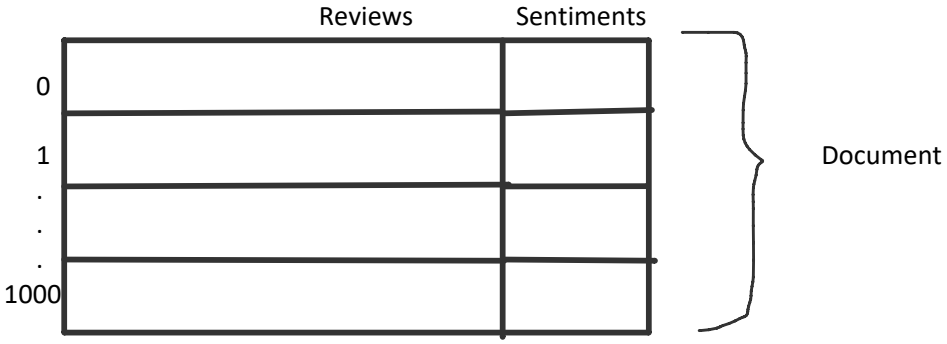
NLP

Thursday, August 5, 2021 8:11 PM

- How to treat the text data in columns ?

1. Bag of Words (Bow) :



Combinations of Documents are called Corpus.

1. There are simple text cleaning techniques that can be used as a preprocessing step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don't contain much information, called stop words, like "a," "of," etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. "play" from "playing") using stemming algorithms.

2. Second, we try to learn Vocabulary. Vocabulary contains unique words of corpus. For Example,

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,

['best', 'times', 'worst', 'age', 'wisdom', 'foolishness']

3. We have to make Documents Vector from Corpus.

best	times	worst	age	wisdom	foolishness
1	1	0	0	0	0
0	1	1	0	0	0
0	0	0	1	1	0
0	0	0	1	0	1

Disadvantages of Bag of Words :

1. It just takes frequency of words, it has no meaningful extraction like other advanced methods. It does not make semantic relationships with other tokens.

TF-IDF(Term Frequency - Inverse Document Frequency) :

Sentences

D_1 = Prime Minister India

D_2 = Prime Minister Nepal

D_3 = Corona Virus India

Freq

Prime → 2

Minister → 2

India → 2

Nepal → 1

Corona → 1

Virus → 1

Term Freq ⇒ $\frac{\text{No of Repetation of Word in a Sentence}}{\text{No of Word in a Sentence}}$

Inverse Document Freq $= \log\left[\frac{\text{No of sentences}}{\text{No of sent Containing tokens}}\right]$

TF

Words	D_1	D_2	D_3	IDF
Prime	0.33	0.33	0	0.176
minist	0.33	0.33	0	0.176
India	0.33	0	0.33	0.176
Nepal	0	0.33	0	0.477
Corona	0	0	0.33	0.477
virus	0	0	0.33	0.477

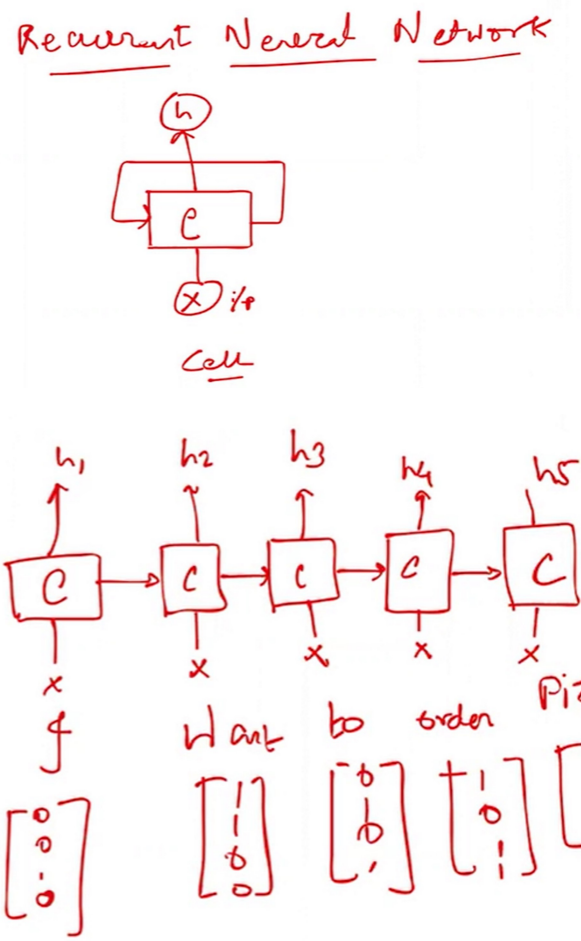
TF-IDF	Prime	Minister	India	Corona	Virus	Nepal
D_1	0.058	0.058	0.058	0	0	0

TF-IDF	Prime	Minister	India	Corona	Virus	Nepal
D1	0.058	0.058	0.058	0	0	0
D2	0.058	0.058	0	0	0	0.15
D3	0	0	0.058	0.15	0.15	0
	0.116	0.116	0.116	0.15	0.15	0.15

Those who have same values are interlinked with each other and higher values are most frequent.

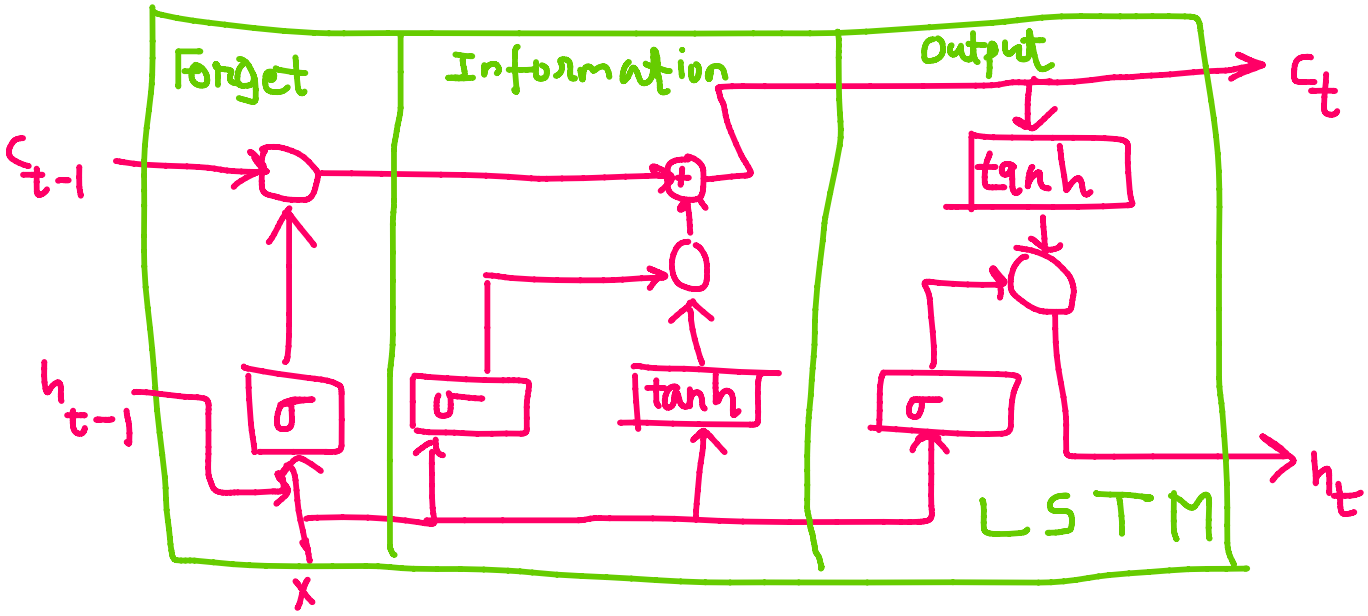
RNN(Recurrent Neural Network):

It is deep learning approach to solve text related problems. It provides semantic relationships between tokens that vectorizers not provide. It helps to achieve meaningful extraction from words.



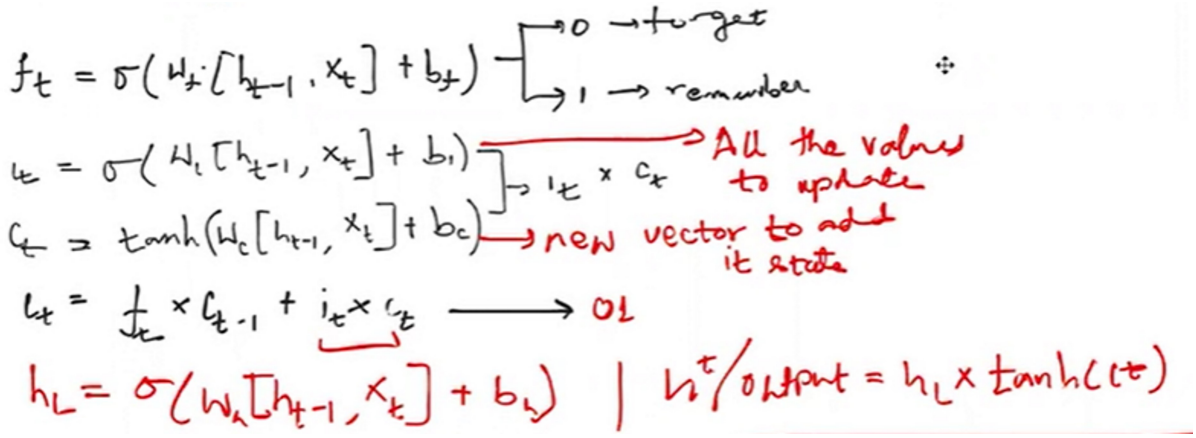
If we get very big sentence, it cannot handle such sentences. It starts getting vanishing gradients because it not remember large sentences.

LSTM(Long Short Term Memory):
It has ability to forgot unwanted tokens and stores only necessary tokens. It provides weightage to necessary tokens.



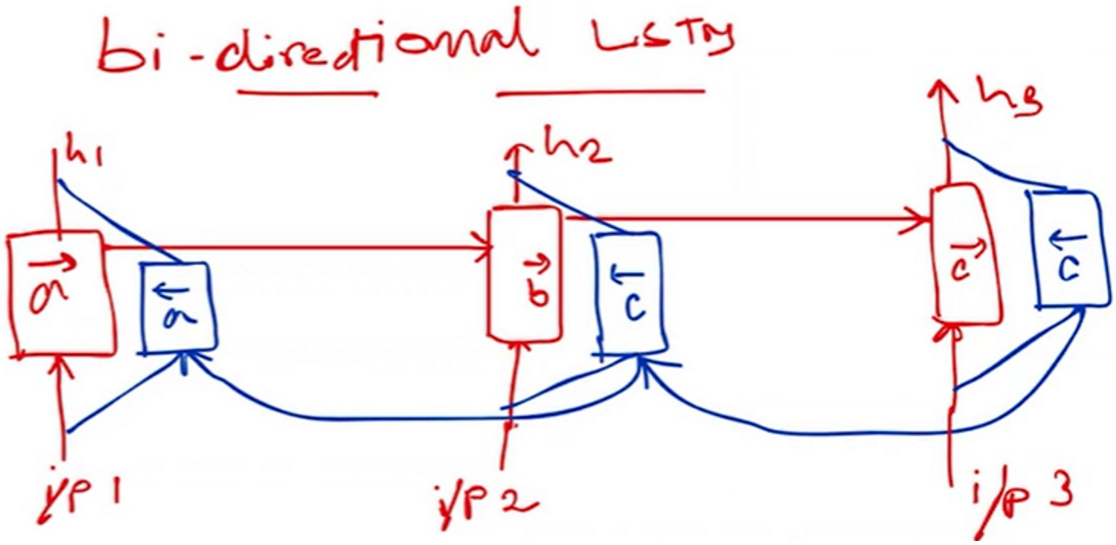
Ct :- Cell State or weights
ht-1 :- Hidden State or inputs from previous cell
X :- inputs

Forget layer :- used to forget unwanted tokens
Information layer :- stores important tokens
Output layer :- combines current input with previous input and provides o/p to next cell



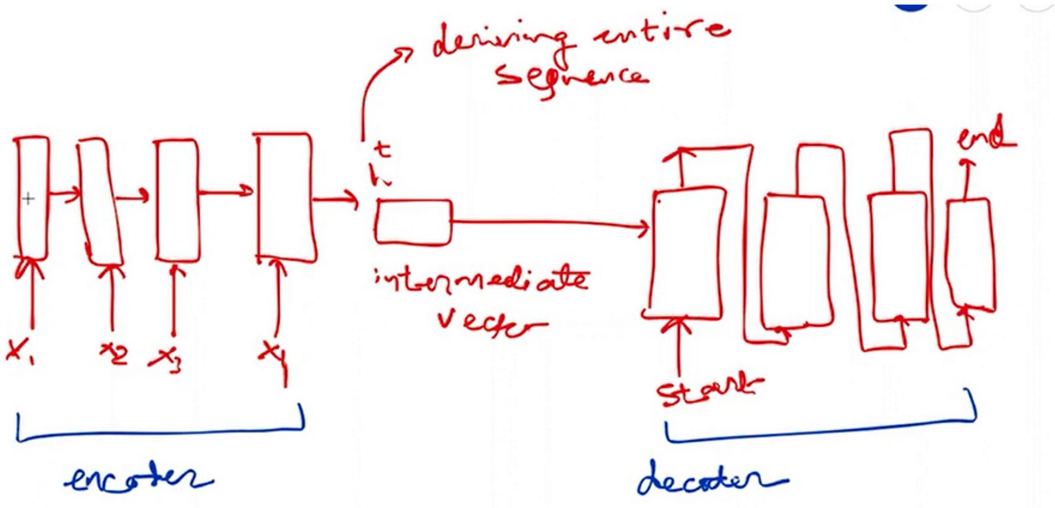
Disadvantages of LSTM:
LSTM is good when we need only sequence of sentence but it fails in case we want to go reverse. We can't able to relate a token with its previous token. We cannot change vector of previous token based on current token.
For Example,
Tite Kubo's earliest influence is from Shigeru Mizuki's manga Gegege no Kitaro. He remembers trying to sketch its characters and found his own designs to be simpler then that of Mizuki's.
In above case, LSTM cannot relate '*He*' with '*Tite Kubo*' here. It only moves in forward direction.

Bidirectional LSTM :



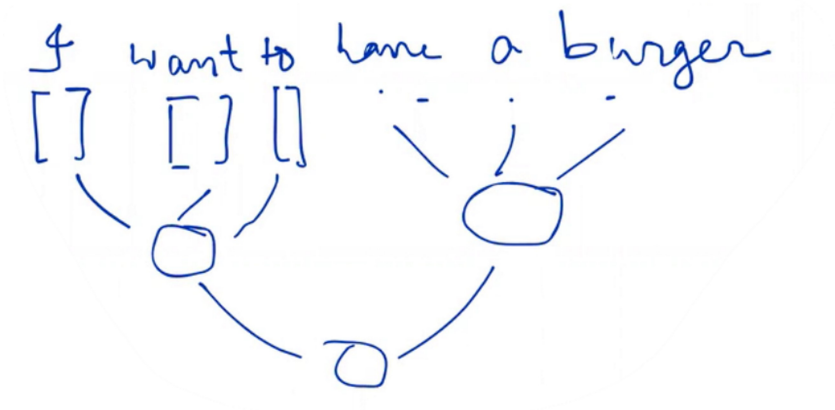
Disadvantages of Bidirectional LSTM:
If we have very long sentences or if we have multiple references in text. It couldn't able relate it with text. For Example, if we have multiple person name in text it not able to relate it with text.

Encoder & Decoder :

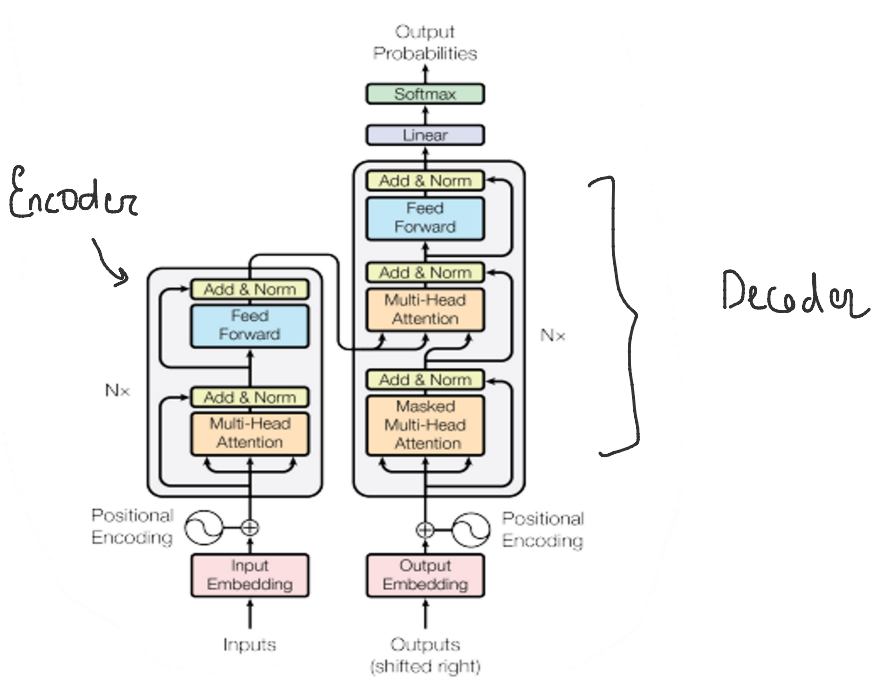


Encoder can be RNN or LSTM used to take input, it not generate any intermediate hidden state output and directly generate final output which is intermediate vector, is transferred to decoder.
Decoder has multiple LSTM cells which are connected which helps us to generate final output.
The Application of Encoder and Decoder is Neural Machine Translation which is translation of language to other.

Attention is all you need:



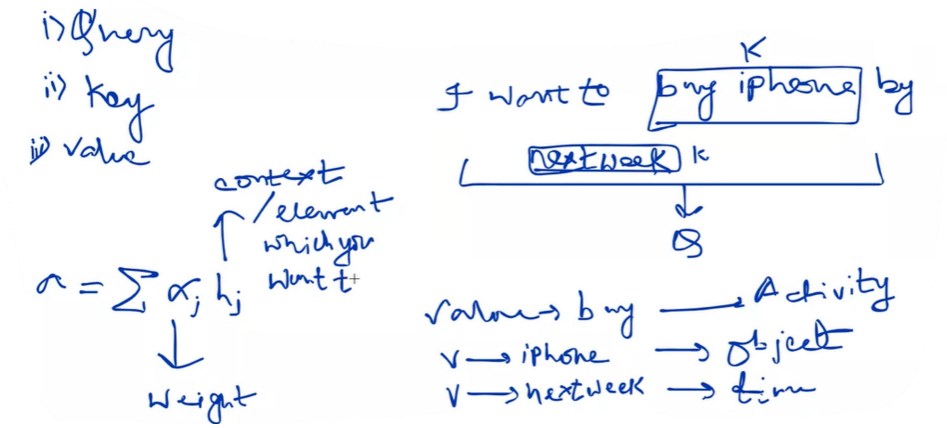
Like this we have multiple sentences where we assign each token a attention vector, Multiple tokens forms another vector and multiple sentences forms another vector. In this way, attention vector gives some weightage to each token and more focus on important vectors.



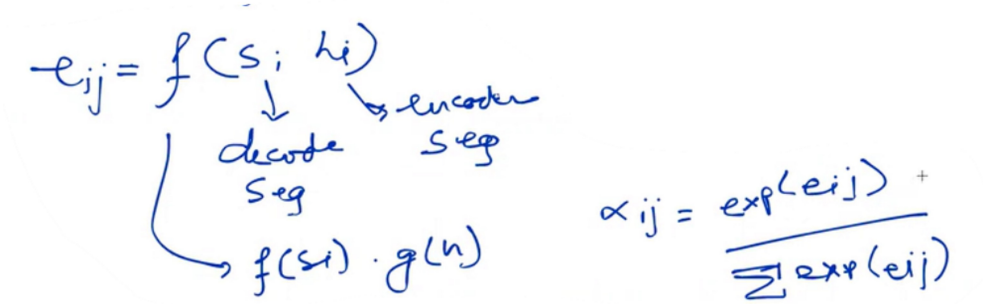
Positional Encoding : includes (i) Sentence embedding : Finds no of sentences
(ii) Positional Embedding : finds semanticness between the tokens
(iii)Token Embedding : token embedding

Multi-Head Attention : It provides some attention vector to each tokens, multiple tokens having different attention. That's why it is called Multi Head Attention.

Masked-Multi Head Attention : It mask or omit the important token/ word to decoder so it can learn it. Some time it omit few tokens next time it omit next few token.



a - attention vector



Softmax = $\frac{QK}{\sqrt{d}}$. a Q - no of query , K - no of keys, d - no of tokens and a - attention vector