Recurrent Neural Networks

Applications

- ☆ Sequence
- Text (translation)
- Speech (to text)
- Video

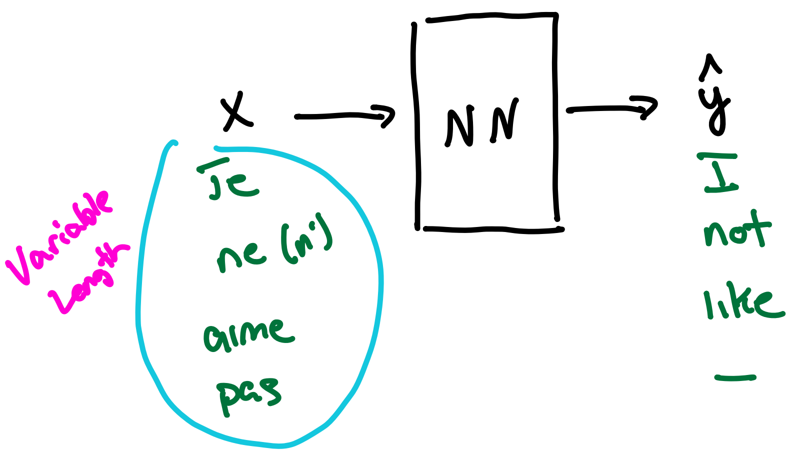
} Supplanted by Transformers

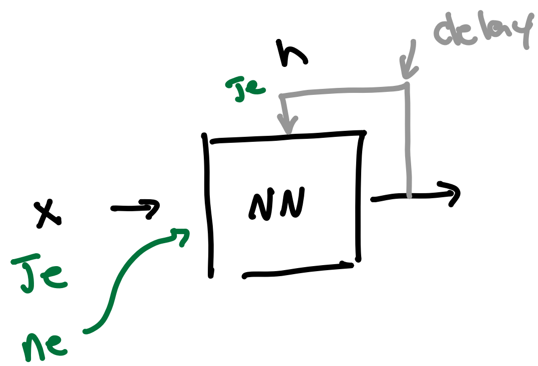
Example

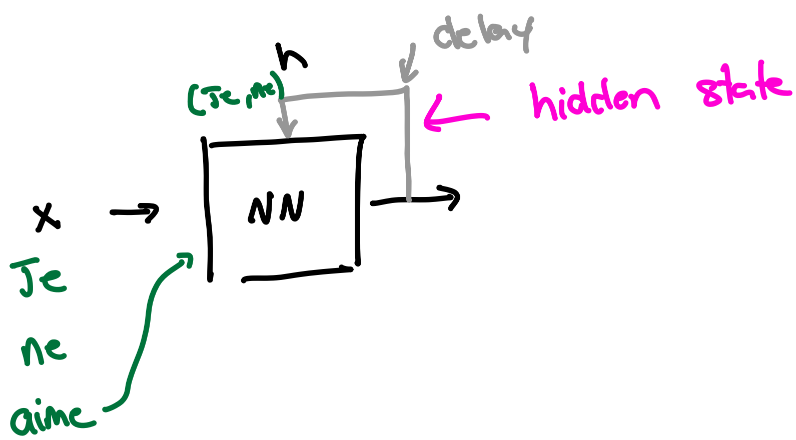
Machine translation

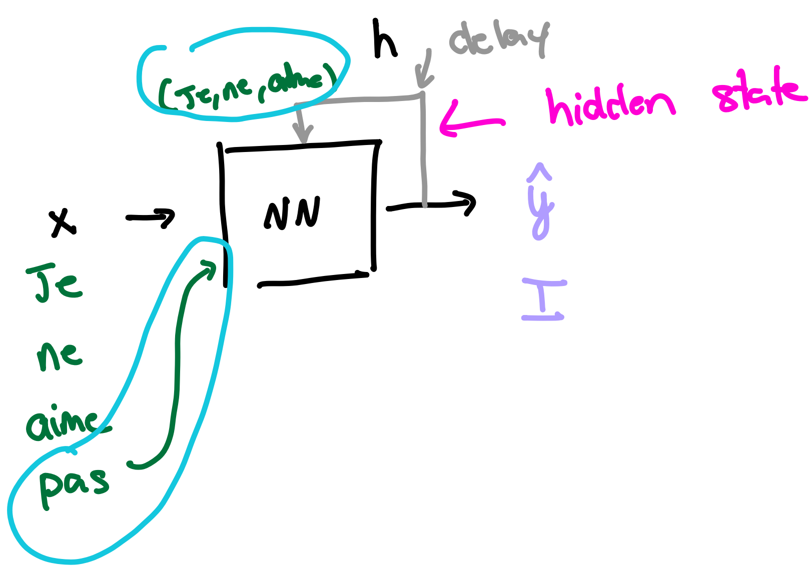
Je n'aime pas. → (I do not like.)

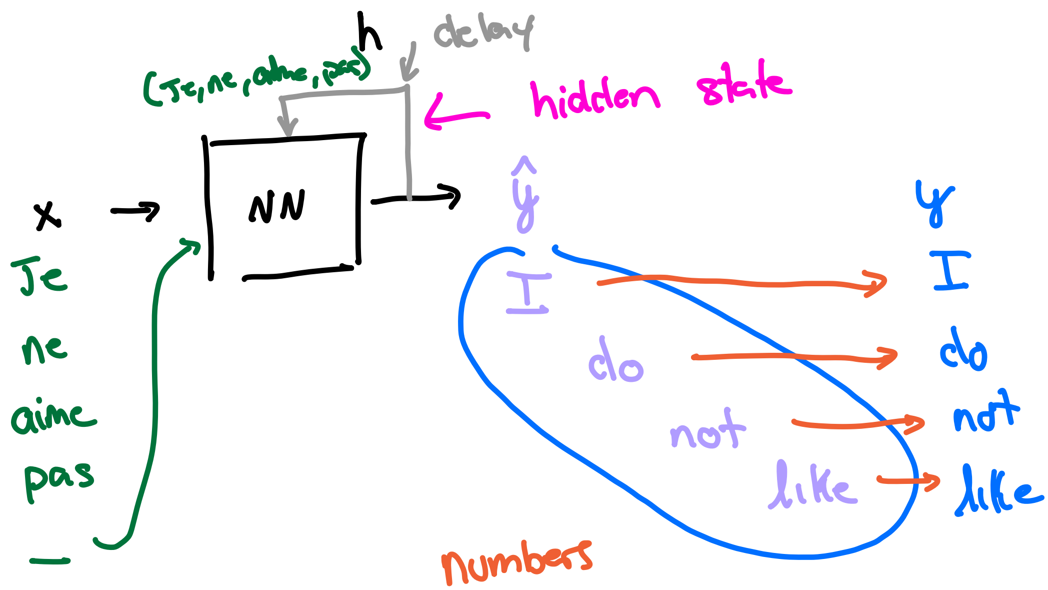
(I not like)




delay
h
Je
NN
x
Je
ne


delay
h
(Je, ne)
hidden state
NN
x
Je
ne
aime


h
delay
(Je, ne, aime)
hidden state
NN
x
ŷ
I
Je
ne
aime
pas

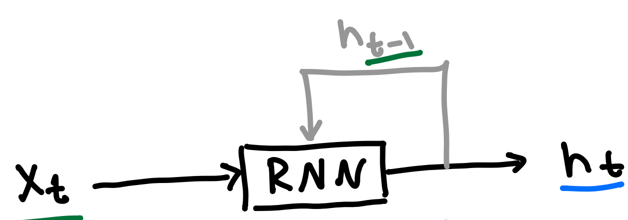


RNN Layer Model

numbers

$$X = [\underset{x_1}{Je}, \underset{x_2}{ne}, \underset{x_3}{aime}, \underset{x_4}{pas}]$$

$$h = [h_0, h_1, h_2, h_3]$$



activation function

$$h_t = \tanh(W_{IH} x_t + b_{IH} + W_{HH} h_{t-1} + b_{HH})$$

W_{IH}: weights from input to hidden

W_{HH}: weight from hidden to hidden

Pseudocode

class Model ← zero Init
  hidden_state (h)

  W_ih →
  W_hh →   Kaiming
  b_ib →   Init
  b_hh →

m = Model(nx, hx)

Inference:

```
for word in french_sent
    model(word)

for word in english_sent
    yhat = model(null_word)
    loss = criterion(yhat, word)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Language Dataset

x = [   ] } single sentence
         L

$X = (L, N, X_{in})$

- X ← French sentence Data
- L ← Sequence Length
- N ← Batch Size
- X_{in} ← Input Size

$Y = (L_{out}, N)$

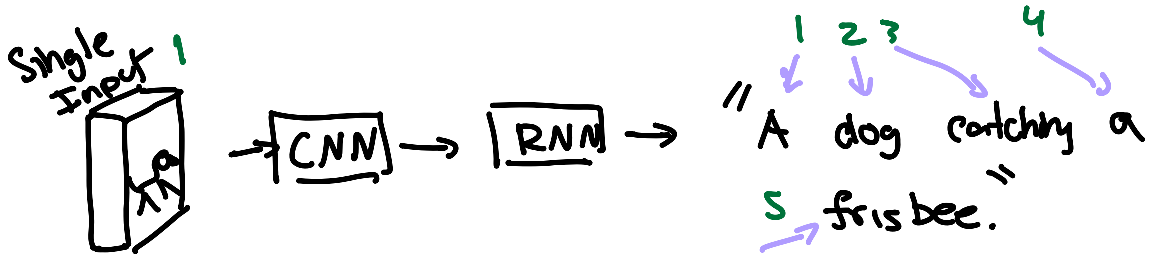
- Y ← English sentences

```
for epoch
    for batch
        for word
```

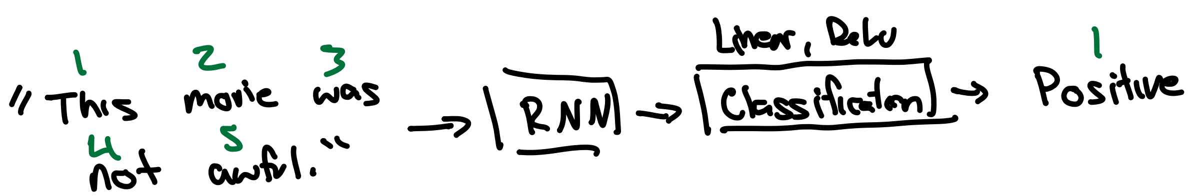
Paradigms

- one-to-one → traditional feed forward NN (no recurrent layers)

- one-to-many
  e.g. image captioning



- many-to-one
  e.g. sentiment analysis



- many-to-many
  e.g. translation

The classic RNN has a bottleneck issue.

- LSTM → Long - Short Term Memory
- GRU

These add additional hidden state