CSCI 1470

Eric Ewing

Monday, 3/10/25

Deep Learning

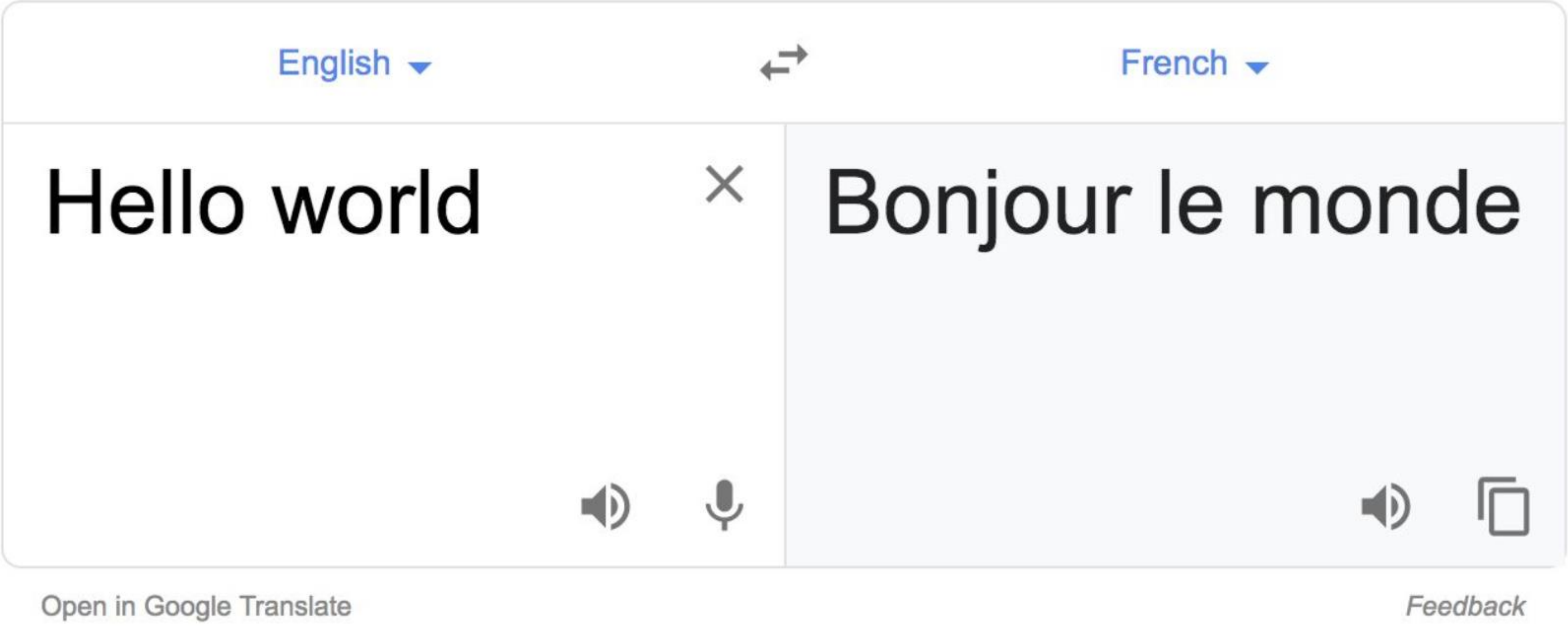
Day 21: Seq2Seq and Attention

Logistics

- Submit your form for the final project, even if you haven’t formed a group. You can indicate a general preference for project area.
- Weekly Quiz is up
- Office hours today will only be from 3-4pm (not 3-5pm)
  - Send me an email if you were planning to come in after 4 and we’ll work something out.

Machine Translation

Software that translates one language to another



Why is this an interesting problem to solve?

- **Complex:** languages evolve rapidly and don’t have a clear and well-defined structure
  - Example of language change: “awful” originally meant “full of awe”, but is now strictly negative

- **Important**: billions per year spent on translation services
  - >CA$2.4 billion spent per year by Canadian government
  - >£100 million spent per year by UK government

Parallel Corpora

- We need pairs of equivalent sentences in two languages, called ***parallel corpora***

Canadian Hansards

- Hansards are transcripts of parliamentary debates
- Canada’s official languages are English and French, so everything said in parliament is transcribed in both languages

Canadian Hansards: Examples

English	French
What a past to celebrate.	Nous avons un beau passé à célébrer.
We are about to embark on a new era in health research in this country.	Le Canada est sur le point d'entrer dans une nouvelle ère en matière de recherche sur la santé.

Canadian Hansards

- We can use this as a dataset for MT!
- Not perfect:
  - *Translations aren't literal*: in the example, "this country" is translated to "Le Canada"
  - *Biased in style*: not everyone speaks like politicians in parliamentary debate
  - *Biased in content*: some topics are never discussed in parliament

Other parallel corpora

- Europarl, a parallel corpus of 21 languages used in the European Parliament
- EUR-Lex, a parallel corpus of 24 languages used in EU law and public documents
- Japanese-English Bilingual Corpus of Wikipedia's Kyoto Articles

Problems with parallel corpora

Any questions?

- Expensive to produce
- Tend to be biased towards particular types of text – e.g. government documents containing formal language
- Translations aren't necessarily literal - e.g. "this country" -> "Le Canada"
- Parallel corpora are necessary, **but never perfect**

LM approach

- Language modelling works on a word-by-word basis, taking only previous words as input

$$P(w_{t,i}) = P\left(w_{t,i} \mid w_{s,i-1}, w_{s,i-2}, \dots, w_{s,0}\right)$$

- Where $w_{t,i}$ is the i^{th} word in the target sentence, and $w_{s,i}$ is the i^{th} word in the source sentence

Will it work for MT task?

Why our LM approach doesn't work for MT

- Language modelling works on a word-by-word basis, taking only previous words as input

$$P(w_{t,i}) = P(w_{t,i} \mid w_{s,i-1}, w_{s,i-2}, \ldots, w_{s,0})$$

- Where $w_{t,i}$ is the i^{th} word in the target sentence, and $w_{s,i}$ is the i^{th} word in the source sentence
- However, **it is not a given that the information we need comes in the preceding words**
- The order and length of the source and target sentences are not necessarily equal

Example from Hansards

- For example, take the first entry in Hansard's:

edited hansard number 1

hansard révisé numéro 1

What should we do?

Further examples

French: “Londres me manque”

Naive translation: “London I miss”

Correct translation: “I miss London”

French: “Je viens de partir”

Naive translation: “I come of to go”

Correct translation: “I just left”

Sequence to Sequence (seq2seq)

Thus, we cannot simply use the previous words – we need to ***summarize the source sentence first***

This is called sequence to ***sequence to sequence learning***, or ***seq2seq***

Sequence to Sequence (seq2seq)

- Instead of:

$$P(w_{i,t}) = P(w_{i,t} \mid w_{i-1,s}, w_{i-2,s}, \dots, w_{0,s})$$

Let's do:

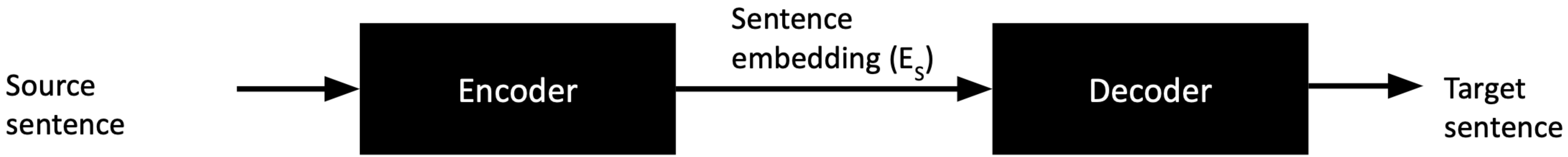
$$P(w_{i,t}) = P(w_{i,t} \mid E_S, w_{i-1,t}, w_{i-2,t}, \dots, w_{0,t})$$

Where E_S is a summary, or ***embedding***, of the sentence taken from the source language, and w_i is the i^{th} word of the sentence in the target language

What will the neural net look like?

Any ideas?

What will the neural net look like?



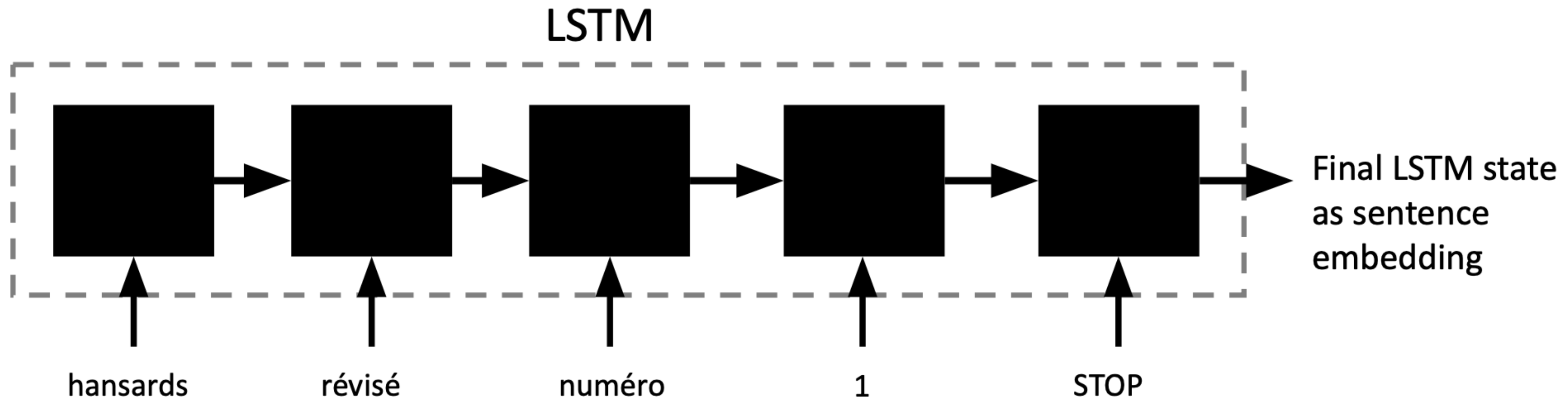
Origin of the encoder/decoder terminology: information theory

- The encoder "compresses" the source sentence into a compact "code"
- The decoder recovers the sentence (but in the target language) from this code

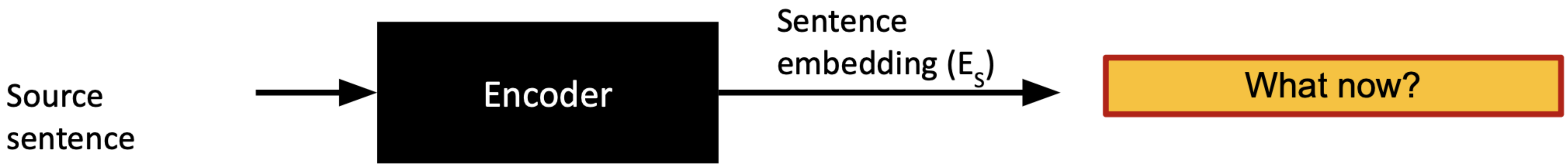
Encoder

- To generate the sentence embedding, we need an encoder
- Use an LSTM
- Feed in the source sentence
- Take the final LSTM state as the sentence embedding
- This will be a ***language-agnostic*** representation of the sentence
    - i.e. it will represent the *meaning* of the sentence without being tied to any particular language

Encoder architecture

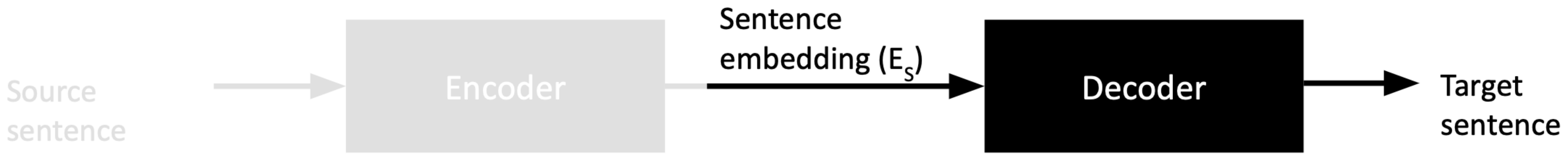


What will the neural net look like?



What will the neural net look like?

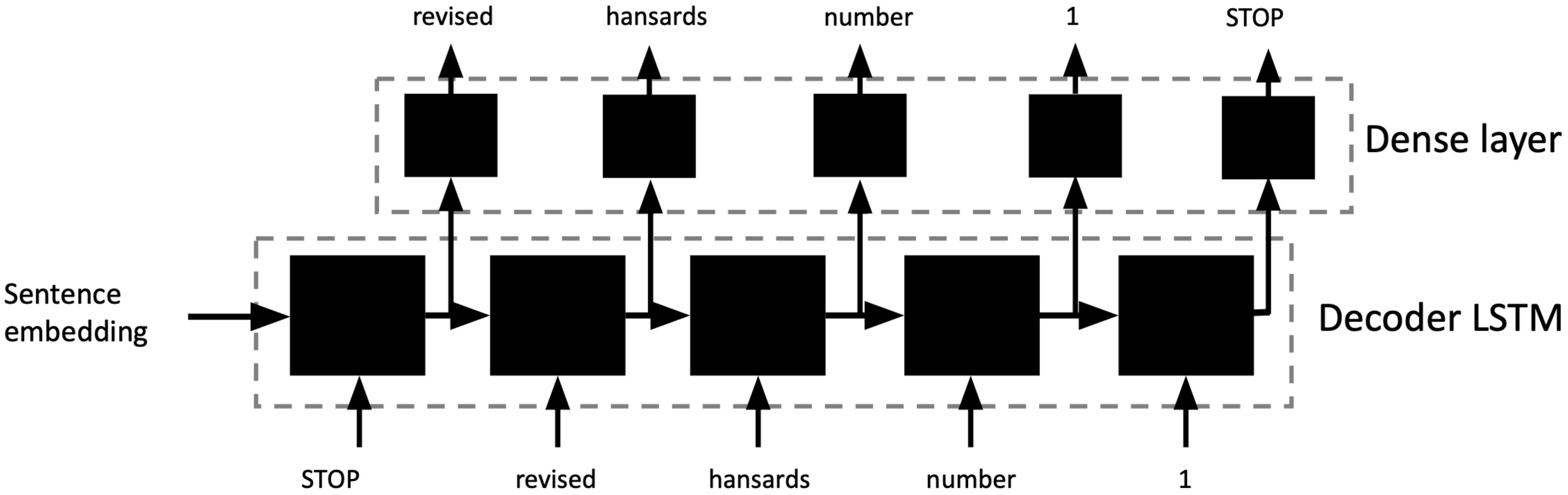
Any ideas?



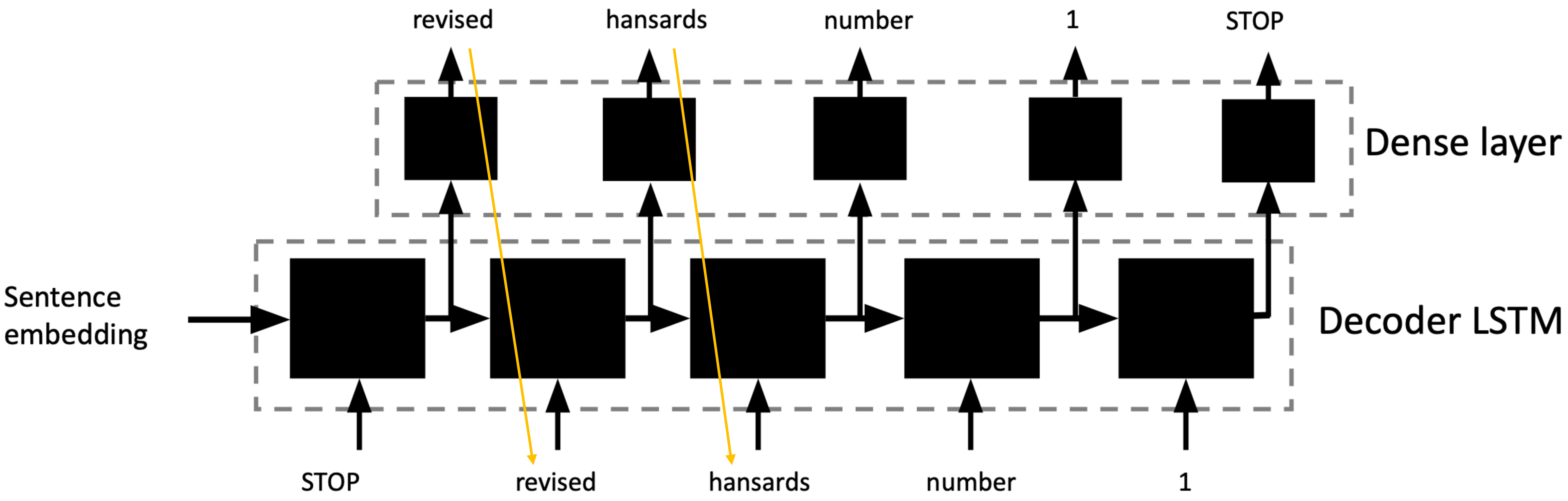
Decoder

- We now have a sentence embedding representing the meaning of the source sentence
- Now, let’s generate a sentence in the target language with the same meaning
- Use an LSTM again, **with the sentence embedding** as its initial hidden state
- The rest is just like language modeling:
  - Input to the LSTM is the previous word from the target sentence
  - Take each LSTM output and put it through a fully connected layer
  - Softmax to convert to probability distribution over next word in target language

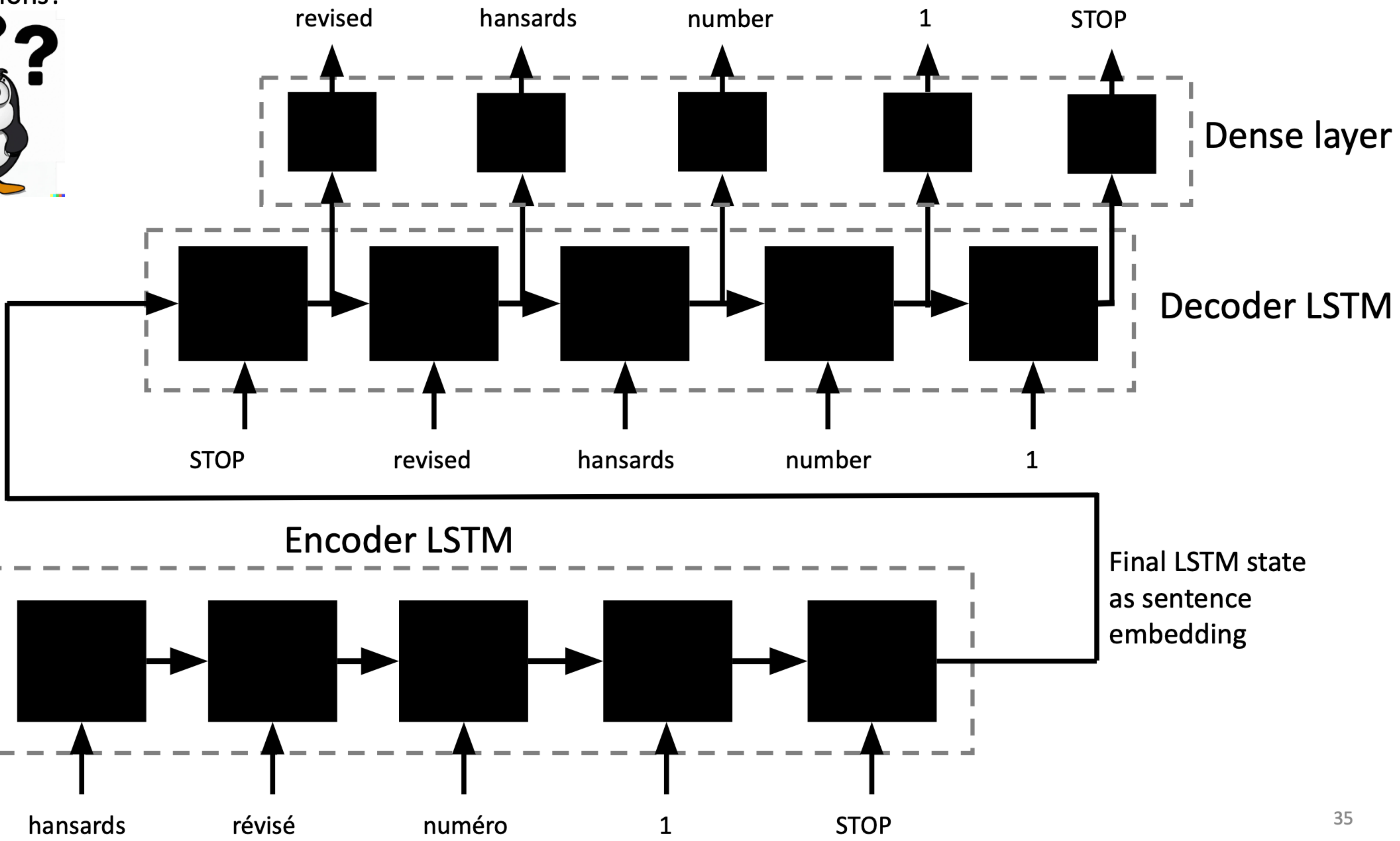
Decoder architecture



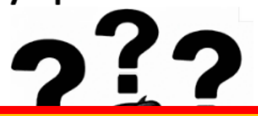
Decoder architecture



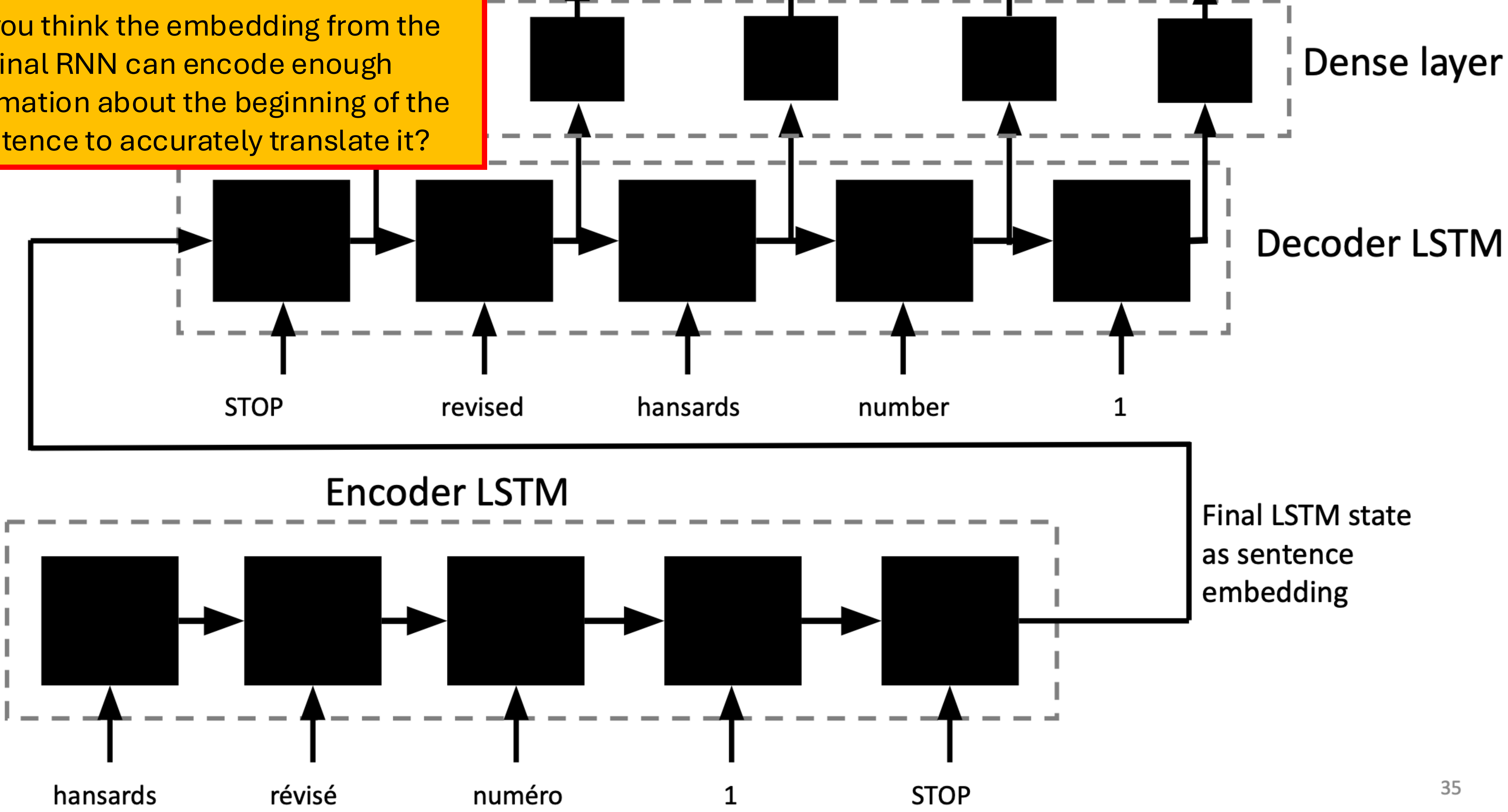
Any questions?



Any questions?

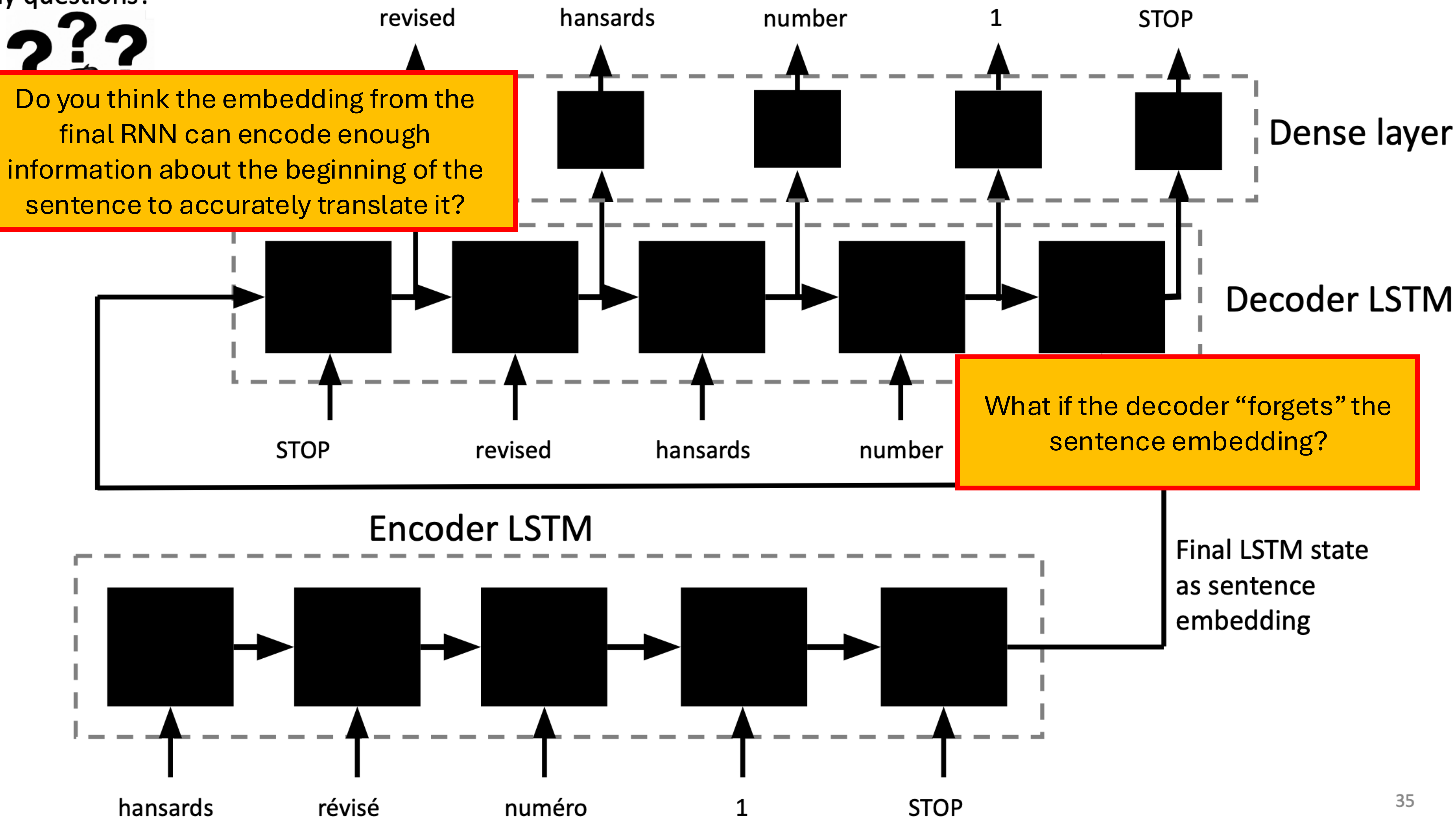


Do you think the embedding from the final RNN can encode enough information about the beginning of the sentence to accurately translate it?



Any questions?

Do you think the embedding from the final RNN can encode enough information about the beginning of the sentence to accurately translate it?

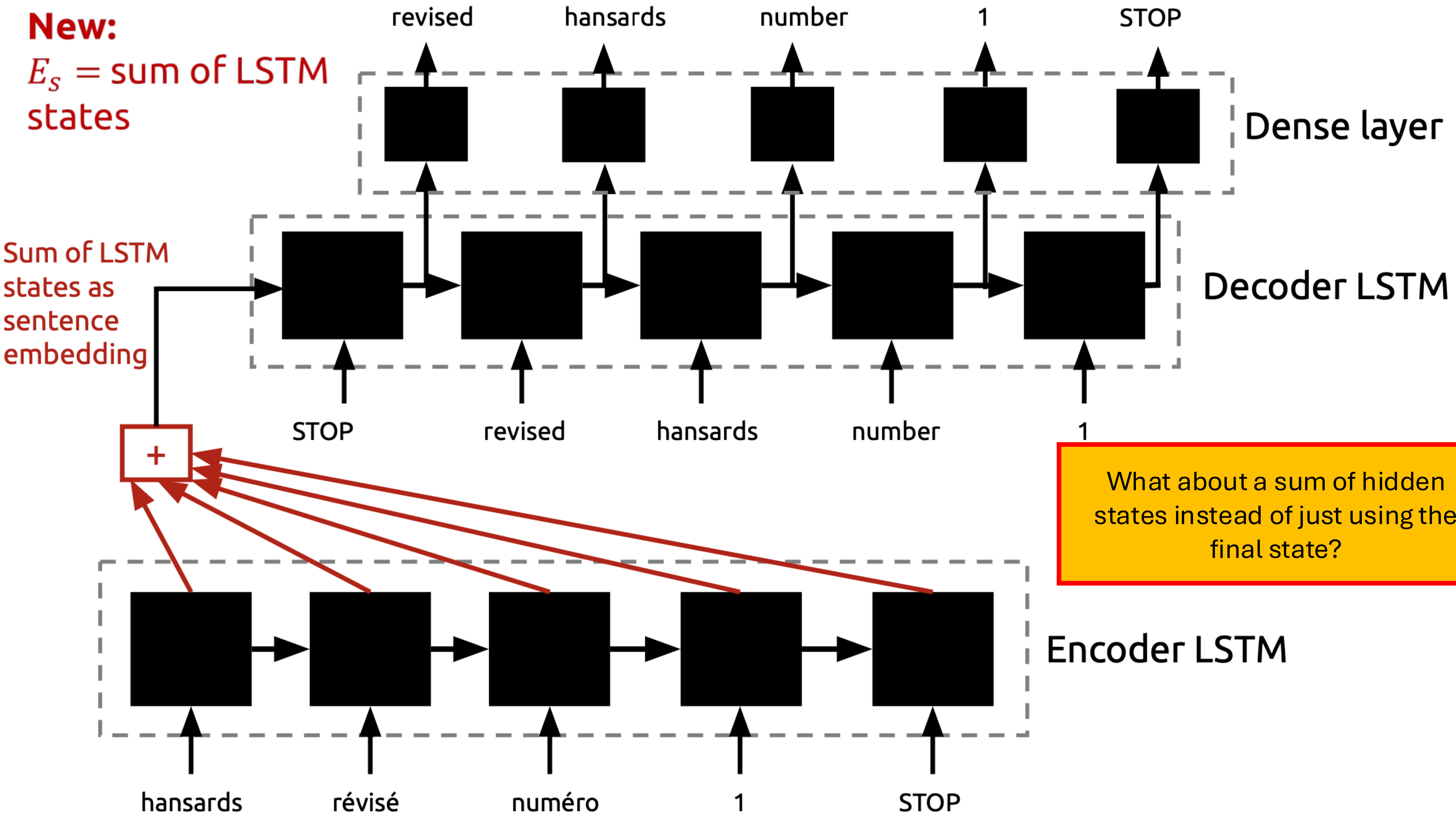


What if the decoder "forgets" the sentence embedding?

Issues with RNNs for Seq2Seq

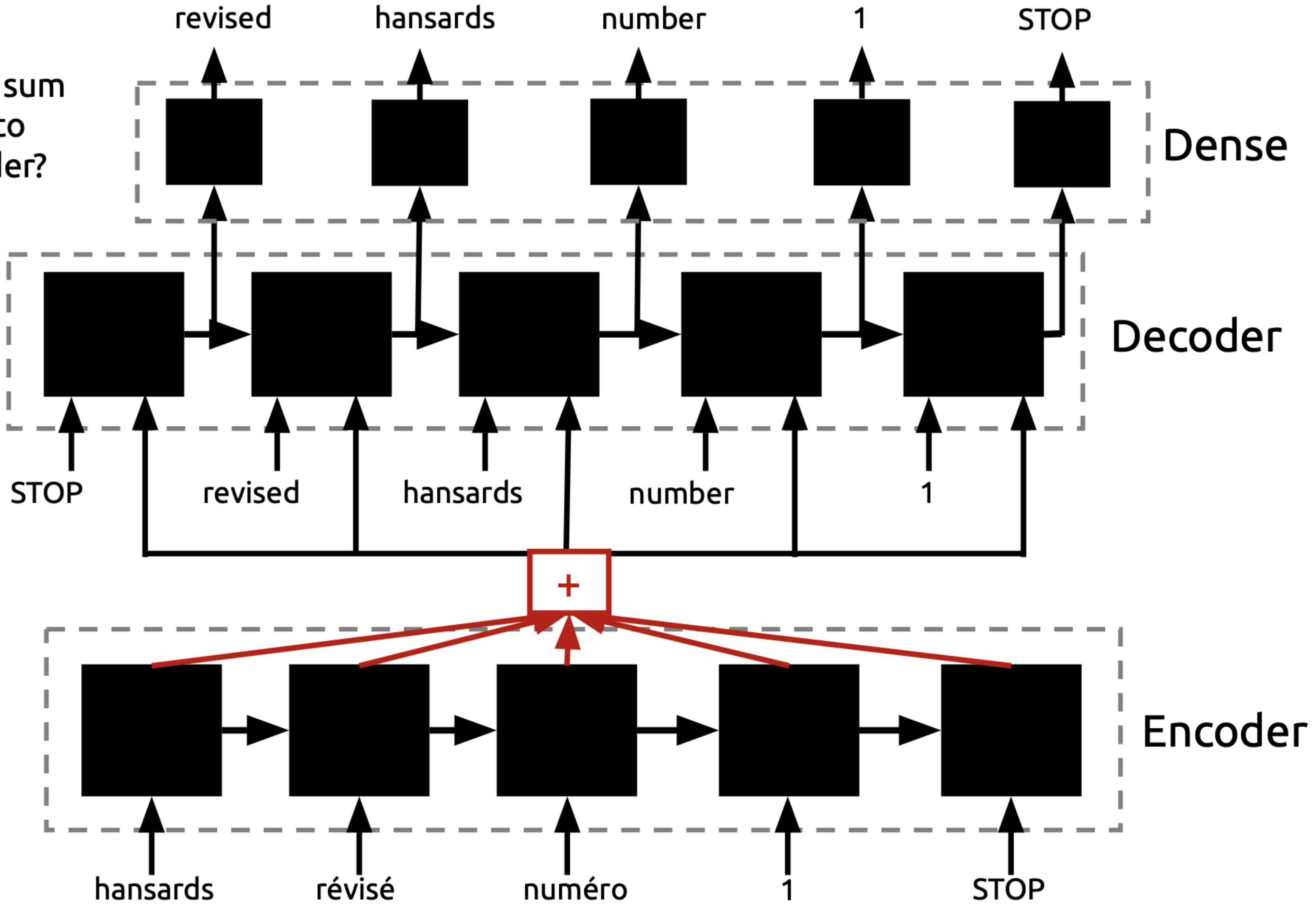
- RNNs may “forget” the beginning of the sentence in the encoder
- RNNs (even LSTMs) may “forget” the embedding in the decoder

New:
E_s = sum of LSTM states



What about a sum of hidden states instead of just using the final state?

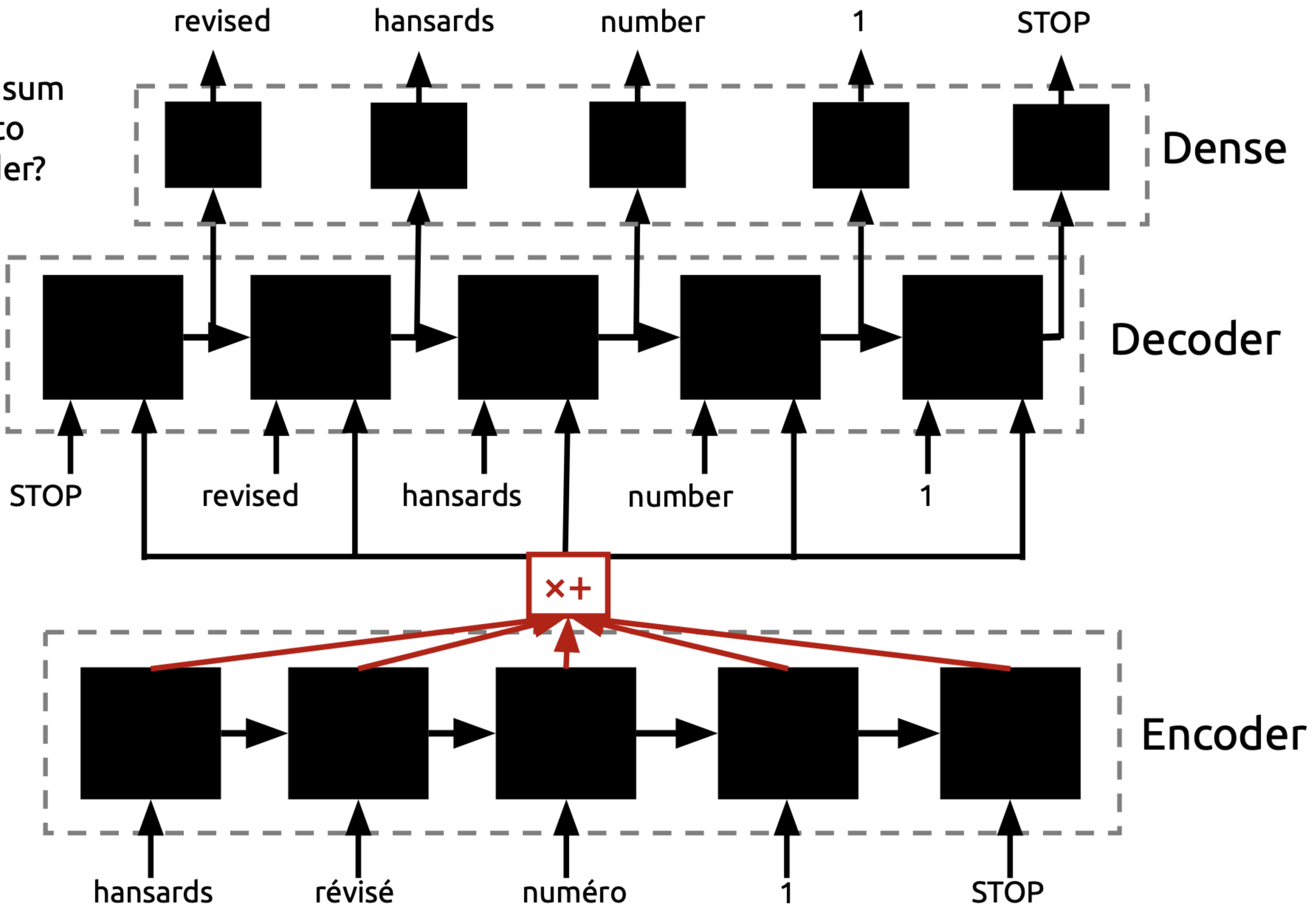
What if we passed the sum of our encoder states to ***every cell*** in the decoder?



What if we passed the sum of our encoder states to ***every cell*** in the decoder?

What if the sum was a ***weighted sum*** instead?

- Idea: different words in the input carry different importance

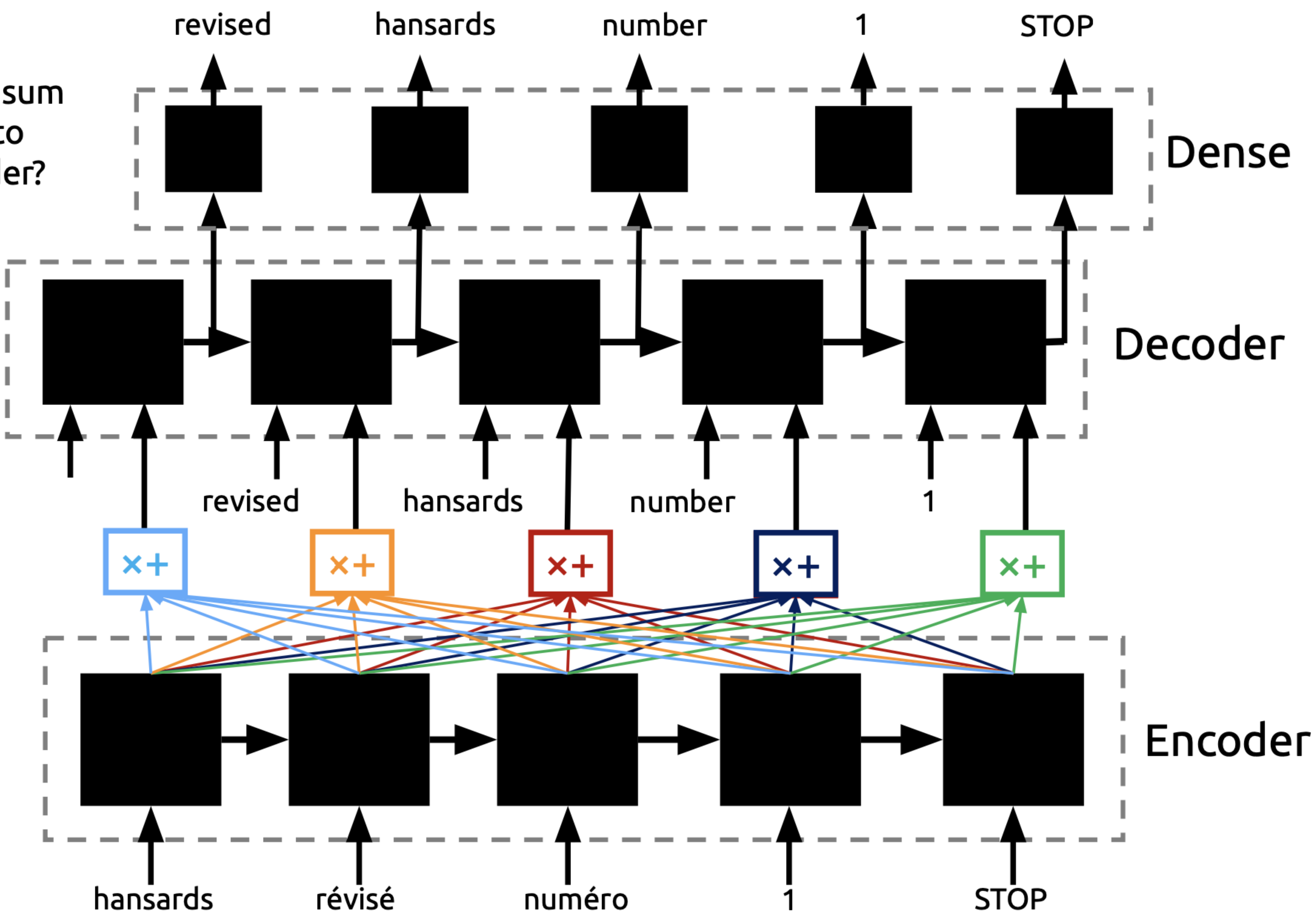


What if we passed the sum of our encoder states to ***every cell*** in the decoder?

What if the sum was a ***weighted sum*** instead?

What if each decoder cell received a ***different*** weighted sum?

- Idea: different words in the input carry different importance *for each word in the output*



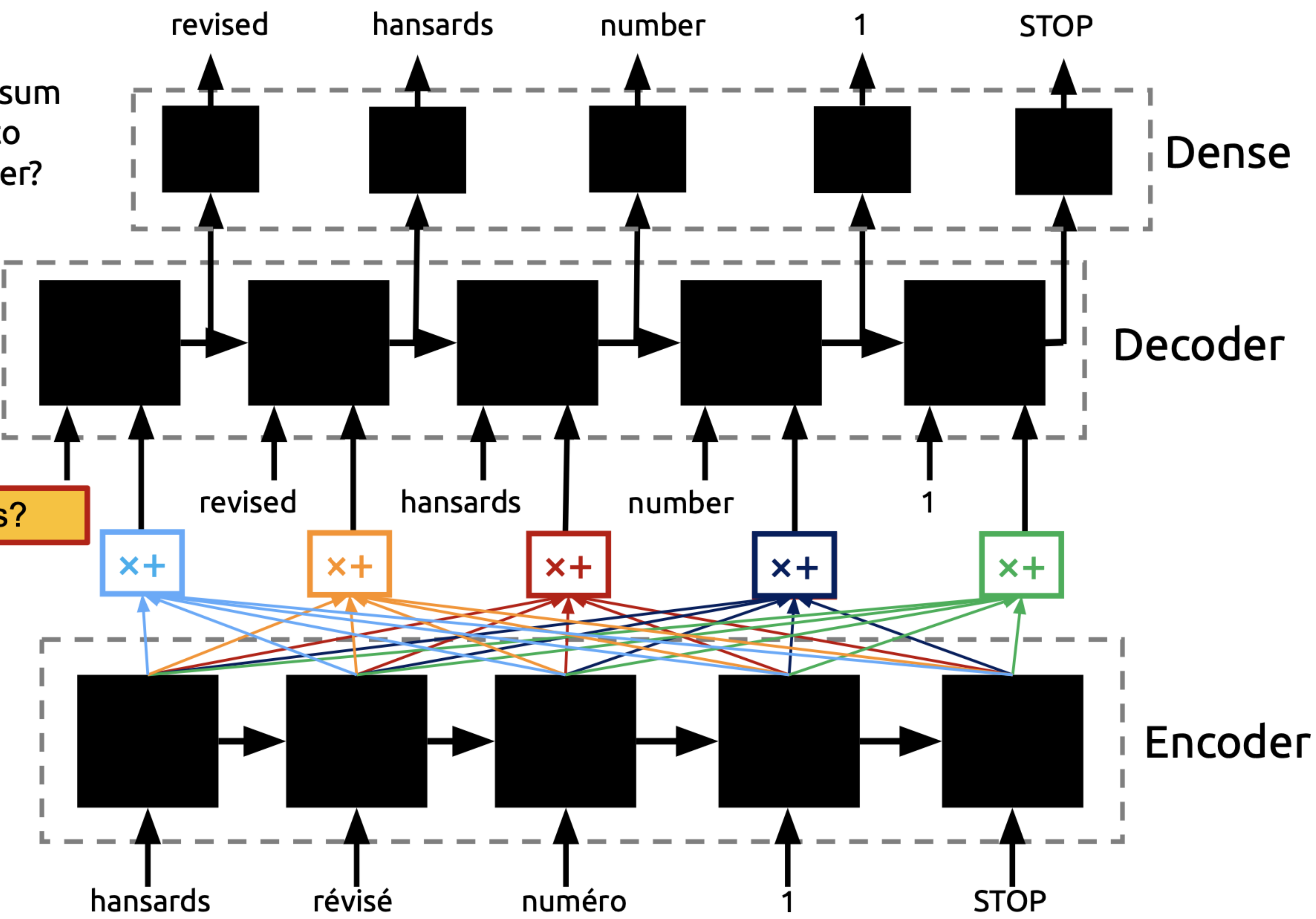
What if we passed the sum of our encoder states to ***every cell*** in the decoder?

What if the sum was a ***weighted sum*** instead?

How do we achieve this?

What if each decoder cell received a ***different*** weighted sum?

- Idea: different words in the input carry different importance *for each word in the output*



“Attention”

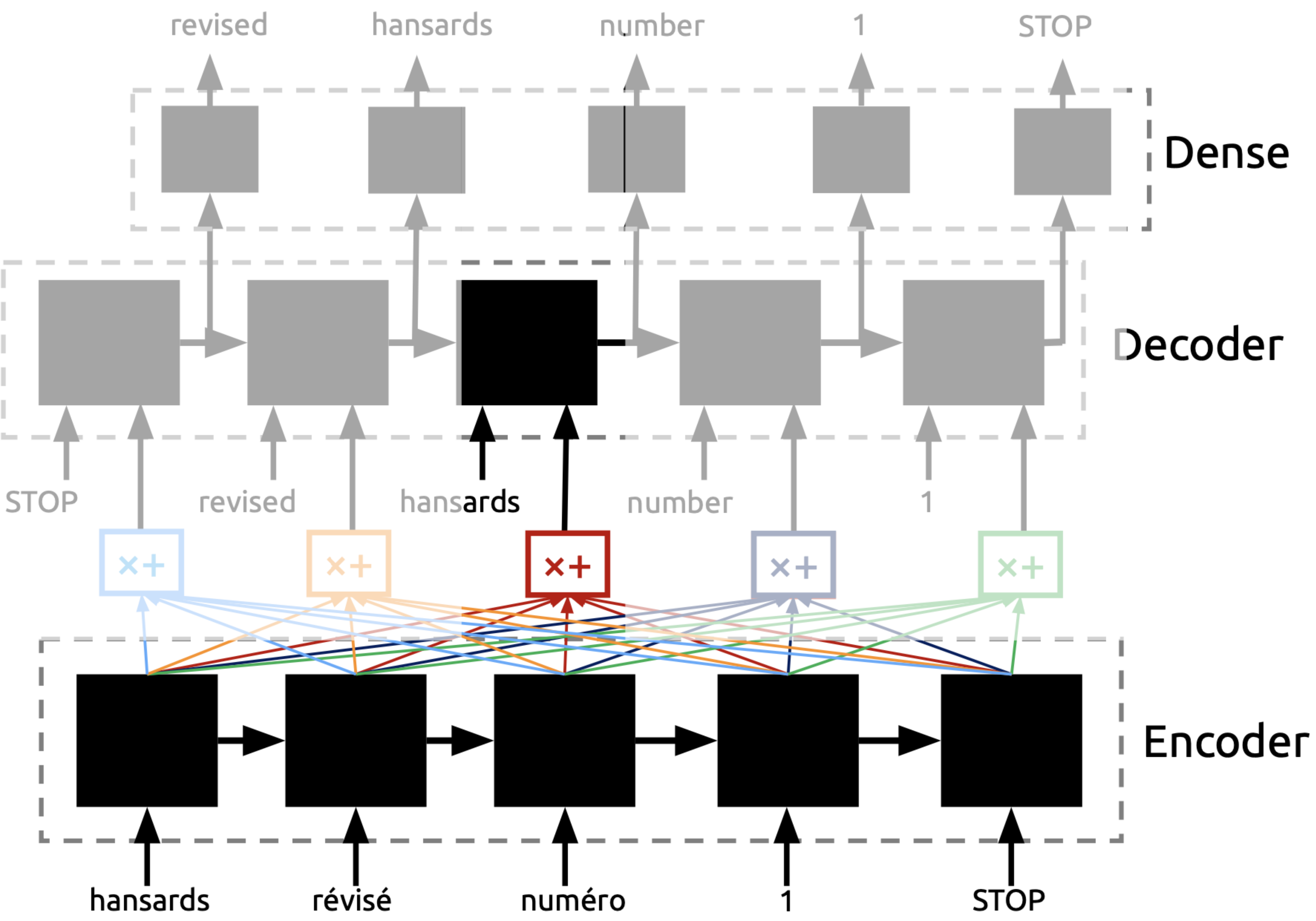
This idea of passing each cell of the decoder a weighted sum of the encoder states is called ***attention.***

- Different words in the output “pay attention” to different words in the input

“Attention” - intuition

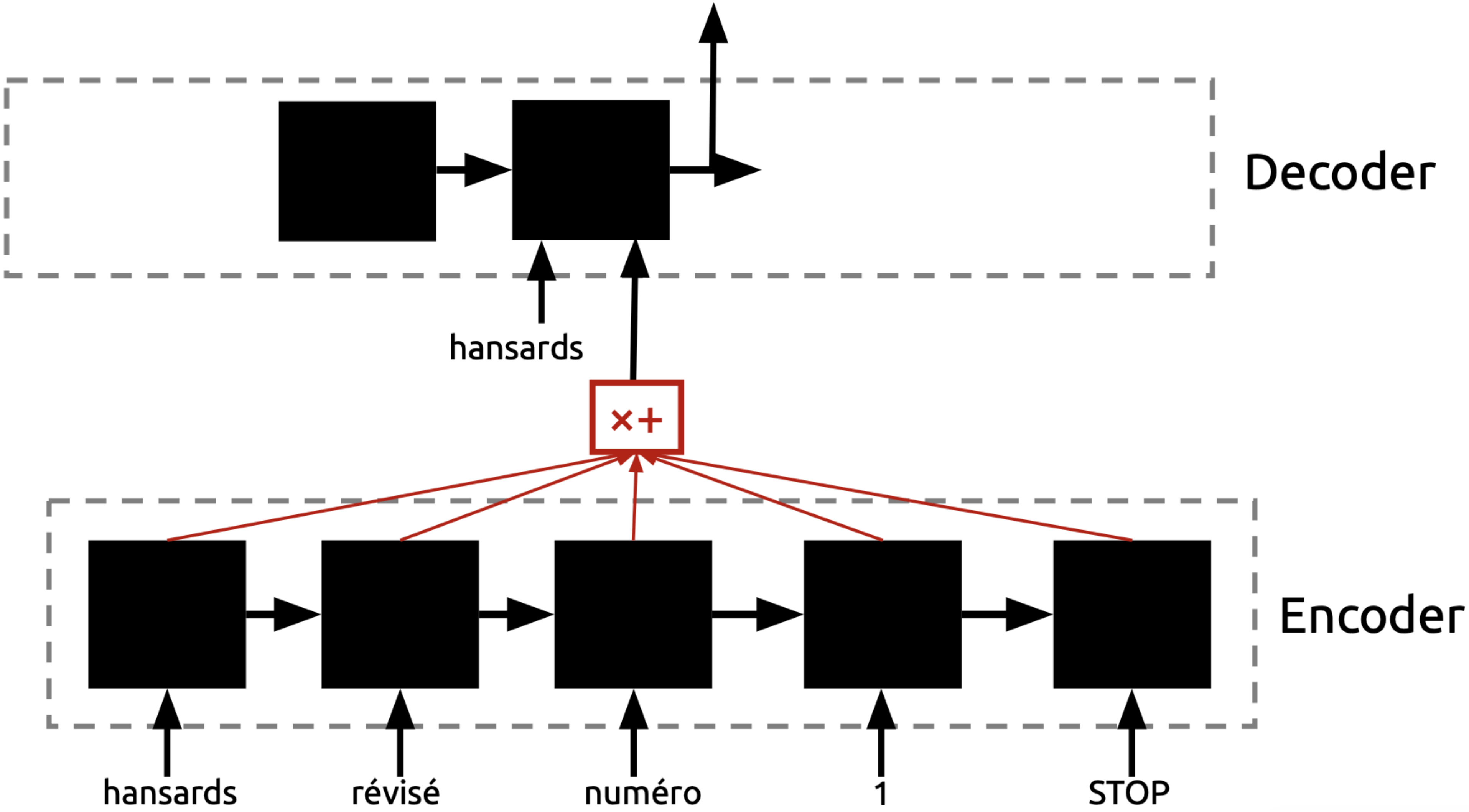
“Park”

How about we let model learn what is relevant for a particular output

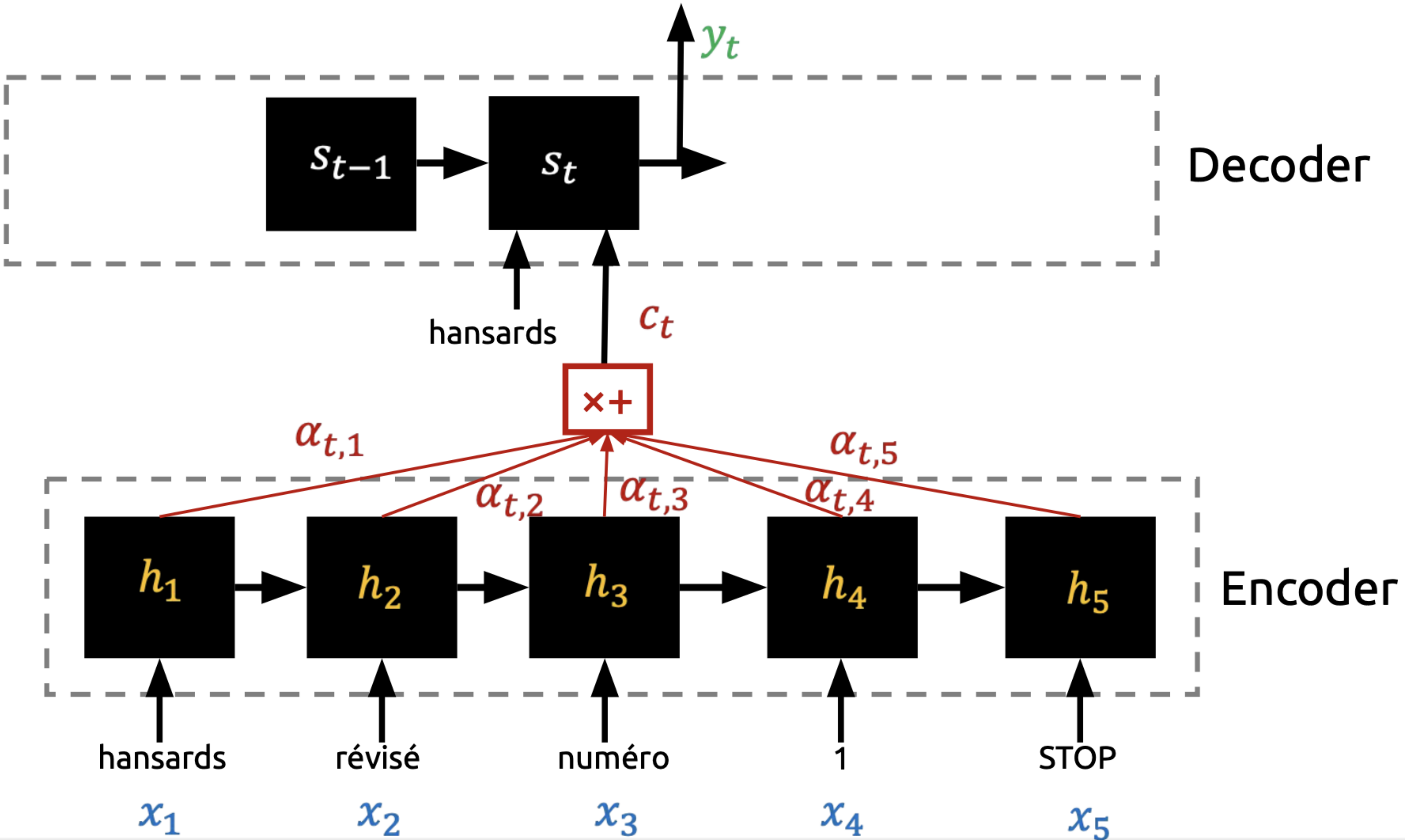


revised
hansards
number
1
STOP
Dense
Decoder
STOP
revised
hansards
number
1
×+
×+
×+
×+
×+
Encoder
hansards
révisé
numéro
1
STOP

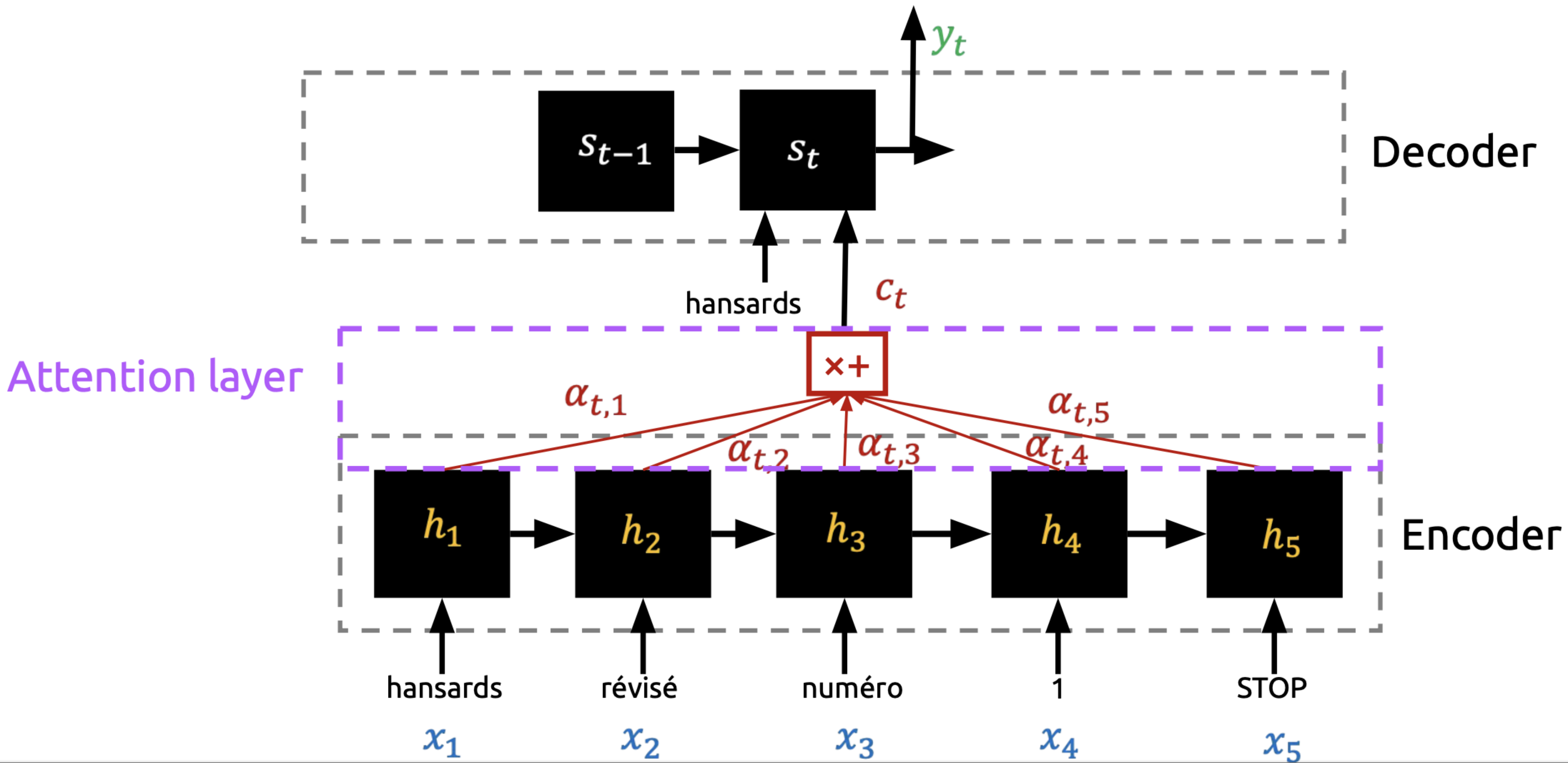
Attention - implementation



Attention - implementation



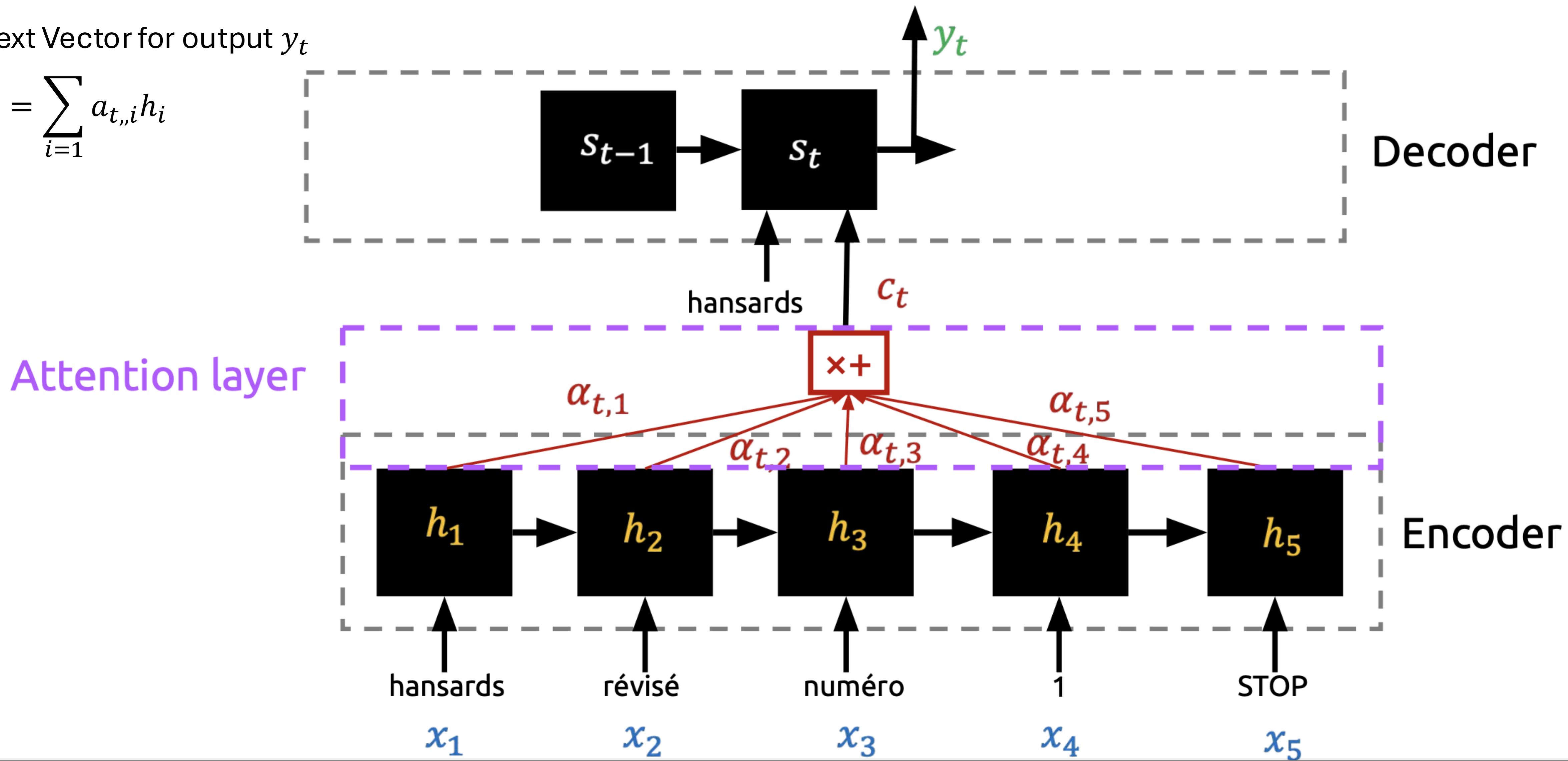
Attention - implementation



Attention - implementation

Context Vector for output y_t

$$c_t = \sum_{i=1} a_{t,,i} h_i$$



Attention - implementation

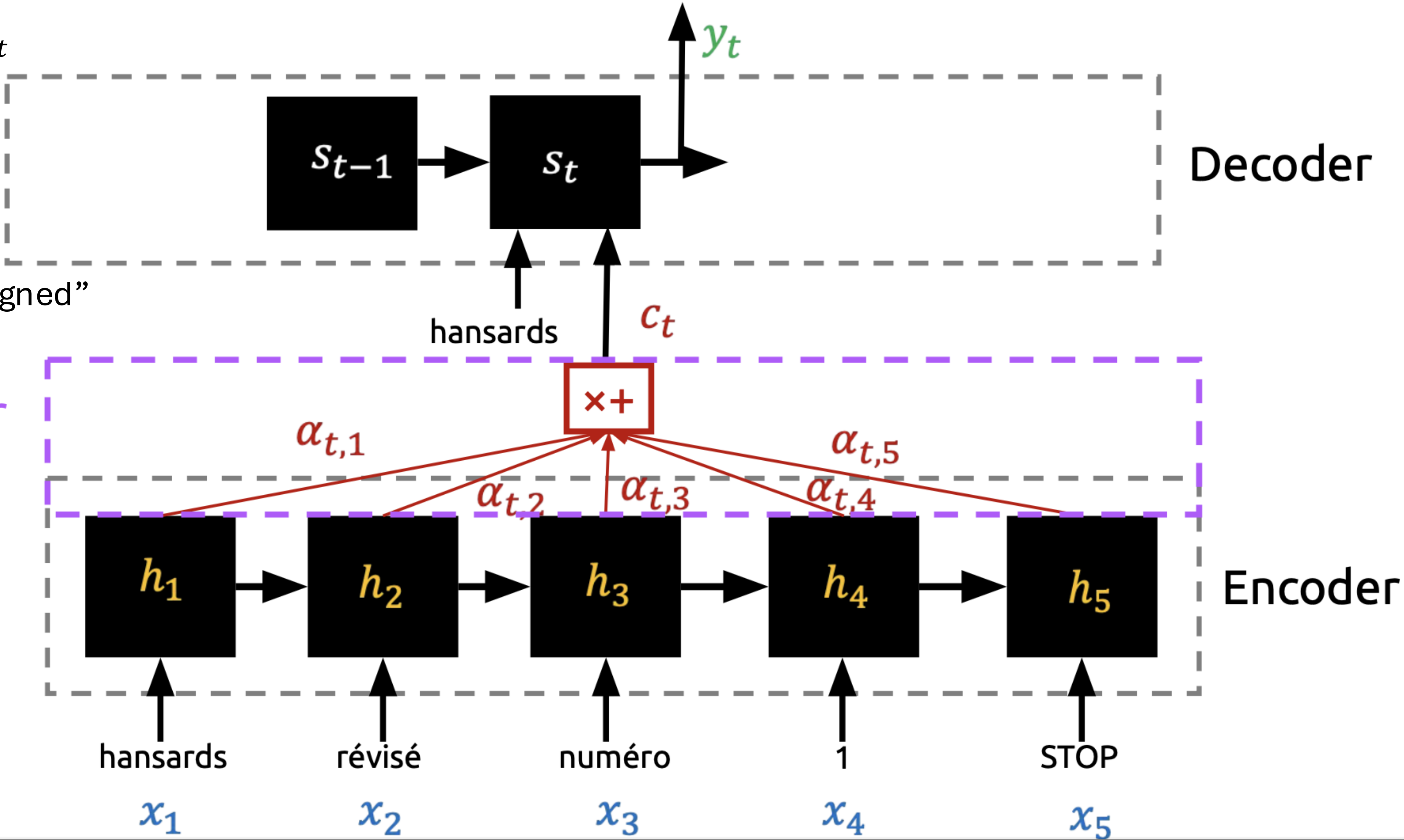
Context Vector for output y_t

$$c_t = \sum_{i=1} a_{t,,i} h_i$$

How well two words are “aligned”

$$a_{t,i} = align(x_t, y_t)$$

Attention layer



Attention - implementation

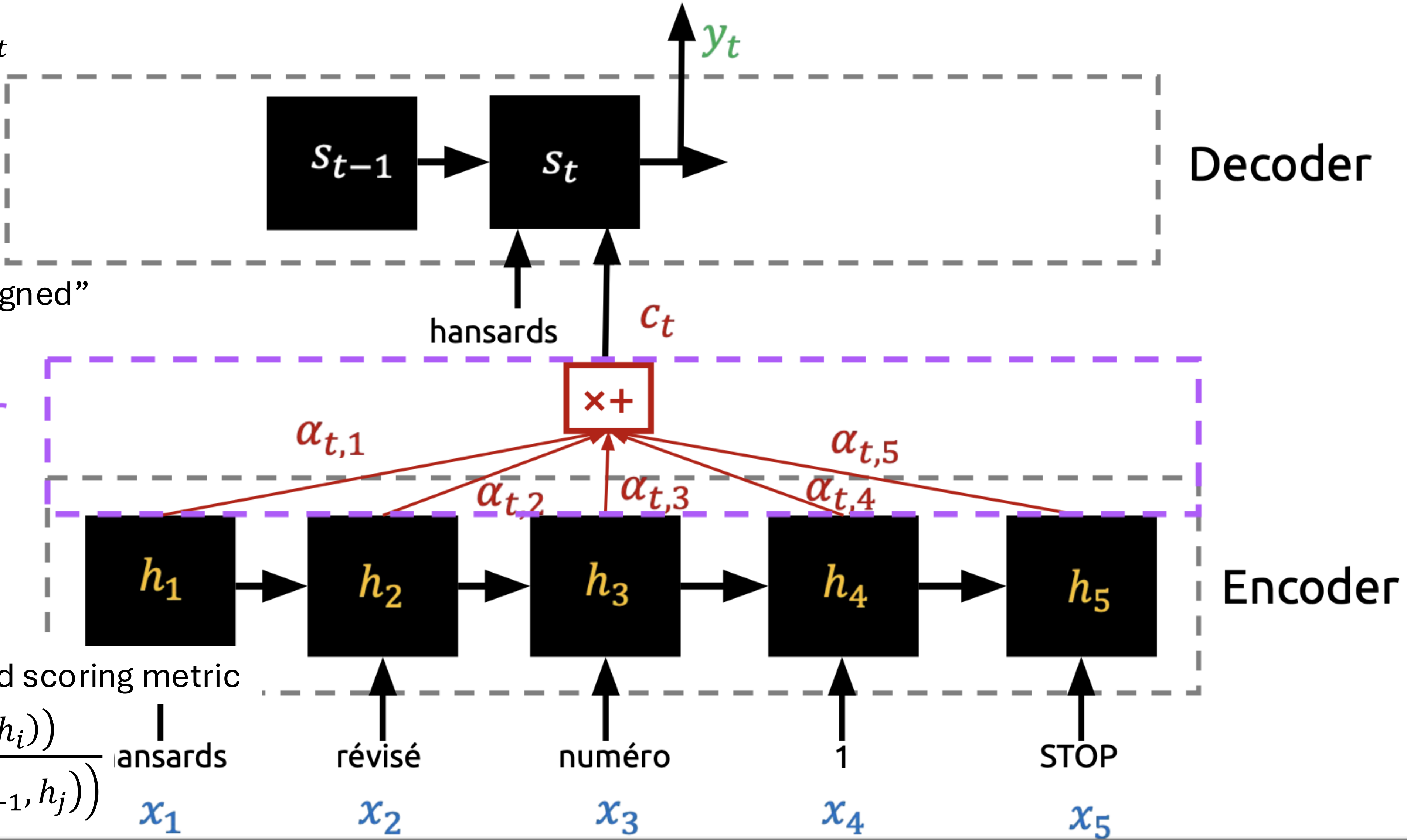
Context Vector for output y_t

$$c_t = \sum_{i=1} a_{t,,i} h_i$$

How well two words are “aligned”

$$a_{t,i} = align(x_t, y_t)$$

Attention layer



Softmax of some predefined scoring metric

$$a_{t,i} = \frac{\exp\left(score(s_{t-1}, h_i)\right)}{\sum_{j=1} \exp\left(score(s_{t-1}, h_j)\right)}$$

Attention Alignment Score

- Need to determine how well output word y_t aligns with each input word x_i
- How can we determine the similarity between two words (or at least the vectors that represent them)?

Attention Alignment Score

- Need to determine how well output word y_t aligns with each input word x_i
- How can we determine the similarity between two words (or at least the vectors that represent them)?

$$\text{Cosine Similarity}(h_i, s_{t-1}) = \frac{h_i s_{t-1}}{||h_i|| * ||s_{t-1}||}$$

Attention Alignment Score

- Need to determine how well output word y_t aligns with each input word x_i
- How can we determine the similarity between two words (or at least the vectors that represent them)?

Cosine Similarity$(h_i, s_{t-1}) = \frac{h_i s_{t-1}}{||h_i|| * ||s_{t-1}||}$

Dot product similarity$(h_i, s_{t-1}) = h_i s_{t-1}$

Attention Alignment Score

- Need to determine how well output word y_t aligns with each input word x_i
- How can we determine the similarity between two words (or at least the vectors that represent them)?

Cosine Similarity$(h_i, s_{t-1}) = \frac{h_i s_{t-1}}{||h_i|| * ||s_{t-1}||}$

Dot product similarity$(h_i, s_{t-1}) = h_i s_{t-1}$

Generalized Similarity$(h_i, s_{t-1}) = h_i W_a s_{t-1}$

Learned weight matrix
How much do we care about each part of embedding?

There are many ways to measure similarity...

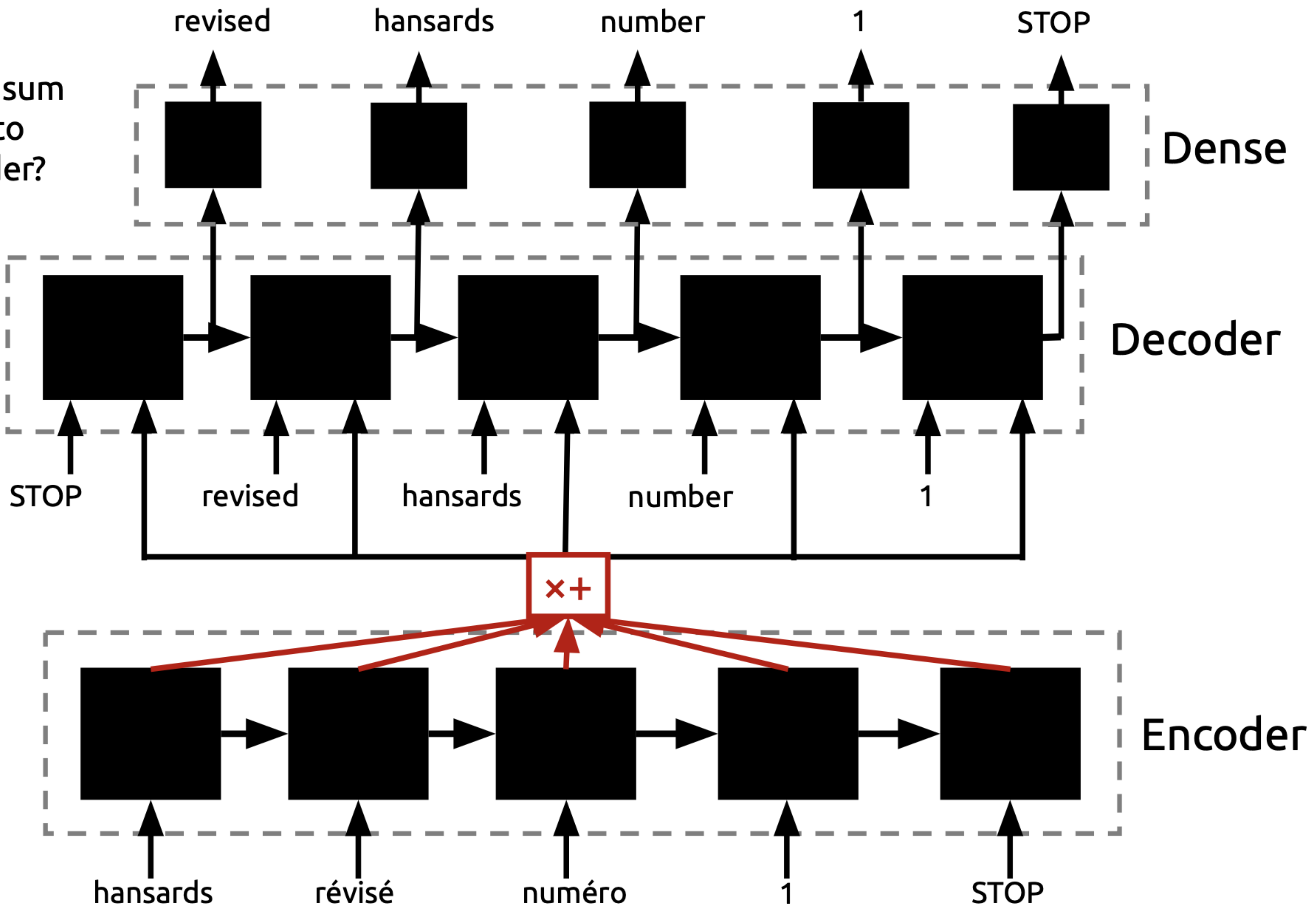
Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

What if we passed the sum of our encoder states to ***every cell*** in the decoder?

What if the sum was a ***weighted sum*** instead?

- Idea: different words in the input carry different importance



Attention Example

We can represent the attention weights as a matrix:

Columns: words in the input

Rows: words in the output

	hansards	révisé	numéro	1	STOP
revised	1/2	1/4	1/4	0	0
hansards	1/4	$\boxed{1/2}$	1/4	0	0
number	0	1/4	1/2	1/4	0
1	0	0	1/4	1/2	1/4
STOP	0	0	1/4	1/4	1/2

$\alpha_{j,i}$: how much 'attention' output word j pays to input word i

What do the values in this particular matrix imply about the attention relationship between input/output words?

Attention Example

Target: “Der Hund <u>bellte</u> mich an.”

Input: “The dog barked at me.”

[0, 1/4, 1/2, 1/4, 0]

We see that when we apply the attention to our inputs, we will pay attention to relatively important words for translation when predicting “bellte”.

Attention is great!

- Attention significantly improves MT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself

Attention is a general deep learning technique

More general definition of attention:

Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

Credit: Stanford CS224n

Recap

Machine Translation is a Seq2Seq Learning Task

Encoder-Decoder Architectures work well for Seq2Seq learning problems

Attention helps solve some of the memory issues that RNNs face for Seq2Seq learning tasks