ecnológico
Monterrey

NLP 2

Inteligencia artificial avanzada para la ciencia de datos II Modulo 5 NLP 2

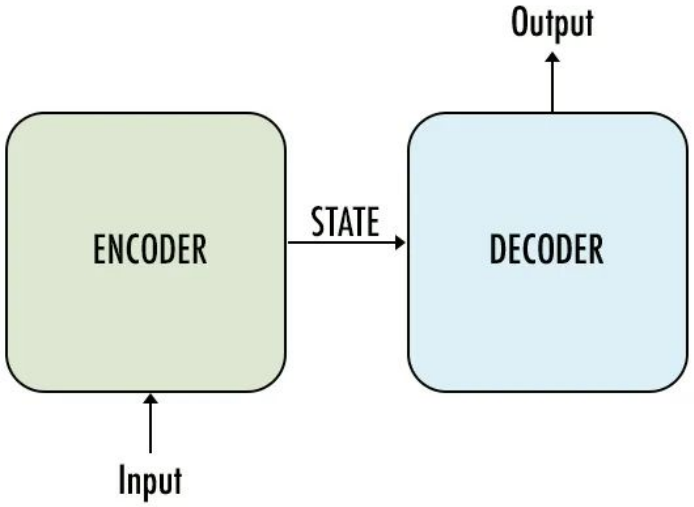
seq2seq

Sequence 2 Sequence

We discuss in RNN lecture that some models are designed to receive a sequence and produce a sequence **(many to many)** .

This models were very popular for language translation, receive a sentence in english and produce the equivalent sentence in spanish.

This was first propose by using two RNN (many to one and one to many) this was called an encoder decoder architecture



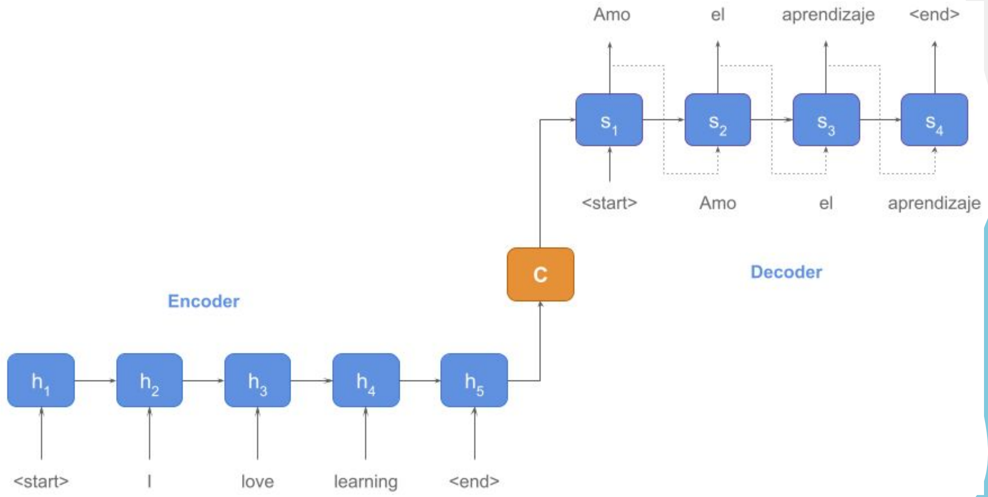
Encoder / Decoder

Encoder model was first tested for translation purposes, the architecture considers this first phase to read all the events of the input and produces a context vector.

The **context vector** represents the whole input information and it is useful to train an output model using it as new input, in other words is the **compression of the input in a vectorial representation.**

Decoder model receives the context vector and creates a new sequence, this is very useful because sometimes output length is not the same as input length
Example :

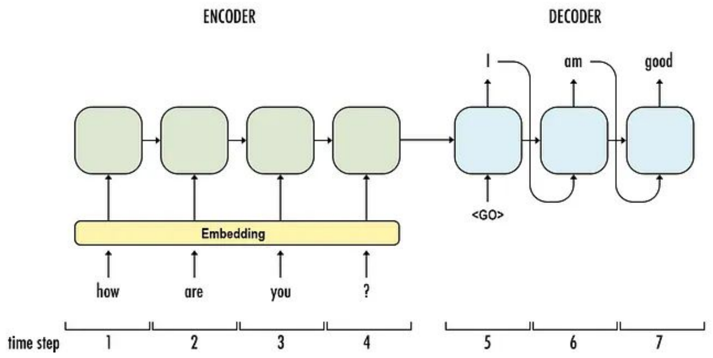
“I will love u” => “Yo te amaré”

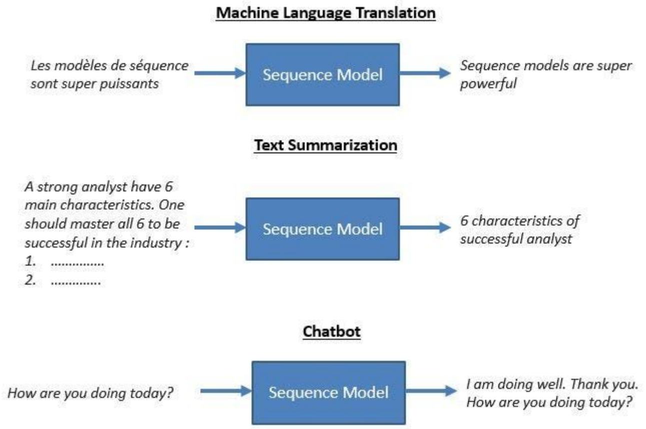


Other consequences

This models opened new opportunities, probably the main benefit is the text **generation problem.**

If we can compress and represent a whole sequence to produce another , then we will focus only on the task to solve (seq 2 seq models).



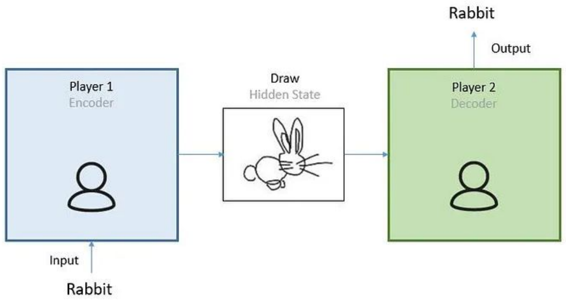

Machine Language Translation
Les modèles de séquence sont super puissants
Sequence Model
Sequence models are super powerful
Text Summarization
A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :
1. ....
2. ....
Sequence Model
6 characteristics of successful analyst
Chatbot
How are you doing today?
Sequence Model
I am doing well. Thank you. How are you doing today?

Some problems

By intuition the idea of **seq2seq** is good enough to solve N:M problems.

- With LSTM / GRU we have long memory enough for normal inputs
- We do not depend on the length of our sequences
- We can summarize , transform, translate into a whole new sequence but we have some problems:

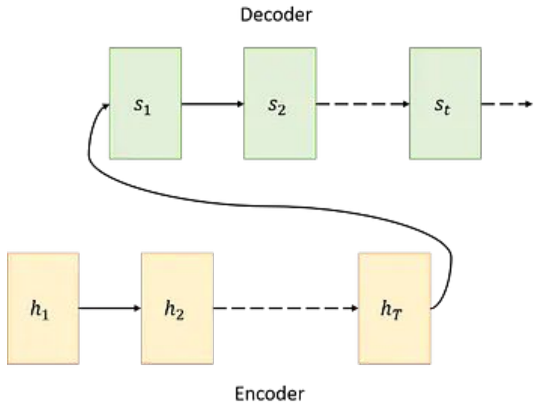
1. All the input is compressed in one vector (complexity is terrible)
2. All the inputs contribute in the **same importance**, the only variable is time



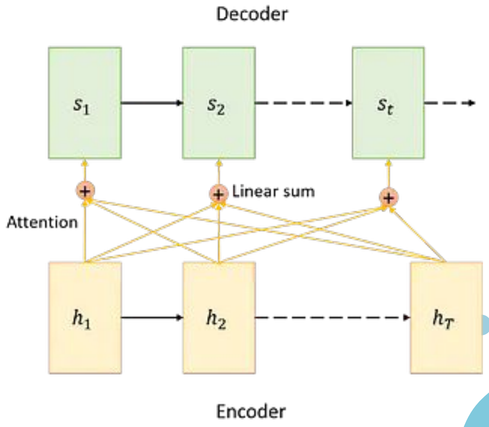
Attention component

Attention is our superhero !!

The attention mechanism is a method that removes the context hidden state between models in order to create an attention relation of every event in the input sequence with every possible output in the output.



(a) Vanilla Encoder Decoder Architecture



(b) Attention Mechanism

Attention in a practical view

Suppose a translation problem,
You want to translate to spanish the following sequence:

I love u

The spanish translation is:

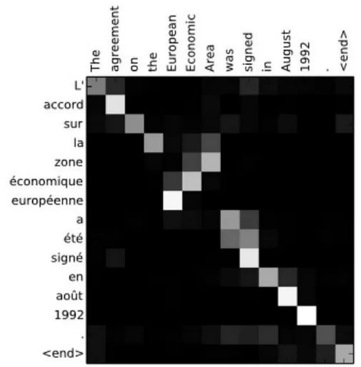
Te amo

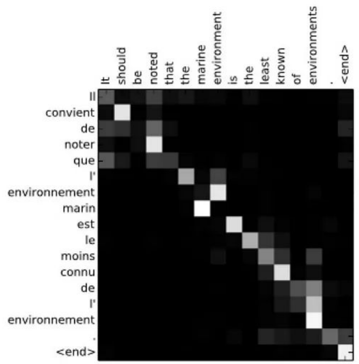
Now lets move to the decode model, in the first event in the sequence you want to produce **“te”** which actually is produced because of the english token **“you”** and “**I”**.

Then you need to produce “**amo”** which is produced because in the input the event **“love”** happens.

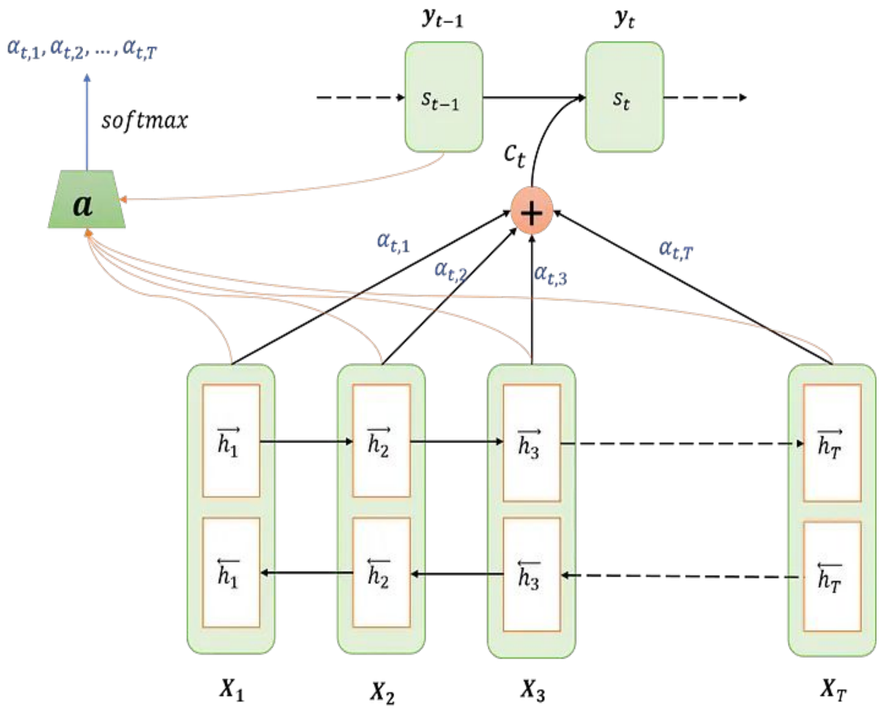
If every step we produce a token we can score a grade of importance of each input token, then we have an attention mechanism.

In the example, knowing that “**I” and “u”** needs attention in order to produce “**te”** or **“love”** needs attention when we want to produce **“amo”** is the attention mechanism.





Architecture

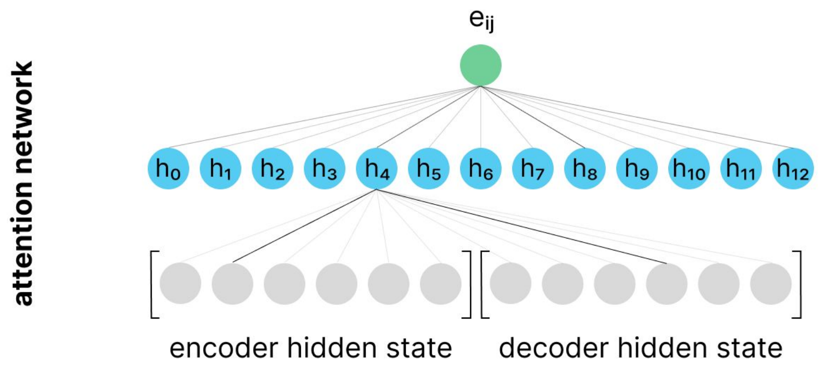


Now in the encoder we will need no store N context vector where N is the number of steps (events) in the sequence.

Then by using a context weight vector que can produce a Ct vector to produce an output in every decoder step by using the softmax function.

The final of the sequence generated is given when mask tokens happens <end>.

Training attention FNN

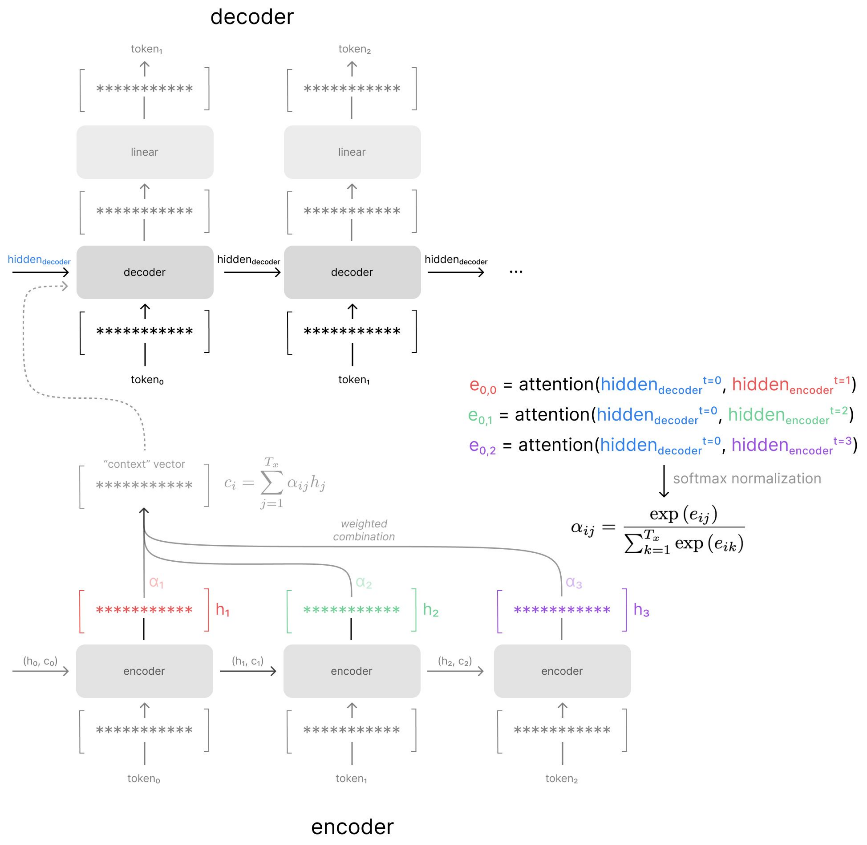


One possibility and some architectures uses is the incorporation of a traditional FNN.

This network is trained to obtain the attention weights required,

The input layer represents both, encoder and decoder hidden states and the output layer is the amount of attention

Whole architecture working



$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$e_{0,0}$ = attention($\text{hidden}_{\text{decoder}}^{t=0}$, $\text{hidden}_{\text{encoder}}^{t=1}$)

$e_{0,1}$ = attention($\text{hidden}_{\text{decoder}}^{t=0}$, $\text{hidden}_{\text{encoder}}^{t=2}$)

$e_{0,2}$ = attention($\text{hidden}_{\text{decoder}}^{t=0}$, $\text{hidden}_{\text{encoder}}^{t=3}$)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Benefits of LSTM with Attention

1. Enhanced Sequence Modeling: LSTM with attention mechanisms can focus on the most relevant parts of a sequence, improving performance in tasks such as machine translation and text summarization.

2. Improved Contextual Understanding: Attention helps the model better understand the context by weighing the importance of different elements in the input sequence, leading to more accurate predictions.

3. Handling Variable-Length Sequences: LSTM with attention can effectively handle input sequences of varying lengths, making it suitable for tasks with irregular data.

Limitations of LSTM with Attention

1. Computational Intensity: The addition of attention mechanisms increases the computational demands, which can result in longer training times and higher resource requirements.

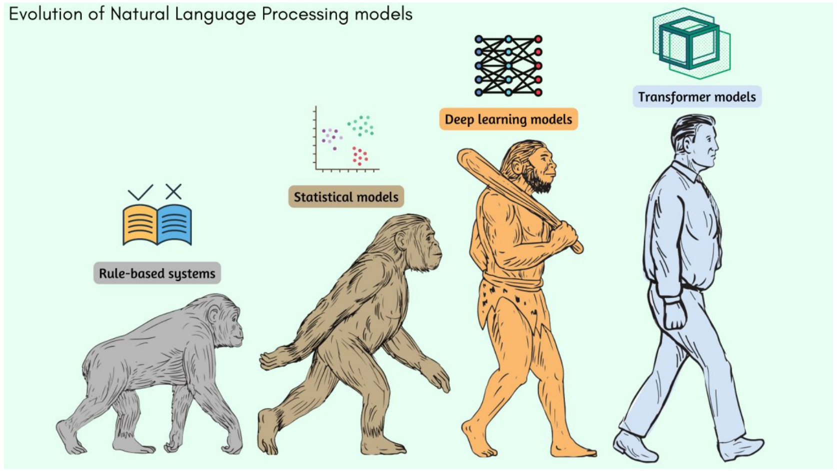
2. Increased Model Complexity: Combining LSTM and attention mechanisms makes the model more complex and challenging to fine-tune effectively.

3. Data Efficiency: Like regular LSTMs, LSTM with attention mechanisms may require a substantial amount of data for effective training.

Whats next?

- Encoder/decoder architecture
- RNN
- Embeddings
- Attention mechanism

==



Thanks

Do you have any questions?

emmanuel.paez@tec.mx
Slack #module-5-nlp-1