8. Attention

LING-581-Natural Language Processing 1

Instructor: Hakyung Sung

October 7, 2025

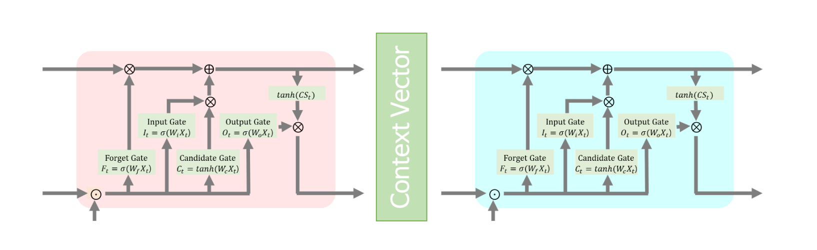
*Acknowledgment: These course slides are based on materials from CS224N @ Stanford University; Dr. Kilho Shin @ Kyocera

Table of contents

Seq2Seq

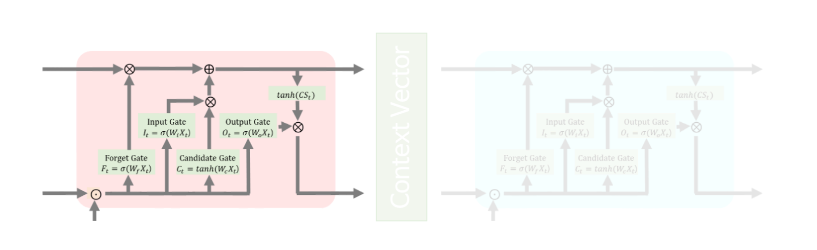
Seq2Seq

The Seq2Seq model is an important architecture widely used in NLP tasks such as <u>machine translation</u>.



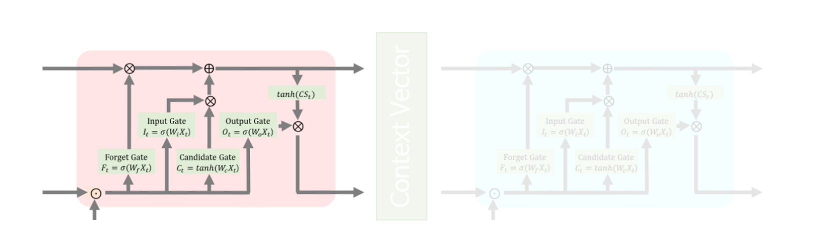
Seq2Seq

In most cases, the basic unit of a Seq2Seq model is an LSTM network.

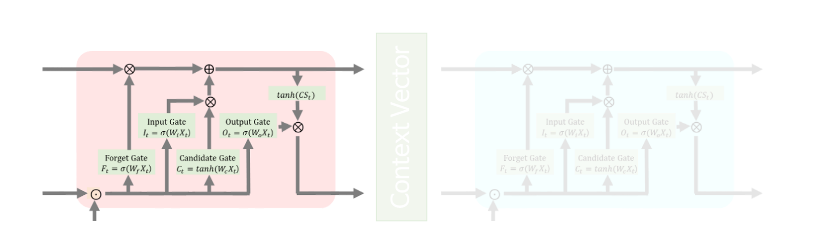


Seq2Seq

In the task of machine translation, LSTM outperformed the standard RNN because...

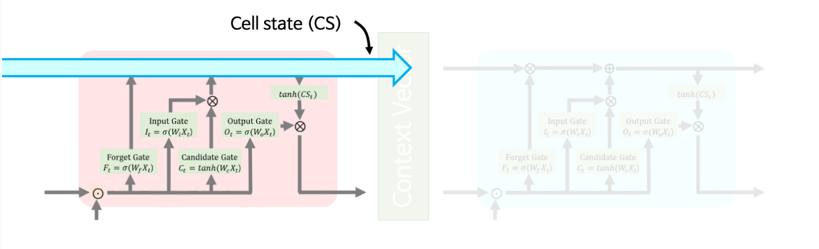


The key reason is that LSTM uses two parallel flows of information.



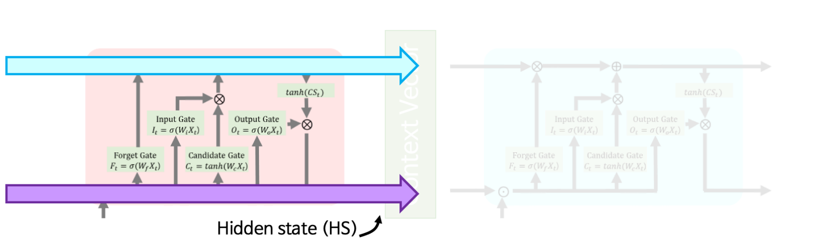
Seq2Seq

The key reason is that LSTM uses two parallel flows of information.



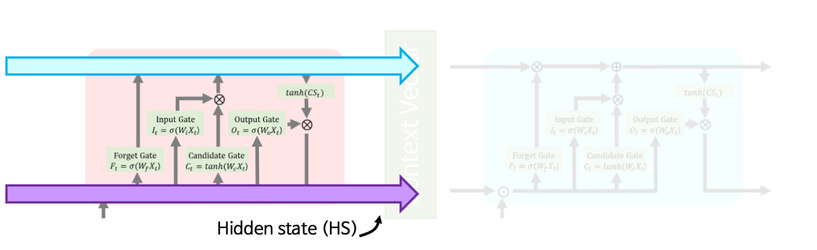
Seq2Seq

The key reason is that LSTM uses two parallel flows of information.



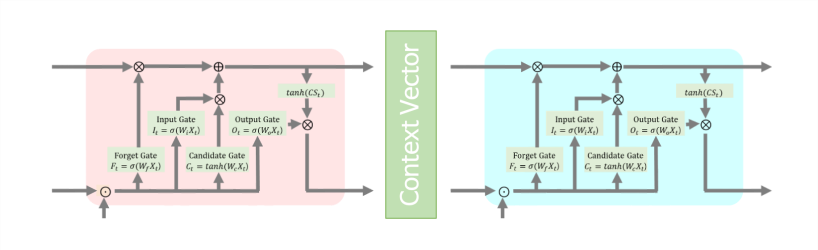
Seq2Seq-LSTM

Therefore, LSTM can overcome long-term dependencies in sentences by using its cell state (CS) and hidden state (HS).



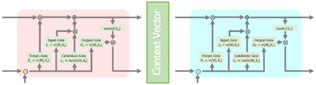
Seq2Seq

The biggest challenge in machine translation is the difference in word order and sentence length between languages. The Seq2Seq model solved this problem in a groundbreaking way.



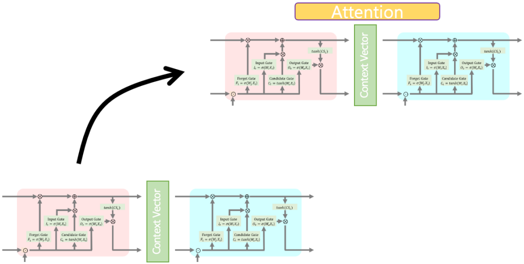
Seq2Seq

Then, Seq2Seq evolved further...



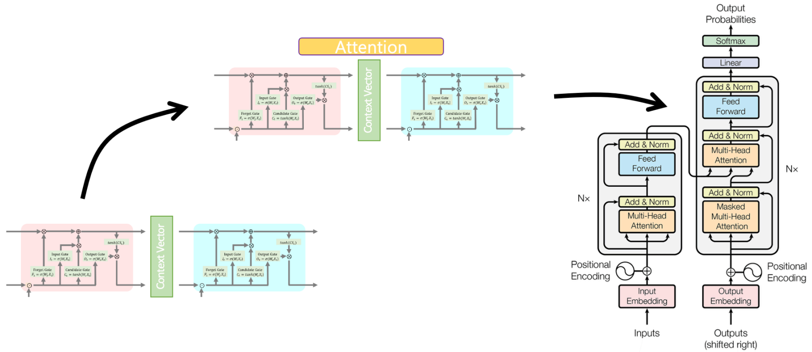
Seq2Seq

…by introducing the **Attention** algorithm,

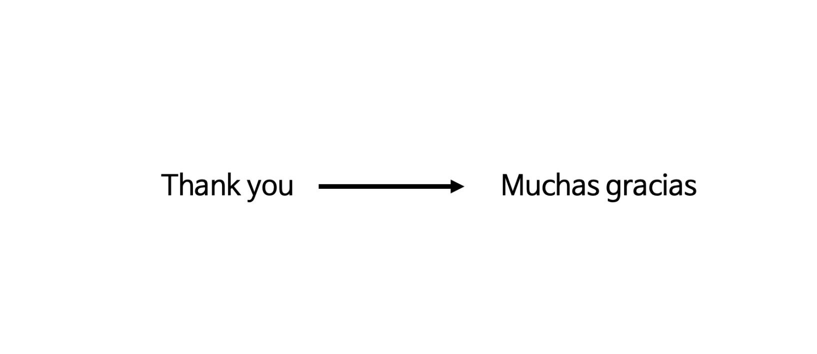


Seq2Seq

…which eventually led to the development of the **Transformer** model.

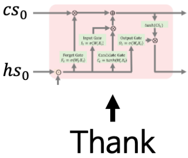


How does the Seq2Seq model work?



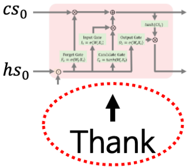
Seq2Seq

First, we input the word “Thank” into the LSTM.



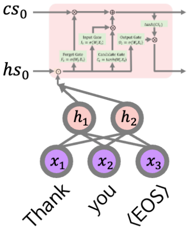
Seq2Seq

To process this word numerically, we apply word2vec embeddings.



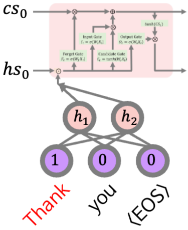
Seq2Seq

*Suppose a simple dictionary composed of three tokens: “Thank,” “you,” and “<EOS>.”



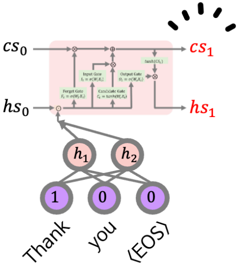
Seq2Seq

We input the first token, “Thank.”



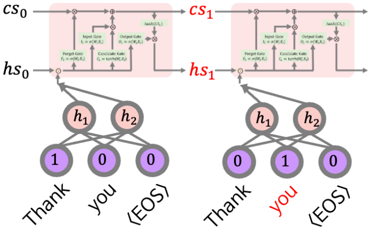
Seq2Seq

Through forward propagation, “Thank” generates cs1 and hs1.



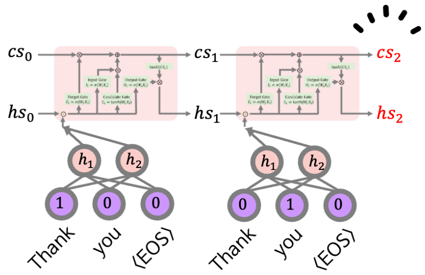
Seq2Seq

Next, we input the word "you."



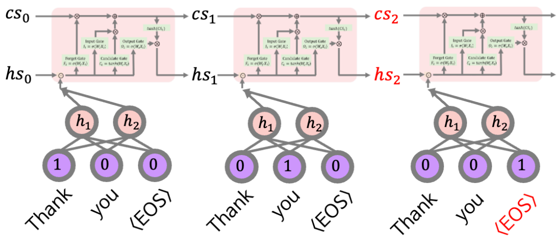
Seq2Seq

The previous cs1 and hs1, together with the word vector for "you," produce new states cs2 and hs2.



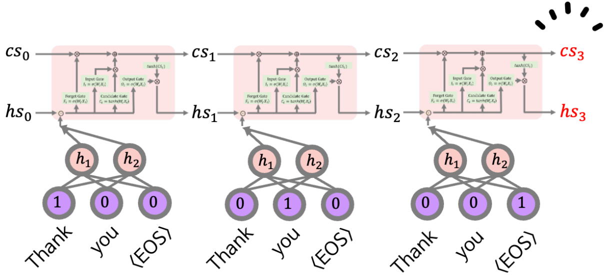
Seq2Seq

Since the English sentence ends here, we input the token “<EOS>” (End of Sentence).



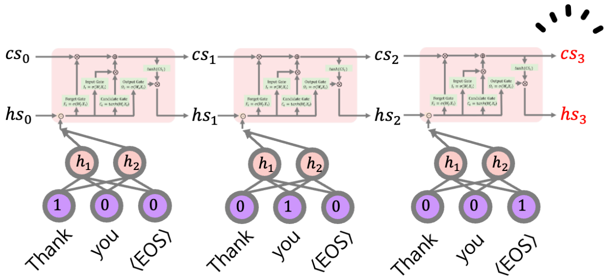
Seq2Seq

“<EOS>” generates cs3 and hs3.



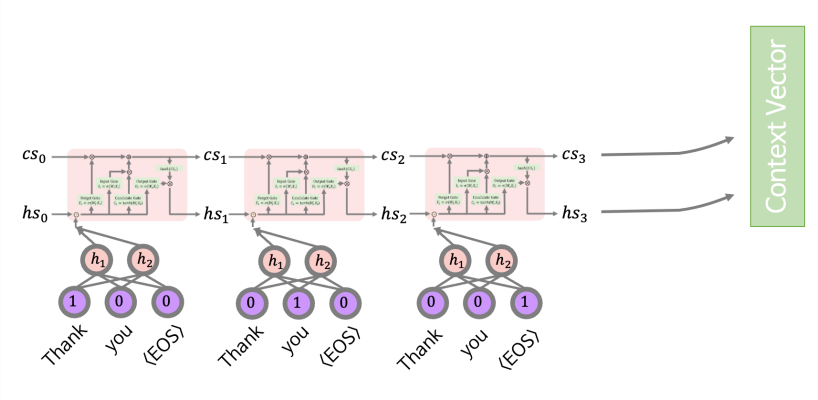
Seq2Seq

These cs3 and hs3 encode the long- and short-term information of all words in the sentence.



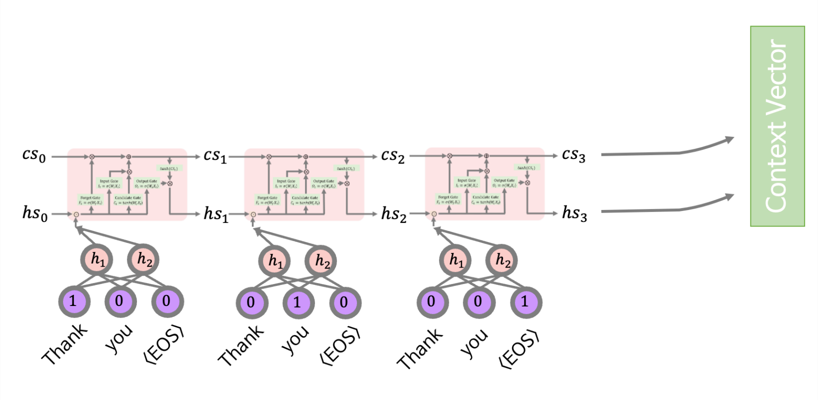
Seq2Seq

In the Seq2Seq model, we combine cs3 and hs3 and call it the **context vector**.



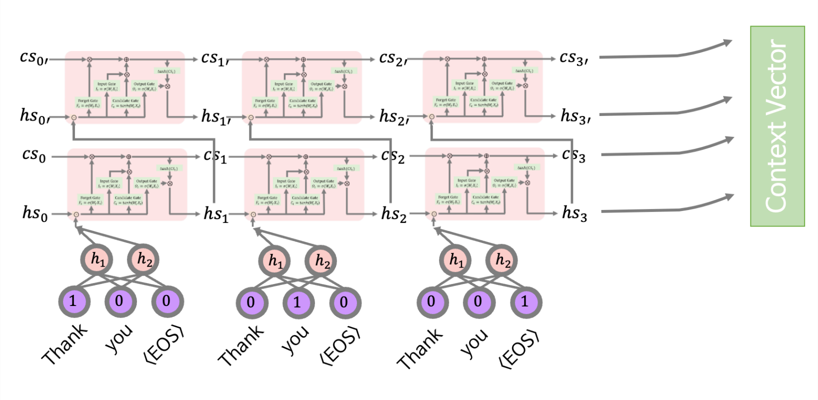
Seq2Seq

An LSTM can have a single layer...



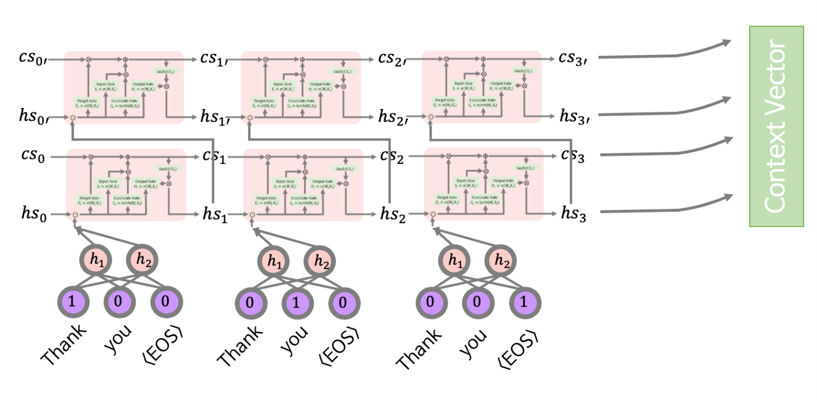
Seq2Seq

...or we can stack two layers to obtain a richer context vector.



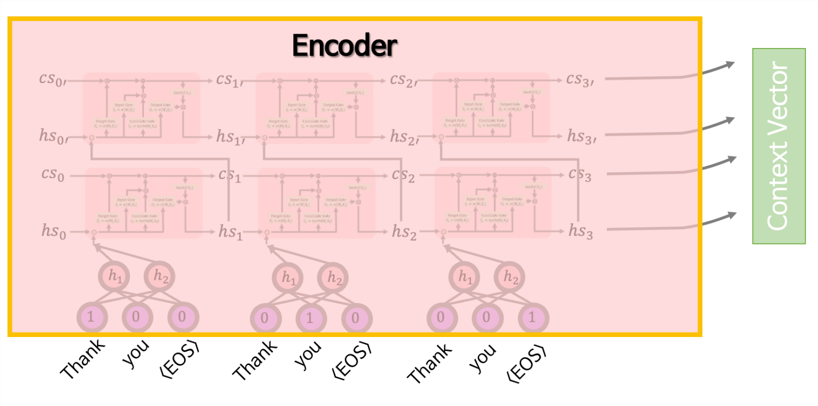
Seq2Seq

In this case, the second LSTM layer has its own independent weights and biases.



Seq2Seq

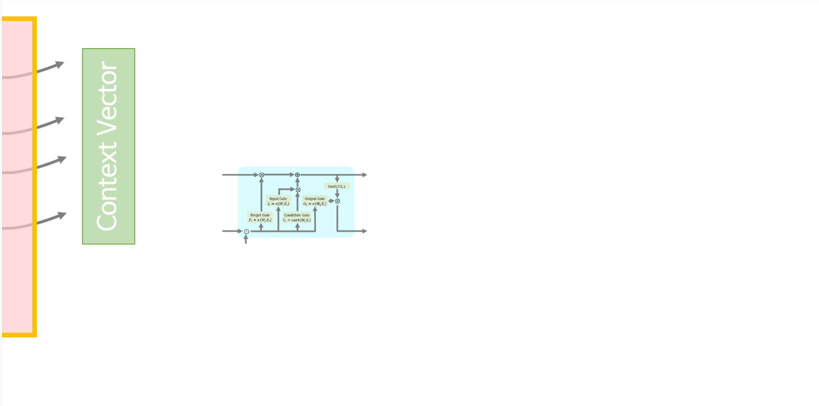
This part is called the **encoder** in Seq2Seq.



Now, using the context vector, the model begins the generation process.

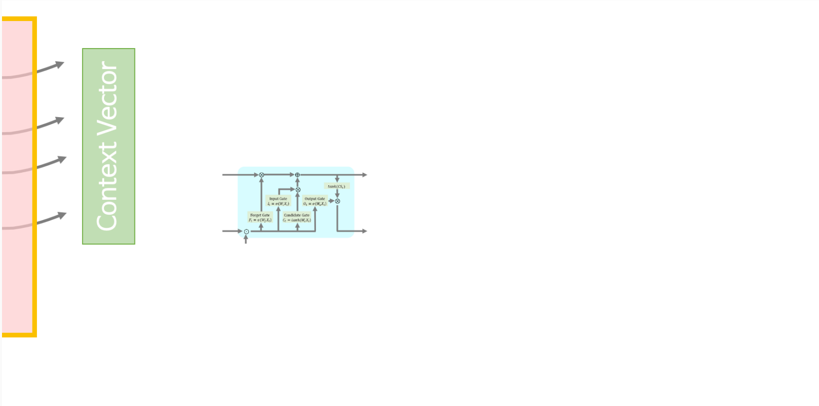
Seq2Seq

The decoder also uses LSTM.



Seq2Seq

This LSTM is completely new; its weights and biases are independent from the encoder's LSTM.

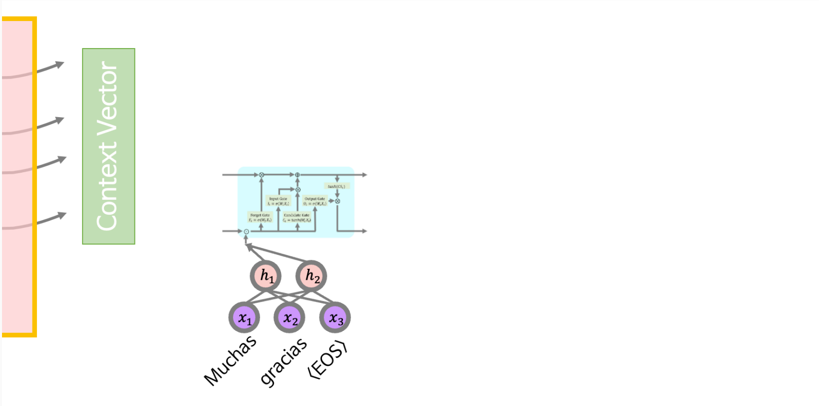


Seq2Seq

The decoder uses a new word2vec embedding;

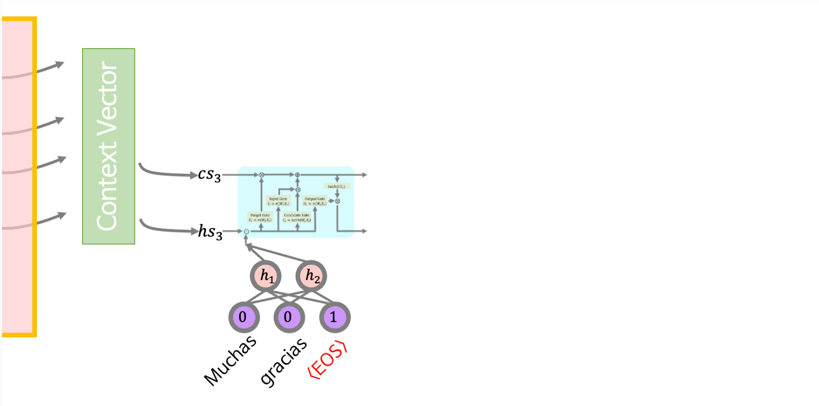
Seq2Seq

The decoder uses a new word2vec embedding;
For Spanish tokens “Muchas”, “gracias”, and “<EOS>”.



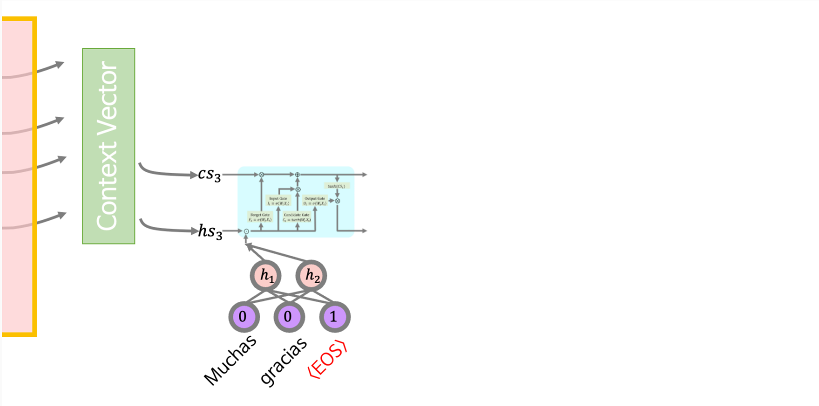
Seq2Seq

It starts by taking cs3 and hs3 from the context vector as the initial input.



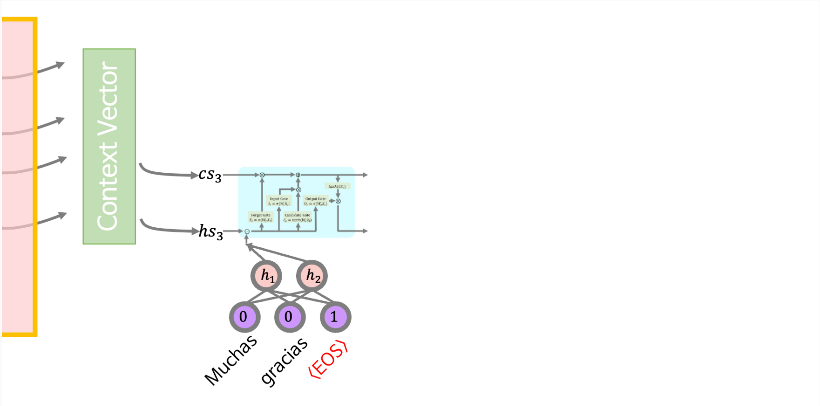
Seq2Seq

At the same time, the sentence begins with the token “<EOS>.”



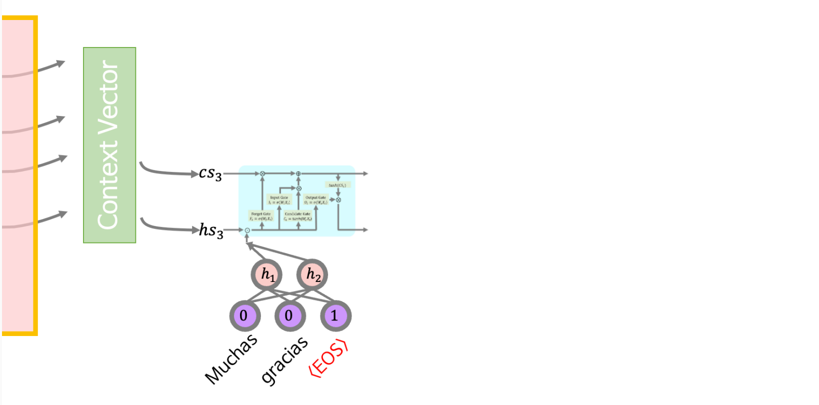
Seq2Seq

Although semantically it might make more sense to use “<SOS>” (Start of Sentence),



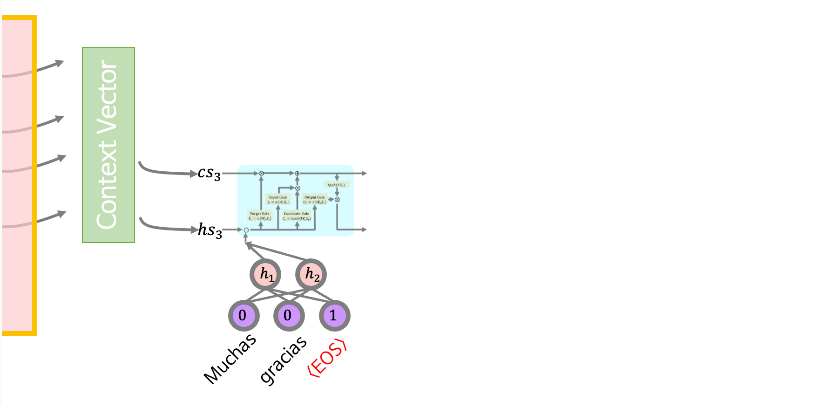
Seq2Seq

…in many cases, the end of one sentence also signals the start of another.



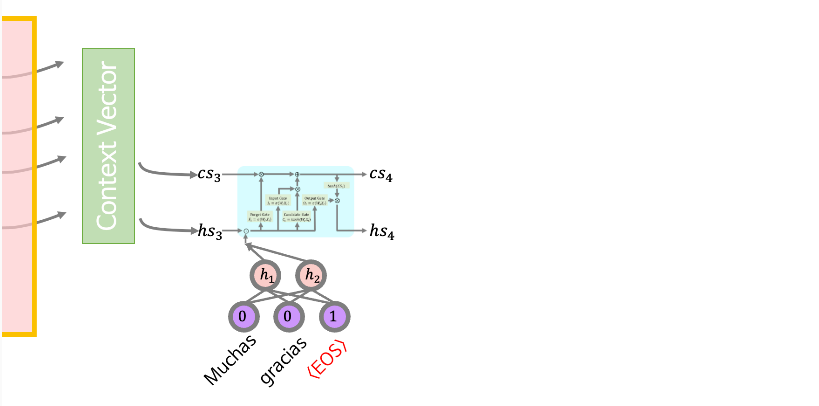
Seq2Seq

Thus, the decoder usually starts with the context vector and the token “<EOS>.”



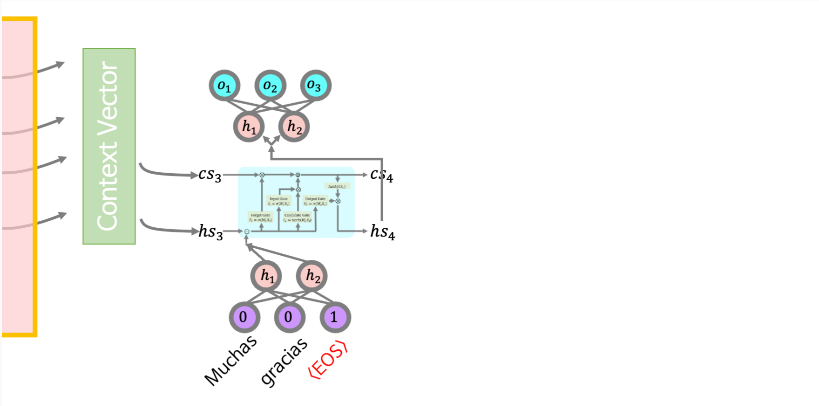
Seq2Seq

The LSTM then produces cs4 and hs4.



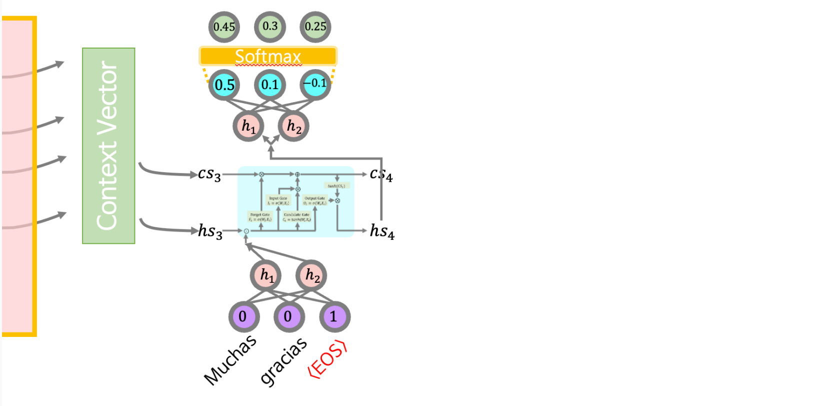
Seq2Seq

The decoder's word2vec component receives hs4 and computes its output.

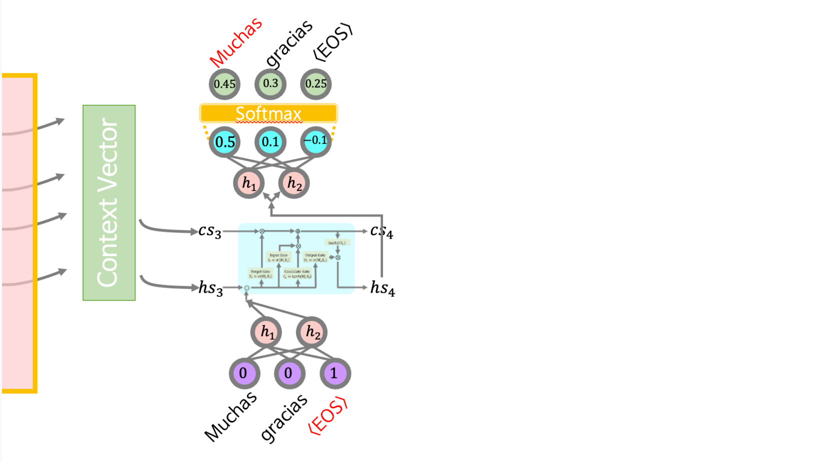


Seq2Seq

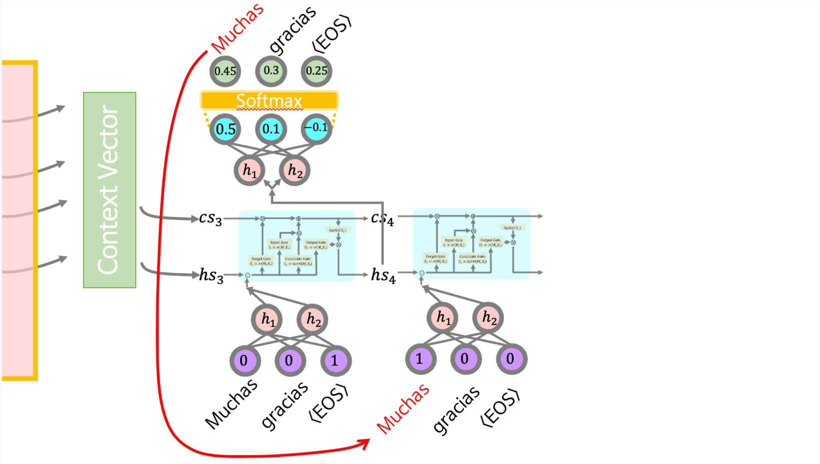
The output of word2vec is converted into probabilities using a softmax function.



Based on the softmax output, the LSTM produces the word "gracias."

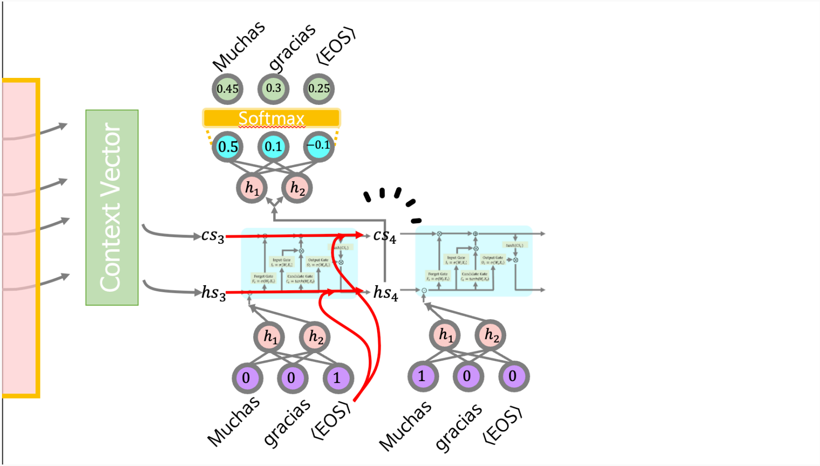


This output word is then fed back into the LSTM as the next input.



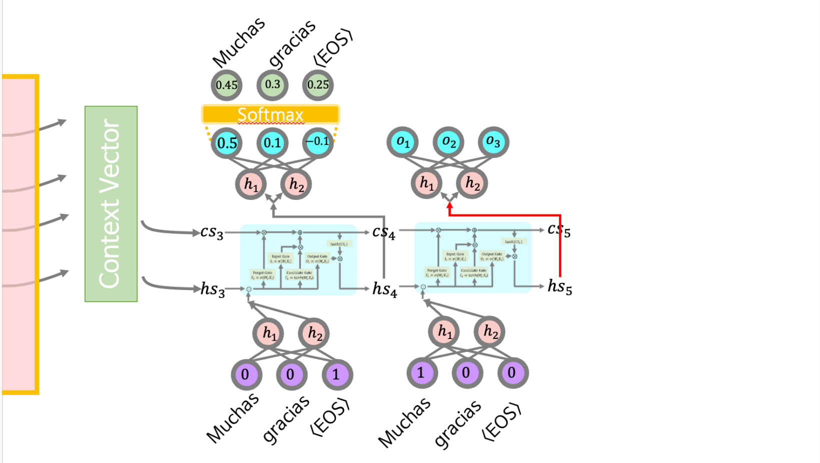
Seq2Seq

The input “gracias,” together with cs4 and hs4, generates new states cs5 and hs5.



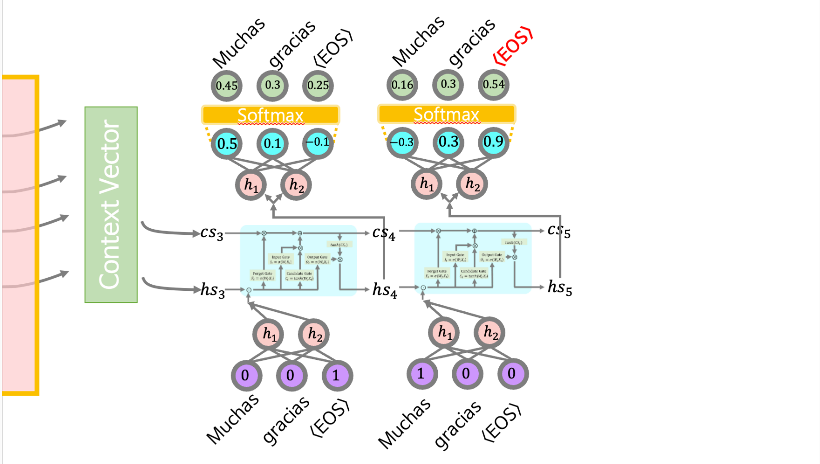
Seq2Seq

We then repeat the same process—passing through the decoder and applying softmax again.



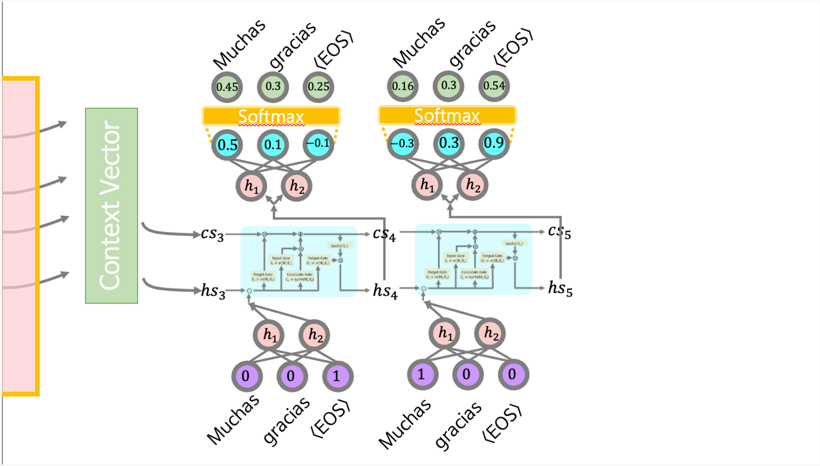
Seq2Seq

The softmax output this time becomes “<EOS>,” marking the end of translation.



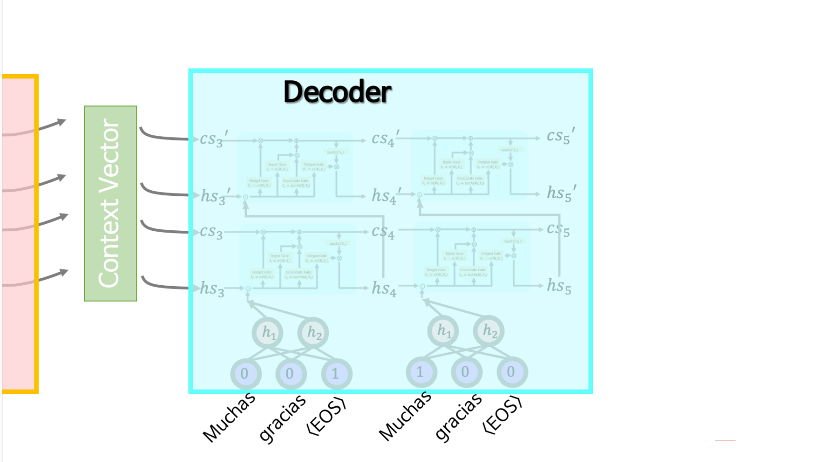
Seq2Seq

This completes the **forward pass** of the Seq2Seq model. (The backward pass updates weights from the context vector back to the encoder.)



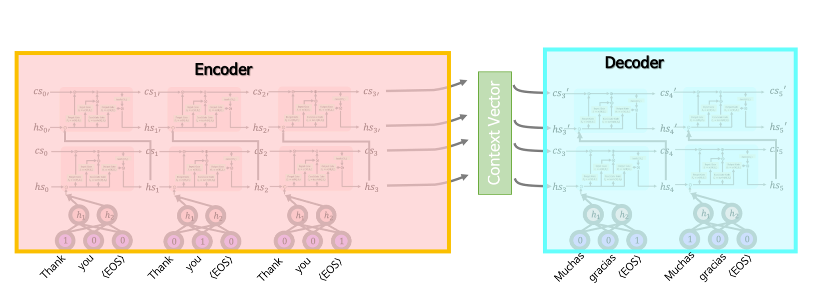
Seq2Seq

This entire decoding process is called the **decoder** in Seq2Seq.



Seq2Seq

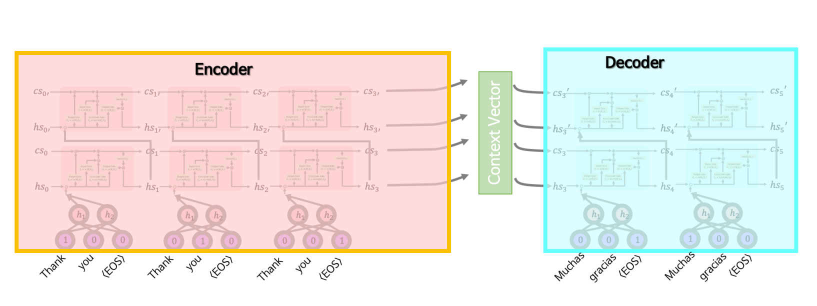
Overall, the Seq2Seq model can be viewed as an Encoder–Context–Decoder structure.



Seq2Seq

Because Seq2Seq has this Encoder–Context Vector–Decoder structure, it can translate between languages with very different word orders and sentence lengths.

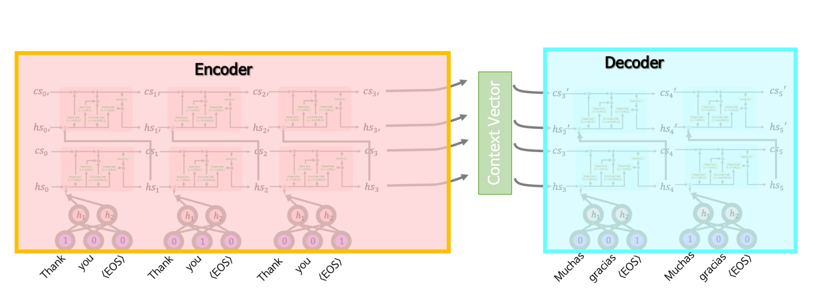
- English $\rightarrow$ Spanish: *How are you?* $\rightarrow$ *¿Cómo estás?*
- English $\rightarrow$ French: *How are you?* $\rightarrow$ *Comment ça va ?*



Seq2Seq

Seq2Seq is also widely used in conversational applications like chatbots.

- User: “How’s the weather today?”
- Bot: “It’s sunny and warm!”
- User: “Great, should I go hiking?”
- Bot: “Yes, that sounds like a great idea!”

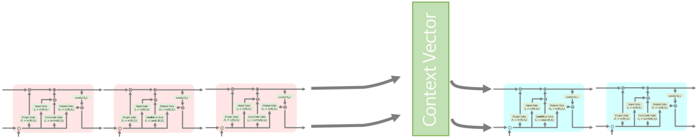


Seq2Seq: Problem

Seq2Seq is great, BUT...

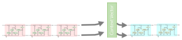
Seq2Seq: Problem

There was a limitation.



Seq2Seq: Problem

There was a limitation.

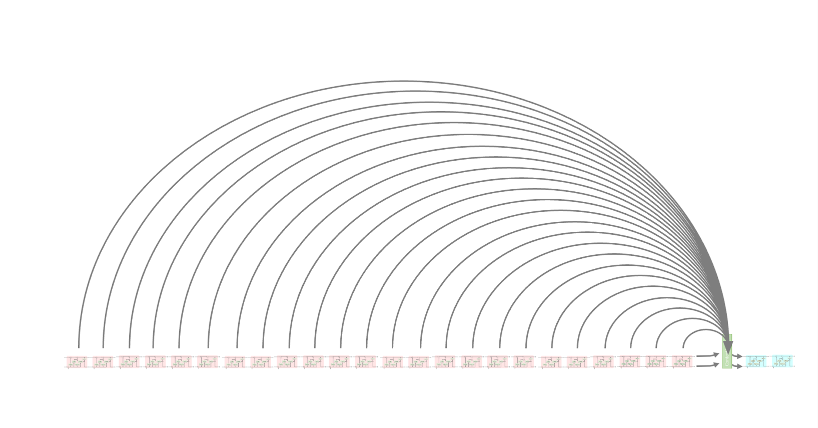


Seq2Seq: Problem

If the input sequence becomes this long...

Seq2Seq: Problem

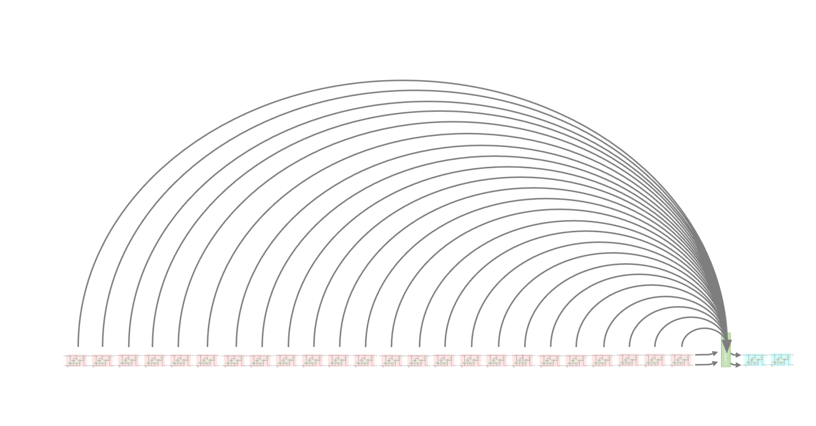
It becomes quite difficult to pack all the information of the input sequence into a fixed-length context vector.



This is called **Bottleneck** problem.

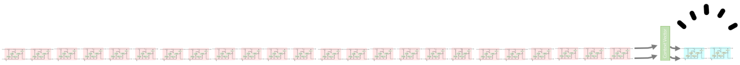
Seq2Seq: Problem

The attention mechanism was introduced to address this problem.



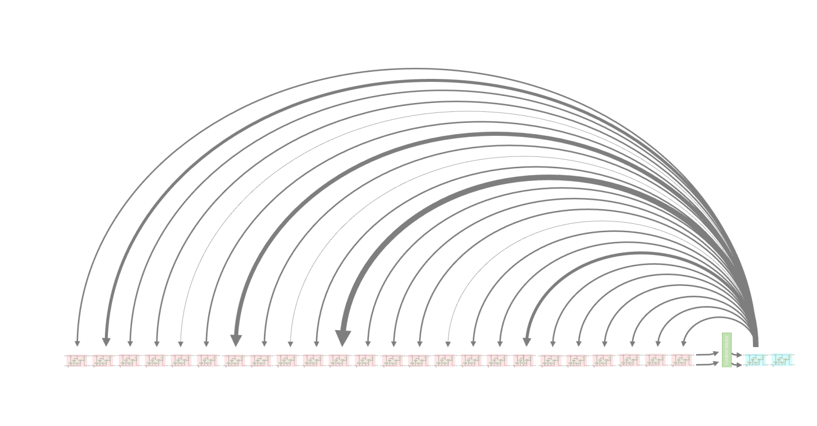
Seq2Seq: Problem

When the decoder generates each word in the output sequence, the attention mechanism



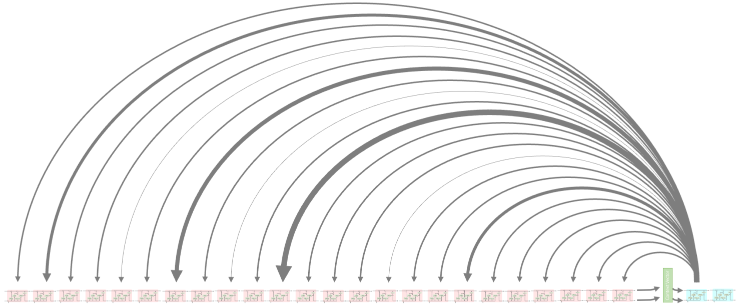
Seq2Seq: Problem

is an algorithm that makes it “attend” to which parts of the input sequence are important.



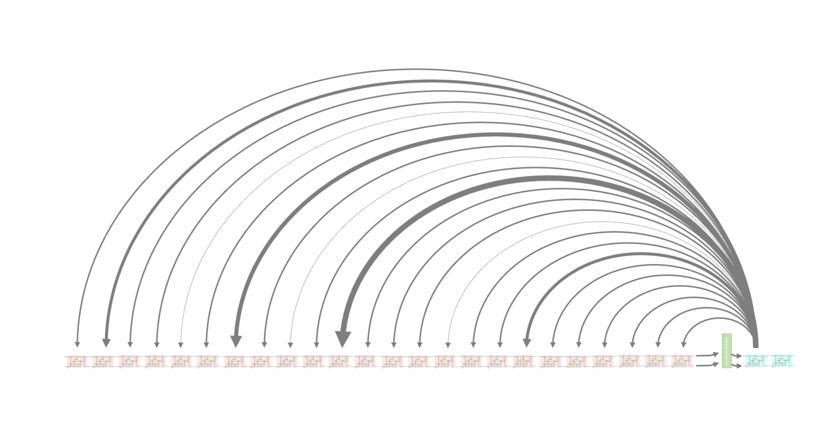
Seq2Seq: Problem

Introducing attention enables the model to handle longer sequences and improves translation quality, among other benefits.



Seq2Seq: Problem

Its effect is especially prominent in tasks with complex sentence structure or long-distance dependencies.



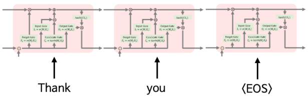
New technique: Attention

Attention

Now, let's look at what the attention mechanism is.

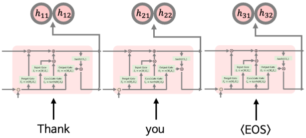
Attention

Suppose the input sequence comes in like this:



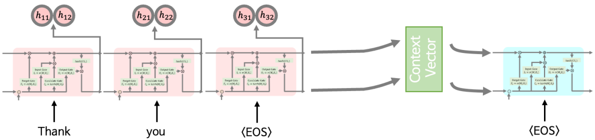
Attention

We store the hidden state for each input word separately, as follows.



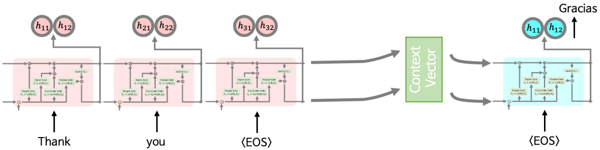
Attention

Then, as in plain Seq2Seq, we build a context vector and feed it to the decoder,



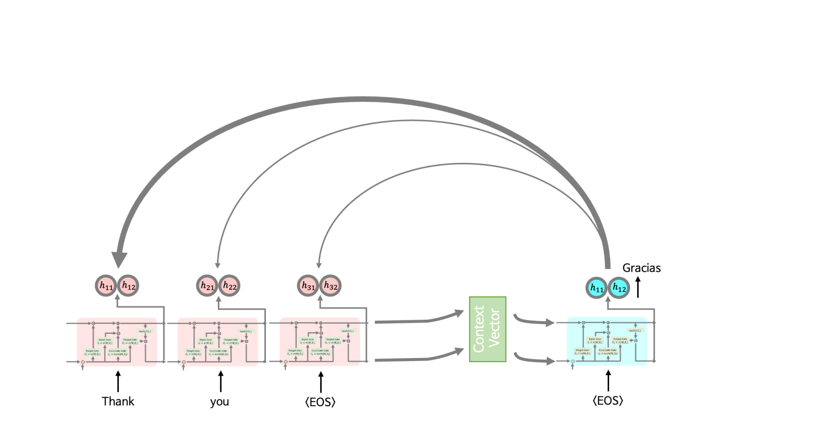
Attention

and obtain the decoder’s hidden state and output as follows.



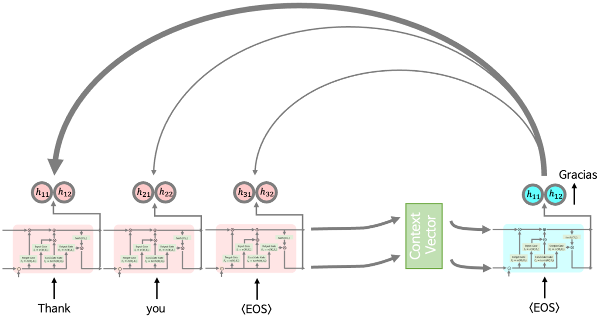
Attention

The core of attention is to find the relationship between the current decoder hidden state and the input that is estimated to be most relevant.



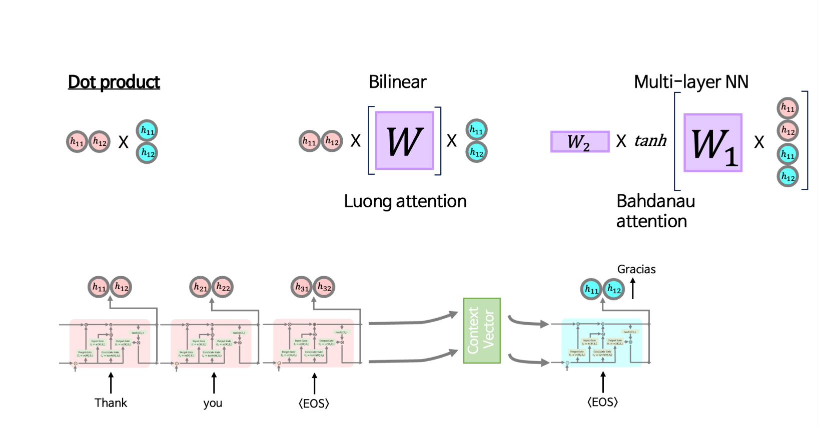
Attention

Here, the measure that determines the relationship between two hidden states is, again, the similarity between two vectors.



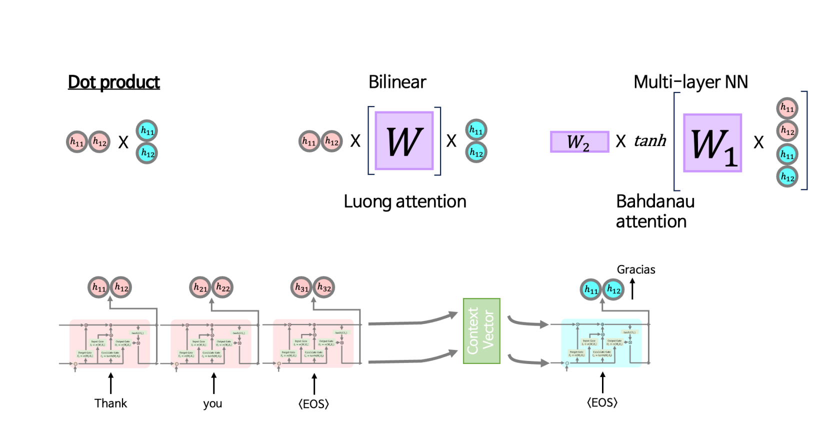
Attention

The methods for computing attention scores—i.e., vector similarity—can be broadly divided into three types.



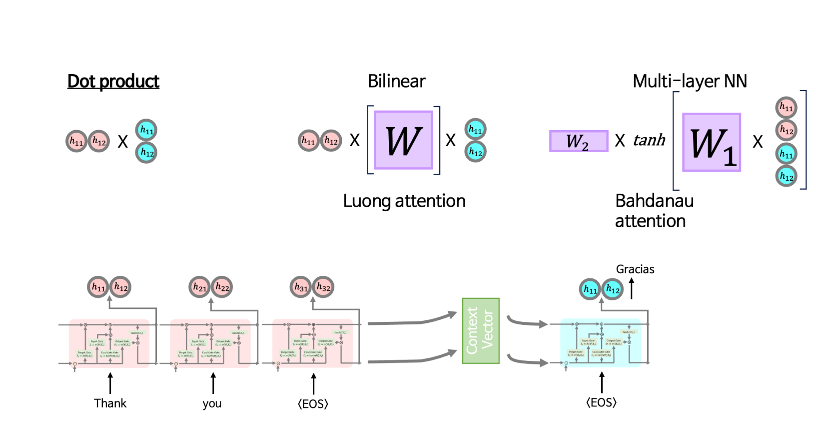
Attention

Each method has its own characteristics and computational complexity, but



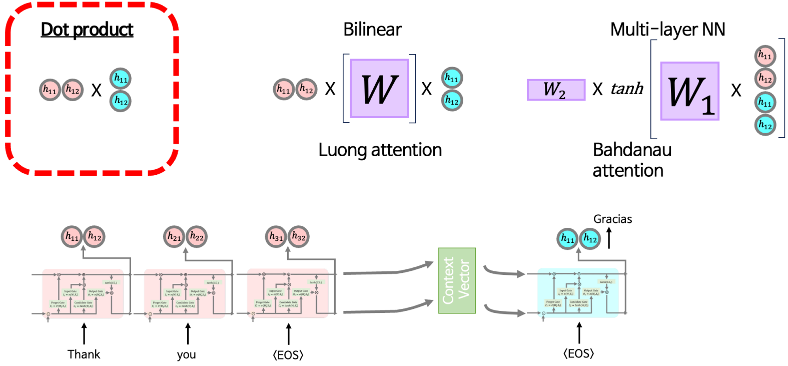
Attention

ultimately **they all amount to comparing vector similarity** between the input and output sequences to produce attention scores.



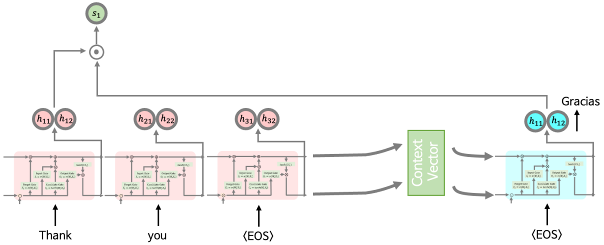
Attention

We will use the simplest method: the dot product.



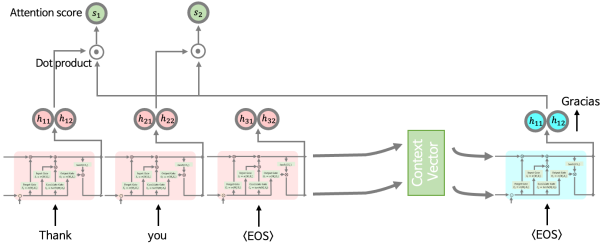
Attention

For example:



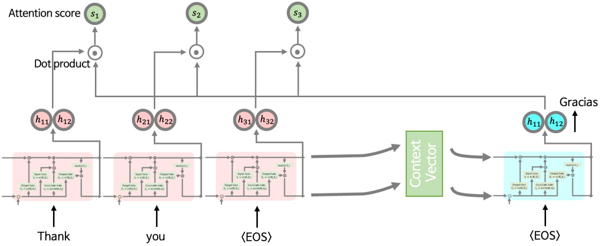
Attention

For example:



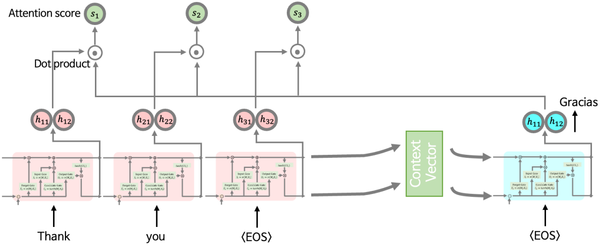
Attention

For example:

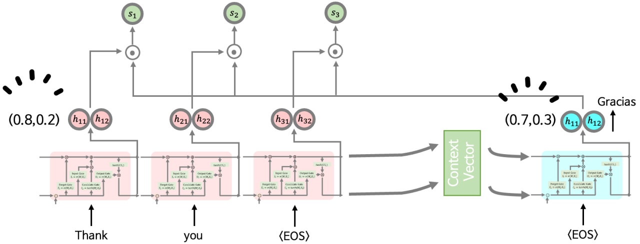


Attention

For example:

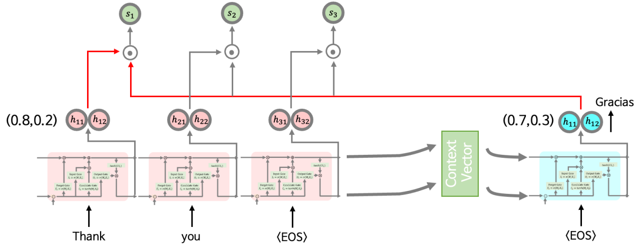


Suppose the encoder hidden state is (0.8, 0.2) and the decoder hidden state is (0.7, 0.3).

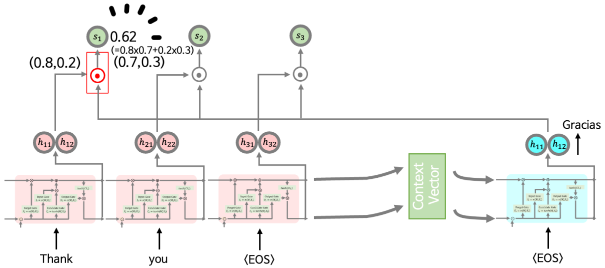


Attention

Suppose the encoder hidden state is (0.8, 0.2) and the decoder hidden state is (0.7, 0.3).

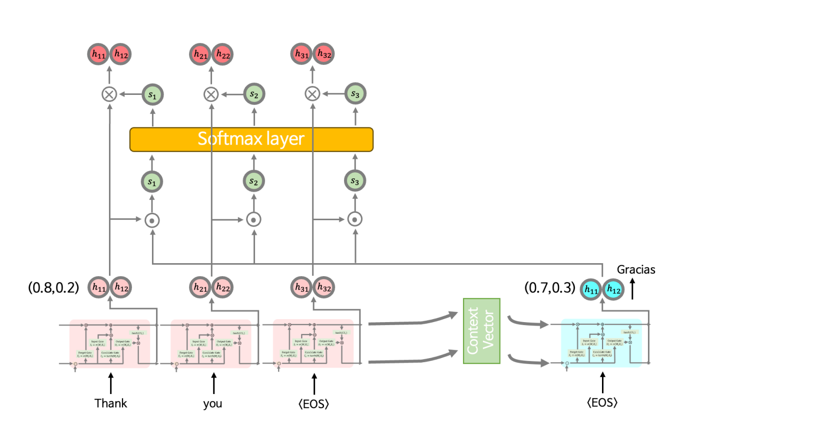


Then the attention score s1 becomes 0.62 through the following dot-product computation.



Attention

Next, we compute the softmax of each attention score.

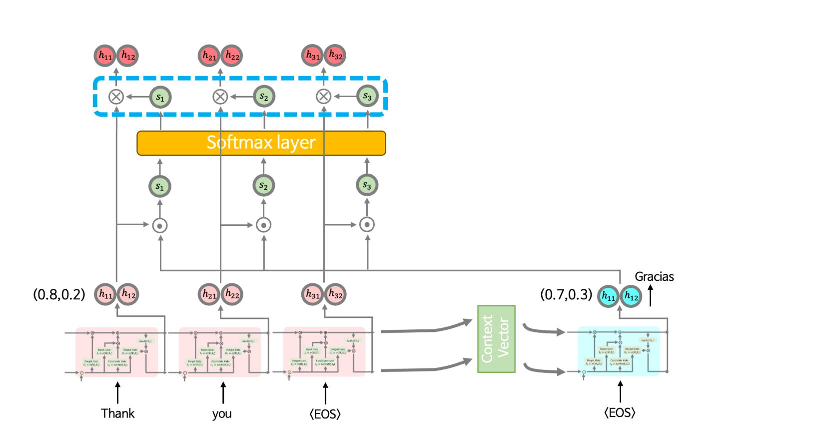


Attention

Why Softmax?

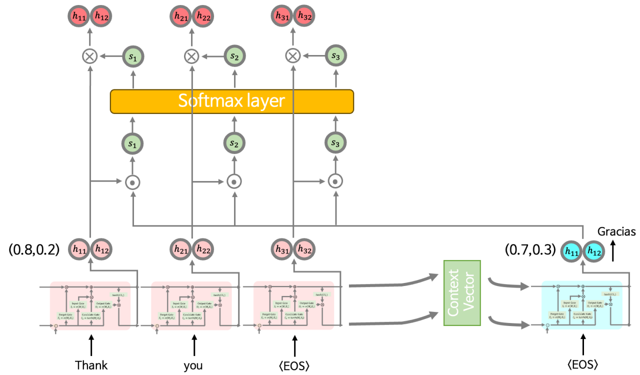
Attention

Why Softmax? To convert the attention scores into a probability distribution and normalize them.

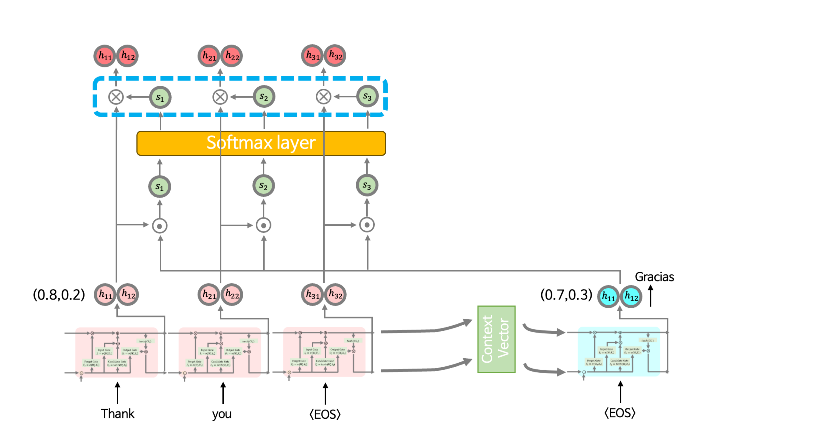


Attention

Next, we multiply each attention score by its corresponding input hidden state.

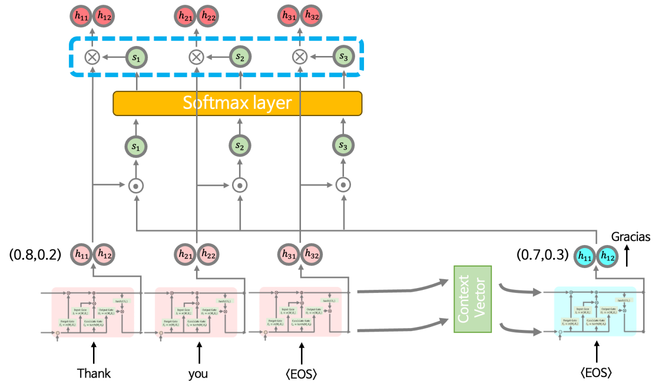


We do this because each attention score is a simple scalar value,



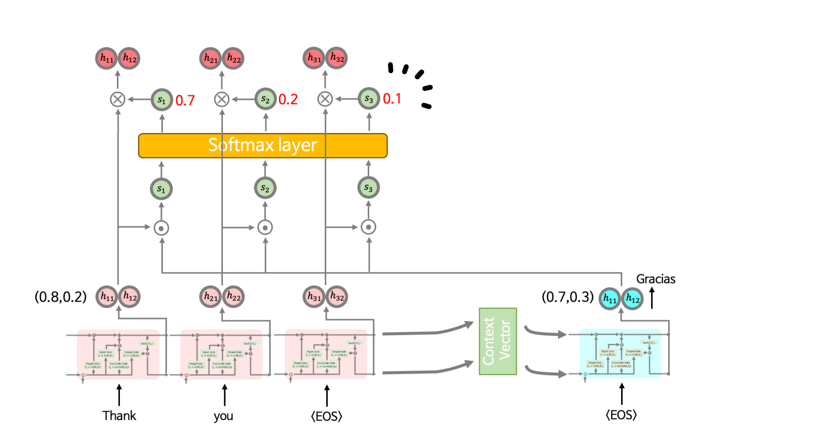
Attention

and the multiplication has **the effect of amplifying the input hidden states** according to their attention weights.



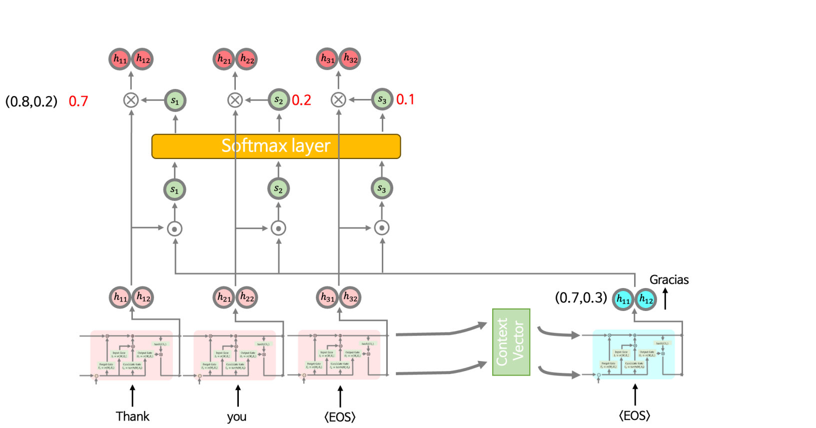
Attention

For example, if the attention scores after the softmax layer are as follows—



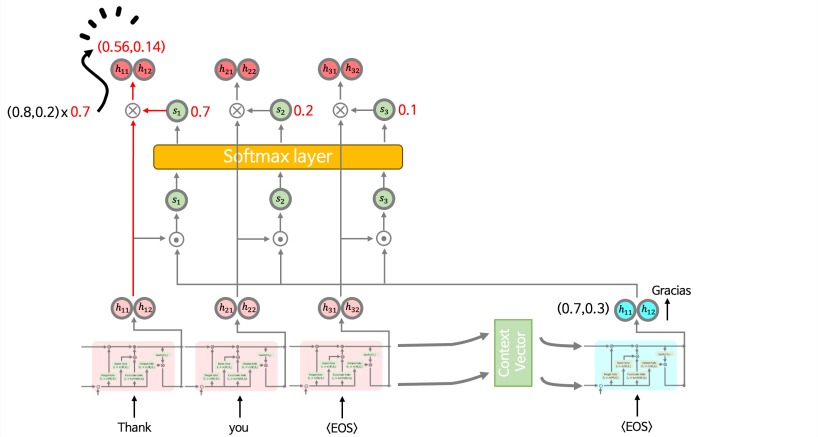
Attention

For example, if the attention scores after the softmax layer are as follows—



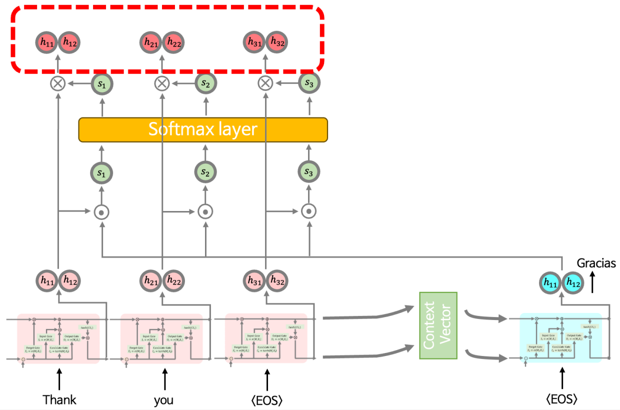
Attention

Then the input state (0.8, 0.2) multiplied by 0.7 becomes (0.56, 0.14).



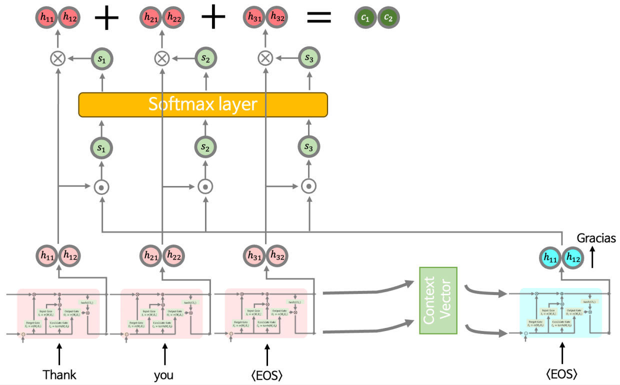
Attention

Now, take these attention-weighted input hidden states,



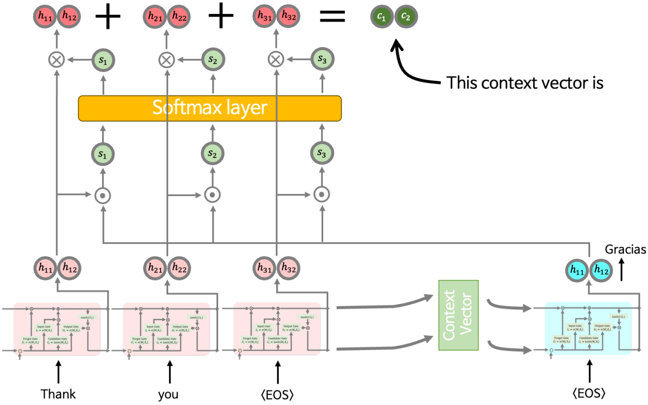
Attention

sum them up, and you obtain a new context vector.



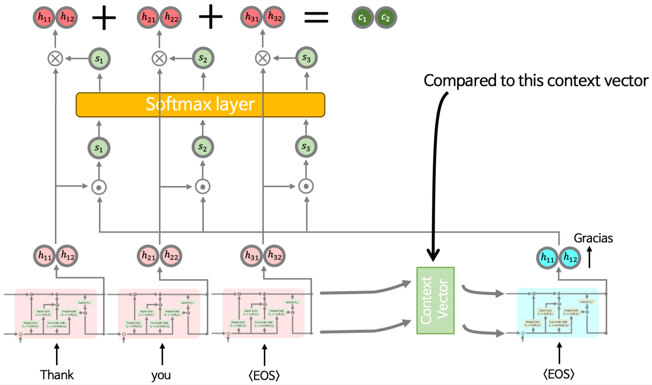
Attention

sum them up, and you obtain a new context vector.



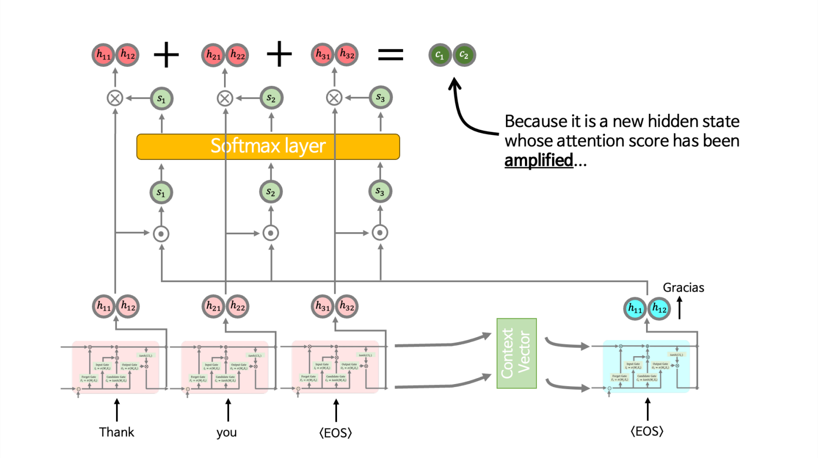
Attention

sum them up, and you obtain a new context vector.



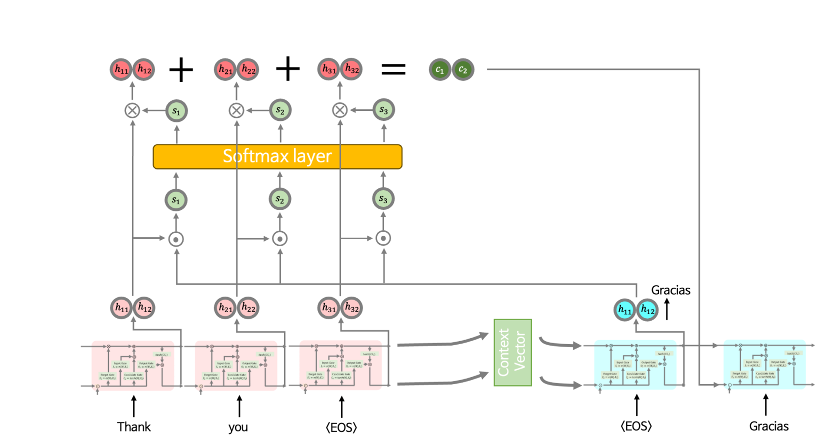
Attention

sum them up, and you obtain a new context vector.



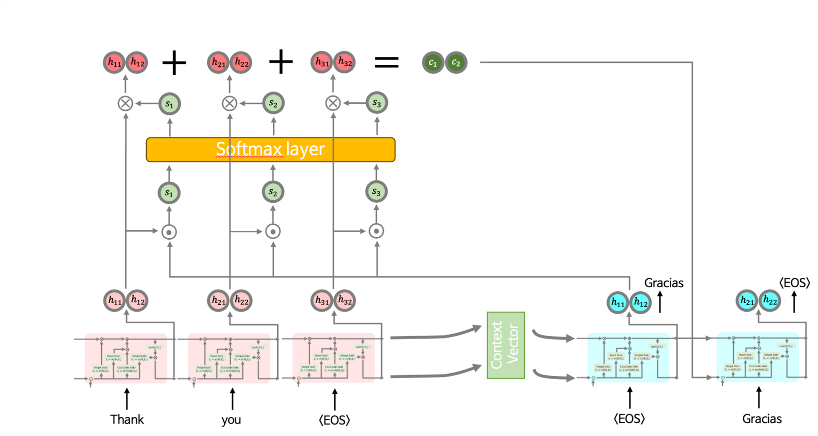
Attention

Feed this into the next decoder LSTM hidden state, and provide the required values in turn,



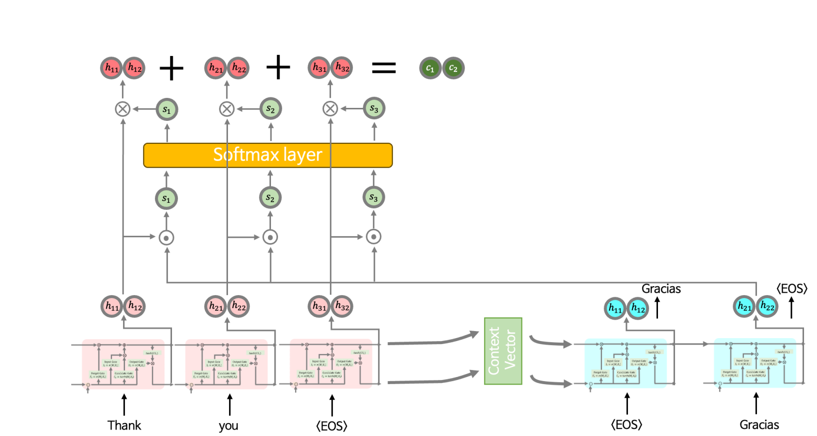
Attention

and you can compute the next hidden state,

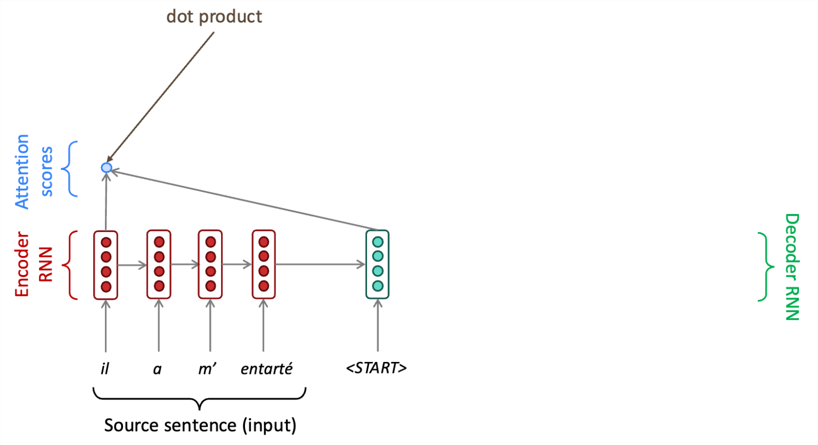


Attention

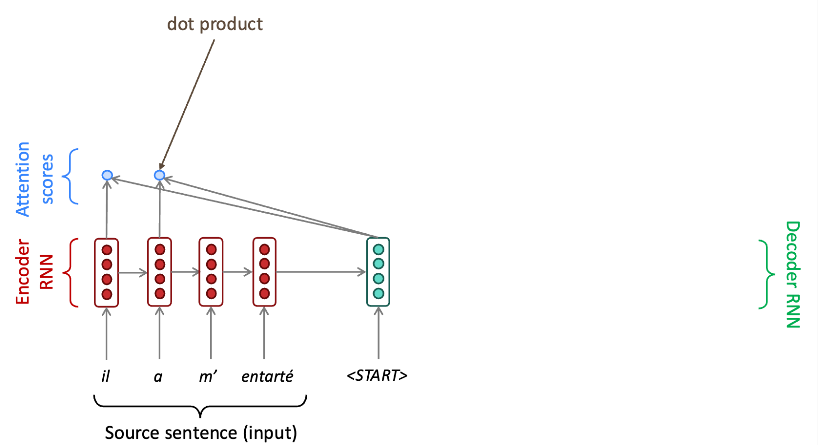
and in the same way compute the attention context vector for the following step.



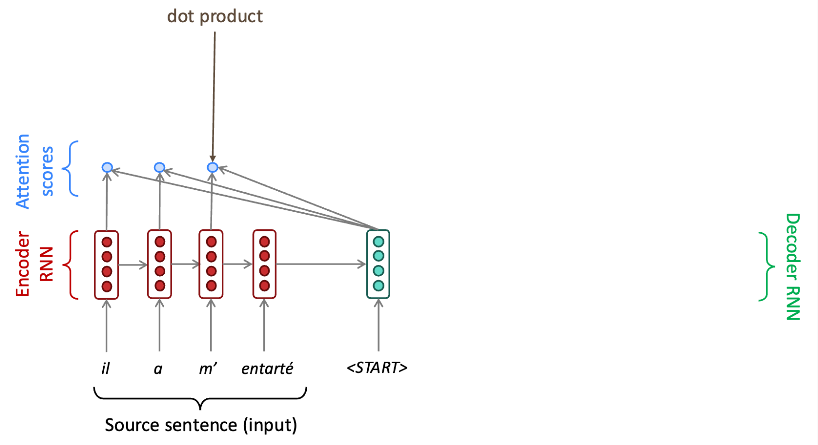
Seq2Seq with attention



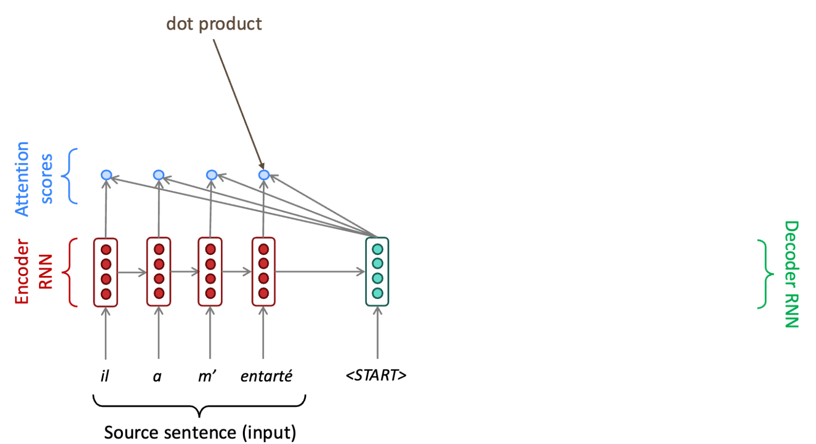
Seq2Seq with attention



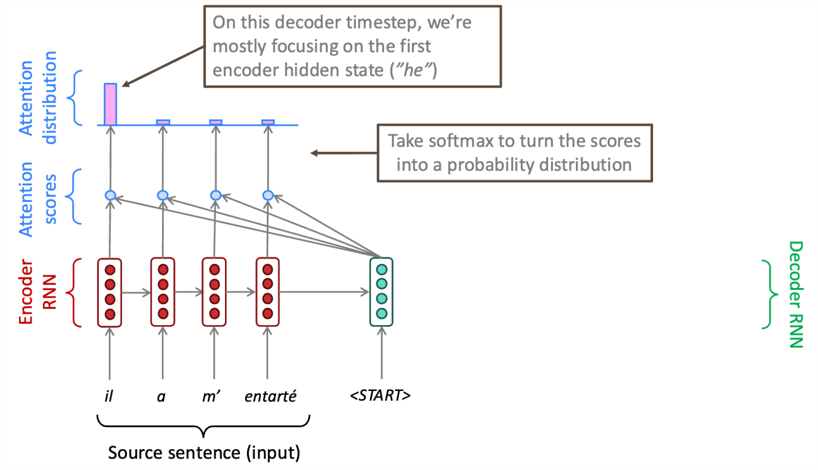
Seq2Seq with attention



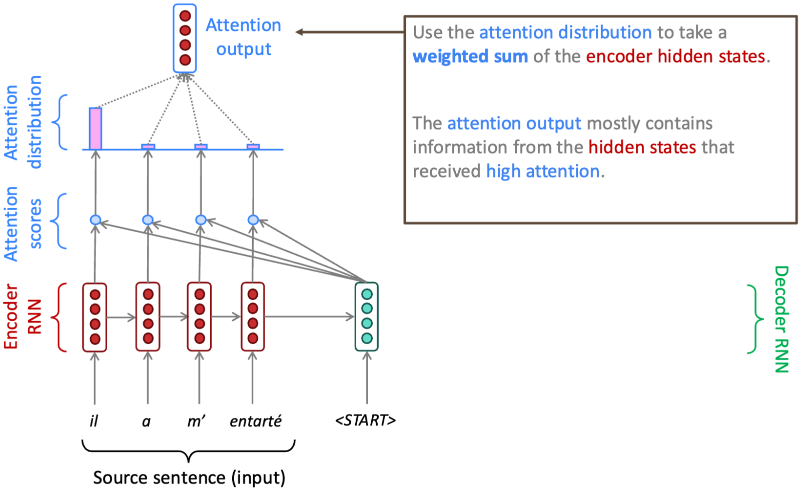
Seq2Seq with attention



Seq2Seq with attention



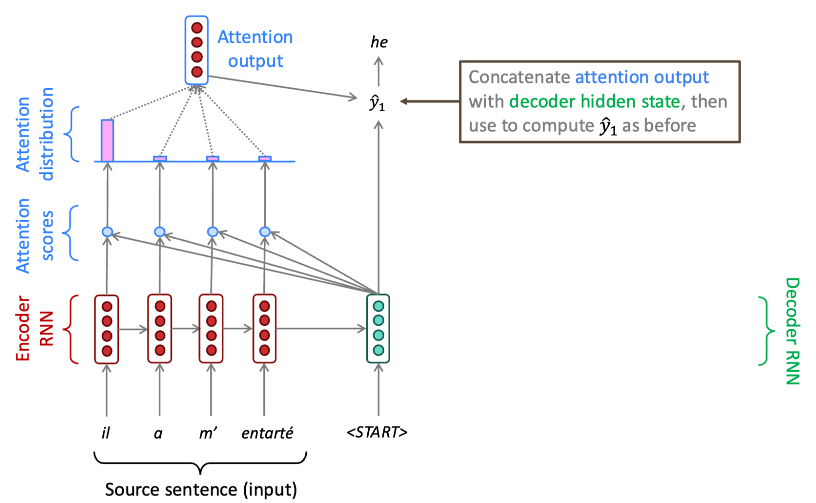
Seq2Seq with attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Seq2Seq with attention



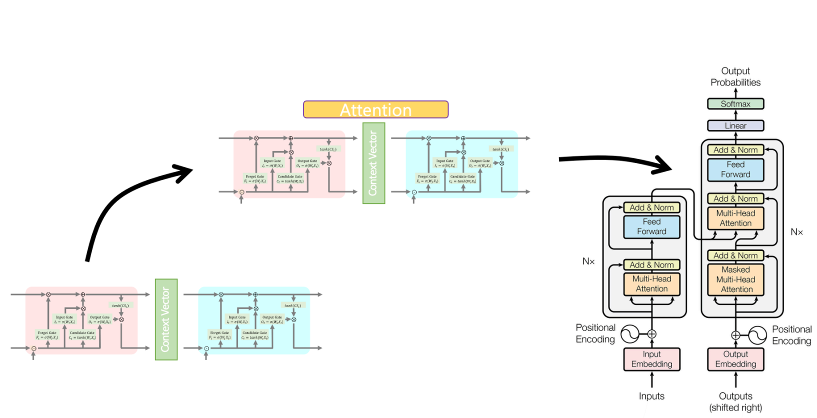
Attention is a *general* deep learning technique

Upshot:

- Attention has become the powerful, flexible, general way pointer and memory manipulation in deep learning models. (A new idea from 2010).

The transformer model

Eventually led to the development of the **Transformer** model.



Attention in transformers, step-by-step

Multi-head attentions
`https://www.youtube.com/watch?v=eMlx5fFNoYc`

Reminder

- **10/9:** Group meeting (after the meeting, Background research topic submission by 10th)
- **10/14:** Fall Break
- **10/16:** Quiz (Online)

Group Meeting Schedule

Time	Group
12:30–12:38	Group 1
12:38–12:46	Group 2
12:46–12:54	Group 3
12:54–1:02	Group 4
1:02–1:10	Group 5
1:10–1:18	Group 6
1:18–1:26	Group 7
1:26–1:34	Group 8
1:34–1:42	Group 9
1:42–1:45	*Buffer*

- Each group has about 8 minutes. If one finishes early, the next group may begin right away.
- Please bring your Background Research Brief draft
- The final version should be submitted by Friday (10/10)

Quiz (Online) - Updated!

- Worth **10 points**: Designed to help you review key concepts covered in class
- **Open book**, but **use of AI tools is not allowed**; Focuses on the essential ideas discussed in class — take it as an opportunity to check your own understanding!
- **75 minutes** to complete once you begin (you cannot pause or restart after starting)
- **Available on Thursday (10/16) 9 AM–5 PM**
- **Question types:**
  - Multiple Choice: 21 questions
  - Short Written Response: 6 questions (approximately 100–150 words, one paragraph each)
  - Long Written Response: 1 question
- **Scope:** All topics covered from the first class (Word Vectors) through the last class