Recurrent Neural Networks (RNNs)

Michel RIVEILL
michel.riveill@univ-cotedazur.fr

Motivation

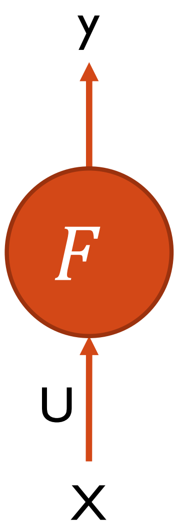
- Humans don't start their thinking from scratch every second
  - Thoughts have persistence
- Traditional neural networks can't characterize this phenomena
  - Ex: classify what is happening at every point in a movie
  - How a neural network can inform later events about the previous ones
- Recurrent neural networks address this issue
  - Some applications
    - NER - Naming Entity Recognition
      - Same word may have a different label depending on the context.
        - Apple CEO Tim Cook eat an apple
    - Forcasting - Time-series Prediction
- How?
  - Add state to artificial neurons

What are RNNs?

- Main idea is to make use of sequential information
- How RNN is different from neural network?
  - Vanilla neural networks (MLP) assume all inputs and outputs are independent of each other
  - But for many tasks, that's a very bad idea
- What RNN does?
  - Perform the same task for every element of a sequence (that's what recurrent stands for)
  - Output depends on the previous computations!
- Another way of interpretation – RNNs have a "memory"
  - To store previous computations

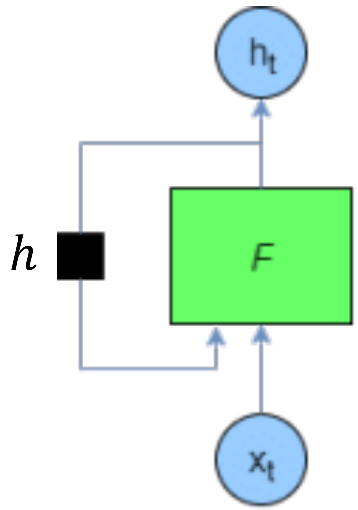
From vanilla NN to recurrent NN

- Vanilla cell
  - $y = F(U.X)$



From vanilla NN to recurrent NN

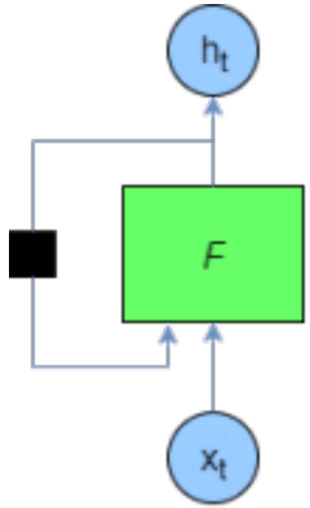
- Vanilla cell
  - $y = F(U.X)$
- Recurrent cell → use 2 weights matrix
  - Add an internal variable: h
  - The output depends to the current entry and the previous internal variable:
    - $h_t = F(W.h_{t-1} + U.Xt)$
    - Could be rewritten on $h_t = F(W.[h_{t-1}, Xt])$



6

From vanilla NN to recurrent NN

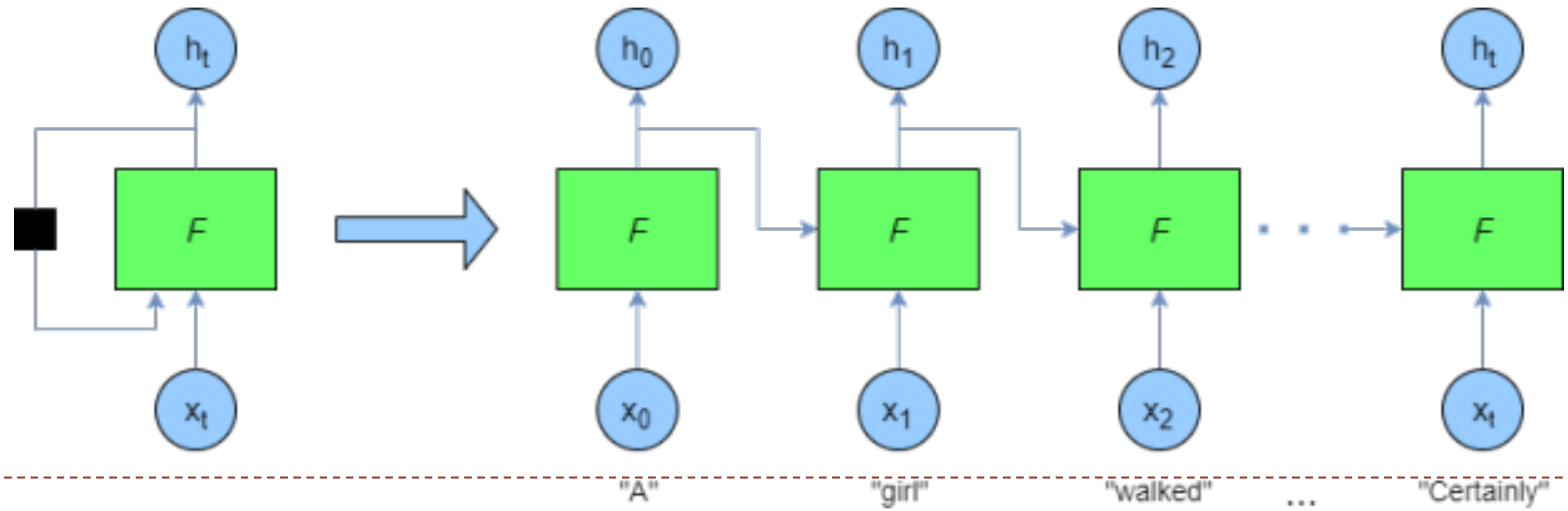
- Vanilla cell
  - $y = F(U.X)$
- Recurrent cell → use 2 weights matrix
  - $h_t = F(W.[h_{t\,-\,1}, X_t])$
- Recurrent layer, step by step
  - at each time step
    - A new entry is being supplied
    - And a new output (h_t) is calculated using:
      - ☐ The new input X_t
      - ☐ The output of the previous step $h_{t\,-\,1}$



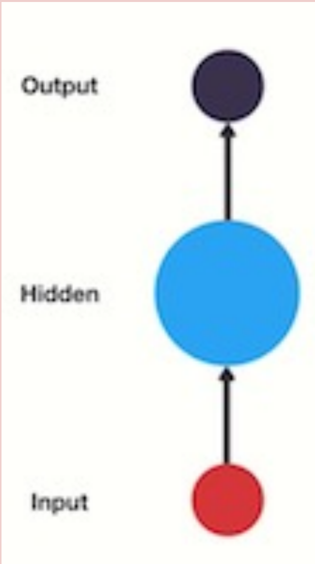
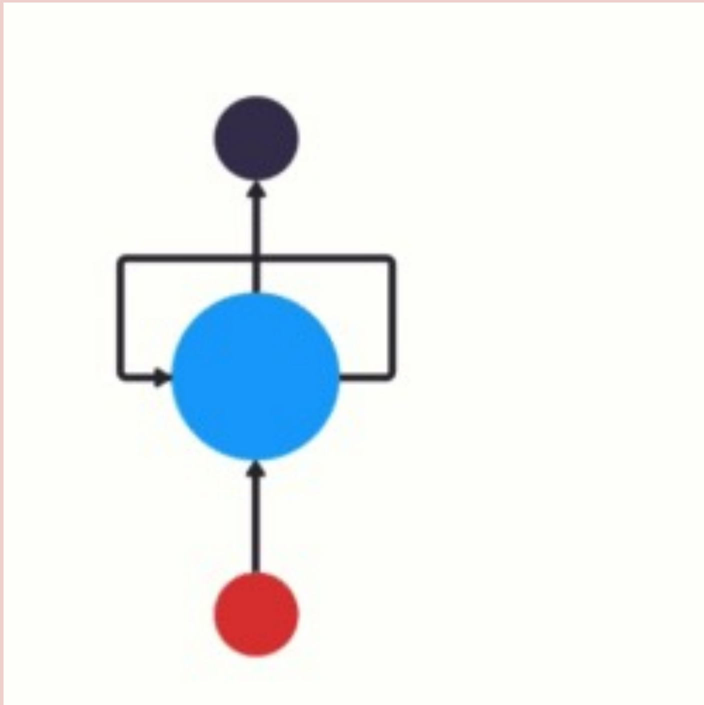
- $h_1 = F(W.[h_0, X_1])$
- $h_2 = F(W.h_1, X_2])$
- $h_3 = F(W.h_2, X_3])$
- ...

From vanilla NN to recurrent NN

- Vanilla cell
  - $y = F(U.X)$
- Recurrent cell
  - $h_t = F(W.[h_{t\,-\,1}, X_t])$
- Recurrent neural networks are "unrolled" programmatically during training and prediction
  - All neurons share the same weight matrix

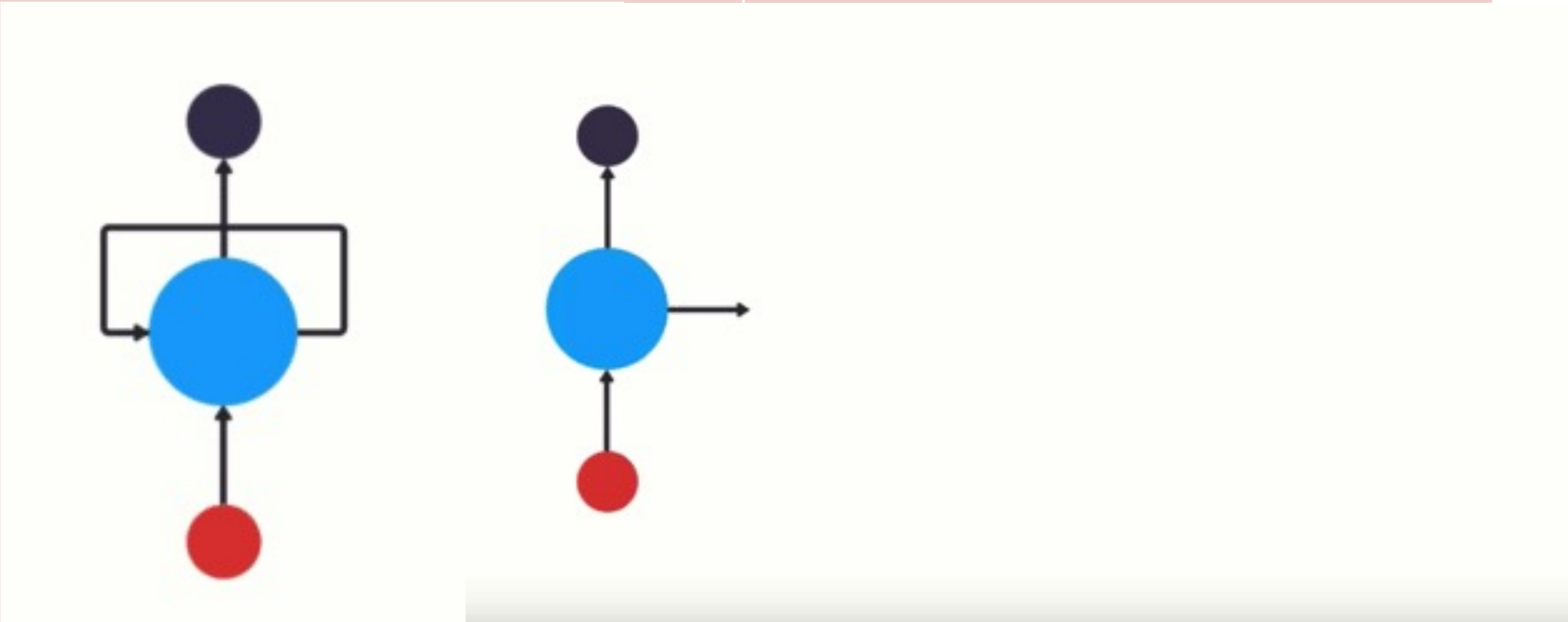


Remember

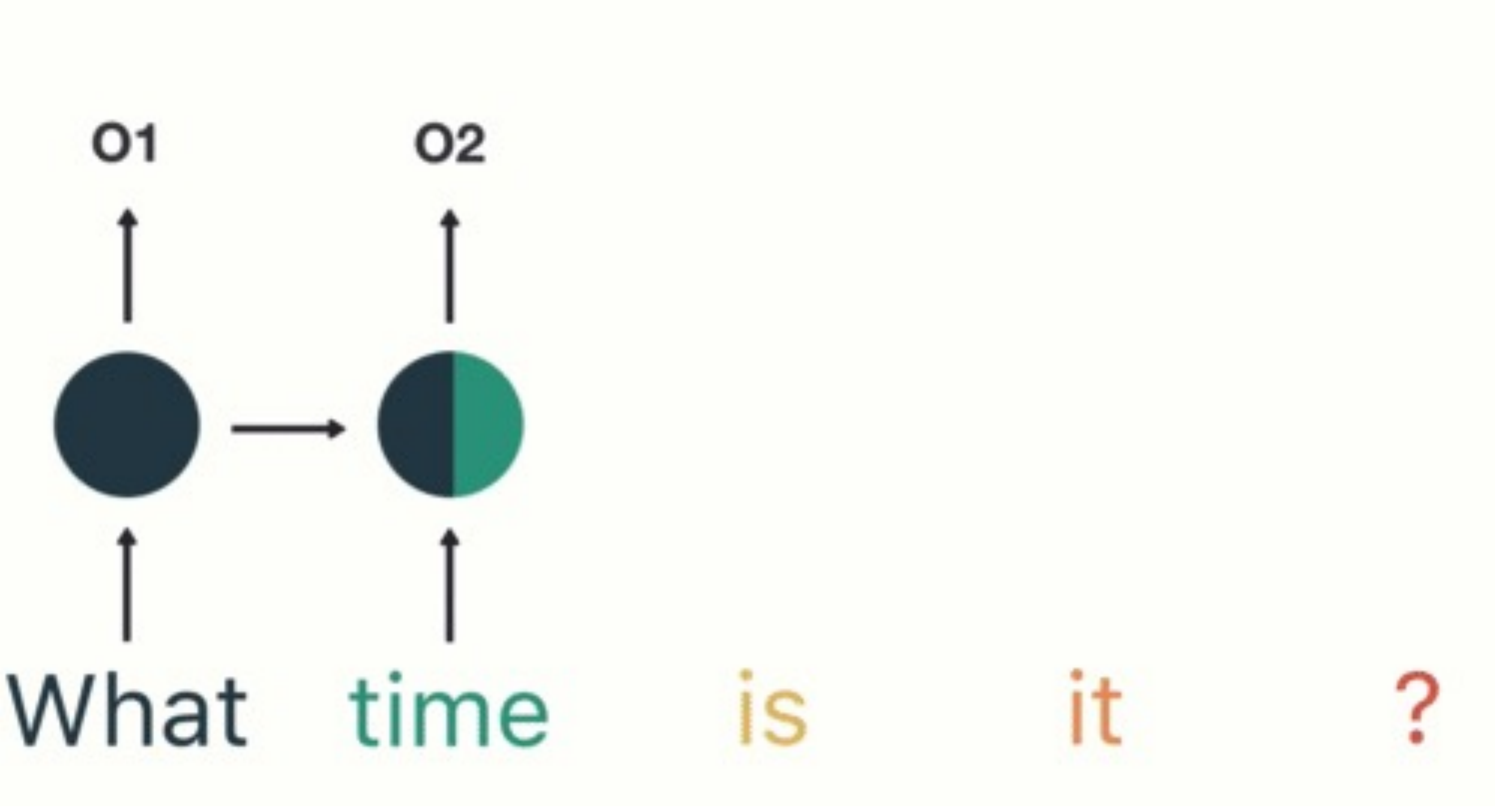
From	To
	

Remember

From	To

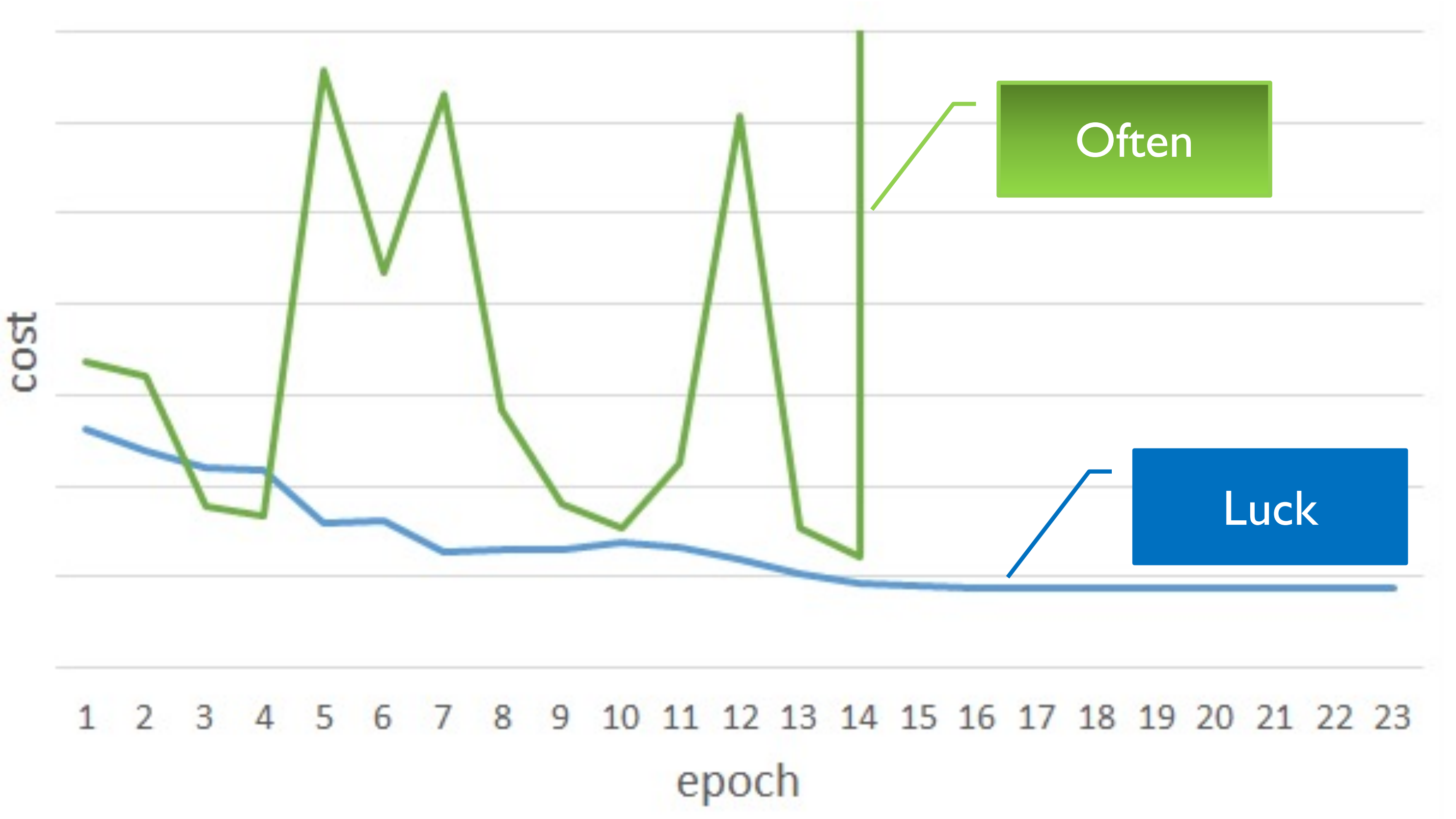


RNN in action



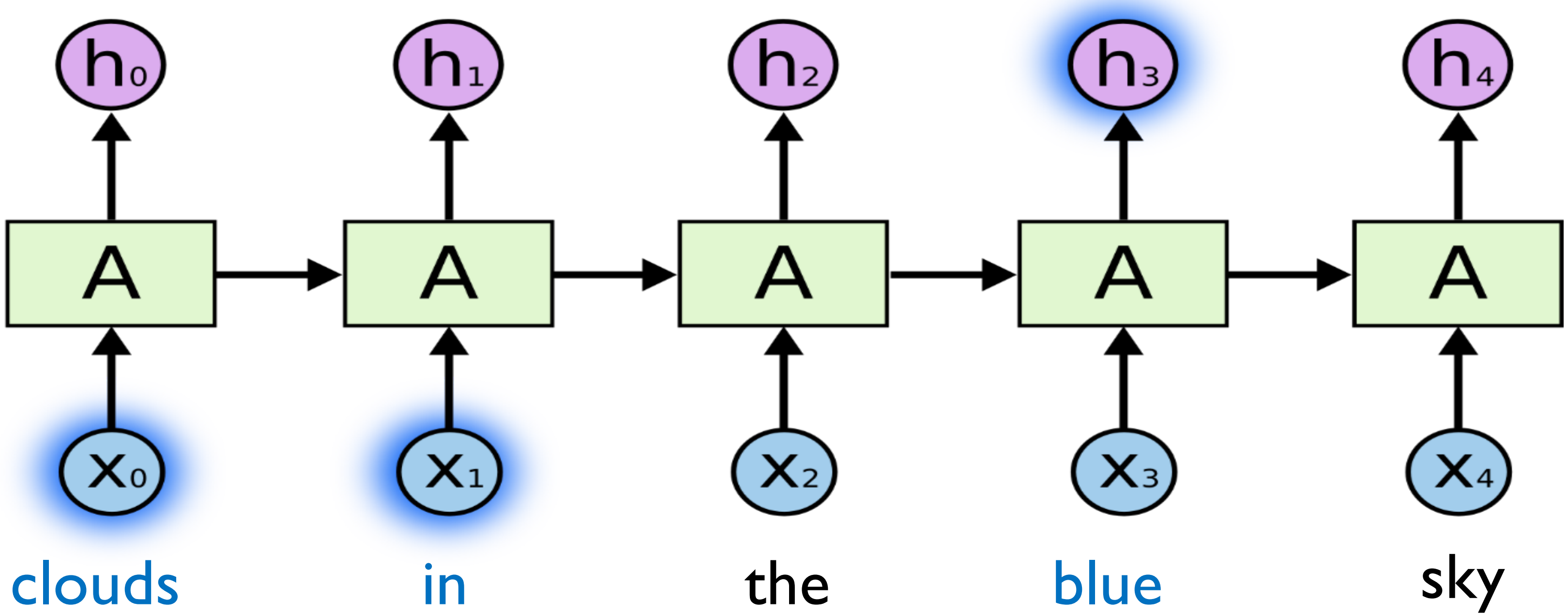
Problems with naive RNN

- RNNs do not learn easily
- Unfolding the network for learning leads to vanishing gradient problems!



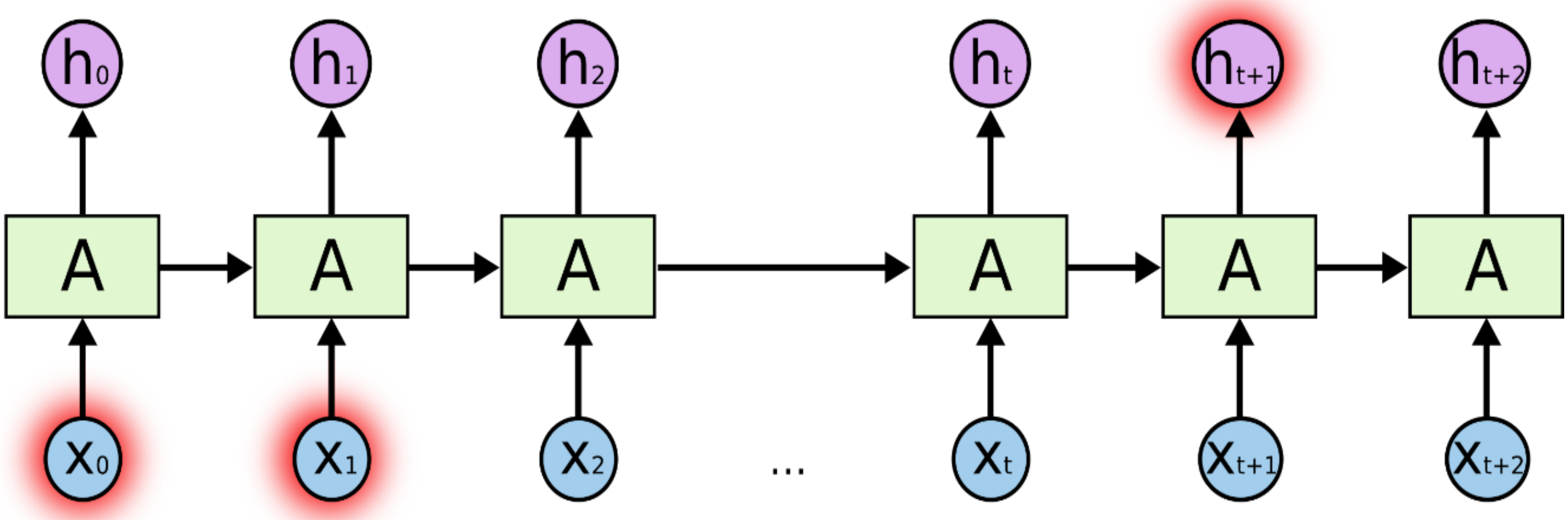
From vanilla RNN...

- The context is close to the word to be predicted
  - Few iterations separate them.
  - No problem



… to LSTM (Long Short Term Memory)

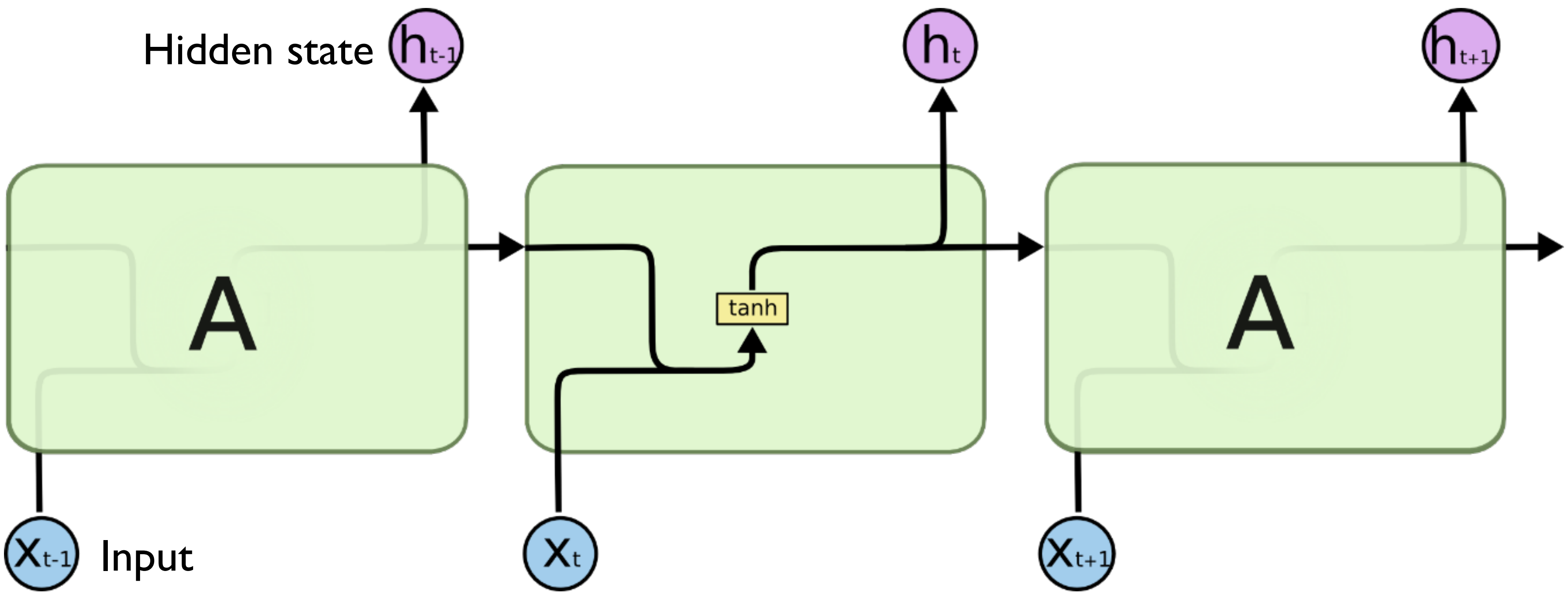
- The context is far from the word to predict
  - Many iterations separate them!
  - Possible gradient problem



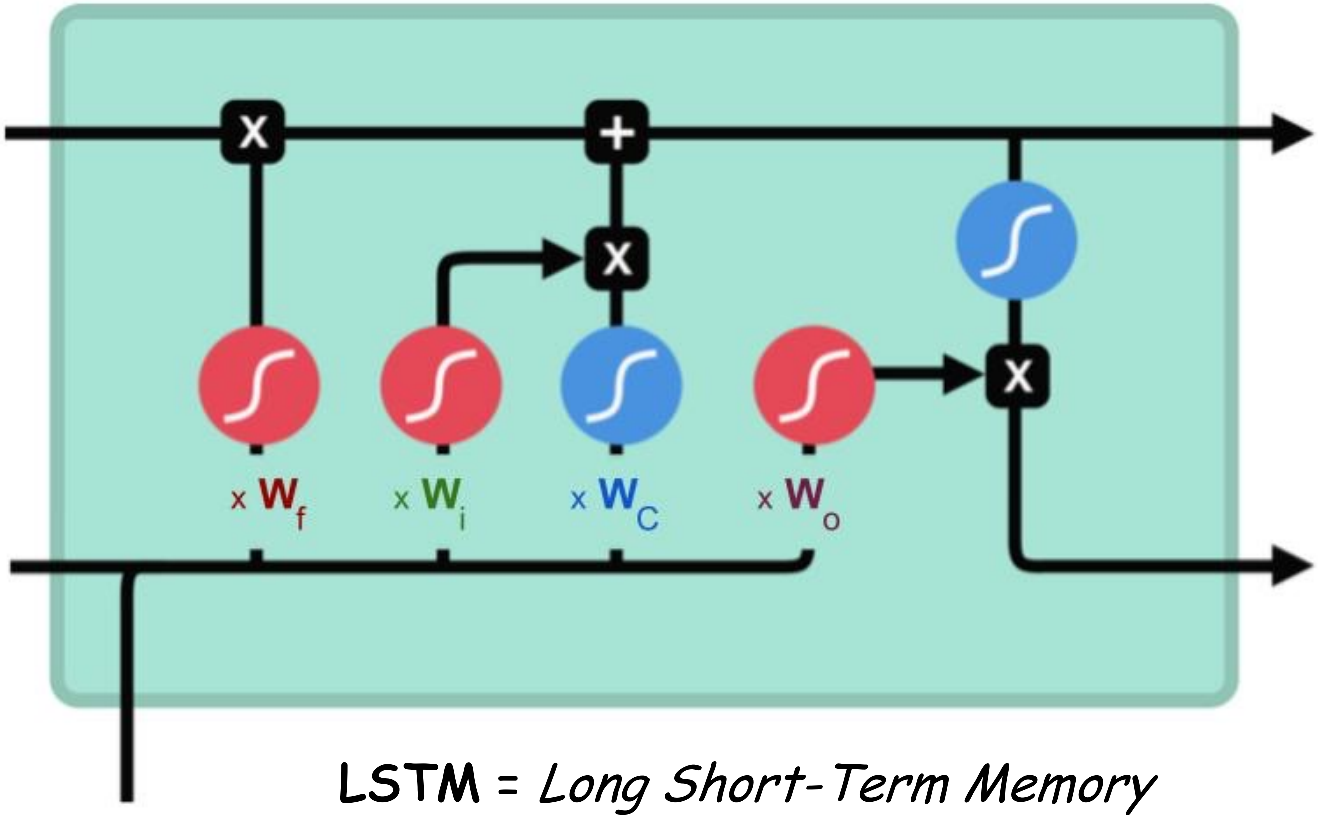
I grew up in France… I speak French…

From Vanilla to LSTM Cells

- It is necessary to prevent the gradient from disappearing...
- Normally, the network memory is
  - $h_t = tanh(W \square [h_{t-1}, xt])$
  - Involves a single level of processing
  - Creating the risk of the evanescent gradient.



Dealing with the vanishing gradient problem → LSTM cell



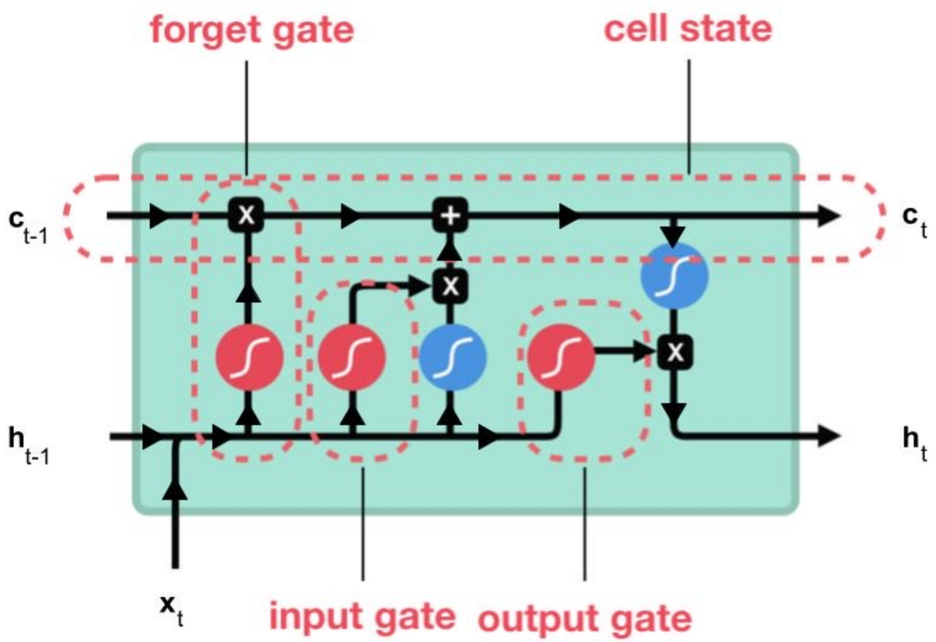
LSTM = *Long Short-Term Memory*

(crédit : image modifiée de Michaël Nguyen)

LSTM cell

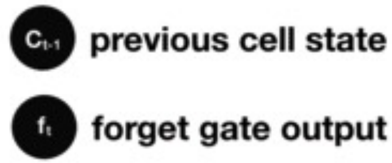
- Cellule composée de trois “portes” : ce sont des zones de calculs qui régulent le flot d’informations (en réalisant des actions spécifiques)
  - Forget gate (porte d’oubli)
  - Input gate (porte d’entrée)
  - Output gate (porte de sor

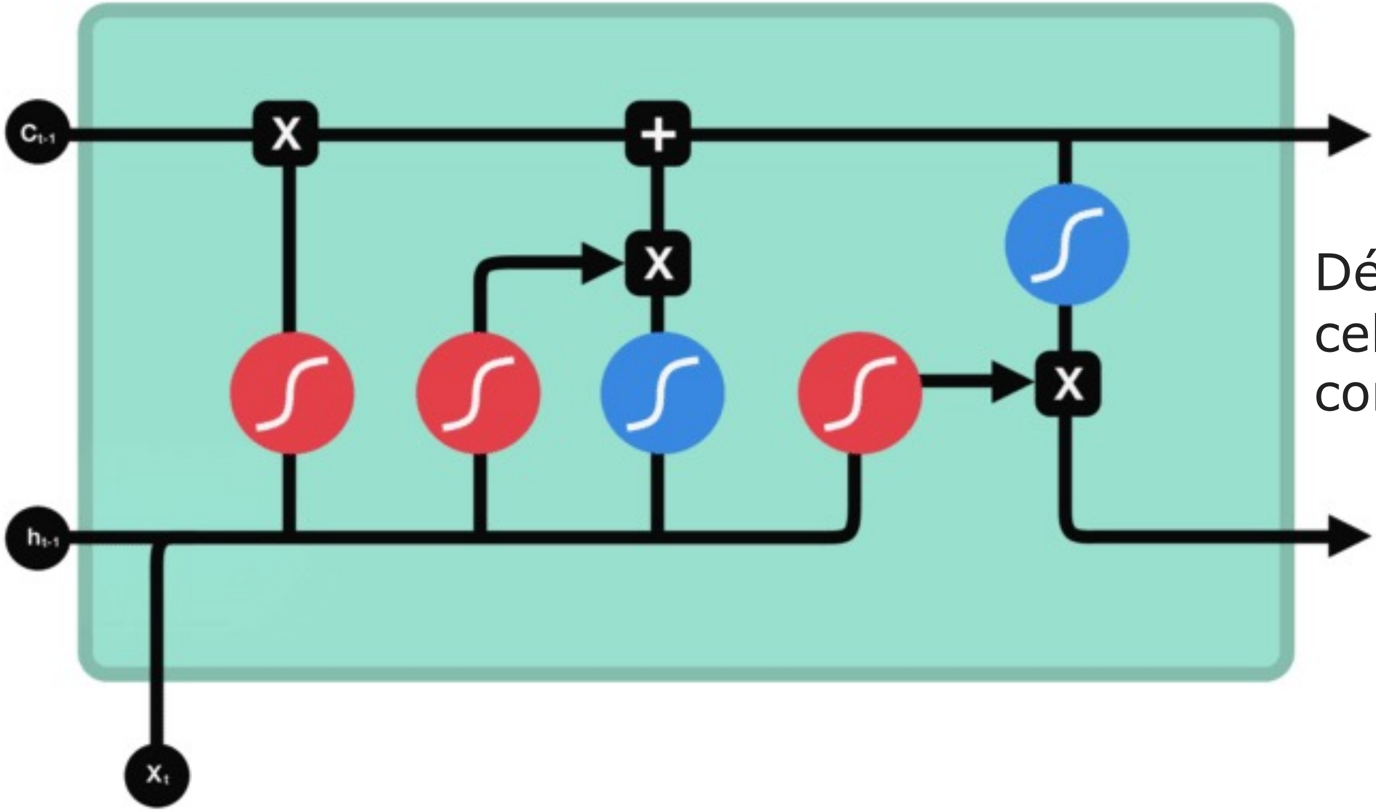
- Hidden state (état caché)
- Cell state (état de la cellul
  - Like residual



(crédit : image modifiée de Michaël Nguyen)

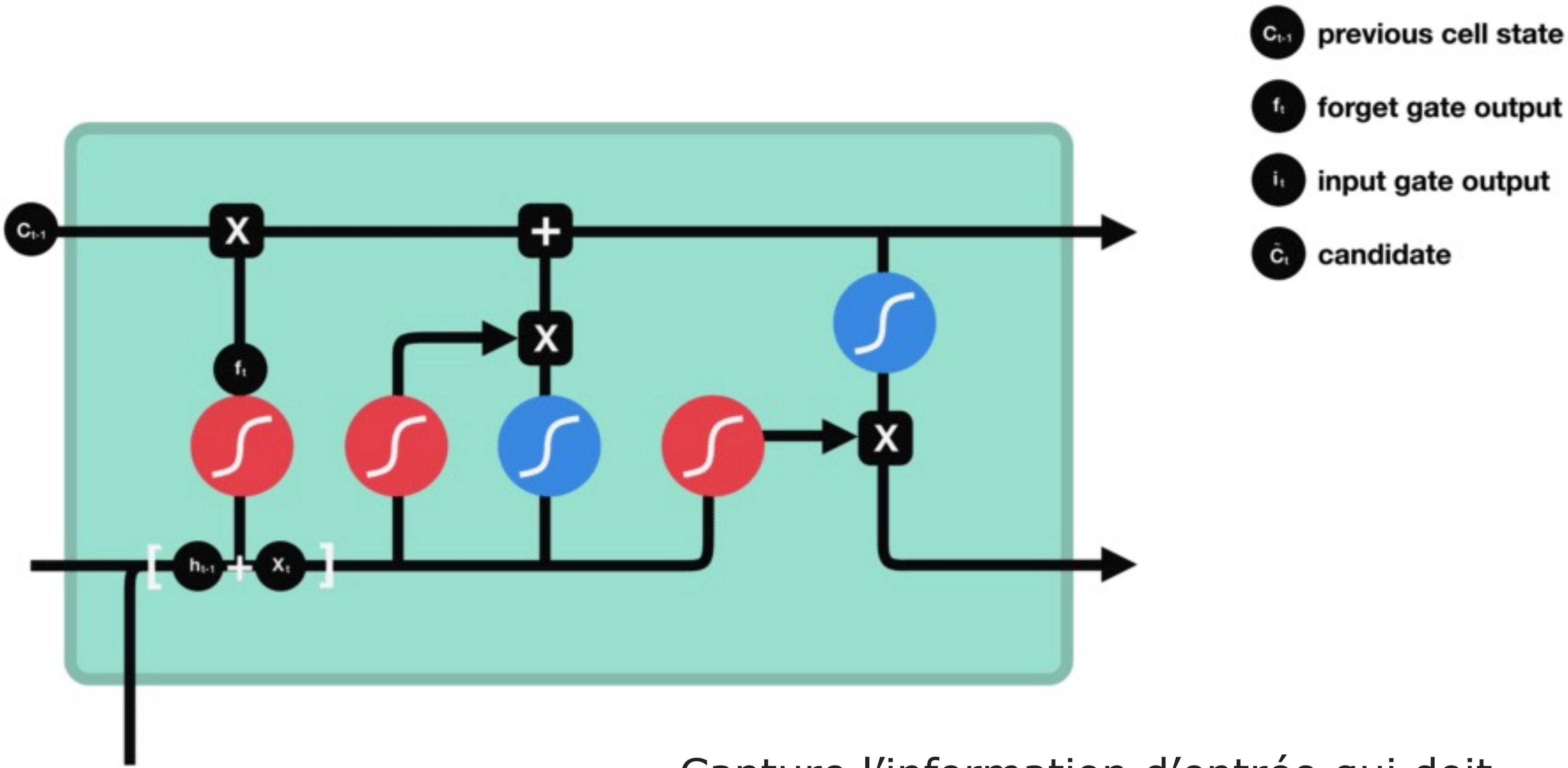
LSTM cell (porte oubli / forget get)





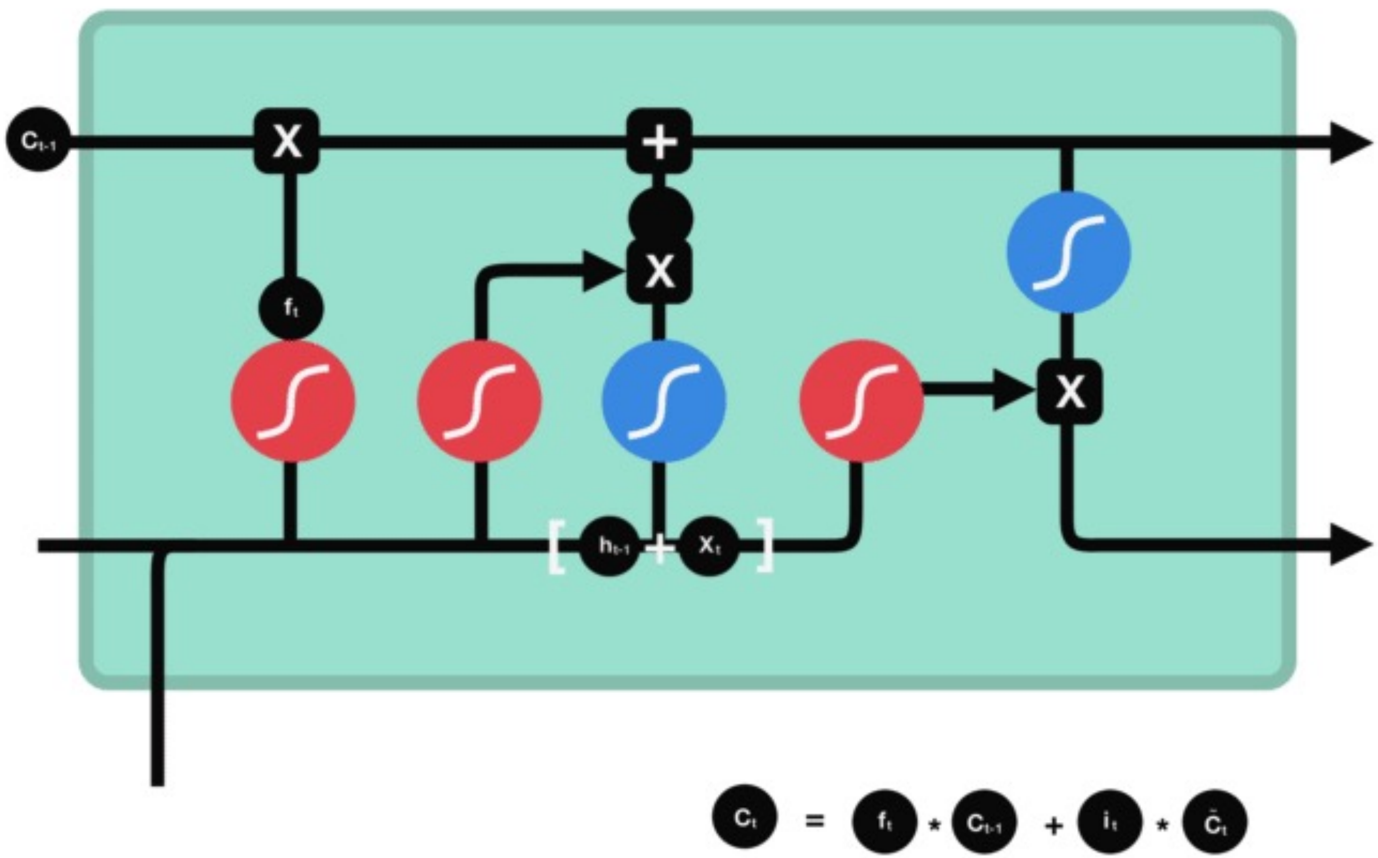
Décide si l’état de la cellule doit être conservé ou non.

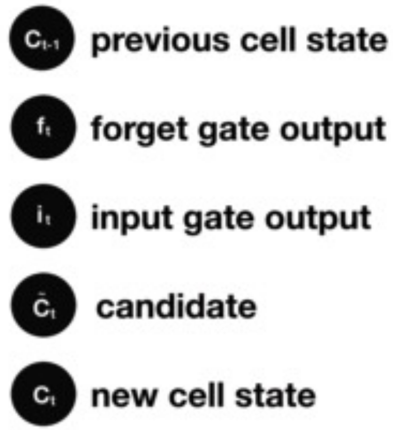
LSTM cell (porte entrée / input get)



Capture l'information d'entrée qui doit être incluse dans l'état de la cellule

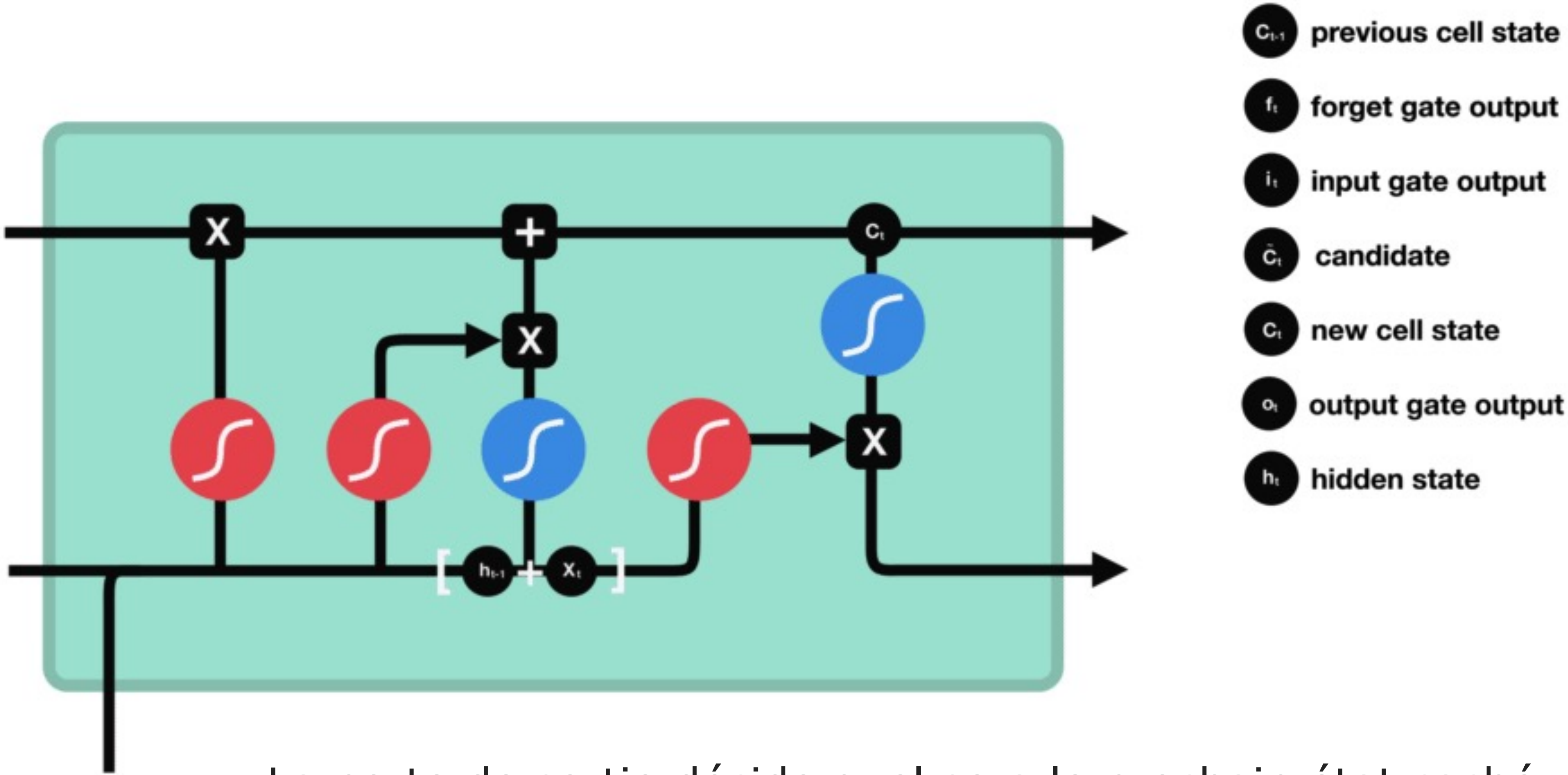
LSTM cell (état de la cellule / cell state)





L’état de la cellule se calcule assez simplement à partir de la porte d’oubli et de la porte d’entrée

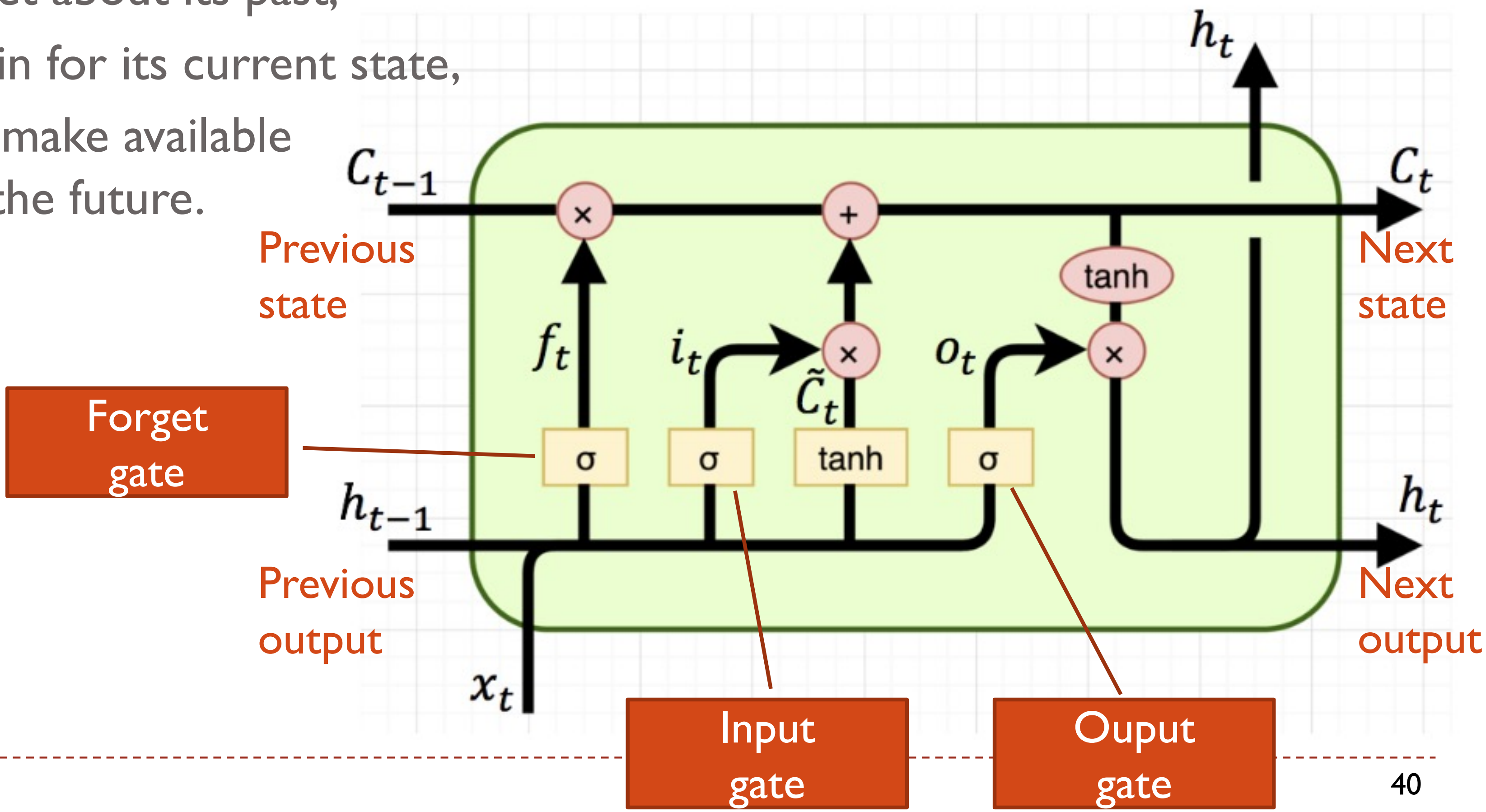
LSTM cell (porte de sortie / output gate)



La porte de sortie décide quel sera le prochain état caché. Il contient des informations sur les entrées précédentes du réseau et sert aux prédictions.

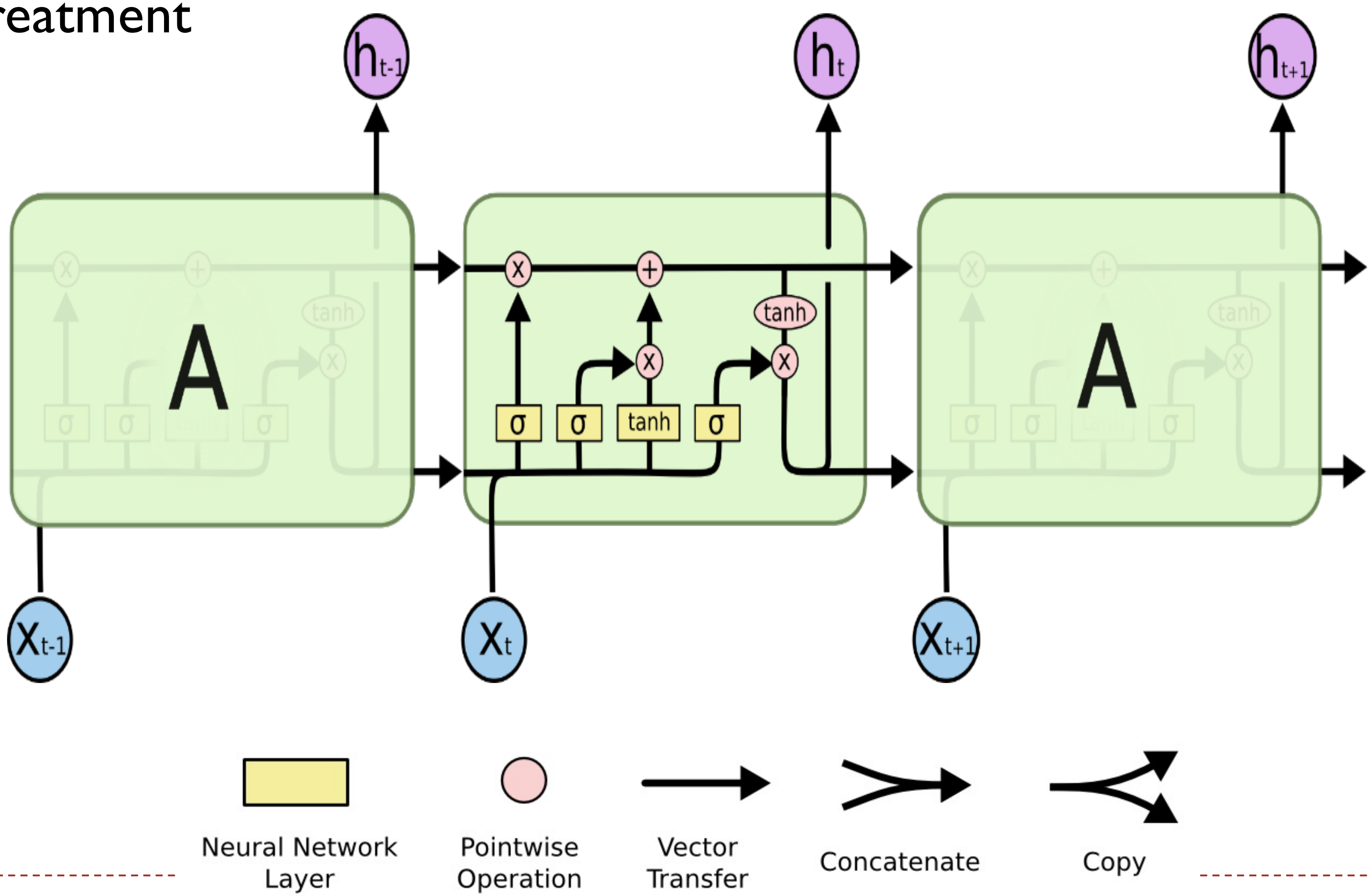
LSTM Cells

- Adds a context memory that affects the information flow and its processing (cell state).
- Three gates decide what a cell should
  - forget about its past,
  - retain for its current state,
  - and make available for the future.



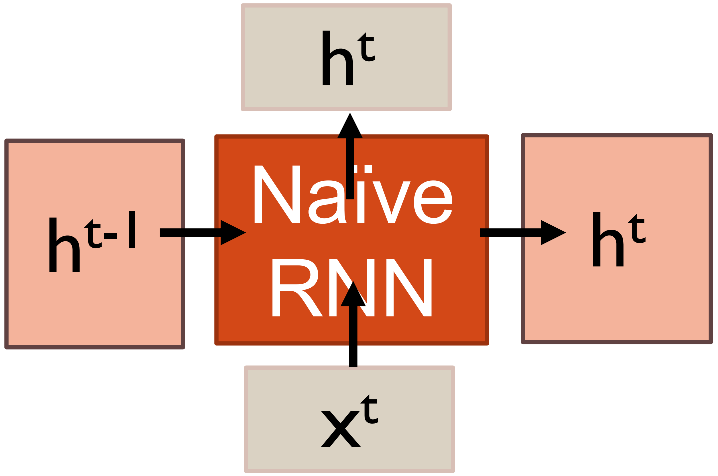
LSTM Cells

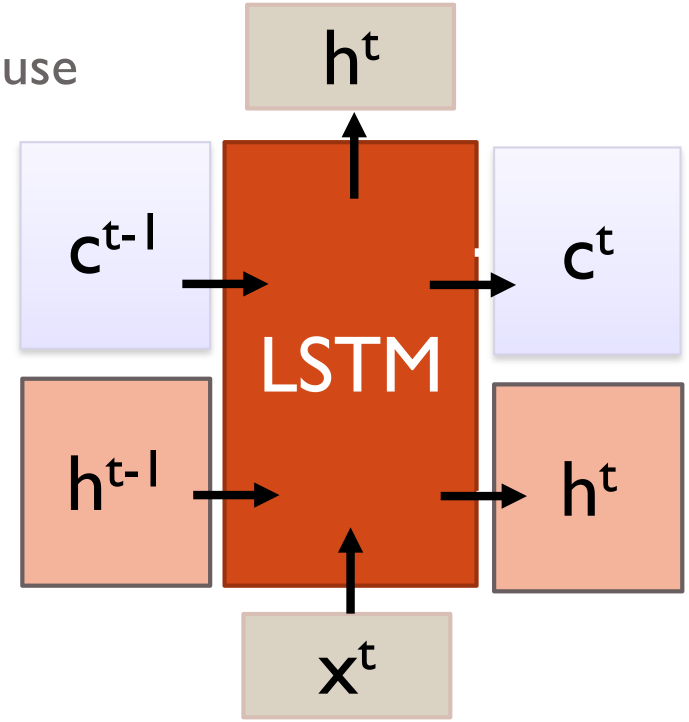
- Concretely, recurrence in an LSTM cell involves 4 levels of treatment



Naïve RNN vs LSTM

- Naïve RNN
  - Reuse at each step the previous output
- LSTM
  - At each step 3 gate control the use use of Input value, Cell state and previous output



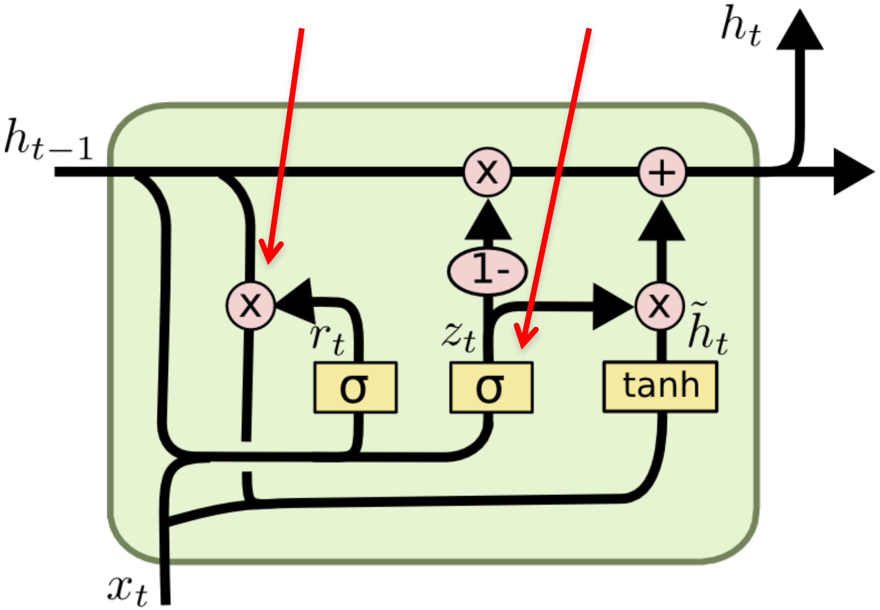


c changes slowly → c^t is c^{t-1} added by something

h changes faster → h^t and h^{t-1} can be very different

GRU – gated recurrent unit

- GRU = a light LSTM Cell



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

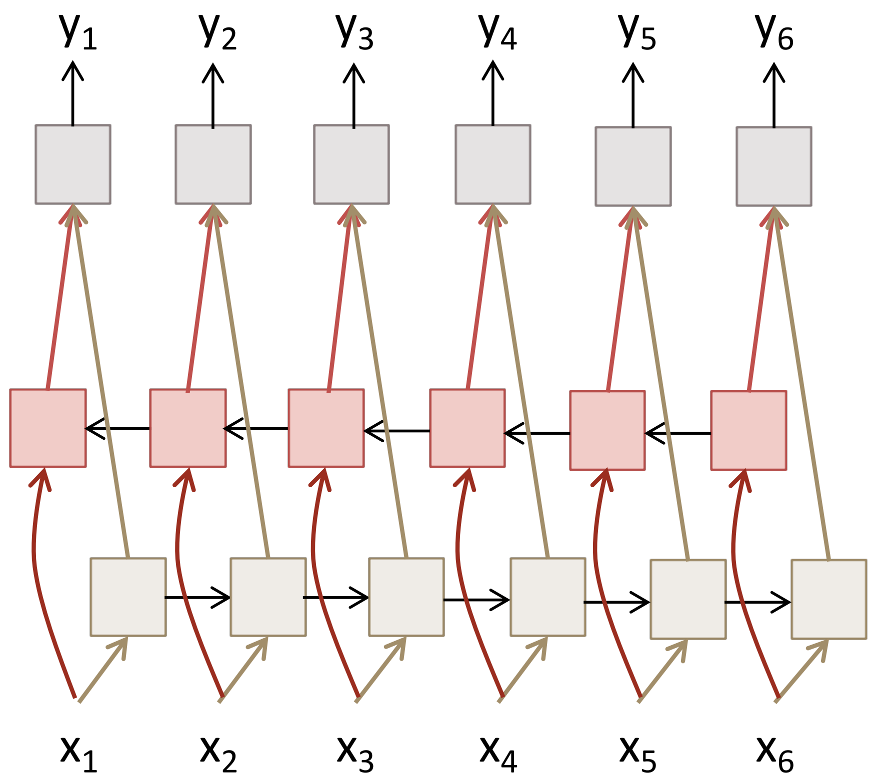
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- It combines the forget and input into a single update gate.
- It also merges the cell state and hidden state.

→ This is simpler than LSTM.

Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction



- Popular in speech recognition, could be used also with text

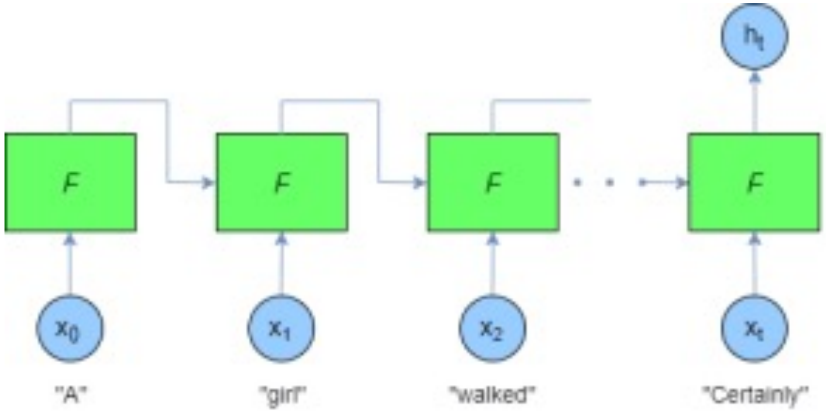
RNN cell in Keras

Keras Long Short-Term Memory Cell

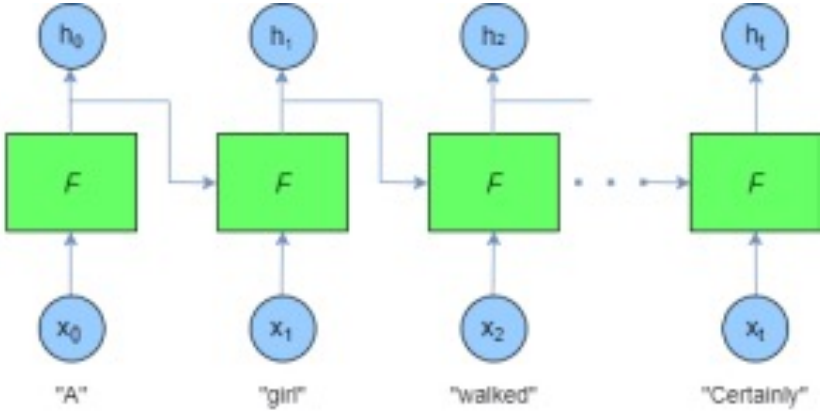
- **from** tf.keras.layers **import** LSTM

- Main params
  - **Units:** dimension of output space

  - **return_sequences:** True or False
    - If False return only the last output
    - If True return the full sequence of the output sequence
      - Output sequence = hidden state (the vocabulary change regardind documentation)
  - **return_state**: True or False
    - If True return 3 values
      - The full output sequence or only the last one (depend on return_sequences)
      - The last output sequence
      - The cell state
    - If False return nothing
  - **stateful**: True or False
    - If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
    - You have to put shuffle=False in a fit method



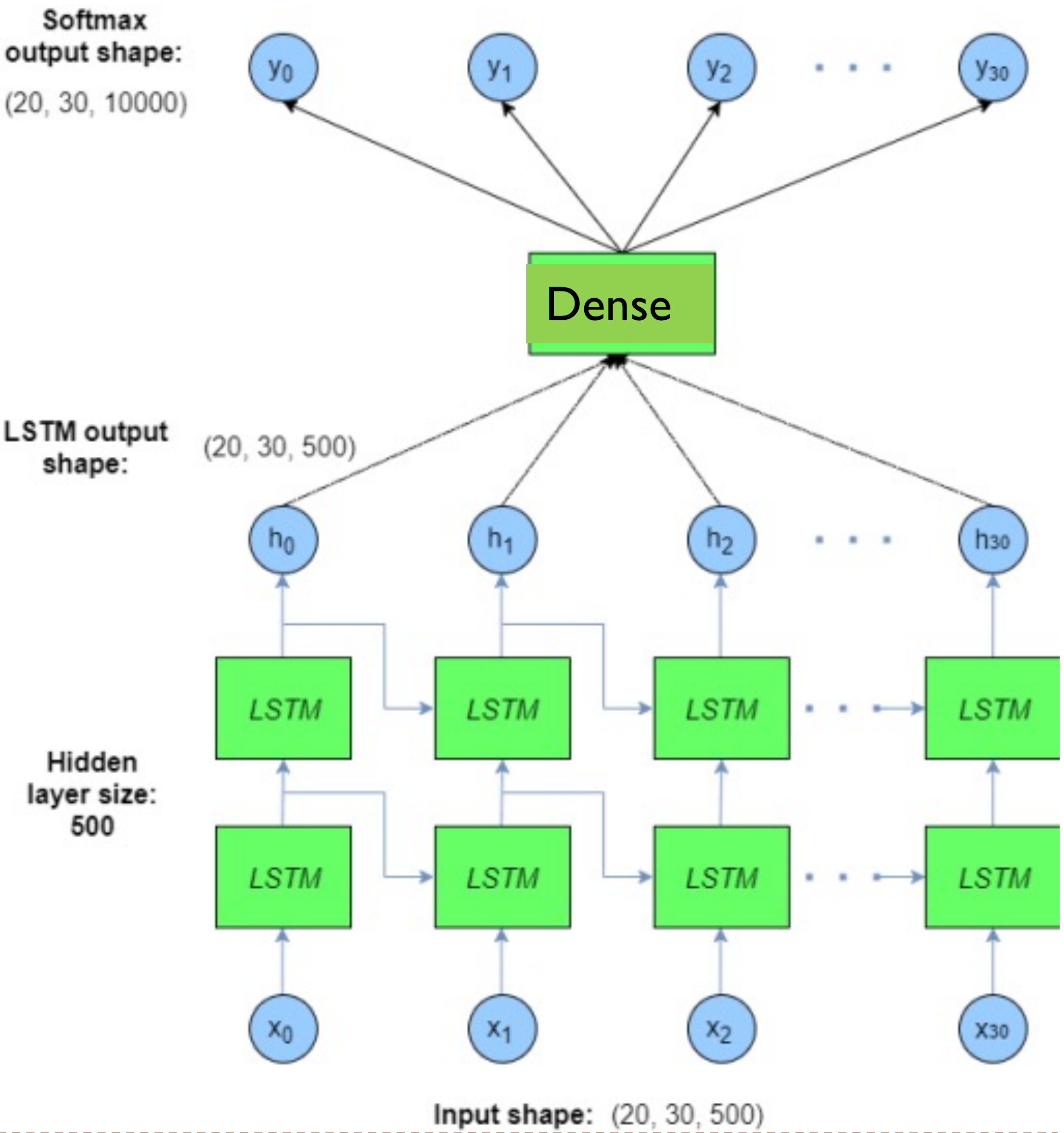
return_sequences = False



return_sequences = True

with return_sequences=False, Dense layer is applied only once at the last cell

If return_sequences=True Dense layer is applied to every timestep



Keras Gated Recurrent Unit Cell

- **from** keras.layers **import** GRU

- Main params (similar to LSTM)
  - **Units:** dimension of output space

  - **return_sequences:** True or False
    - If False return only the last output
    - If True return the full sequence of the output sequence
      - Output sequence = hidden state (the vocabulary change regardind documentation)
  - **return_state**: True or False
    - If True return 3 values
      - The full output sequence or only the last one (depend on return_sequences)
      - The last output sequence
      - The cell state
    - If False return nothing
  - **stateful**: True or False
    - If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.
    - You have to put shuffle=False in a fit method

A basic example

- inputs = Input(shape=(SEQUENCE_SIZE,))
- embedding = Embedding(VOCABULARY_SIZE,
  EMBEDDING_SIZE,
  input_length=SEQUENCE_SIZE)(inputs)
- output = LSTM(16, return_sequences=False,
  activation='relu')(embedding)
- predictions = Dense(nb_classes,
  activation='softmax')(output)

- Fit by batch
  - Model.fit(X, y, ….). ← all item have the same length
- Fit by item
  - For i in range(len(X)): ← could be different length
    - Model.fit(X[i], y[i], …)

Some use of RNN
→ Text Classification / Sentiment analysis

Affect a label to a text

- Classify a
  - restaurant review from Yelp!
  - movie review from IMDB

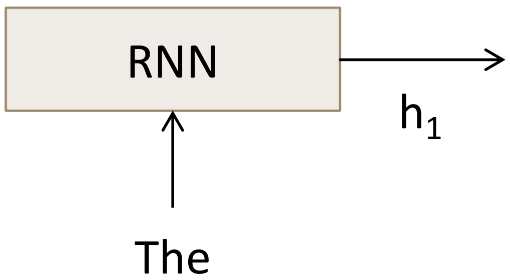
  …

  as positive or negative

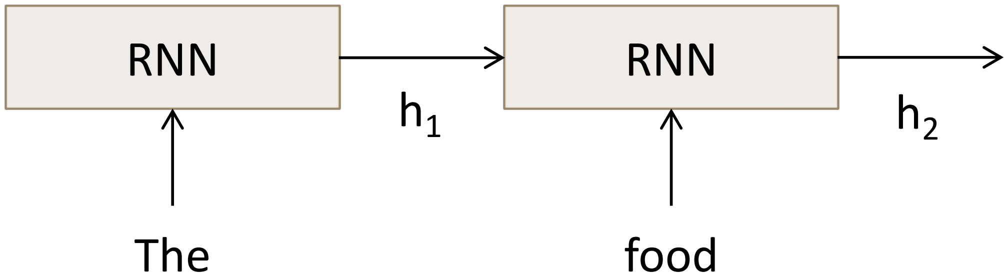
- Inputs:
  - Multiple words, one or more sentences
- Outputs:
  - Positive / Negative classification

- "The food was really good"
- "The chicken crossed the road because it was uncooked"

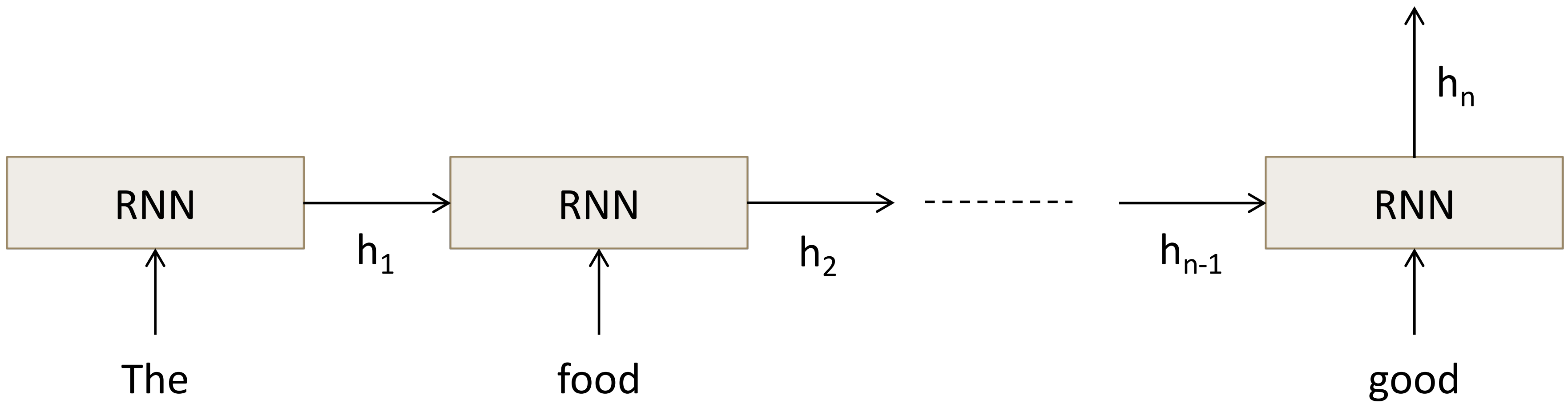
Sentiment analysis



Sentiment analysis

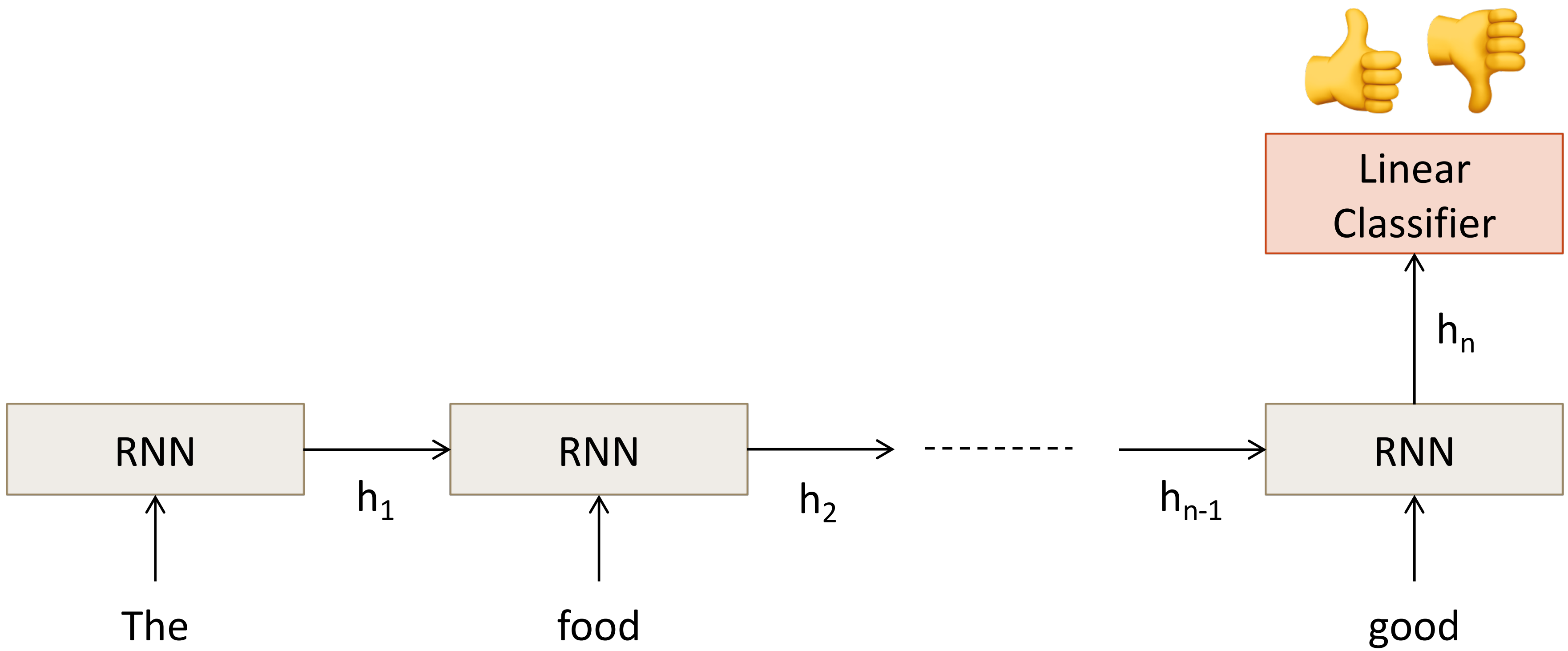


Sentiment analysis



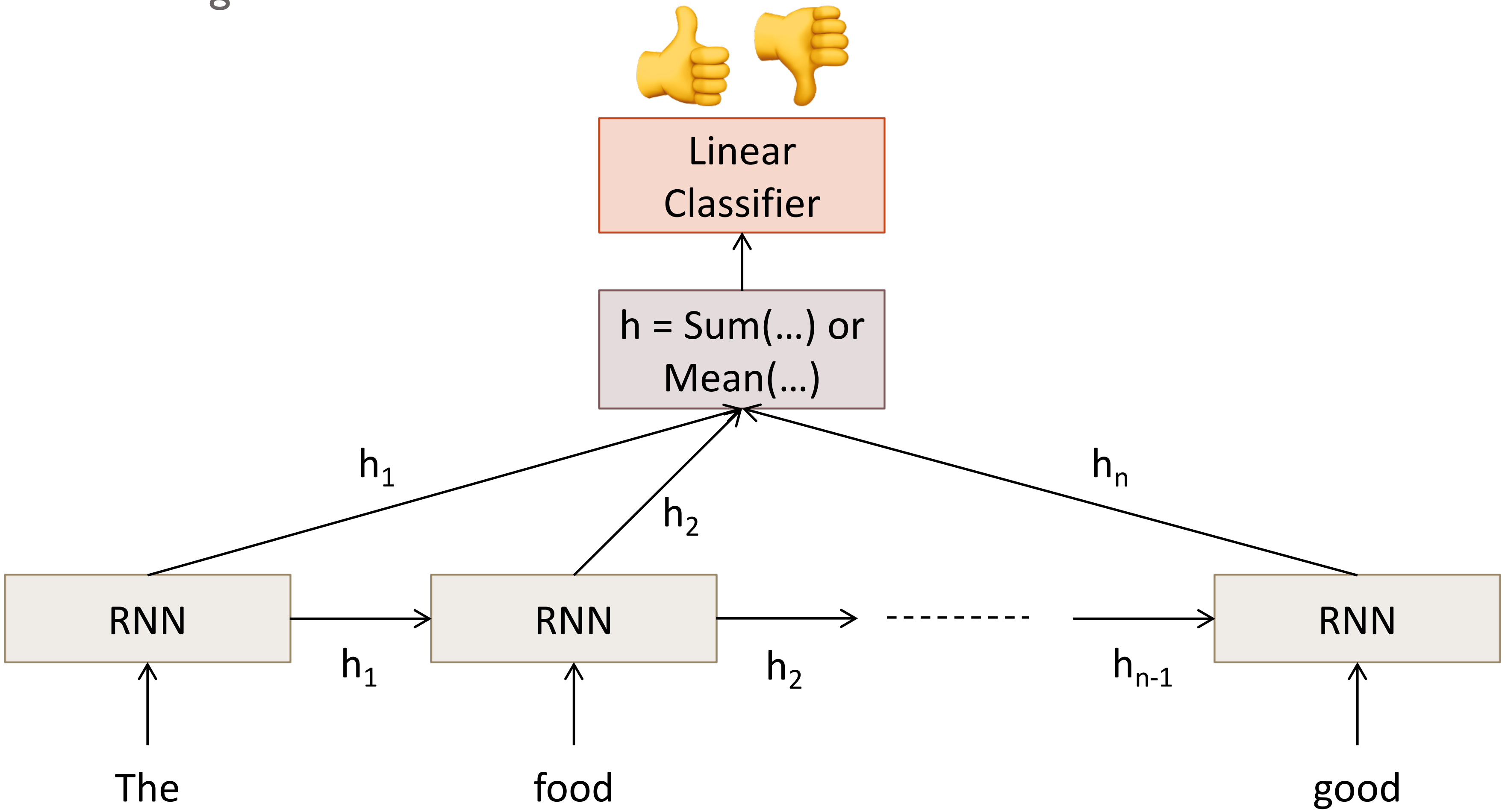
Sentiment analysis - solution 1

- retrieve only the last state



Sentiment analysis – solution2

- Other possible architecture
  - Average all the internal state



Some use of RNN

→ Named Entity Recognition / Part of Speech Tagging

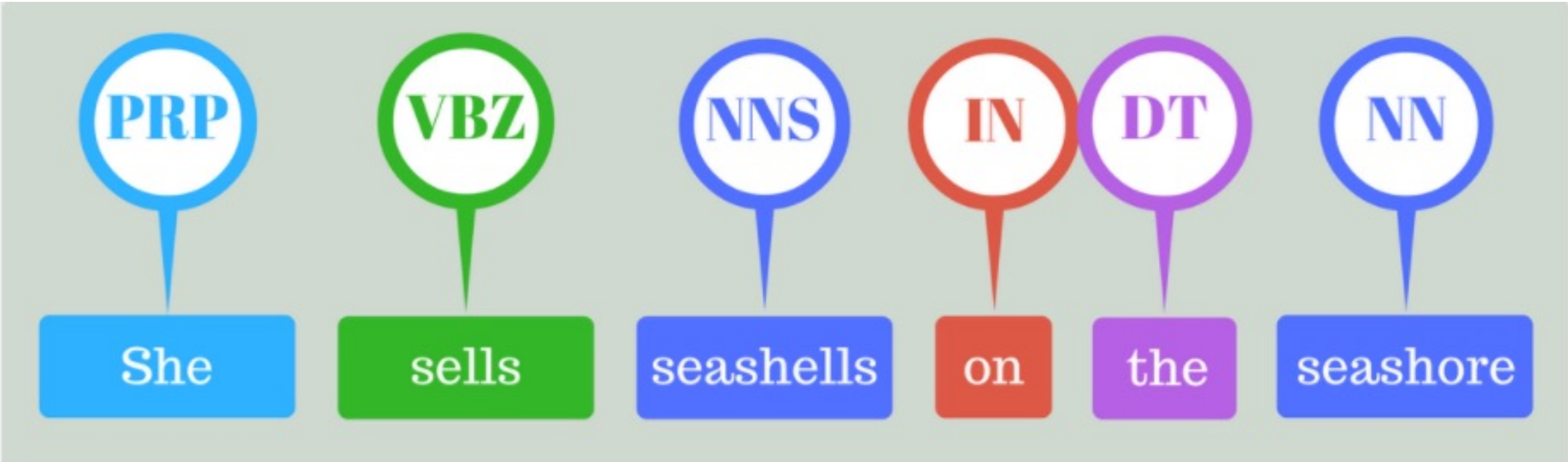
- Affect a label to each word
  - **find** and **classify** names in text
    - Could bean entity : number, country, person, … (NER)
    - Could be a function : noun, verb, adverbs, … (POS)

In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

Some use of RNN
→ Named Entity Recognition / Part of Speech Tagging

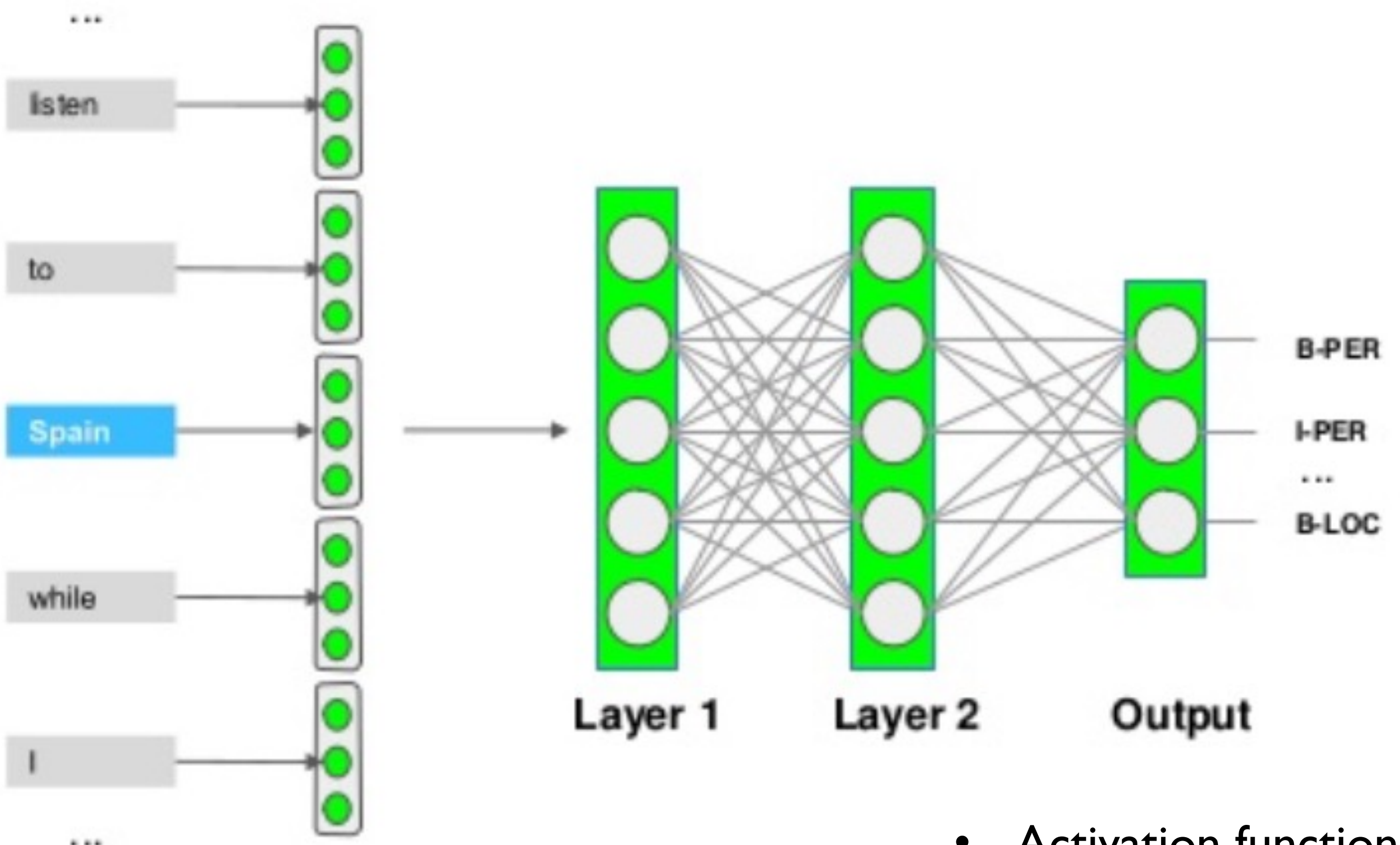
- Affect a label to each word
  - **find** and **classify** names in text
    - Could bean entity : number, country, person, … (NER)
    - Could be a function : noun, verb, adverbs, … (POS)



Label representation – BIO tags

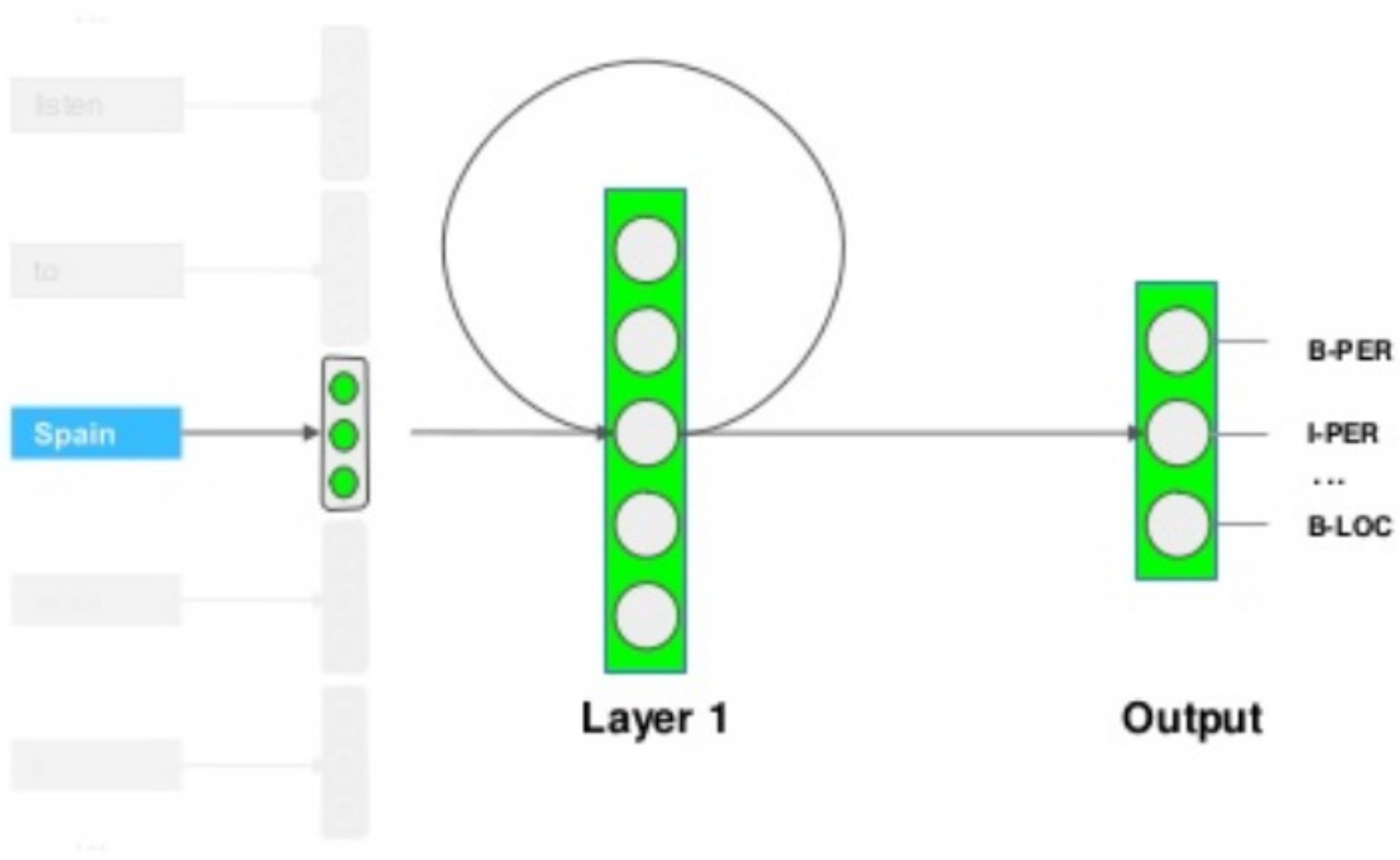
- Labels can be for words or groups of words
  - Adam Smith works for IBM , London .
- To represent this, a "BIO" representation is generally used.
  - B beginning of an entity
  - I continues the entity
  - O word outside the entity
- For example
  - [‘Adam’, ‘Smith’, ‘works’, ‘for’, ‘IBM’, ‘,’, ‘London’, ‘.’]
  - Without BIO: [PER, PER, O, O, ORG, O, GEO, O]
  - With BIO: [B_PER, I_PER, O, O, B_ORG, O, B_GEO, O]

MLP for NER

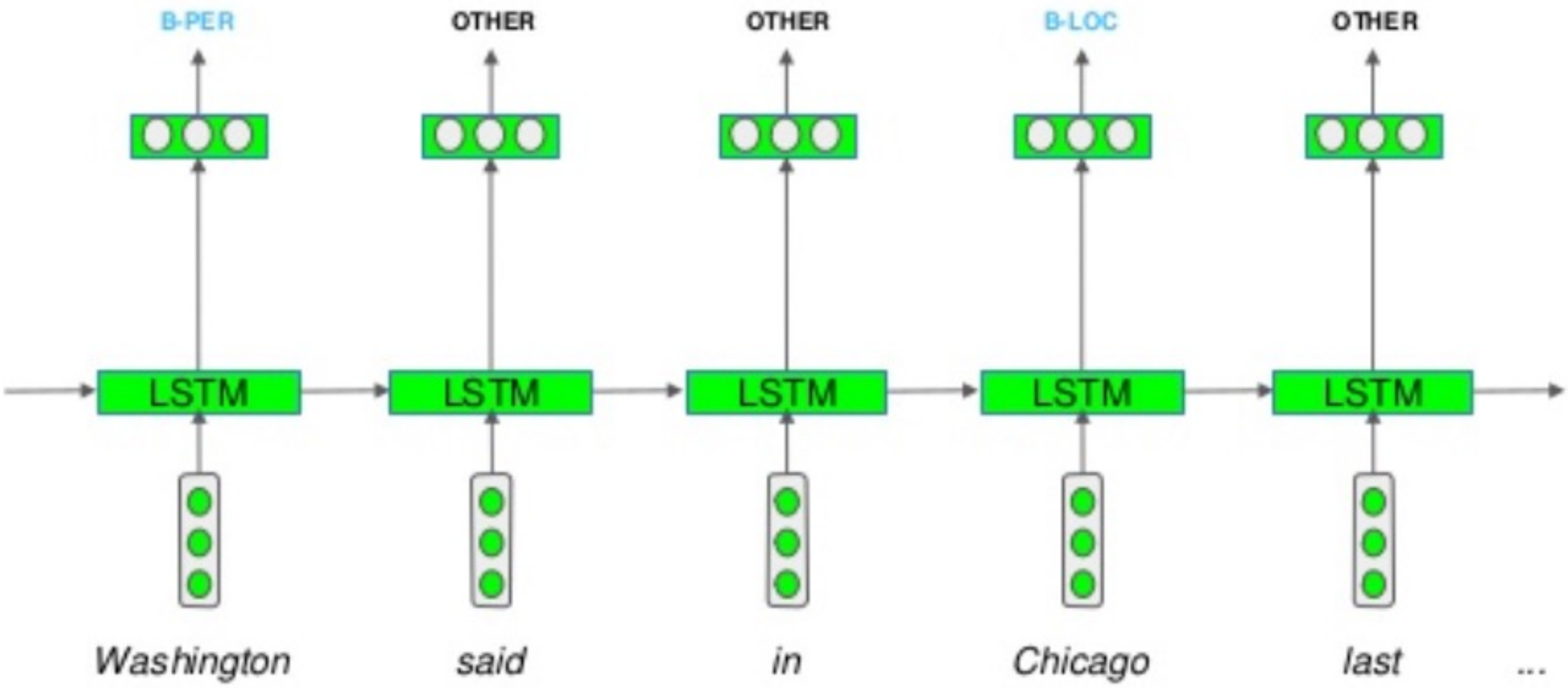


- Activation function for output: softmax
- Labels are OneHotEncoded

Recurrent neural network for NER

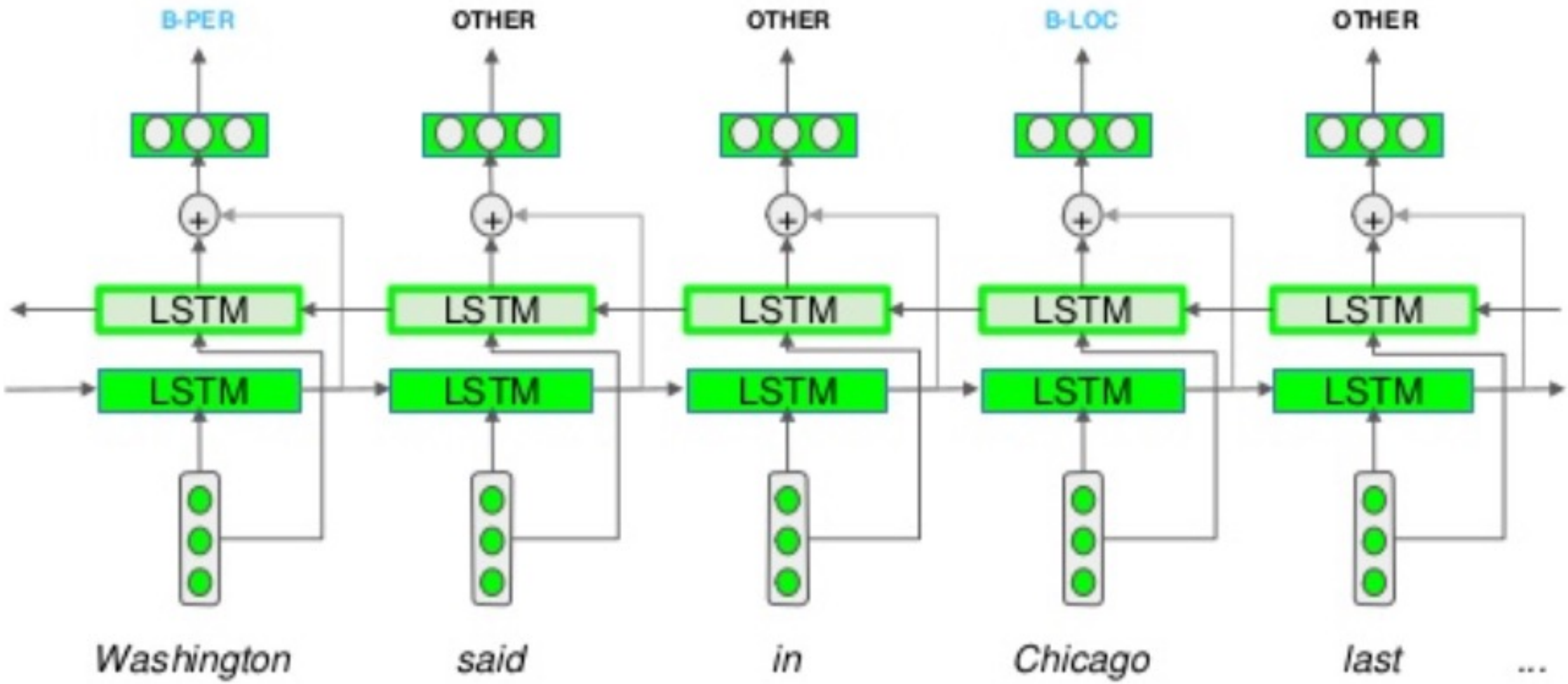


Recurrent neural network for NER (unfolded)



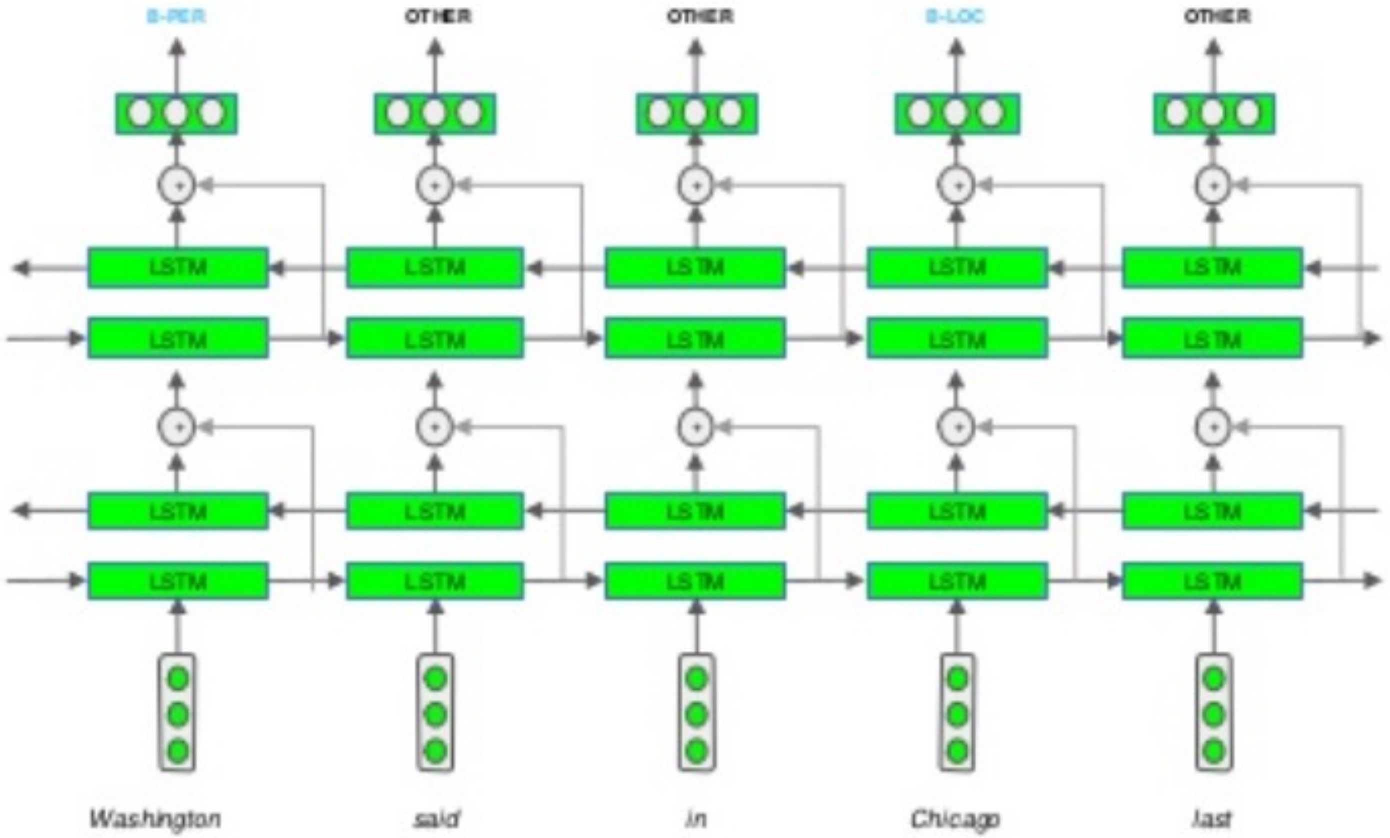
- Activation function for output: softmax
- Labels are OneHotEncoded

Bi directional recurrent neural network for NER



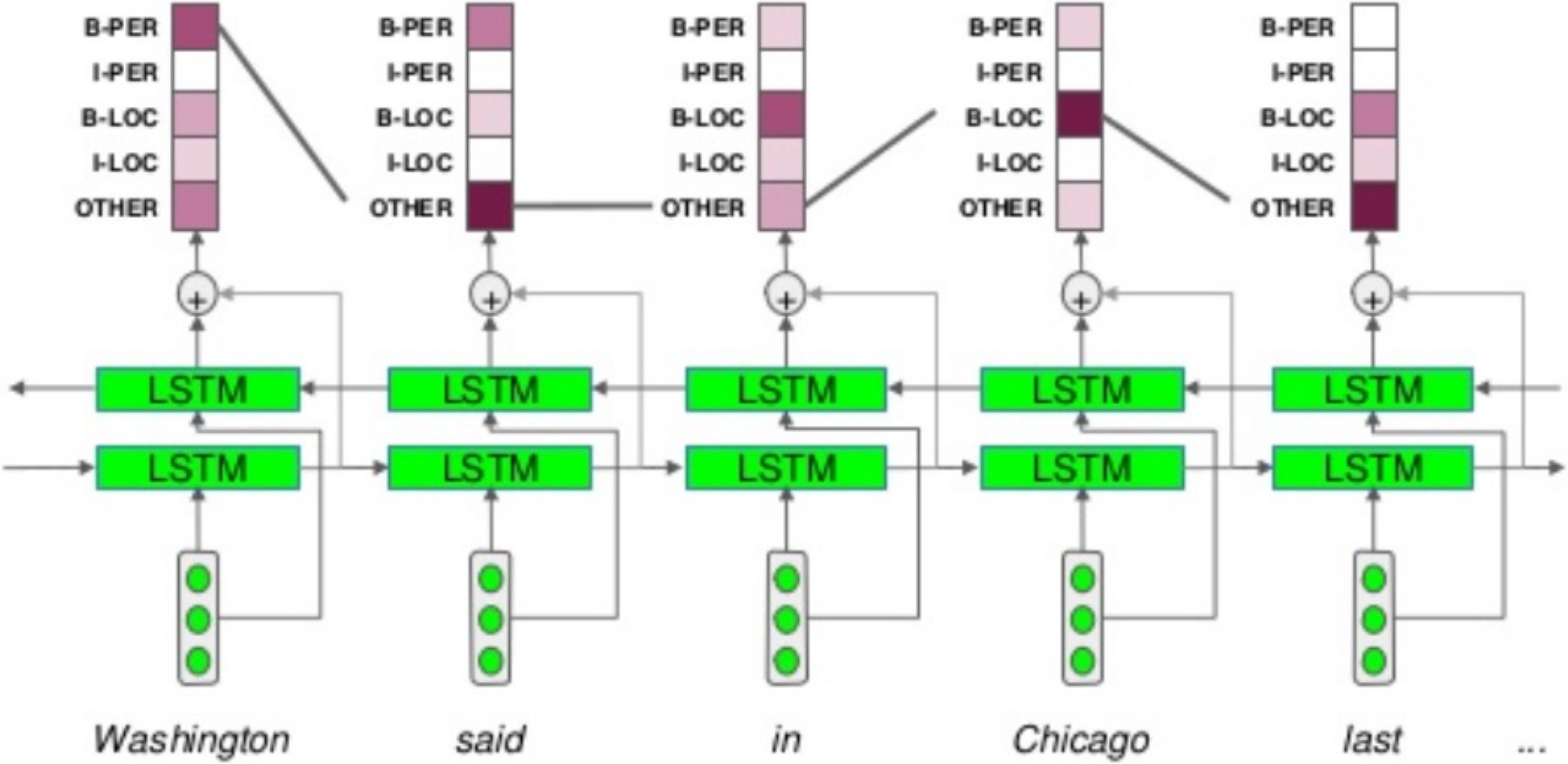
- Activation function for output: softmax
- Labels are OneHotEncoded

Stacked Bi-RNN



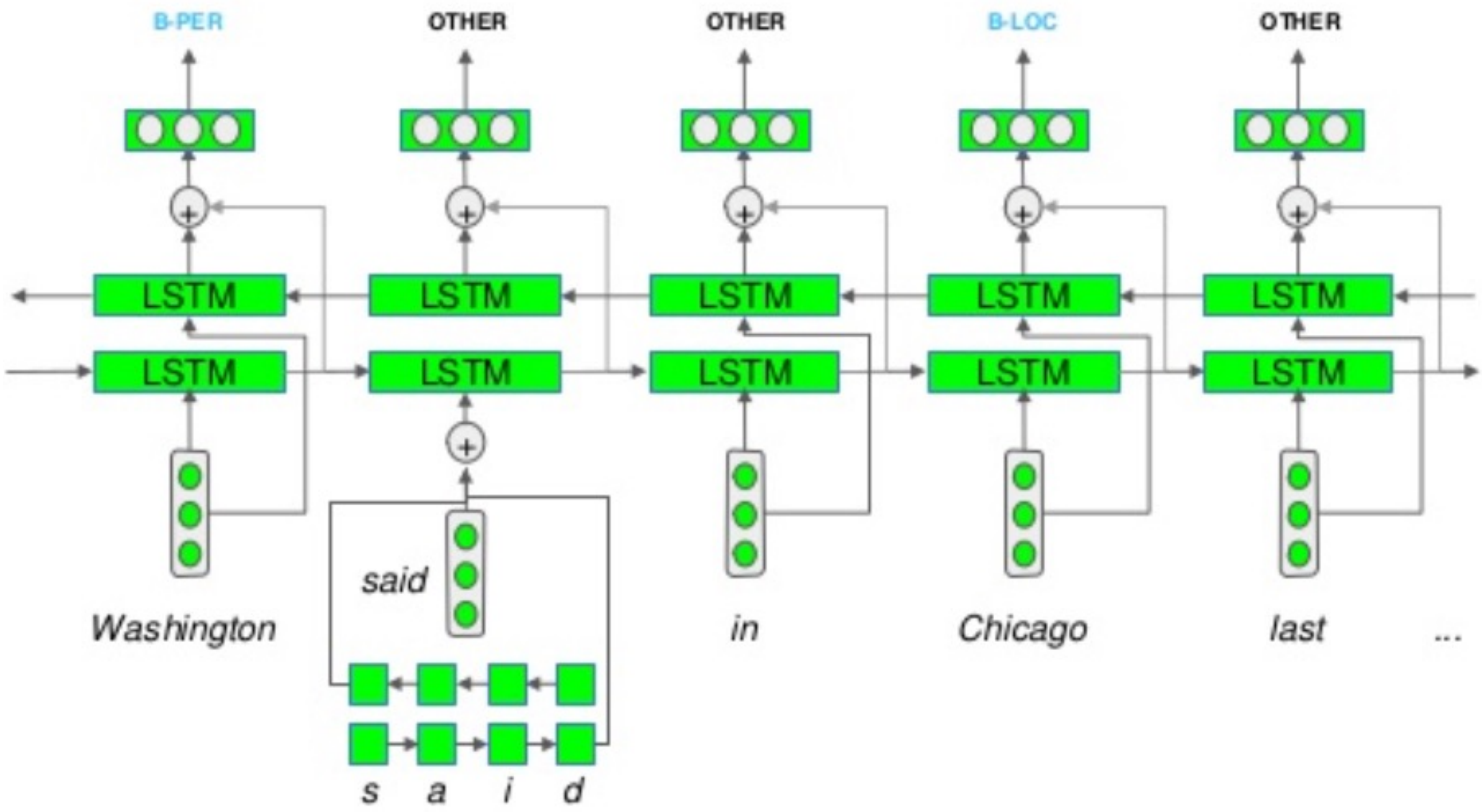
- Activation function for output: softmax
- Labels are OneHotEncoded

Bi-RNN + CRF



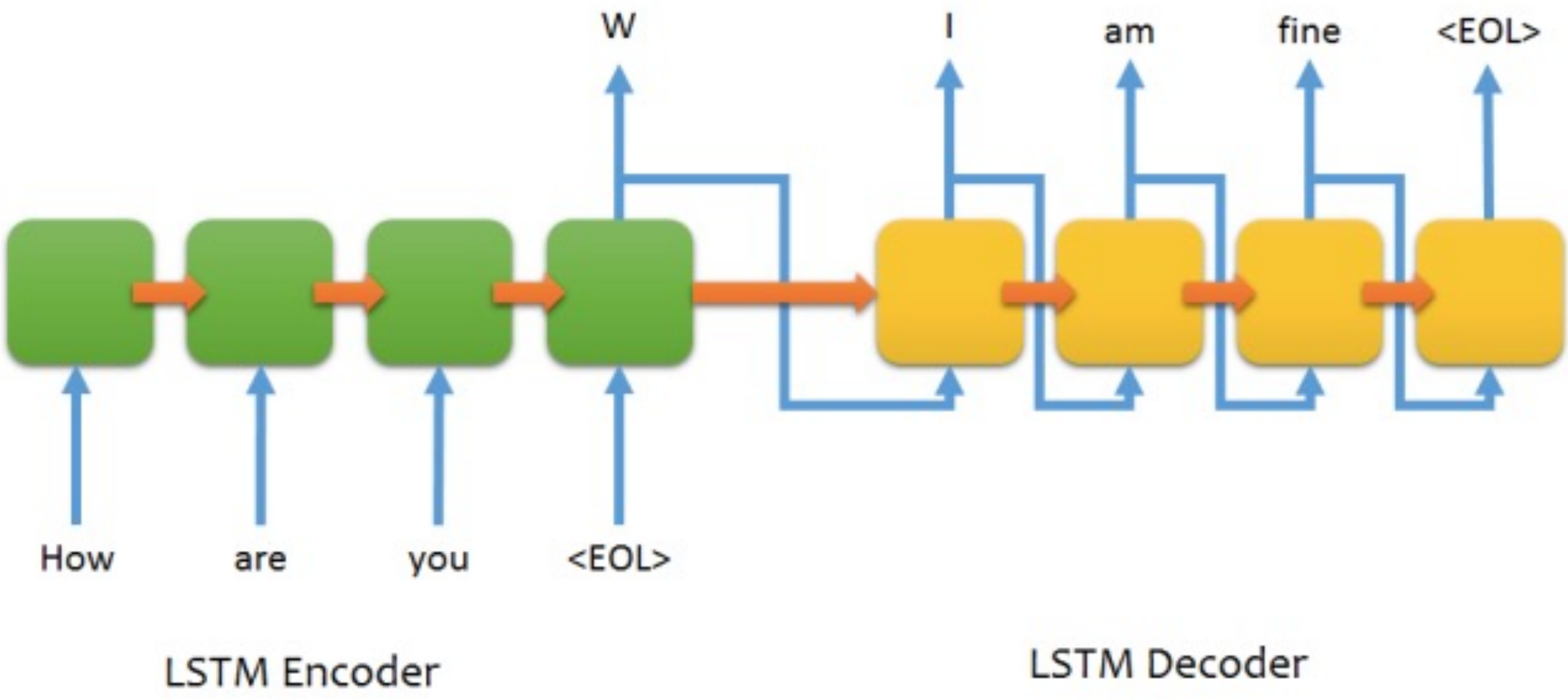
- CRF output activation function

Multi-level encoding
char encoding + word encoding



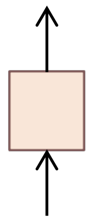
Other use of RNN
→ Sequence2sequence model

- Used for
  - Translation
  - Chatbot



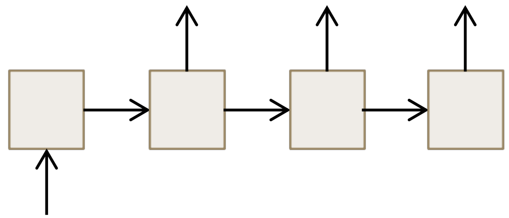
Some use of RNN
→ Input – Output Scenarios

Single - Single



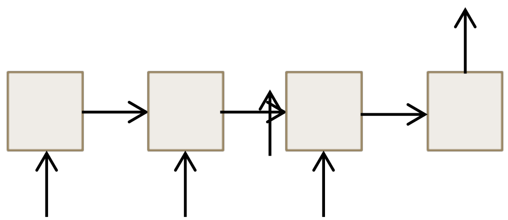
One input
One output
→ Feed-forward network

Single - Multiple



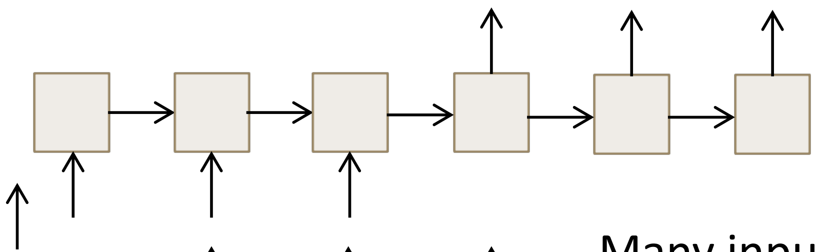
One input
Many output
Image annotation

Multiple - Single

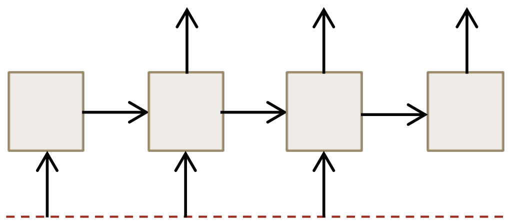


Many input
Many output
Text classification / Sentiment analysis

Multiple – Multiple



Many input / Many output
length(input)!=length(output)
Translation
Chat bot



Many input / Many output
length(input)==length(output)
Named Entity Recognition
Part-Of-Speech tagging

Other Useful Resources / References

- http://cs231n.stanford.edu/slides/winter1516_lecture10.pdf
- http://www.cs.toronto.edu/~rgrosse/csc321/lec10.pdf

- R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, ICML 2013
- S. Hochreiter, and J. Schmidhuber, Long short-term memory, Neural computation, 1997 9(8), pp.1735-1780
- F.A. Gers, and J. Schmidhuber, Recurrent nets that time and count, IJCNN 2000
- K. Greff , R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, LSTM: A search space odyssey, IEEE transactions on neural networks and learning systems, 2016
- K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014
- R. Jozefowicz, W. Zaremba, and I. Sutskever, An empirical exploration of recurrent network architectures, JMLR 2015