



STEVENS INSTITUTE OF TECHNOLOGY

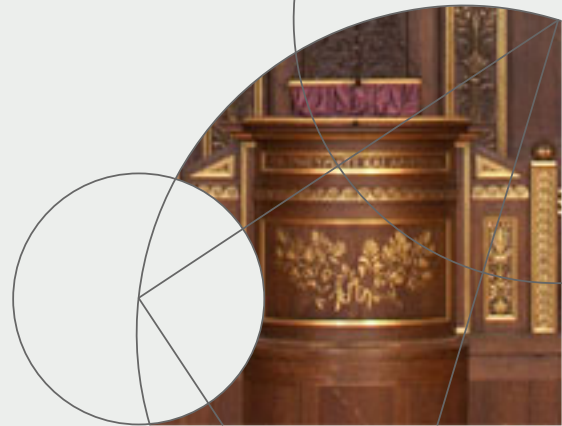


Data Acquisition and Processing II: Deep Learning @ Recurrent Neural Network

Rensheng Wang,

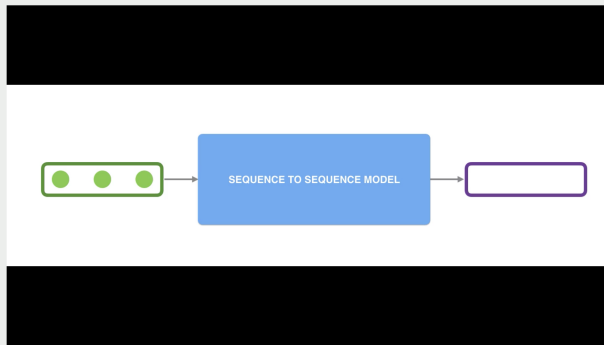
rwang1@stevens.edu

Dept. of Electrical and Computer Engineering
Stevens Institute of Technology



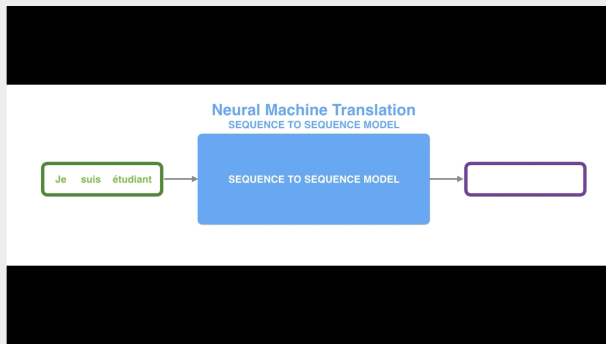
Sequence-to-Sequence Model

- A sequence-to-sequence model is a model that takes a sequence of items (words, letters, features of an images \dots etc) and outputs another sequence of items. A trained model would work like this:



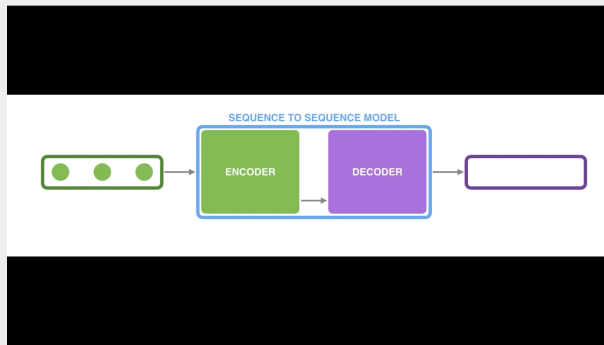
Sequence-to-sequence Model

- In neural machine translation, a sequence is a series of words, processed one after another. The output is, likewise, a series of words:



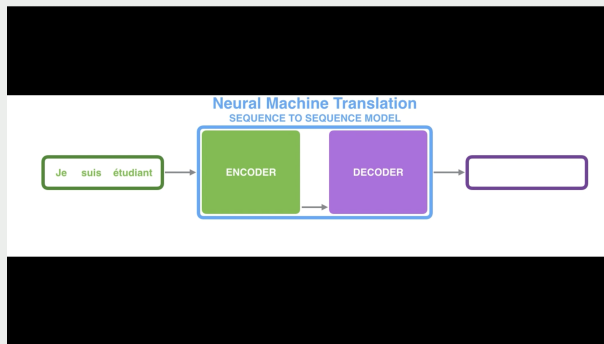
Encoder and Decoder

- Under the hood, the model is composed of an encoder and a decoder.
- The encoder processes each item in the input sequence, it compiles the information it captures into a vector (called the context).
- After processing the entire input sequence, the encoder send the context over to the decoder, which begins producing the output sequence item by item.



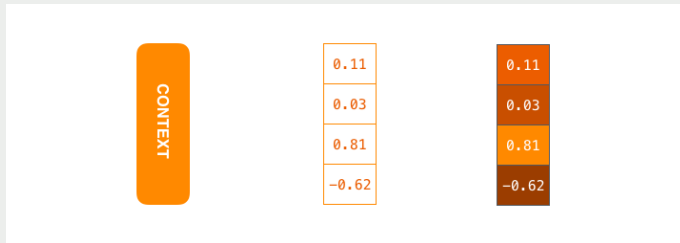
Encoder and Decoder

- Under the hood, the model is composed of an encoder and a decoder.
- The same applies in the case of machine translation.



Context Vector

- ❑ The context is a vector (an array of numbers, basically) in the case of machine translation. The encoder and decoder tend to both be recurrent neural networks.
- ❑ The context is a vector of floats. Later in this post we will visualize vectors in color by assigning brighter colors to the cells with higher values.






- 👉 You can set the size of the context vector when you set up your model. It is basically the number of hidden units in the encoder RNN. These visualizations show a vector of size 4, but in real world applications the context vector would be of a size like 256, 512, or 1024.



Word Embedding

- By design, a RNN takes two inputs at each time step: an input (in the case of the encoder, one word from the input sentence), and a hidden state. The word, however, needs to be represented by a vector.
- To transform a word into a vector, we turn to the class of methods called “word embedding” algorithms. These turn words into vector spaces that capture a lot of the meaning/semantic information of the words (e.g. king - man + woman = queen).

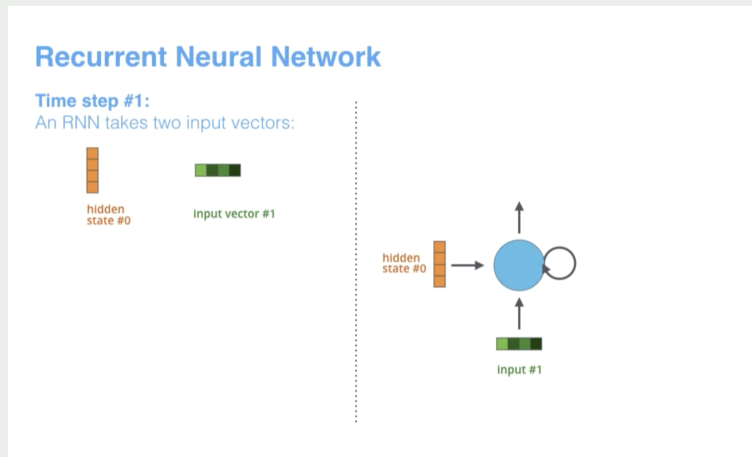
Input					
Je	0.901	-0.651	-0.194	-0.822	
suis	-0.351	0.123	0.435	-0.200	
étudiant	0.081	0.458	-0.400	0.480	

- We can use pre-trained embeddings or train our own embedding on our dataset. Embedding vectors of size 200 or 300 are typical, we're showing a vector of size four for simplicity.



Recurrent Neural Network (RNN)

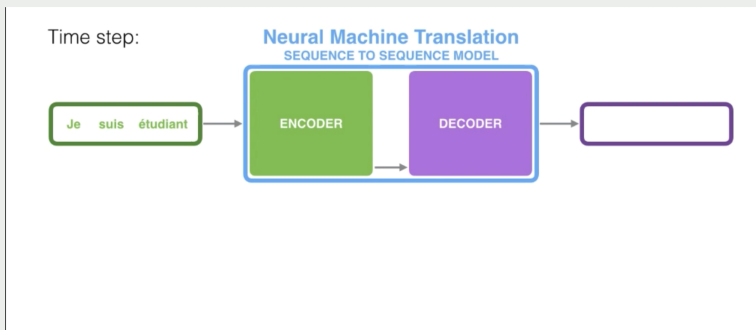
- Time Step #1: An RNN takes two input vectors:



- The next RNN step takes the second input vector and hidden state #1 to create the output of that time step.

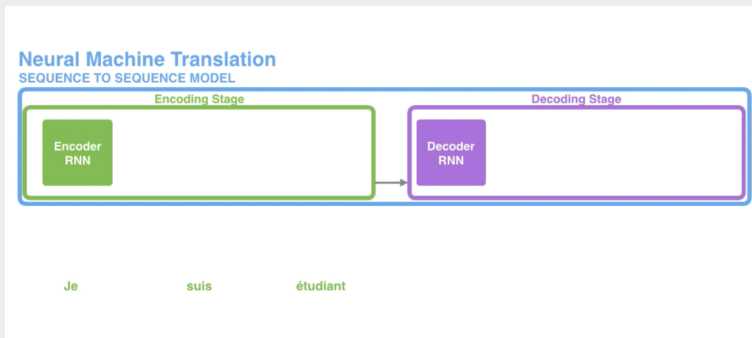
Hidden States of Encoder

- Since the encoder and decoder are both RNNs, each time step one of the RNNs does some processing, it updates its hidden state based on its inputs and previous inputs it has seen.
- Let us look at the hidden states for the encoder. Notice how the last hidden state is actually the context we pass along to the decoder.



Hidden States of Decoder

- ❑ The decoder also maintains a hidden states that it passes from one time step to the next.
- ❑ Let's look at the hidden states for the encoder. Notice how the last hidden state is actually the context we pass along to the decoder.



Attention Mechanism

- The context vector turned out to be a bottleneck for these types of models. It made it challenging for the models to deal with long sentences. A solution was proposed in Bahdanau et al., 2014 and Luong et al., 2015. These papers introduced and refined a technique called “Attention”, which highly improved the quality of machine translation systems. Attention allows the model to focus on the relevant parts of the input sequence as needed.

Time step: 7

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Attention Mechanism

- At time step 7, the attention mechanism enables the decoder to focus on the word “etudiant” (“student” in french) before it generates the English translation. This ability to amplify the signal from the relevant part of the input sequence makes attention models produce better results than models without attention.

Time step: 7

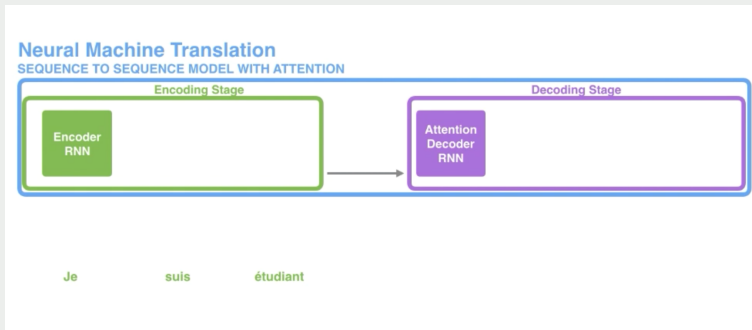
Neural Machine Translation SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Attention Model

An attention model differs from a classic sequence-to-sequence model in two main ways:

- First, the encoder passes a lot more data to the decoder. Instead of passing the last hidden state of the encoding stage, the encoder passes all the hidden states to the decoder:



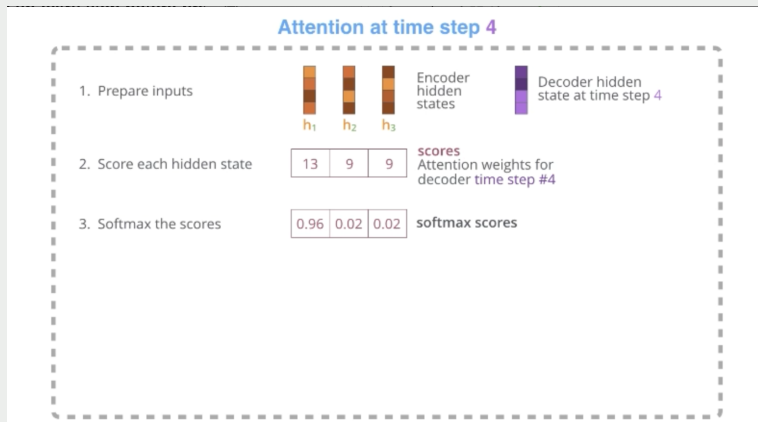
Attention Mechanism

- Second, an attention decoder does an extra step before producing its output. In order to focus on the parts of the input that are relevant to this decoding time step, the decoder does the following:
 - ① Look at the set of encoder hidden states it received – each encoder hidden states is most associated with a certain word in the input sentence
 - ② Give each hidden states a score (let us ignore how the scoring is done for now)
 - ③ Multiply each hidden states by its softmaxed score, thus amplifying hidden states with high scores, and drowning out hidden states with low scores
- This scoring exercise is done at each time step on the decoder side.



Attention Mechanism

- Second, an attention decoder does an extra step before producing its output. In order to focus on the parts of the input that are relevant to this decoding time step, the decoder does the following:



Attention Decoder

The decoder does the following:

- ① Look at the set of encoder hidden states it received - each encoder hidden states is most associated with a certain word in the input sentence
- ② Give each hidden states a score (let's ignore how the scoring is done for now)
- ③ Multiply each hidden states by its softmaxed score, thus amplifying hidden states with high scores, and drowning out hidden states with low scores

This scoring exercise is done at each time step on the decoder side.



Attention Process

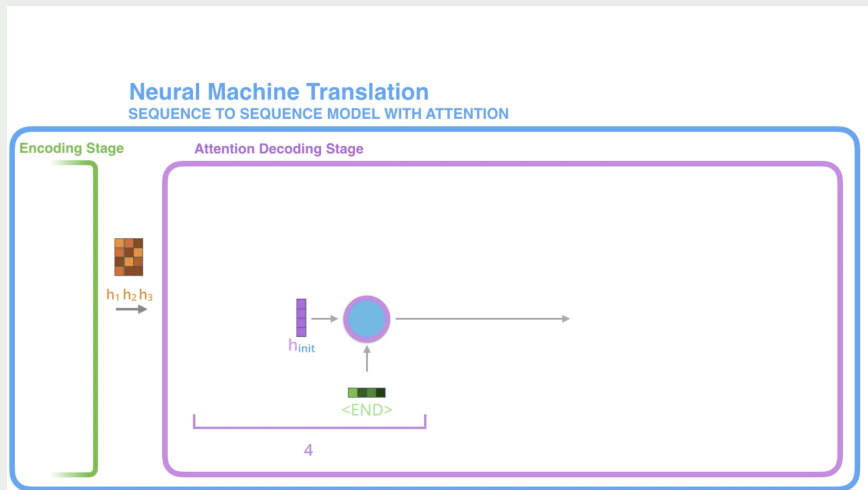
□ How the attention process works:

- 1 The attention decoder RNN takes in the embedding of the END_i token, and an initial decoder hidden state.
- 2 The RNN processes its inputs, producing an output and a new hidden state vector (h_4). The output is discarded.
- 3 Attention Step: We use the encoder hidden states and the h_4 vector to calculate a context vector (C_4) for this time step.
- 4 We concatenate h_4 and C_4 into one vector.
- 5 We pass this vector through a feedforward neural network (one trained jointly with the model).
- 6 The output of the feedforward neural networks indicates the output word of this time step.
- 7 Repeat for the next time steps



Attention Process

Let us now bring the whole thing together in the following visualization:



Attention Process

- This is another way to look at which part of the input sentence we're paying attention to at each decoding step:

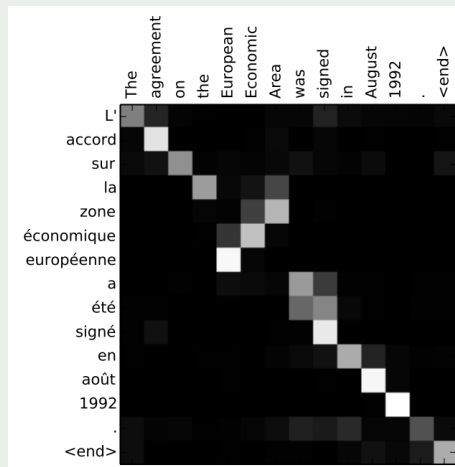


- Note that the model isn't just mindless aligning the first word at the output with the first word from the input. It actually learned from the training phase how to align words in that language pair (French and English in our example).



Attention Process

- An example for how precise this mechanism can be comes from the attention papers:



- You can see how the model paid attention correctly when outputting "European Economic Area". In French, the order of these words is reversed ("europenne conomique zone") as compared to English. Every other word in the sentence is in similar order.

