GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Deep Learning

Lecture 12: Natural language processing

Alexander Ecker
Institut für Informatik, Uni Göttingen

alxecker

https://alexanderecker.wordpress.com

– Credit: some of the slides based on Fei Fei Li, Justin Johnson and Serena Yeung's slides –

Today’s topics

Working with language data

Word2Vec

ELMo

Transformer

BERT

Working with text data

Processing text: Tokenization

Input: Friends, Romans, Countrymen, lend me your ears.

Output: Friends | , | Romans | , | Countrymen | , | lend | me | your | ears | .

Tokenization is a non-trivial problem

O'Neill

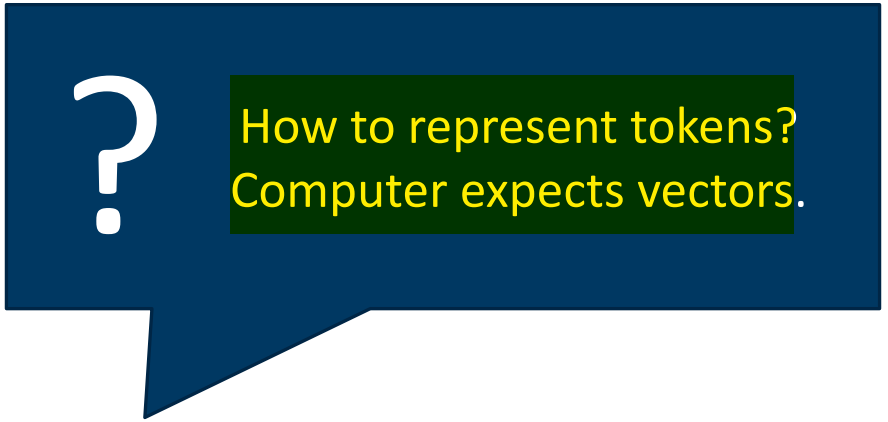
- o' | neill
- o | neill
- oneill
- o'neill

aren't

- aren't
- arent
- are | n't
- aren | t

Processing text: how to represent tokens?

Tokenized: Friends , Romans , Countrymen , lend me your ears .



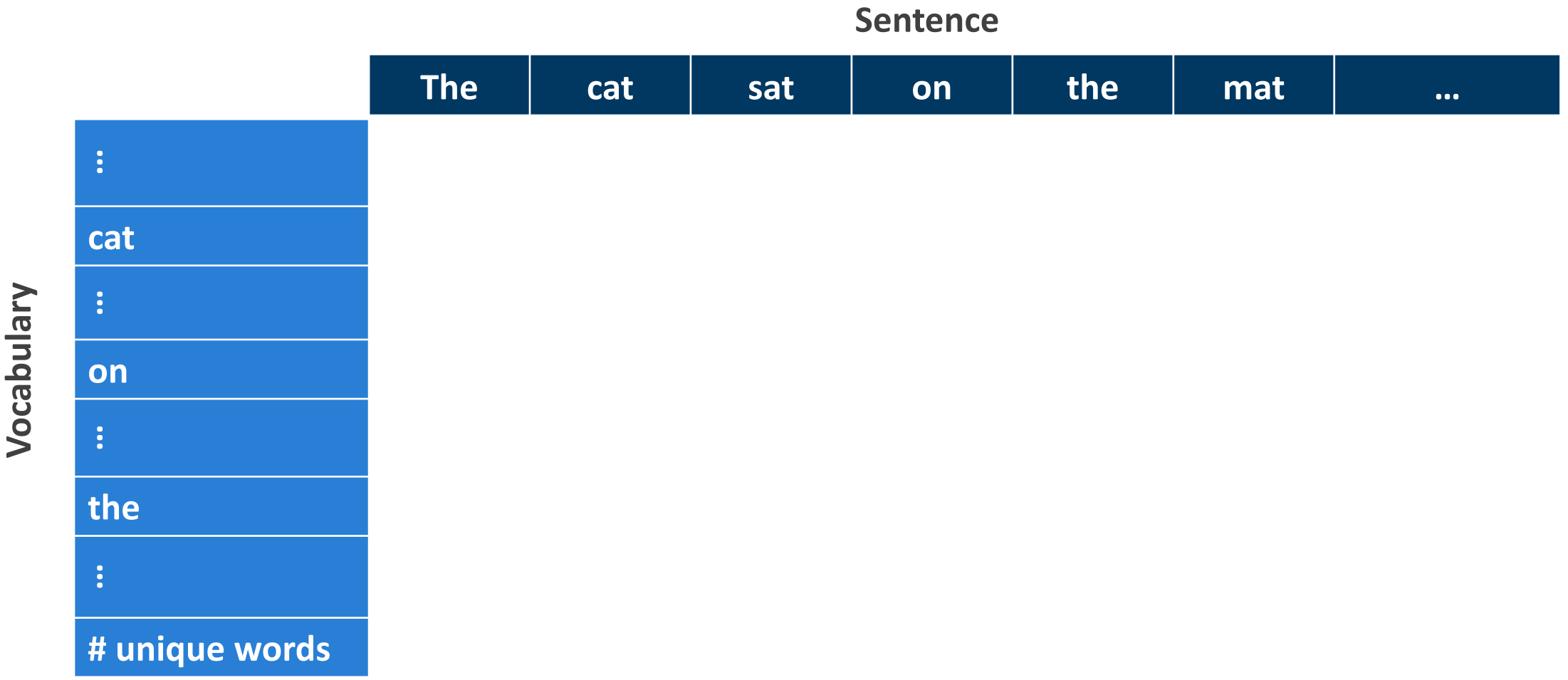
Transform them into a vector and take the characters and select a representation

One-hot encodings of tokens

Sentence

The	cat	sat	on	the	mat	…

One-hot encodings of tokens



One-hot encodings of tokens

First tokenise and than do one-hot encoding
-> take the position

Vocabulary	Sentence						
	The	**cat**	**sat**	**on**	**the**	**mat**	**…**
⋮							
cat	0	**1**	0	0	0	0	
⋮							
on	0	0	0	**1**	0	0	
⋮							
the	**1**	0	0	0	**1**	0	
⋮							
# unique words	0	0	0	0	0	0	

Word2Vec

SEMANTIC EMBEDDINGS OF WORDS

Word embeddings

Generating embeddings for each word, that keeps relations
One-hot encoding do not keep any relations

"King" =

[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

represent the encoding with colors

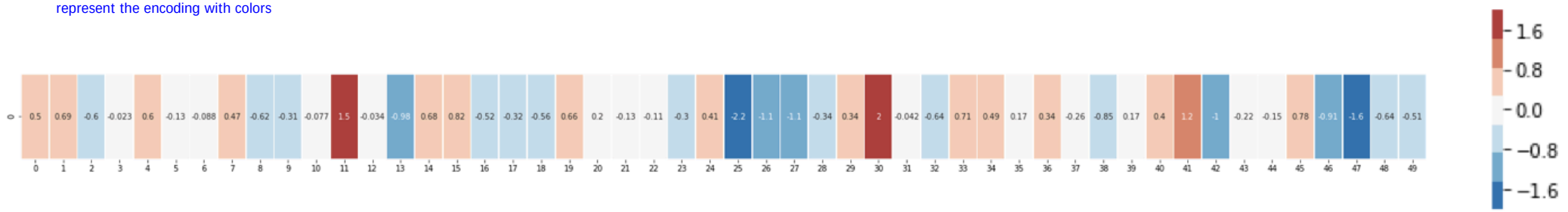


Image source: http://jalammar.github.io/illustrated-word2vec/

Word embeddings

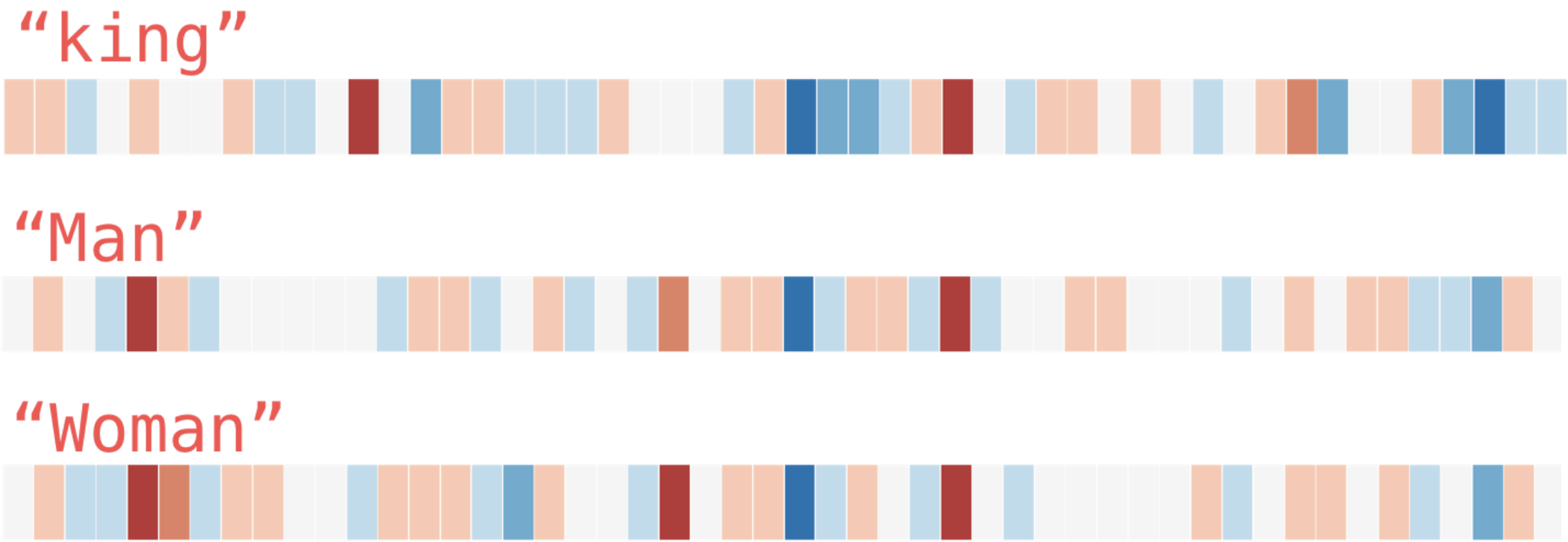


Image source: http://jalammar.github.io/illustrated-word2vec/

Word embeddings

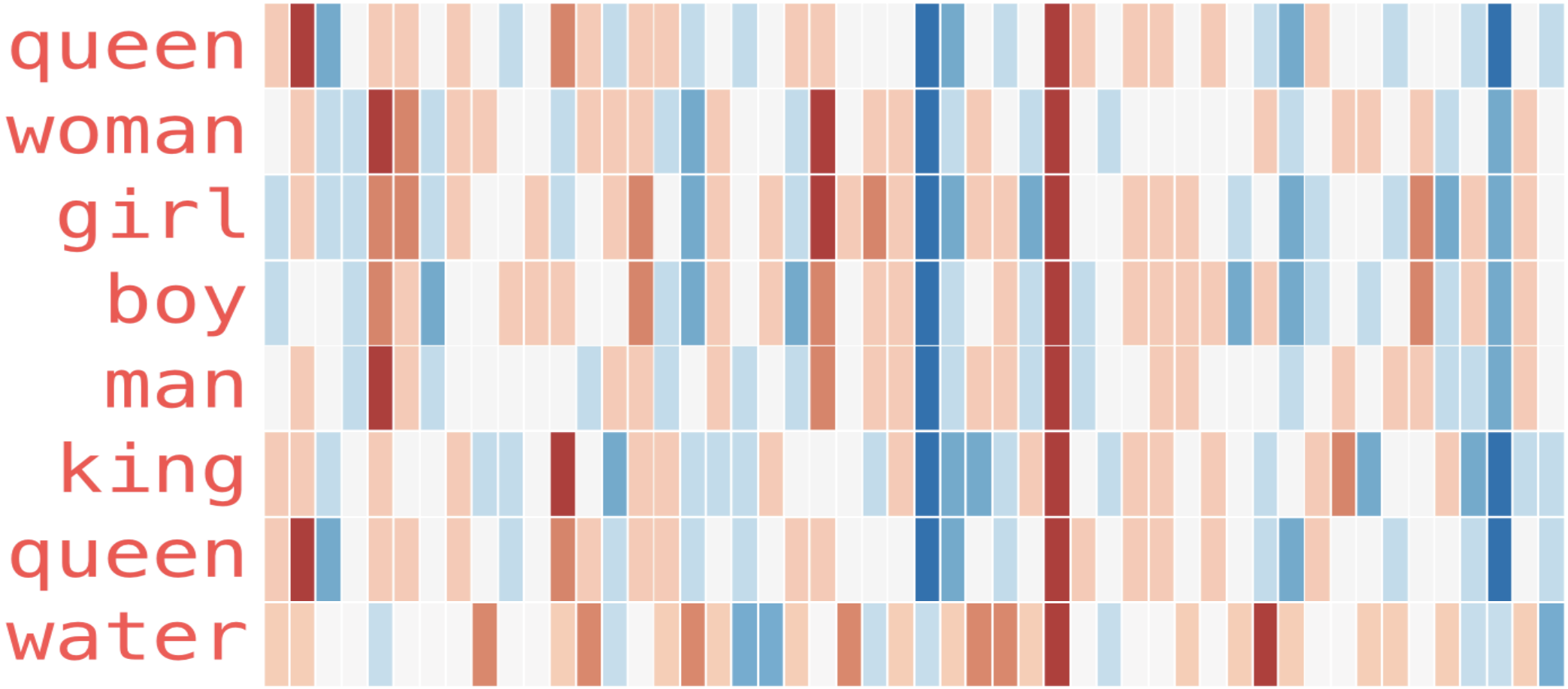
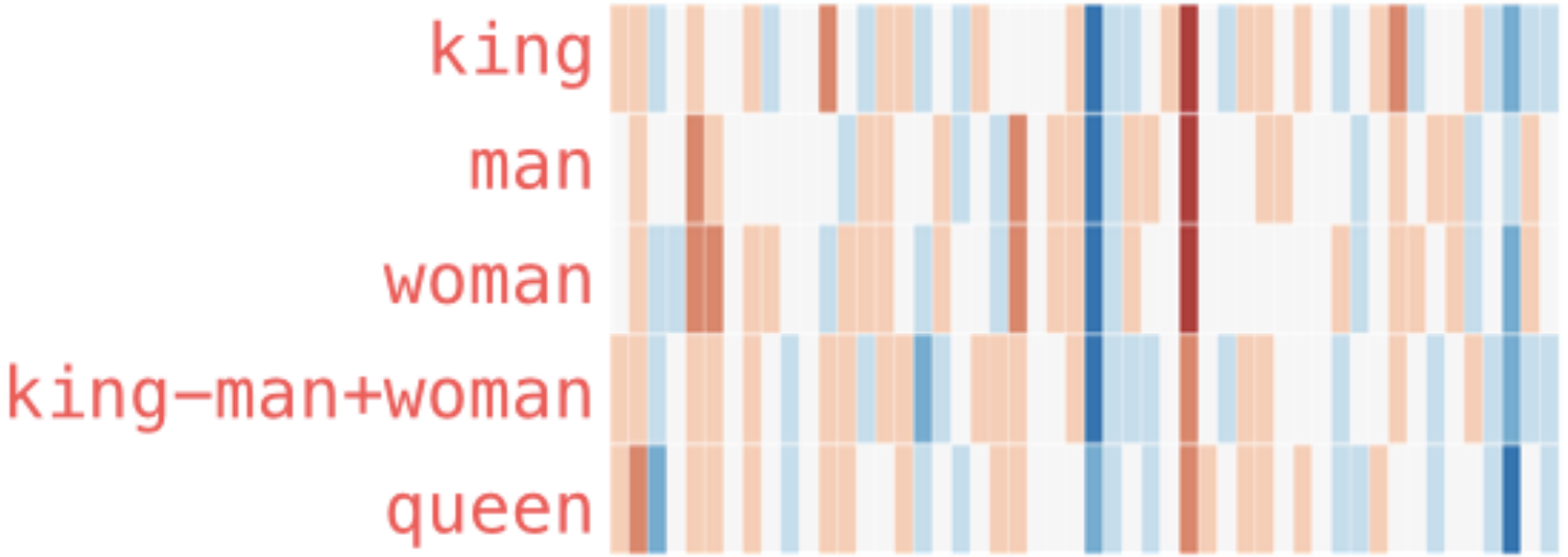


Image source: http://jalammar.github.io/illustrated-word2vec/

Math with word embeddings (1/2)

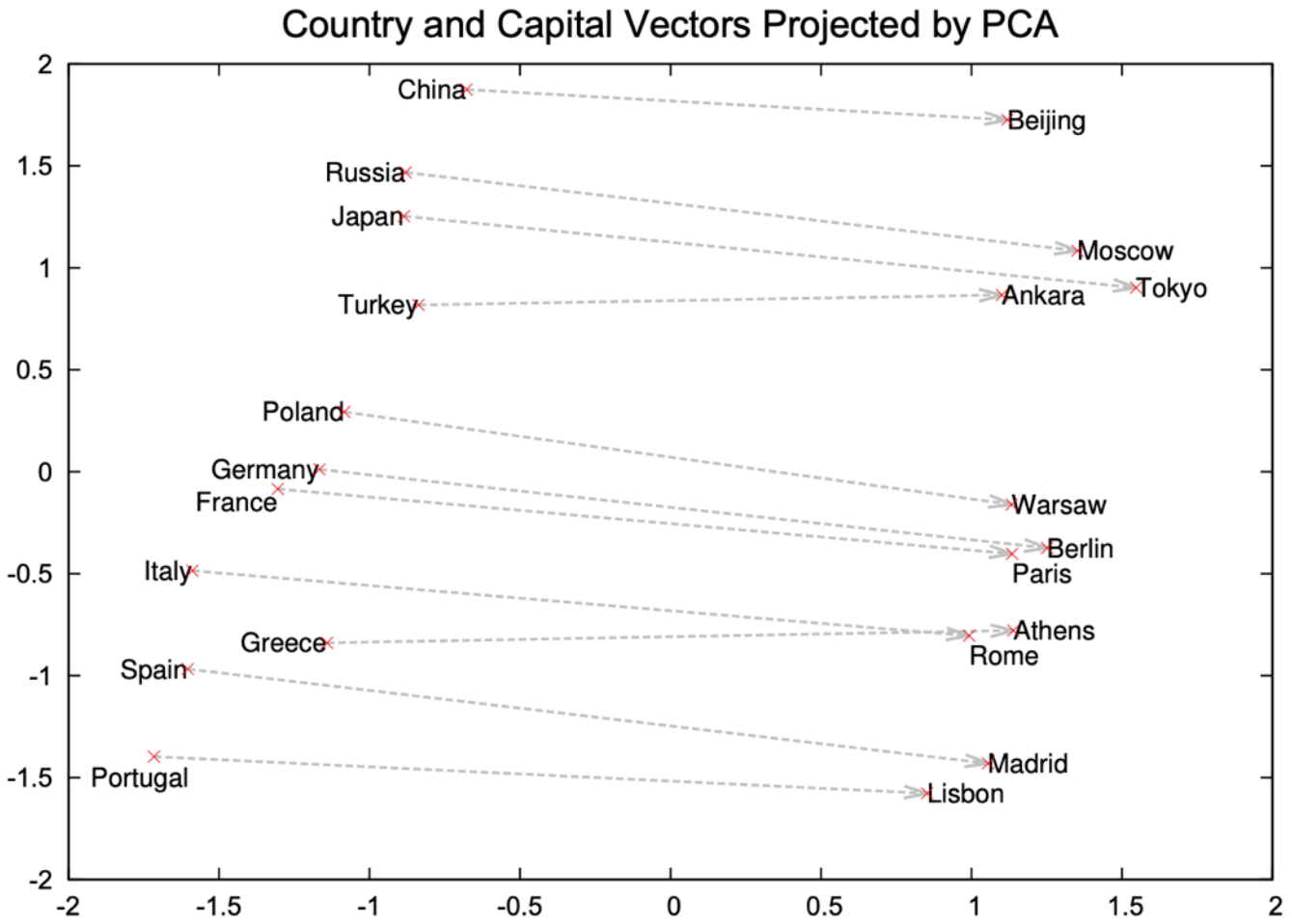


$$\frac{u^{\mathrm{T}}v}{\|u\| \cdot \|v\|} = 0.85$$

What's the closest word in the vocabulary of 40,000 words?

Image source: http://jalammar.github.io/illustrated-word2vec/

Math with word embeddings (2/2)



Mikolov et al., *NeurIPS* 2013

Training

-> download the data and train a language model on this data

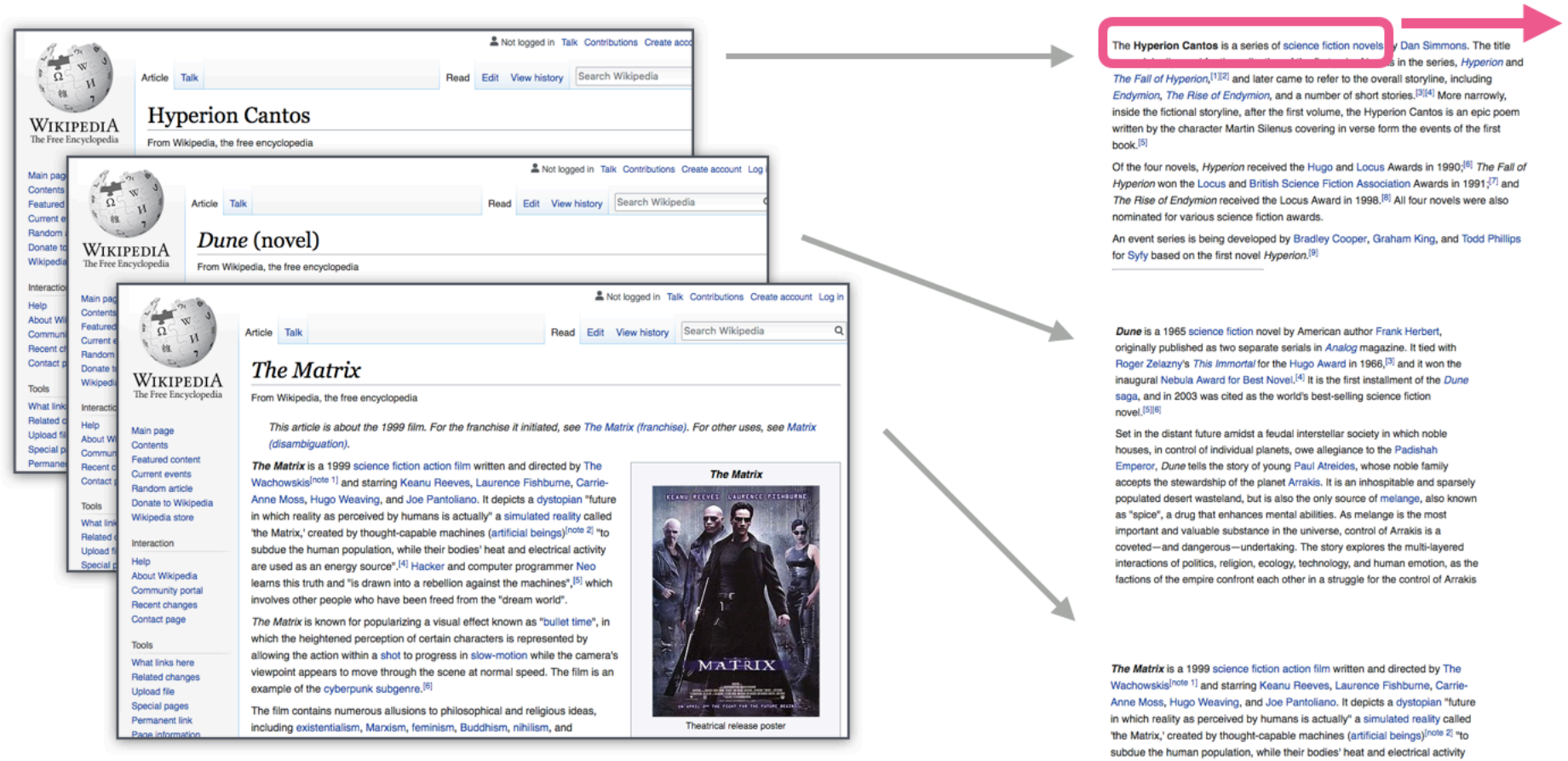


Image source: http://jalammar.github.io/illustrated-word2vec/

1 Continuous bag of words (CBOW)

take a sentence, a sliding window is sliding over the text
_> take the actual 2 words as an input
-> the third word as an output
-> do it like this for the whole text

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	…

Dataset

input 1	input 2	output
thou	shalt	not

Image source: http://jalammar.github.io/illustrated-word2vec/

1 Continuous bag of words (CBOW)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	…
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make

Image source: http://jalammar.github.io/illustrated-word2vec/

1 Continuous bag of words (CBOW)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	…
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make
not	make	a
make	a	machine
a	machine	in

Image source: http://jalammar.github.io/illustrated-word2vec/

1 Look both ways

Predicting is for us sometimes easy, but sometimes not that clear. If we have the words before and the word after it gets more clear
-> so not also take the words before as the input

Jane was hit by a _____

car, bus, lightning?

Jane was hit by a _____ bus

Image source: http://jalammar.github.io/illustrated-word2vec/

1 Continuous bag of words (CBOW)

Jane was hit by a _____ bus in

by	a	red	bus	in

input 1	input 2	input 3	input 4	output
by	a	bus	in	red

Image source: http://jalammar.github.io/illustrated-word2vec/

2 Skip-gram

now we have one word as an input (here: red) and wie try to predict 4 outputs (words around it).

Jane was hit by a red bus in

by	a	red	bus	in

input	output
red	by
red	a
red	bus
red	in

Image source: http://jalammar.github.io/illustrated-word2vec/

2 Skip-gram training data

Thou shalt not make a machine in the likeness of a human mind

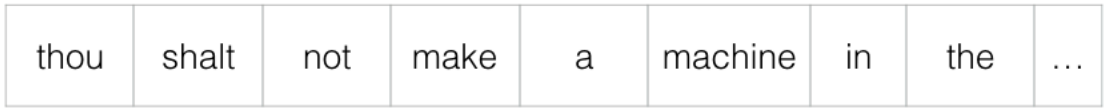
thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a

Image source: http://jalammar.github.io/illustrated-word2vec/

2 Skip-gram training data

Thou shalt not make a machine in the likeness of a human mind



input word	target word
not	thou
not	shalt
not	make
not	a

Image source: http://jalammar.github.io/illustrated-word2vec/

2 Skip-gram training data

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

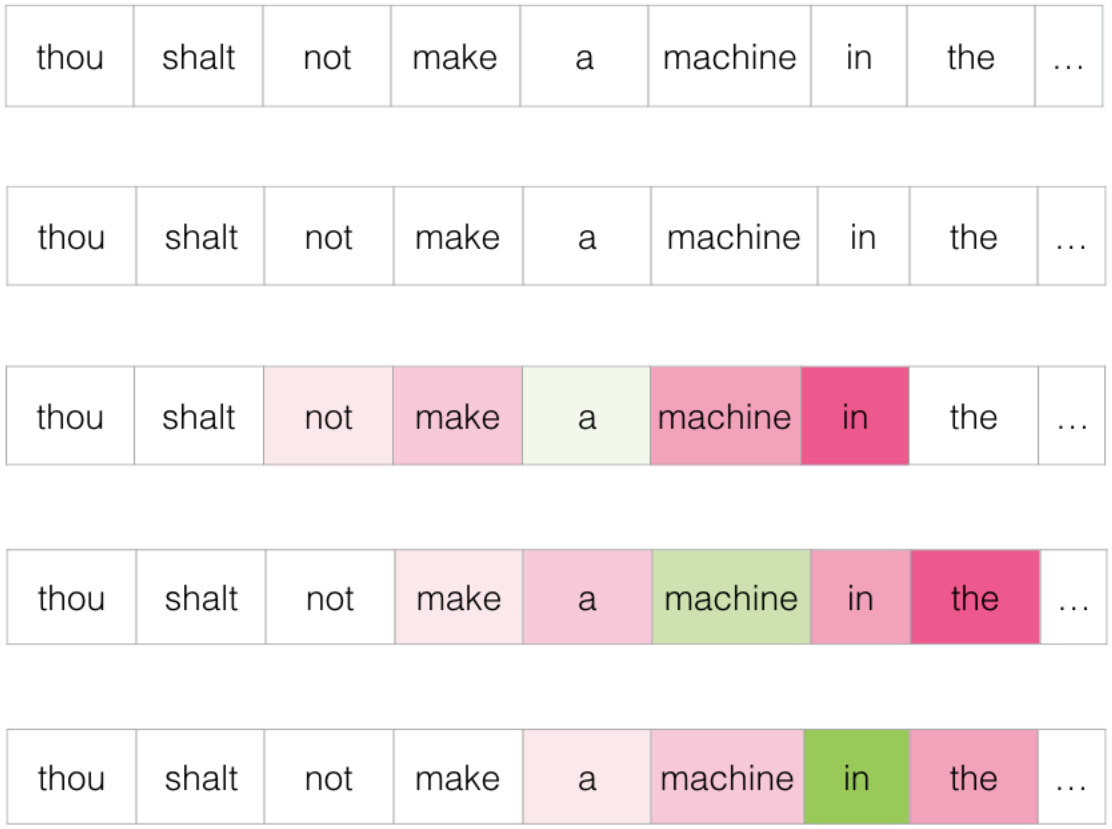
thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

Image source: http://jalammar.github.io/illustrated-word2vec/

2 Skip-gram training data

Thou shalt not make a machine in the likeness of a human mind



input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

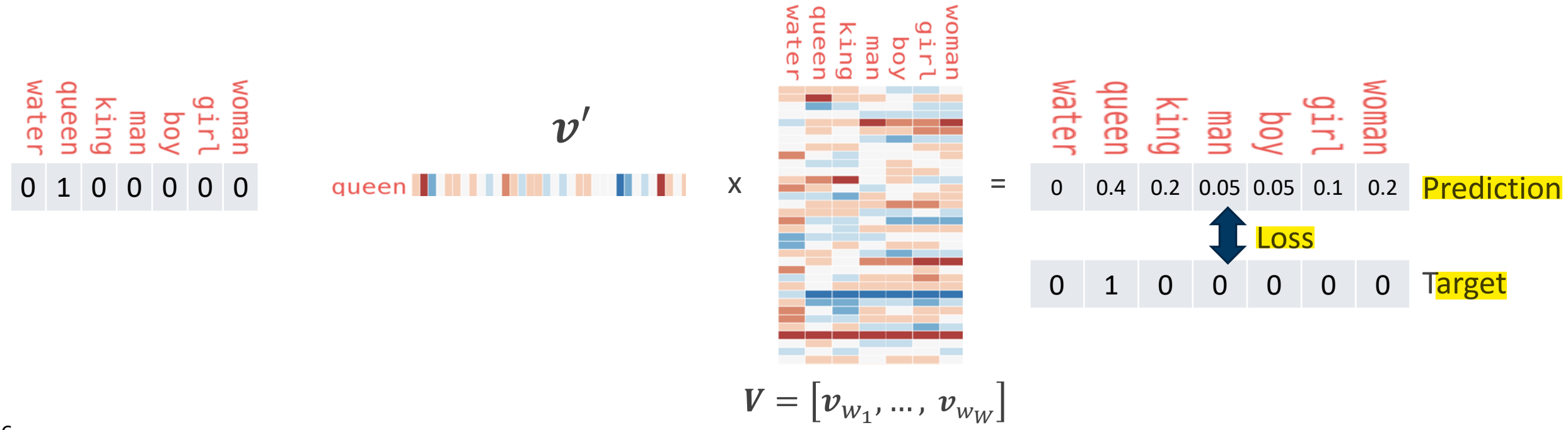
Image source: http://jalammar.github.io/illustrated-word2vec/

Word2Vec model

we have two vectors as an input, encoding and decoding vectors. We try to maximize the product and normalize it with a softmax (sum -> Nenner).

-> linear encoding model

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\mathrm{T}} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_{w_O}}^{\mathrm{T}} v_{w_I}\right)}$$



$$V = [v_{w_1}, \ldots, v_{w_W}]$$

ELMo

EMBEDDINGS FROM **L**ANGUAGE **MO**DELS

Problem with Word2Vec: understanding context

"bank"

"He robbed the bank" "He sat on the bank"

Context matters for the meaning of words!

ELMo

Training via language modeling

Input sequence: Let’s stick to improvisation for this skit.

Task: predict next word

Let’s stick to improvisation for this _____

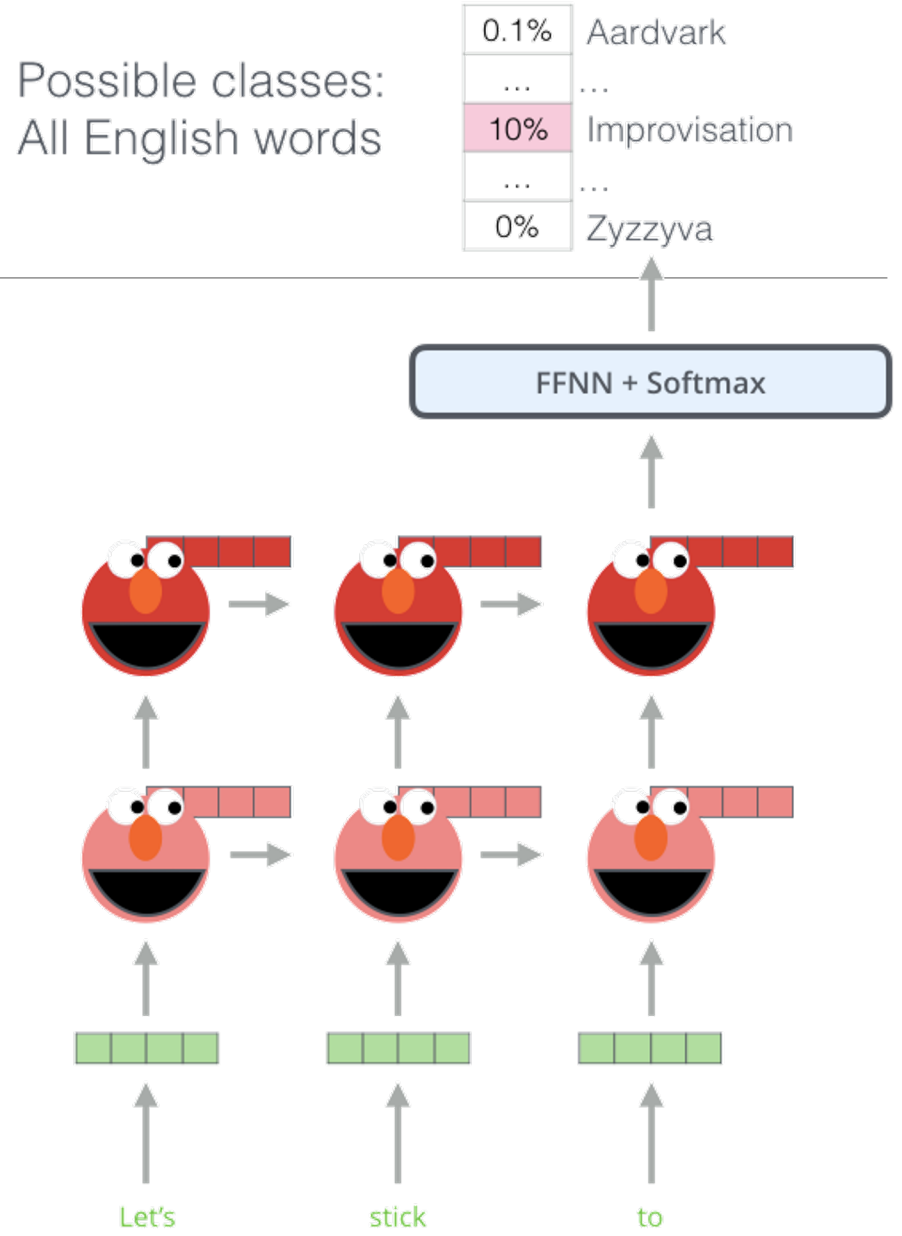
Not only the last word, predict also the other words before!

ELMo

Training via language modeling

Task: predict next word

Our embedding depend on the whole sequence!
-> context word embedding

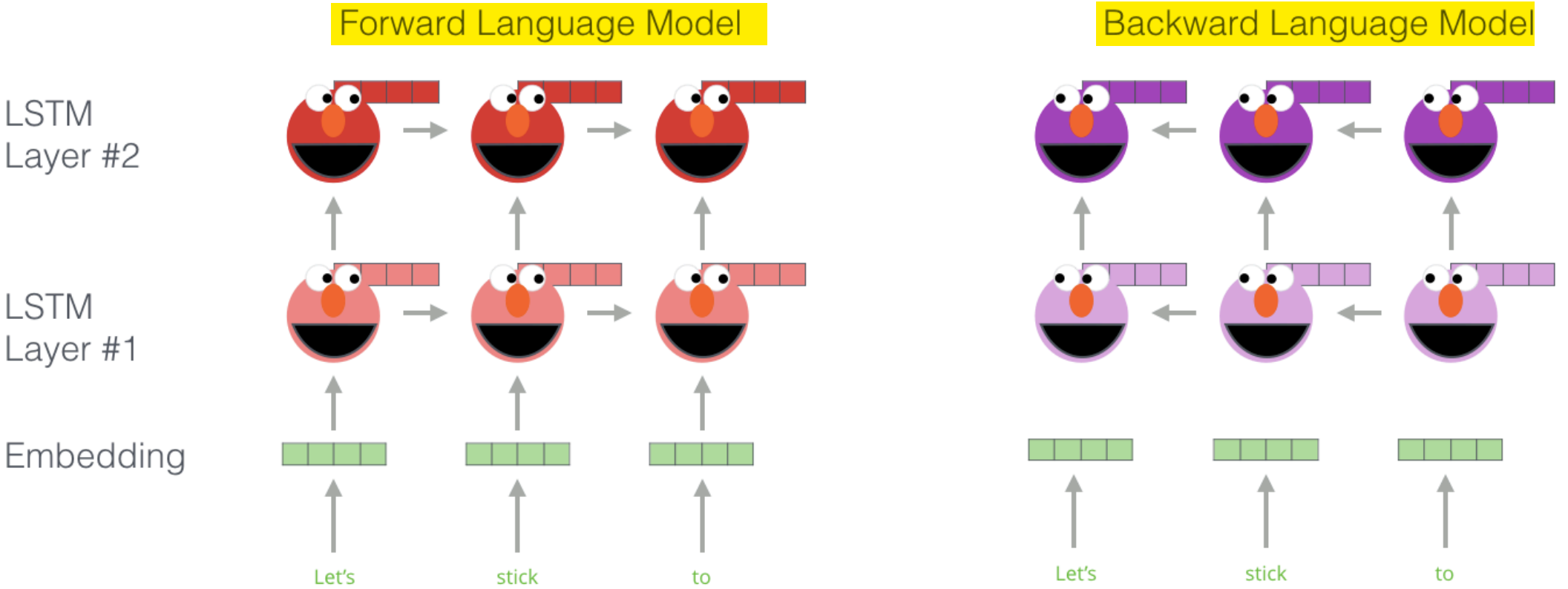


ELMo: forward + backward language modeling

Embedding of “stick” in “Let’s stick to” - Step #1

Have a different set of parameters, but overall the same idea as the forward process

cardinate the embedding from the forward and the backward to one embedding
-> this is than our final embedding



Transformer

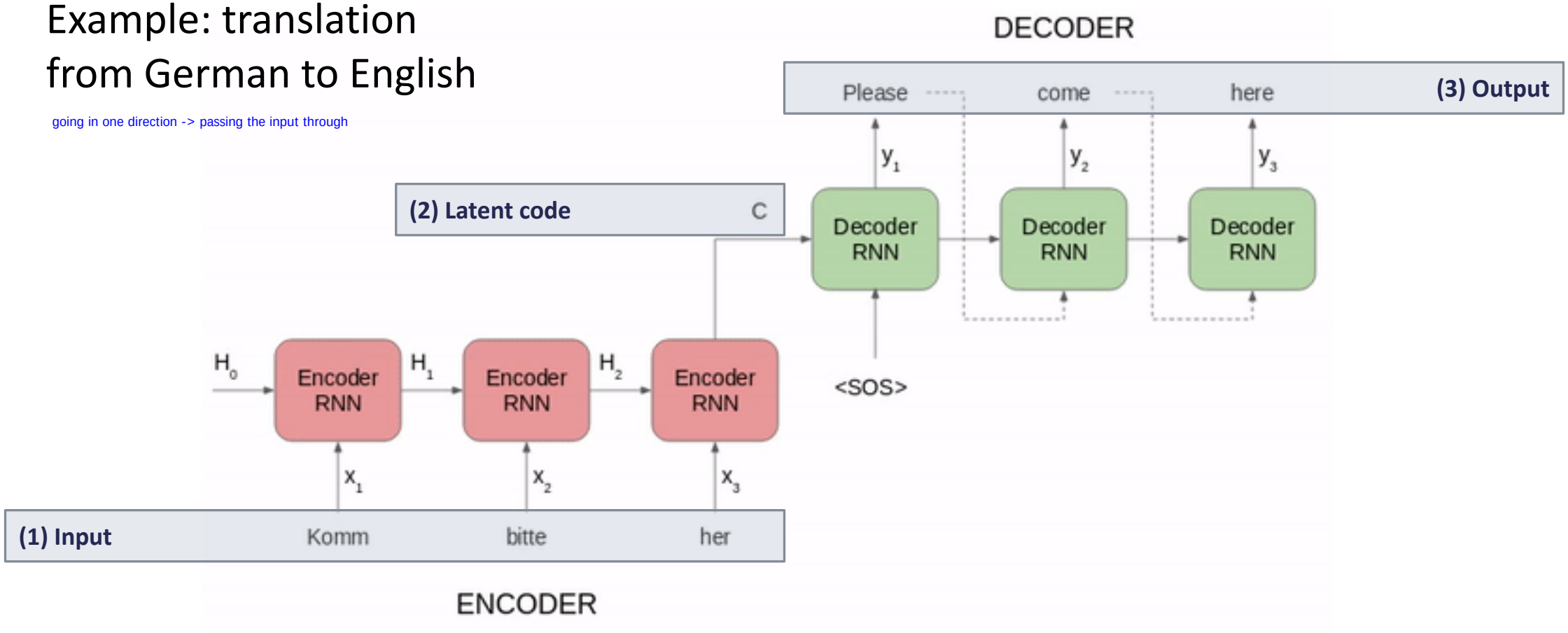
REPLACING RECURRENT PROCESSING BY ATTENTION

similar the image processing -> same idea

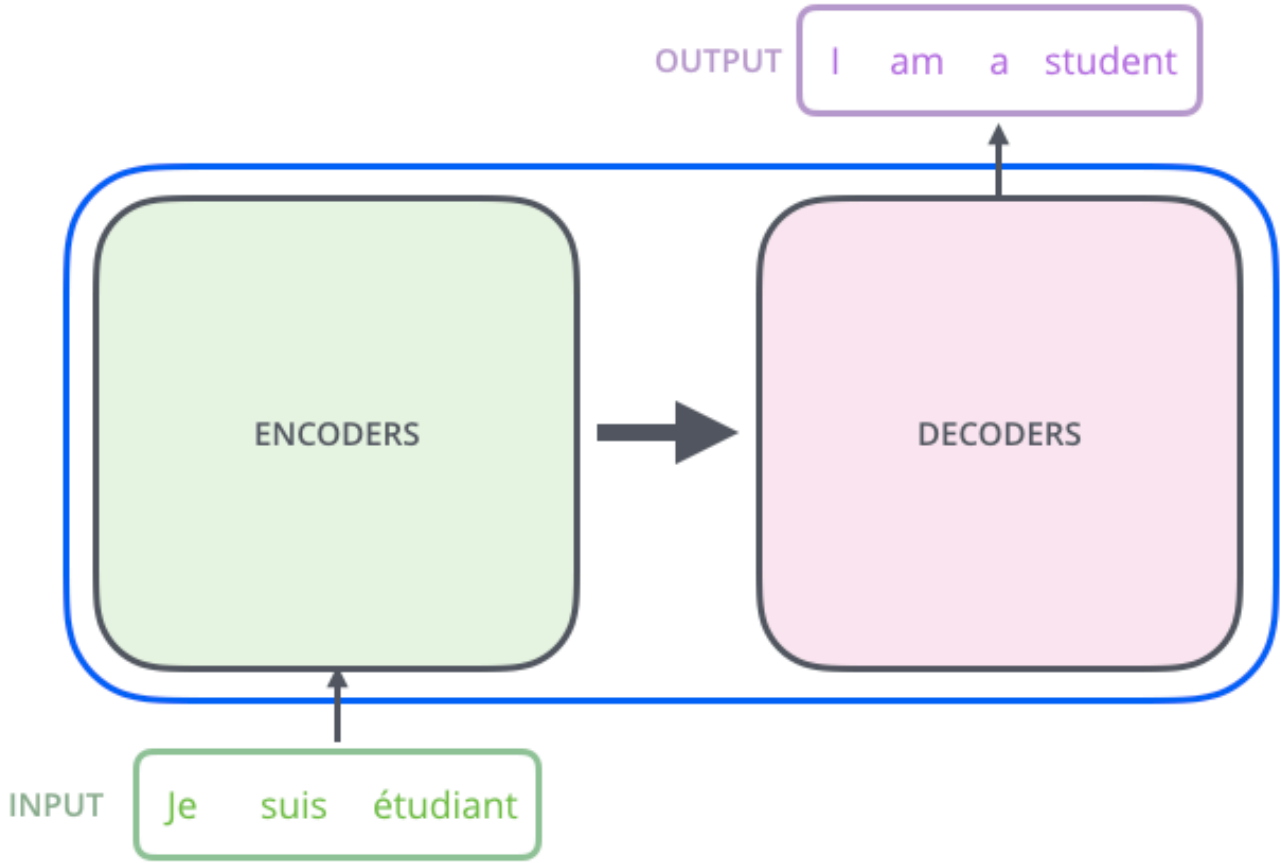
Recap: sequence-to-sequence (seq2seq) models

Example: translation
from German to English

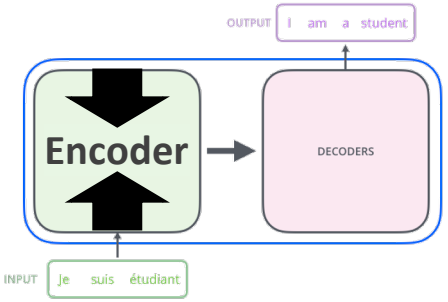
going in one direction -> passing the input through



Transformer: architecture overview



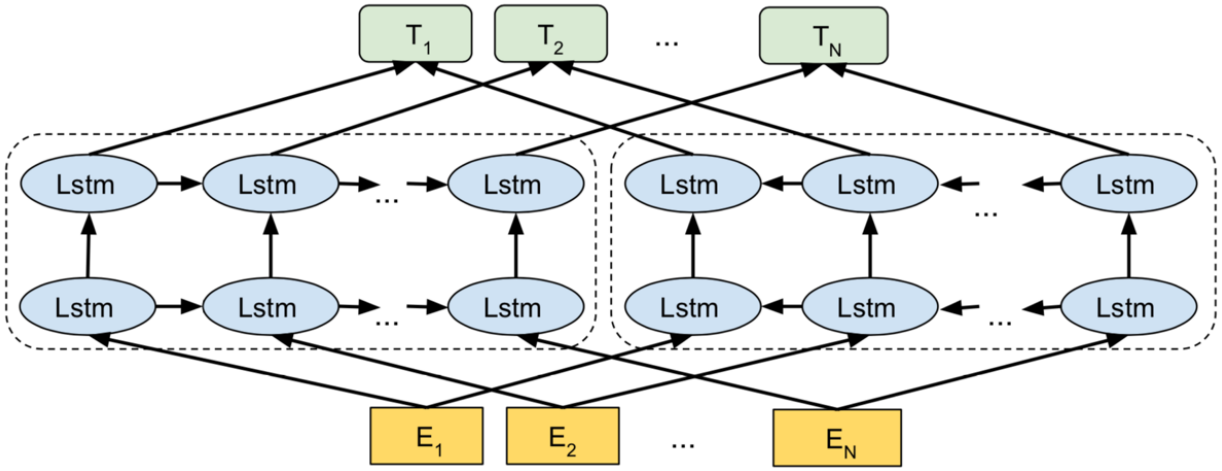
Transformer: idea



Only difference is how the Encoder looks like!

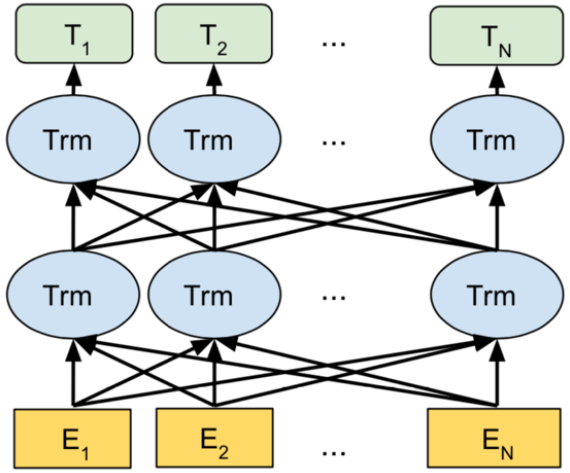
getting grid of the concept of time (passing forward and backward)
every Encoders/ Tokens can comunicate with each other

Example: ELMo



Bidirectional encoding via recurrent net (LSTM)

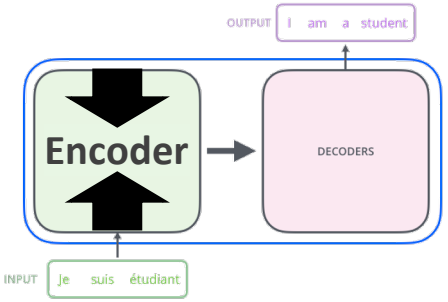
Transformer: Encoder



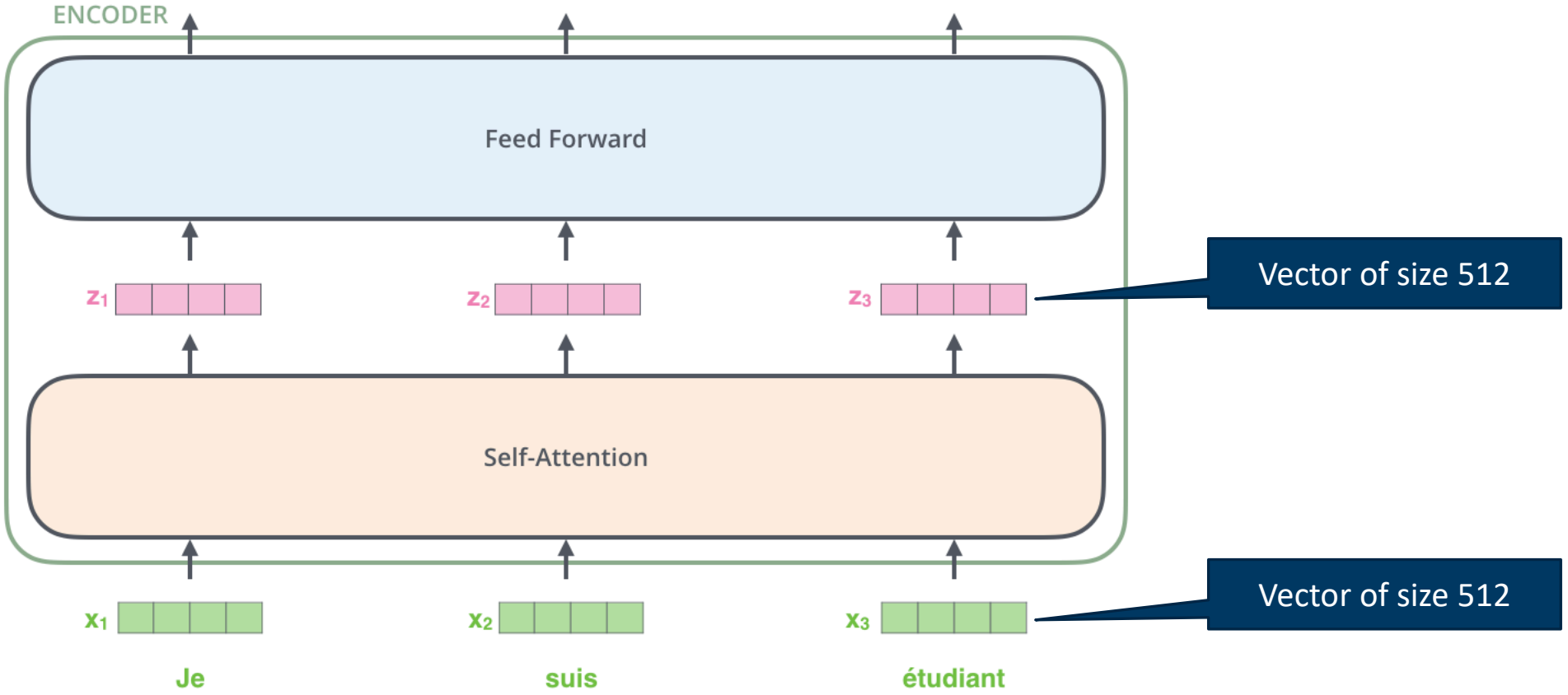
All-to-all connections within sequence
→ **Attention**

Devlin et al., *NAACL* 2019

Transformer: Encoder

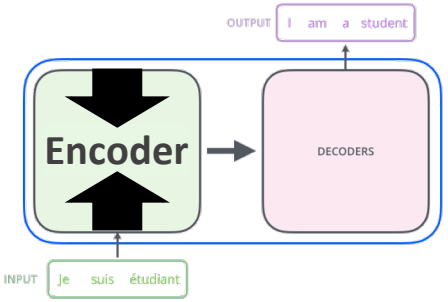


2 Steps: Self-Attention, fully connected Feed Forward



https://jalammar.github.io/illustrated-transformer/

Transformer: Encoder



ENCODER #2

ENCODER #1

The embedding is on token level

The ffed forward has a weight matrix

r_1 r_2 ...

Feed Forward Neural Network

Feed Forward Neural Network

z_1 z_2 ...

Self-Attention

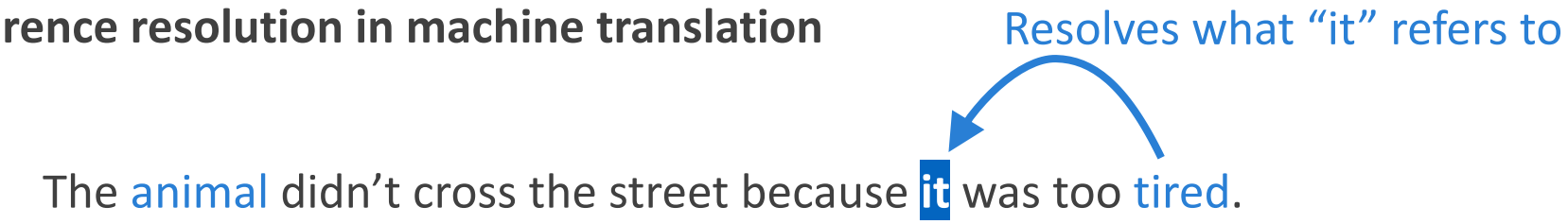
x_1 x_2 ...

Thinking Machines

https://jalammar.github.io/illustrated-transformer/

Why do we need attention?

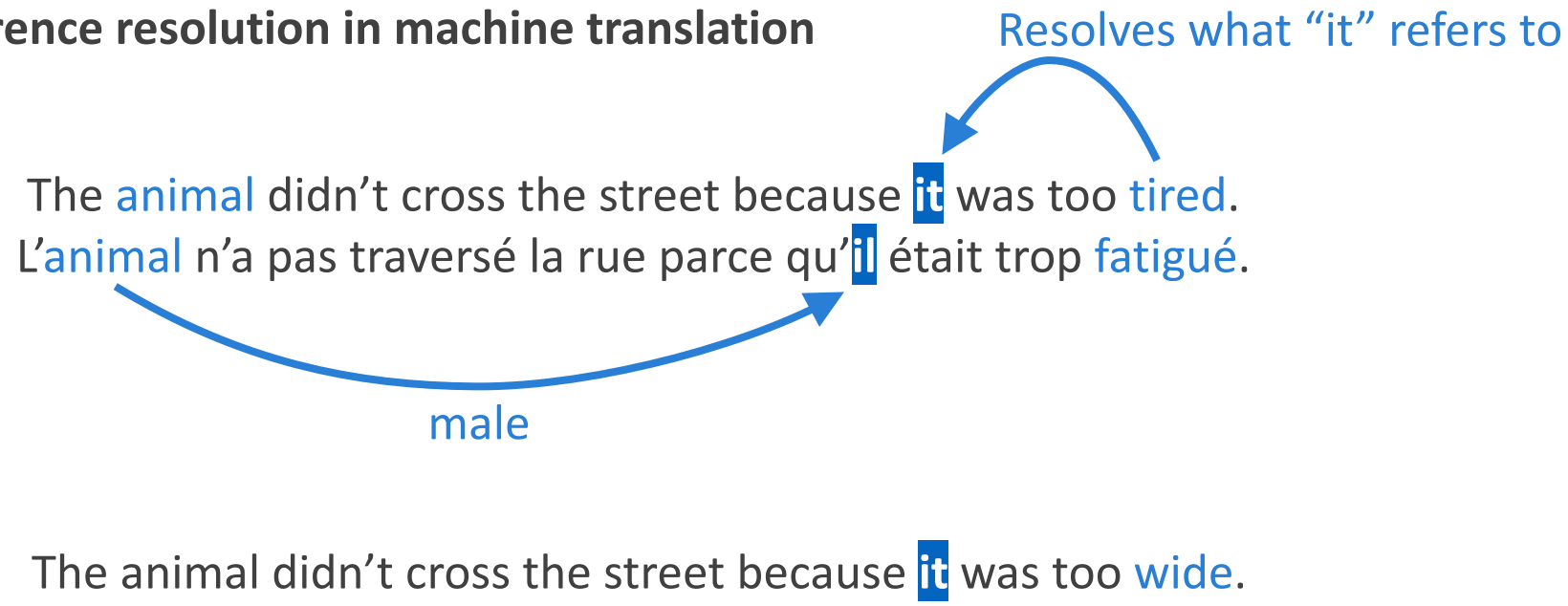
Example: coreference resolution in machine translation



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

Why do we need attention?

Example: coreference resolution in machine translation

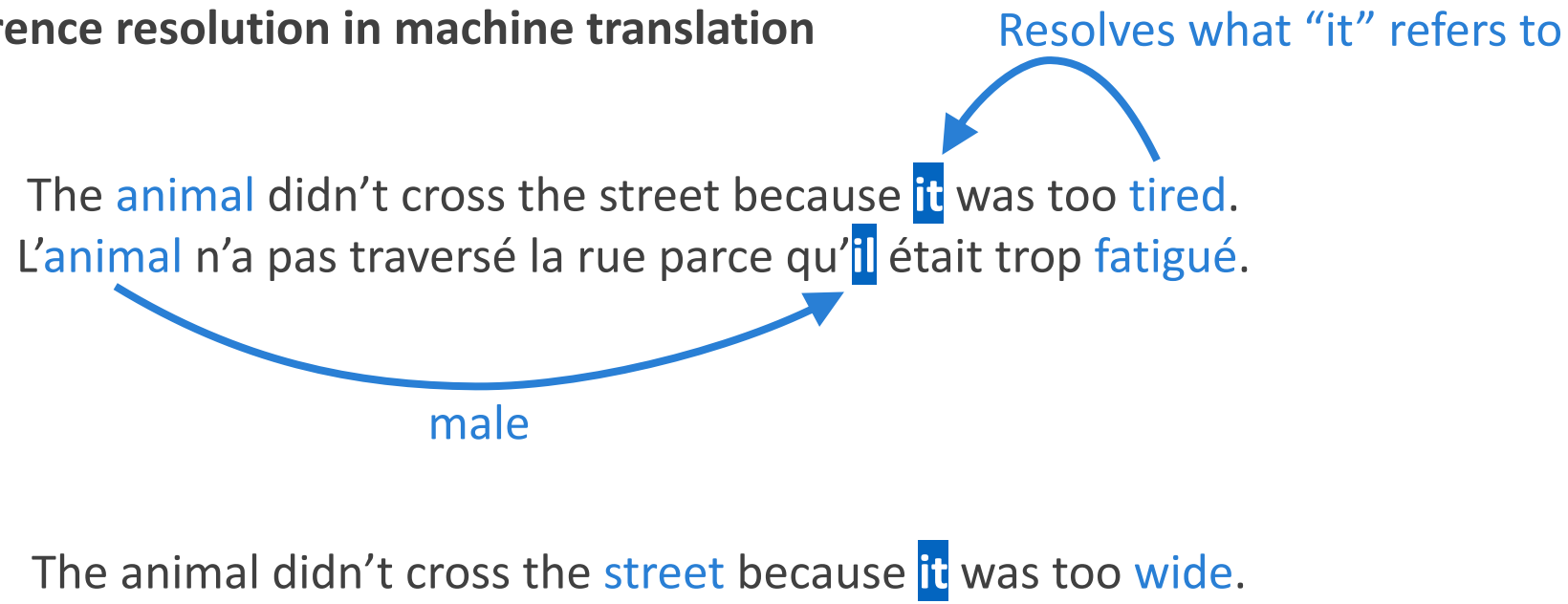


The animal didn’t cross the street because it was too wide.

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

Why do we need attention?

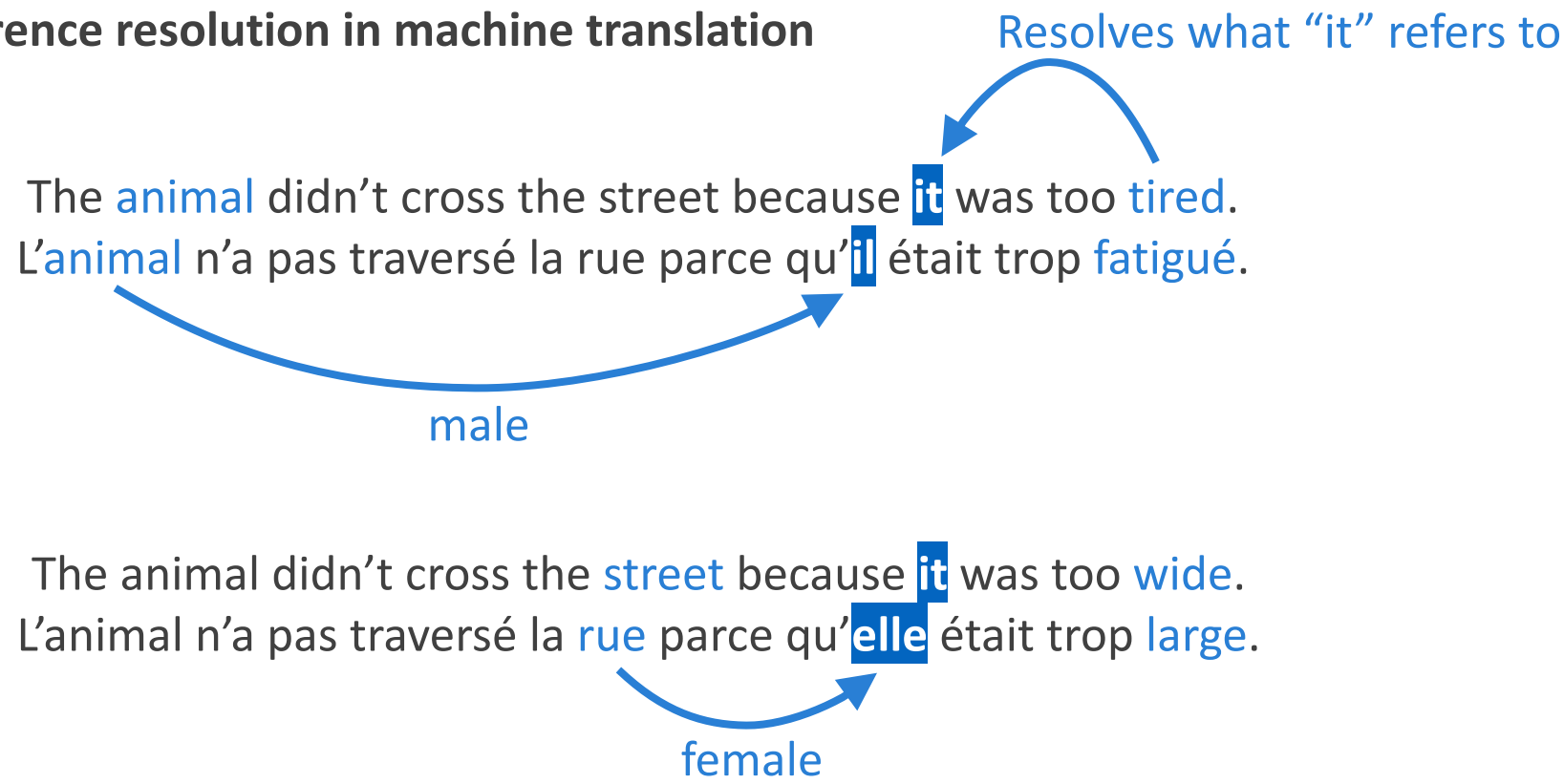
Example: coreference resolution in machine translation



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

Why do we need attention?

Example: coreference resolution in machine translation

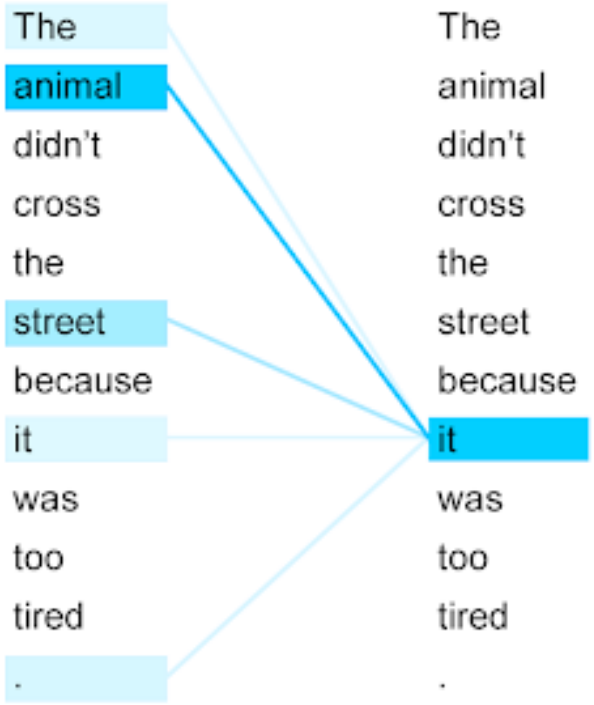


Source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

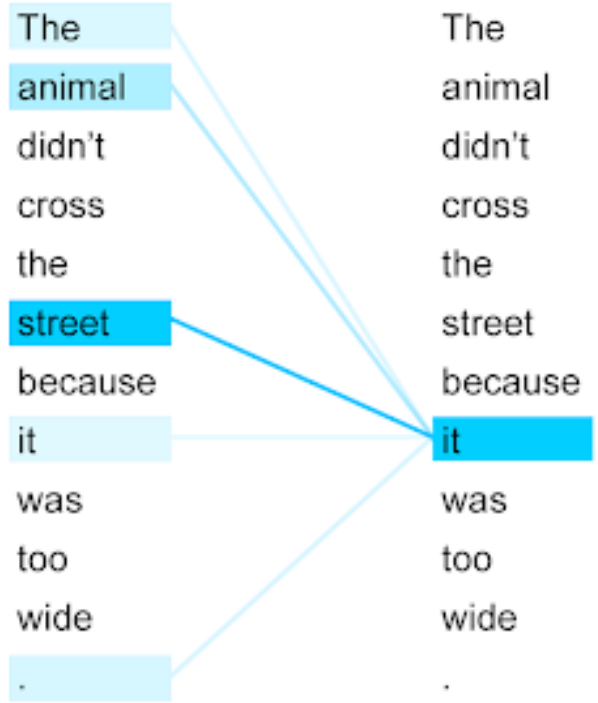
Problem when it depends on the context
-> we have to carry information for a long distance
-> we have to share the information in both directions

Transformer encoder attends to correct reference

The animal didn't cross the street because it was too tired.
L'animal n'a pas traversé la rue parce qu'il était trop fatigue.



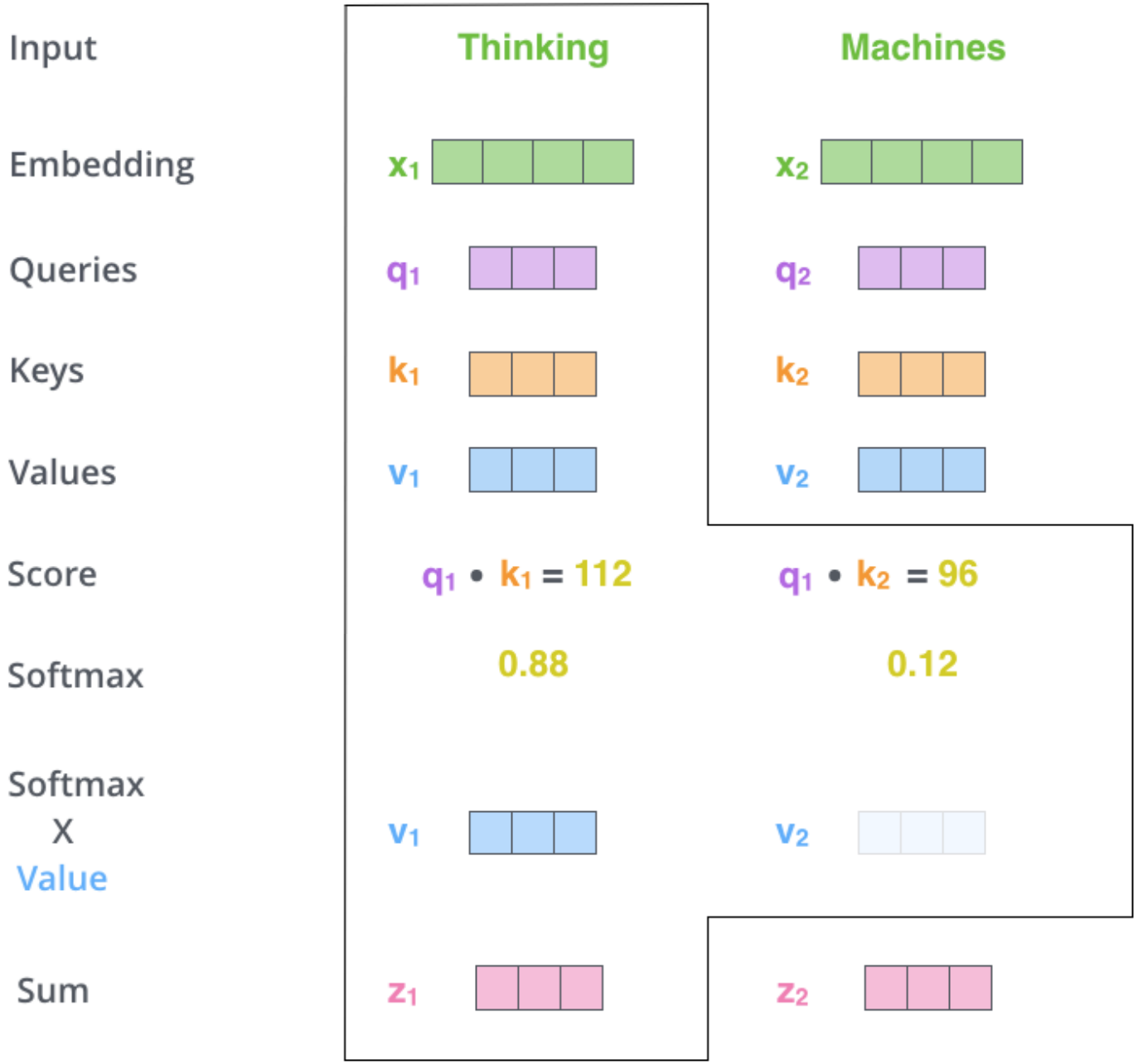
The animal didn't cross the street because it was too wide.
L'animal n'a pas traversé la rue parce qu'elle était trop large.



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

Self Attention

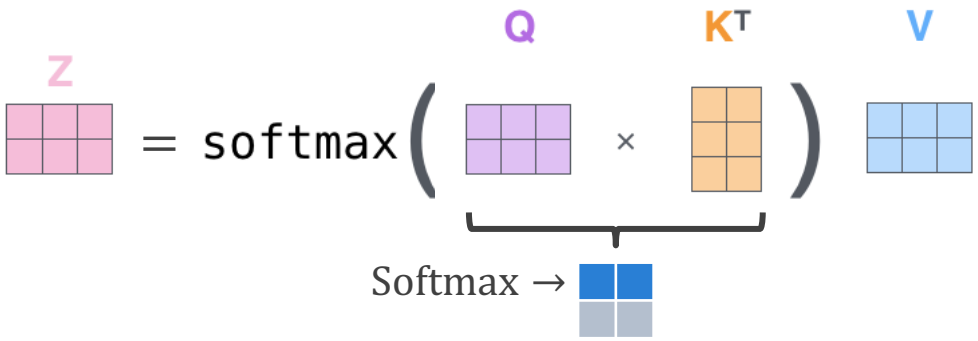
$$\boldsymbol{Z} = \text{Softmax}\left(\boldsymbol{QK^T}\right)\boldsymbol{V}$$

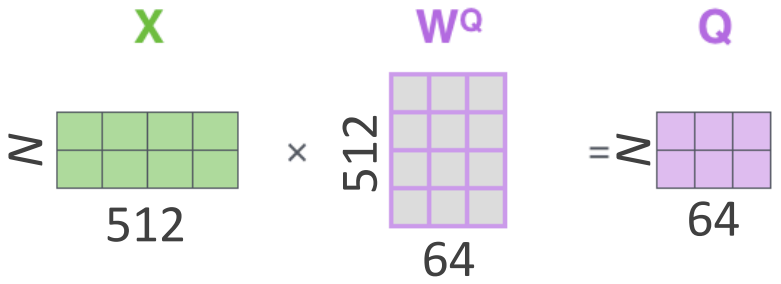


https://jalammar.github.io/illustrated-transformer/

Self Attention

$$\boldsymbol{Z} = \text{Softmax}\left(\boldsymbol{QK^T}\right)\boldsymbol{V}$$

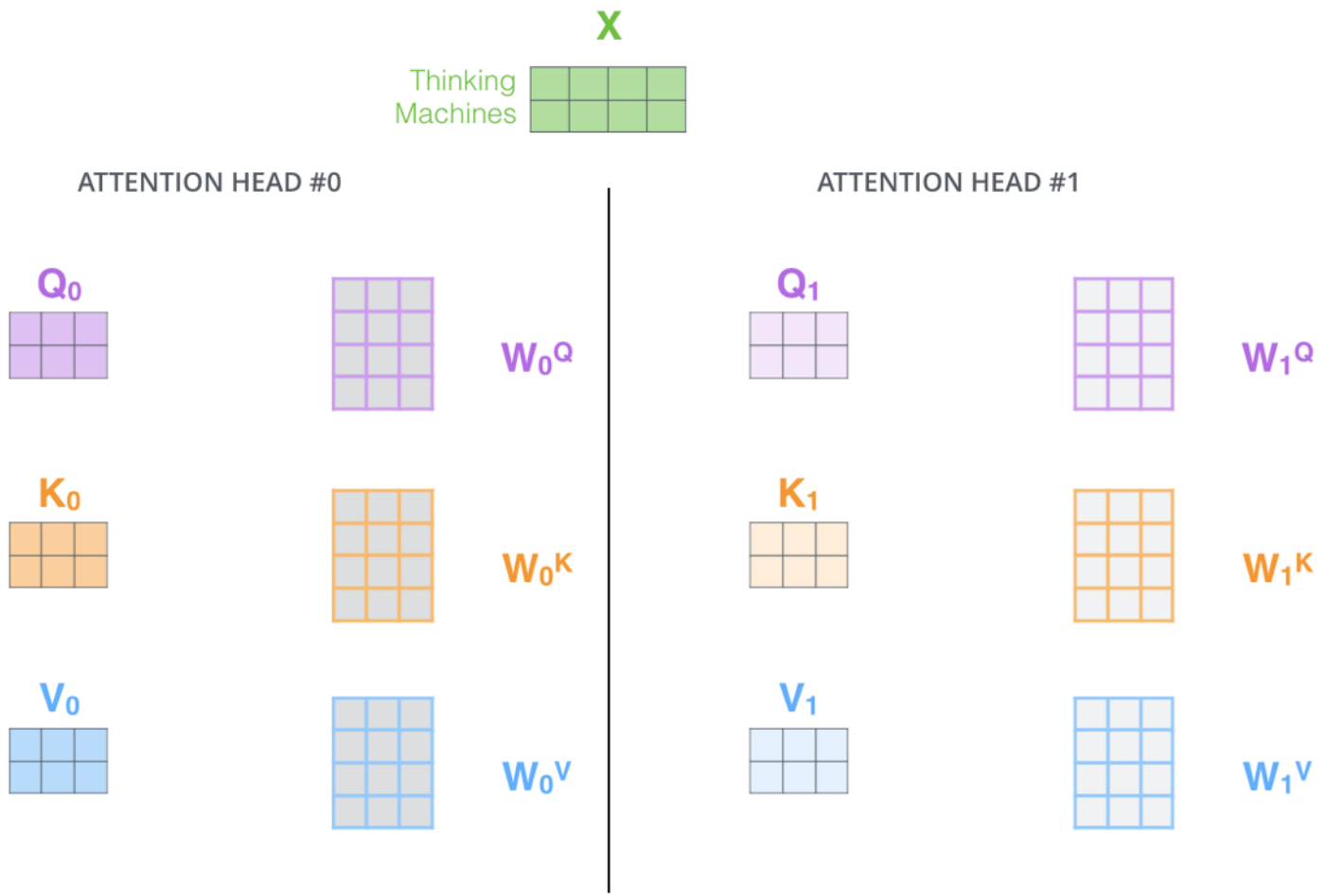




X - input matrix
w^Q - query weight matrix
...
-> trying to learning weights

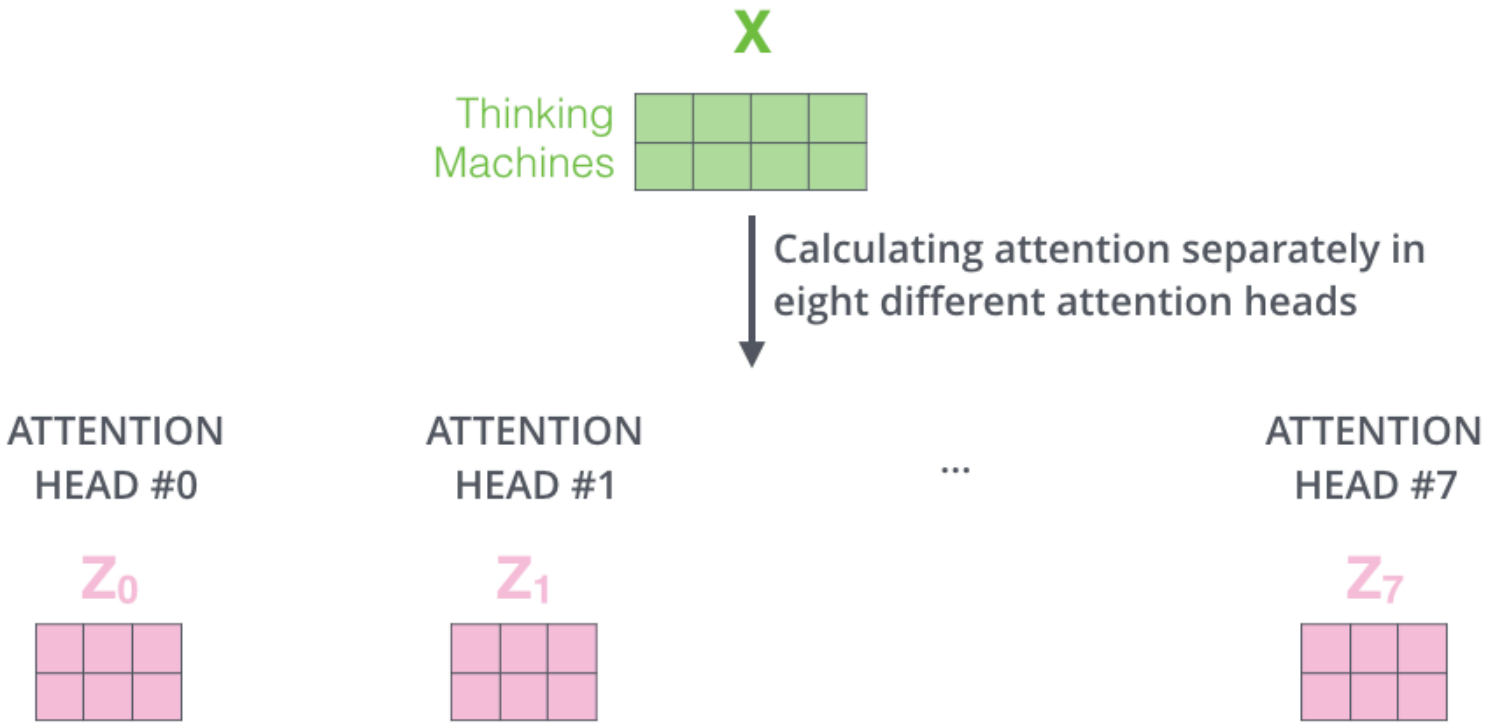
https://jalammar.github.io/illustrated-transformer/

Self Attention: multiple attention “heads”



https://jalammar.github.io/illustrated-transformer/

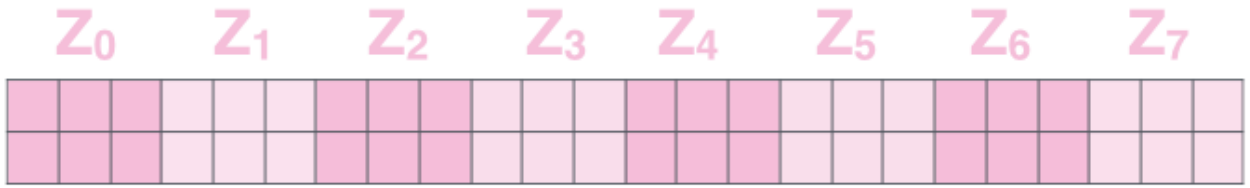
Self Attention: multiple attention “heads”



https://jalammar.github.io/illustrated-transformer/

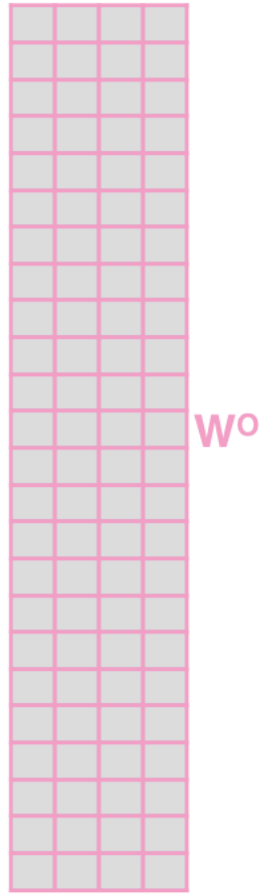
Combining outputs of all attention heads

1) Concatenate all the attention heads



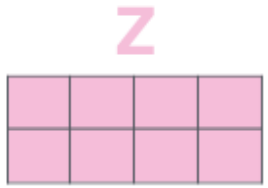
2) Multiply with a weight matrix W^O that was trained jointly with the model

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

=



https://jalammar.github.io/illustrated-transformer/

Self Attention: summary

1) This is our input sentence*

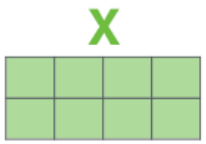
2) We embed each word*

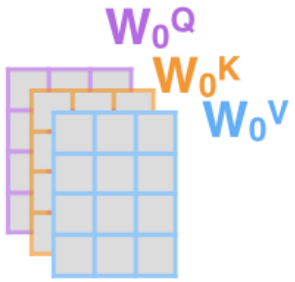
3) Split into 8 heads. We multiply X or R with weight matrices

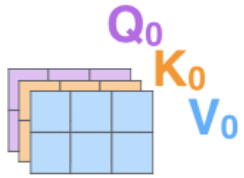
4) Calculate attention using the resulting Q/K/V matrices

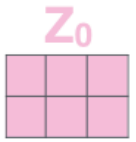
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

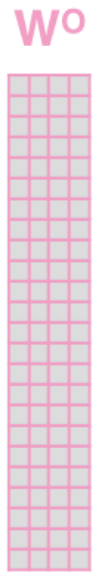
Thinking Machines

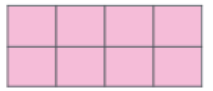


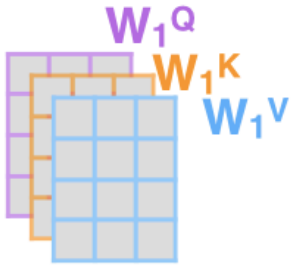


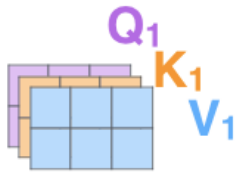


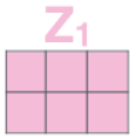




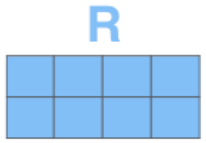


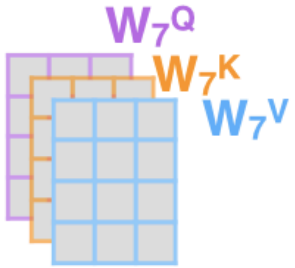


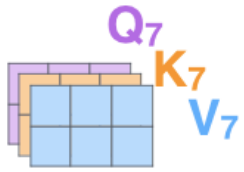


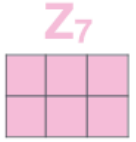


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one







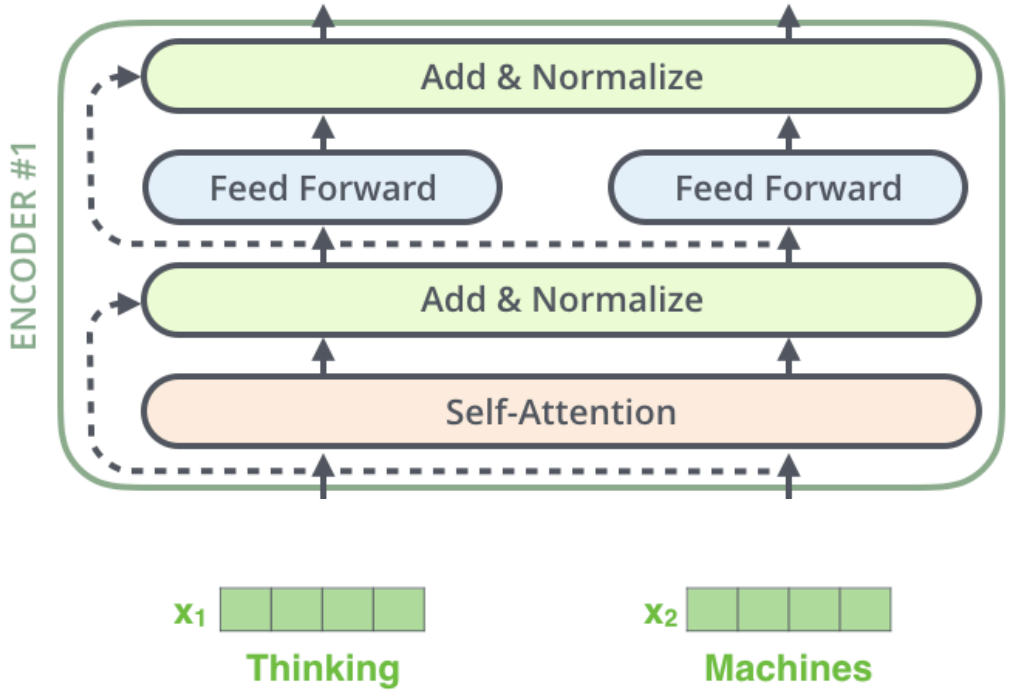


https://jalammar.github.io/illustrated-transformer/

siehe dazu auch [F36f]

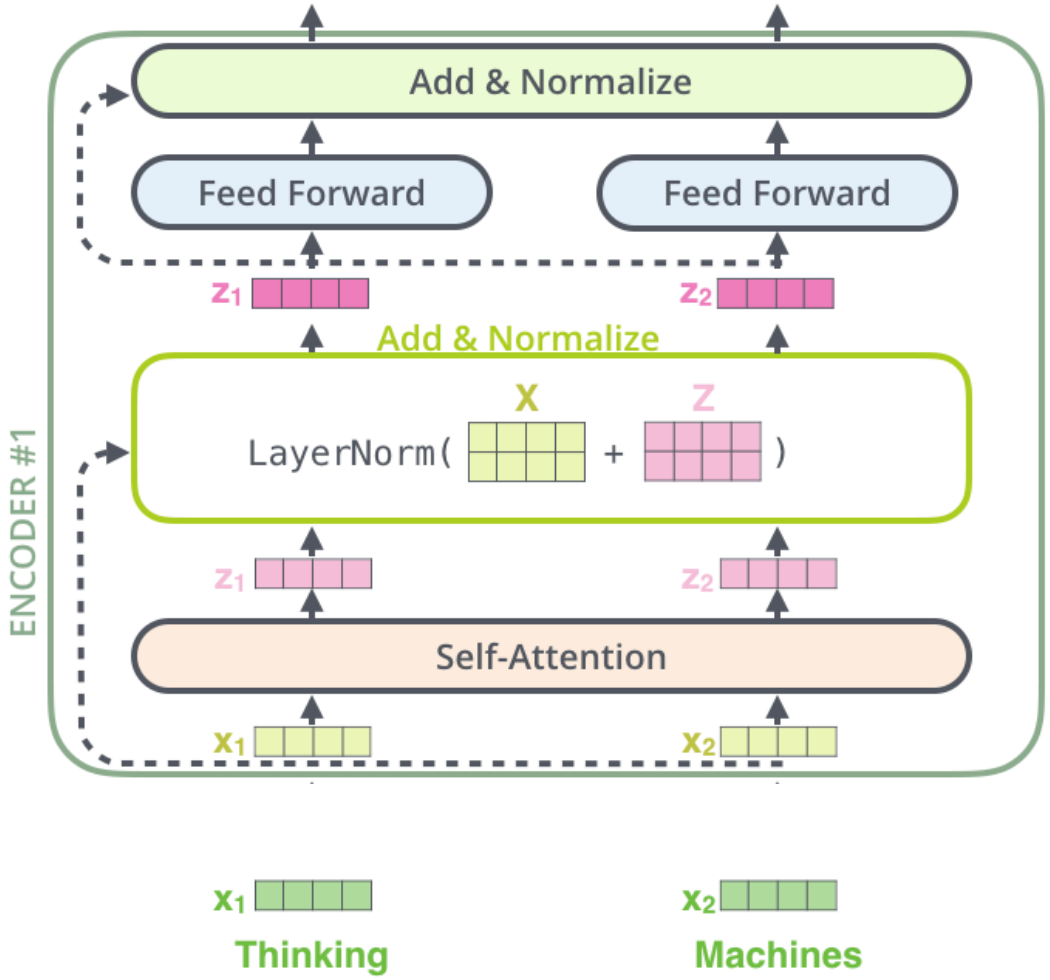
Encoder: residual connections + layer normalization

skip connection from the input to Add&Normalize part



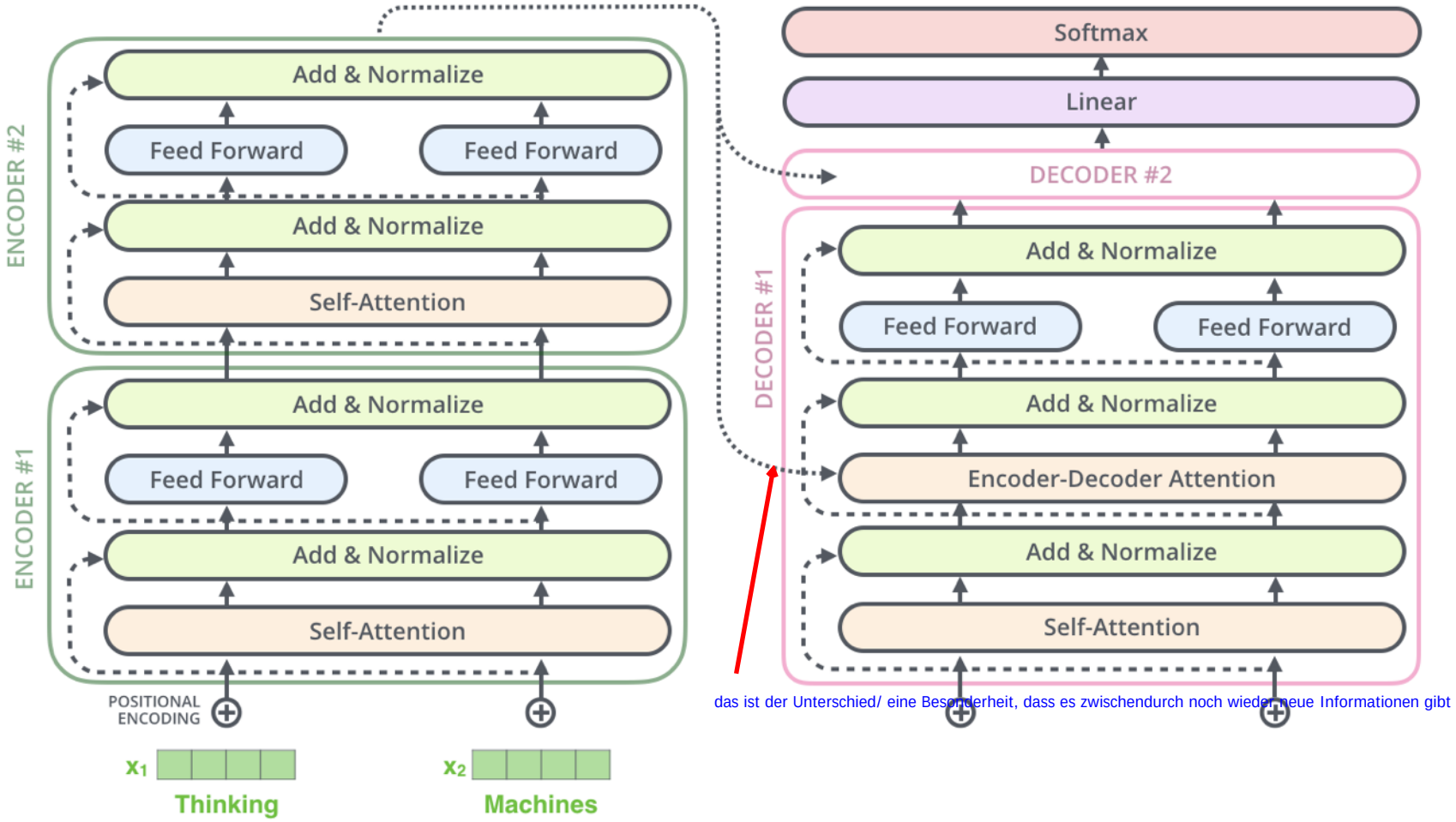
https://jalammar.github.io/illustrated-transformer/

Encoder: residual connections + layer normalization



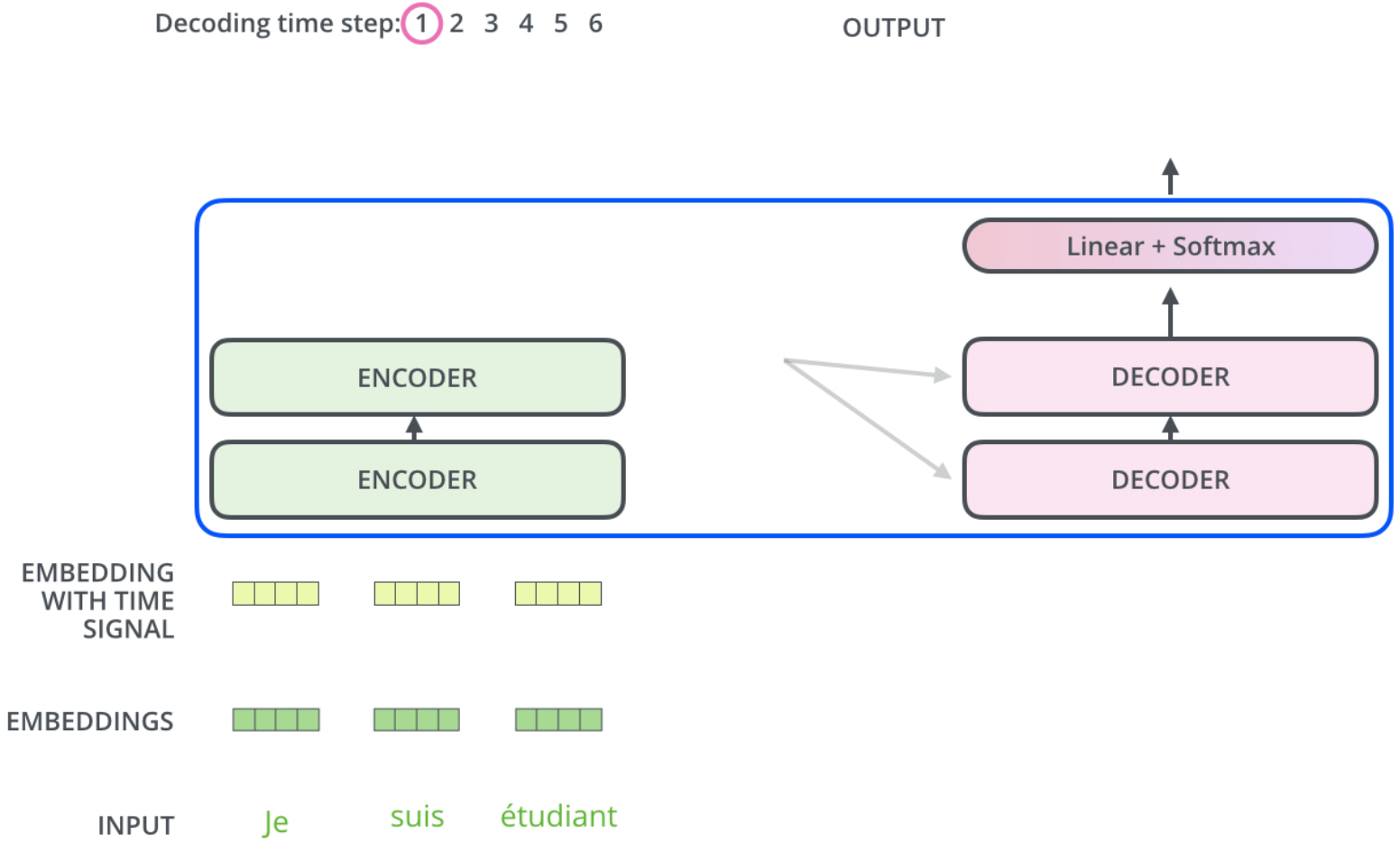
https://jalammar.github.io/illustrated-transformer/

Transformer: Decoder



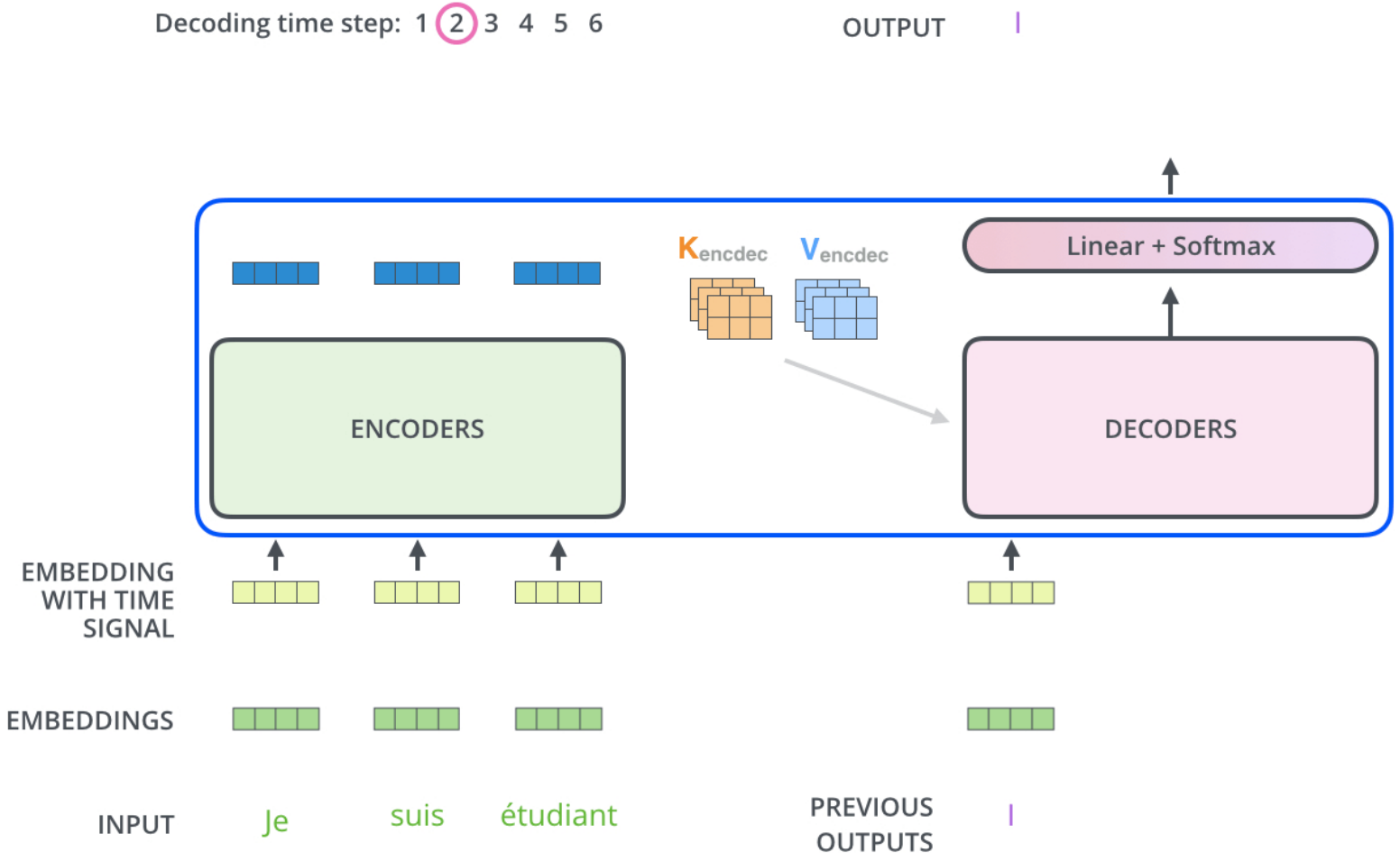
https://jalammar.github.io/illustrated-transformer/

Transformer: Decoder



https://jalammar.github.io/illustrated-transformer/

Transformer: Decoder



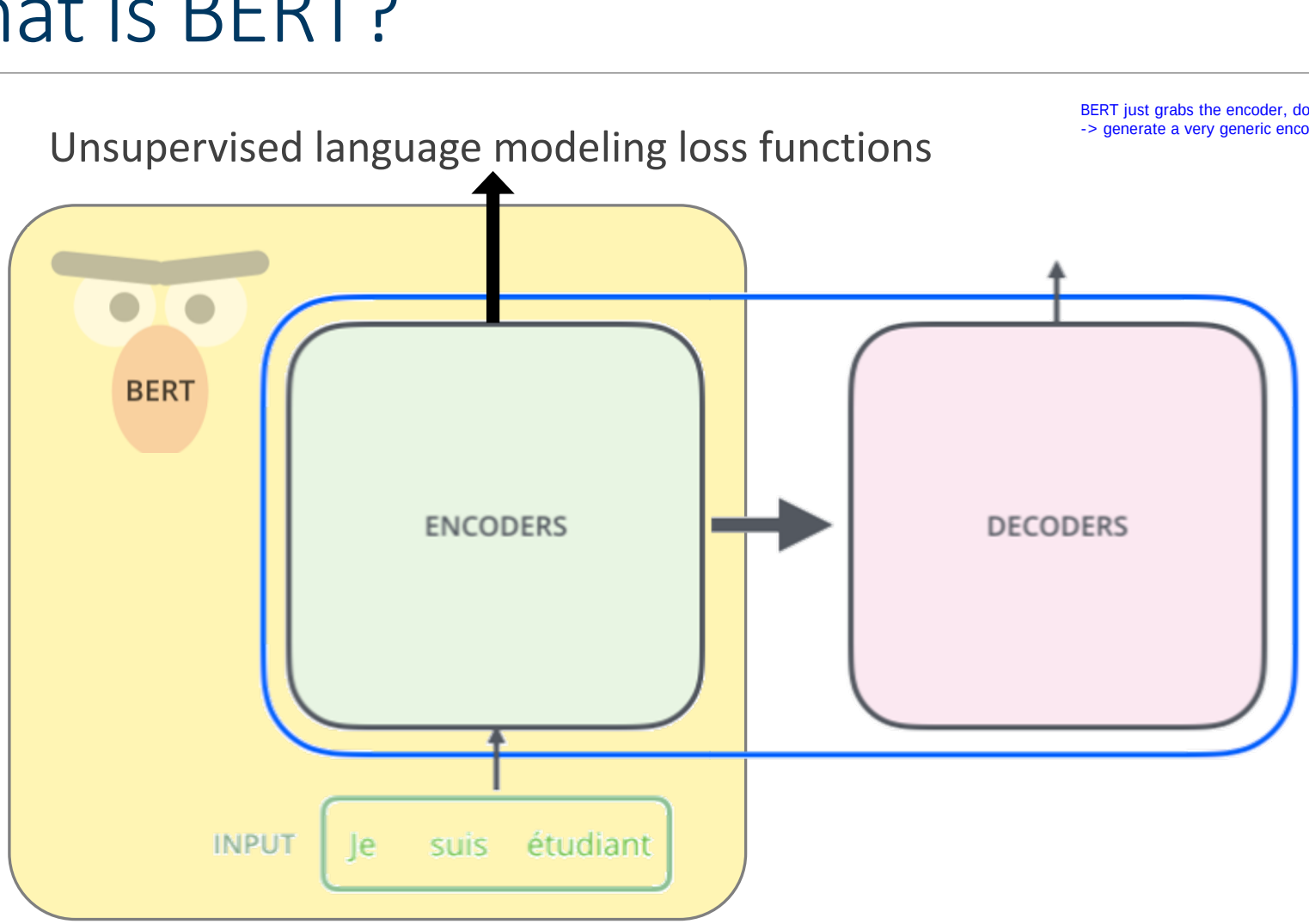
https://jalammar.github.io/illustrated-transformer/

BERT

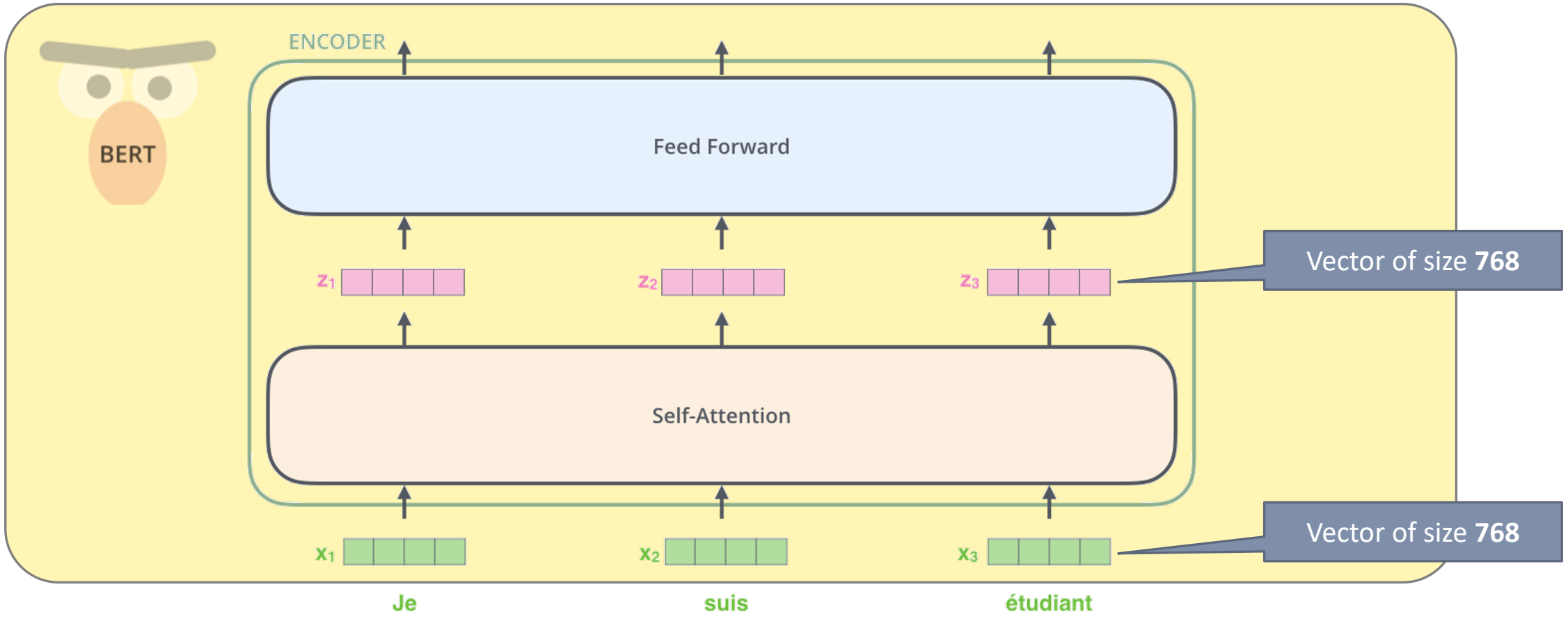
BIDIRECTIONAL **E**NCODER **R**EPRESENTATIONS FROM **T**RANSFORMERS

What is BERT?

BERT just grabs the encoder, do nothing with the decoder
-> generate a very generic encoder!



BERT: Transformer encoder with 768D embedding



https://jalammar.github.io/illustrated-transformer/

BERT: input tokenization

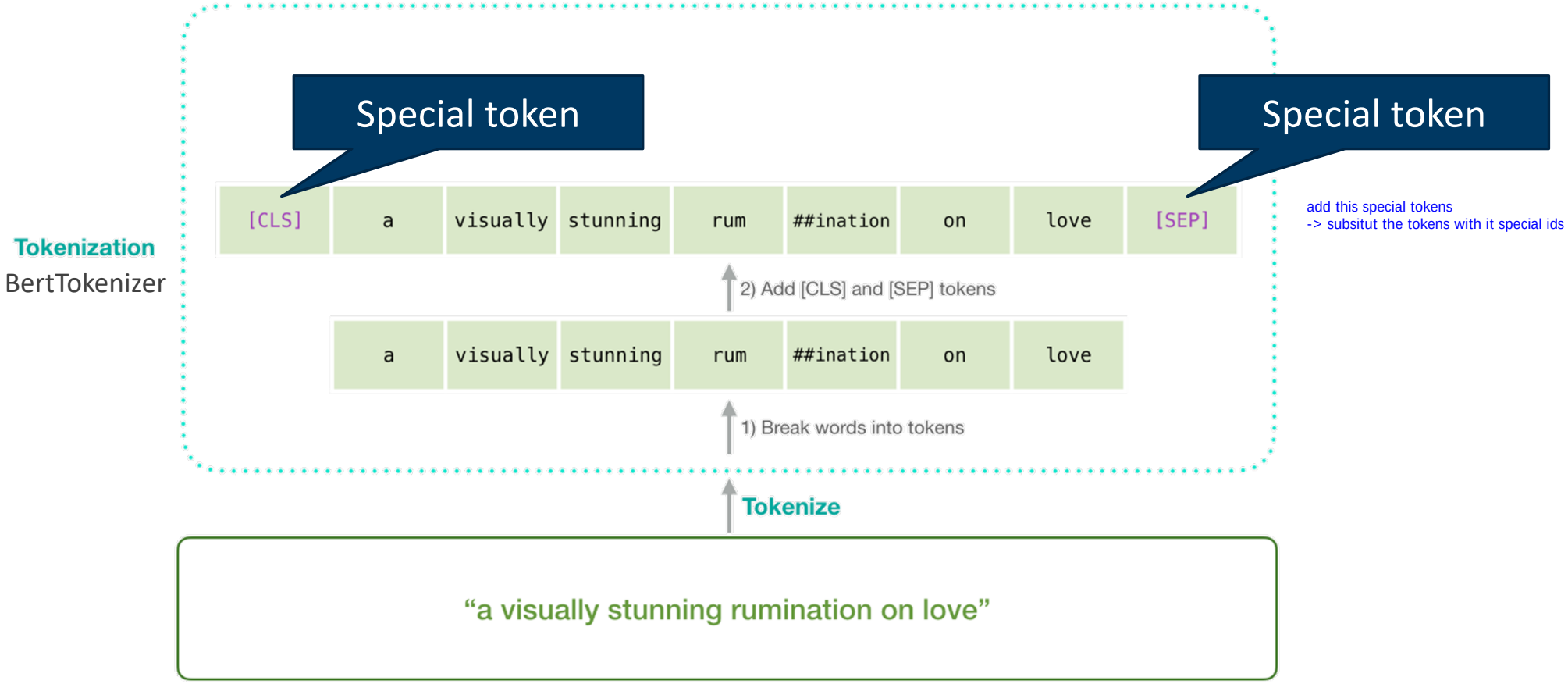


Image source: http://jalammar.github.io/illustrated-bert/

BERT: input tokenization

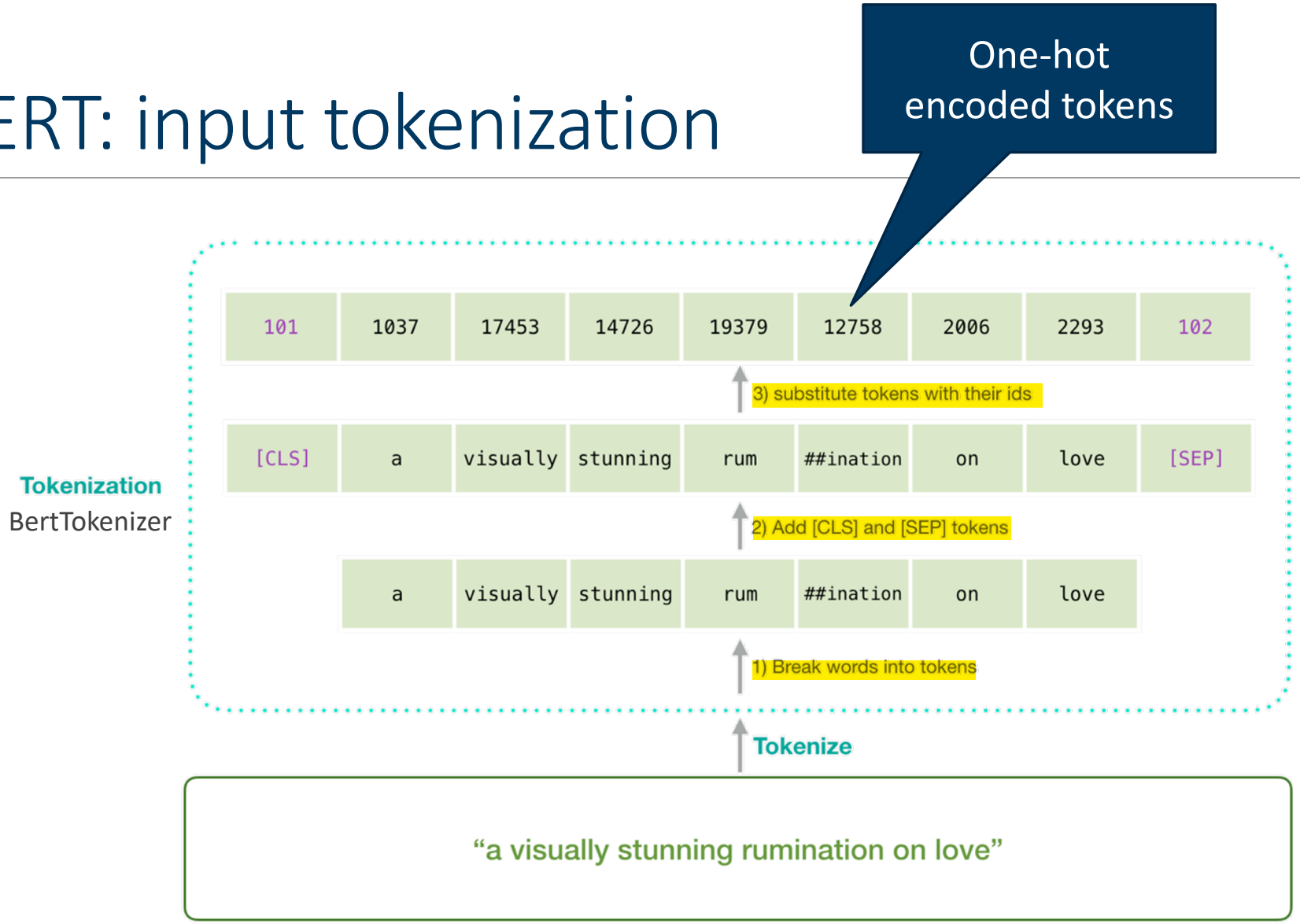


Image source: http://jalammar.github.io/illustrated-bert/

BERT training

Training optimizes two objectives:

1. **Predict masked words** (a.k.a. “cloze task”)

   Let’s stick to _____ in this skit.

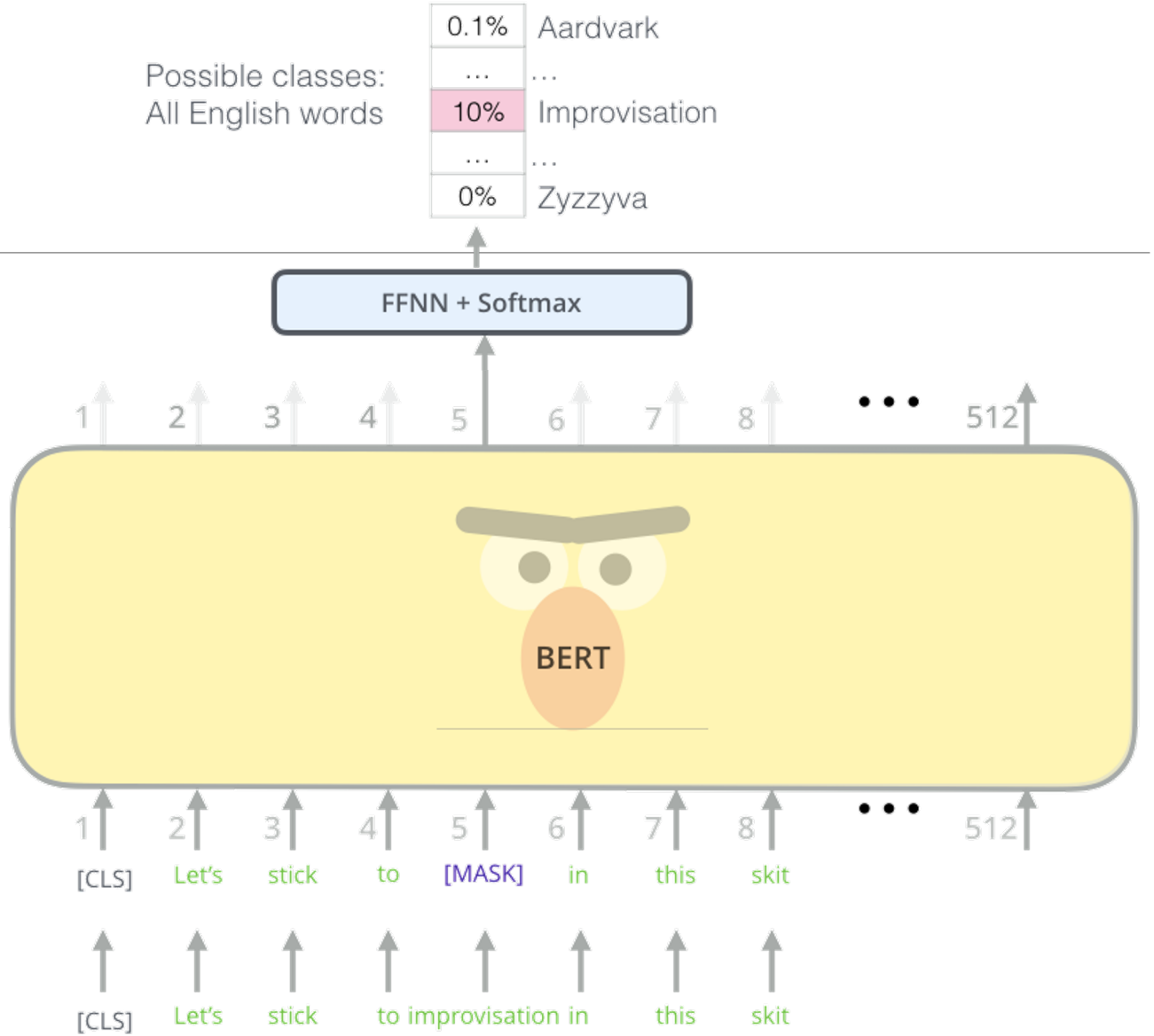
2. **Classify next sentence**

   The man went to the store. → Penguin are flightless birds.

   → **He bought a bottle of whiskey.**

BERT training

1 **Predict masked words**
(a.k.a. “cloze task”)



Randomly mask 15% of tokens

Input

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

BERT training

2 **Classify next sentence**

This first layer has to summerise all the information about the two sentences to decide if prob. the second follows on the first one.

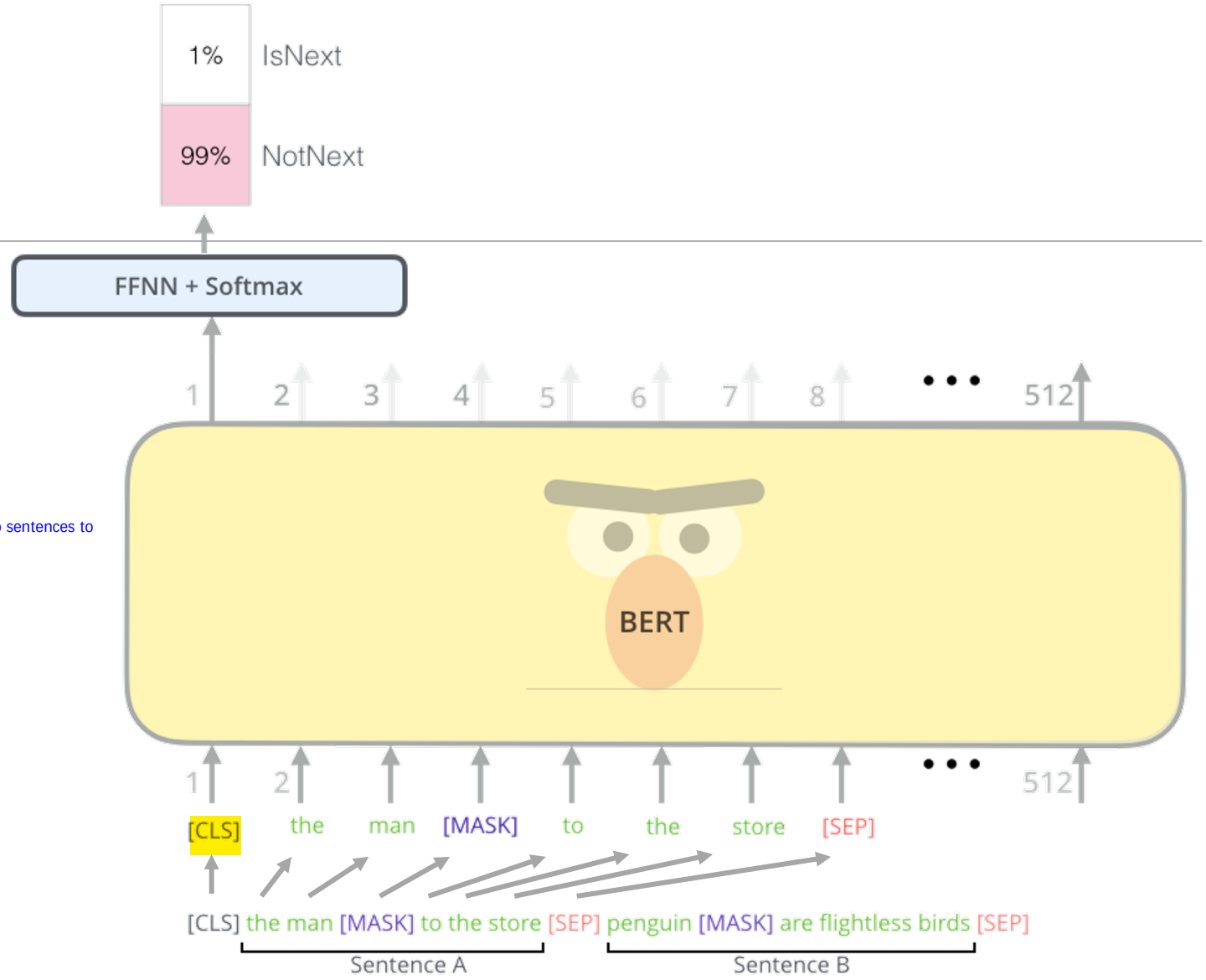
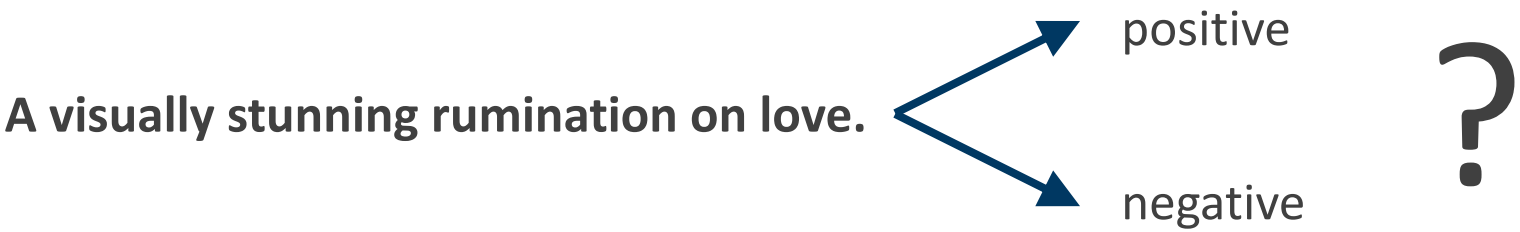


Image source: http://jalammar.github.io/illustrated-bert/

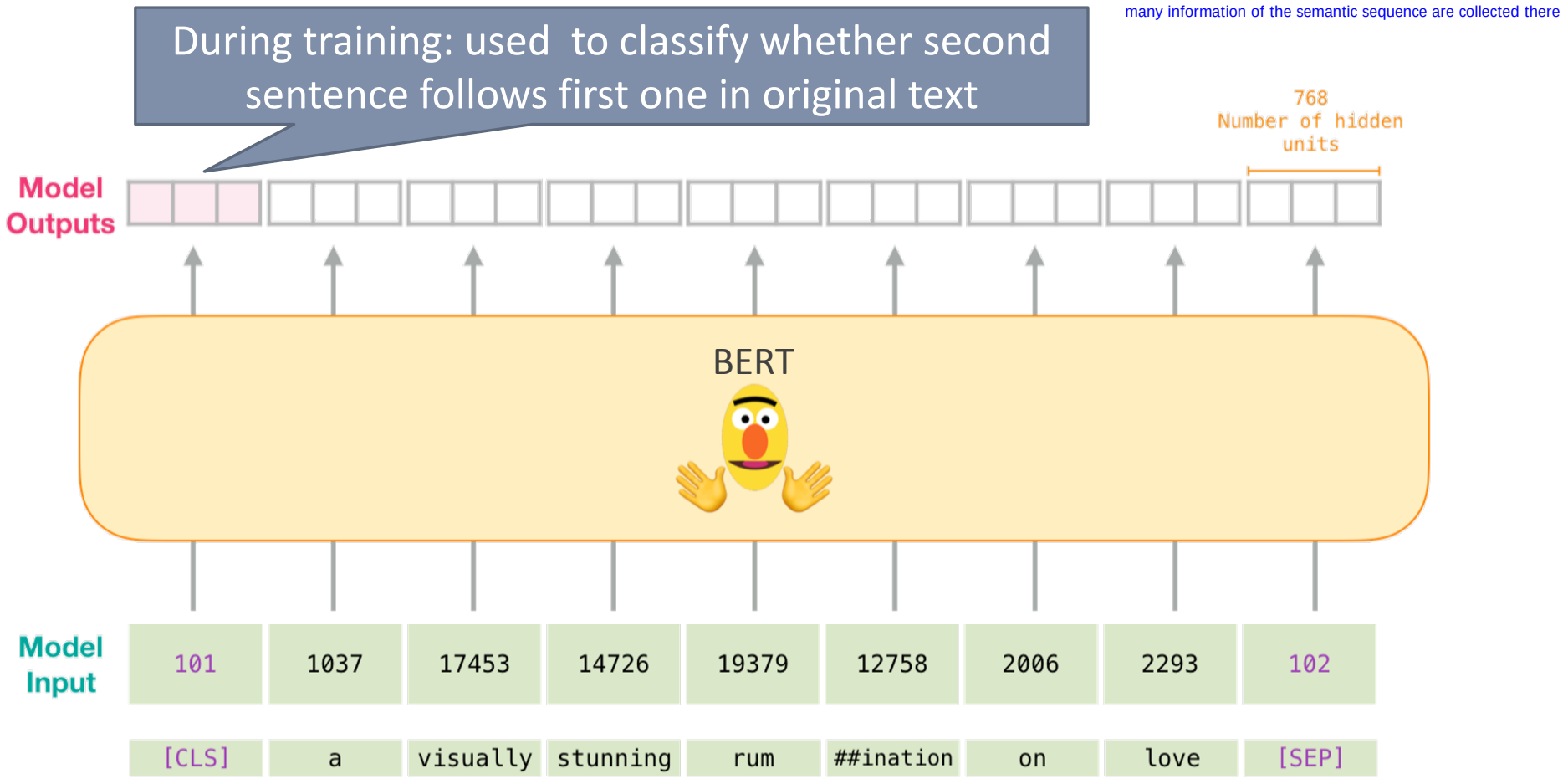
BERT for transfer learning

EXAMPLE: SENTIMENT CLASSIFICATION

Example: sentiment classification

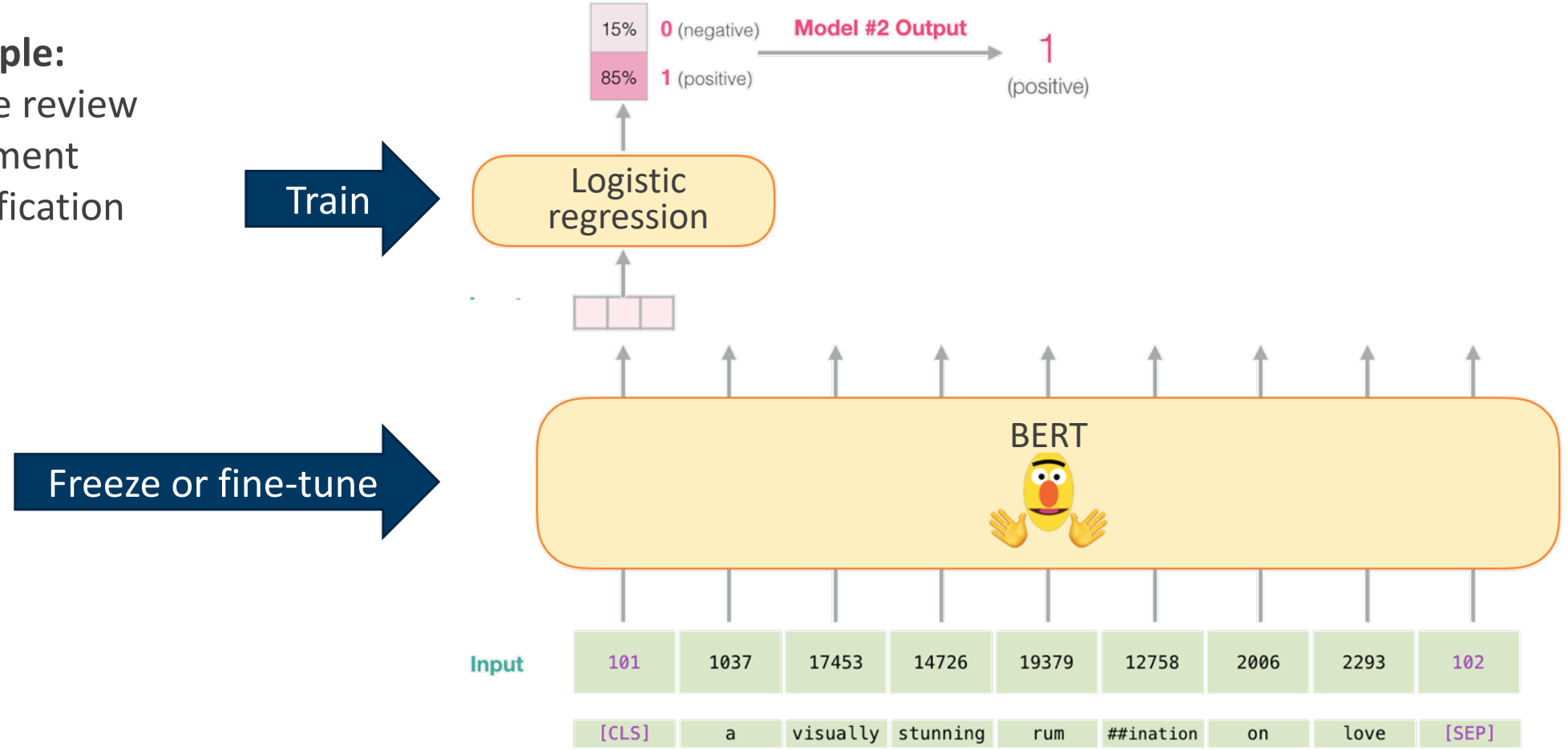


BERT for transfer learning



BERT for transfer learning

Example:
Movie review sentiment classification



Questions!