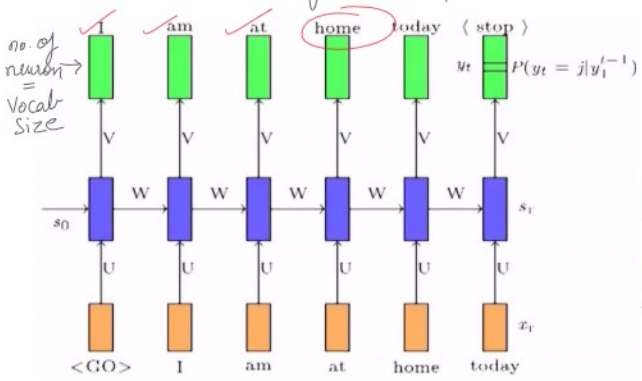
RNN NLP Part 4

11 October 2023 11:46 AM

Let's revisit the task of auto completion, where in the task is to predict the next word given a word as input.



<GO>, <stop> → are special words to tell/signify the start and end respectively for text completion.

Vocab = { I, at, today, home, am }

Q By looking at the vocab, tell how many neurons will be present in the outer layer?

Ans 5 neurons

j=0	0.1	I
j=1	0.1	at
j=2	0.2	today
j=3	0.1	home
j=4	0.5	am

{Softmax Activation}

← final prediction as the probability is the highest

So at any time step 't', we want to compute:

{ y_t → output at t^{th} time step }

max will only give the highest probability, arg max will give the value of 'j' for which the probability is highest.

$$\arg\max_{j \in V} P(y_t = j \mid y_{t-1}, y_{t-2}, \cdots, y_1)$$

j → word, V → Vocabulary

What is the probability that at time step 't' the word is j { j = 0, 1, 2, 3, 4 } given all the previous words $y_{t-1}, y_{t-2}, \cdots, y_1$

Eg $P(y_t = \text{home} \mid \text{at, am, I})$

and using RNN we are trying to model the above probability.

So using RNN,

{ Modelling choice: like in Machine Learning we assume that "y" is a linear composition of inputs, and using this we try to find the best model }

$$P(y_t = j \mid y_{t-1}, y_{t-2}, \cdots, y_1) = [\text{softmax}(vS_t + c)]_j$$

S_t captures all the information until time step 't'.

Softmax in last layer.

Vector

This vector is trying to predict

j=0	0.1	I
j=1	0.1	at
j=2	0.2	today
j=3	0.1	home
j=4	0.5	am

finds this 'j' value

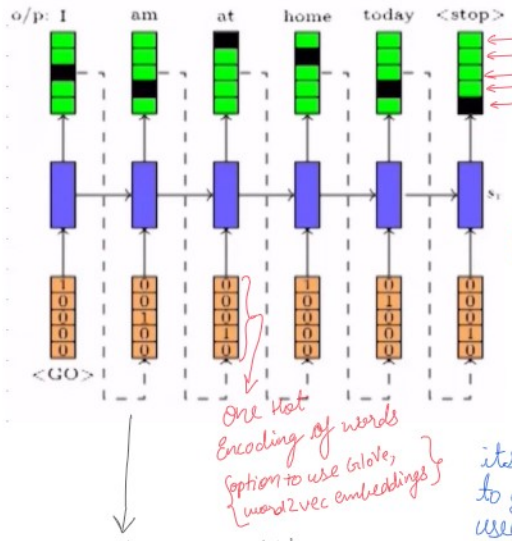
Because we want to find probability of j^{th} neuron, which contains the j^{th} word.

↳ this entire thing

☆ Now, we know that RNN has a limit and cannot just keep on accumulating information without the drawbacks

☆ Here the RNN is trying to learn a probability vector over the entire vocabulary and it tries to model all that into the hidden state S_t.

£ Input for Network



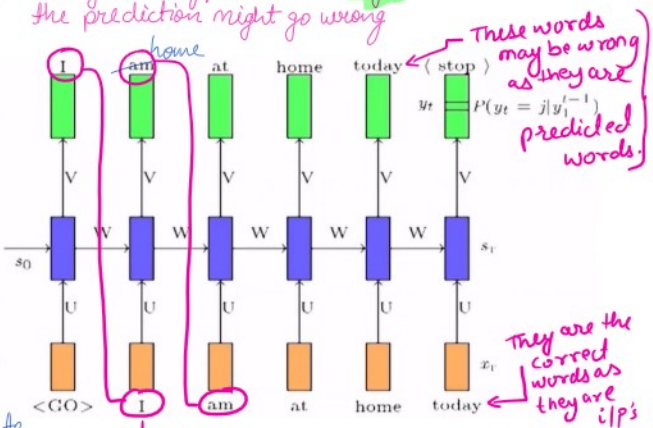
Every position is dedicated to a word and the word with highest probability is selected

→ Let's say during the training time the word "home" is predicted inplace of "am", then it doesn't mean we will put "home" in the next time step. "home" will have a loss and will update its parameters during backpropagation to get the correct prediction but won't be used in next time step.

One hot Encoding of words (option to use Glove, word2vec embeddings)

However at test time 't', whatever the prediction it will be used in the next time step.

* During training phase there might be a case where the prediction might go wrong



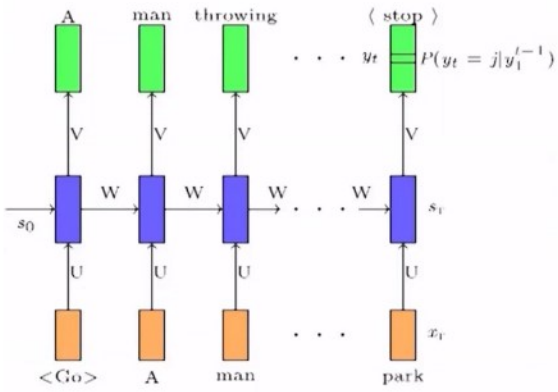
These words may be wrong as they are predicted words.

They are the correct words as they are i/p's

→ So there is no reason for these 2 words to be exactly same.

$ Image Captioning

Task: Given an image as input, the task is to put a caption on the image describing the image.



A man throwing a frisbee in a park

In layman terms, we want to generate a sentence given an image.

We are interested in

$$P(y_t = j \mid y_{t-1}, y_{t-2}, \dots, y_1, I)$$

(I → Image)

So basically, given the image information and all the previous words that have been predicted, what is the probability of next word.

⇓

Now the reason for doing this is because we are not going to predict the entire caption in one go, we do so word by word. because the natural language has a sequential structure i.e. the sequence of the word matters. for eg. a park throwing a frisbee in a man, has the same permutation of the vocab but is absolutely wrong in meaning as the sequence of the words is not correct. Basically whatever has been said / seen previously decides / determines the next word.

Q What kind of architecture can be used for the above task?

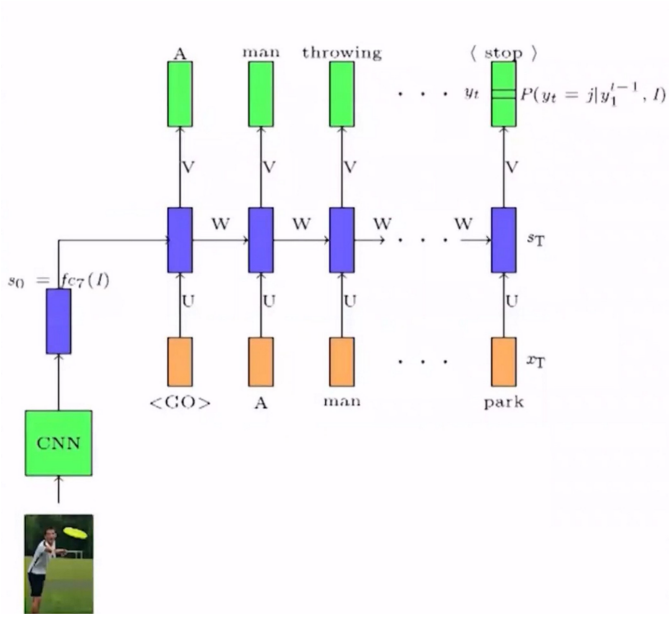
Ans Since caption generation problem has a sequential / temporal nature → RNN can be used

But, the input consists of an image: So to learn good feature representations → CNN can be used.

So, input → CNN
output → RNN

P.T.O.

Since, CNN architectures are good for images, we utilize them to learn important features from the image and then pass it on to the RNN architecture/model

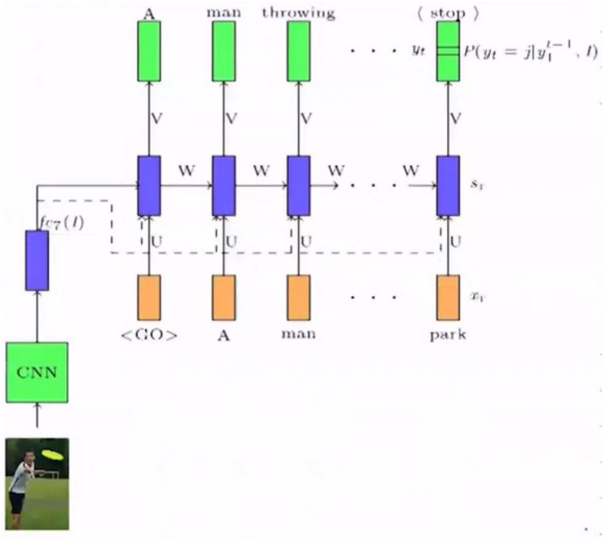


Rather than passing those features to CNN, it sends to RNN

CNN learns the features of the input image

Option - 1

↳ Output of CNN is directly passed to time step - 1

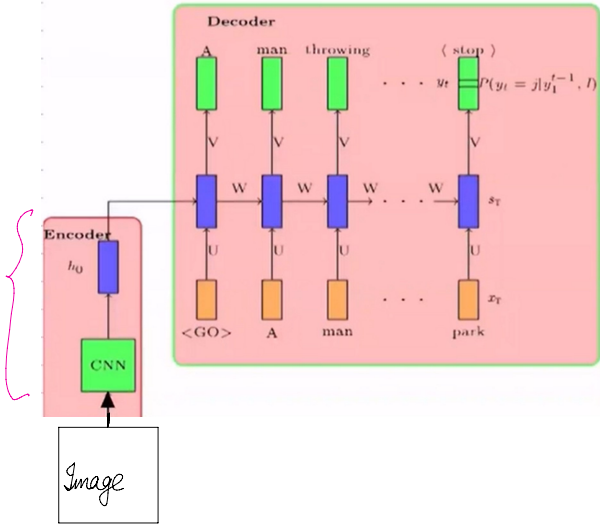


→ Option - 2

↳ Output of CNN is directly passed to all time steps

So in general the architecture will look like ↓

Encoder Decoder Architecture



The decoder recieves the input from encoder and produces the output.

Learn the important and good representation of the input and pass it to decoder

In the example discussed above : Encoder → CNN, Decoder → RNN } Now this is not all permanent, depending upon the task that we are trying to solve, we would like to use different encoder, decoder architecture

In image captioning,
→ A CNN is used to "encode" the image
↳ learn good feature representation of the input.
→ An RNN is used to "decode" a sentence from this encoding.

Task: Image Captioning

Data: $\{x_i = \text{image}, y_i = \text{caption}\}_{i=1}^{N}$

Model: It has two parts: Encoder: $S_0 = CNN(x_i)$ ← Refers to a very deep network with many filters and pooling layers

Decoder: $S_t = RNN(S_{t-1}, y_{t-1})$ } We are using this to make a prediction over the probability of all possible words

S_{t-1}: Previous time step. y_{t-1}: previous predicted word

Parameters of the model: U, V, W, b, C and all weights and biases of CNN

and then, we train ALL the parameters of Encoder Decoder Together

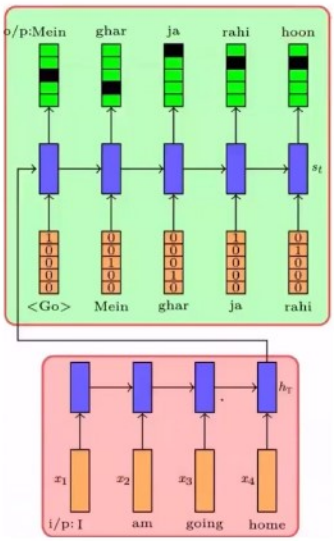
NOTE: The training of encoder & decoder is not separate, the entire architecture is trained at once.

This type of model is also called as end to end model.

Real Life Use Case of Encoder Decoder

→ Machine Translation

Translate english to hindi



Encoder → RNN → To handle the text input
Decoder → RNN → To convert & give text output

Decoder { Using the representation, it tries to predict the output }

Encoder { It tries to make a good representation of the input by understanding the syntax & semantics and pass it to decoder }

→ Video captioning

↳ Video is a sequence of images, and now for that entire sequence we want to put up a caption

↳ since CNN is good for images we use that, but since the images are in a sequence, i.e. the current frame influences the next frame, we put the entire thing in an RNN

⇒ Encoder = (CNN + RNN)