Recurrent Neural Networks (RNNs)

Michel RIVEILL
michel.riveill@univ-cotedazur.fr

Motivation

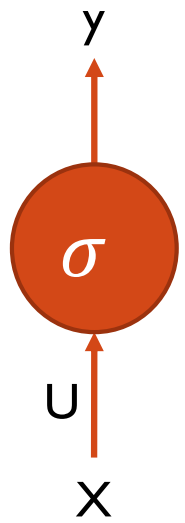
- Humans don't start their thinking from scratch every second
  - Thoughts have persistence
- Traditional neural networks can't characterize this phenomena
  - Ex: classify what is happening at every point in a movie
  - How a neural network can inform later events about the previous ones
- Recurrent 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?

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

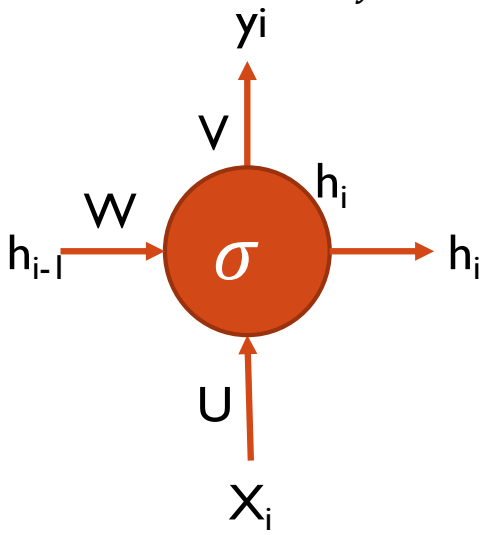
From vanilla NN to recurrent NN

- Vanilla cell
  - $y = \sigma(W.X)$



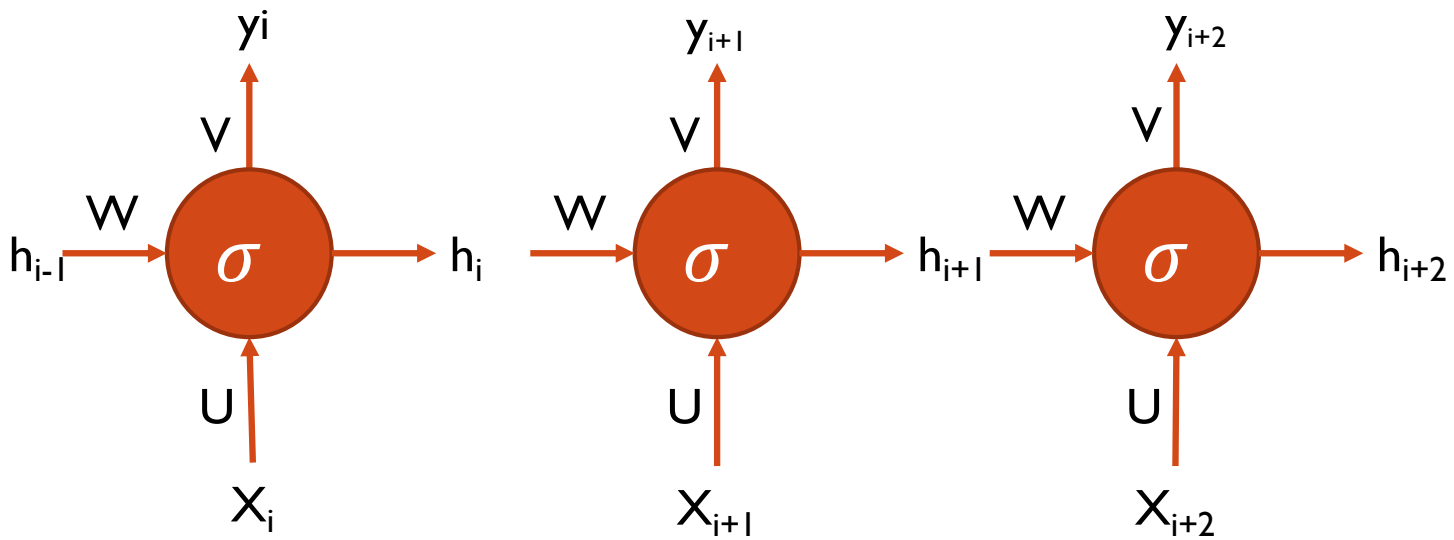
From vanilla NN to recurrent NN

- Vanilla cell
  - $y = \sigma(U.X)$
- Recurrent cell
  - Add an hidden state (internal state): $h_i = \sigma_h(W.h_{i\_1\_} U.Xi)$
    - σ_h *is generally tanh or ReLU*
  - The output depend of the current entry and the previous internal state:
    - $y_i = \sigma_y(V.hi)$, σ_y *is could be sigmoid, sofmax, tanh or ReLU depend of the problem*



From vanilla NN to recurrent NN

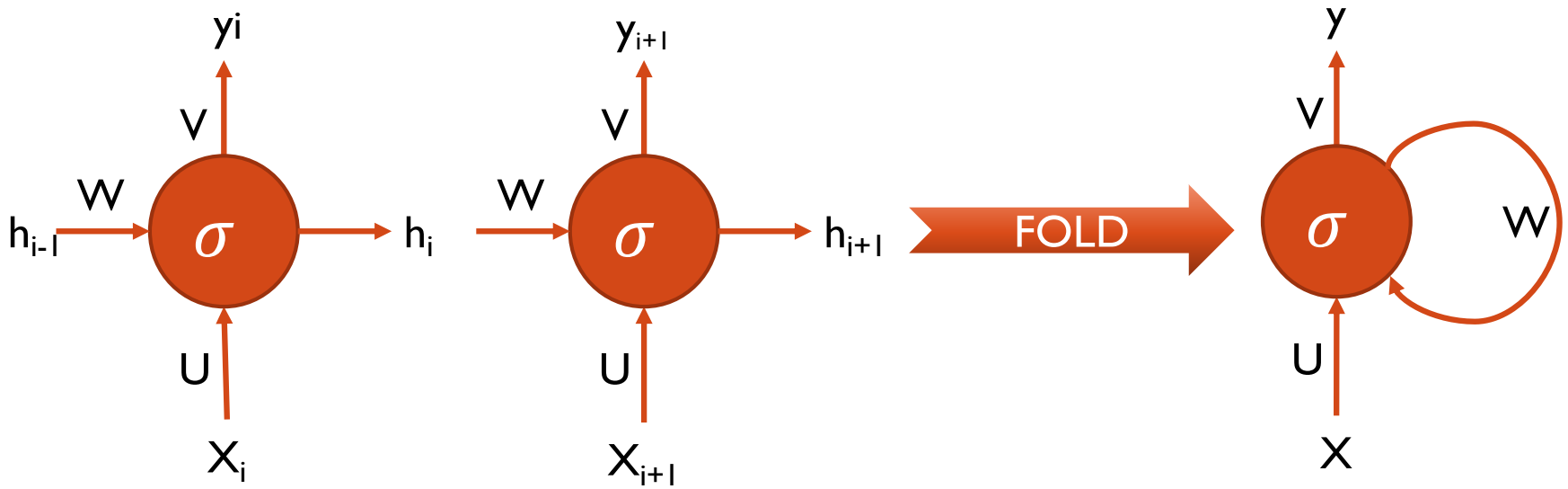
- Vanilla cell
  - $y = \sigma(U.X)$
- Recurrent cell
  - $h_i = \sigma_h(W.h_{i\_1} + U.Xi)$
  - $y_i = \sigma_y(V.hi)$
- Recurrent layer



8

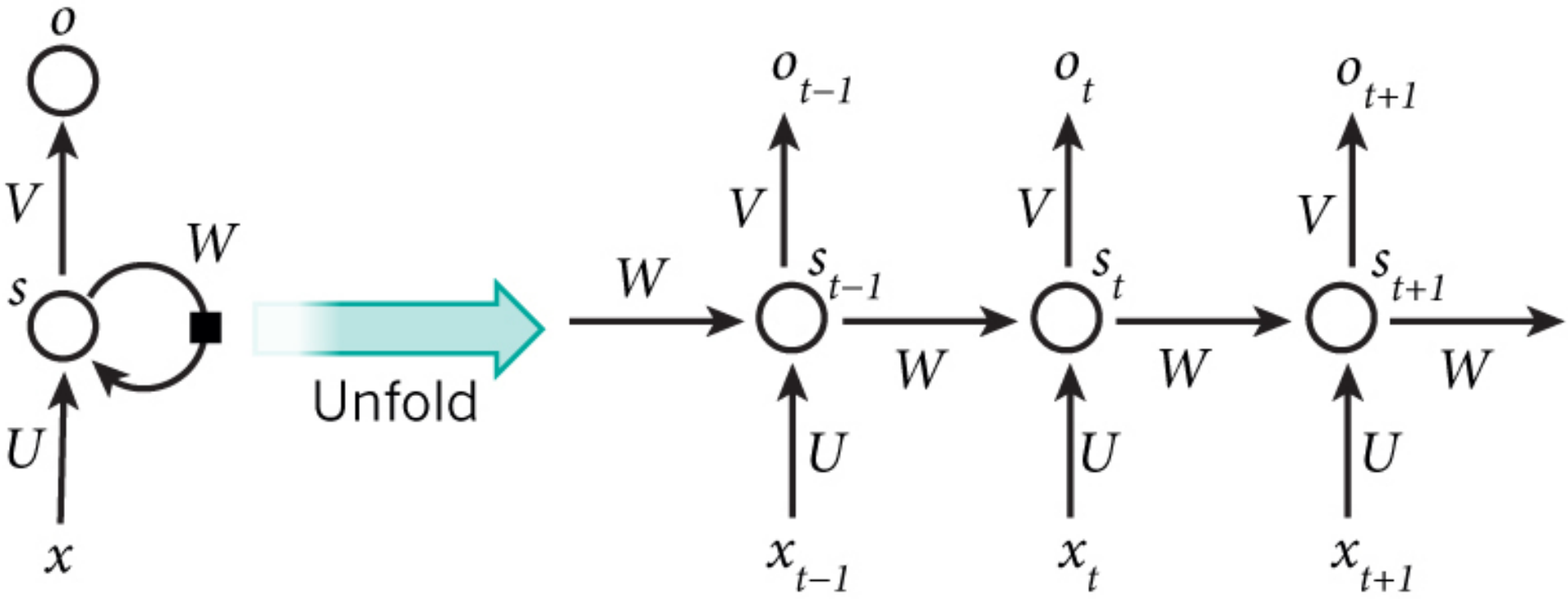
From vanilla NN to recurrent NN

- Vanilla cell
  - $y = \sigma(U.X)$
- Recurrent cell
  - $h_i = \sigma_h(W.h_{i\_1} + U.Xi)$
  - $y_i = \sigma_y(V.hi)$
- Recurrent layer



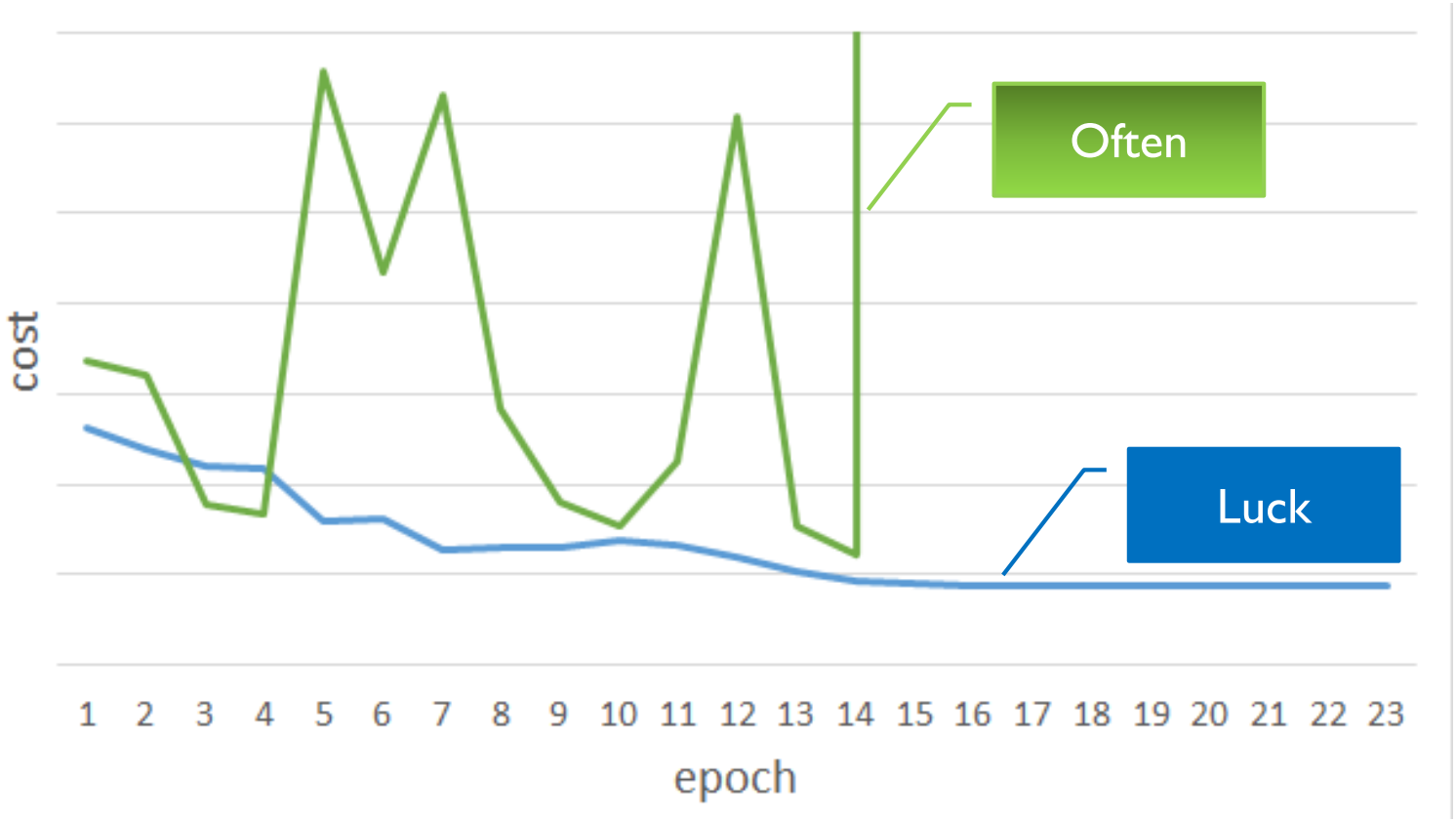
How does RNN reduce complexity

- In the very first approach
  - V = I (output = hidden state)
- All the matrix weights are shared



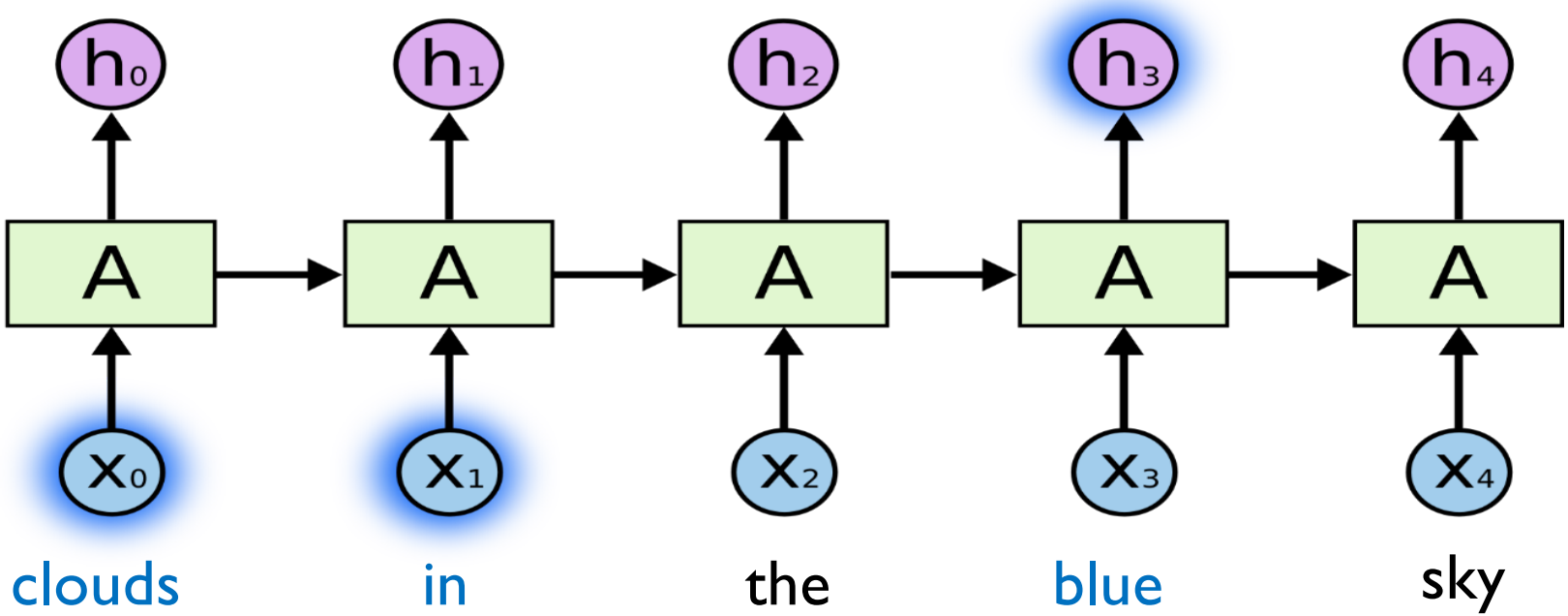
Problems with naive RNN

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



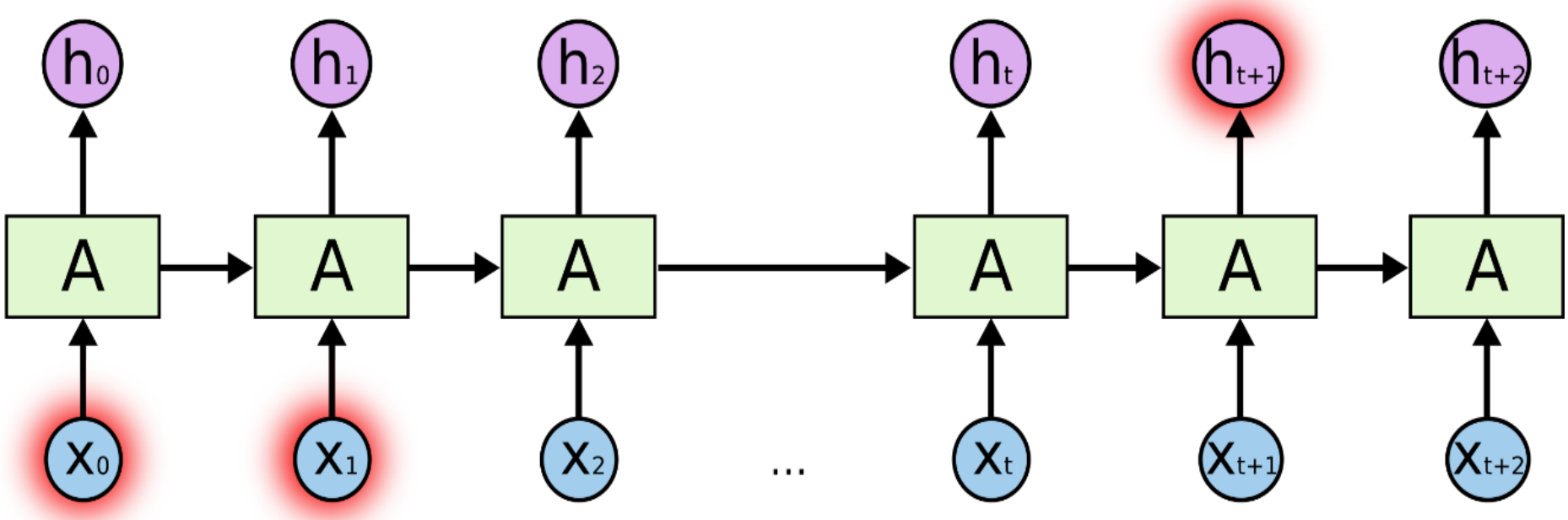
Long Short Term Memory

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



Long Short Term Memory

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

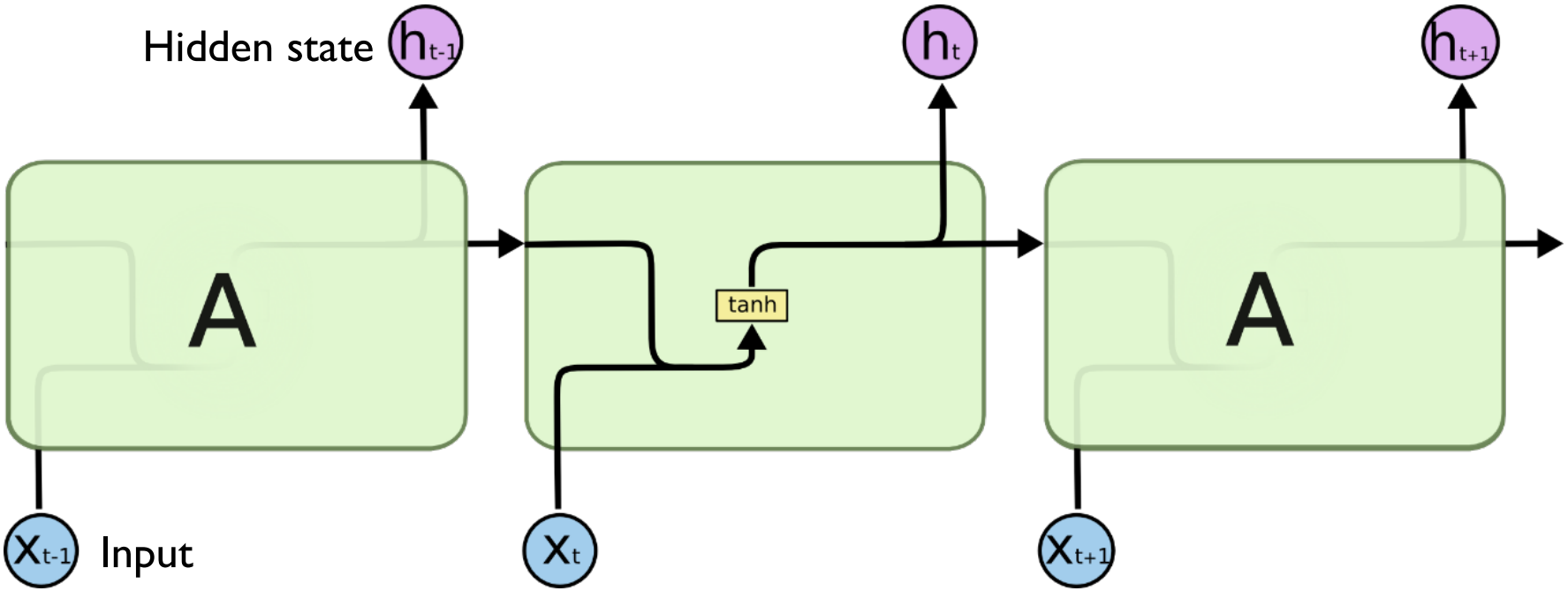


I grew up in France...

I speak French...

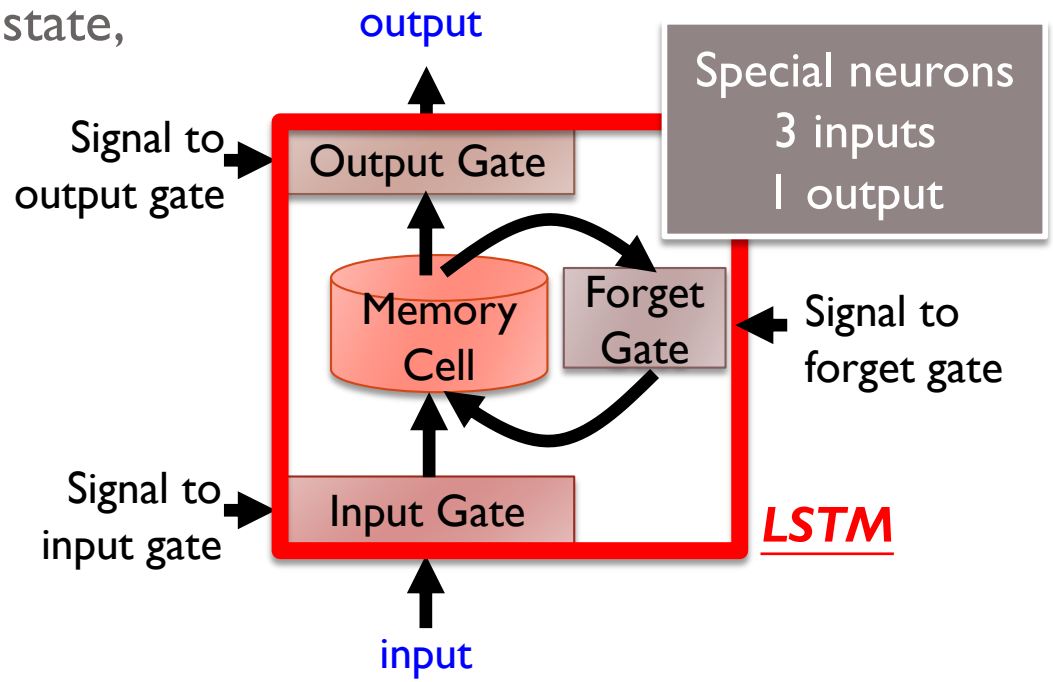
From Vanilla to LSTM Cells

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



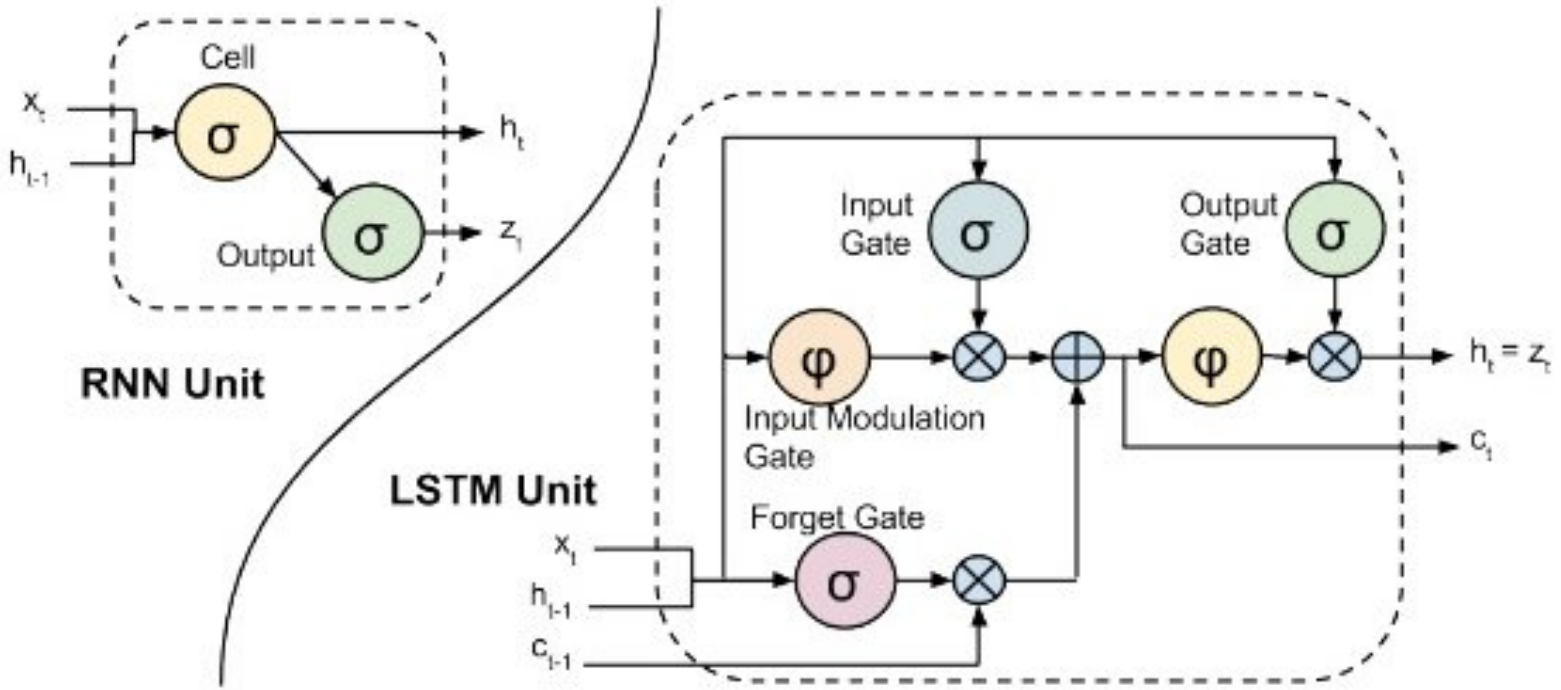
LSTM Cells

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



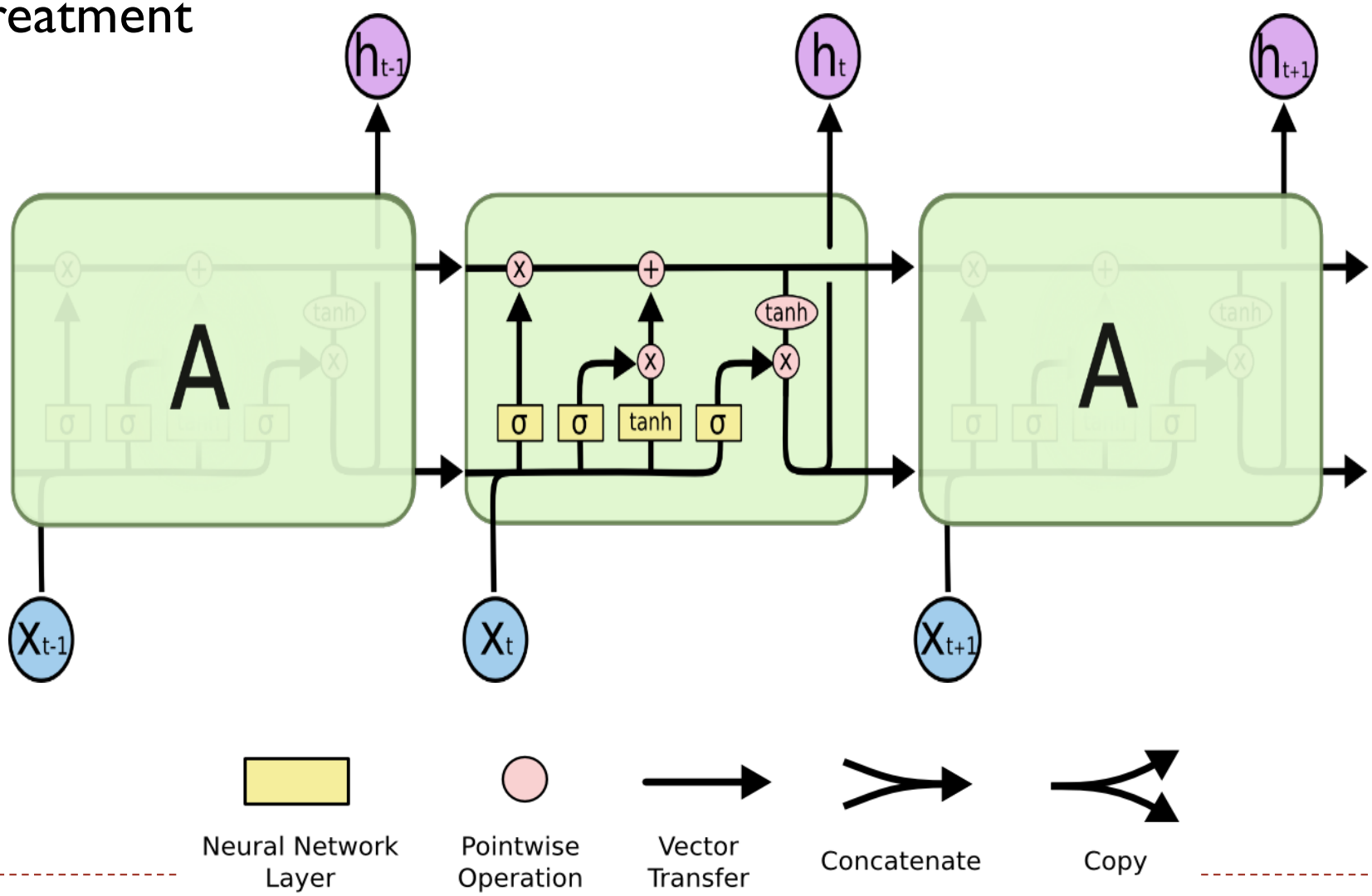
The control signals typically come from perceptrons

From Vanilla to LSTM Cells

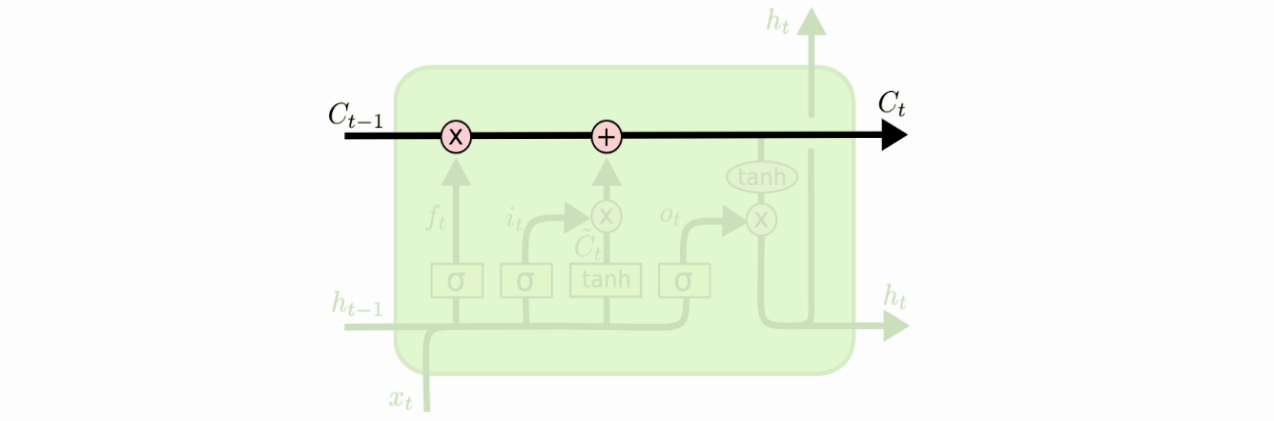


LSTM Cells

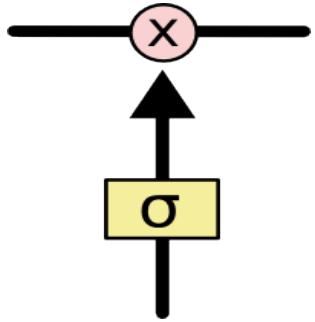
- Concretely, recurrence in an LSTM cell involves 4 levels of treatment



The details

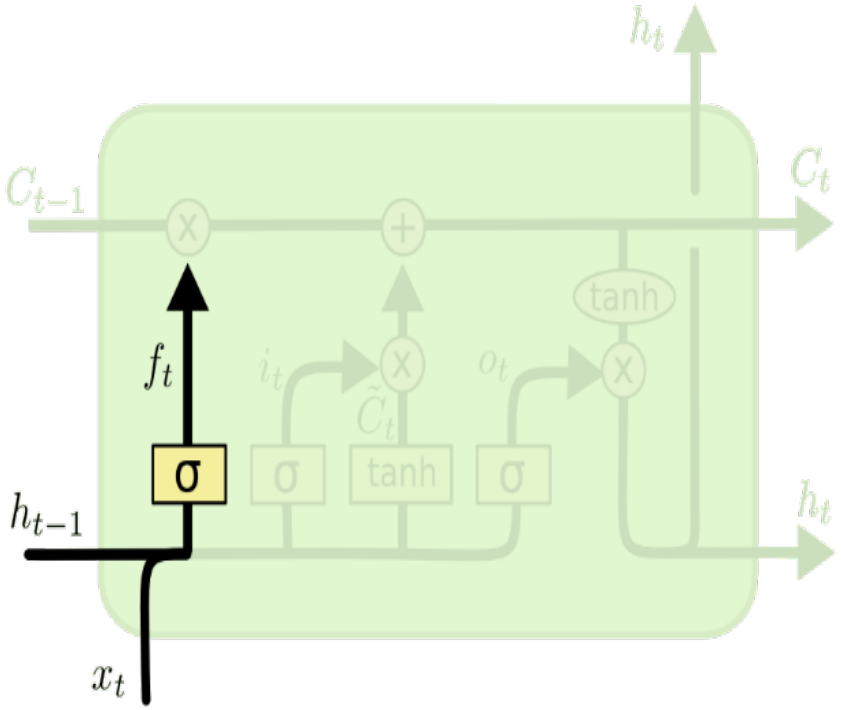


An LSTM cell acts as an information conveyor controlled by configurable gates (by learning).



Each gate is a network in itself (perceptron)

Forget gate

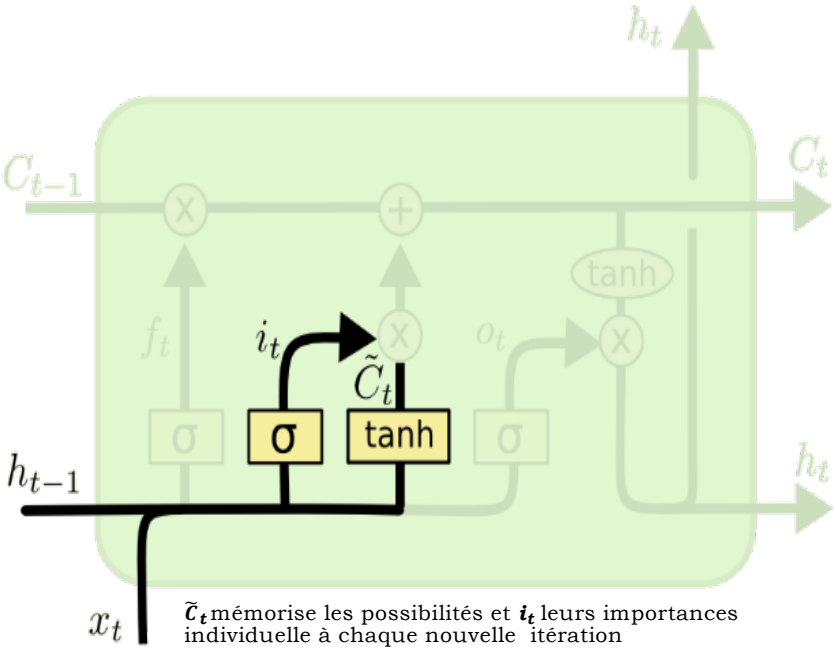


Decides the contribution of the previous context to the exit of the network

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Ex.: dans la prédiction des mots d'une nouvelle phrase, il n'est pas utile de garder le genre du sujet dans la phrase précédente

Input gate



Decides the contribution of the memory at the output of the network

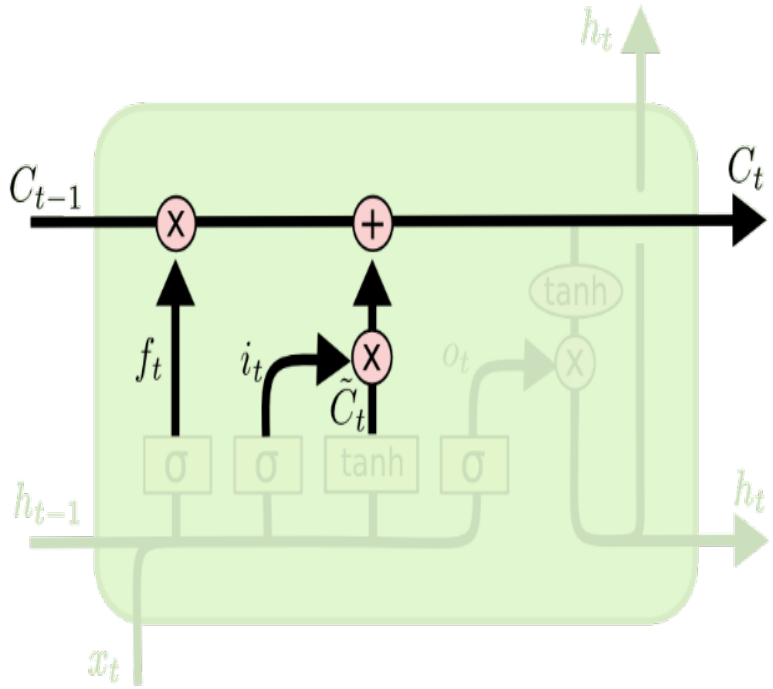
$\tilde{C}_t$ is the equivalent of h_t in a standard RNN

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Ex.: dans la prédiction des mots d'une nouvelle phrase, il est utile de remplacer le genre du sujet de la phrase précédente par le nouveau.

Cell update

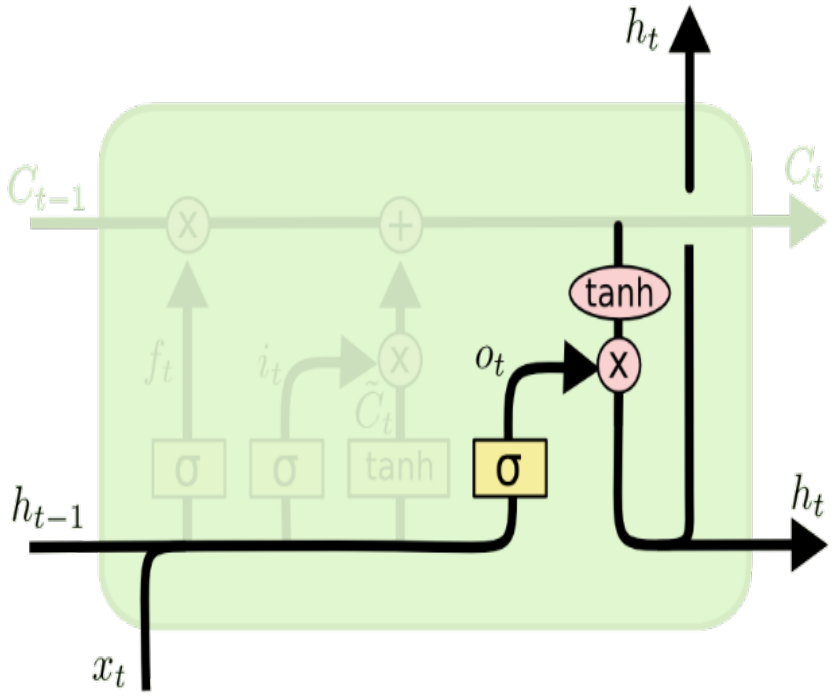


We combine what has been retained from the previous state and what is used from the current memory.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Agit comme une nouvelle entrée au réseau adaptée un contexte

Output gate



Decides what to make public of the new context

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Ex.: dans la prédiction des mots d'une nouvelle phrase, il peut être utile de retenir la pluralité du sujet à la vue d'un verbe, ce qui peut simplifier la conjugaison après.

Put all together

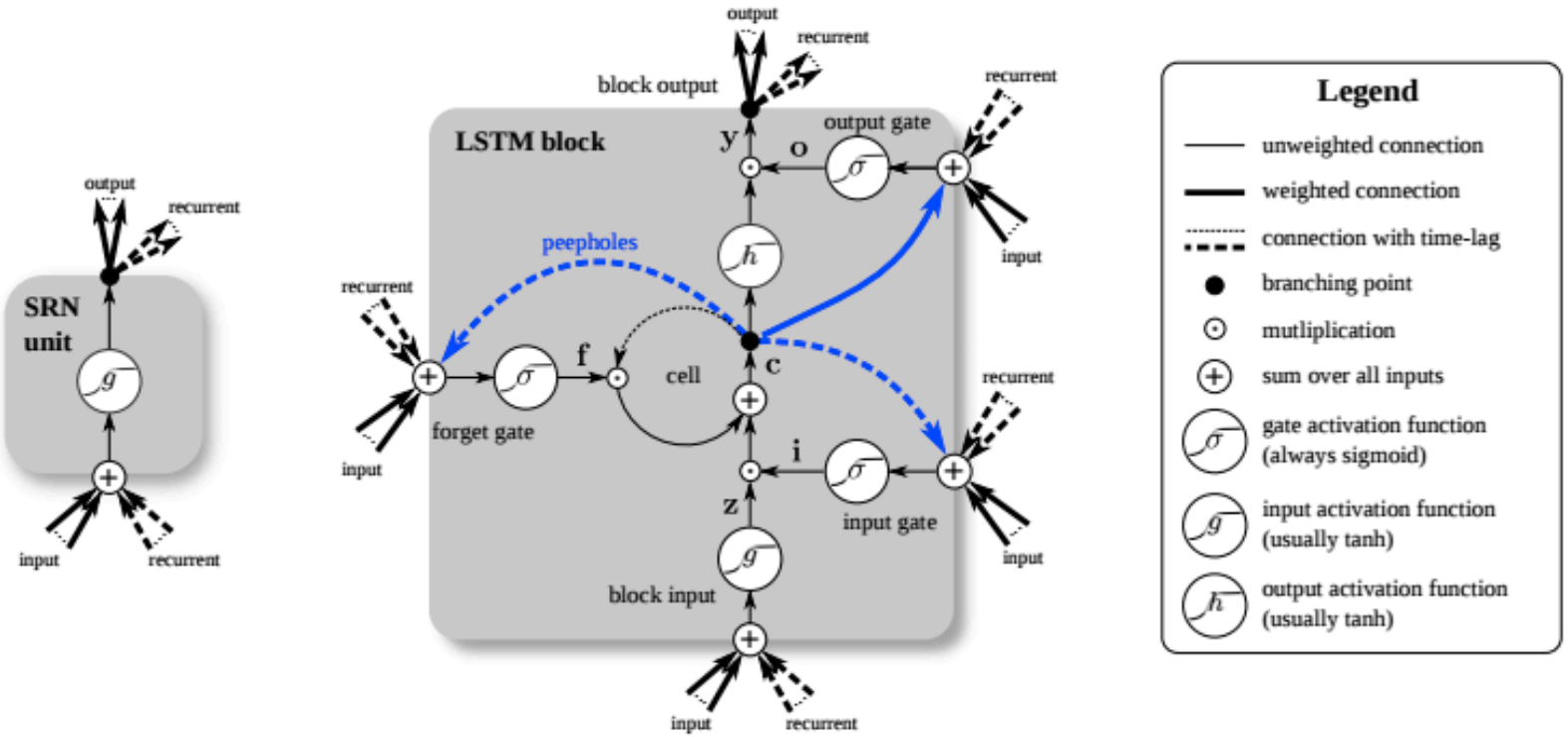
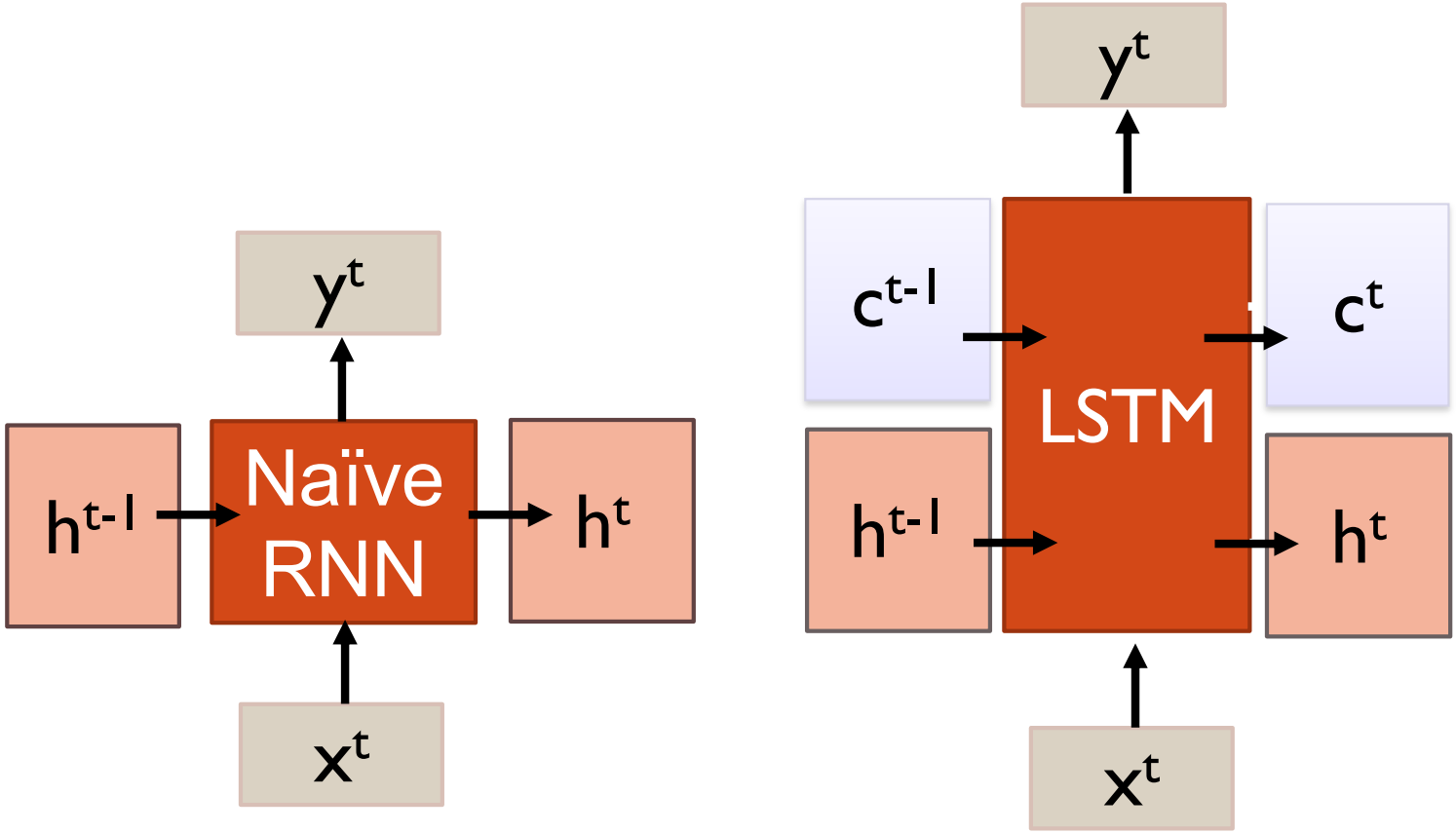


Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

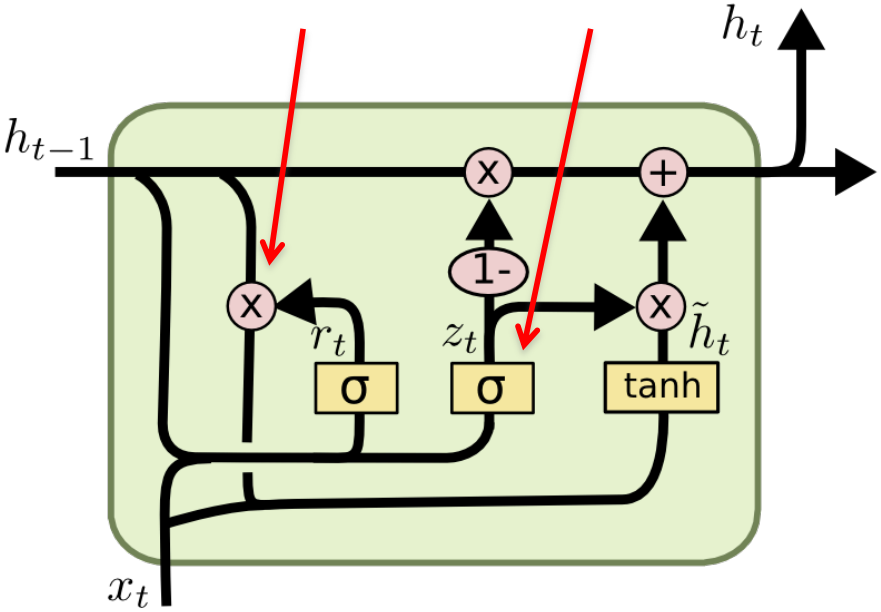
Naïve RNN vs LSTM



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 (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.

There are many other variants too.

LSTM vs GRU

Unit	# of Units	# of Parameters
Polyphonic music modeling		
LSTM	36	$\approx 19.8 \times 10^3$
GRU	46	$\approx 20.2 \times 10^3$
tanh	100	$\approx 20.1 \times 10^3$
Speech signal modeling		
LSTM	195	$\approx 169.1 \times 10^3$
GRU	227	$\approx 168.9 \times 10^3$
tanh	400	$\approx 168.4 \times 10^3$

Table 1: The sizes of the models tested in the experiments.

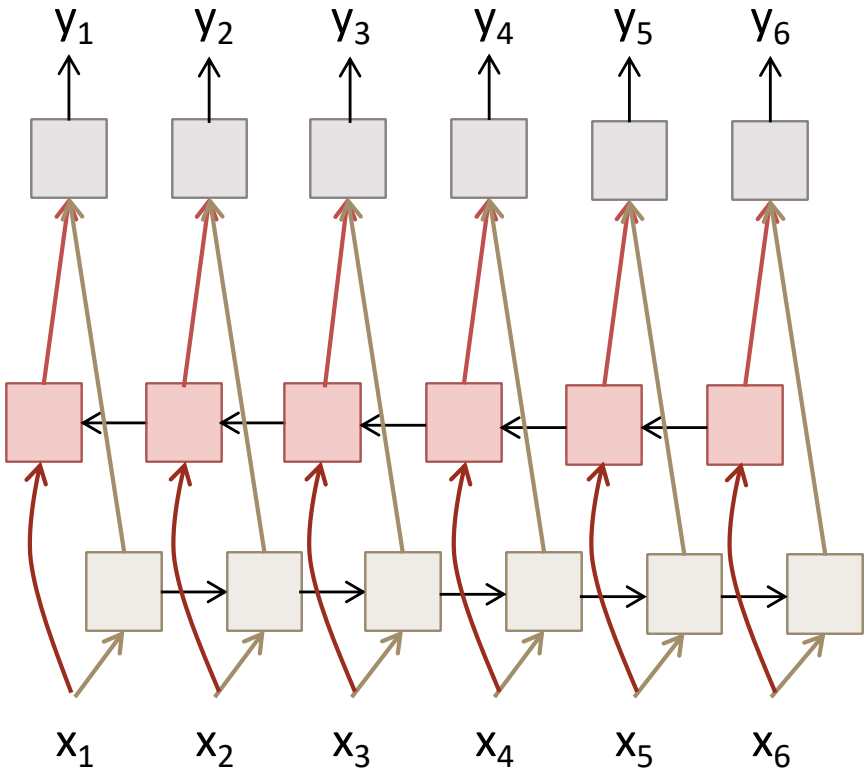
			tanh	GRU	LSTM
Music Datasets	Nottingham	train	3.22	2.79	3.08
		test	**3.13**	3.23	3.20
	JSB Chorales	train	8.82	6.94	8.15
		test	9.10	**8.54**	8.67
	MuseData	train	5.64	5.06	5.18
		test	6.23	**5.99**	6.23
	Piano-midi	train	5.64	4.93	6.49
		test	9.03	**8.82**	9.03
Ubisoft Datasets	Ubisoft dataset A	train	6.29	2.31	1.44
		test	6.44	3.59	**2.70**
	Ubisoft dataset B	train	7.61	0.38	0.80
		test	7.62	**0.88**	1.26

Table 2: The average negative log-probabilities of the training and test sets.

- From: https://medium.com/paper-club/grus-vs-lstms-e9d8e2484848

Bi-directional RNNs

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



- Popular in speech recognition

RNN cell in Keras

Keras Long Short-Term Memory Cell

- **from** 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.

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)

Some use of RNN
→ Sentiment analysis

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

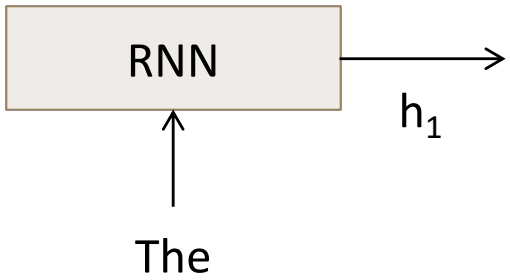
  …

  as positive or negative

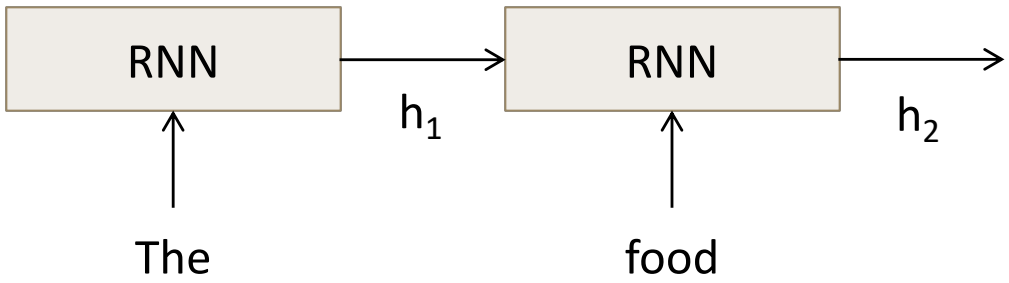
- Inputs:
  - Multiple words, one or more sentences
- Outputs:
  - Positive / Negative classification

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

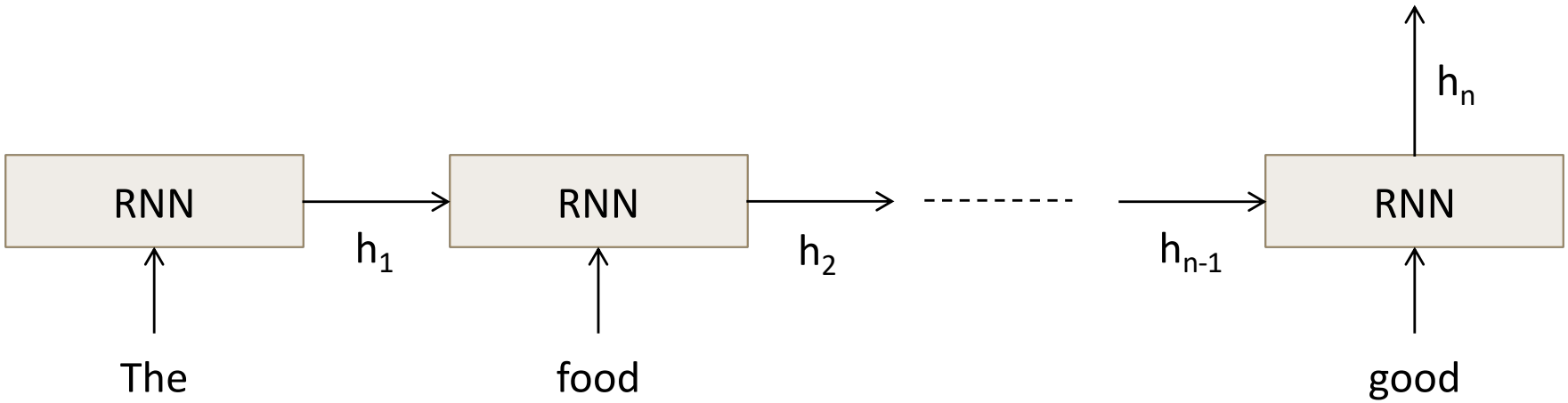
Sentiment analysis



Sentiment analysis

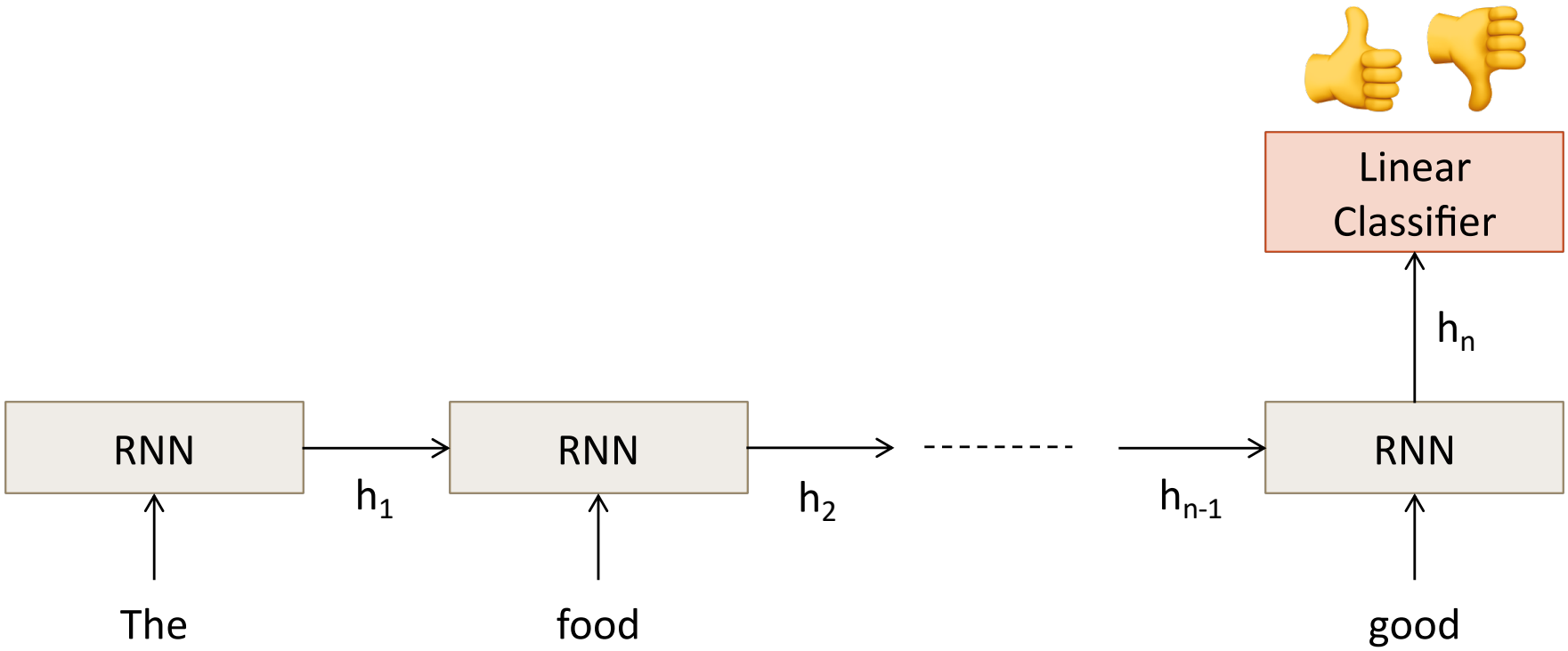


Sentiment analysis



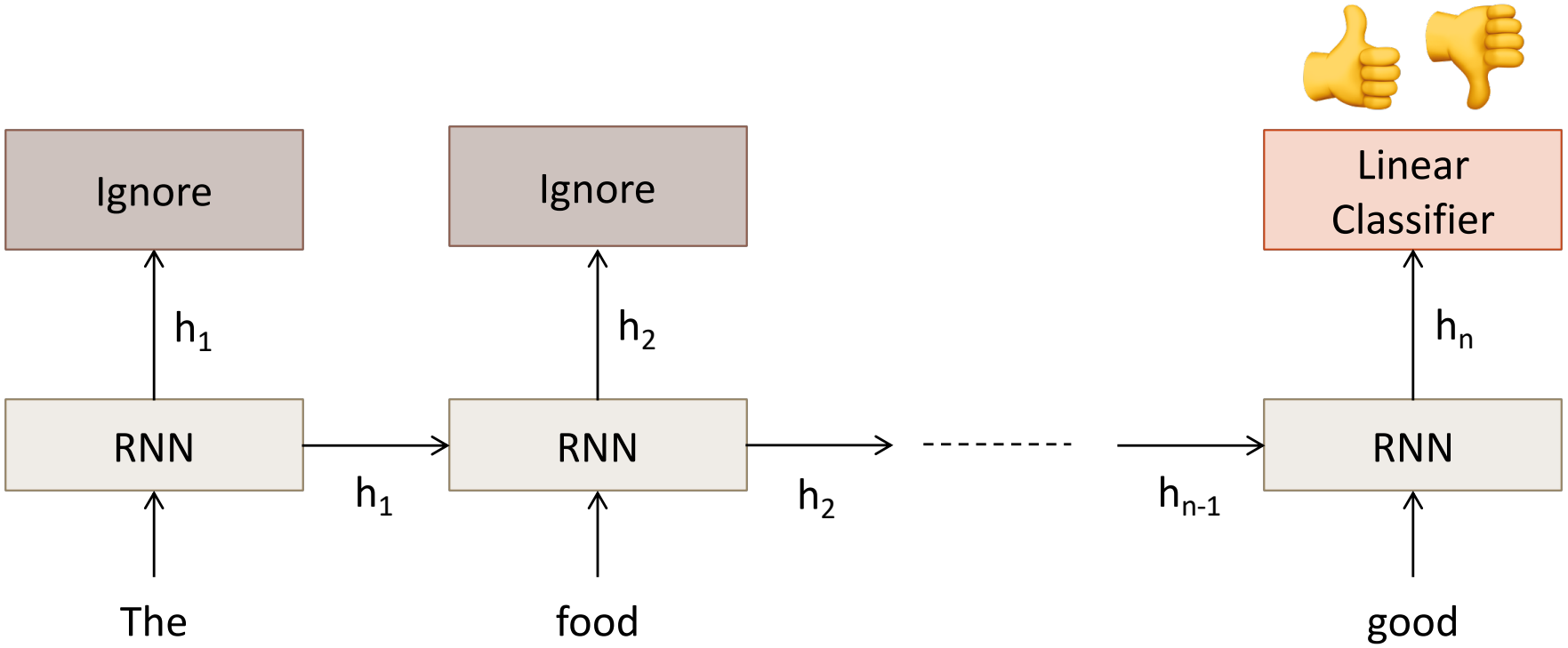
Sentiment analysis

- retrieve only the last state



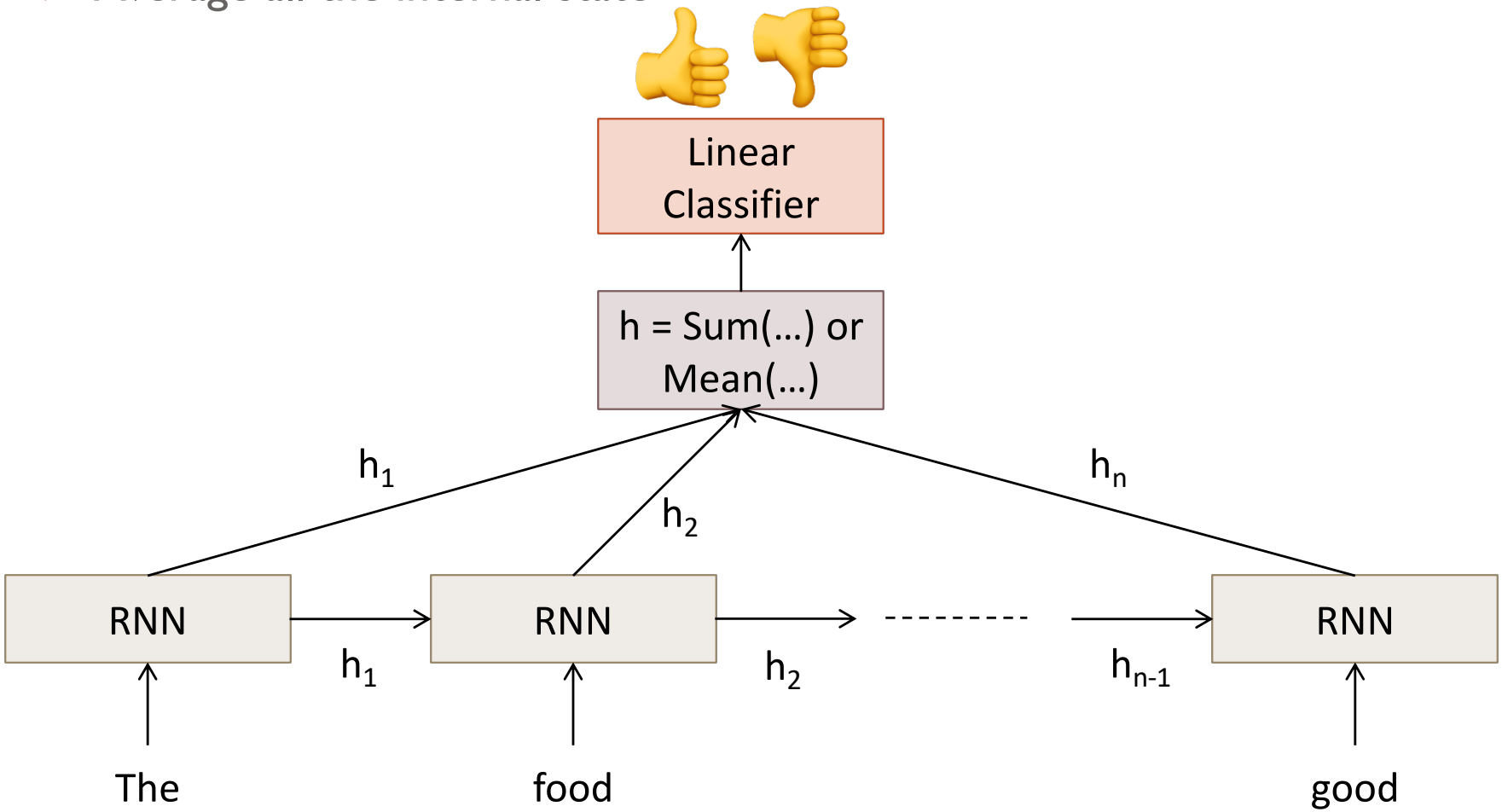
Sentiment Classification

- retrieve only the last state
- or retrieve the entire list but keep only the last one.



Sentiment analysis

- Other possible architecture
  - Average all the internal state



http://deeplearning.net/tutorial/lstm.html

Open source library

- To be used if you want to analyze text
  - but the data are not annotated
  - on the condition of finding a trained model on a nearby corpus.

- It is sometimes possible to "fine train" the model

- https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html

Open source library for Sentiment Analysis

- AFINN lexicon: https://github.com/fnielsen/afinn
  - Score beetween [-20, +20]

```
from afinn import Afinn
af = Afinn()

# compute sentiment scores (polarity)
sentiment_scores = [af.score(article) for article in corpus]

# compute polarity
sentiment_category = ['positive' if score > 0
                      else 'negative' if score < 0
                          else 'neutral'
                              for score in sentiment_scores]
```


58

Open source library for Sentiment Analysis

- *TextBlob: https://textblob.readthedocs.io/en/dev/*
  - *Score beetween [-1, +1]*

- **from textblob import TextBlob**

- **# compute sentiment scores (polarity)**
- **sentiment_scores = [TextBlob(article).sentiment.polarity**
  for article in corpus]
- **sentiment_category = ['positive' if score > 0**
  else 'negative' if score < 0
  else 'neutral'
  for score in sentiment_scores]

Some use of RNN
→ Named Entity Recognition

- Try to predict the categories of each word
  - **find** and **classify** names in text

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 .

NE Definition

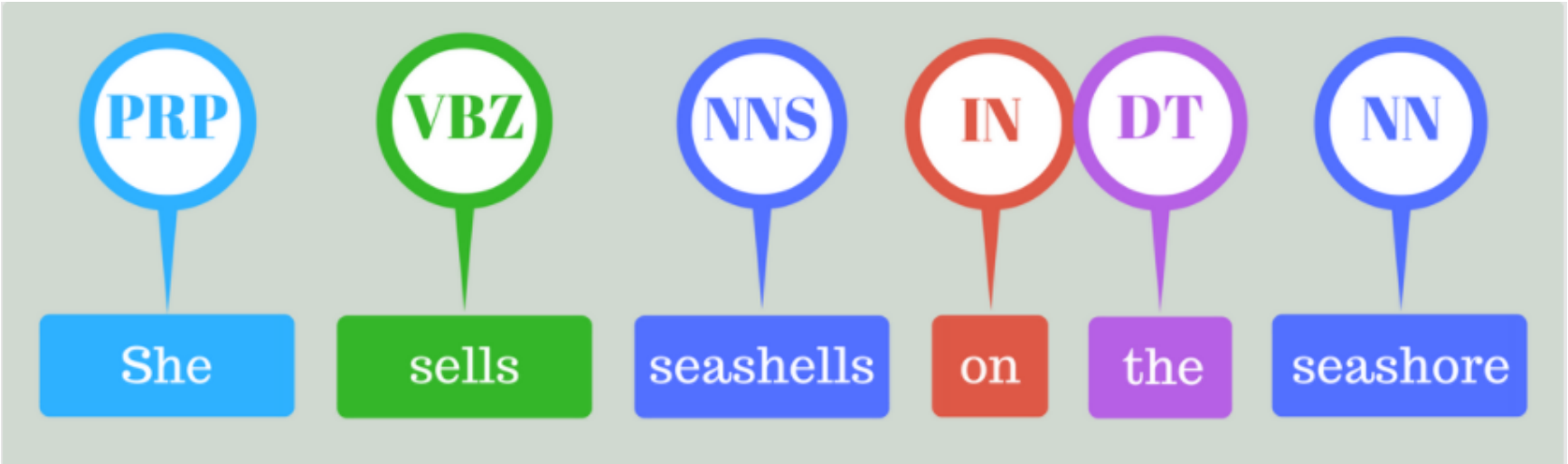
- NE involves identification of proper names in texts, and classification into a set of predefined categories of interest.
- Three universally accepted categories:
  - person, location and organisation
- Other common tasks:
  - recognition of date/time expressions,
  - measures (percent, money, weight etc),
  - email addresses etc.
- Other domain-specific entities:
  - names of drugs,
  - medical conditions,
  - names of ships,
  - bibliographic references etc.

- NE is not just matching text strings with pre-defined lists of names.
  - Recognises entities which are being used as entities in a given context.

- NE is not easy!

Some use of RNN
→ POS Tagging task

- Try to predict the role of each word in a sentence
  - Part-of-speech tagging - POS
  - role = nouns, verbs, adjectives, adverbs, etc.



- Same king of network than NER
  - Detail only NER

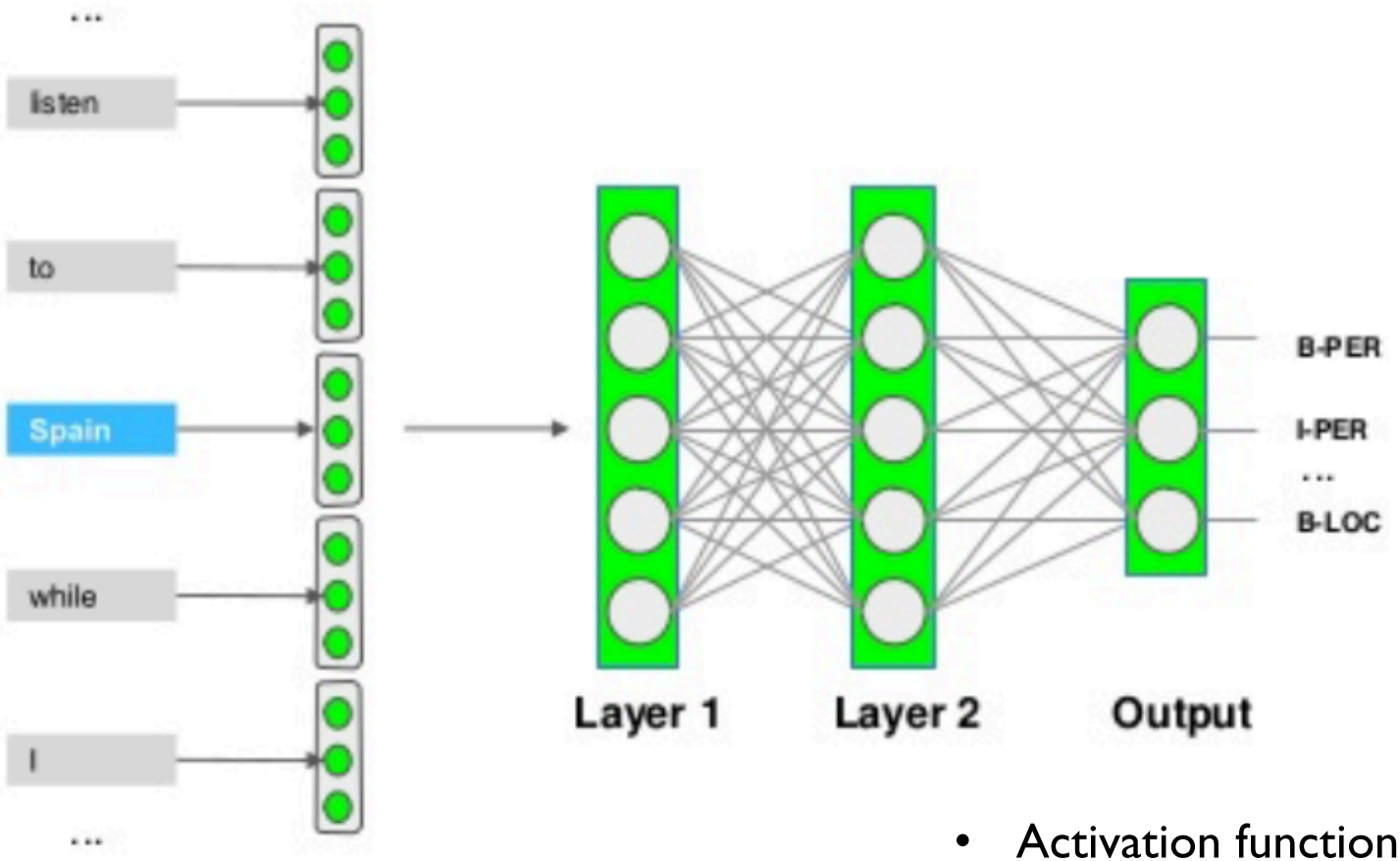
General approach for NER

- *Supervised* learning
  - labeled training examples
  - methods:
    - Hidden Markov Models / Conditional Random Fields,
    - k-Nearest Neighbors,
    - Decision Trees / Random Forest
    - AdaBoost
    - SVM
  - example: NE recognition, POS tagging, Parsing

Label representation – BIO tags

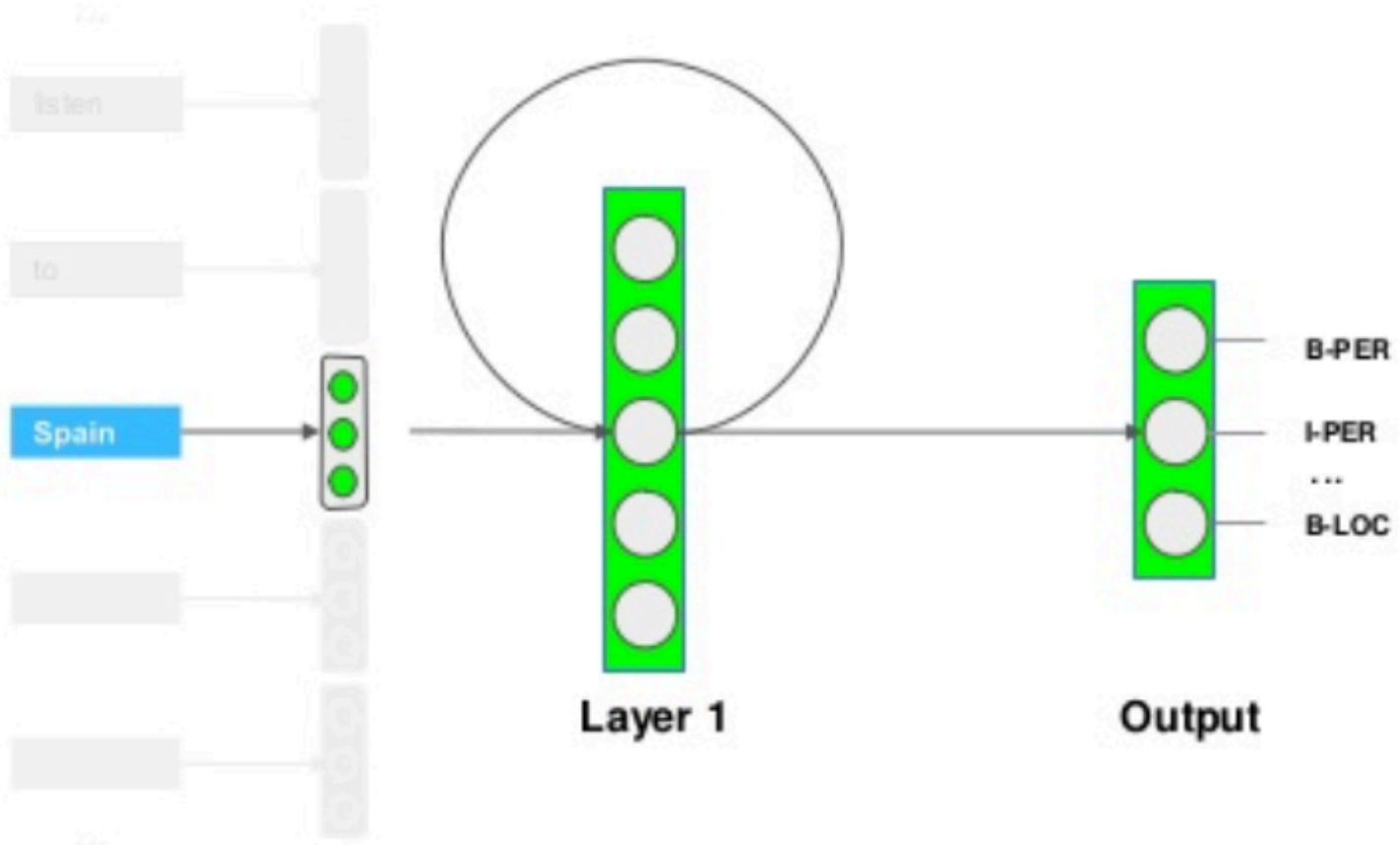
- Labels can be for words or groups of words
  - Adam Smith works for IBM , London .
- To represent this, a "BIO" representation is generally used.
  - B beginning of an entity
  - I continues the entity
  - O word outside the entity
- For example
  - [‘Adam’, ‘Smith’, ‘works’, ‘for’, ‘IBM’, ‘,’, ‘London’, ‘.’]
  - 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

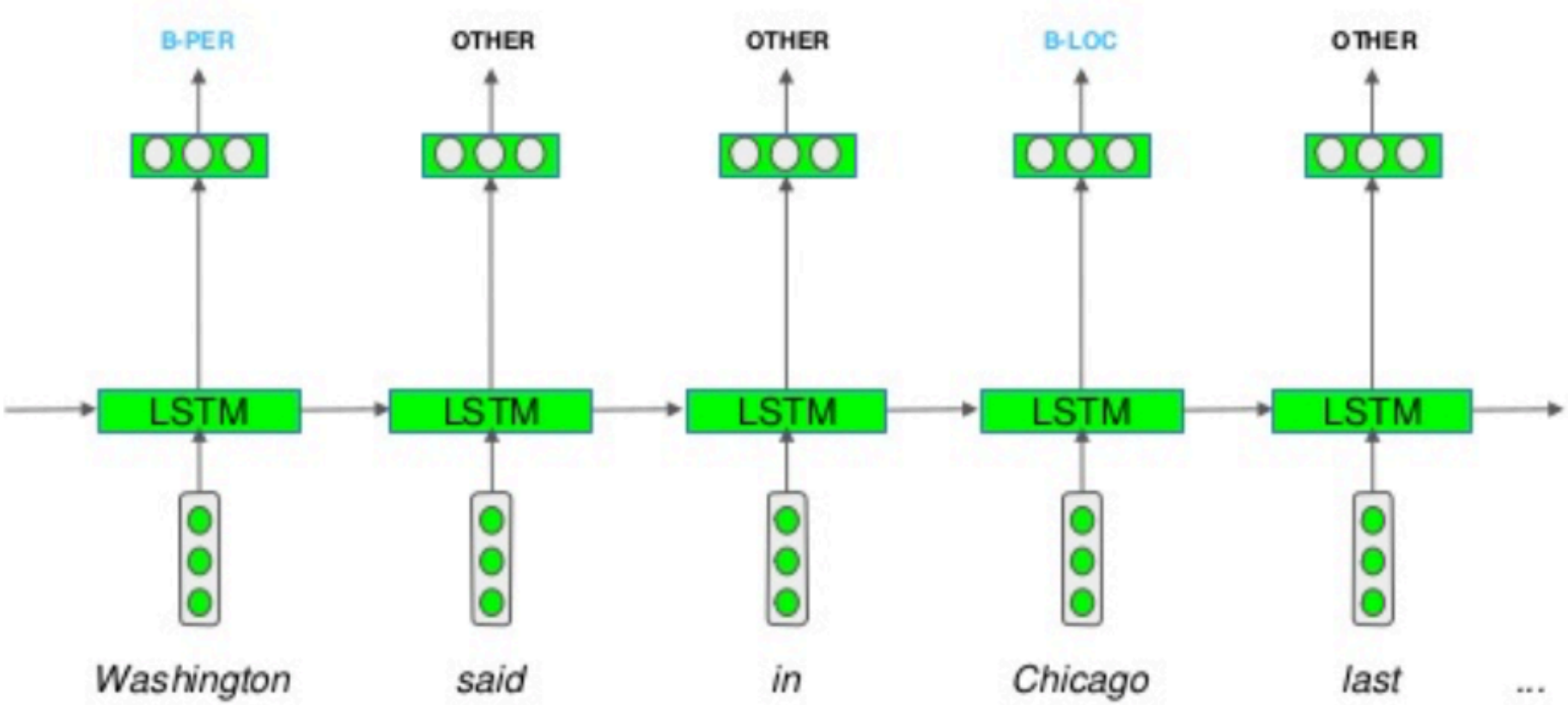


- Activation function for output: softmax
- Labels are OneHotEncoded

Recurrent neural network for NER

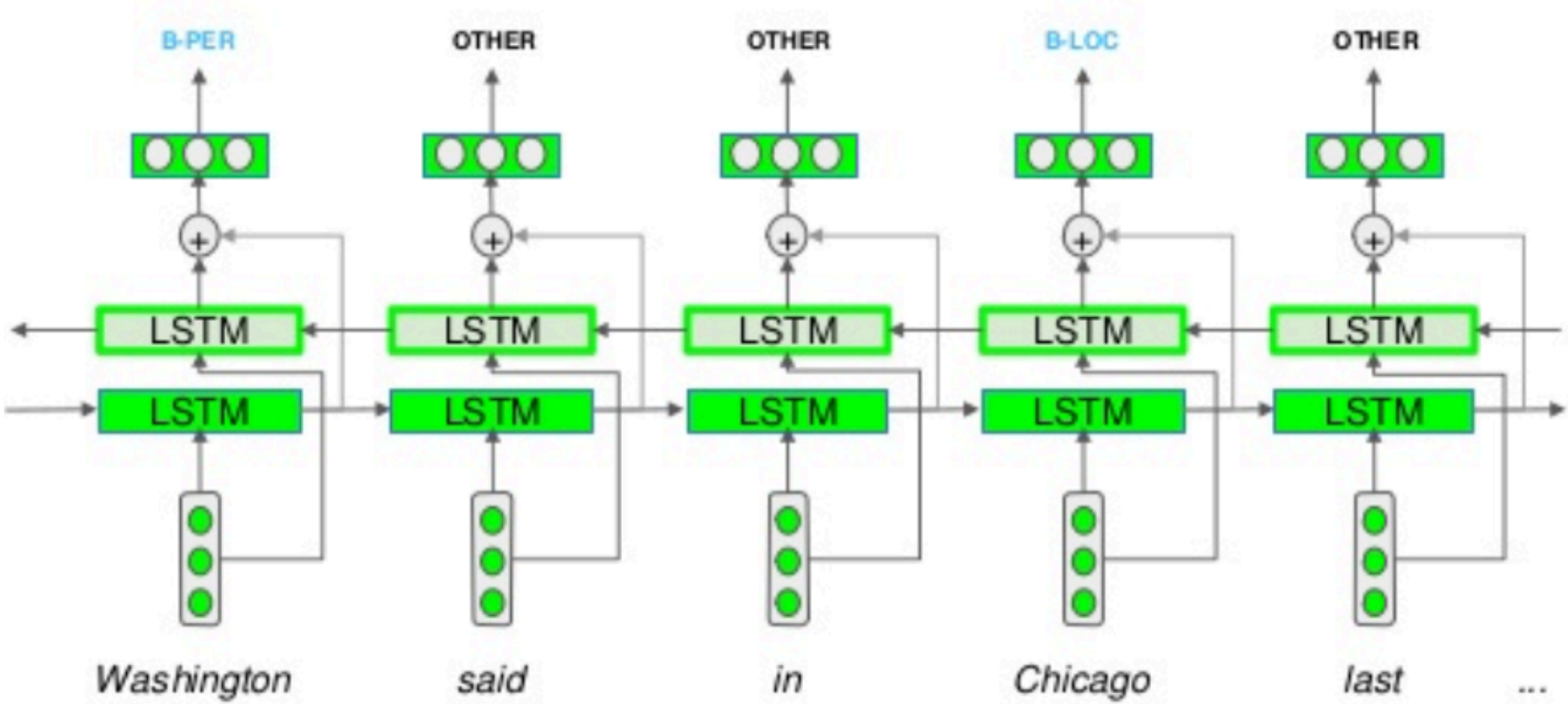


Recurrent neural network for NER



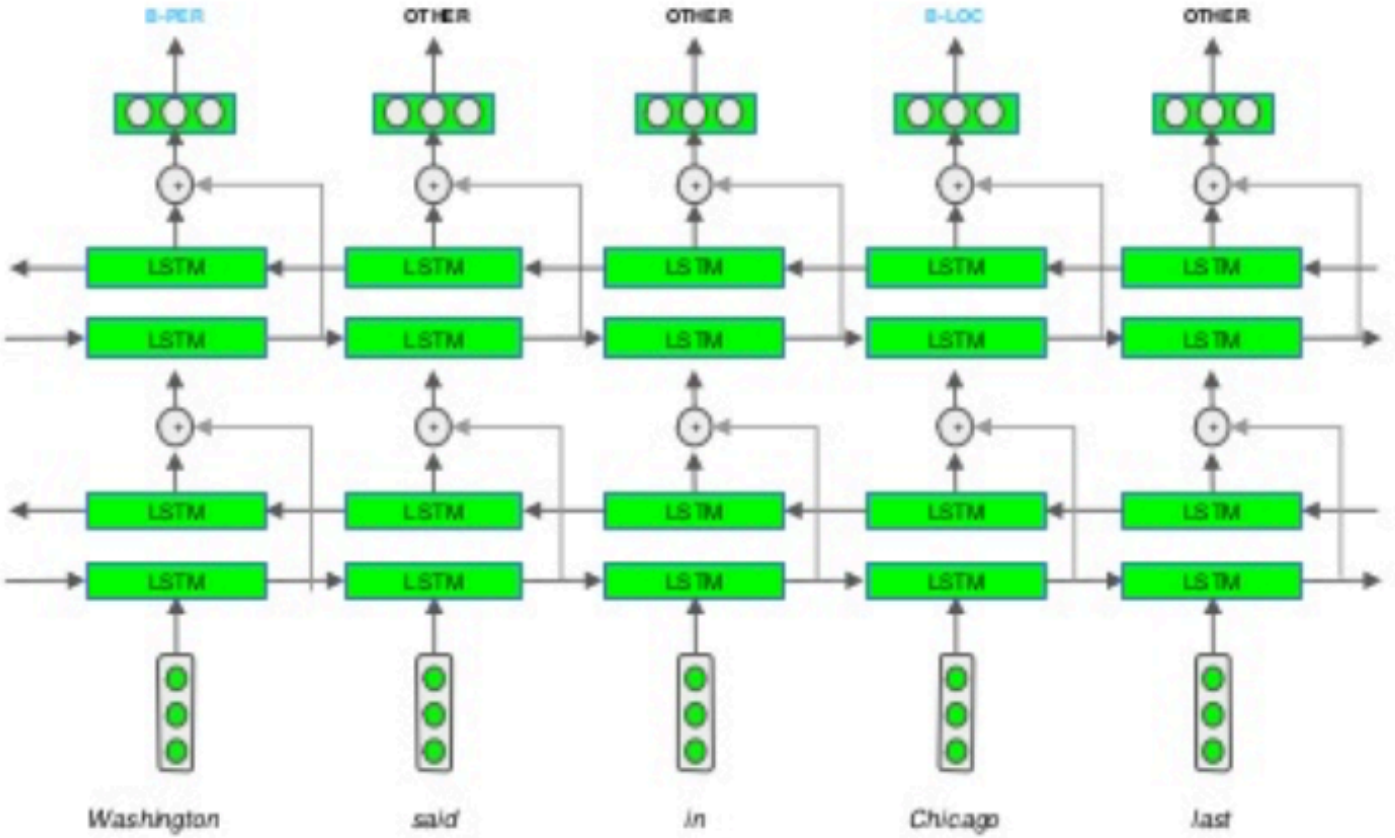
- Activation function for output: softmax
- Labels are OneHotEncoded

Bi directional recurrent neural network for NER



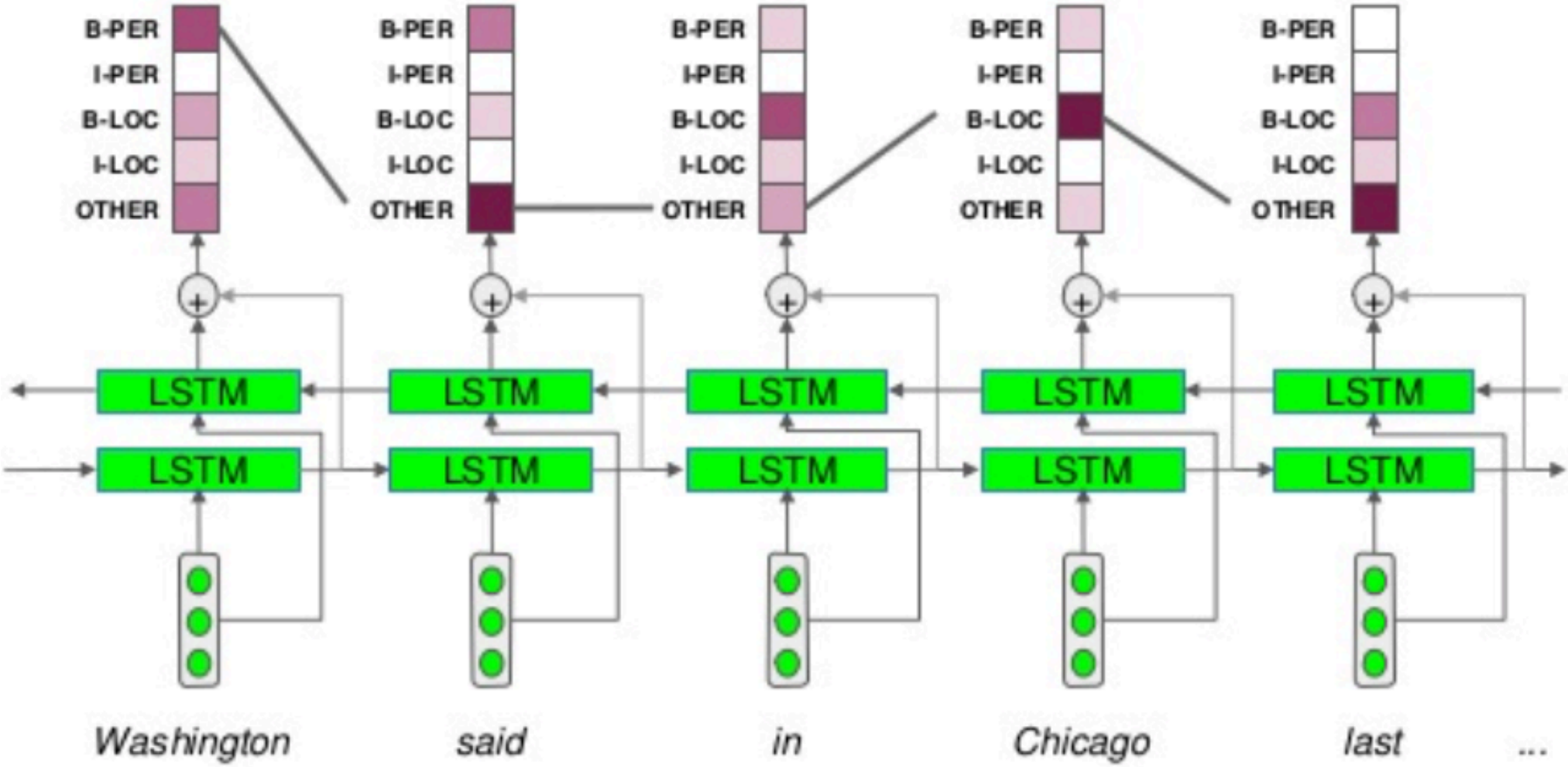
- Activation function for output: softmax
- Labels are OneHotEncoded

Stacked Bi-RNN



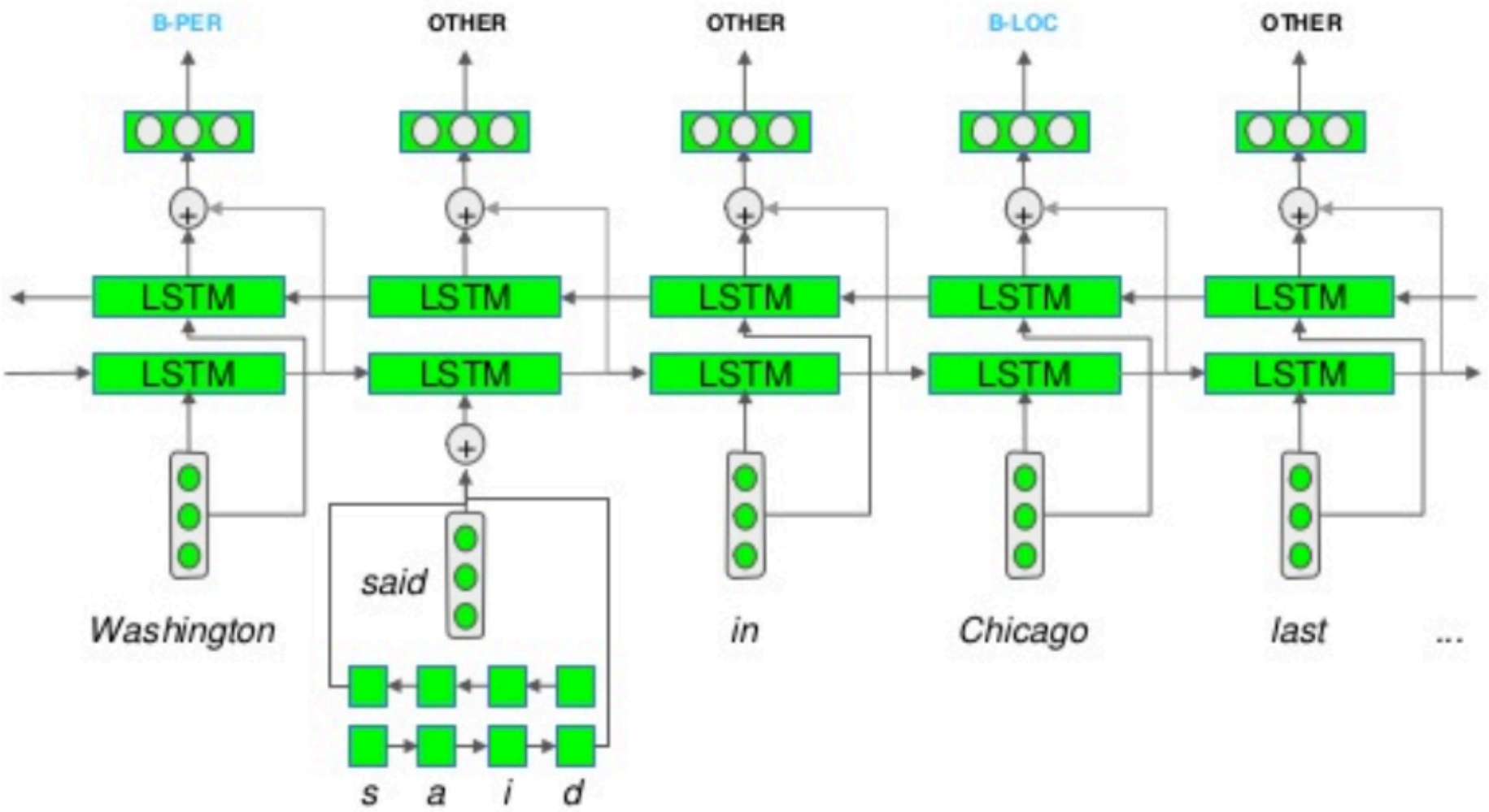
- Activation function for output: softmax
- Labels are OneHotEncoded

Bi-RNN + CRF



- CRF output activation function

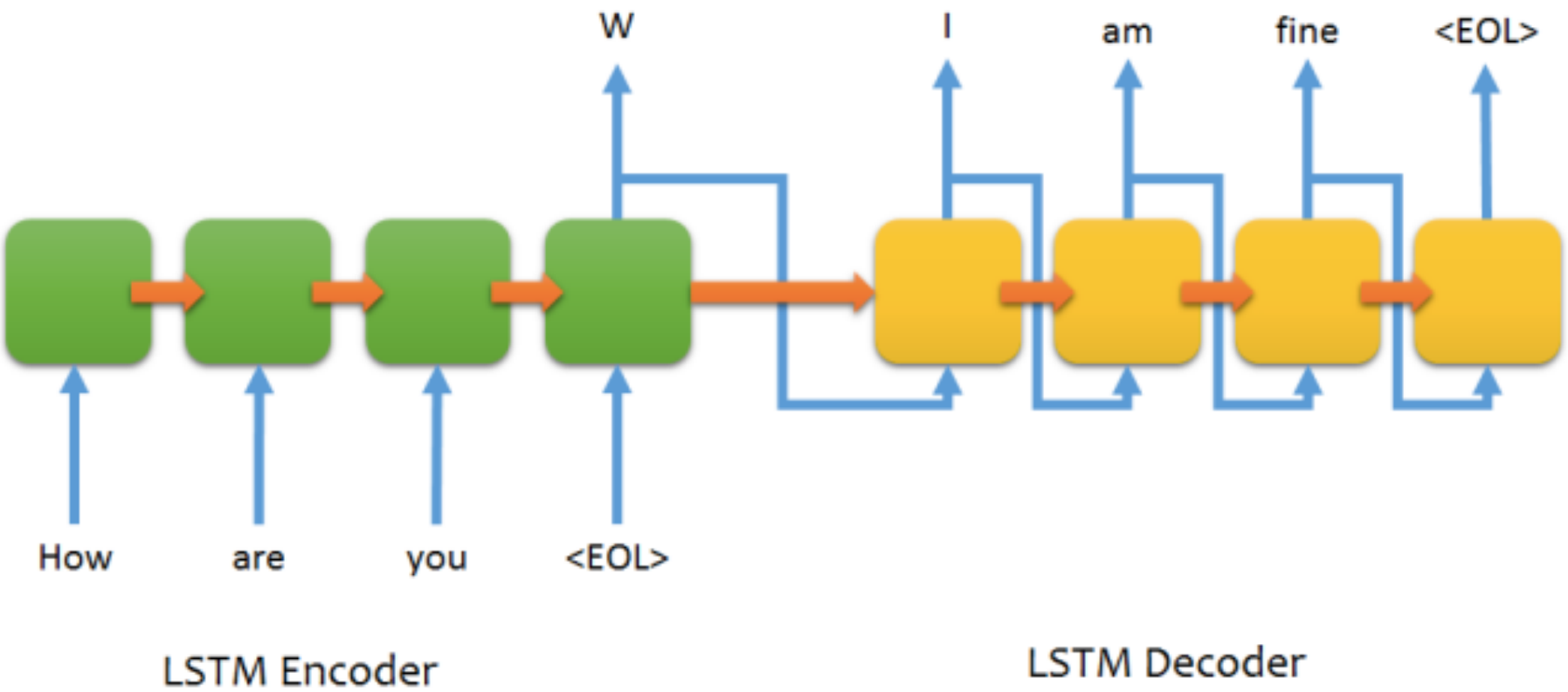
Multi-level encoding



Open source library for NER

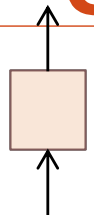
- ***Stanford's Named Entity Recognizer***
  - Based on an implementation of linear chain Conditional Random Field (CRF) sequence models
  - Only trained on instances of ***PERSON***, ***ORGANIZATION*** and ***LOCATION*** types
- SpaCy
  - A Python framework known for being fast and very easy to use.
  - It has an excellent statistical system that you can use to build customized NER extractors.

Some use of RNN
→ Sequence to sequence chat model



Some use of RNN
→ Input – Output Scenarios

Single - Single



Feed-forward Network

Single - Multiple

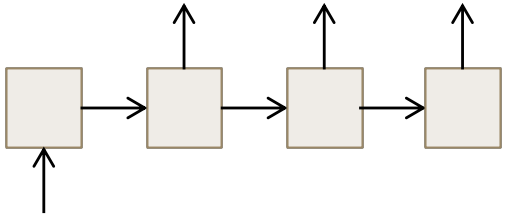
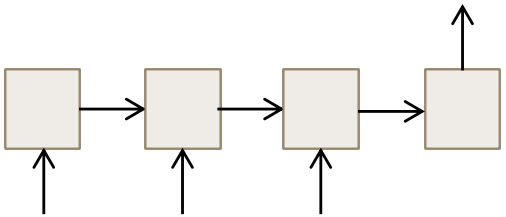


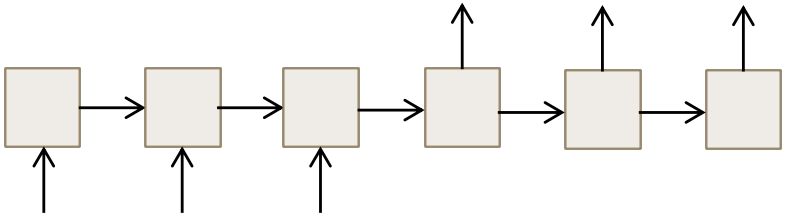
Image Captioning

Multiple - Single

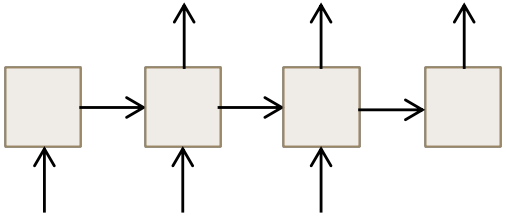


Sentiment analysis

Multiple – Multiple



Translation/Chat bot
seq2seq architecture



Named Entity Recognition

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