JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Language Translator using Machine Translation with Attention Mechanism

Anshul Agrawal

Software Engineer

Abstract: With the borderless boundaries, many people work or communicate with people from different country who speak different language. And because of this language barrier the information is not delivered properly or not received the correct rational. The best possible solution is that people must be multilingual, but this is not a feasible solution. And to help these people now days we have several language translation systems available. This paper proposes Sequence-to-Sequence translation models. A typical Seq2Seq models consists of an encoder and a decoder which are themselves two separate neural networks combined into a single giant network

Index Terms - Language Translator, Machine Translation, French to English translation

I. INTRODUCTION

Sequence-to-Sequence model is basically a learning or training models which convert the sequences from one domain to another domain, irrespective of the sequence length. Sequence-to-Sequence model comprises of an encode and a decoder which themselves are neural networks combined into a single giant network. These encoder and decoder are typically LSTM (Long short-term memory) model.

The main application of Sequence-to-Sequence model are Neural machine translation, Image captioning, speech recognition, chat-bot, time-series forecasting etc.

The encode network must take the input and understand the sequence and convert it into a small dimensional representation. Which is then used by the decoder network that generates the output. For e.g.: - input to the encoder may be an encoded sentence (in case of neural machine translation) or image features (in case of image captioning) or even sound waves (in case of speech recognition) Fig 1, Fig 2, Fig 3 respectively.

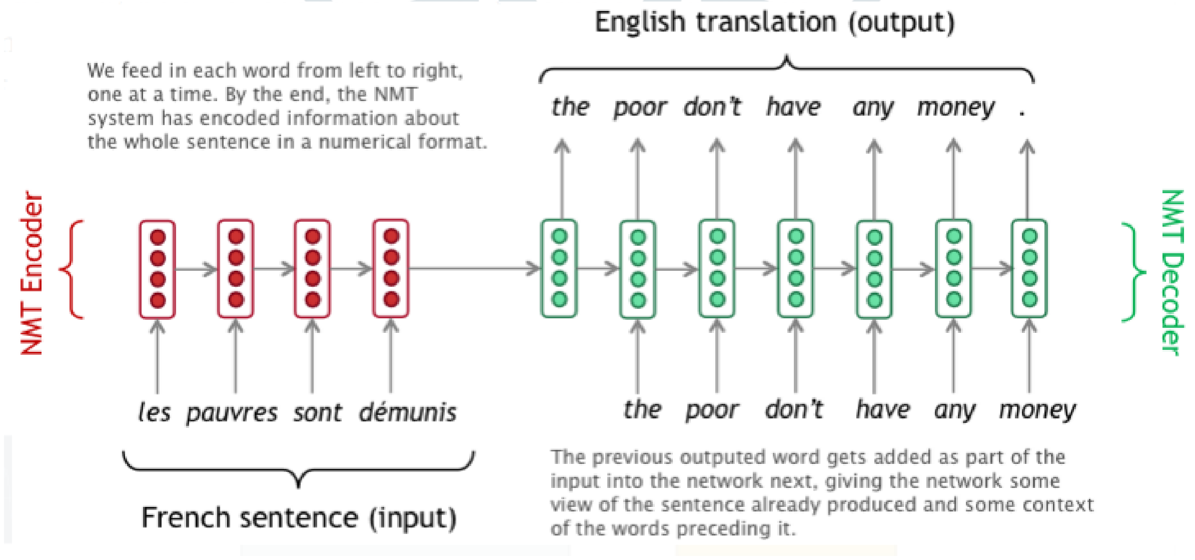
English translation (output)
We feed in each word from left to right, one at a time. By the end, the NMT system has encoded information about the whole sentence in a numerical format.
the poor don't have any money .
NMT Encoder
NMT Decoder
les pauvres sont démunis
the poor don't have any money
French sentence (input)
The previous outputed word gets added as part of the input into the network next, giving the network some view of the sentence already produced and some context of the words preceding it.

Fig.1 Neural Machine Translation

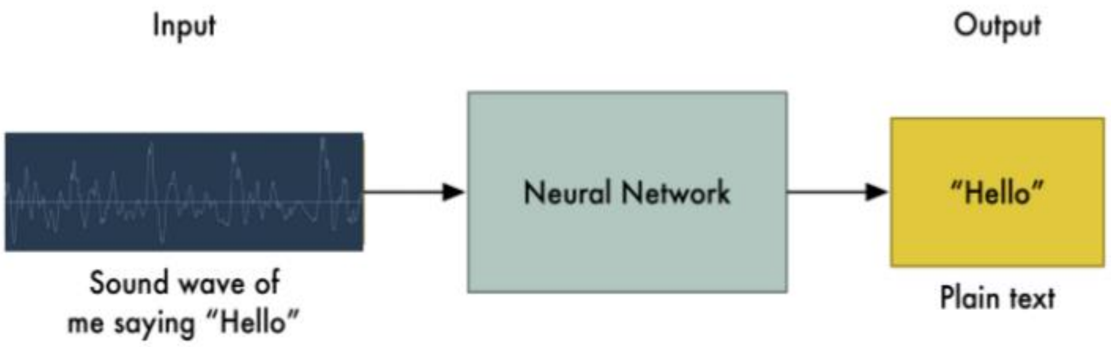

Fig.2 Speech Recognition

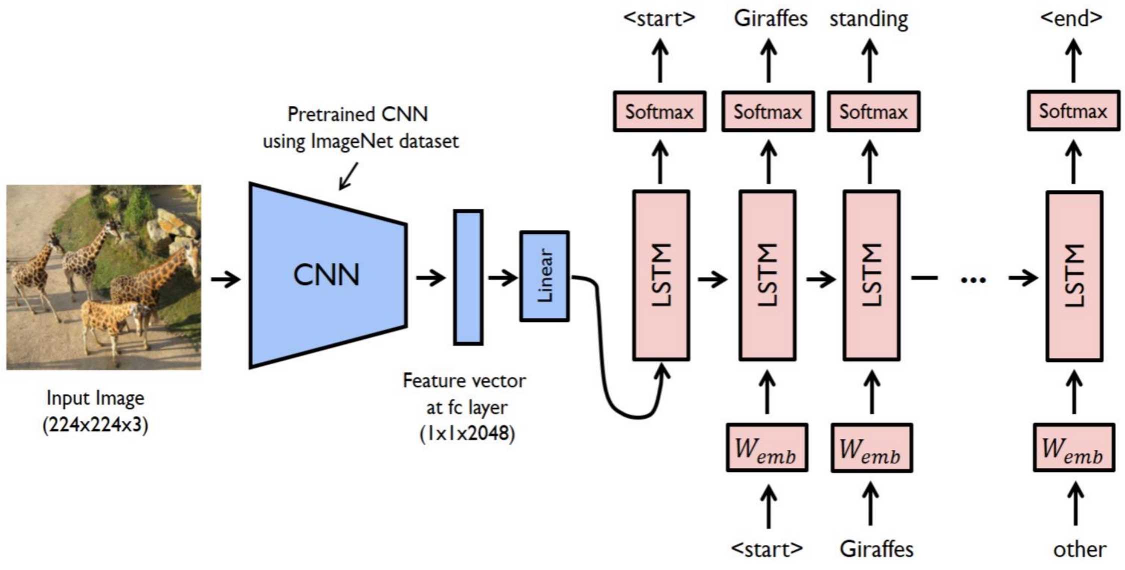

Fig.3 Image Captioning

II. Utility

The model proposed in this paper can being utilized across the globe in several different manner. The machine translation is used in the industry for the business purpose and the online/apps for commercial use. For e.g.: - Kantan machine translation, Systran, Canopy Innovations (Healthcare), Lingua Custodia (Finance) etc.

III. Related Work

Any language translation mode uses NLP (Natural Language Processing). It's a subset of linguistics, computer science, and artificial intelligence which focus on the interaction between computers and human language. With the help of the NLP, we can process the language and extract multiple things like rational of the statement (tone was a happy, sad, angry etc.), can remove the useless words and symbols. Neural Network and Recurrent Neural Network also helps in deduction the pattern in the statement.

IV. Proposed Model

Fig 4 is the architecture of the proposed model. The first step is to remove non-letter character and then normalizing tokenizing and converting words to index. Once the input data is cleaned then the processed data is then feedforward to the encoder which will give a vector and a hidden state value for each word, which will be useful for the next set of input.
The input data then make their way to the decoder, which uses the encoder's output as input before producing words that are then put to produce a translation. Prior to this, though, the Attention Mechanism, which is positioned above the decoder, is crucial because it enables the decoder to pay attention to different elements of the output provided by the encoder at each stage of the decoder's output. The decoder then forecasts the eventual translation of the words.

4.1Encoder

Below is the encoder architecture diagram (Fig. 5). For now, let's consider our input sentence has 'n' words. And for simplicity we have only single sentence.

- **Xi**→ The input at each time stamp will be each word in the phrase because we're going to utilize word-level encoding. This indicates that X1 = "Je," X2 = "ne," and so on, up to X5 = "anglais." The input at each time stamp would have been a single character if we were utilizing a character-level encoder-decoder model: X1: "J," X2: "e," X3: "n," and so on. A vector is used to represent each word. Each word is changed to its word index equivalent from the corpus for this. The word index of the most frequent words is lower than that of the least frequent ones.

- **'hi' and 'ci'**→ Following the time stamp I are the hidden state and context vectors. Simply put, these vectors show what the encoder has observed up to this time stamp. For instance, the network has already seen "Je ne parle," as h3 and c3 will recall. Each of these vectors has a length equal to the number of LSTM/GRU units. The decoder's initial states are the state that was retrieved following the most recent time stamp.
- **'Yi'** → Finished at time stamp i. Using the SoftMax activation function, the probability distribution for the full language is produced. The outputs from the encoder network will be discarded because we don't need them. We only care about the hidden/context vectors as encoder outputs.

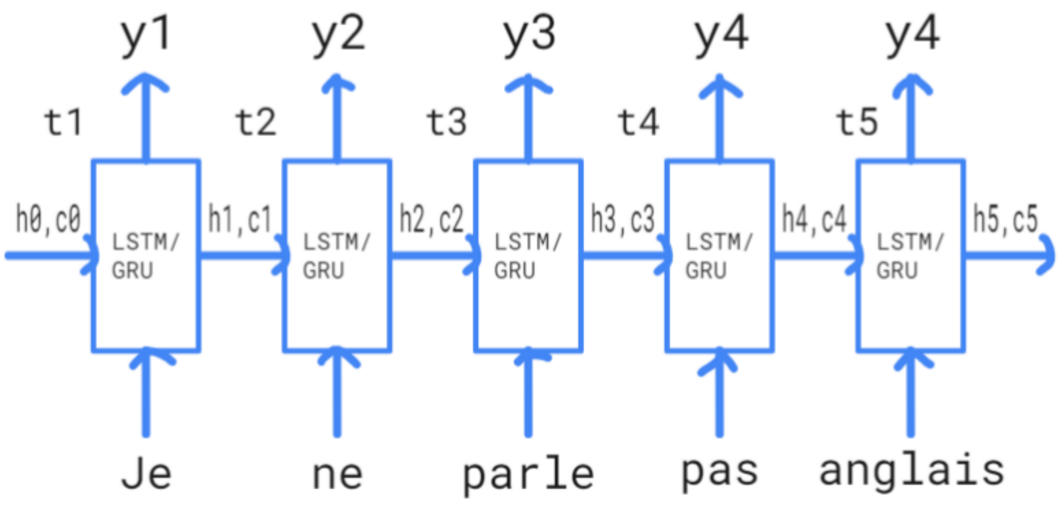

Fig. 5 Encoder Architecture

4.2 Decoder

Let's move on to the next section after learning about the encoder: the decoder network. Decoders behave differently during the training and inference phases than encoders do. Additionally, for the reasons outlined below, we must add two special tokens to the output sentence. "start>" (found at the start of the string) and "end>" are these tokens (at the end of the string).

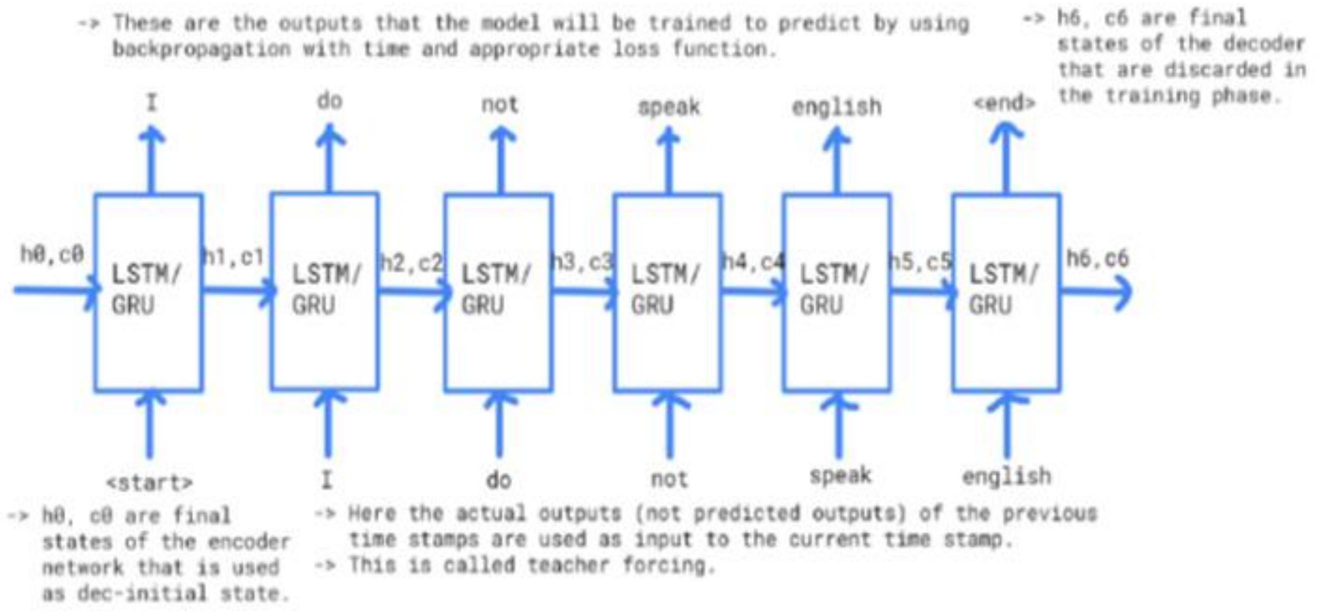
-> These are the outputs that the model will be trained to predict by using backpropagation with time and appropriate loss function.
-> h6, c6 are final states of the decoder that are discarded in the training phase.
I
do
not
speak
english
<end>
h0,c0
h1,c1
h2,c2
h3,c3
h4,c4
h5,c5
h6,c6
LSTM/ GRU
<start>
I
do
not
speak
english
-> h0, c0 are final states of the encoder network that is used as dec-initial state.
-> Here the actual outputs (not predicted outputs) of the previous time stamps are used as input to the current time stamp.
-> This is called teacher forcing.

Fig. 6 Encoder Architecture

4.3 Attention

- Everything up to this point has been a straightforward encoder-decoder model without the attention mechanism.
- This model's main flaw is that as they start processing more information, people tend to forget what came before. Longer sentences shouldn't use it. Look at the illustration below.

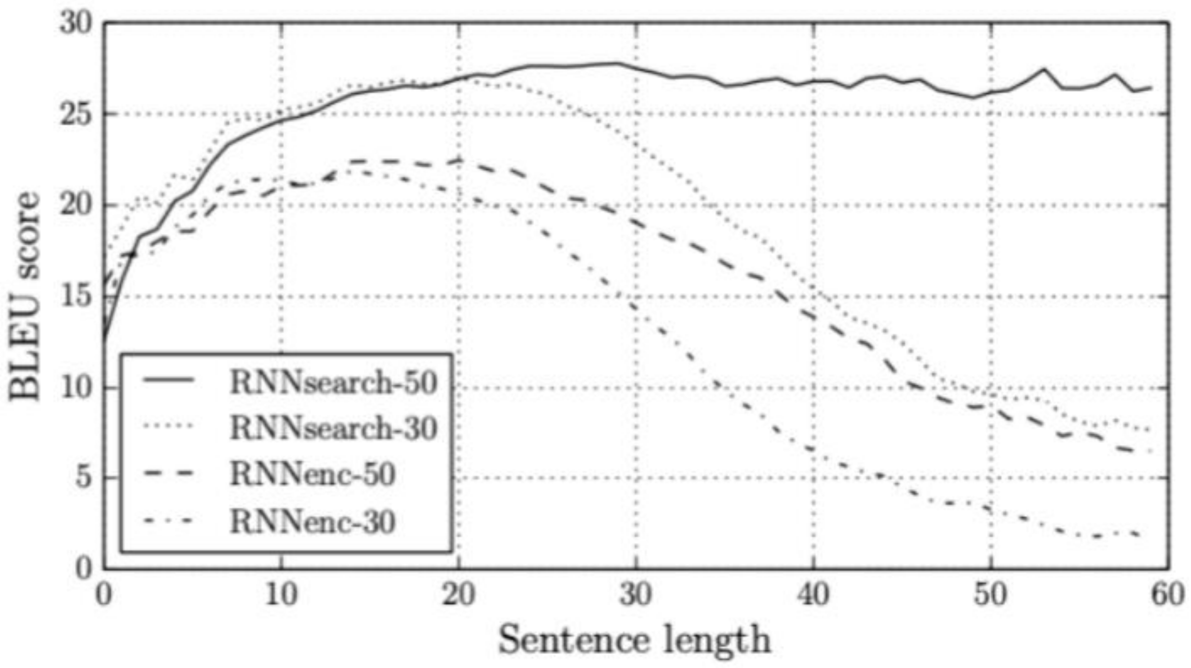

Fig. 7 Sentence Length vs BLEU Score

- A plot of sentence length and BLEU score is shown in the diagram above. The first model (RNNSearch-50), in contrast to the other three models, uses an attention mechanism.
- We can plainly observe that for the last three models, BLEU scores fall as sentence length rises, but for RNNSearch-50, they remain stable. As a result, paying attention is a highly important factor while working with lengthier sequences. We'll talk about it in this part.

4.4 Working of Attention

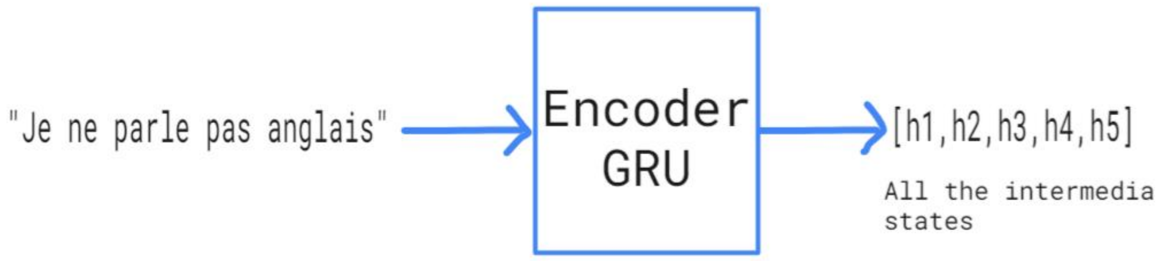

Fig. 8 Encoder GRU

- **Getting encoder hidden states: -** First, the encoder is used to acquire the concealed states following each time stamp, as illustrated in the above diagram.
- **Setting encoder final state as decoder initial state: -** The decoder won't have any internal states now because we are attempting to anticipate the first word of the output sequence. Due to this, the initial decoder state will be the final encoder states (h5).
- **Computing Scores: -** We now train a straightforward feed-forward neural network with all the encoder states and the current decoder state, which will learn to recognize important encoder states by producing high scores for them and lower scores for the irrelevant states. For instance, the important information for the word "say" prediction can be found in stages h1, h2, and h3, whereas the remaining states h4 and h5 may not be significant. Our feed-forward-NN will learn to give the first three states high scores and the last two states lesser points. Let these scores be, successively, [s1, s2, s3, s4 and s5].

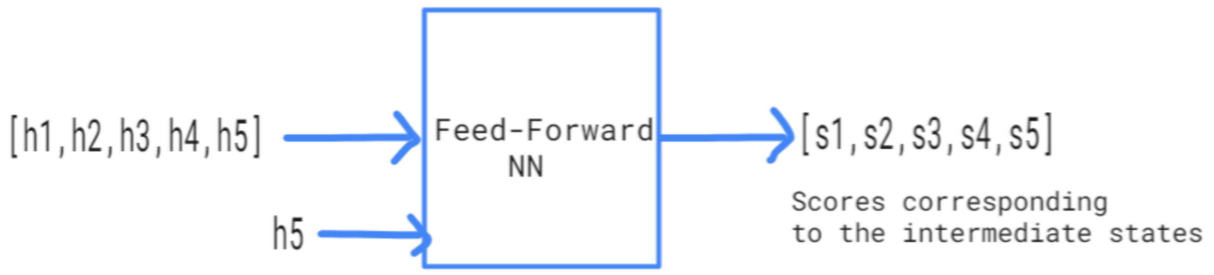

Fig. 9 Computing Scores

- **Getting attention weights: -** The "SoftMax" function receives the scores from the previous step to produce the attention weights. We'll use the following weights: e = [e1, e2, e3, e4, e5]. All of these weights added together equal one. They provide a nice probabilistic interpretation as a result.

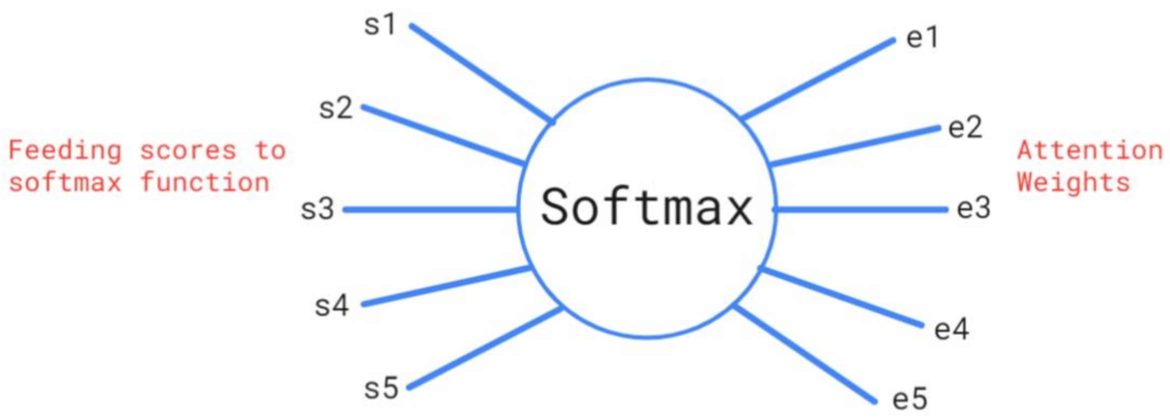

Fig. 10 getting attention weights from SoftMax

- **Computing context vector: -** Context vector is computed as follows after receiving the attention weights. To forecast the subsequent word, it will be utilized as a decoder.
  context-vec (cv) = e1*h1 + e2h2 + e3*h3 +e4*h4 + e5h5

V. Conclusion

- Test data translations are generally reliable. Some of them fall short of expectations.
- For better translations, we can add more data and run it for a lot more epochs.
- Additionally, we can employ several attention score functions (dot and general score functions).
- Translation can be significantly enhanced by switching to using LSTM with bidirectional wrapper rather than GRU.

VI. References

[1] https://arxiv.org/pdf/1409.0473.pdf
[2] https://arxiv.org/pdf/1409.0473.pdf
[3] https://arxiv.org/pdf/1409.0473.pdf
[4] Keras tutorial for seq-to-seq learning.
[5] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Attention
[6] https://medium.com/@martin.monperrus/sequence-to-sequence-learning-program-repair-e39dc5c0119b (Using seq-to-seq learning for program repair).
[7] https://towardsdatascience.com/light-on-math-ml-attention-with-keras-dc8dbc1fad39