Deep Learning for NLP - 2

About the Module

- Sequence to Sequence Models
- Recurrent Neural Networks
- Long Short Term Memory Units
- LSTMs for Text Generation

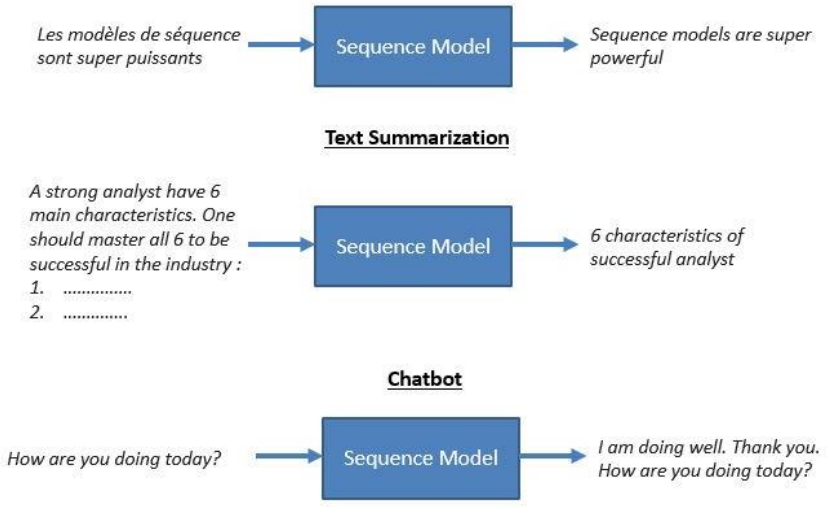
Sequence to Sequence Models

- Sequence model learns the probability of occurrence of an element based on previous elements
- Sequence Data Examples :

- Audio, Signals, Time Series
- Text: Sequence of words

Applications with NLP

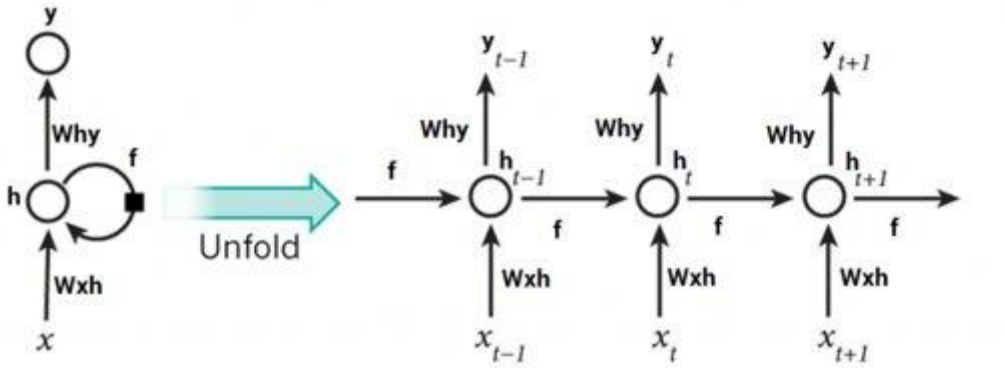
- Machine Translation
- Text Summarization
- Chatbots
- Text Generation



Recurrent Neural Networks

- Feed-Forward Networks - Activation outputs propagates only in one direction

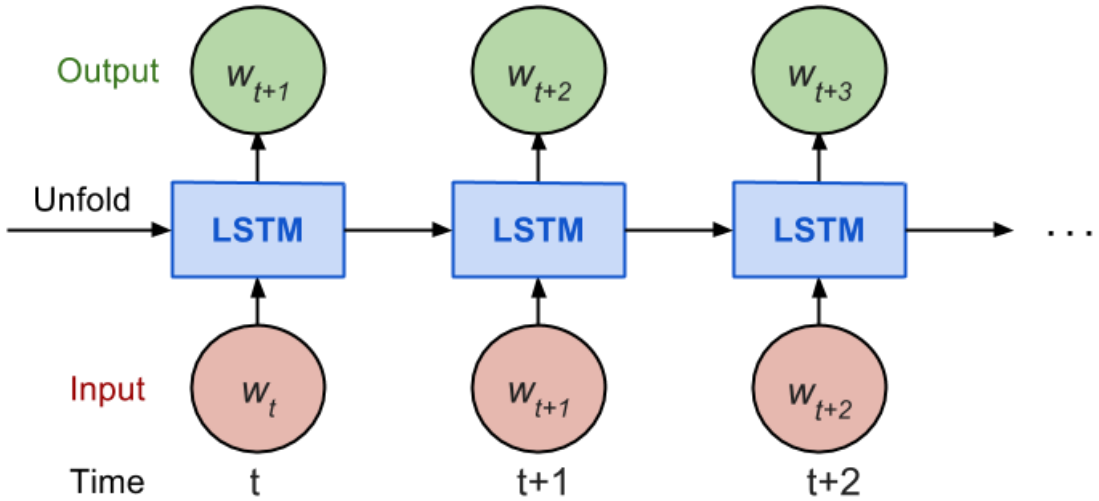
- Recurrent Networks - Activation outputs propagates in both directions
  - From Inputs to Outputs, From Outputs to Inputs



- Loops act as 'memory state' of the neurons
- Allows the neurons an ability to remember what have been learned so far
- Vanishing Gradient problem : network tends to forget was learned in the earlier layers

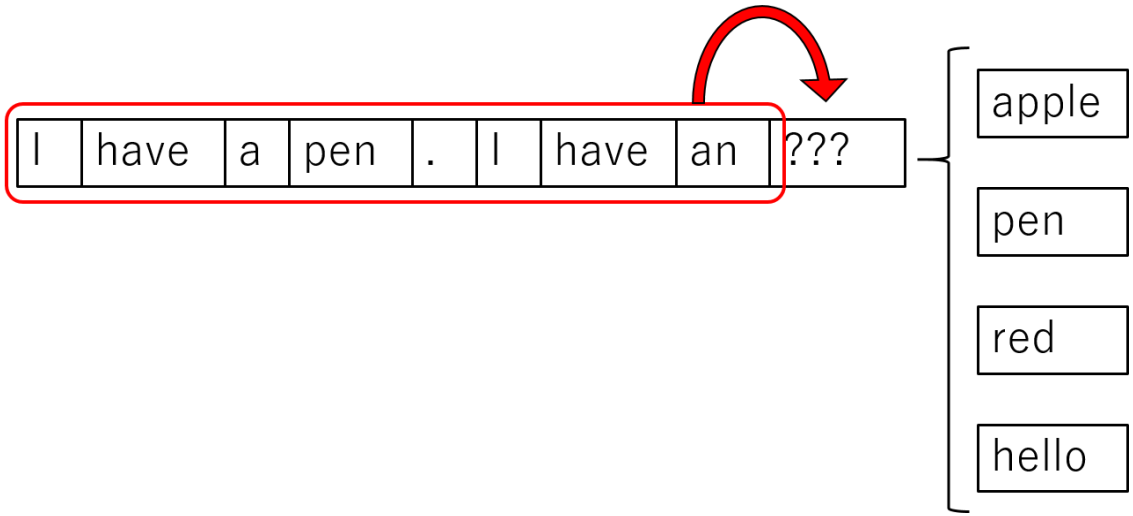
Long Short Term Memory (LSTMs)

- LSTMs are a special types of RNNs with an additional state 'cell state'
- Used to make adjustment in the information flow.
- With cell state, model can remember or forget the leanings more selectively.



LSTMs for Text Generation

- Predict the likelihood of occurrence of a word based on the previous sequence of words



Dataset Preparation

Create Word Index
Text: I will play football today

Tokens : [I, will, play, football, today]
Token Index : [0, 1, 2, 3, 4]

Convert text into sequence of tokens
N grams: Sequence of Tokens

Ngram	Sequence of Tokens	Padded Sequence
i will	[0, 1]	[0, 1, -1, -1, -1]
i will play	[0, 1, 2]	[0, 1, 2, -1, -1]
i will play football	[0, 1, 2, 3]	[0, 1, 2, 3, -1]
i will play football today	[0, 1, 2, 3, 4]	[0, 1, 2, 3, 4]

LSTM Model Architecture

Input -> Embedding -> LSTM -> MultiClass Classification

```
model = Sequential()
input_len = max_sequence_len – 1

# Add Input Embedding Layer
model.add(Embedding(total_words,  10, input_length=input_len))

# Add Hidden Layer 1 - LSTM Layer
model.add(LSTM(100))

# Add Output Layer
model.add(Dense(total_words,  activation='softmax'))

model.compile(loss='categorical_crossentropy',   optimizer='adam')
```