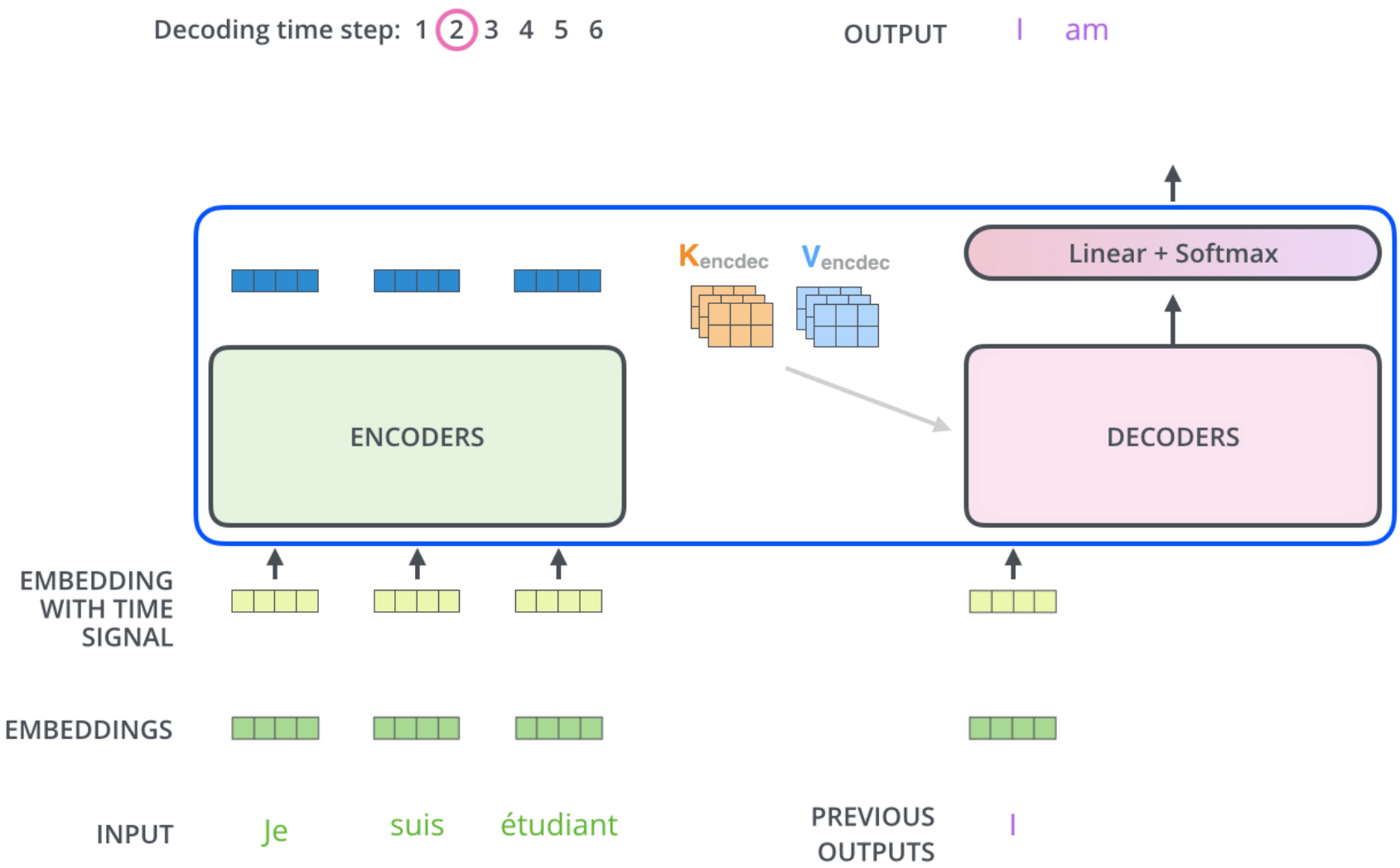
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

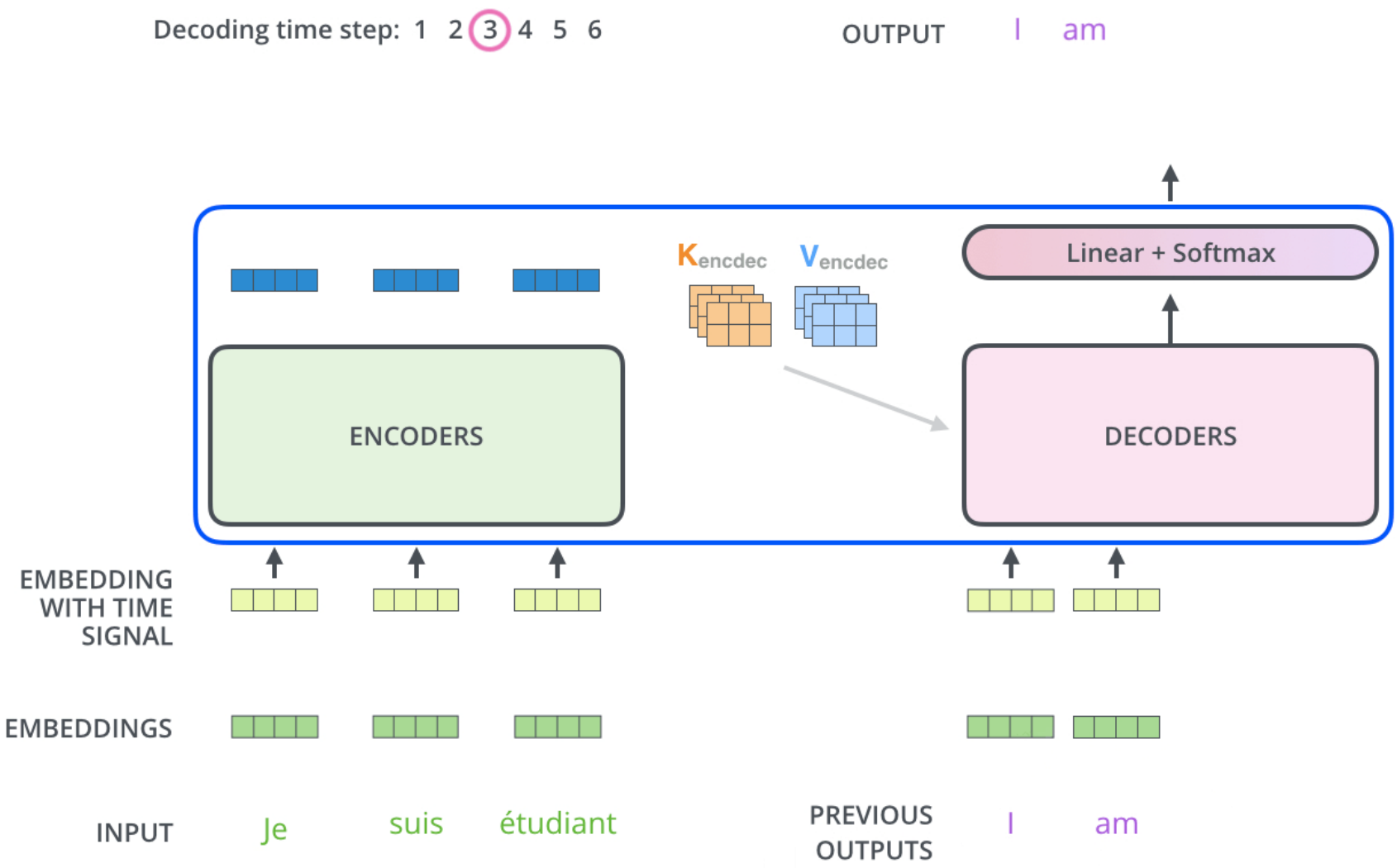
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
am
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

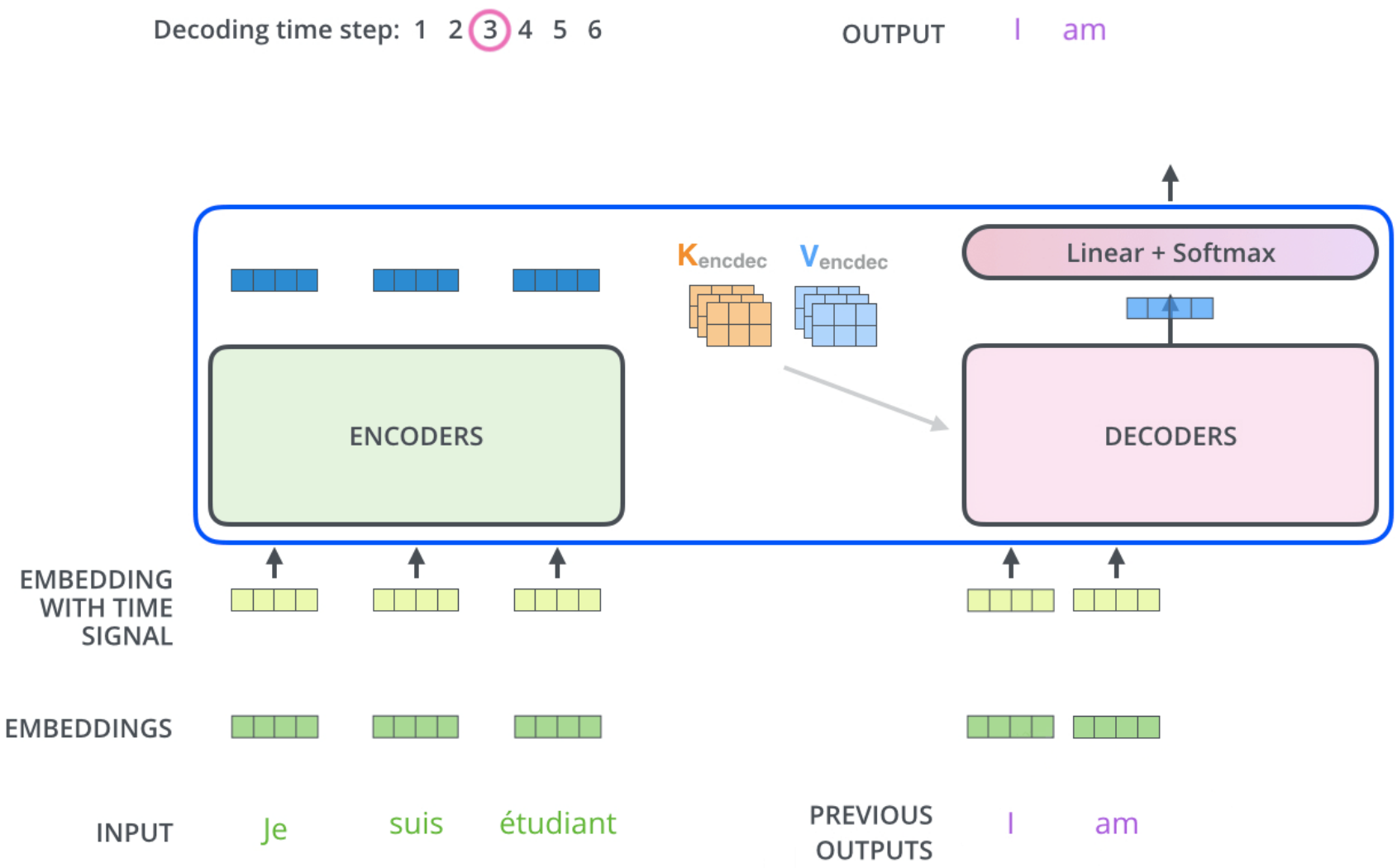
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT I am
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT Je suis étudiant
PREVIOUS OUTPUTS I

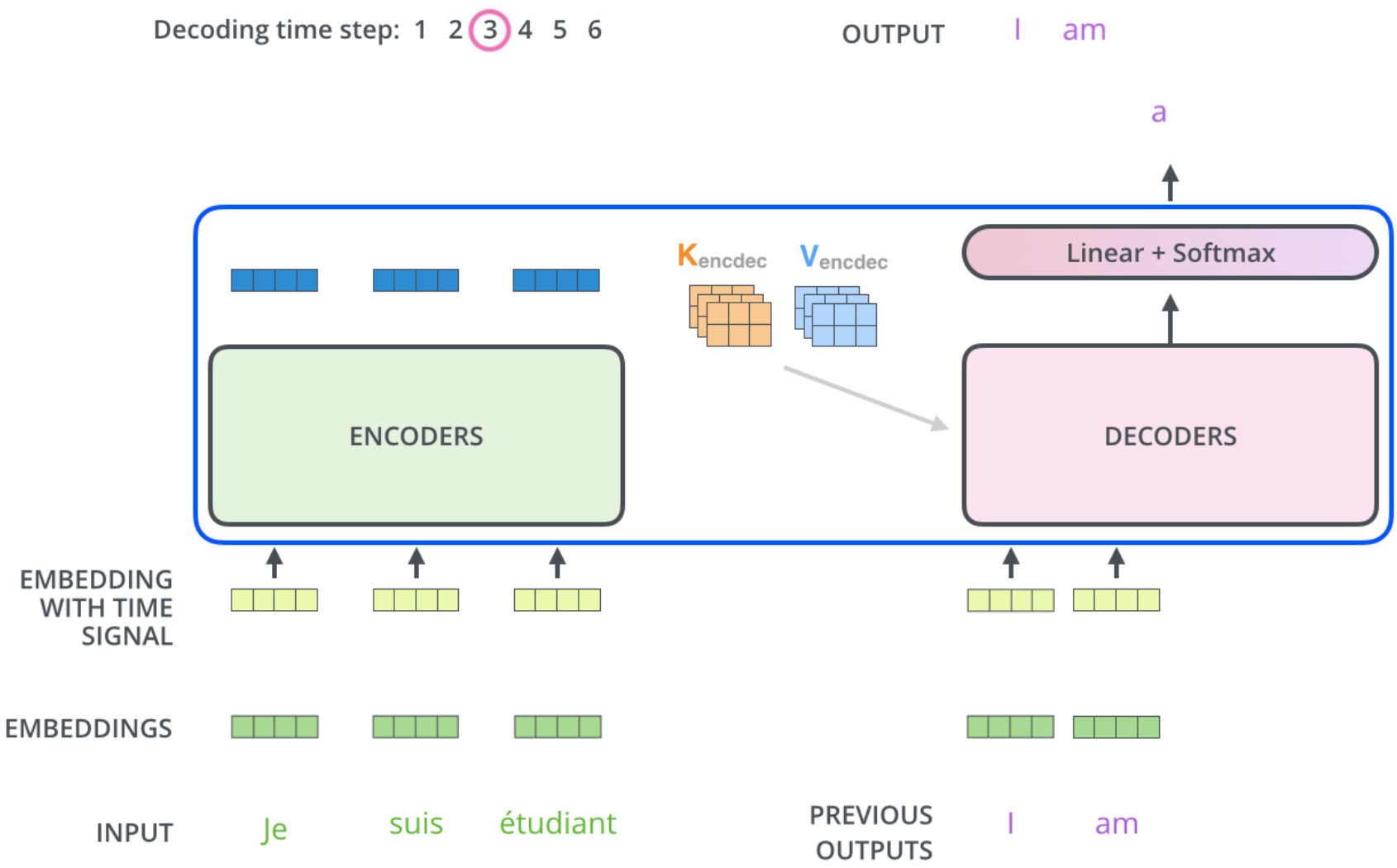
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
am
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I
am

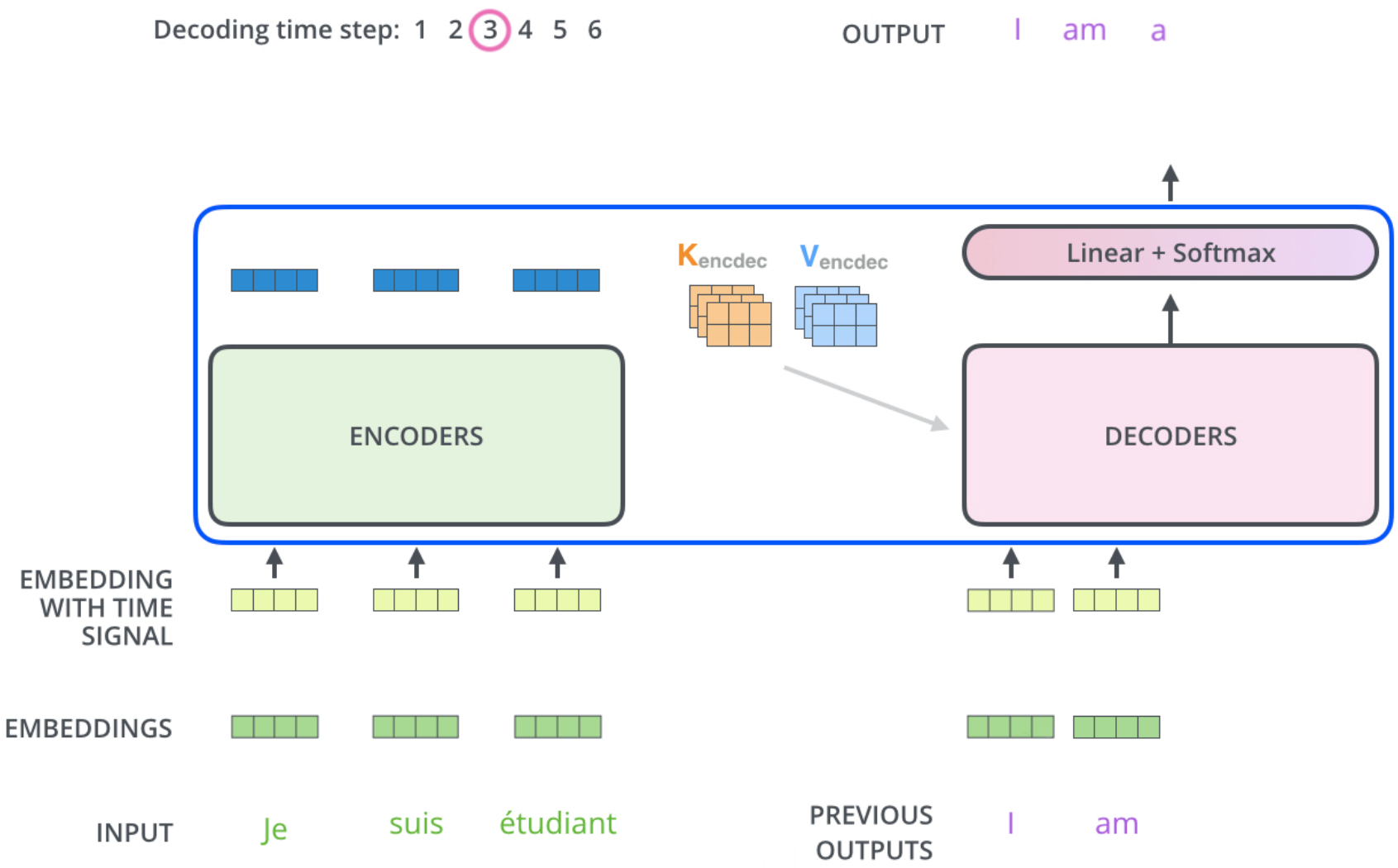
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT I am
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT Je suis étudiant
PREVIOUS OUTPUTS I am

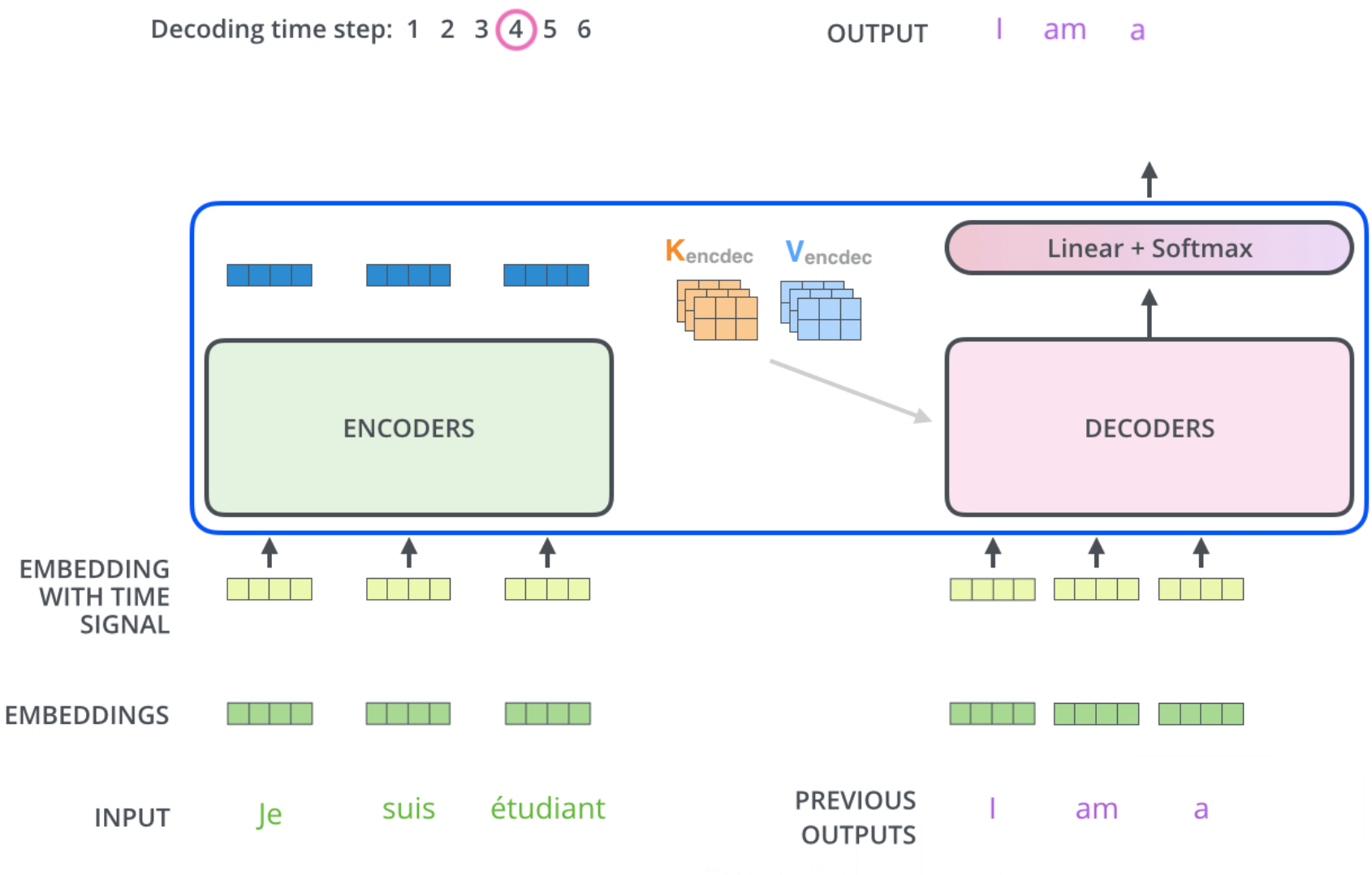
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am
a
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I
am

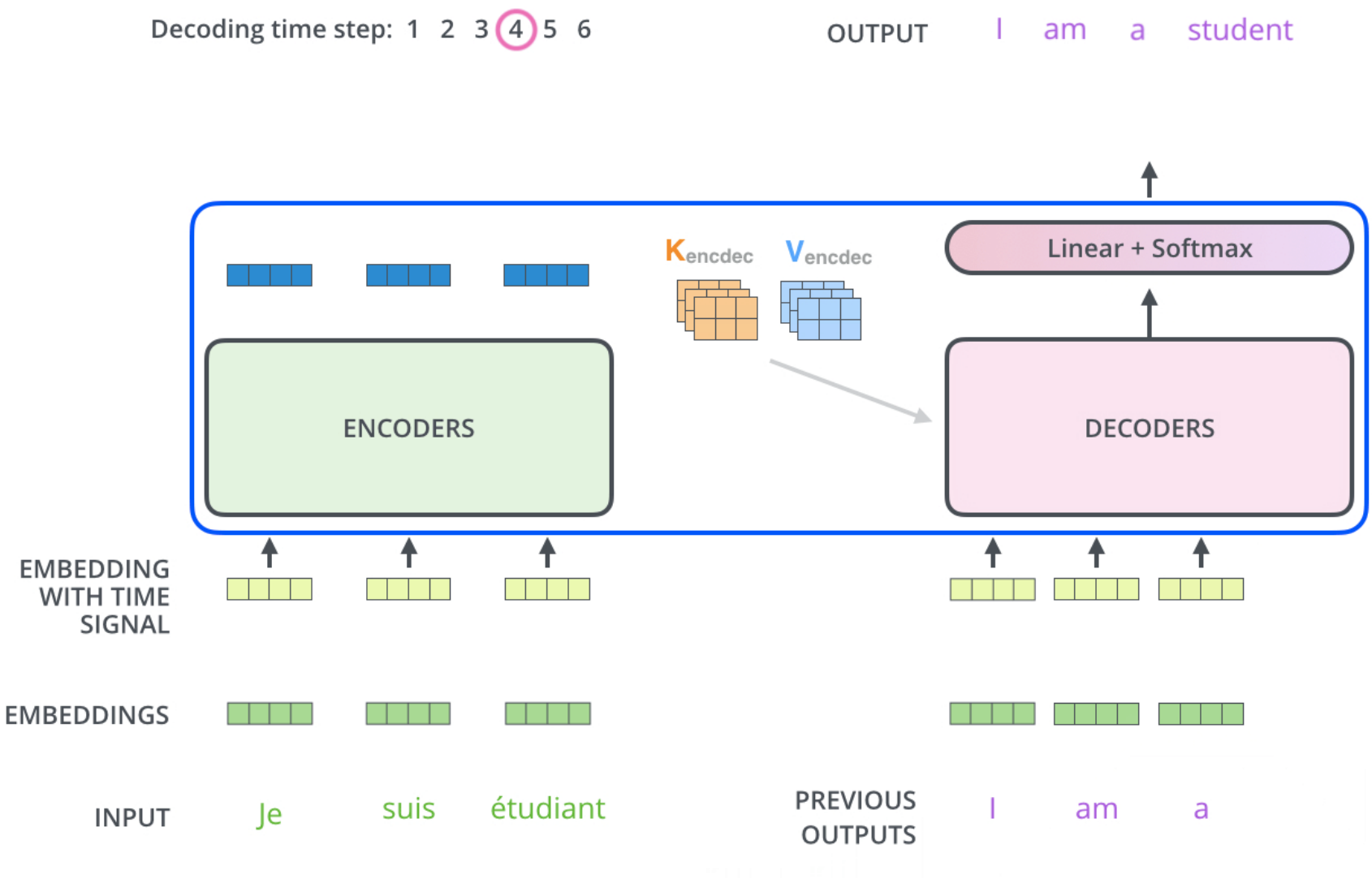
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am

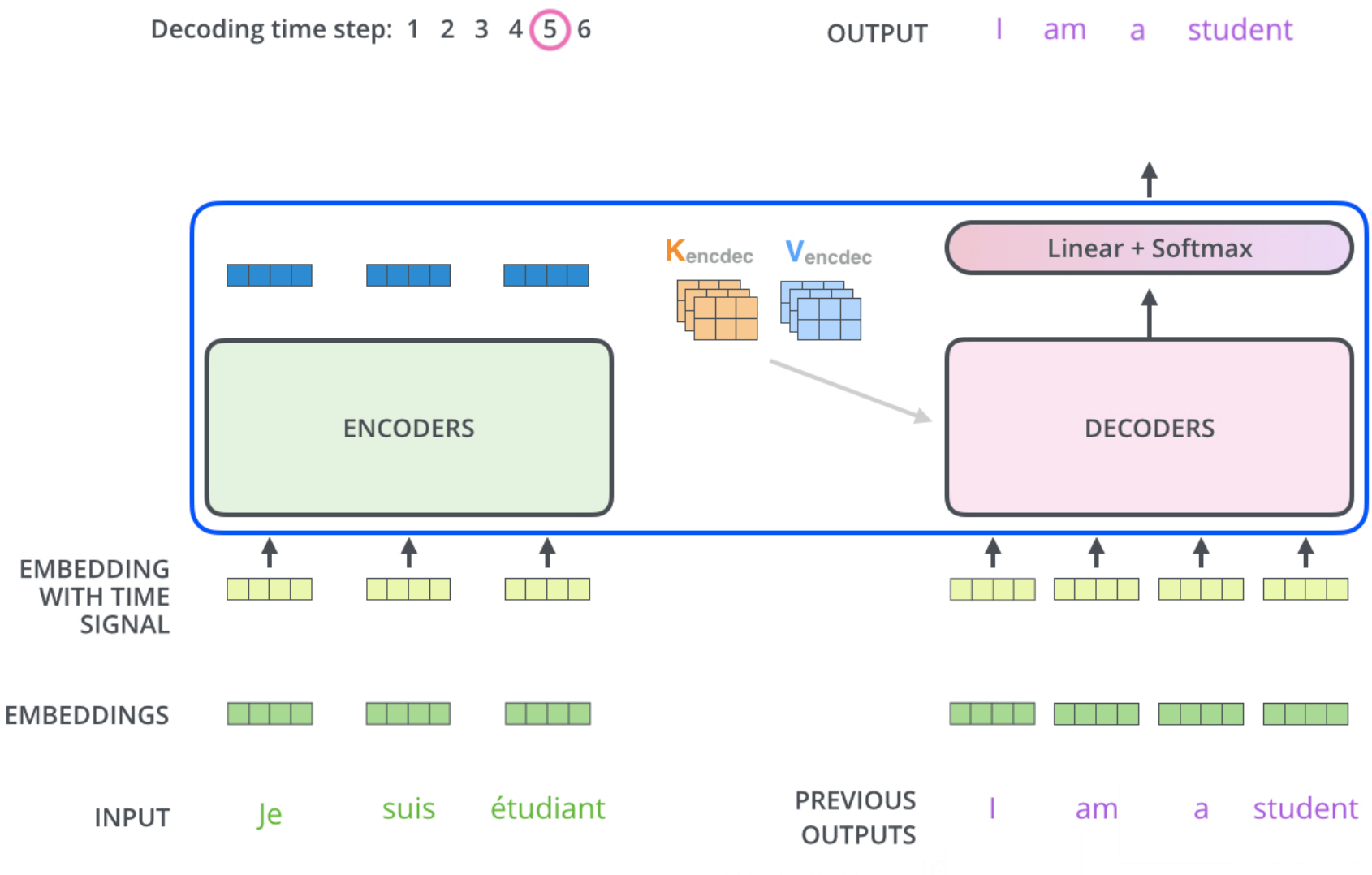
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am a

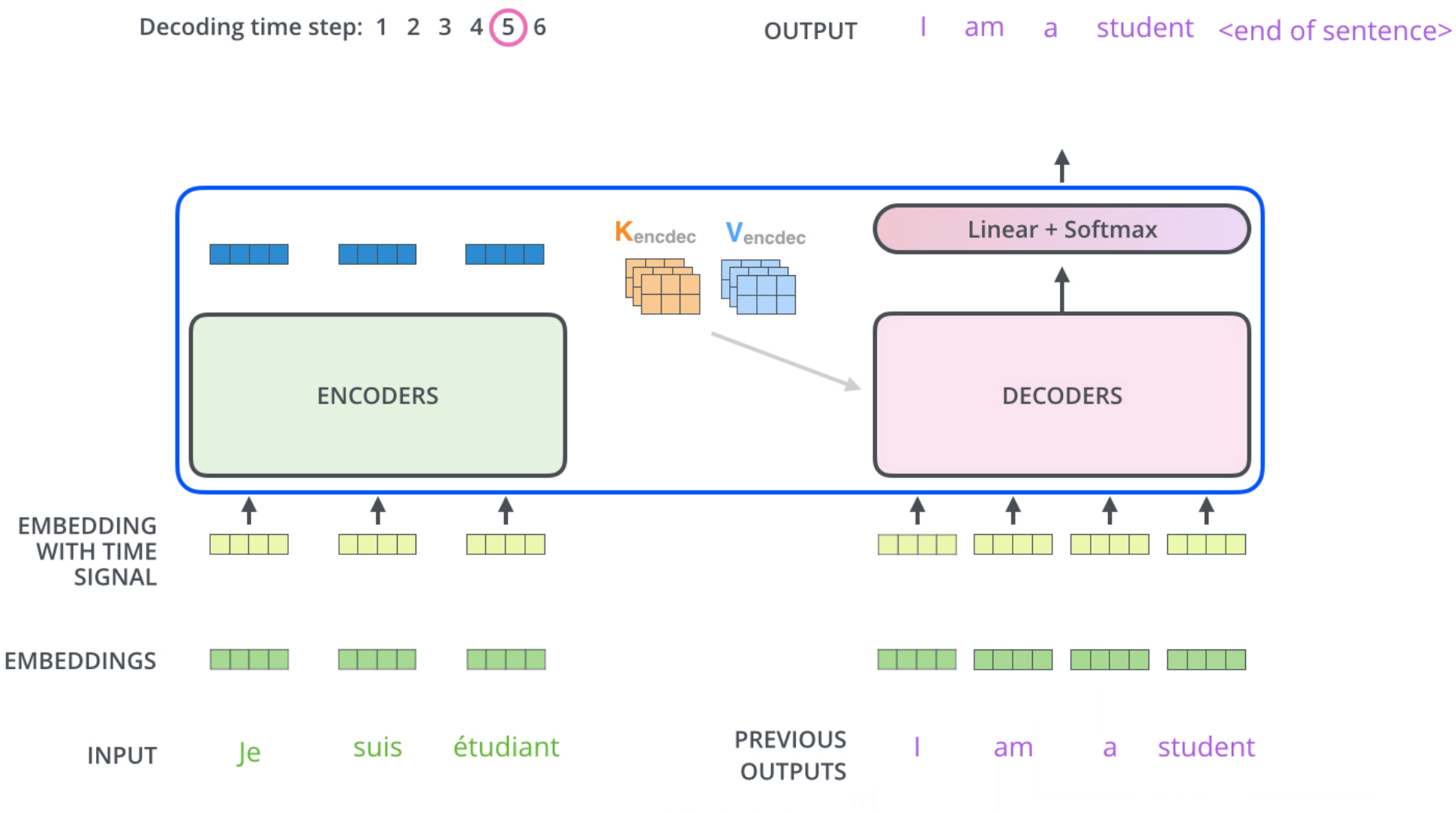
Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a student
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am a

Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a student
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am a student

Decoding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I am a student <end of sentence>
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je suis étudiant
PREVIOUS OUTPUTS
I am a student

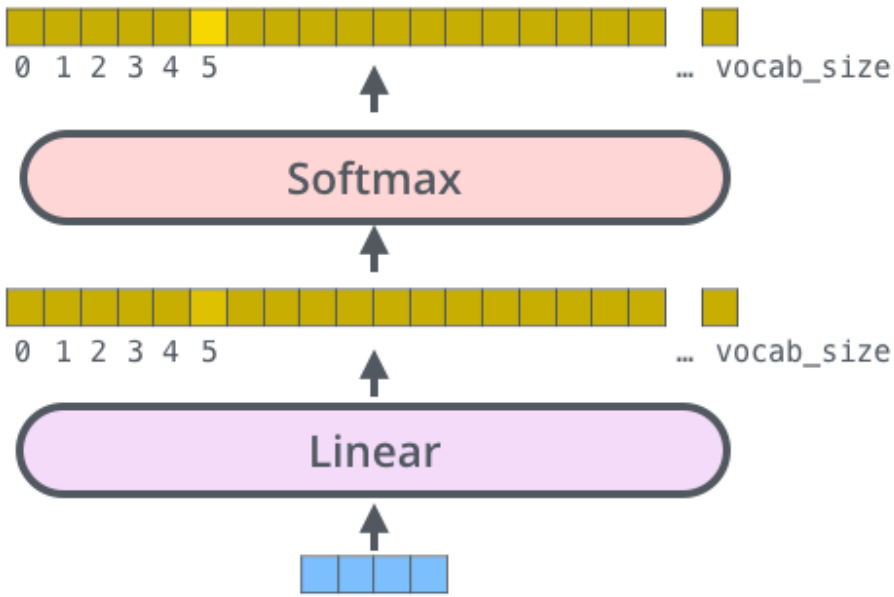
The Final Linear and Softmax Layer

Which word in our vocabulary is associated with this index?

am

Get the index of the cell with the highest value (**argmax**)

5



Results

- Machine Translation: WMT-2014 BLEU

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	**28.4**	**41.8**

Transformer models trained >3x faster than the others

Content

Content

How can we define a language? (Recap)

— The probabilistic approach

- A language can also be defined as a probabilistic distribution over all the possible sentences.
- A statistical language model is a probability distribution over sequences of words (sentences) in a given language L:

$$\sum_{s \in V^+} P_{LM}(s) = 1$$

- Or:

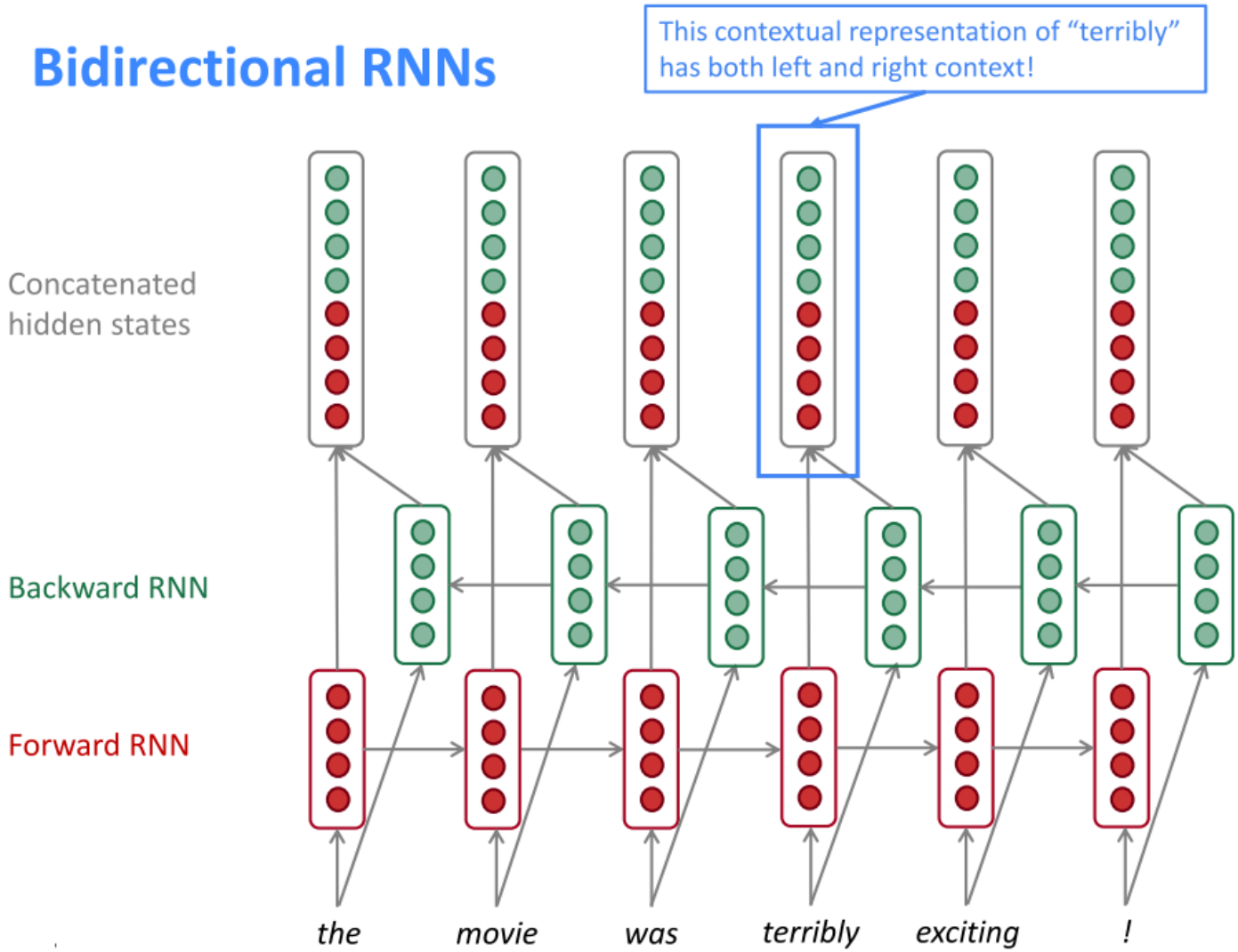
$$\sum_{\substack{s = w_1 w_2 \ldots w_n \\ w_i \in V, n > 0}} P_{LM}(s) = 1$$

NLP tasks as Conditional LMs (Recap)

x "input"	w "**text** output"
An author	A document written by that author
A topic label	An article about that topic
{SPAM, NOT_SPAM}	An email
A sentence in French	Its English translation
A sentence in English	Its French translation
A sentence in English	Its Chinese translation
An image	A text description of the image
A document	Its summary
A document	Its translation
Meterological measurements	A weather report
Acoustic signal	Transcription of speech
Conversational history + database	Dialogue system response
A question + a document	Its answer
A question + an image	Its answer

Chris Dyer, Conditional LMs (slides)

Bidirectional RNN Language Models (Recap)



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

Language models as encoders

- The motivation of language modeling is to estimate to what extent a word sequence is fluent in a certain language.
- A neural language model not only gives a scoring for a sentence, but also provides a contextualized word embedding for each word in the sentence.
- As contextualized word embeddings, NLM could be used as features for other NLP tasks, which may be better than context free word embeddings.

Language models as encoders

- The success of NMT has proved that language models are good encoders which keep the meaning of the source text for generating its translation.
- Further, the training data for a NLM could be any unannotated text, which means we can use nearly unlimited raw text to train an NLM and obtain contextualized word embeddings to help other NLP tasks.

Pre-trained language models (PLMs): introduction

- A Pre-trained Language Model (PLM) is a neural language model which is pre-trained on a large amount of unannotated text, and fune-tuned on task specific annotated data to solve downstream NLP tasks.
- The use of PLMs in NLP is similar to the use of image classification models pre-trained on ImageNet dataset in computer vision.

Pre-trained language models (PLMs): introduction

- PLMs provide contextualized word embeddings, rather than context free word embeddings like word2vec.
- PLMs adopt an unsupervised (or self-supervised) leaning method.
- Applying PLMs in other NLP tasks could be viewed as an inductive transfer learning method.

Inductive Transfer Learning

- Transfer Learning: Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.
- Inductive Transfer Learning vs Transductive Transfer Learning:
  - Inductive Transfer Learning: downstream task is unknown.
  - Transductive Transfer Learning: downstream task is known.
- PLMs adopt a inductive transfer learning method, which means it can be used in any downstream tasks.

Content

BERT: Bidirectional Encoder Representations from Transformers

- Main ideas
  - Propose a new pre-training objective so that a deep bidirectional Transformer can be trained
    - **The "masked language model" (MLM)**: the objective is to predict the original word of a masked word based only on its context
    - "**Next sentence prediction**"
- Merits of BERT
  - Just fine-tune BERT model for specific tasks to achieve state-of-the-art performance
  - BERT advances the state-of-the-art for eleven NLP tasks

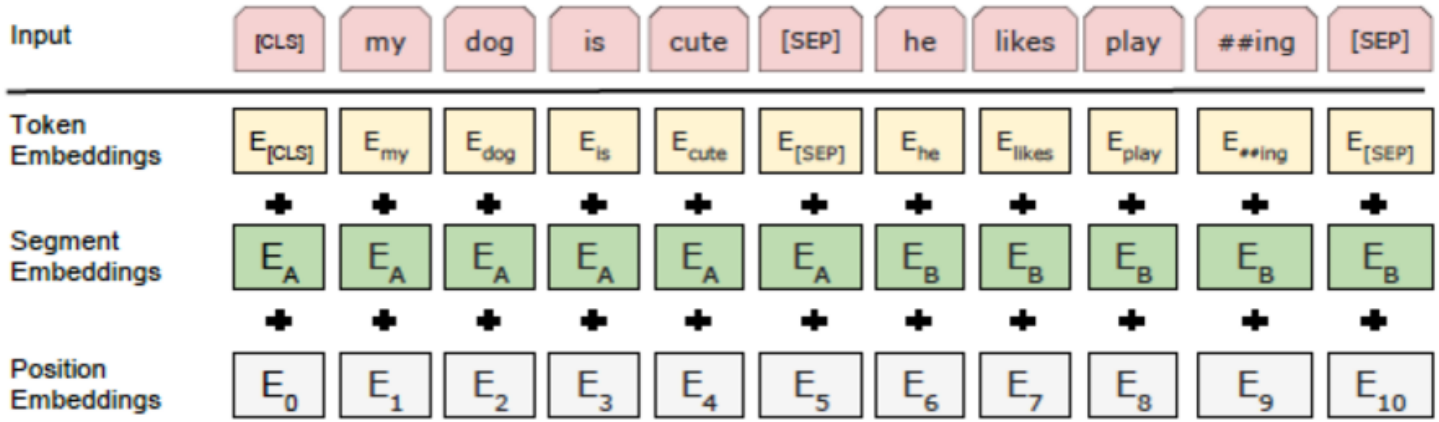
Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT - model architecture

- BERT's model architecture is a multi-layer **bidirectional Transformer encoder**
  - (Vaswani et al., 2017) "Attention is all you need"
- Two models with different sizes were investigated
  - $\text{BERT}_{\text{BASE}}$: L=12, H=768, A=12, Total Parameters=110M
    - (*L*: number of layers (Transformer blocks), *H* is the hidden size, *A*: the number of self-attention heads)
  - $\text{BERT}_{\text{LARGE}}$: L=24, H=1024, A=16, Total Parameters=340M

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: Input Representation



- Token Embeddings: Use pretrained WordPiece embeddings
- Position Embeddings: Use learned Position Embeddings
  - Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutionalsequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.
- Added sentence embedding to every tokens of each sentence
- Use [CLS] for the classification tasks
- Separate sentences by using a special token [SEP]

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: pre-training tasks

Task#1: Masked LM

- 15% of the words are masked at random
  - and the task is to predict the masked words based on its left and right context
- Not all tokens were masked in the same way (example sentence “My dog is hairy”)
  - 80% were replaced by the <MASK> token: “My dog is <MASK>”
  - 10% were replaced by a random token: “My dog is apple”
  - 10% were left intact: “My dog is hairy”

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: pre-training tasks

Task#2: Next Sentence Prediction

- Motivation
  - Many downstream tasks are based on understanding the relationship between two text sentences
    - Question Answering (QA) and Natural Language Inference (NLI)
  - Language modeling does not directly capture that relationship
- The task is pre-training binarized next sentence prediction task

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label = isNext

Input = [CLS] the man [MASK] to the store [SEP]penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

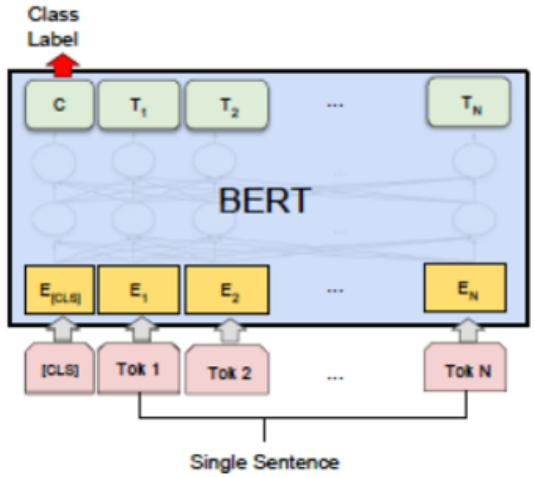
Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure

Fine-tuning procedure

- For sequence-level classification task
  - Obtain the representation of the input sequence by using the final hidden state (hidden state at the position of the special token [CLS]) $C \in R^H$
  - Just add a classification layer and use softmax to calculate label probabilities. Parameters W $\in R^{K \times H}$

$$P = \text{softmax}(CW^T)$$



Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure

Fine-tuning procedure

- For sequence-level classification task
  - All of the parameters of BERT and W are fine-tuned jointly
- Most model hyperparameters are the same as in pre-training
  - except the batch size, learning rate, and number of training epochs

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure

Fine-tuning procedure

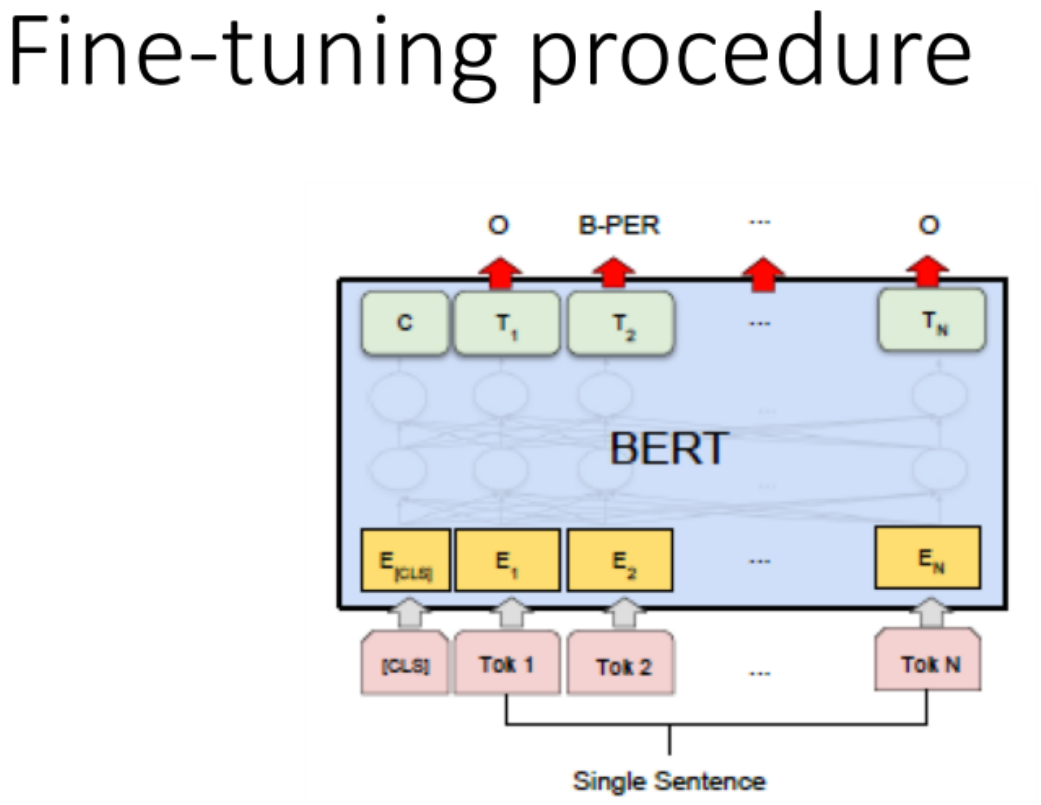
- Token tagging task (e.g., Named Entity Recognition)
  - Feed the final hidden representation $T_i \in R^H$ for each token i into a classification layer for the tagset (NER label set)
  - To make the task compatible with WordPiece tokenization

```
Jim   Hen   ##son was a puppet ##eer
I-PER I-PER X     O   O O      X
```

  - Predict the tag for the first sub-token of a word
  - No prediction is made for X

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure



Single Sentence Tagging Tasks: CoNLL-2003 NER

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure

Fine-tuning procedure

- Span-level task: SQuAD v1.1
  - Input Question:
    `Where do water droplets collide with ice crystals to form precipitation?`
  - Input Paragraph:
    `.... Precipitation forms as smaller dropletscoalesce via collision with other rain dropsor ice crystals within a cloud. ...`
  - Output Answer:
    `within a cloud`

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning procedure

Fine-tuning procedure

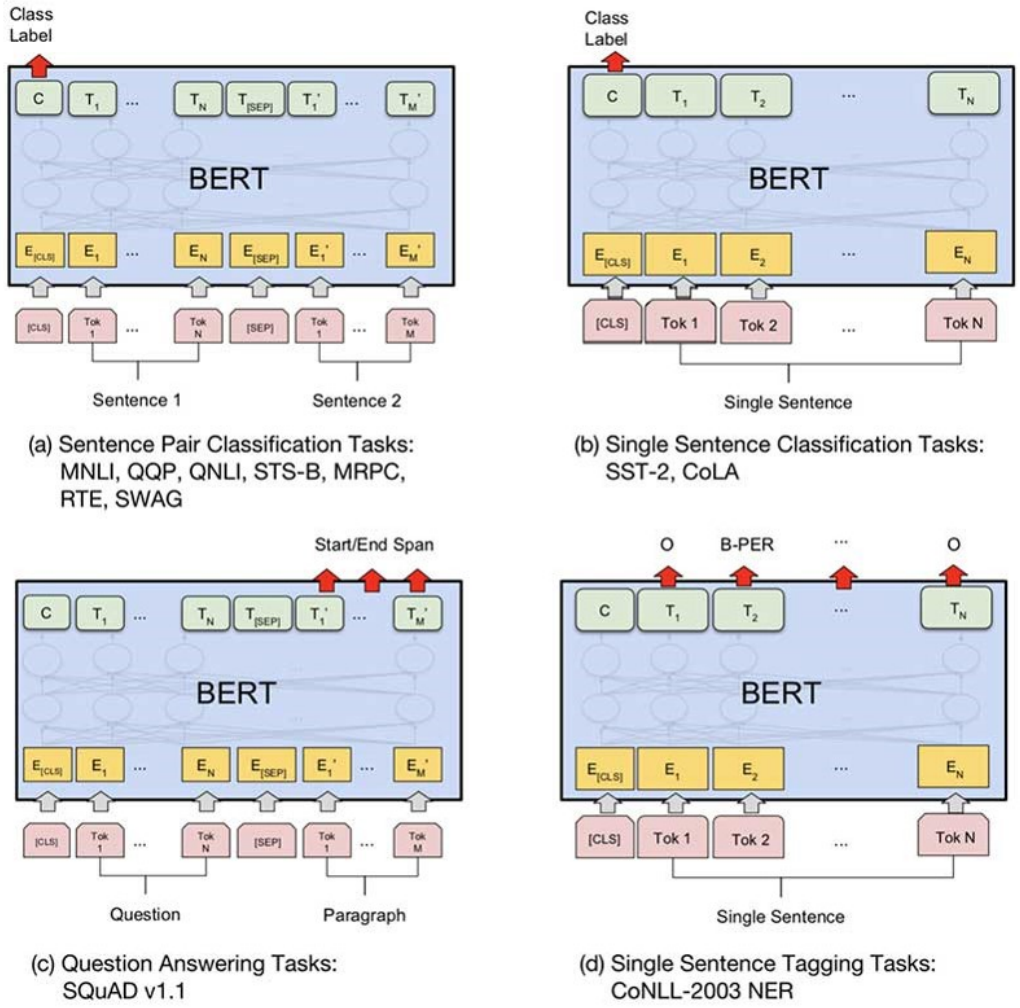
- Span-level task: SQuAD v1.1
  - Represent the input question and paragraph as a single packed sequence
    - The question uses the A embedding and the paragraph uses the B embedding
  - New parameters to be learned in fine-tuning are start vector $S \in \mathbb{R}^H$ and end vector $E \in \mathbb{R}^H$
  - Calculate the probability of word i being the start of the answer span

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

  - The training objective is the log-likelihood the correct and end positions

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: fine-tuning summary



Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: experiments

Experiments

- GLUE (General Language Understanding Evaluation) benchmark
  - Distribute canonical Train, Dev and Test splits
  - Labels for Test set are not provided
- Datasets in GLUE:
  - MNLI: Multi-Genre Natural Language Inference
  - QQP: Quora Question Pairs
  - QNLI: Question Natural Language Inference
  - SST-2: Stanford Sentiment Treebank
  - CoLA: The corpus of Linguistic Acceptability
  - STS-B: The Semantic Textual Similarity Benchmark
  - MRPC: Microsoft Research Paraphrase Corpus
  - RTE: Recognizing Textual Entailment
  - WNLI: Winograd NLI

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: GLUE results

GLUE Results

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

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

BERT: SQuAD v1.1 results

SQuAD v1.1

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

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

Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides))

Content

GPT: introduction

- Following the similar idea of ELMo, OpenAI GPT, short for Generative Pre-training Transformer (Radford et al., 2018), expands the unsupervised language model to a much larger scale by training on a giant collection of free text corpora.
- Despite of the similarity, GPT has two major differences from ELMo:
  - The model architectures are different: ELMo uses a shallow concatenation of independently trained left-to-right and right-to-left multi-layer LSTMs, while GPT is a multi-layer transformer decoder.
  - The use of contextualized embeddings in downstream tasks are different: ELMo feeds embeddings into models customized for specific tasks as additional features, while GPT fine-tunes the same base model for all end tasks.

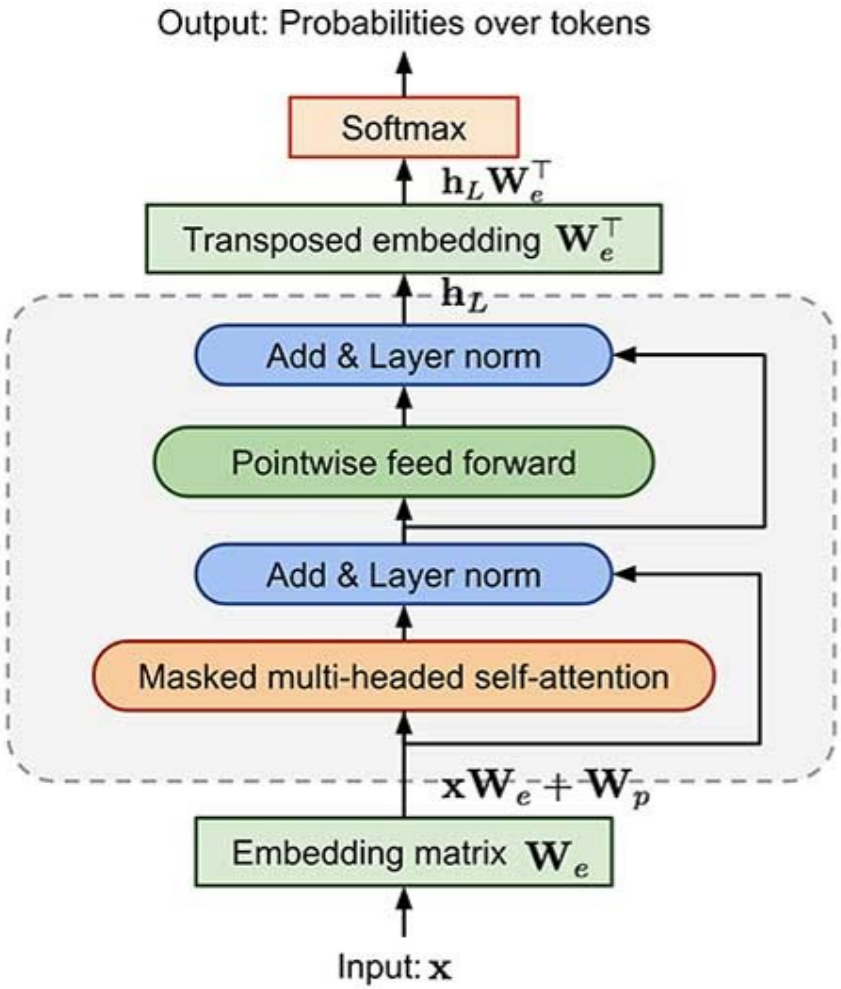
Liliang Wen, Generalized Language Models: ULMFiT & OpenAI GPT (blog)

GPT: Transformer Decoder as Language Model

- Compared to the original transformer architecture, the transformer decoder model discards the encoder part, so there is only one single input sentence rather than two separate source and target sequences.
- This model applies multiple transformer blocks over the embeddings of input sequences. Each block contains a masked multi-headed self-attention layer and a pointwise feed-forward layer. The final output produces a distribution over target tokens after softmax normalization.

Liliang Wen, Generalized Language Models: Ulmfit & OpenAI GPT (blog)

GPT: Transformer Decoder as Language Model

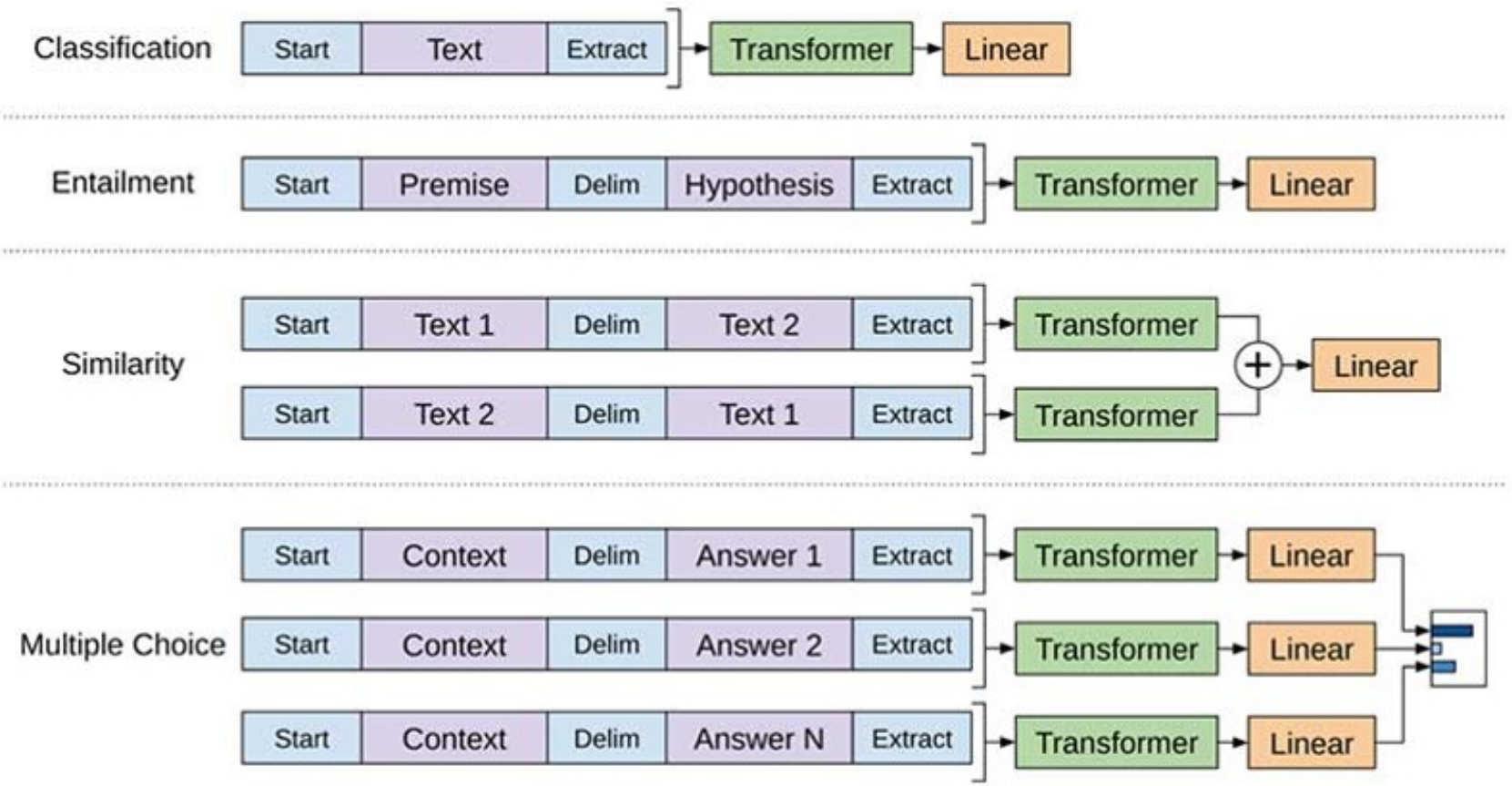


Transformer Block
Repeat x L=12

$$\mathbf{h}_\ell = \text{transformer\_block}(\mathbf{h}_{\ell-1})$$
$$\ell = 1, \ldots, L$$

Liliang Wen, Generalized Language Models: Ulmfit & OpenAI GPT (blog)

GPT: supervised fine-tuning



Liliang Wen, Generalized Language Models: Ulmfit & OpenAI GPT (blog)

GPT-2: Introduction

- The OpenAI GPT-2 language model is a direct successor to GPT. GPT-2 has 1.5B parameters, 10x more than the original GPT, and it achieves SOTA results on 7 out of 8 tested language modeling datasets in a zero-shot transfer setting without any task-specific fine-tuning.
- The pre-training dataset contains 8 million Web pages collected by crawling qualified outbound links from Reddit. Large improvements by OpenAI GPT-2 are specially noticeable on small datasets and datasets used for measuring long-term dependency.

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: zero-shot transfer

The pre-training task for GPT-2 is solely language modeling. All the downstream language tasks are framed as predicting conditional probabilities and there is no task-specific fine-tuning.

- **Text generation** is straightforward using LM.
- **Machine translation** task, for example, English to Chinese, is induced by conditioning LM on pairs of "English sentence = Chinese sentence" and the sentence to be translated "English sentence =" at the end.

  (to be continued...)

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: zero-shot transfer

(...continued)
For example, the conditional probability to predict might look like:

$$p\left(?\,\middle|\,\begin{array}{r}\text{I like green apples.} = \text{我喜欢绿苹果。}\\ \text{A cat meows at him.} = \text{一只猫对他喵。}\\ \text{It is raining cats and dogs.} =\end{array}\right)$$

- **QA** task is formatted similar to translation with pairs of questions and answers in the context.
- **Summarization** task is induced by adding TL;DR: after the articles in the context.

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: Model Modifications

Compared to GPT, other than having many more transformer layers and parameters, GPT-2 incorporates only a few architecture modifications:

- Layer normalization was moved to the input of each sub-block, similar to a residual unit of type “building block” (differently from the original type “bottleneck”, it has batch normalization applied before weight layers).
- An additional layer normalization was added after the final self-attention block.

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: Model Modifications

Compared to GPT, other than having many more transformer layers and parameters, GPT-2 incorporates only a few architecture modifications:

- A modified initialization was constructed as a function of the model depth.
- The weights of residual layers were initially scaled by a factor of $\frac{1}{\sqrt{N}}$ where N is the number of residual layers.
- Use larger vocabulary size and context size.

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: Models

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

Liliang Wen, Generalized Language Models: Bert & OpenAI GPT-2 (blog)

GPT-2: Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Alec Radford, et al.,Language Models are Unsupervised Multitask Learners, OpenAI Blog 1.8 (2019): 9

GPT-2: an example of generated text

SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

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

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

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

Pérez and the others then ventured further into the valley. "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.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.

While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."

However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization," said the scientist.

The generated text is suprisingly fluent and cohesive except for the common sense error (highlighted).

Alec Radford, et al.,Language Models are Unsupervised Multitask Learners, OpenAI Blog 1.8 (2019): 9

GPT-2: an example of generated text

SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

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

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

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

Pérez and the others then ventured further into the valley. "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.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.

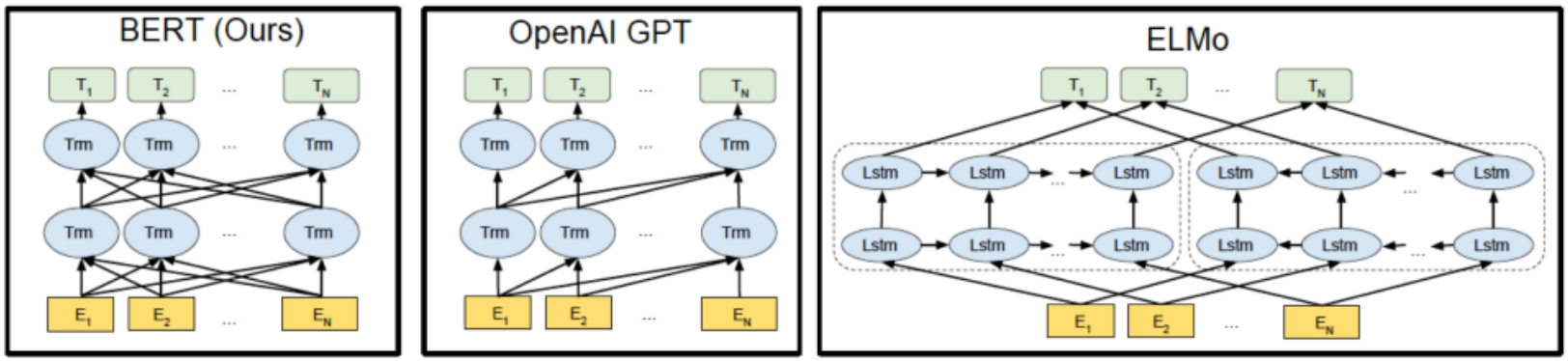
While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."

However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization," said the scientist.

The generated text is suprisingly fluent and cohesive except for the common sense error (highlighted).

Alec Radford, et al.,Language Models are Unsupervised Multitask Learners, OpenAI Blog 1.8 (2019): 9

Differences: BERT, GPT & ELMo



Differences in pre-training model architectures: BERT, OpenAI GPT, and ELMo

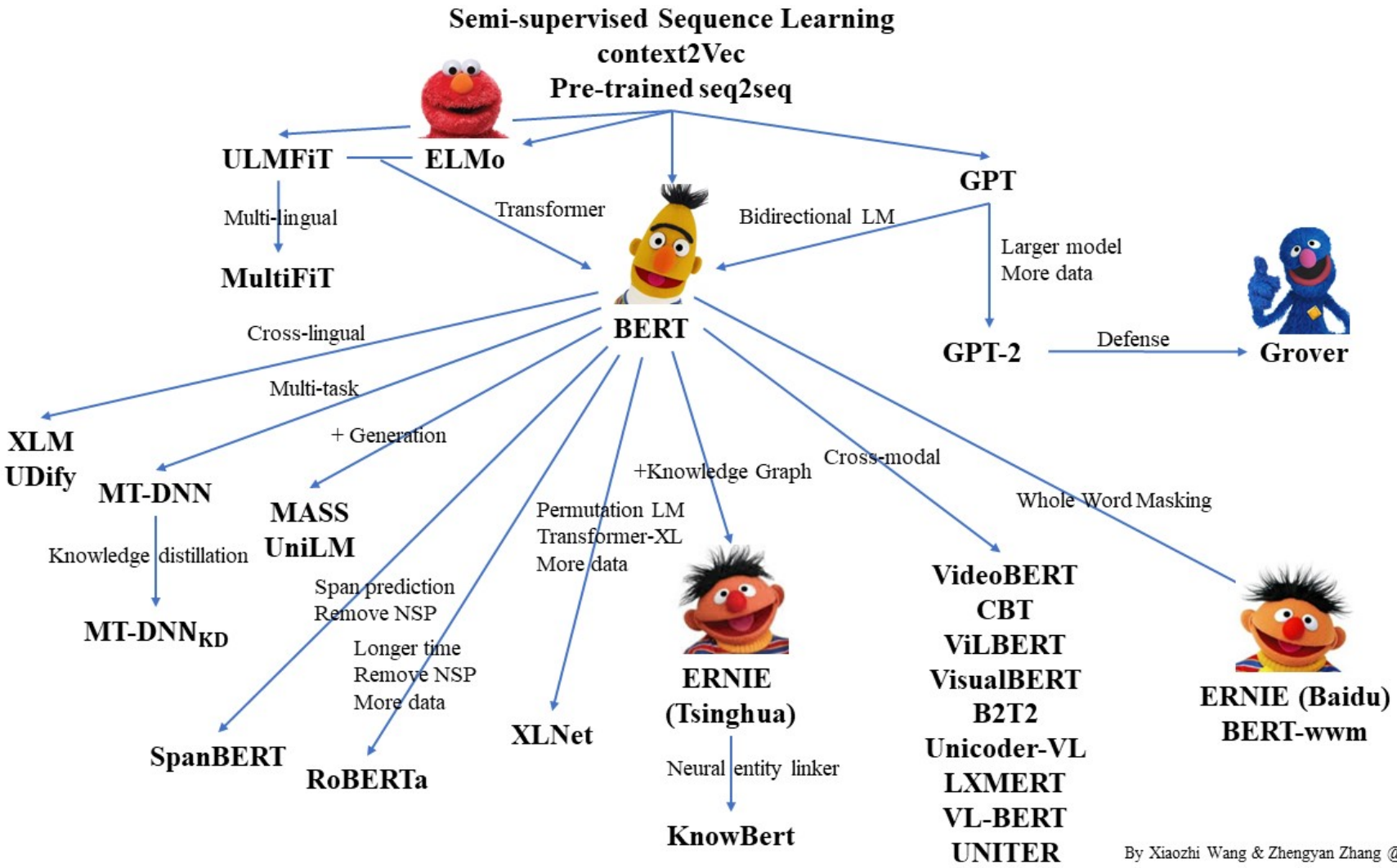
Devlin et al., 2018, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Slides)

Content

The success of PLMs

- PLMs become another huge success in using deep learning in NLP after NMT:
  - PLMs have refreshed the state-of-the-art of most of the NLP tasks.
  - PLMs simplified the design of NN structures for downstream NLP tasks.
- Using PLMs has become a new paradigm for NLP research.

PLM recent progress: model family



https://github.com/thunlp/PLMpapers

Recent progress and applications

- Improved PLMs
- Larger PLMs
- Downsized PLMs
- Knowledge-aware PLMs
- Multilingual PLMs
- Multimodal PLMs
- Text Generation with PLMs
- Go beyond NLP

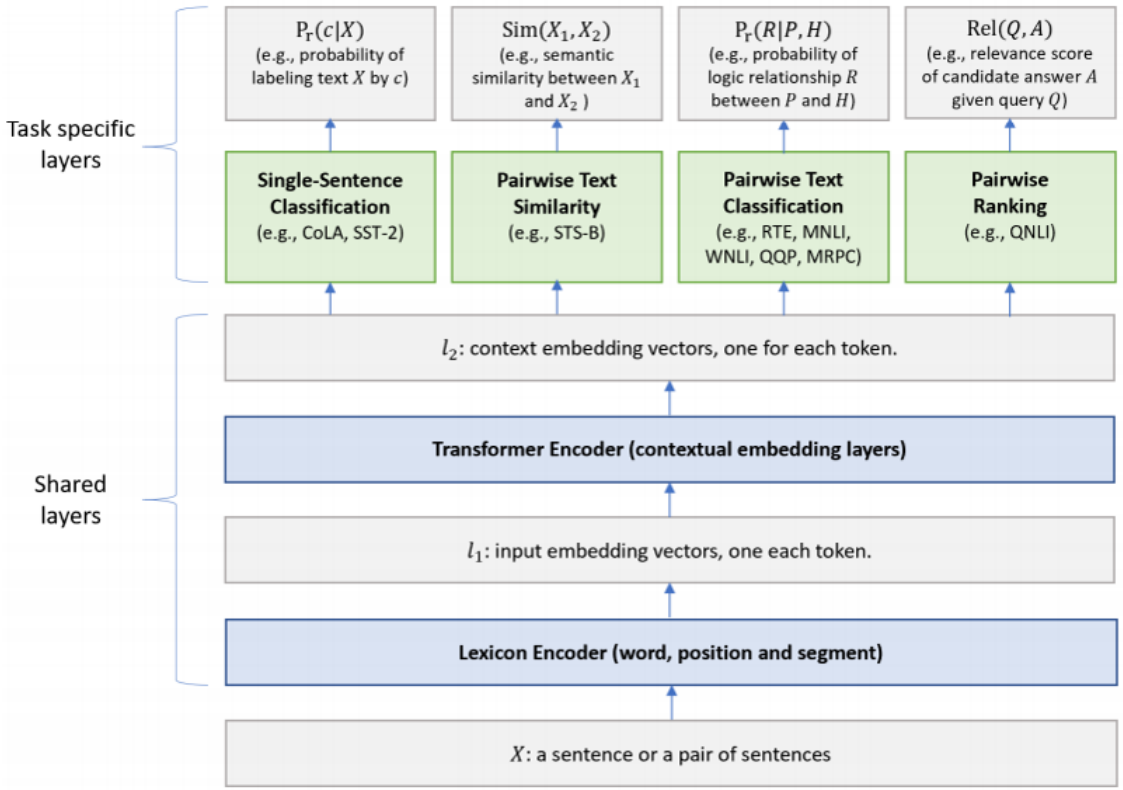
Multi-task Learning with BERT

- Pre-training
  - learning universal language representations by leveraging large amounts of unlabeled data
- Multi-task Learning
  - leveraging supervised data from many related tasks
  - regularization effect; universal representations across tasks
- Four types of tasks:
  - Single-Sentence Classification
  - Text Similarity
  - Pairwise Text Classification
  - Relevance Ranking

Liu et al., 2019, Multi-Task Deep Neural Networks for Natural Language Understanding

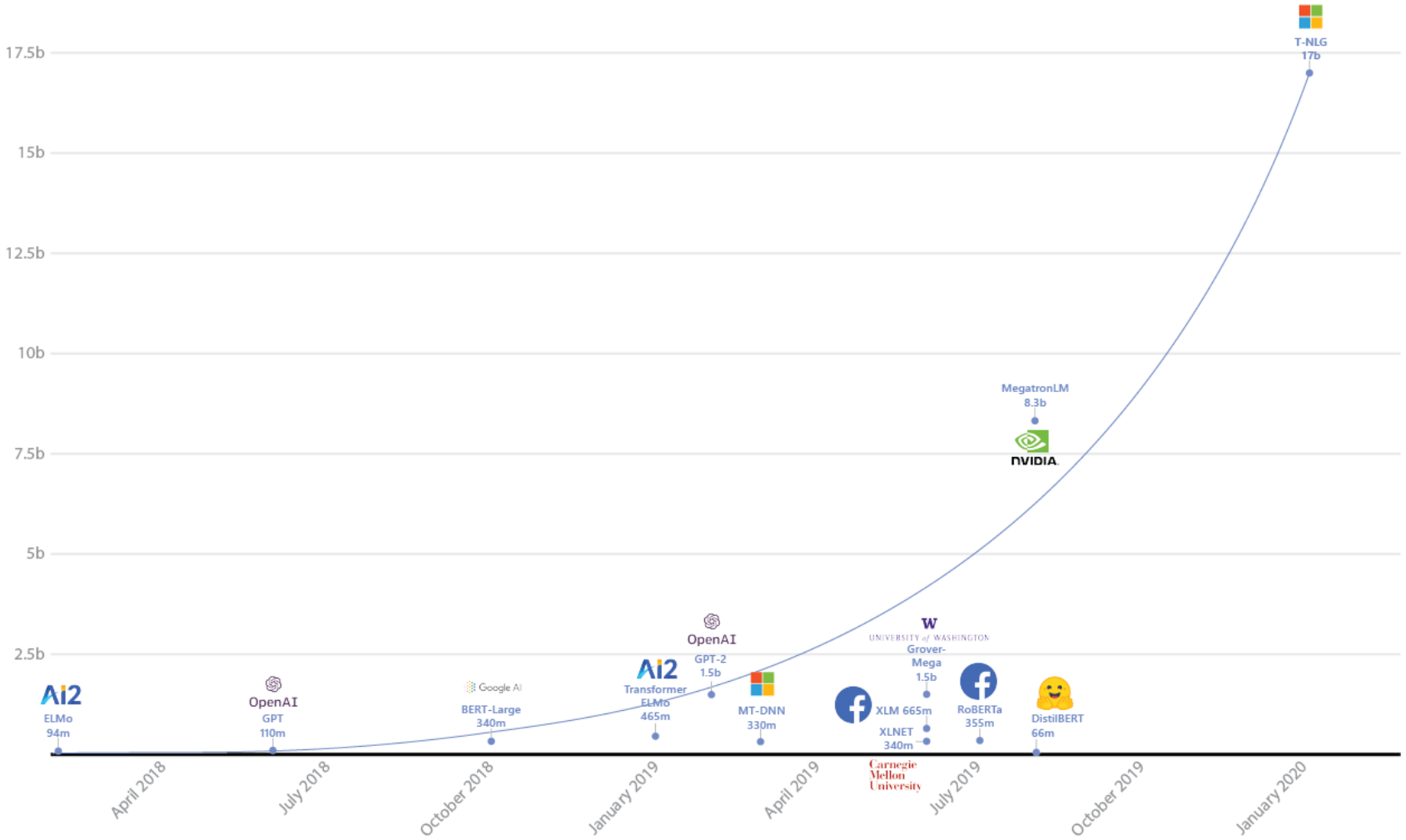
Multi-task Learning with BERT

- Multi-task learning happens in the fine-tuning phase.
- New SOTA on ten NLU tasks



Liu et al., 2019, Multi-Task Deep Neural Networks for Natural Language Understanding

PLM recent progress: model size



Turing-NLG: A 17-billion-parameter language model by Microsoft, Microsoft Research Blog

TinyBERT: Distilling BERT for NLU

- TinyBERT is proposed to execute on resource-restricted devices, for example, mobile phones.
- A novel transformer distillation method specially designed for knowledge distillation (KD) for transformer-based models.
- A new two-stage learning framework, which performs at both the pre-training and task-specific learning stages.
- A novel data augmentation technique is used in the second stage distillation.

Jiao et al., Distilling BERT for Natural Language Understanding, https://arxiv.org/abs/1909.10351, 2019

TinyBERT: Distilling BERT for NLU

- TinyBERT is empirically effective and achieves more than 96% the performance of teacher BERT-base on GLUE benchmark while being 7.5x smaller and 9.4x faster on inference.

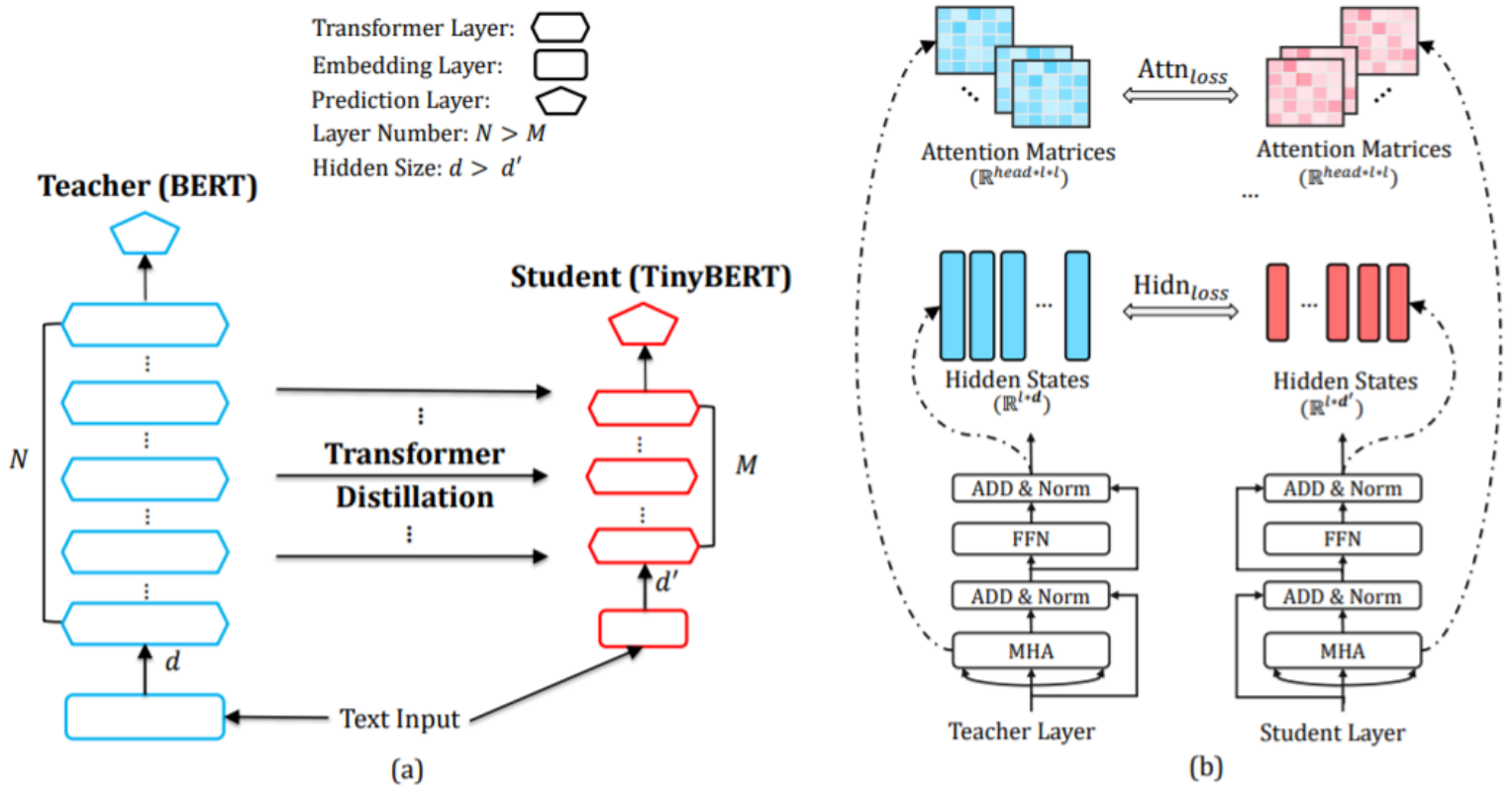
System	MNLI-m	MNLI-mm	QQP	SST-2	QNLI	MRPC	RTE	CoLA	STS-B	Average
$\text{BERT}_{\text{BASE}}$ (Google)	84.6	83.4	71.2	93.5	90.5	88.9	66.4	52.1	85.8	79.6
$\text{BERT}_{\text{BASE}}$ (Teacher)	83.9	83.4	71.1	93.4	90.9	87.5	67.0	52.8	85.2	79.5
$\text{BERT}_{\text{SMALL}}$	75.4	74.9	66.5	87.6	84.8	83.2	62.6	19.5	77.1	70.2
Distilled $\text{BiLSTM}_{\text{SOFT}}$	73.0	72.6	68.2	90.7	-	-	-	-	-	-
BERT-PKD	79.9	79.3	70.2	89.4	85.1	82.6	62.3	24.8	79.8	72.6
DistilBERT	78.9	78.0	68.5	91.4	85.2	82.4	54.1	32.8	76.1	71.9
TinyBERT	82.5	81.8	71.3	92.6	87.7	86.4	62.9	43.3	79.9	76.5

System	Layers	Hidden Size	Feed-forward Size	Model Size	Inference Time
$\text{BERT}_{\text{BASE}}$ (Teacher)	12	768	3072	109M($\times$1.0)	188s($\times$1.0)
Distilled $\text{BiLSTM}_{\text{SOFT}}$	1	300	400	10.1M($\times$10.8)	24.8s($\times$7.6)
BERT-PKD/DistilBERT	4	768	3072	52.2M($\times$2.1)	63.7s($\times$3.0)
TinyBERT	4	312	1200	14.5M($\times$7.5)	19.9s($\times$9.4)

Jiao et al., Distilling BERT for Natural Language Understanding, https://arxiv.org/abs/1909.10351, 2019

TinyBERT: Distilling BERT for NLU

- Transformer distillation:



Jiao et al., Distilling BERT for Natural Language Understanding, https://arxiv.org/abs/1909.10351, 2019

TinyBERT: Distilling BERT for NLU

- Two-stage learning framework:

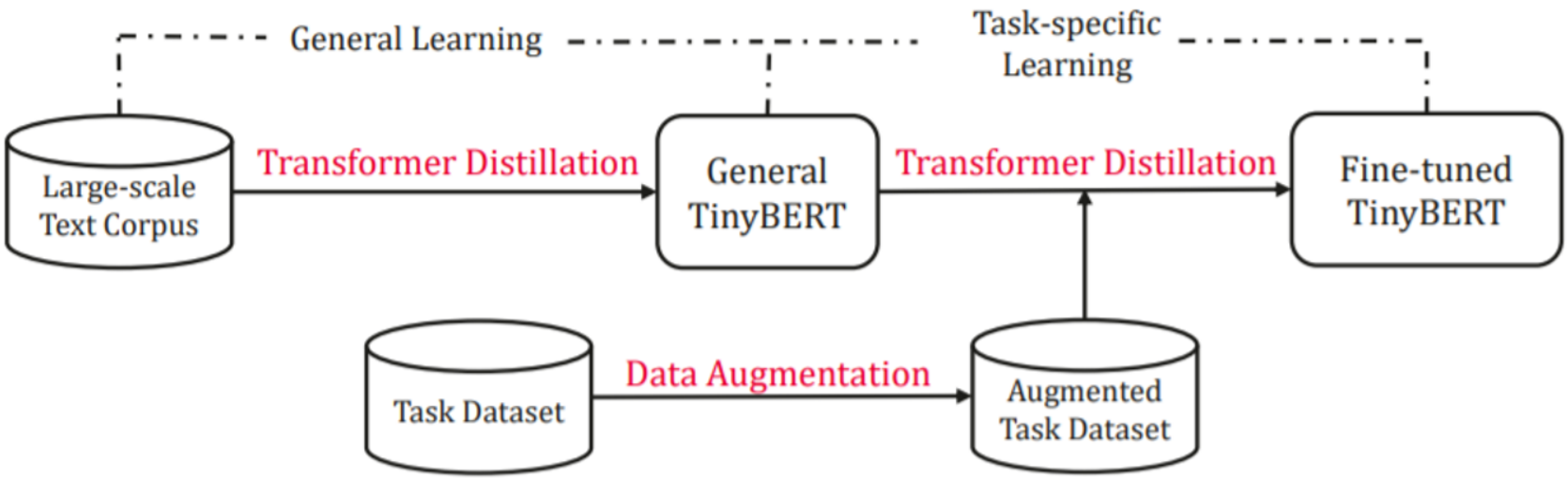


Figure 2: The illustration of TinyBERT learning

System	MNLI-m	MNLI-mm	MRPC	CoLA	Average
TinyBERT	82.8	82.9	85.8	49.7	75.3
No GD	82.5	82.6	84.1	40.8	72.5
No TD	80.6	81.2	83.8	28.5	68.5
No DA	80.5	81.0	82.4	29.8	68.4

Jiao et al., Distilling BERT for Natural Language Understanding, https://arxiv.org/abs/1909.10351, 2019

ERNIE-Baidu

- ERNIE-Baidu
  - Entity-level masking
  - Phrase-level masking
- Outperform BERT on 5 Chinese language processing

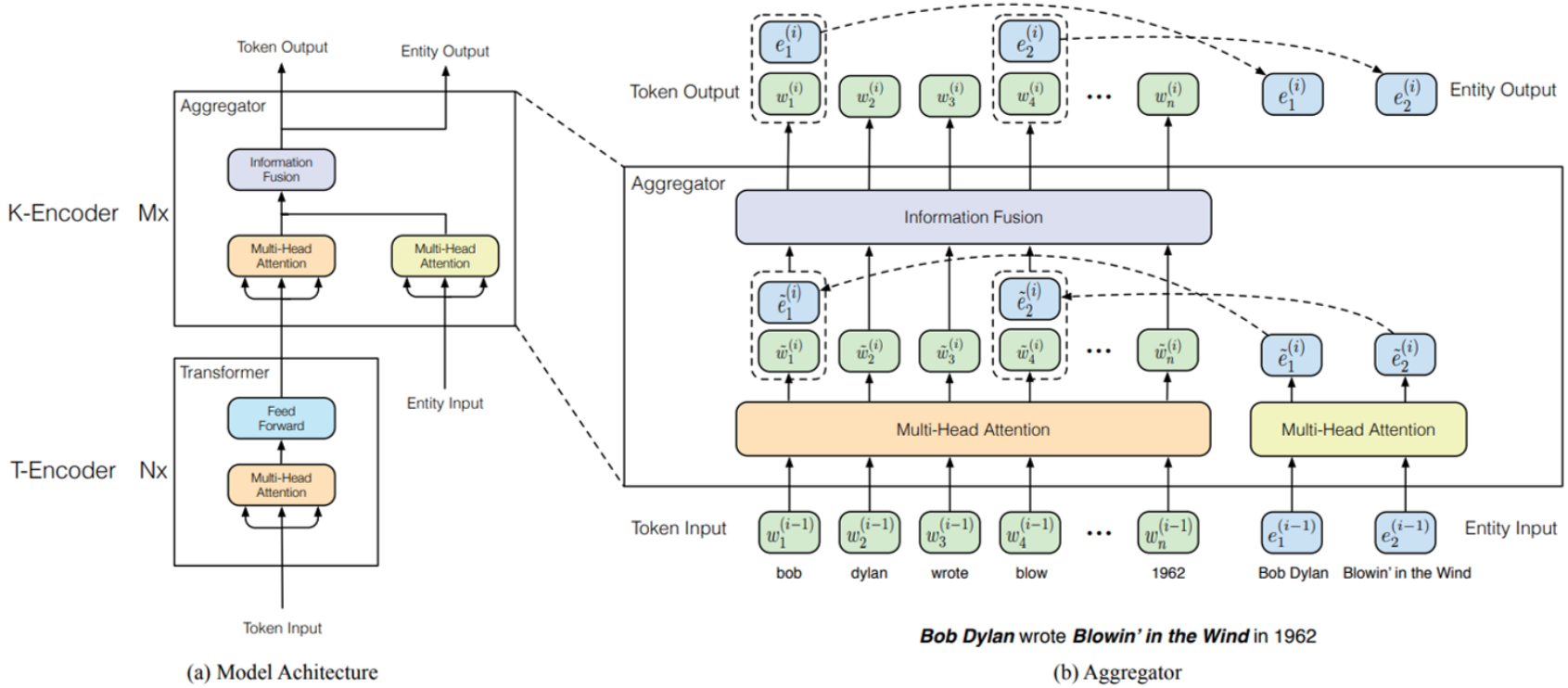
Sentence	Harry	Potter	is	a	series	of	fantasy	novels	written	by	British	author	J.	K.	Rowling
Basic-level Masking	[mask]	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	J.	[mask]	Rowling
Entity-level Masking	Harry	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]
Phrase-level Masking	Harry	Potter	is	[mask]	[mask]	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]

pre-train dataset size	mask strategy	dev Accuracy	test Accuracy
10% of all	word-level(chinese character)	77.7%	76.8%
10% of all	word-level&phrase-level	78.3%	77.3%
10% of all	word-level&phrase-leve&entity-level	78.7%	77.6%
all	word-level&phrase-level&entity-level	79.9 %	78.4%

Sun et al., 2019, ERNIE: Enhanced Representation through Knowledge Integration

ERNIE-Tsinghua

- ERNIE-Tsinghua
  - Entities from knowledge-graph
  - The information fusion adopt a two-layer feed-forward net[illegible]k



Zhang et al., 2019, ERNIE: Enhanced Language Representation with Informative Entities

ERNIE-Tsinghua

- ERNIE-Tsinghua
  - Entity embeddings are pre-trained with TransE
  - Knowledge encoder + Entity prediction task
- Results
  - Comparable performance on normal NLP tasks
  - Better performance on knowledge-driven tasks

Entity Typing tasks:

Model	P	R	F1
NFGEC (LSTM)	68.80	53.30	60.10
UFET	77.40	60.60	68.00
BERT	76.37	70.96	73.56
ERNIE	**78.42**	**72.90**	**75.56**

Table 3: Results of various models on Open Entity (%).

Relation Classification tasks:

Model	FewRel			TACRED		
	P	R	F1	P	R	F1
CNN	69.51	69.64	69.35	70.30	54.20	61.20
PA-LSTM	-	-	-	65.70	64.50	65.10
C-GCN	-	-	-	69.90	63.30	66.40
BERT	85.05	85.11	84.89	67.23	64.81	66.00
ERNIE	88.49	88.44	**88.32**	69.97	66.08	**67.97**

Table 5: Results of various models on FewRel and TACRED (%).

Zhang et al., 2019, ERNIE: Enhanced Language Representation with Informative Entities

Multilingual BERT

- It is impractical to have individual models for each language
  - Learning a universal model for all languages
- Multilingual BERT
  - The languages chosen were the top 100 languages with the largest Wikipedias.
  - The entire Wikipedia dump for each language (excluding user and talk pages) was taken as the training data for each language
  - Balance the data: high-resource languages like English will be under-sampled, and low-resource languages like Icelandic will be over-sampled.

`https://github.com/google-research/bert/blob/master/multilingual.md`

Multilingual BERT

- Multilingual BERT (continued)
  - For tokenization, a 110k shared WordPiece vocabulary was used.
  - The word counts are weighted the same way as the data, so low-resource languages are upweighted by some factor.
  - We intentionally do not use any marker to denote the input language (so that zero-shot training can work).

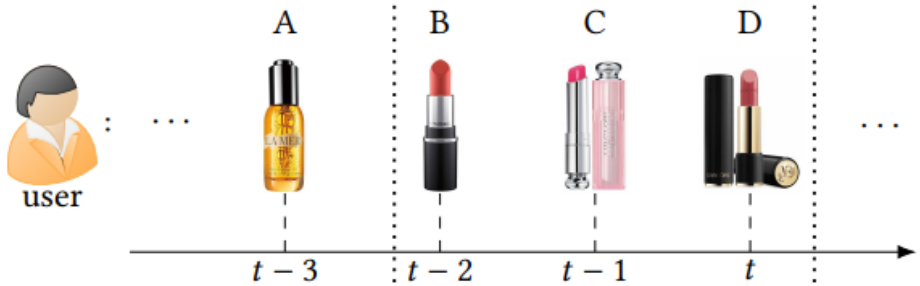
System	English	Chinese	Spanish	German	Arabic	Urdu
XNLI Baseline - Translate Train	73.7	67.0	68.8	66.5	65.8	56.6
XNLI Baseline - Translate Test	73.7	68.3	70.7	68.7	66.8	59.3
BERT - Translate Train Cased	**81.9**	**76.6**	**77.8**	**75.9**	**70.7**	61.6
BERT - Translate Train Uncased	81.4	74.2	77.3	75.2	70.5	61.7
BERT - Translate Test Uncased	81.4	70.1	74.9	74.4	70.4	**62.1**
BERT - Zero Shot Uncased	81.4	63.8	74.3	70.5	62.1	58.3

However, for high-resource languages, the multilingual model is somewhat worse than a single-language model.

https://github.com/google-research/bert/blob/master/multilingual.md

BERT for Recommendation: BERT4Rec

- Drawbacks of sequence modeling
  - Sequential dependencies over long time scales (e.g., from A to [B,C,D]) vs random actions in a short period (e.g., [B,C,D])



- BERT: jointly conditioning on both left and right context

Table 3: Analysis on bidirection and Cloze with $d = 256$

Model	Beauty			ML-1m		
	HR@10	NDCG@10	MRR	HR@10	NDCG@10	MRR
SASRec	0.2653	0.1633	0.1536	0.6629	0.4368	0.3790
BERT4Rec (1 mask)	0.2940	0.1769	0.1618	0.6869	0.4696	0.4127
BERT4Rec	0.3025	0.1862	0.1701	0.6970	0.4818	0.4254

Sun et al., 2019, BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer

Content

1. Subword level and character level NMT
2. Transformer-based NMT
3. Pre-trained language models (PLMs)