SkillFactory

Sequence to Sequence task Attention

Sidorov Nikita

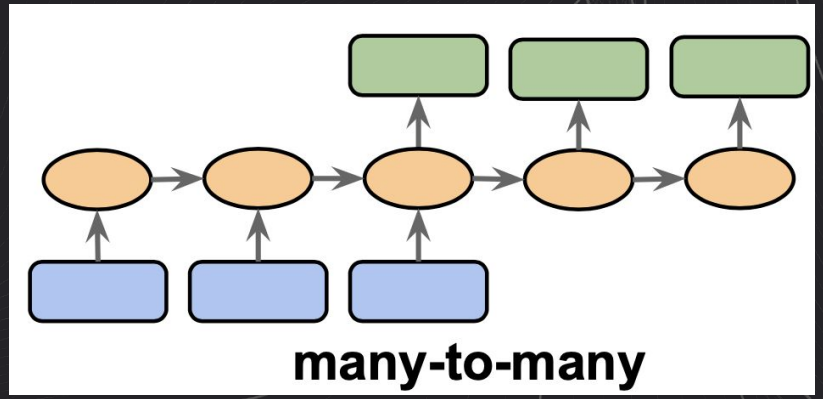
MLE (NLP) Sber

What we will learn today

- Sequence to sequence task;
- applications of seq2seq;
- training seq2seq model
- inference approaches for seq2seq;
- evaluate seq2seq model quality;
- attention mechanism;
- self-attention mechanism;

Sequence to Sequence task

Traditionally seq2seq task is associated with translation from one language to another. It doesn't have to be human languages



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Seq2Seq applications

- Traditionally seq2seq task is associated with translation from one language to another.
- we can “translate” any sequence from source task into target sequence task.

We will discuss this task in terms of traditional human native translation from one language to another.

Statistical machine translation

We want to find best English sentence *y,* for given Russian sentence *x.*

$$\text{argmax}_y P(y|x)$$

It’s all like human deal with this task.

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

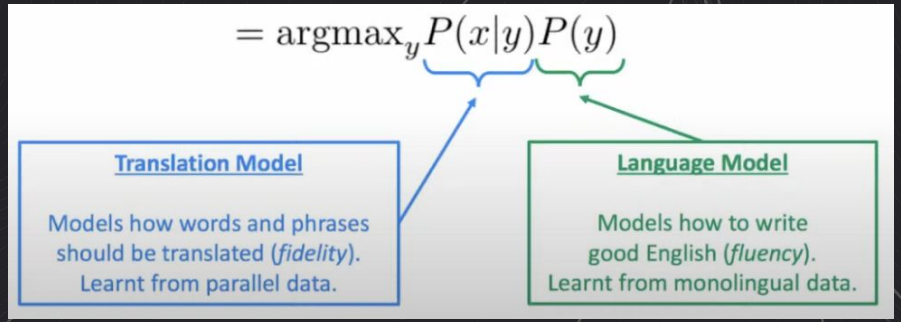
Statistical machine translation

We want to find best English sentence *y,* for given Russian sentence *x.*

$$\text{argmax}_y P(y|x)$$

It’s all like human deal with this task.

If apply Bayes Rule we will break this down into two components:



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Problems of Statistical MT

- systems had many separately-designed components
- lots of feature engineering
- need to design features to capture particular language phenomena
- lots of human effort to maintain
- reported effort for every language pair

Comparison of different MT approaches

Human deal with this task like this

$\text{argmax}_y P(y|x)$

Computer deal with this task like this

$\text{argmax}_y P(y|x, \theta)$

where p - some neural model,

and θ - parameters of model

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Neural approach for MT

Computer deal with this task like this

$$\text{argmax}_y P(y|x, \theta)$$

where p - some neural model, and θ - parameters of model

Questions we need to ask?

modeling
How does the model for p(y| x, θ) look like?

learning
How to find best θ?

search
How to find the argmax?

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

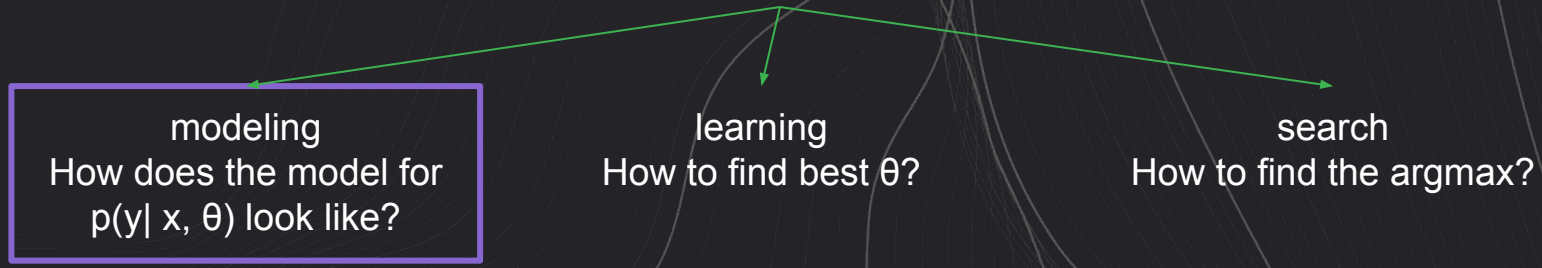
Neural approach for MT

Computer deal with this task like this

$$\text{argmax}_y P(y|x, \theta)$$

where p - some neural model, and θ - parameters of model

Questions we need to ask?

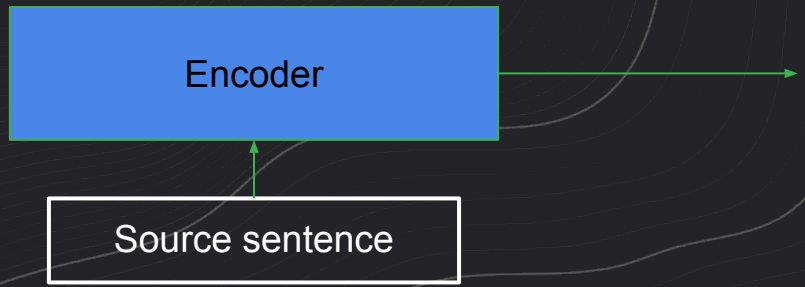


Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Encoder-Decoder Framework

The standard modeling paradigm:

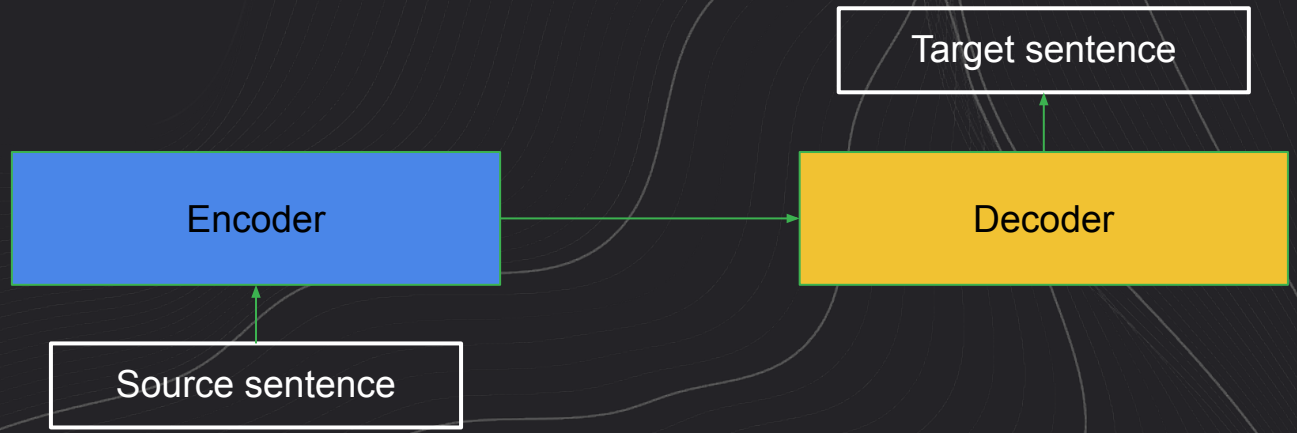
- Encoder - reads the source sentence and builds its representation



Encoder-Decoder Framework

The standard modeling paradigm:

- Encoder - reads the source sentence and builds its representation
- Decoder - uses source representation from the encoder to generate the target sequence



Conditional Language Modeling

Language models:

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

Conditional Language Modeling

Language models:

$$P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$$

Conditional Language models:

$$P(y_1, y_2, \ldots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$$

condition on source x

Neural Machine Translation: how it works

In fact NMT calculates probability P(y|x) - where y - target sentence, and x - source sentence

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

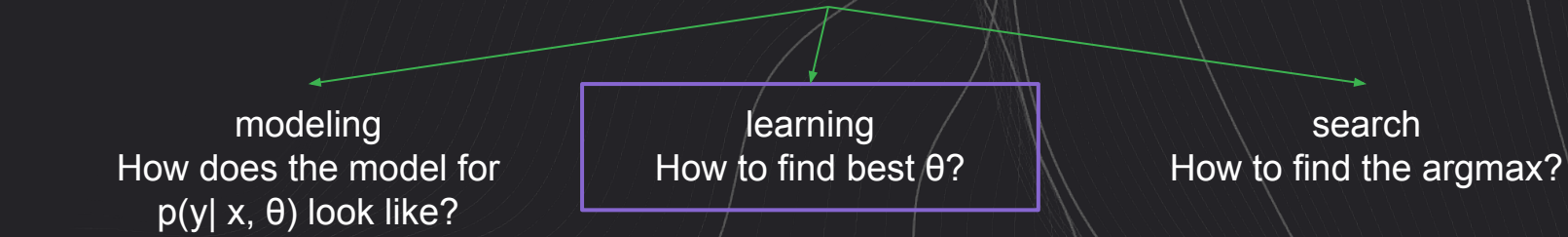
Neural approach for MT

Computer deal with this task like this

$\text{argmax}_y P(y|x, \theta)$

where p - some neural model, and θ - parameters of model

Questions we need to ask?



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

NMT: training

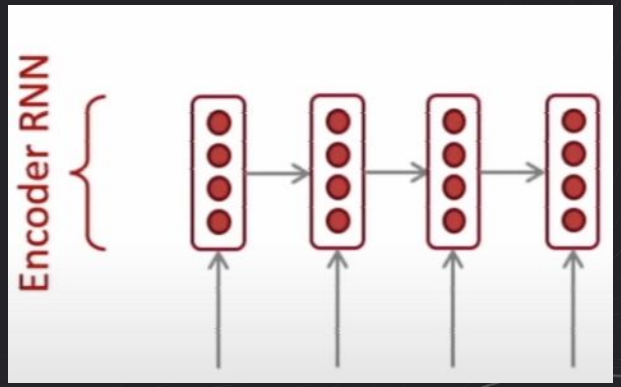
il a m` entarté

Source sentence (from corpus)

he hit me with a pie

Target sentence (from corpus)

NMT: training



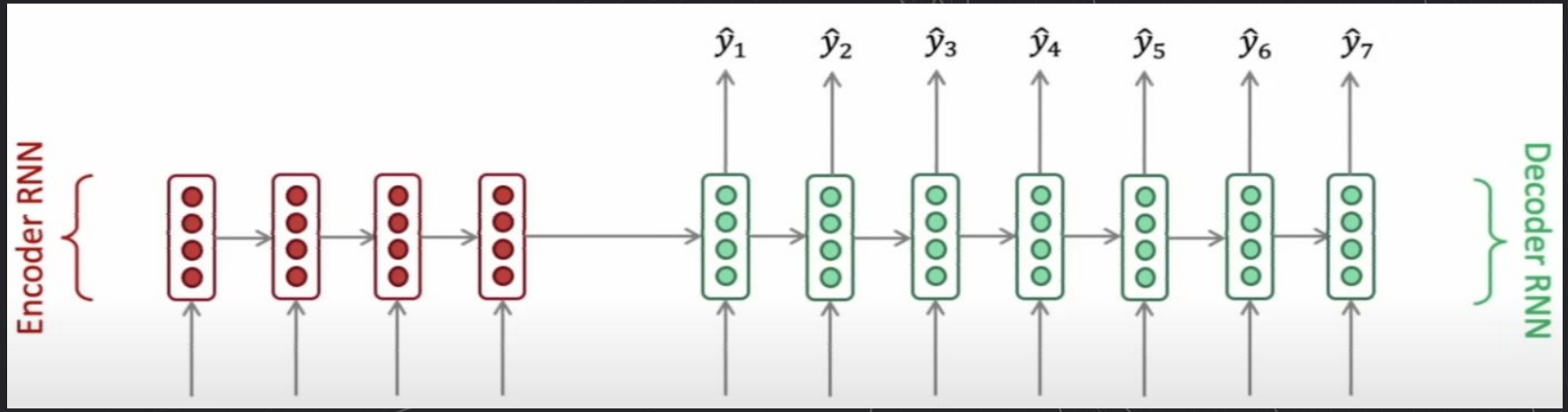
il a m` entarté

Source sentence (from corpus)

<START> he hit me with a pie

Target sentence (from corpus)

NMT: training



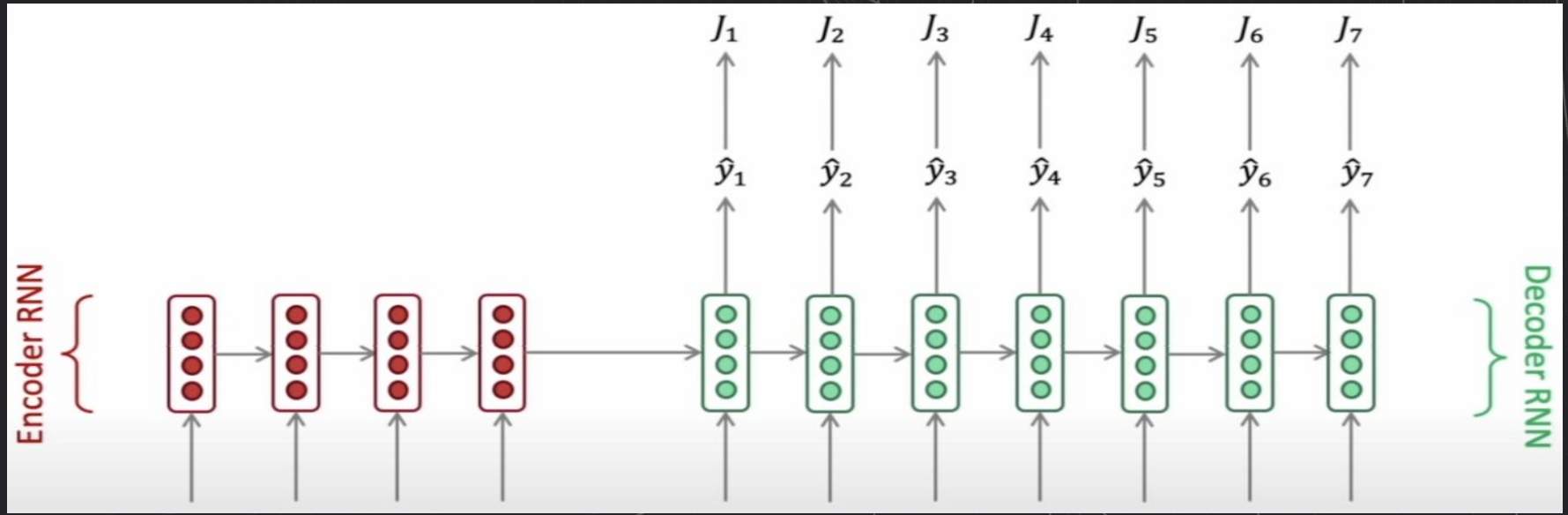
il a m` entarté

Source sentence (from corpus)

<start> he hit me with a pie

Target sentence (from corpus)

NMT: training



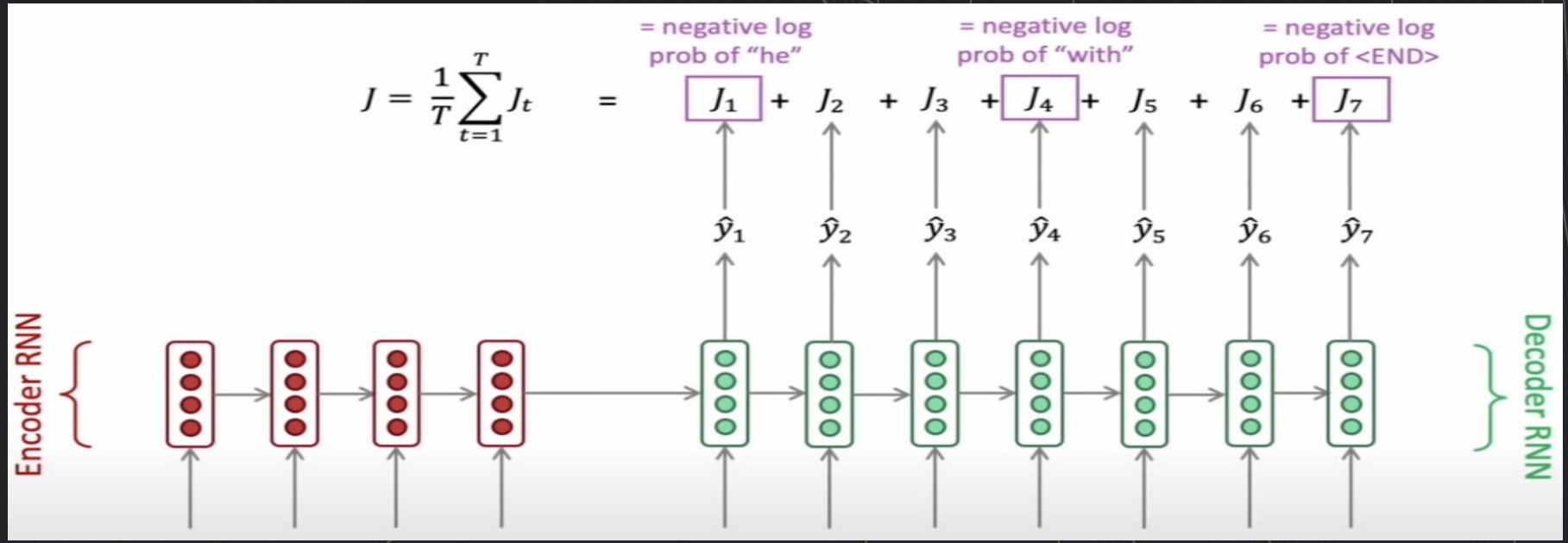
il a m` entarté

Source sentence (from corpus)

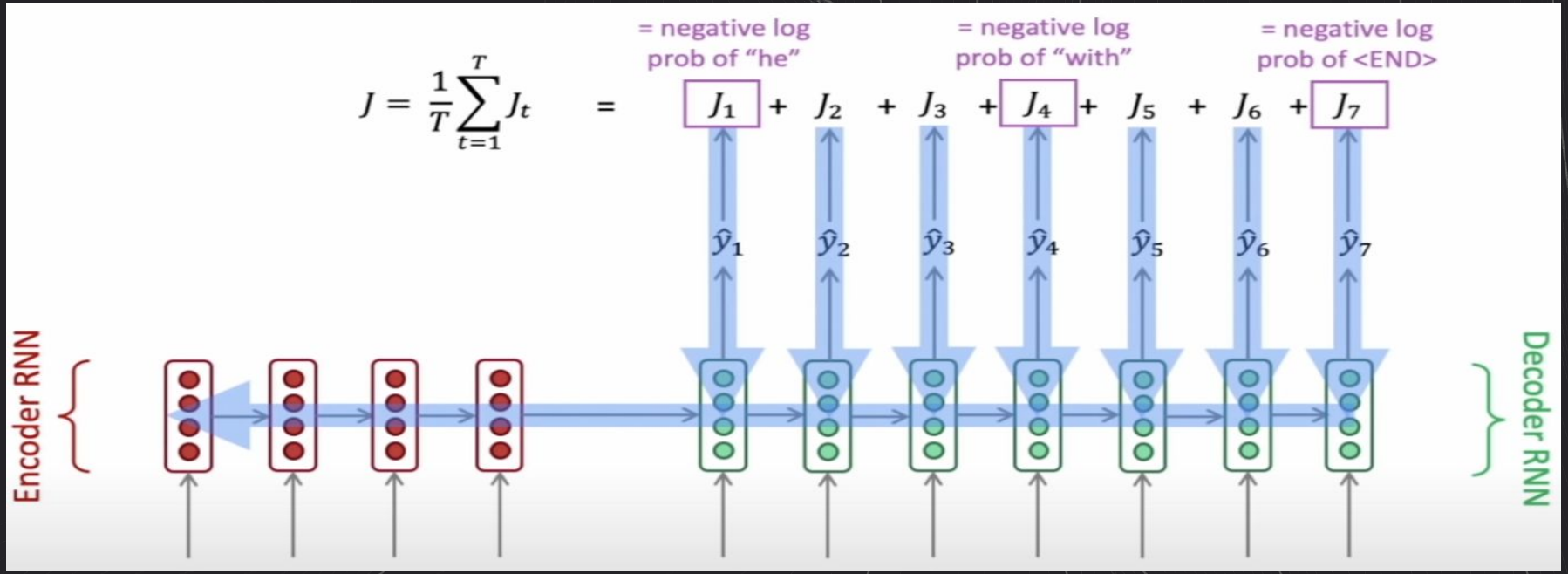
<start> he hit me with a pie

Target sentence (from corpus)

NMT: training



NMT: training



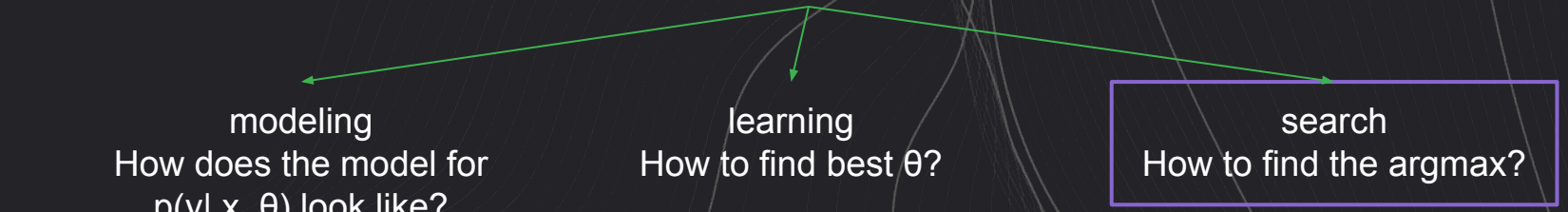
Neural approach for MT

Computer deal with this task like this

$$\text{argmax}_y P(y|x, \theta)$$

where p - some neural model, and θ - parameters of model

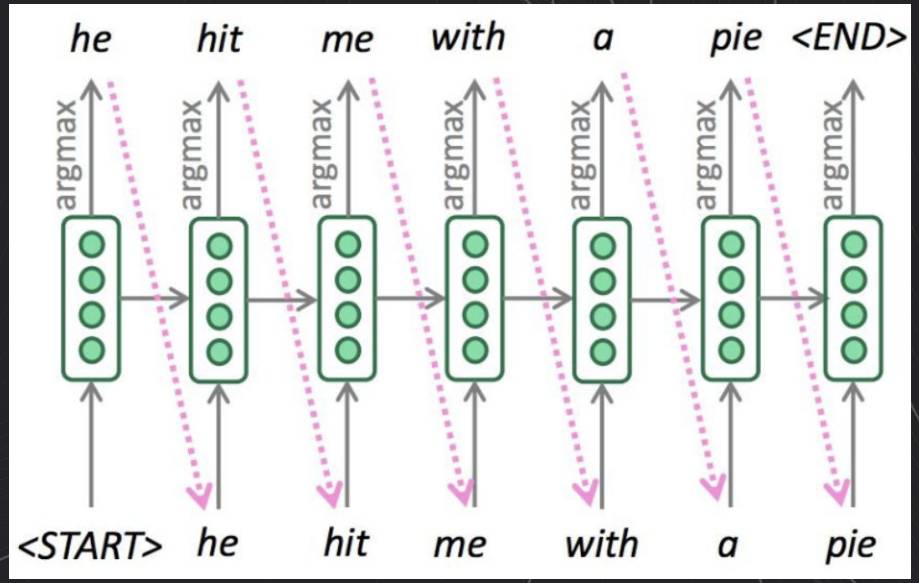
Questions we need to ask?



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Generating translation

We saw how to generate (or “decode”) the target sentence by taking the argmax on each step of the decoder.



This is greedy decoding (take the most probable word on each step).

Greedy decoding

There is no way to undo decisions in greedy decoding.

Input: il a m`entarté (he hit me with a pie)

-> he ____

-> he hit ____

-> he hit a _____ (no way to go back ...)

$$\arg\max_{y} \prod_{t=1}^{n} p(y_t|y_{<t}, x) \neq \prod_{t=1}^{n} \arg\max_{y_t} p(y_t|y_{<t}, x)$$

Exhaustive search decoding

We want to find a (length T) translation *y*, that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

Exhaustive search decoding

We want to find a (length T) translation *y*, that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

We could try to compute all possible sequences *y*.

- this means on each step t of the decoder, we are tracking V^t possible partial translations, where V - size of source vocab
- This $O(V^t)$ complexity is far too expensive.

Beam search decoding

At each time step keep tracking *k* most probable translations, which we call hypothesis

k - is the beam size (in practise usually between 5 and 10)

A hypothesis $y_1, \ldots, y_t$ has a score which is its their log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

Beam search decoding

At each time step keep tracking *k* most probable translations, which we call hypothesis

k - is the beam size (in practise usually between 5 and 10)

A hypothesis $y_1, \ldots, y_t$ has a score which is its their log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- score are all negative - higher score, better hypothesis
- we search for high-scoring hypothesis, keep tracking only top-k of them on each step

Beam search decoding: example

At each step keep several best hypothesis

Beam search decoding: example

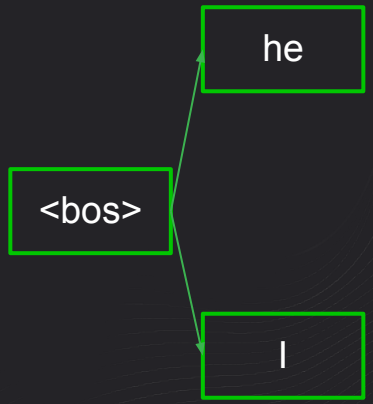
At each step keep several best hypothesis

<bos>

Start with the begin of sentence token or with an empty sequence

Beam search decoding: example

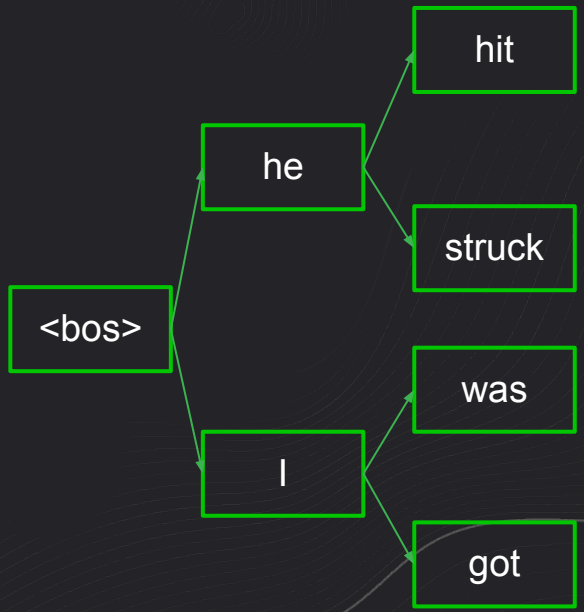
At each step keep several best hypothesis



Generate *beam_size* most probable tokens

Beam search decoding: example

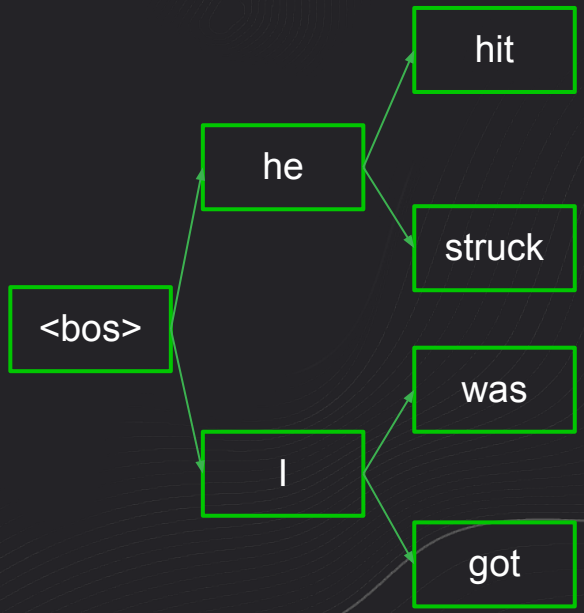
At each step keep several best hypothesis



Generate *beam_size* most probable tokens

Beam search decoding: example

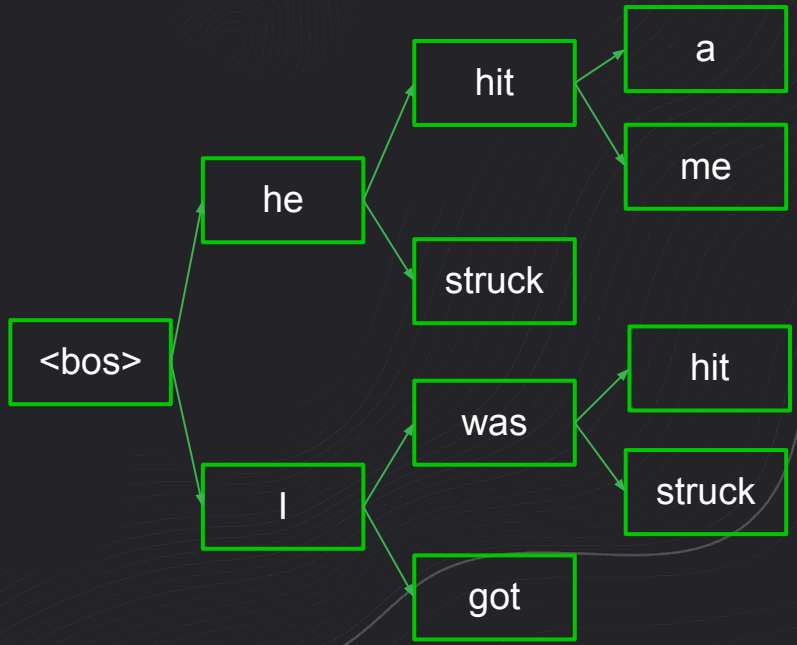
At each step keep several best hypothesis



Look at scores: score(hit| he) > score(was| i) > score(struck| he) > score (got| i)

Beam search decoding: example

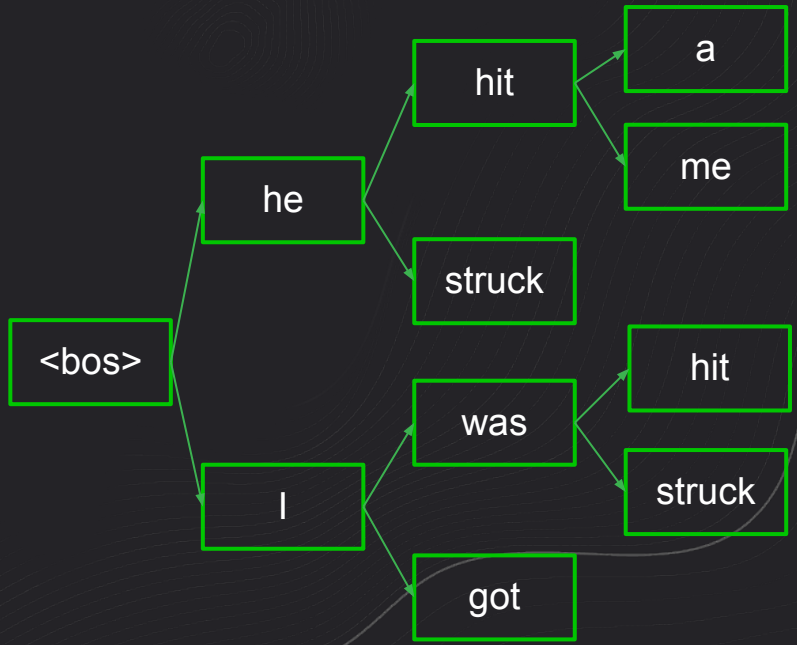
At each step keep several best hypothesis



Generate *beam_size* most probable tokens

Beam search decoding: example

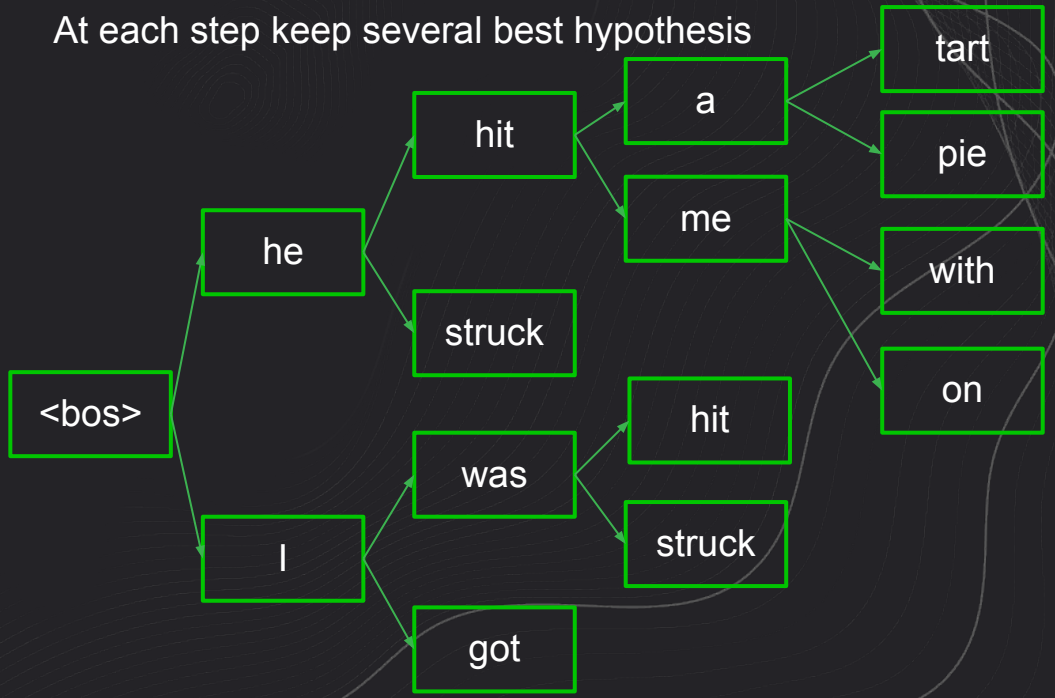
At each step keep several best hypothesis



scores: score(me| he hit) > score(a| he hit) > score(hit| I was) > score (struck| i was)

Beam search decoding: example

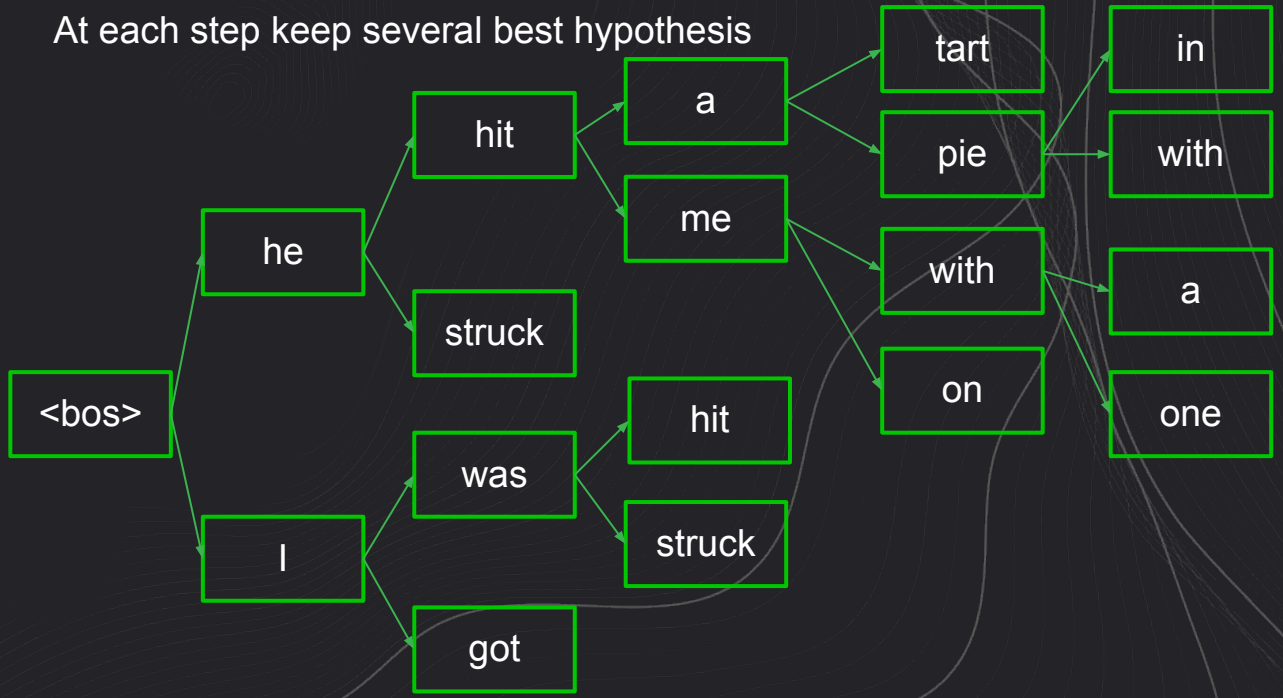
At each step keep several best hypothesis



Generate *beam_size* most probable tokens

Beam search decoding: example

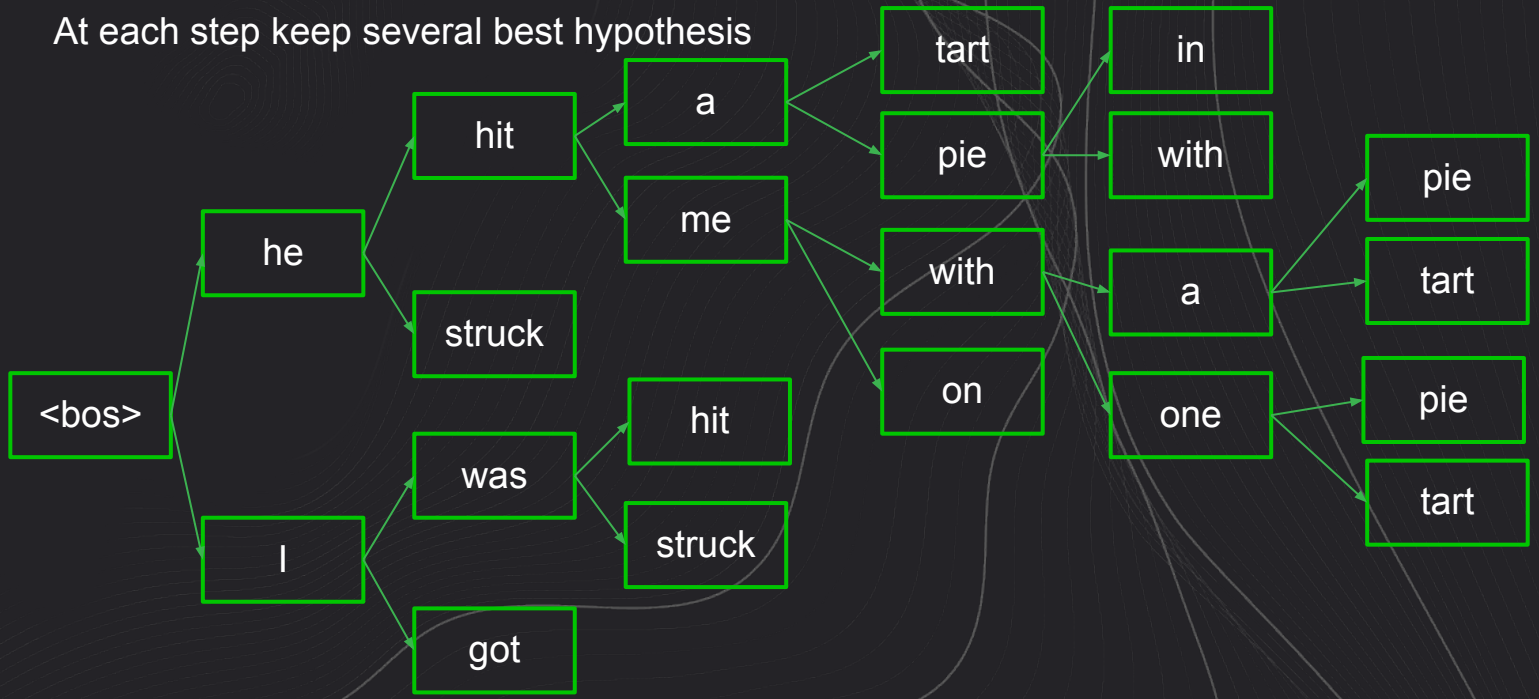
At each step keep several best hypothesis



Generate *beam_size* most probable tokens

Beam search decoding: example

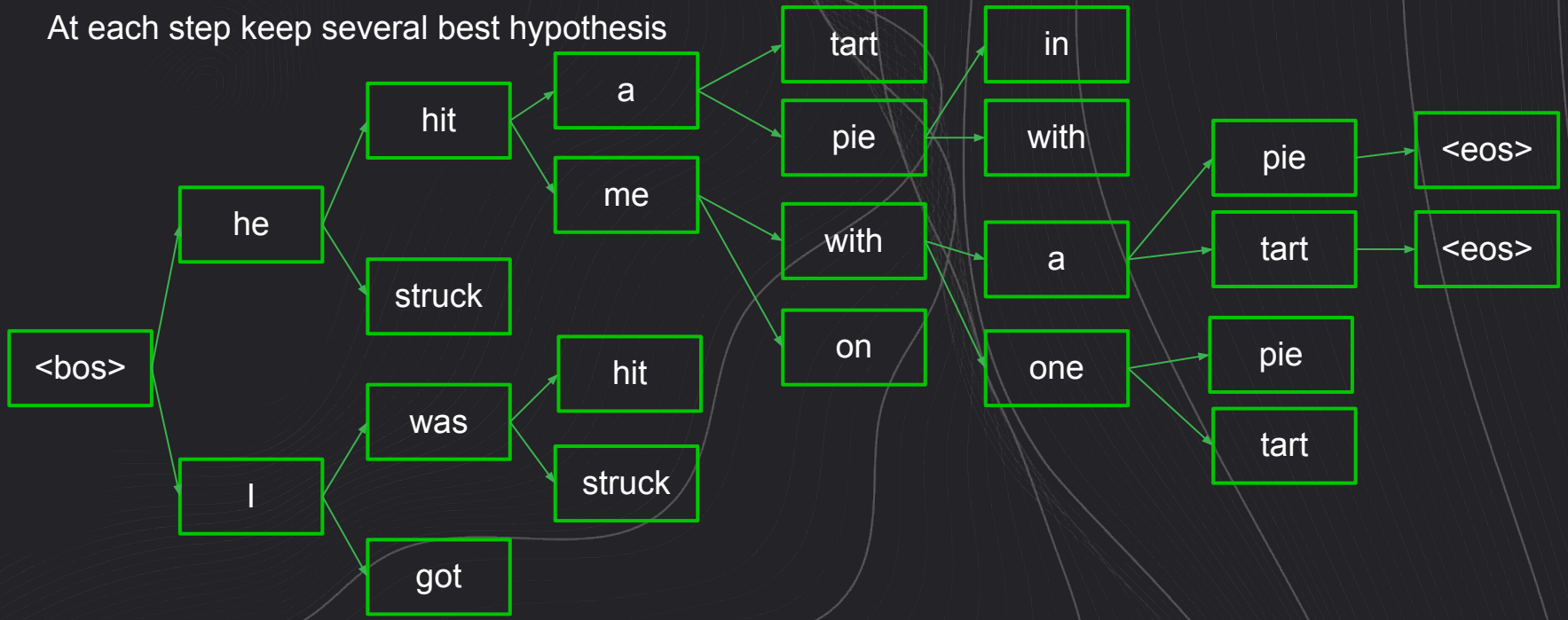
At each step keep several best hypothesis



All hypothesis complete - generation end. Pick one with the highest

Beam search decoding: example

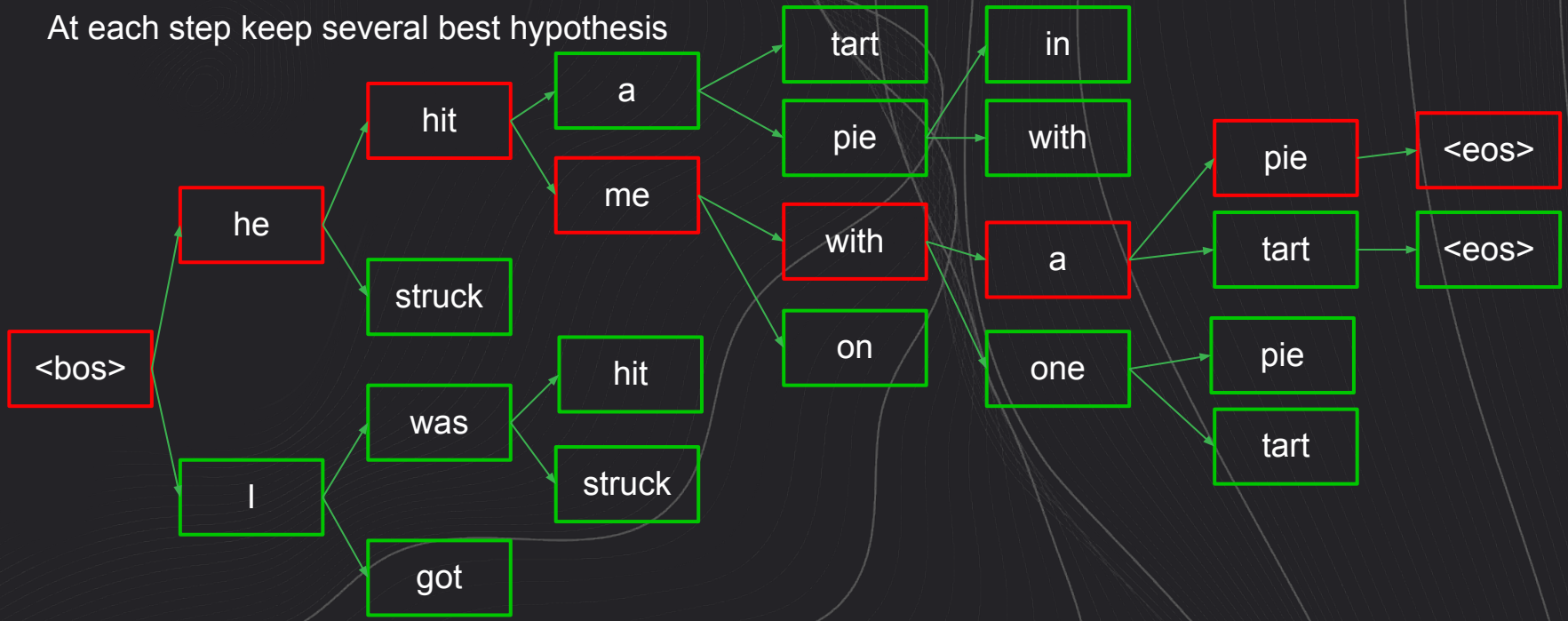
At each step keep several best hypothesis



All hypothesis complete - generation end. Pick one with the highest

Beam search decoding: example

At each step keep several best hypothesis



All hypothesis complete - generation end. Pick one with the highest

Beam search decoding: stopping criteria

- In greedy decoding, usually we decode until the model produces token <EOS>.
- In beam search decoding, different hypothesis can produce <EOS> on different timesteps.
    - when hypothesis complete, place it aside and continue exploring other hypothesis via beam search.
- Usually we continue beam search until:
    - we reach pre-defined timestamp T
    - we have n complete hypothesis.

Beam search decoding

Different hypothesis can have not the same length.

Is it a problem?

Beam search decoding

Different hypothesis can have not the same length.

We can fix it by normalizing by the length of hypothesis!

$$\frac{1}{t}\sum_{i=1}^{t}\log P_{\mathrm{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

Evaluate machine translation

BLEU (BiLingual Evaluation Understudy)

BLEU compares the machine-written translation to one or several human-written translation(s), and compares similarity score based on:

- n-gram precision (usually for 1,2,3,4 n-grams)
- plus a penalty for too short system translations.

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^{n} \text{precision}_i \right)^{1/n} \cdot 100\%$$

$$\text{brevity penalty} = min \left(1, \frac{\text{output length}}{\text{reference length}} \right)$$

BLEU: example

Reference: Israeli officials are responsible for airport security

System A: Israeli officials responsibility for airport safety

System B: Airport security Israeli officials are responsible

Metric	System A	System B
precision (1-gram)	3/6	6/6
precision (2-gram)	1/5	4/5
precision (3-gram)	0/4	2/4
precision (4-gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

Other metrics for MT

- ROUGE
- METEOR
- NIST
- TER
- WER

Sentence representations

“You can’t cram the meaning of a whole sentence ***ing sentence into a single ***ing vector!”

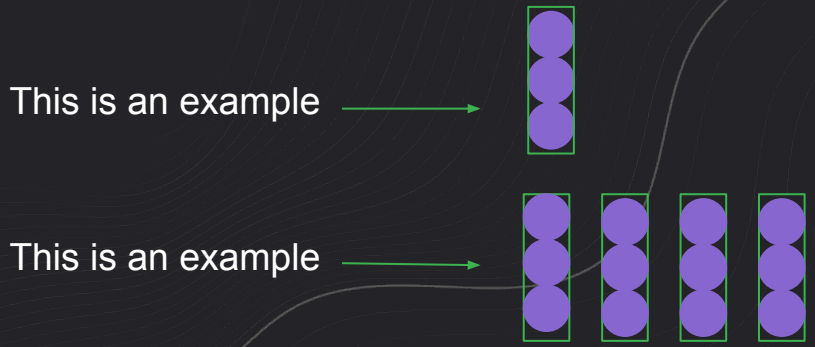
Ray Mooney

Sentence representations

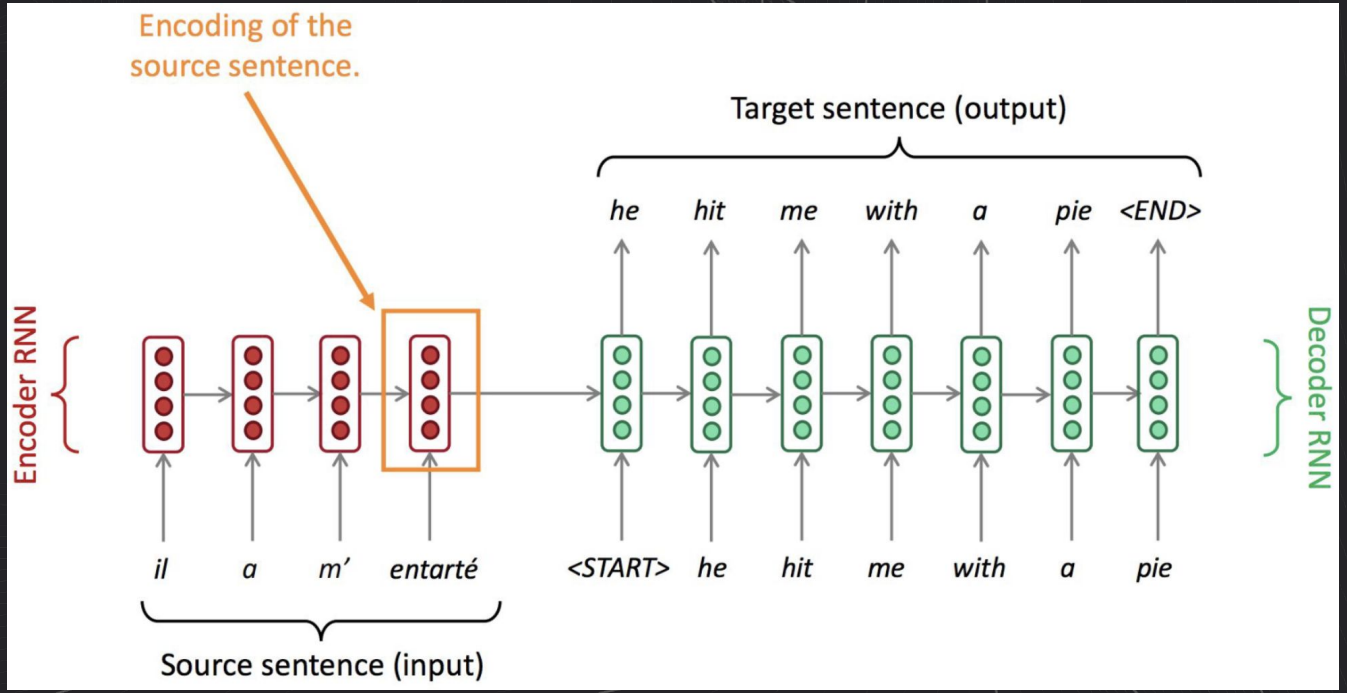
“You can’t cram the meaning of a whole sentence ***ing sentence into a single ***ing vector!”

Ray Mooney

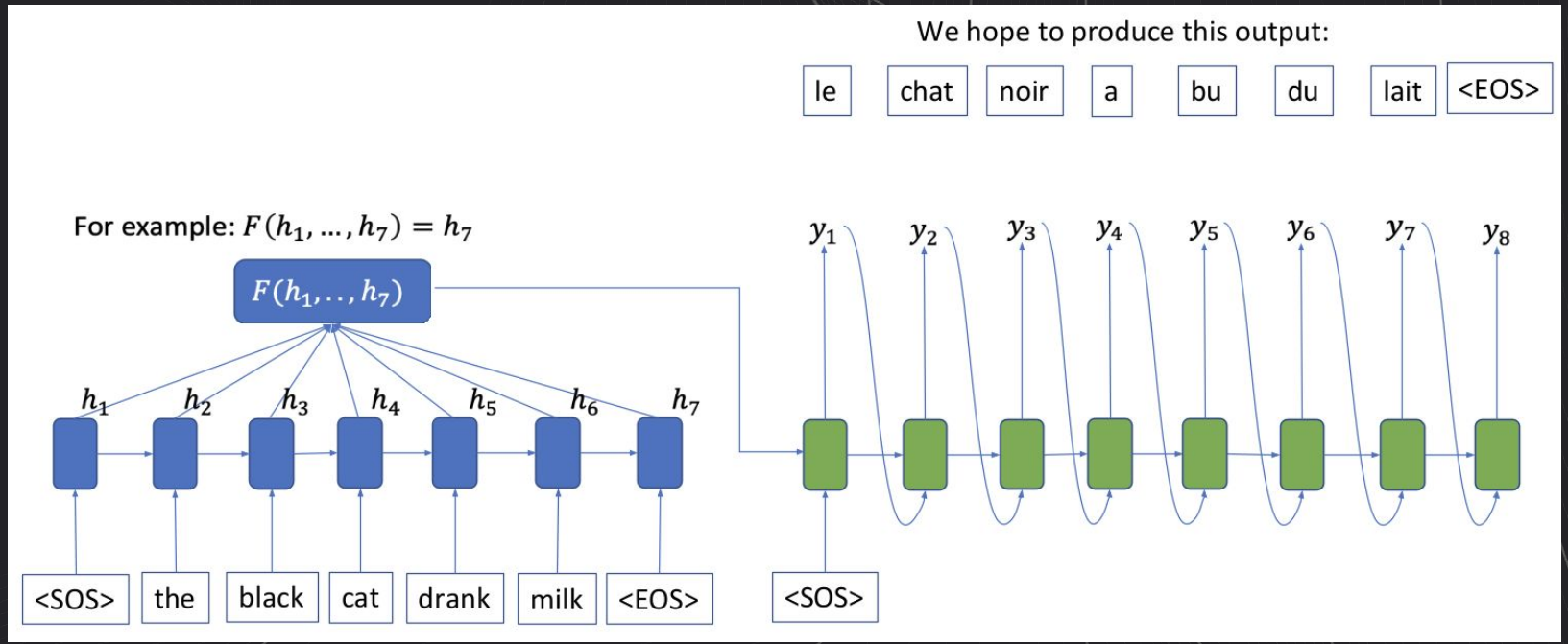
But what if we could use multiple vectors, based on the length of sentence.



Seq2seq: bottleneck problem



How it was

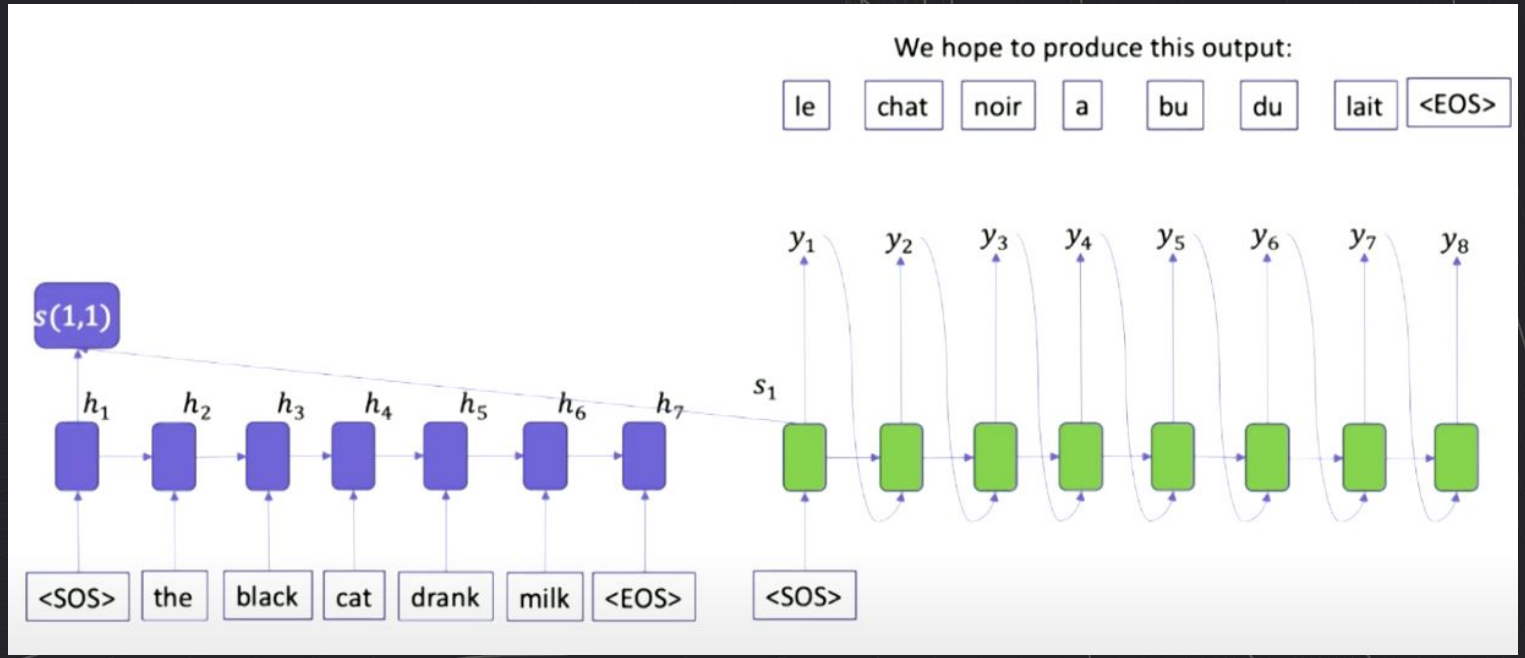

We hope to produce this output:
le
chat
noir
a
bu
du
lait
<EOS>
For example: $F(h_1, \ldots, h_7) = h_7$
$F(h_1, .., h_7)$
h_1 h_2 h_3 h_4 h_5 h_6 h_7
<SOS> the black cat drank milk <EOS>
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
<SOS>

Attention: idea

At different steps, let a model “focus” on different parts of the input.

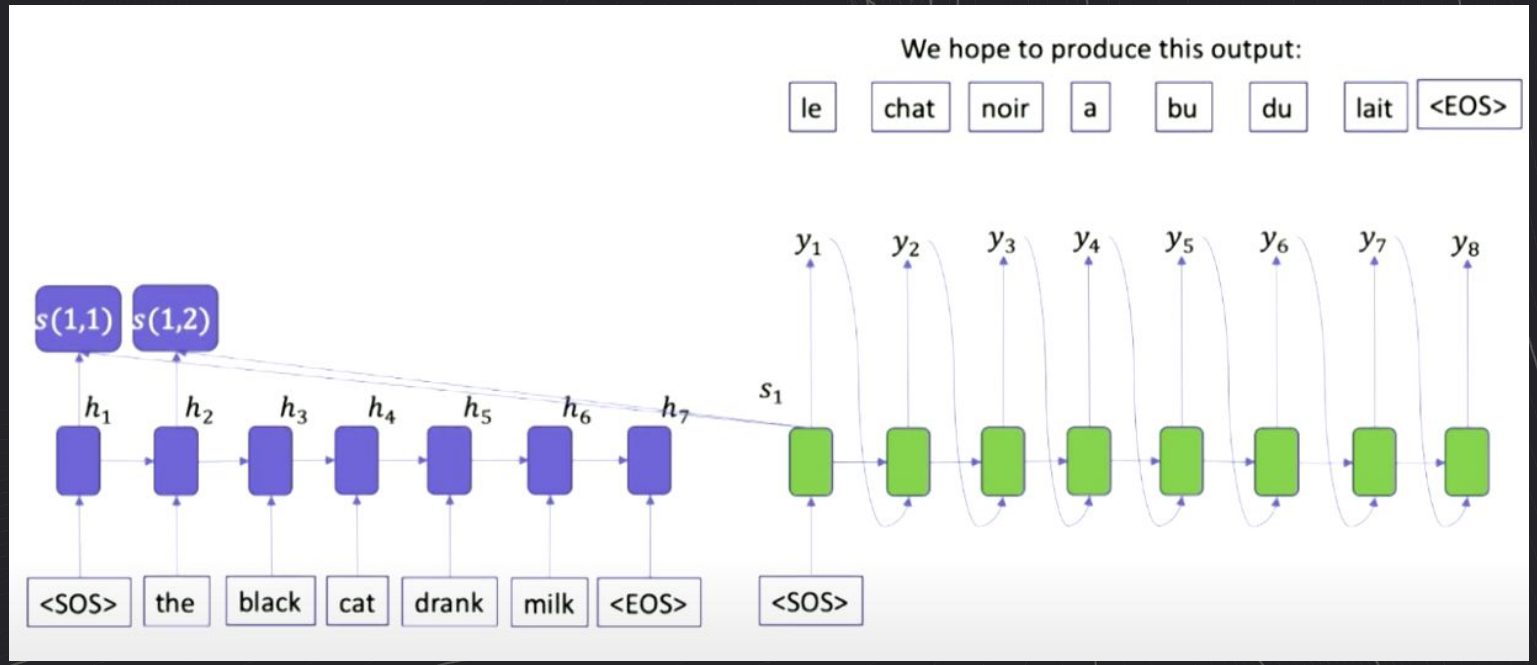
Attention: idea

At different steps, let a model “focus” on different parts of the input.



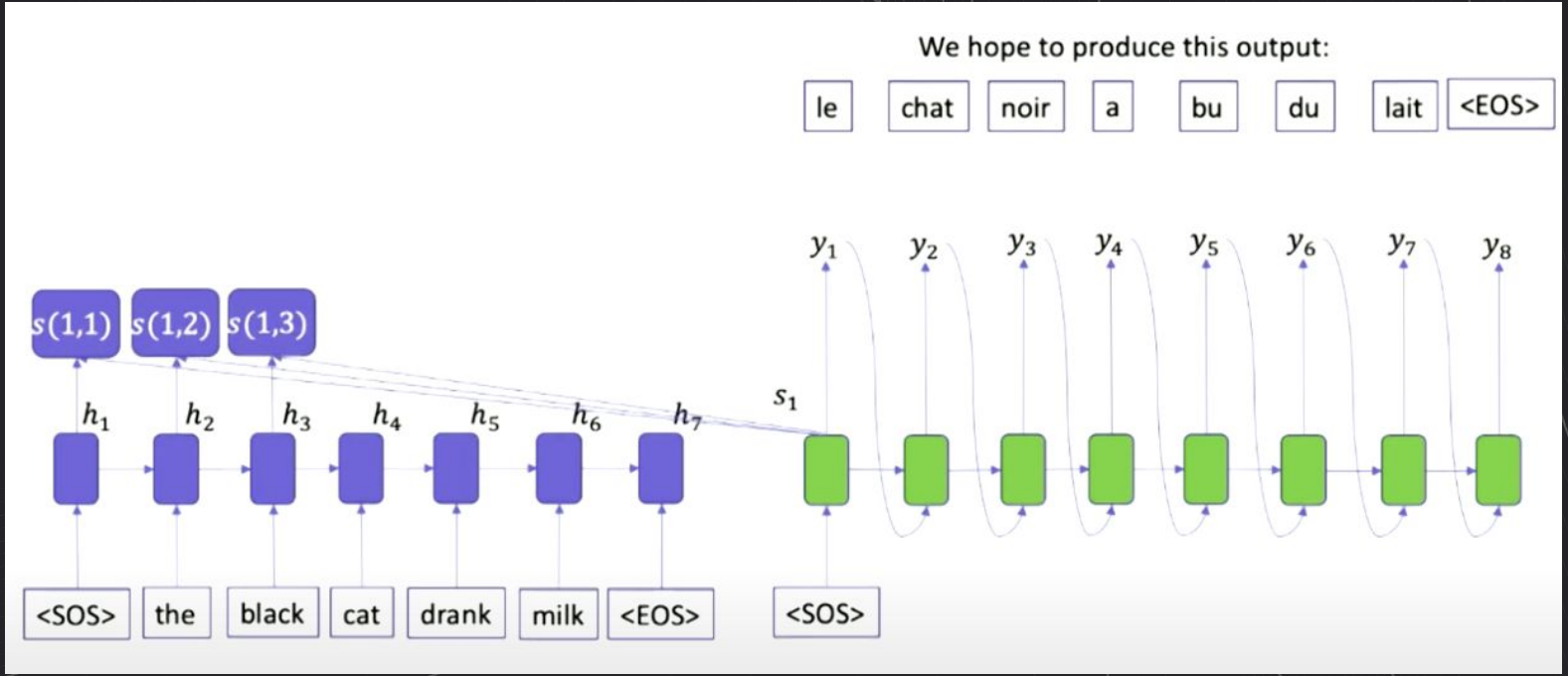
Attention: idea

At different steps, let a model “focus” on different parts of the input.



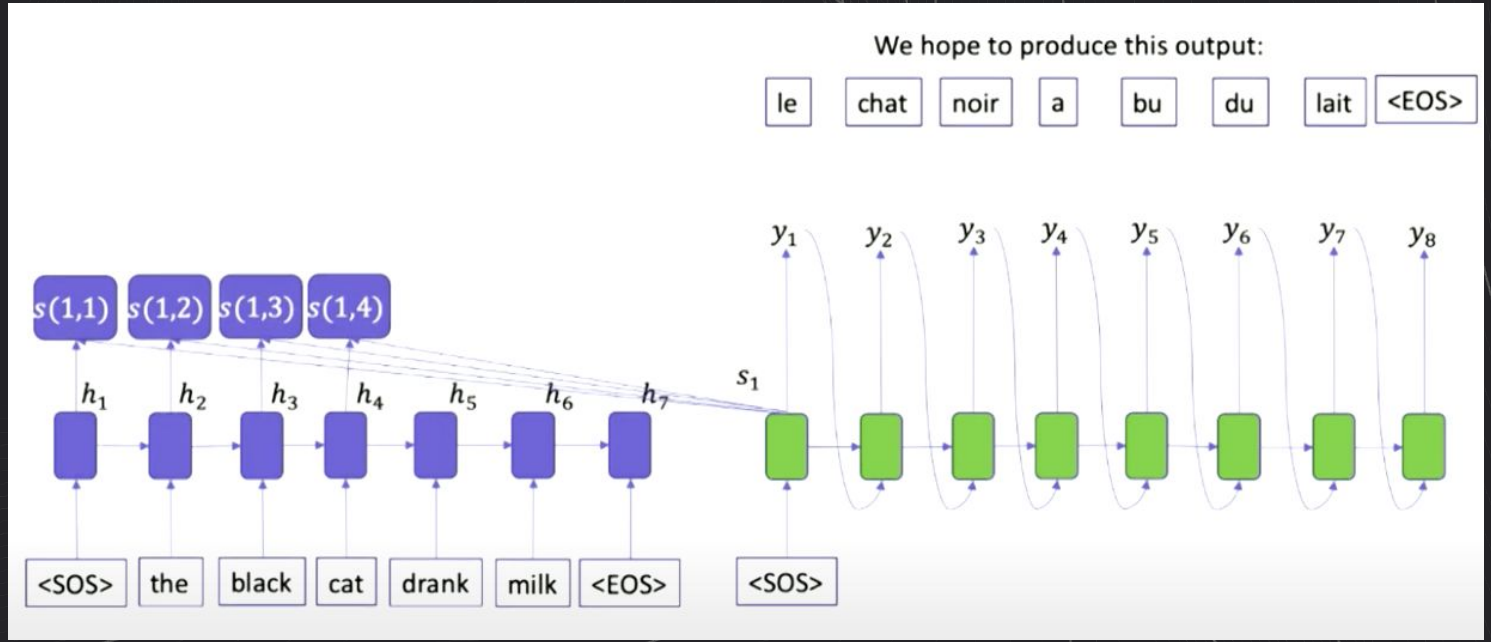
Attention: idea

At different steps, let a model “focus” on different parts of the input.



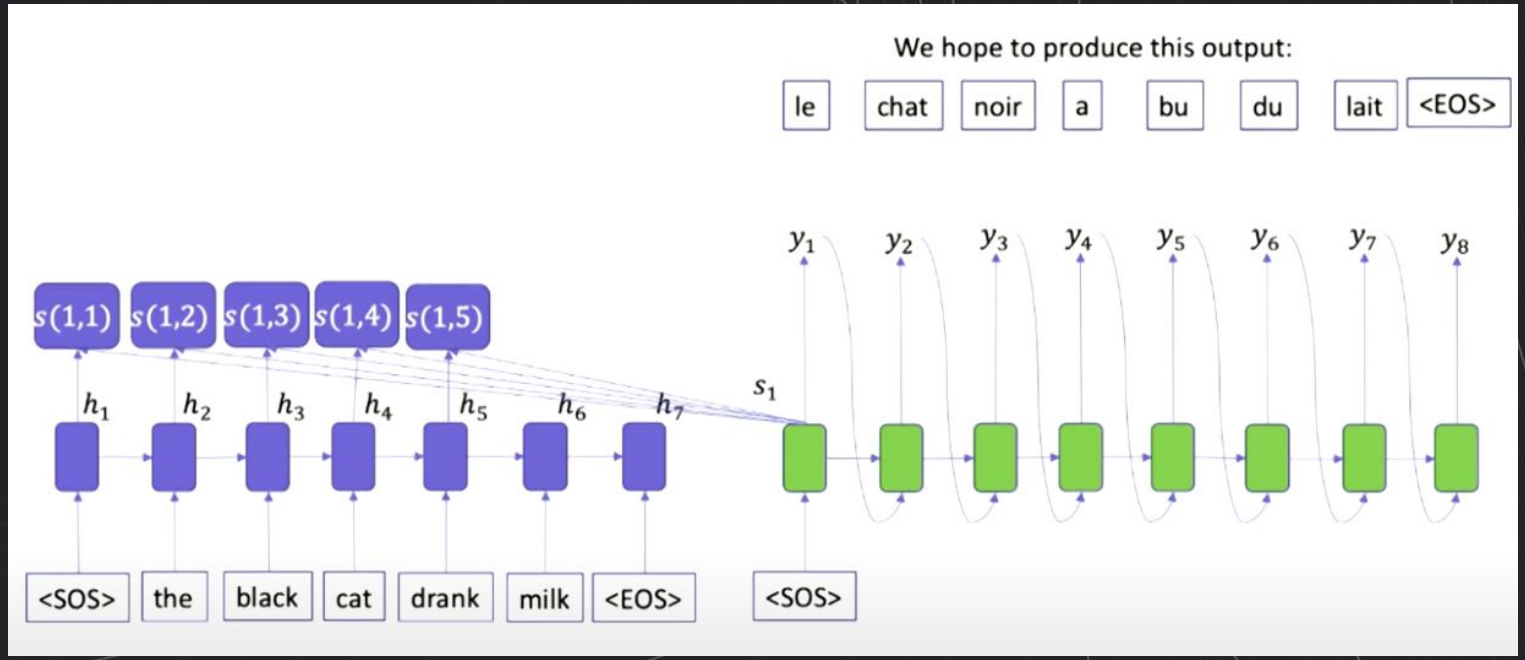
Attention: idea

At different steps, let a model “focus” on different parts of the input.



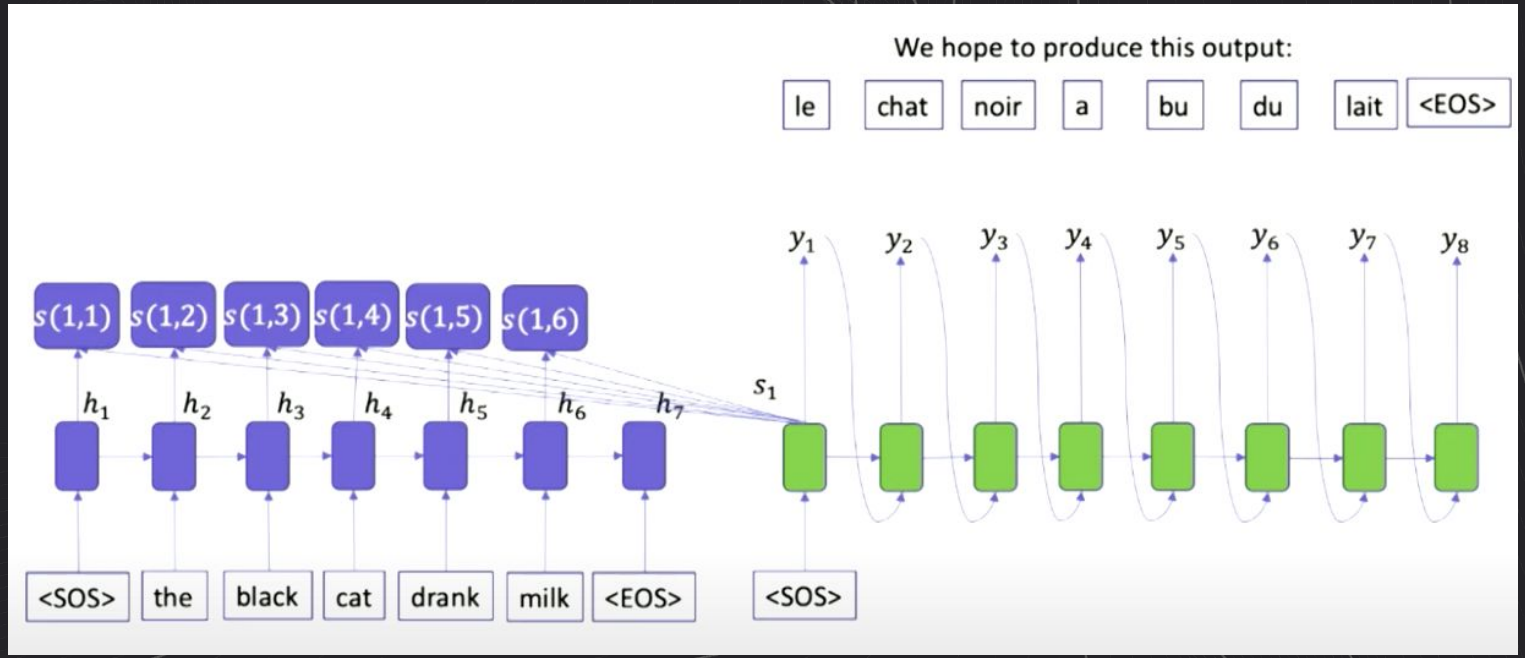
Attention: idea

At different steps, let a model “focus” on different parts of the input.



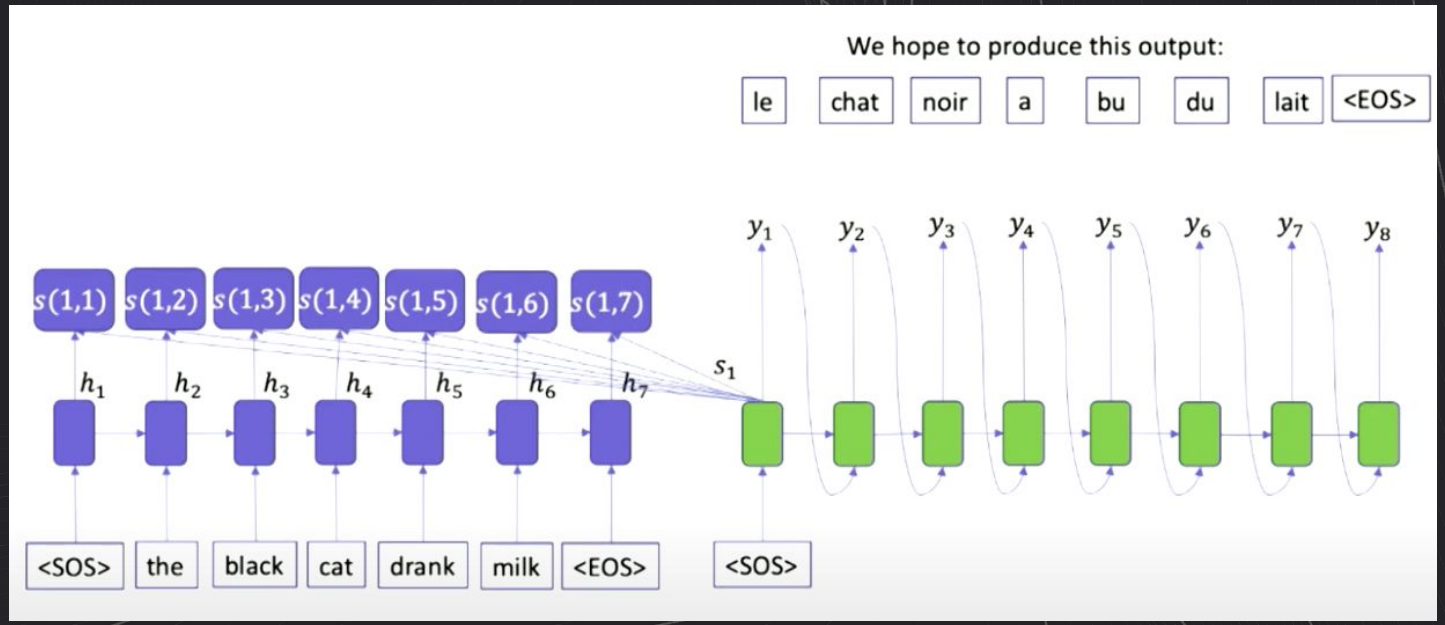
Attention: idea

At different steps, let a model “focus” on different parts of the input.



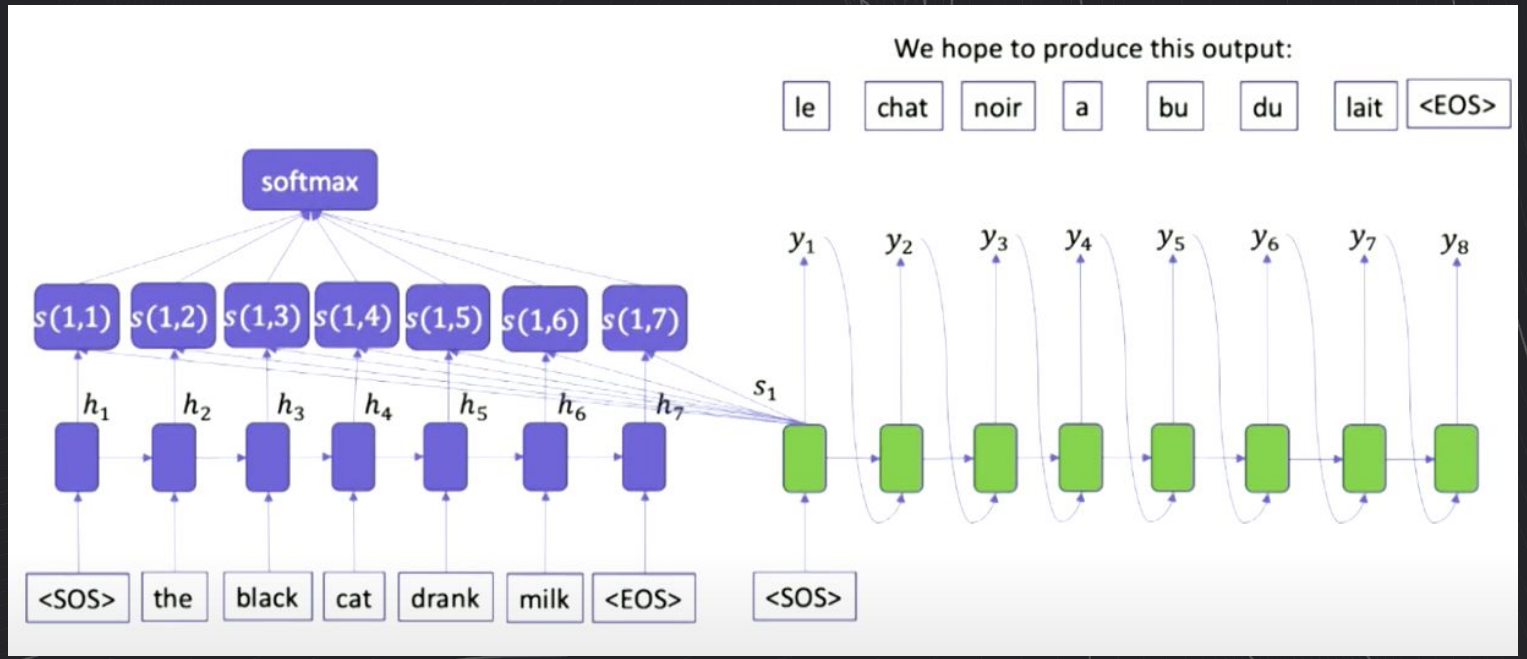
Attention: idea

At different steps, let a model “focus” on different parts of the input.



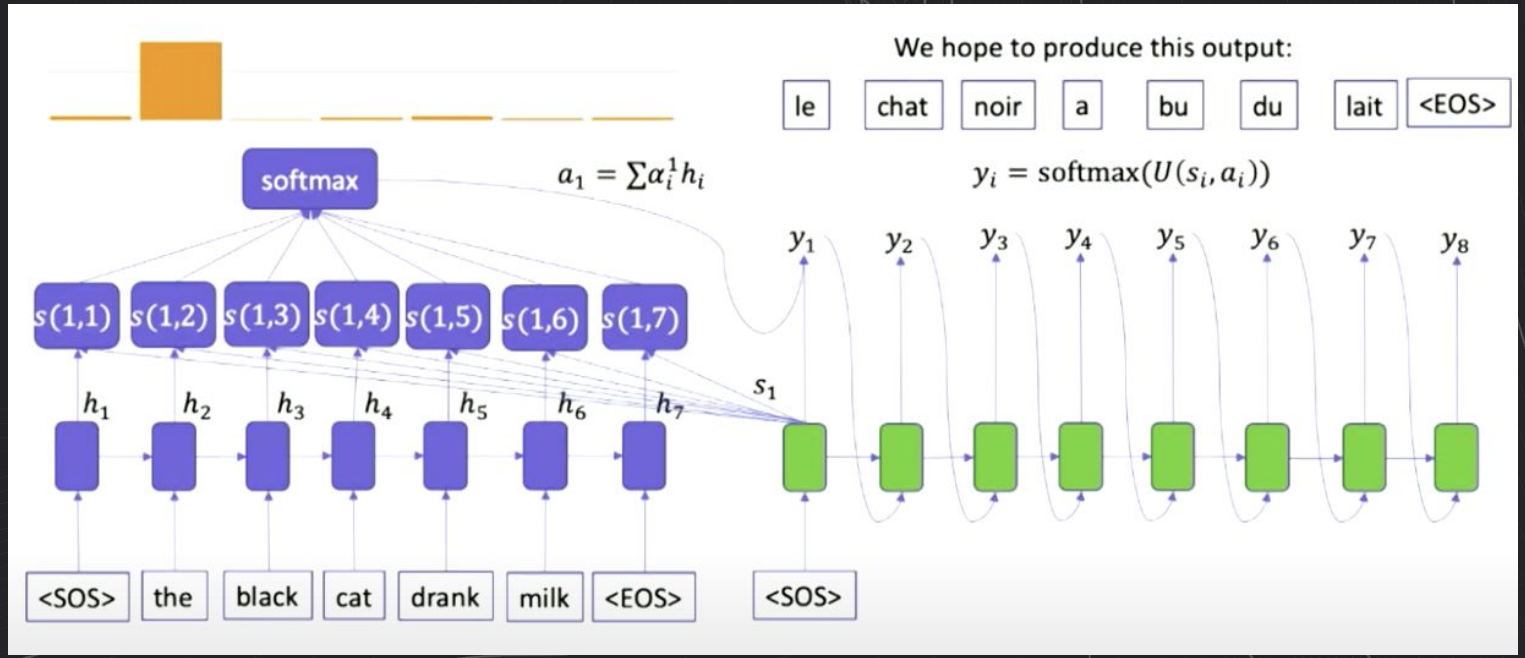
Attention: idea

At different steps, let a model “focus” on different parts of the input.



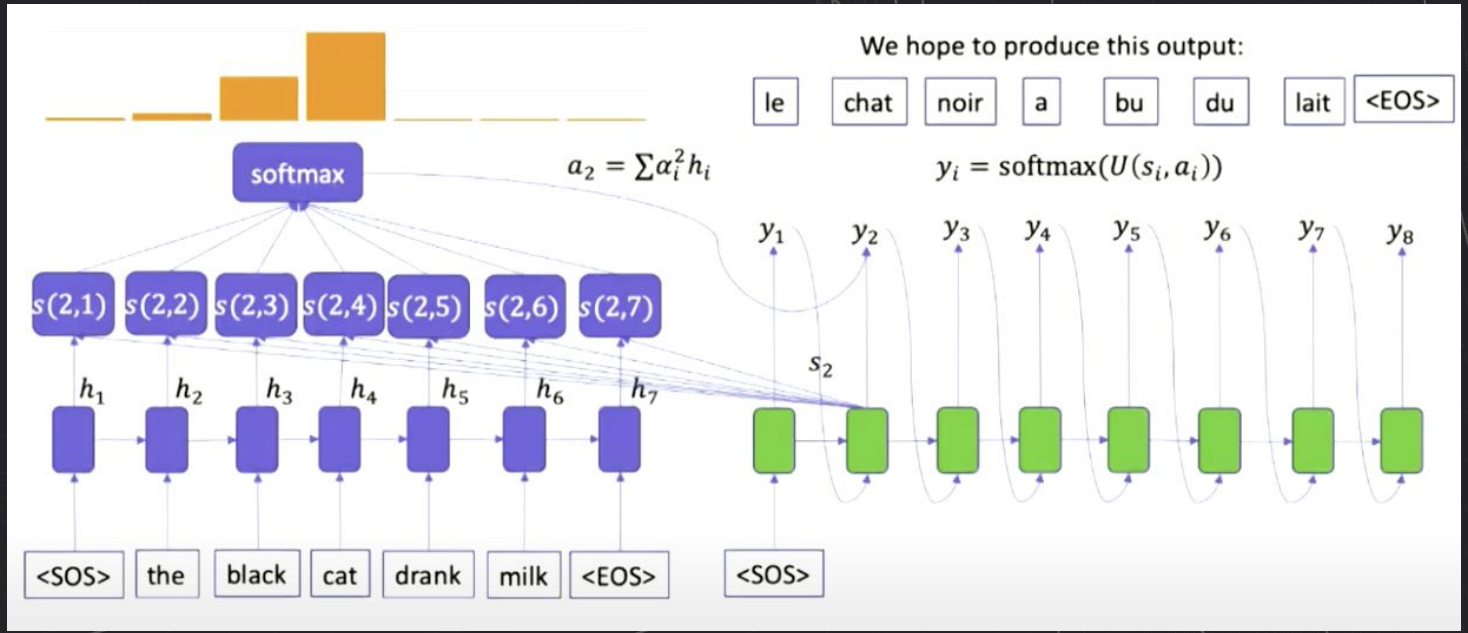
Attention: idea

At different steps, let a model “focus” on different parts of the input.



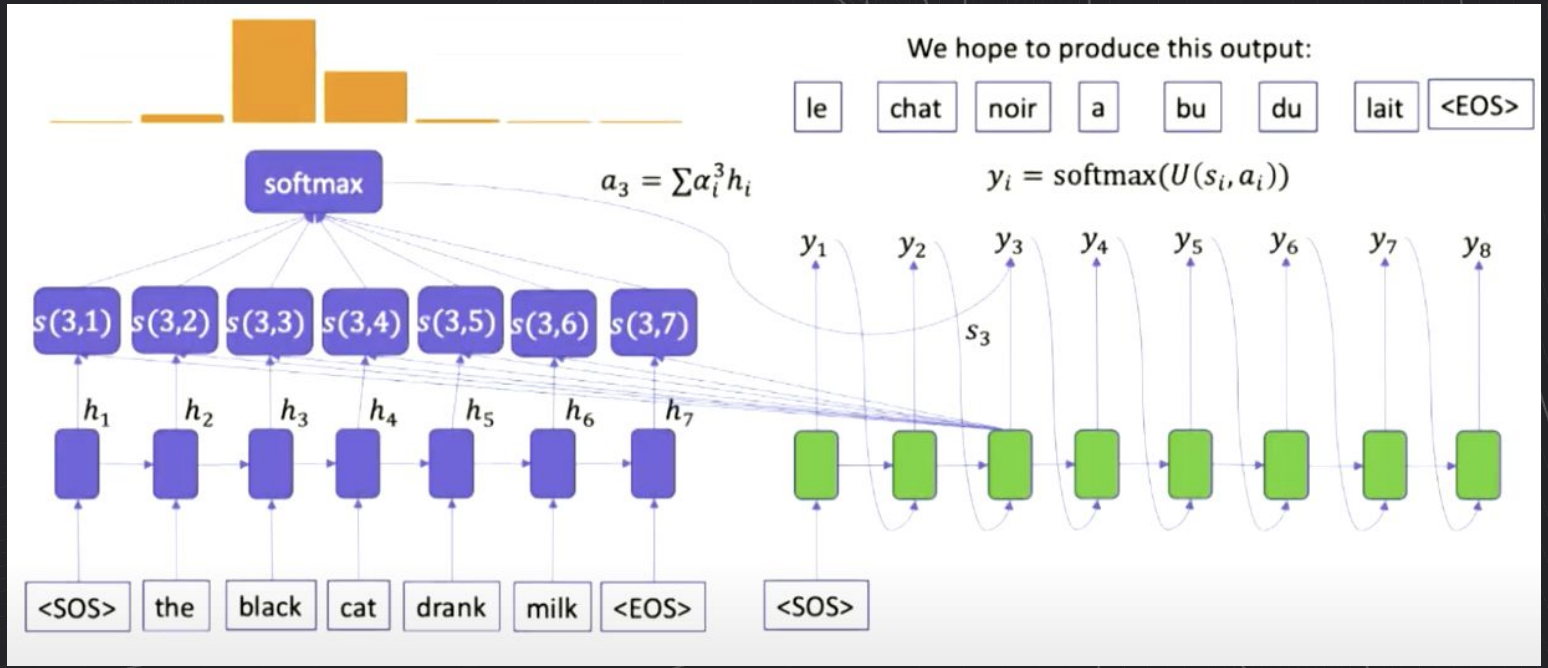
Attention: idea

At different steps, let a model “focus” on different parts of the input.



Attention: idea

At different steps, let a model “focus” on different parts of the input.



Attention: computation pipeline

attention input

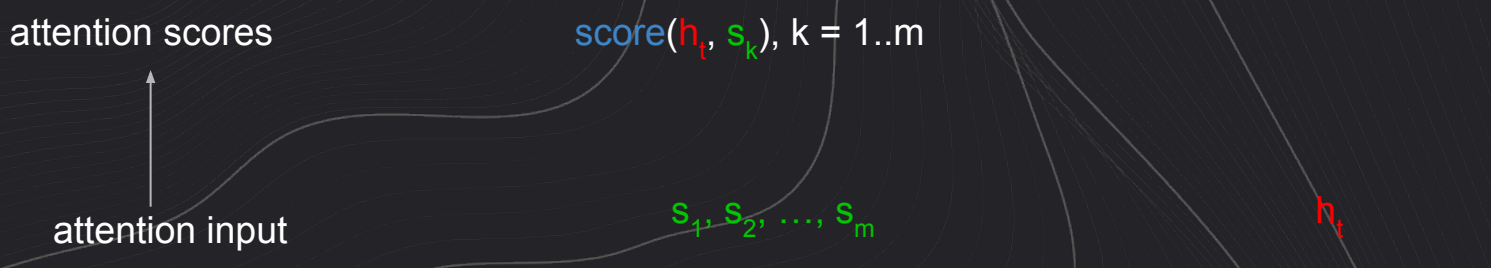
$s_1, s_2, \ldots, s_m$

all encoder states

h_t

one decoder state

Attention: computation pipeline



Attention: computation pipeline

attention scores

attention input

score(h_t, s_k), k = 1..m

“How relevant is source token k for target step t?”

s_1, s_2, …, s_m

all encoder states

h_t

one decoder state

Attention: computation pipeline

attention weights

attention scores

attention input

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

score(h_t, s_k), k = 1..m

"How relevant is source token k for target step t?"

$s_1, s_2, \ldots, s_m$

all encoder states

h_t

one decoder state

Attention: computation pipeline

attention weights

attention scores

attention input

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

“attention weight for source token k at decoder step t”

score(h_t, s_k), k = 1..m

“How relevant is source token k for target step t?”

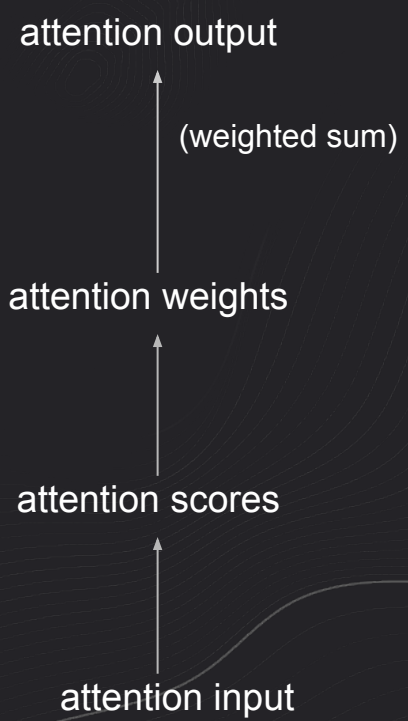
$s_1, s_2, \ldots, s_m$

all encoder states

h_t

one decoder state

Attention: computation pipeline



$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

“attention weight for source token k at decoder step t”

score(h_t, s_k), k = 1..m

“How relevant is source token k for target step t?”

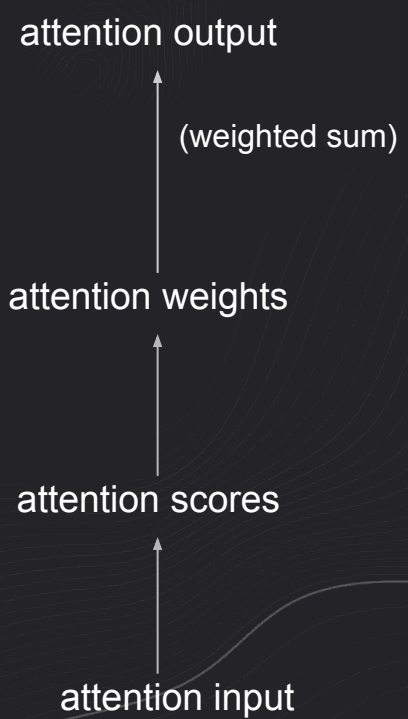
$s_1, s_2, \ldots, s_m$

all encoder states

h_t

one decoder state

Attention: computation pipeline



$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

score(h_t, s_k), k = 1..m

"How relevant is source token k for target step t?"

$s_1, s_2, \ldots, s_m$

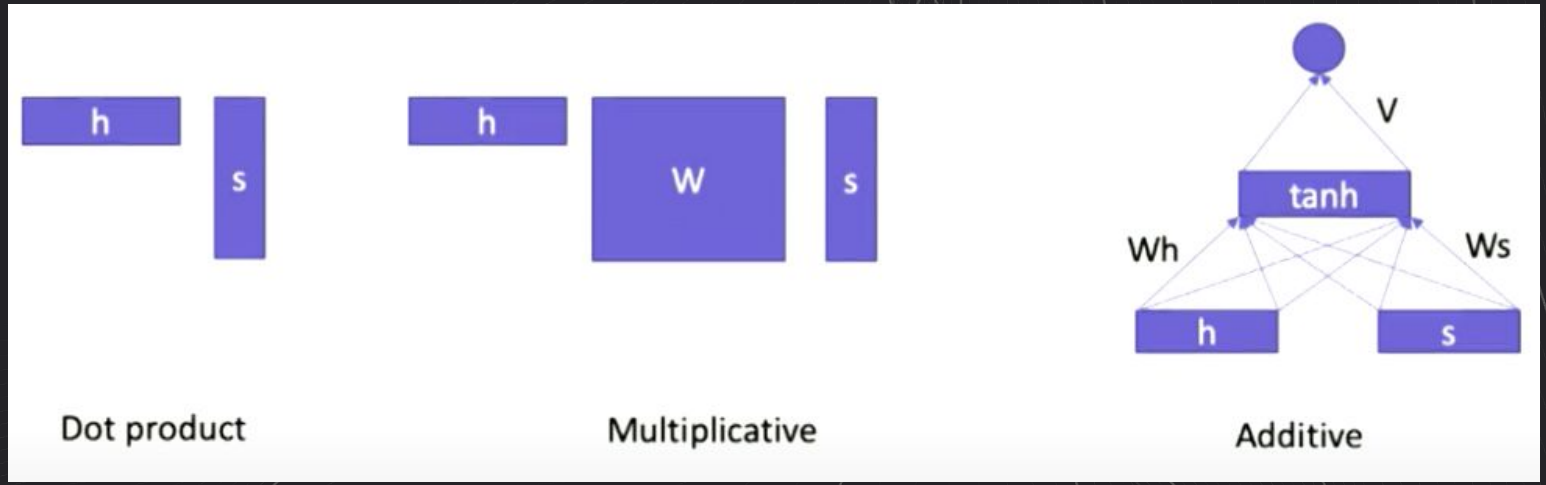
all encoder states

h_t

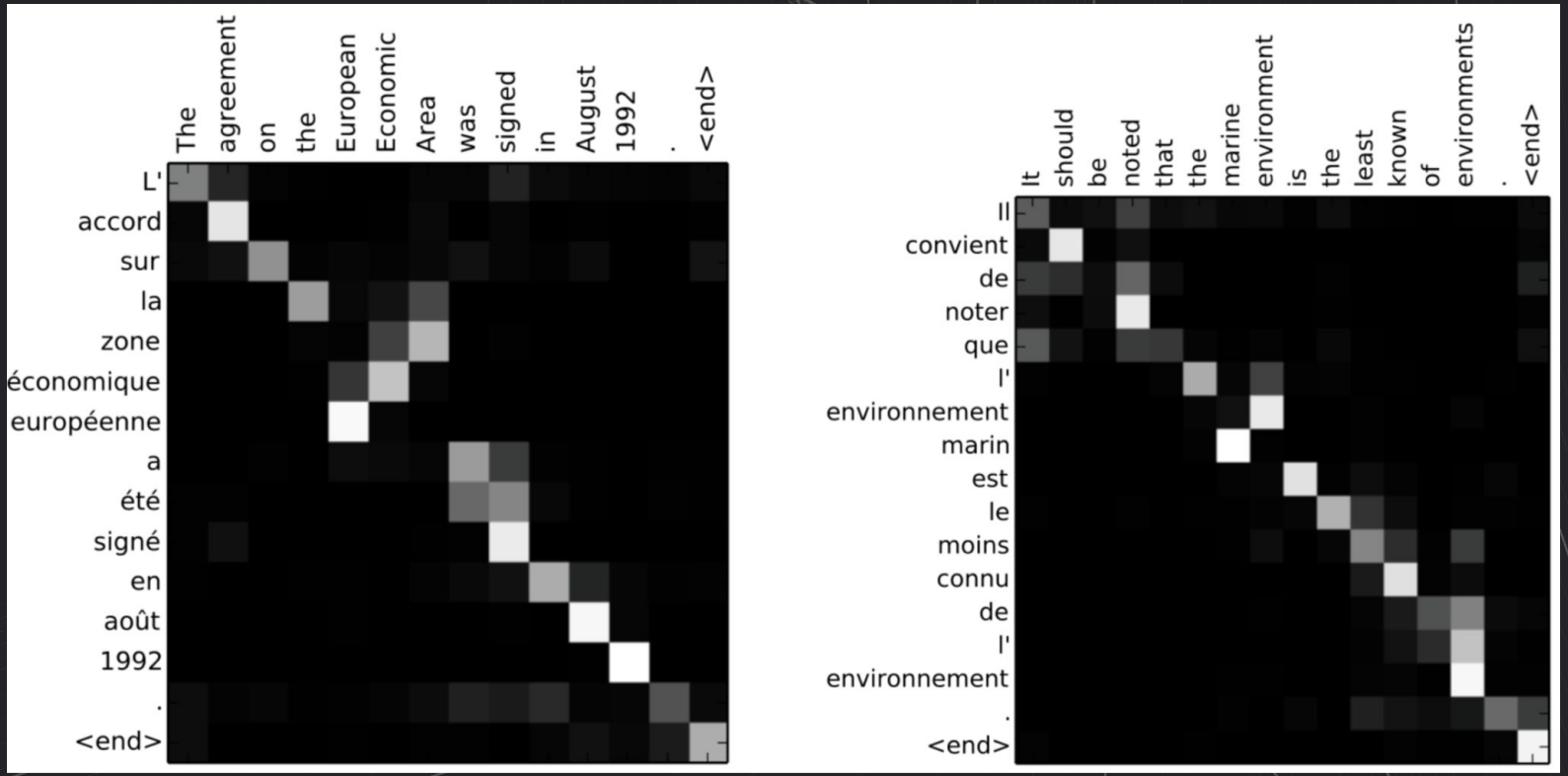
one decoder state

Attention: variations

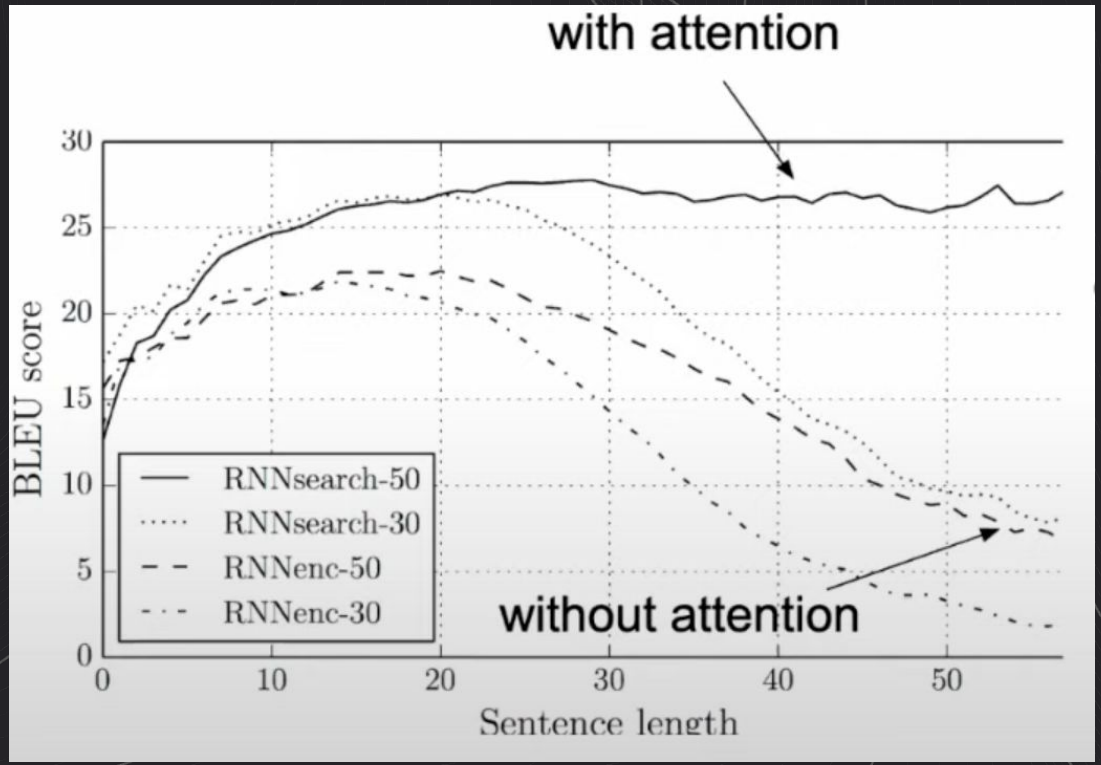
s(*i*, *j*) is a function that gives score that gives for how much the encoder hidden state on each timestamp *j* influence on decoder state *i*.



Attention learned (almost) alignment



Quality of NMT with/without Attention



Self-Attention

Why “Self”?

Decoder-Encoder:

- from: one current decoder state
- at: all encoder states

Self-attention:

- from: each state from a set of states
- at: all other states in the same set

Self-Attention: intuition

"The animal didn't cross the street because it was tired."

What does "it" in this sentence reference to?

We want self-attention to associate "it" with "animals".

Self-Attention: intuition

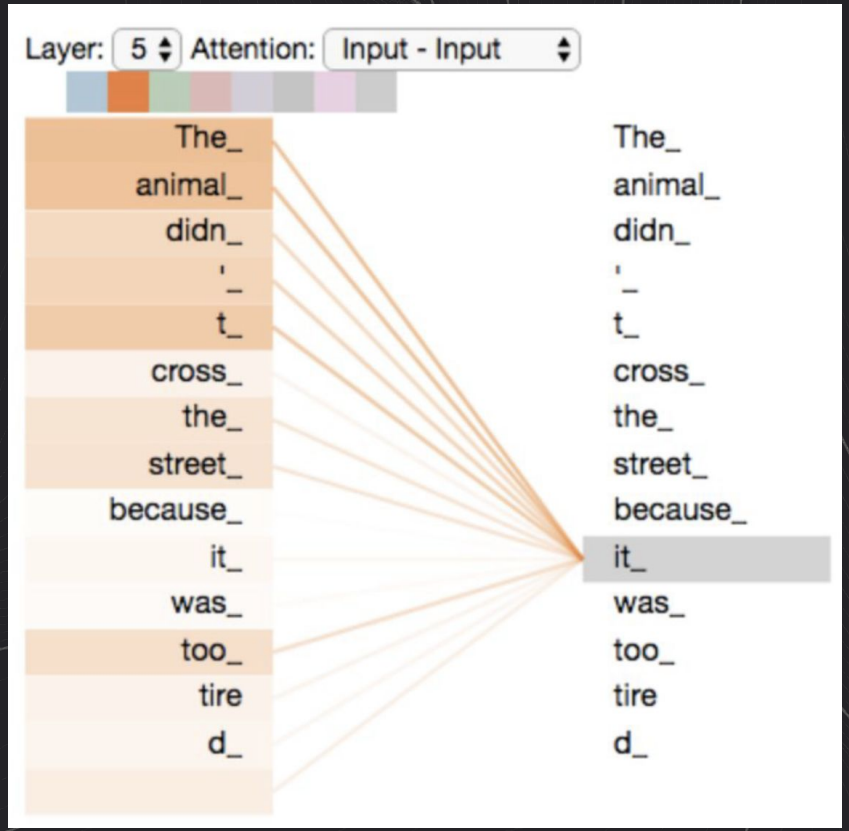
"The animal didn't cross the street because it was tired."

What does "it" in this sentence reference to?

We want self-attention to associate "it" with "animals".

Word, for which we want to get associations we will name "Query".

Self-Attention: intuition



Self-Attention: intuition

As in basic attention we can represent each vector as weighted sum.

Self-attention as weighted sum:

$$A_i = \sum_{j=0}^{T} a_{ij} x_j$$

output corresponding to the i-th input

weight based on similarity between current input x_i and all other inputs

Self-Attention: intuition

As in basic attention we can represent each vector as weighted sum.

Self-attention as weighted sum:

$$A_i = \sum_{j=0}^{T} a_{ij} x_j$$

output corresponding to the i-th input

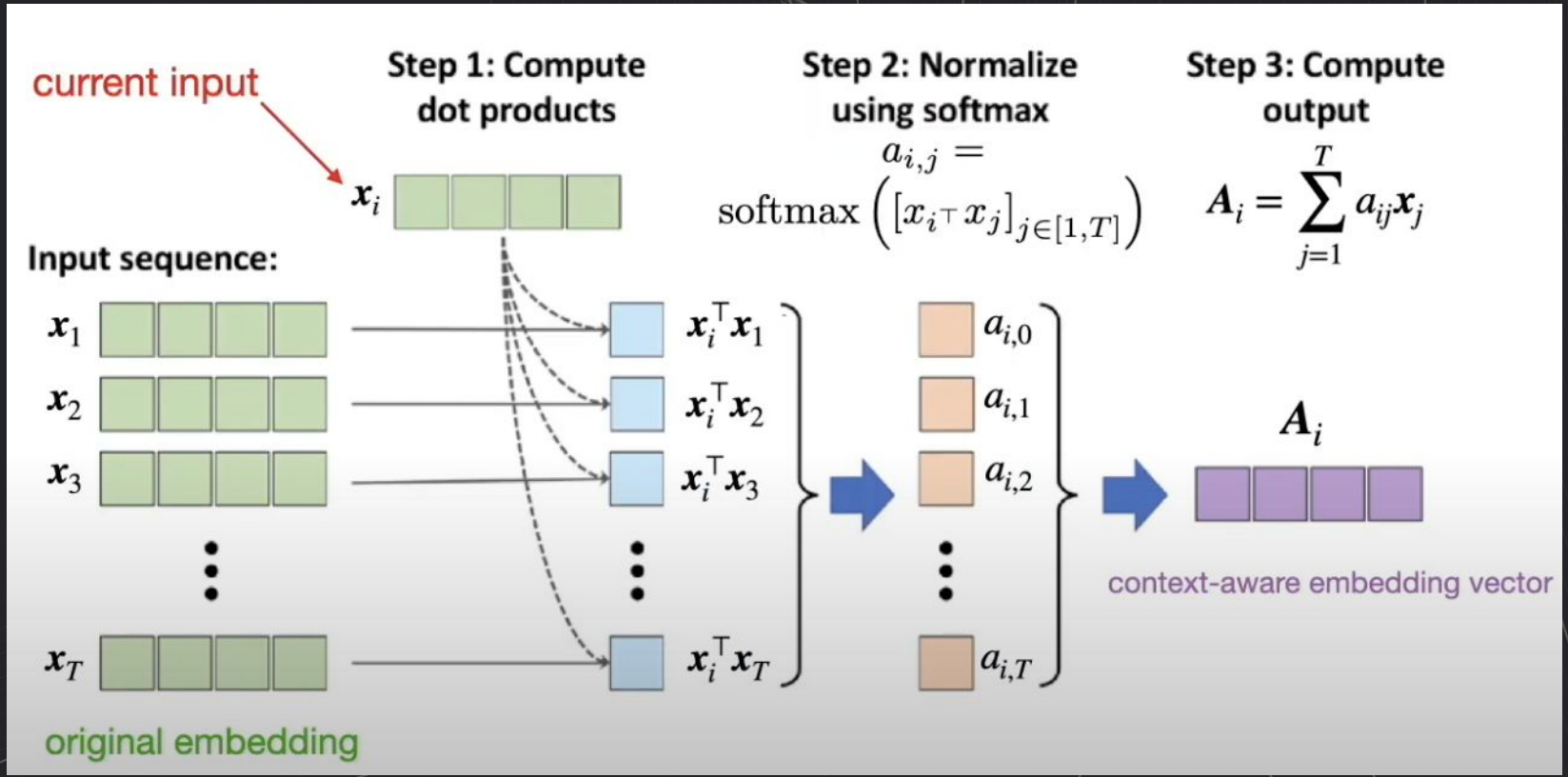
weight based on similarity between current input x_i and all other inputs

Assume that $e_{i,j}$ is some score of similarity

repeat this for all inputs $j \in \{1...T\}$, then normalize

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{j=1}^{T} \exp\left(e_{ij}\right)} = \text{softmax}\left(\left[e_{ij}\right]_{j=1....T}\right)$$

Self-Attention: intuition



Self-Attention: intuition

If we add 3 matrices that will be learning during training, we will get far more information for our language model. This matrices will be for key, query and values. Each of representation will be responsible for each word representation.

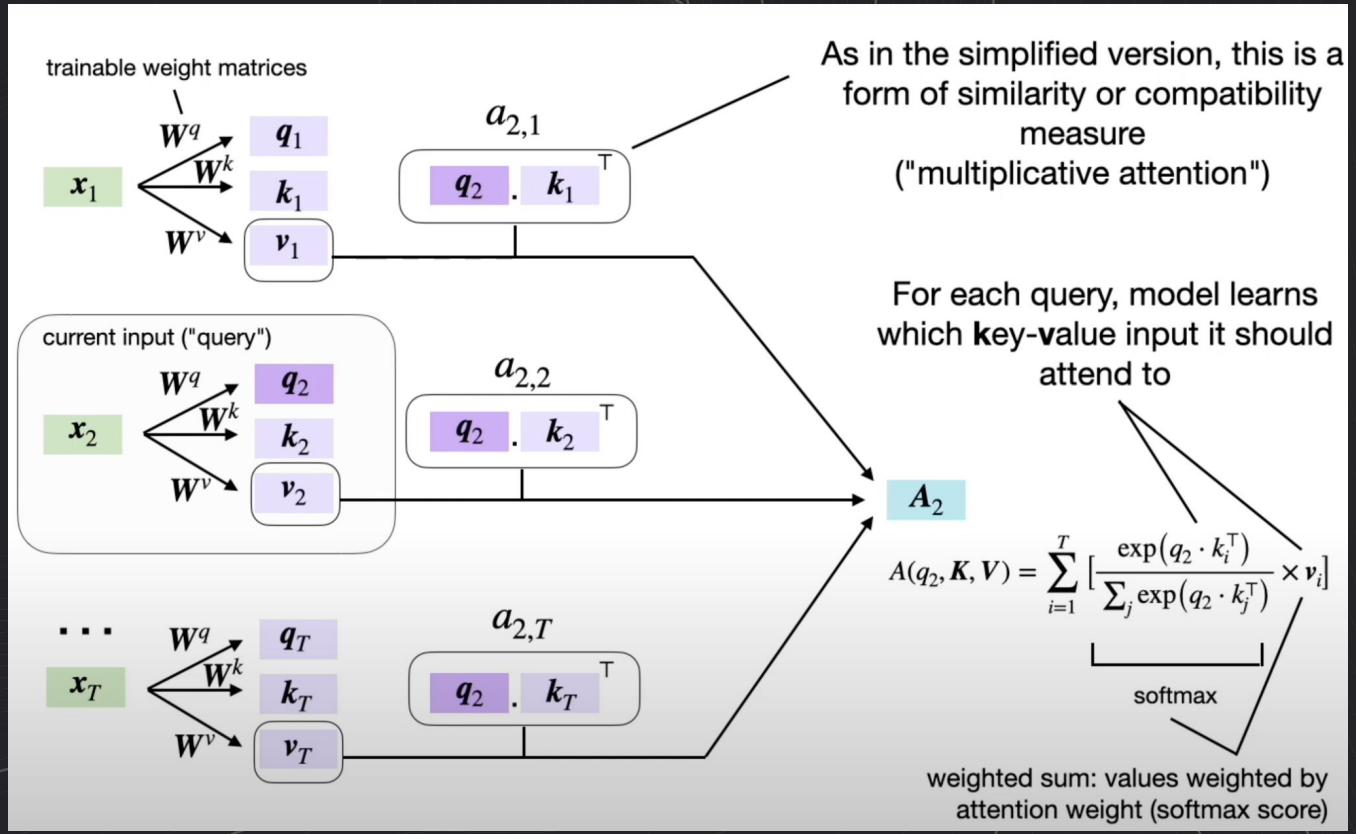
More generally we will write it like this:

query: $W^q x_i$

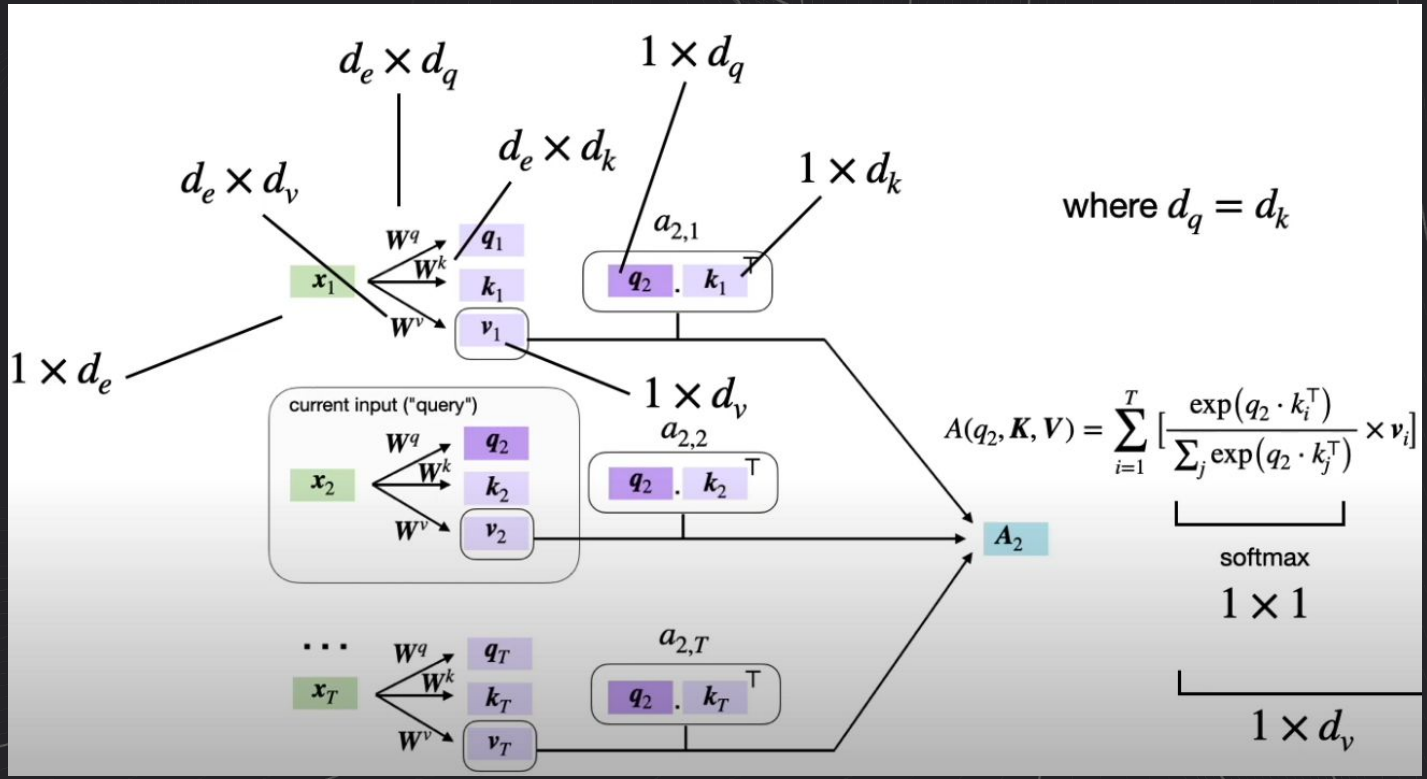
key: $W^k x_i$

value: $W^v x_i$

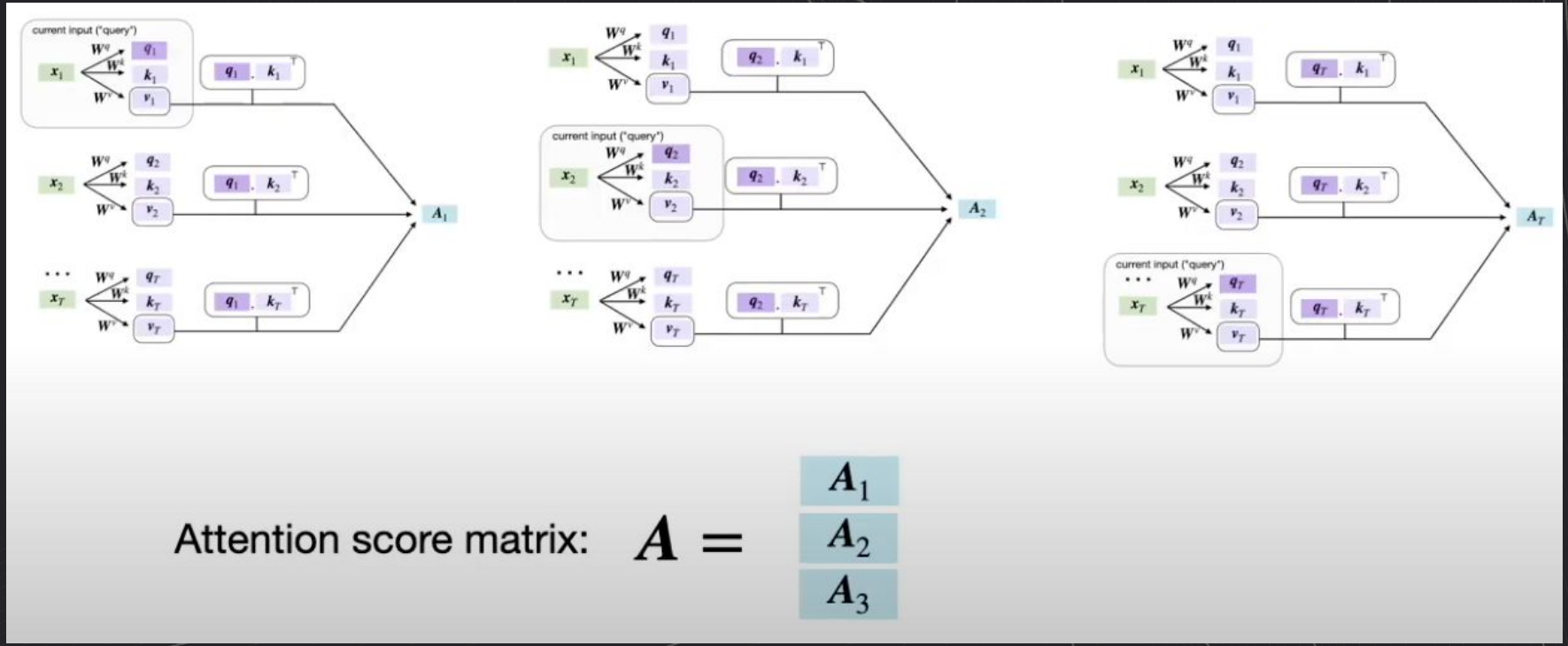
Self-Attention



Self-Attention



Self-Attention



Attention score matrix: $A = \begin{matrix} A_1 \\ A_2 \\ A_3 \end{matrix}$

Self-Attention

To ensure dot products between query and key don’t grow too large we add scaling factor in self-attention formula.

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

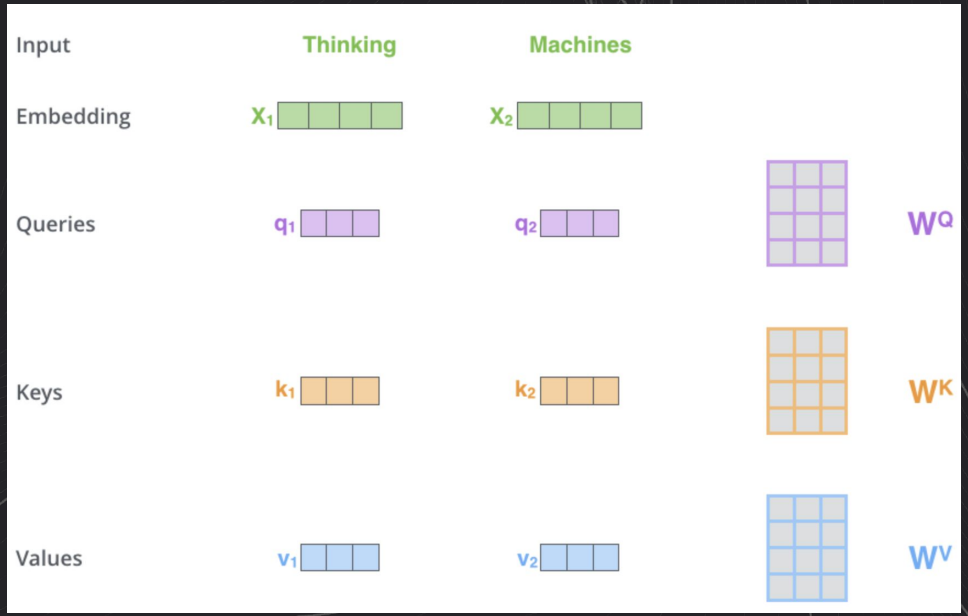
Self-Attention

To ensure dot products between query and key don’t grow too large we add scaling factor in self-attention formula.

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

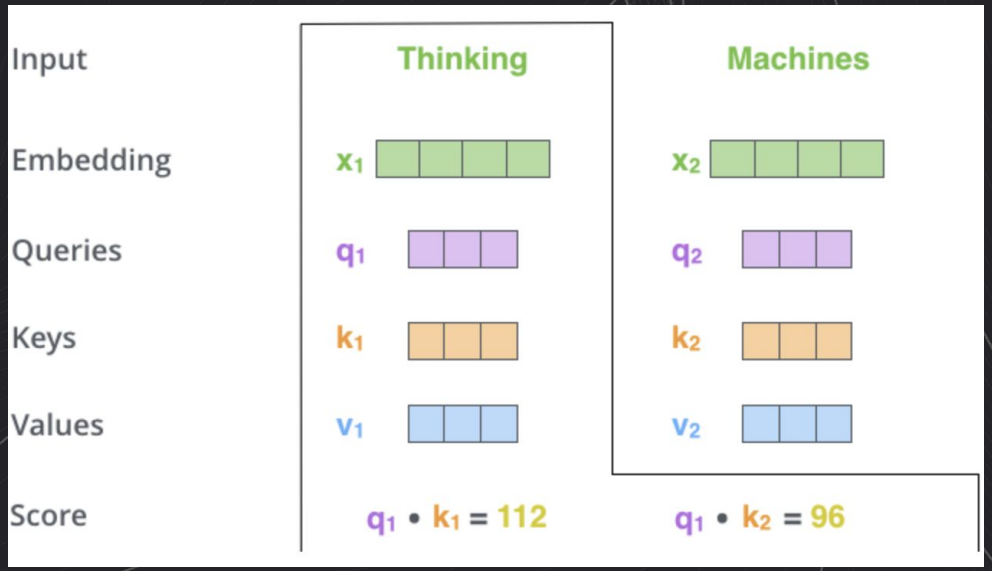
Self-Attention in detail

Step 1. Get three representations of each word vector (Query, Key, Value) by multiplying on corresponding matrix. Dimension of each vector will be 64.



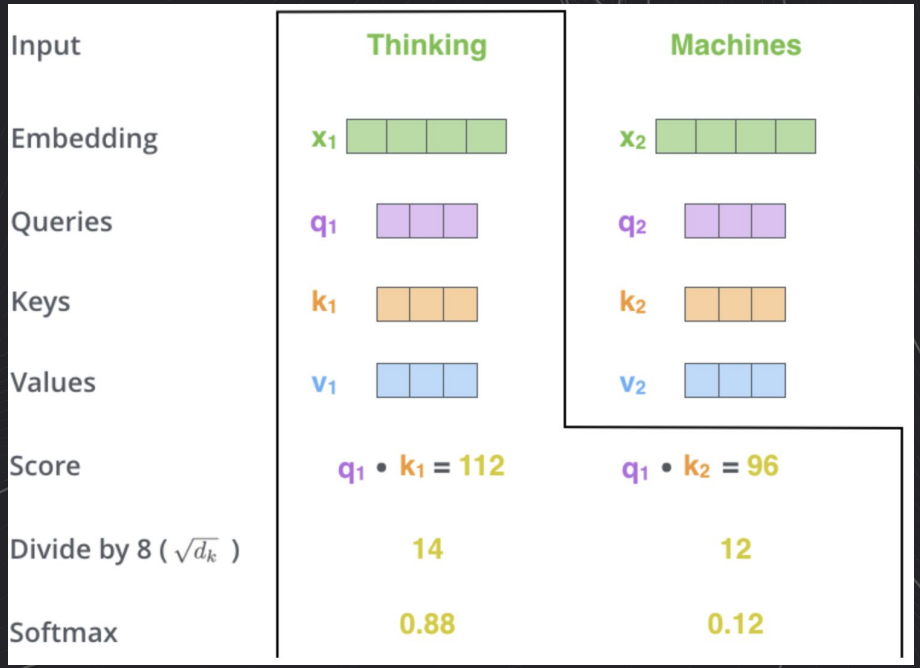
Self-Attention in detail

Step 2. Now let's calculate a score. Say we're calculating the self-attention for the first word in this example, "Thinking". We need to score each word of the input sentence against this word. The score is calculated by taking the dot product of the query vector with the key vector of the respective word we're scoring.



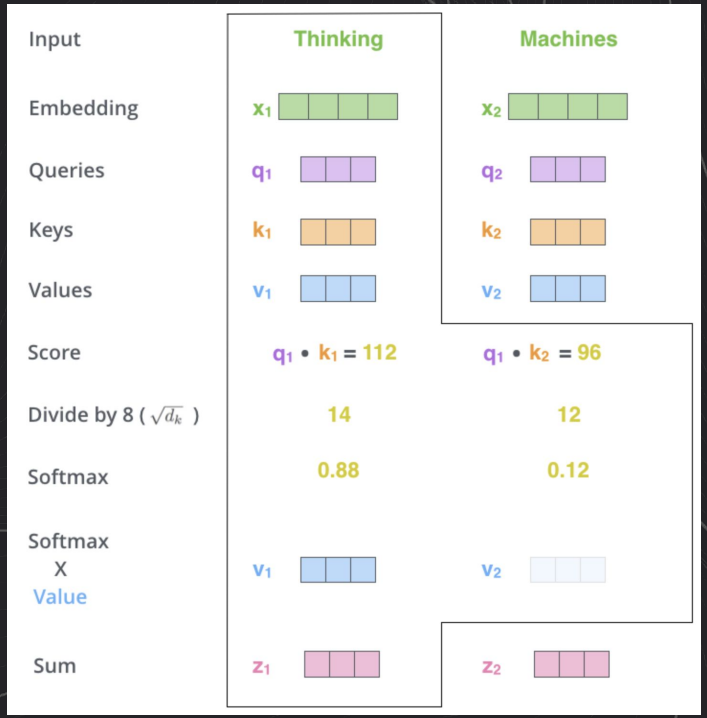
Self-Attention in detail

Step 3 and 4. Divide the scores by the square root of the dimension of the key vectors. Then pass the result through a softmax operation. Softmax normalizes the scores so they're all positive and add up to 1.



Self-Attention in detail

Step 5 and 6. Multiply each value vector by the softmax score. Then sum up the weighted value vectors.



Terminology

- Sequence to sequence
- Neural Machine Translation (NMT)
- Encoder
- Decoder
- Language modeling
- Conditional Language modeling
- Greedy decoding
- Exhaustive search decoding
- Beam search
- BLEU
- Attention
- softmax
- Self-attention