Intro ML
ML. 7.4. Neural networks for sequence processing

DataLab CSIC

Objectives and schedule

Introduce key concepts about neural networks for sequence processing: RNNs (LSTM), Transformers. Relate with time series models. Intro to natural language processing including LLMs

Goodfellow et al 10, Chollet and Allaire 6,

Gallego and Rios Insua, Arxiv 2303.18223, 2304.00612 in github

For intros

https://www.youtube.com/watch?v=UNmqTiOnRfg Intuitive

https://www.youtube.com/watch?v=6niqTuYFZLQ Intuitive but techie

Presentation by David Arroyo on *DL and natural language processing*

Lab 7.3

- LSTM for time series (autoregression)
- LSTM for dynamic regression
- Sentiment analysis for movie reviews with LSTM
- Text generation with LSTM

CPU and Google Collab (GPU) versions

- Drago, Artemisa
- Your institute, your group HPC installation

RNNs. Motivation

Motivation

- Fully connected NNs can approximate any function….
- But training can be super slow and may require lots of data
- In some domains, lots gained through specific archictectures

- In natural language processing, recurrent neural nets (RNNs)
- More generally in sequence processing, RNNs (and successors)

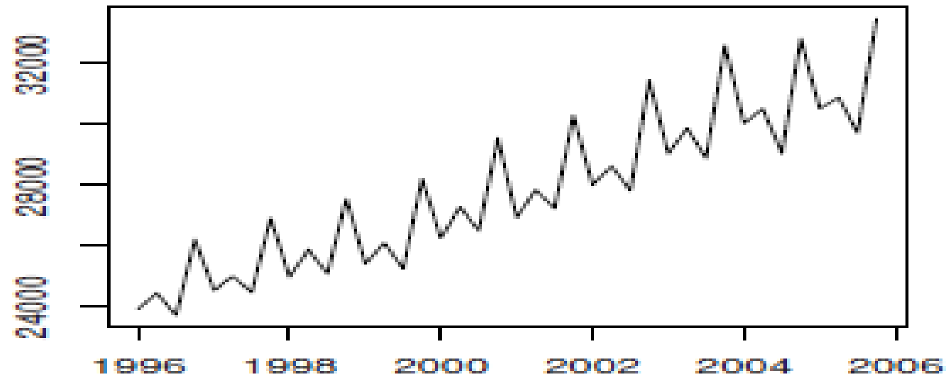
Time series analysis

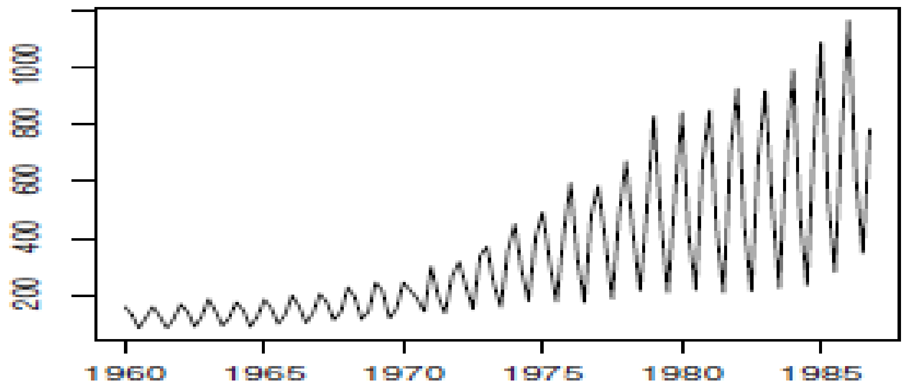
- Traditional models in time domain: **AR**IMA, exponential smoothing
- Models in frequency domain: Spectral analysis
- State space models: Kalman filter, Hidden Markov models, dynamic linear models (plus non linear and non gaussian extensions)

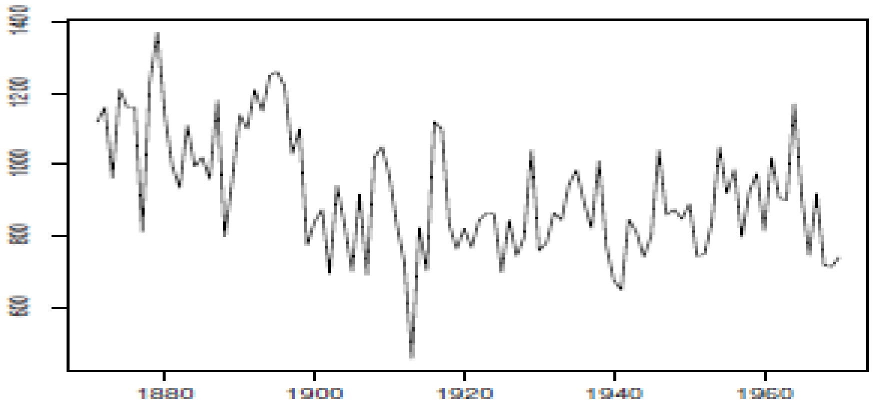
$$
\begin{array}{ccccccccccccc}
\theta_0 & \longrightarrow & \theta_1 & \longrightarrow & \theta_2 & \longrightarrow & \cdots & \longrightarrow & \theta_{t-1} & \longrightarrow & \theta_t & \longrightarrow & \theta_{t+1} & \longrightarrow & \cdots \\
 & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow & & \downarrow & & \\
 & & Y_1 & & Y_2 & & & & Y_{t-1} & & Y_t & & Y_{t+1} & &
\end{array}
$$

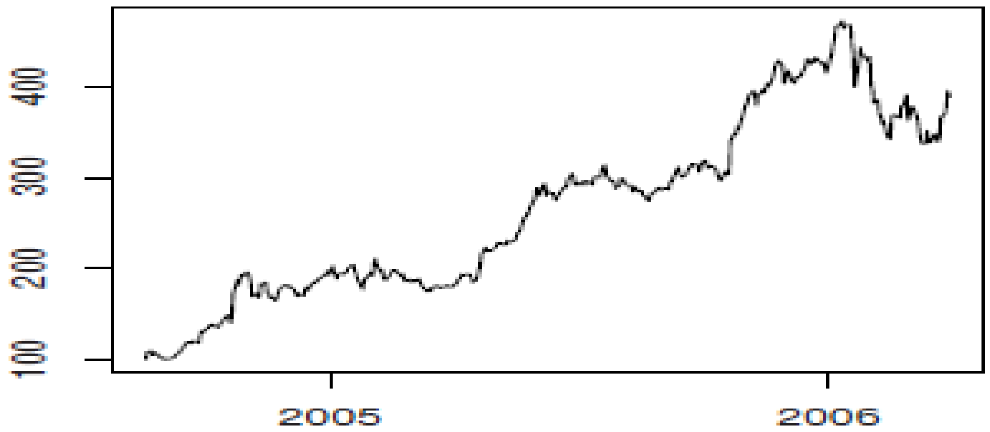
Check Prado and West (2010) for a comprehensive review

Some typical features in time series









Problems of interest

Objective $\pi(\theta_s|y_{1:t})$ and, specially, $\pi(y_s|y_{1:t})$

Filtering $s = t$

Prediction $s > t$

Smoothing $s < t$

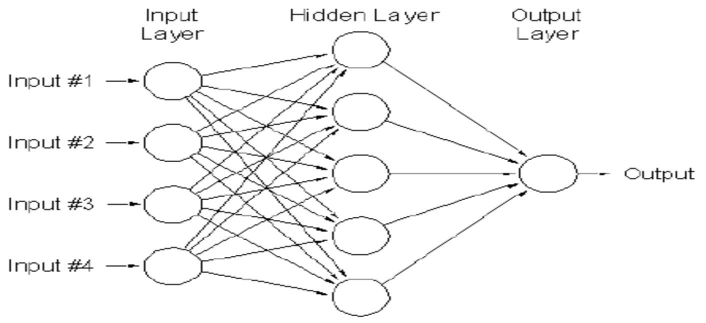
NNs for time series. Nonlinear autoregressions…

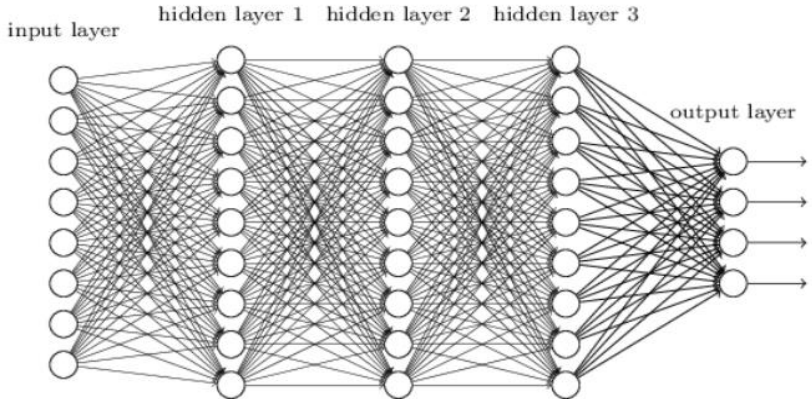
Shallow NN

$$y_j = \sum_{i=1}^{m} \beta_i \psi(x_k \omega_i) + \varepsilon_j$$

$$\min_{\beta, w} \sum_{k=1}^{n} \left(y_k - \sum_{i=1}^{m} \beta_i \psi(x_k \omega_i) \right)^2$$

- Input. Some entries, prior time series observations
- Output, value of time series to be forecast (one step ahead, two steps ahead,….)



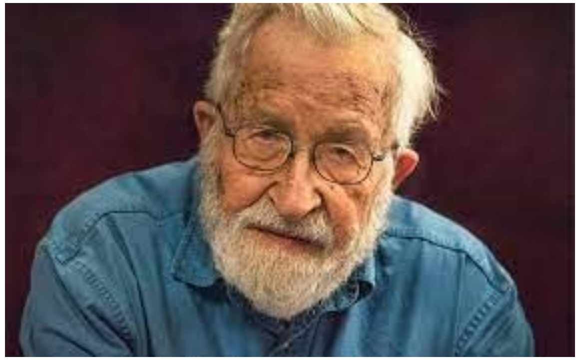


Fully connected… theoretically OK, but lot to be gained from special structure

RNNs. Key ideas

- Specialized for sequential data (numbers, words, letters,….)
- Can process sequences much longer than those achievable with fully connected NNs
- More amenable to parallelisation (transformers)
- Each neuron has an 'internal memory' (hidden) to store info about previous entries
- Trained with variants of standard algos: backpropagation through time
- Created in 80's and late 90's, yet their recent successes (as with CNNs) make them ultra-fashionable.
- Latest wave: Transformers (attention is all you need), LLM, Chat-GPT
- Some applications
  - Speech recognition
  - Language modeling and text generation
  - Automatic translation
  - Image description generation
  - + the usual suspects: prices, sales,…

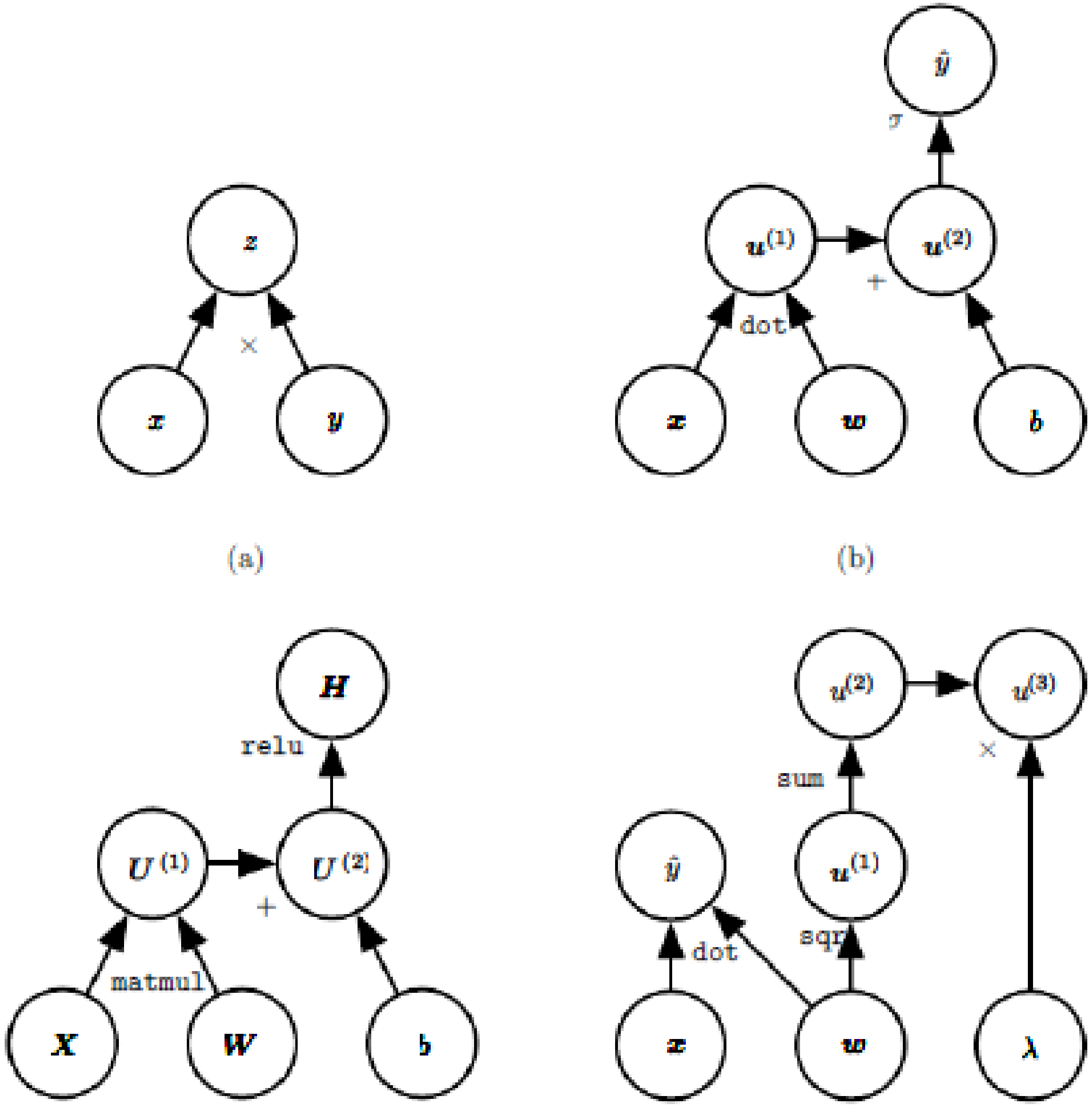
A paradigm change in NLP



- From pre-deep learning
    - Language as a set of elements and rules to be combined
    - Context independent grammars (Chomsky)
    - Closer to artificial languages (programming) than to natural ones
- To statistically based
    - Language as probabilities of word sequences
    - Computing frequencies of words, n-grams,…
    - Closer to natural language
    - Combined with deep NNs, state of the art
    - Almost a commodity (like vision)

Core concepts
Recurrence and computational graphs

Computational graphs

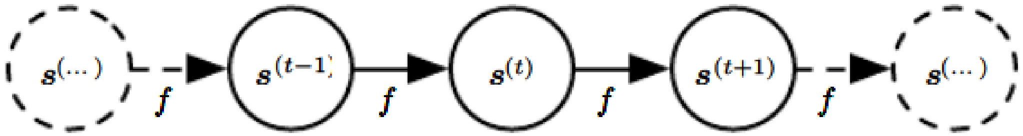


Recurrence and unfolded computational graphs

- Dynamical system. Recurrence unfolded

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

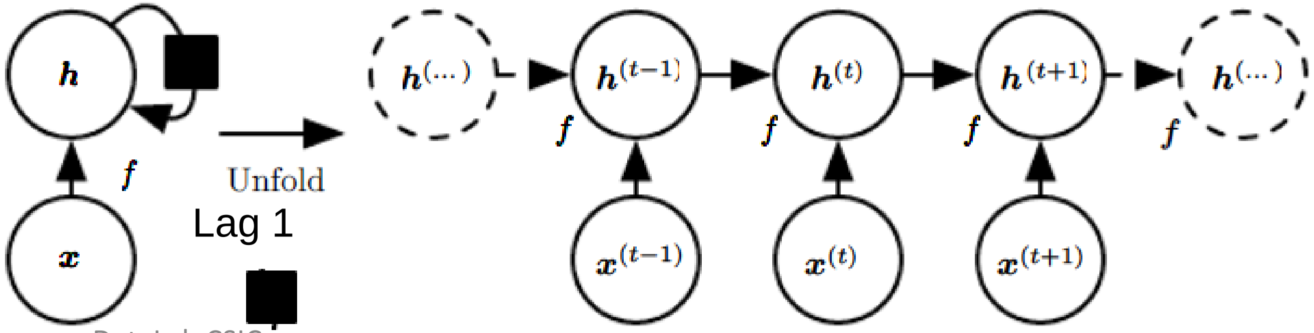
$$\begin{aligned}\boldsymbol{s}^{(3)} &= f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) \\ &= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta})\end{aligned}$$



- Every recurrent function as recurrent NN h (hidden) state

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})\end{aligned}$$



Recurrence in a simple model. State space model

- A p-variate time series (θ_t) and an m-variate time series Y_t
  - (θ_t) is a Markov chain
  - Given (θ_t) , Y_t independent of other observations and depends only on θ_t

$$
\begin{array}{ccccccccccccccc}
\theta_0 & \longrightarrow & \theta_1 & \longrightarrow & \theta_2 & \longrightarrow & \cdots & \longrightarrow & \theta_{t-1} & \longrightarrow & \theta_t & \longrightarrow & \theta_{t+1} & \longrightarrow & \cdots \\
 & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow & & \downarrow & & \\
 & & Y_1 & & Y_2 & & & & Y_{t-1} & & Y_t & & Y_{t+1} & &
\end{array}
$$

$$
\pi(\theta_{0:t}, y_{1:t}) = \pi(\theta_0) \cdot \prod_{j=1}^{t} \pi(\theta_j | \theta_{j-1}) \pi(y_j | \theta_j)
$$

Core concepts:
Working with text data

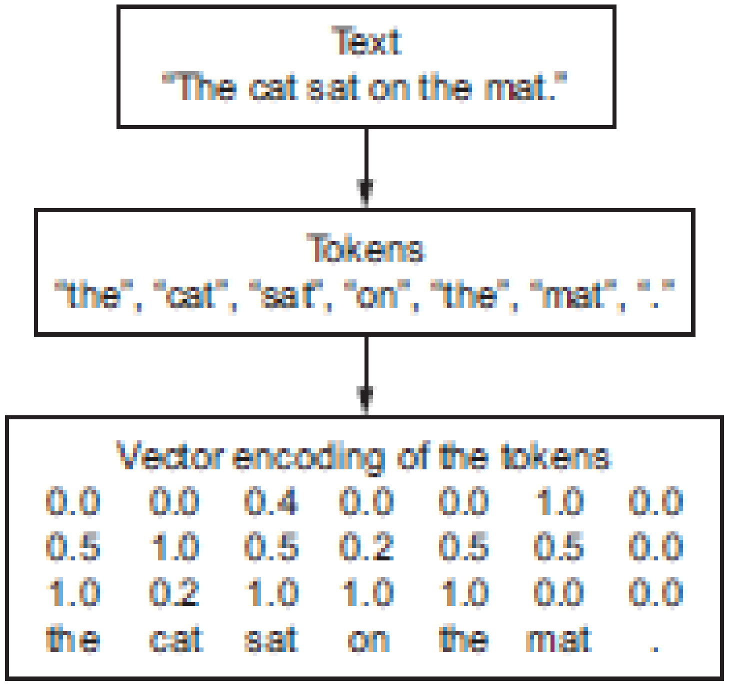
Goodfellow et al 12, Chollet and Allaire 6

For intuitive intro
https://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

Text

Sequence of characters

Sequence of words



Tokenize and vectorize

- Segment text into words, transform each word into a vector
- Segment text into characters, transform each character into a vector
- Extracts n-grams of words or characters, transform each n-gram into a vector

One hot encoding

Unique integer to each word

Word , vector of zeroes, except a 1 for the word

Sparse and high dimensional

Word embeddings

Dense word vectors

Low-dimensional, floating point vectors

Learned from data

- Jointly with the main task. Embedding layer
- Precomputed or pretrained embeddings. Word2vec, Glove,…

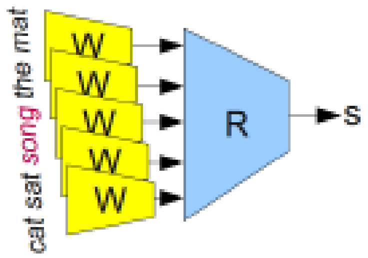
Word embedding. Example

Parametrised function $W : \text{words} \to \mathbb{R}^n$

Train to predict if a 5 ngram is valid

$$R(W(\text{``cat''}),\ W(\text{``sat''}),\ W(\text{``on''}),\ W(\text{``the''}),\ W(\text{``mat''})) = 1$$

$$R(W(\text{``cat''}),\ W(\text{``sat''}),\ W(\text{``song''}),\ W(\text{``the''}),\ W(\text{``mat''})) = 0$$



BERT trains by 300 bn tokens to predict the next word

RNNs

RNNs

- Emerge to process sequences, specially with different length inputs
- Add feedback connections

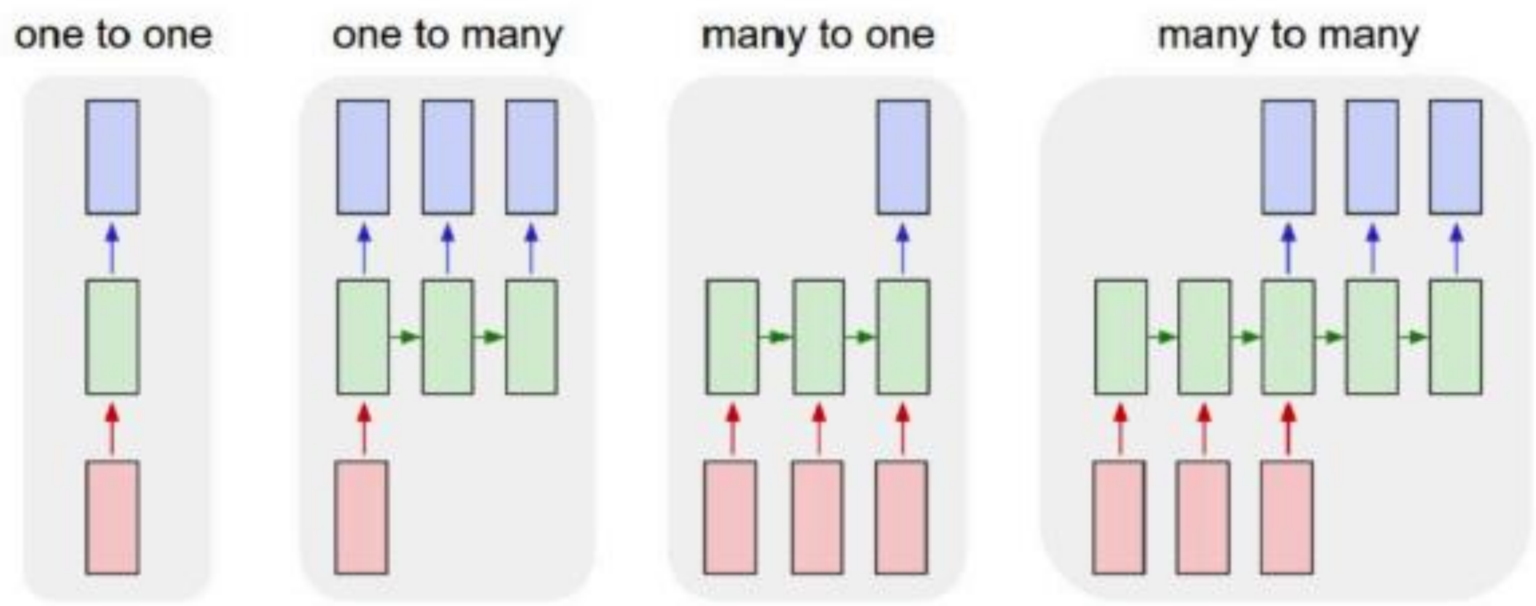


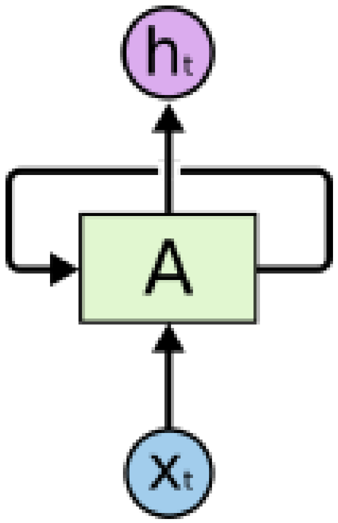
Image-Object

Image-Caption

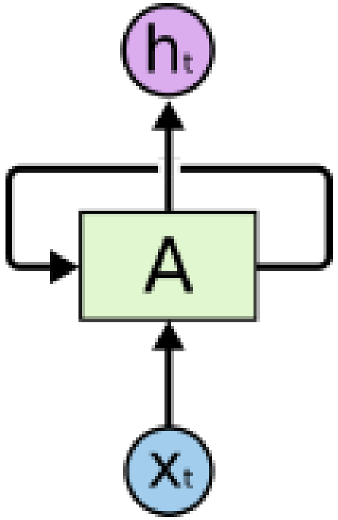
Text-Sentiment

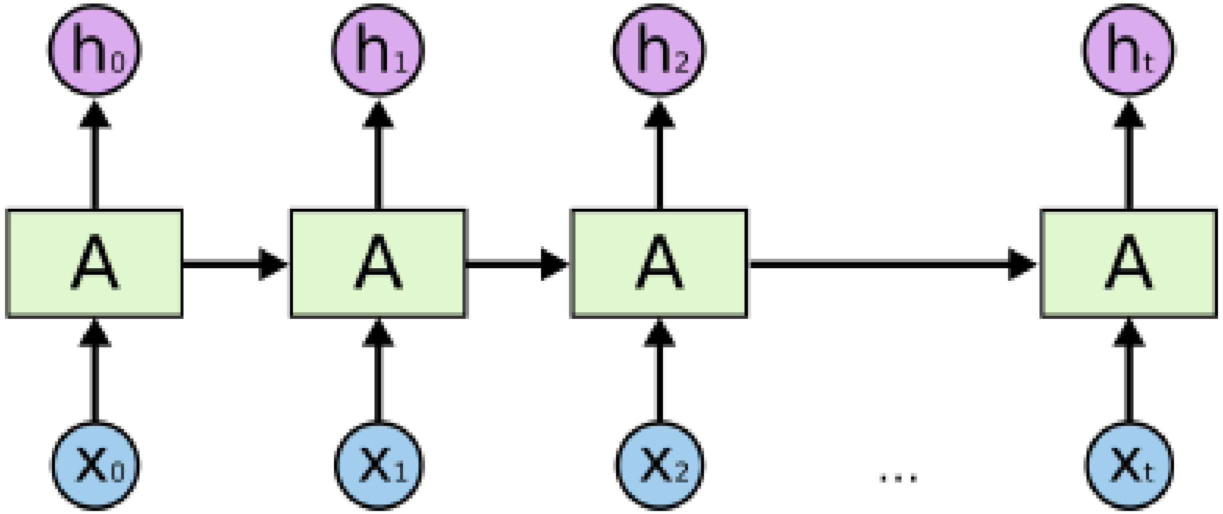
Machine translation

RNNs

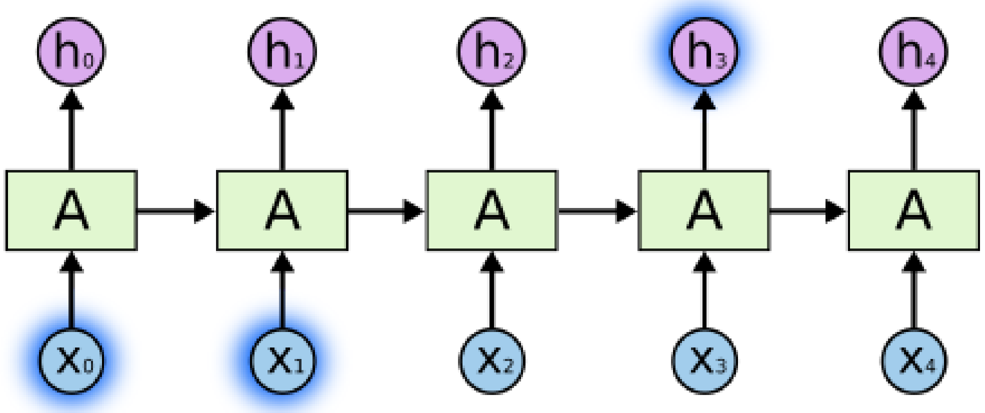


RNNs



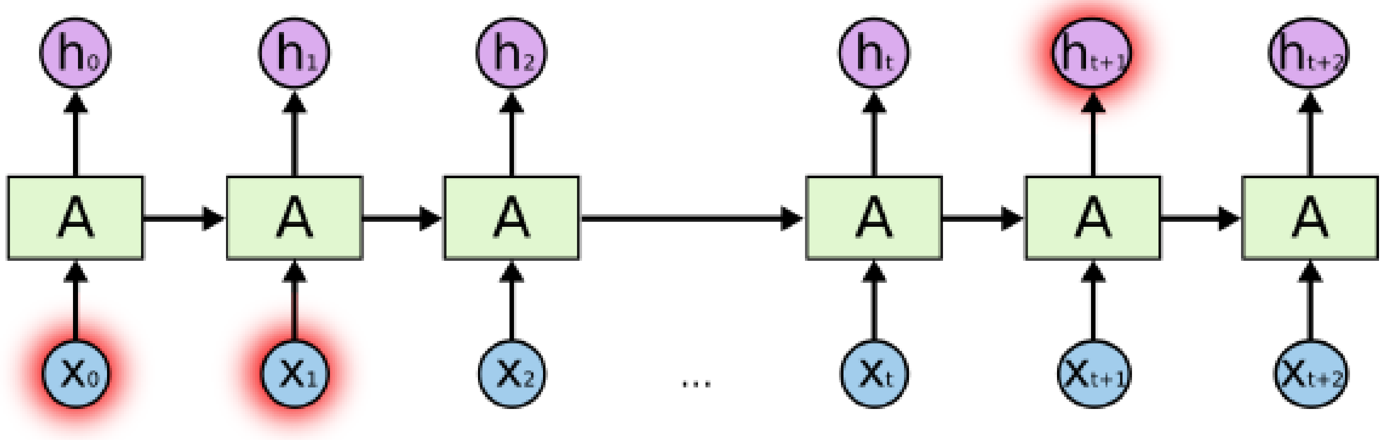


RNNs: Short-term vs Long-term dependencies



The clouds are in the …..

RNNs: Short-term vs Long-term dependencies



I grew up in France…. I speak fluent ….

RNNs. Elman's model

Network updates internal state *h* updated at each step

$$h_t = f_W(h_{t-1}, x_t)$$

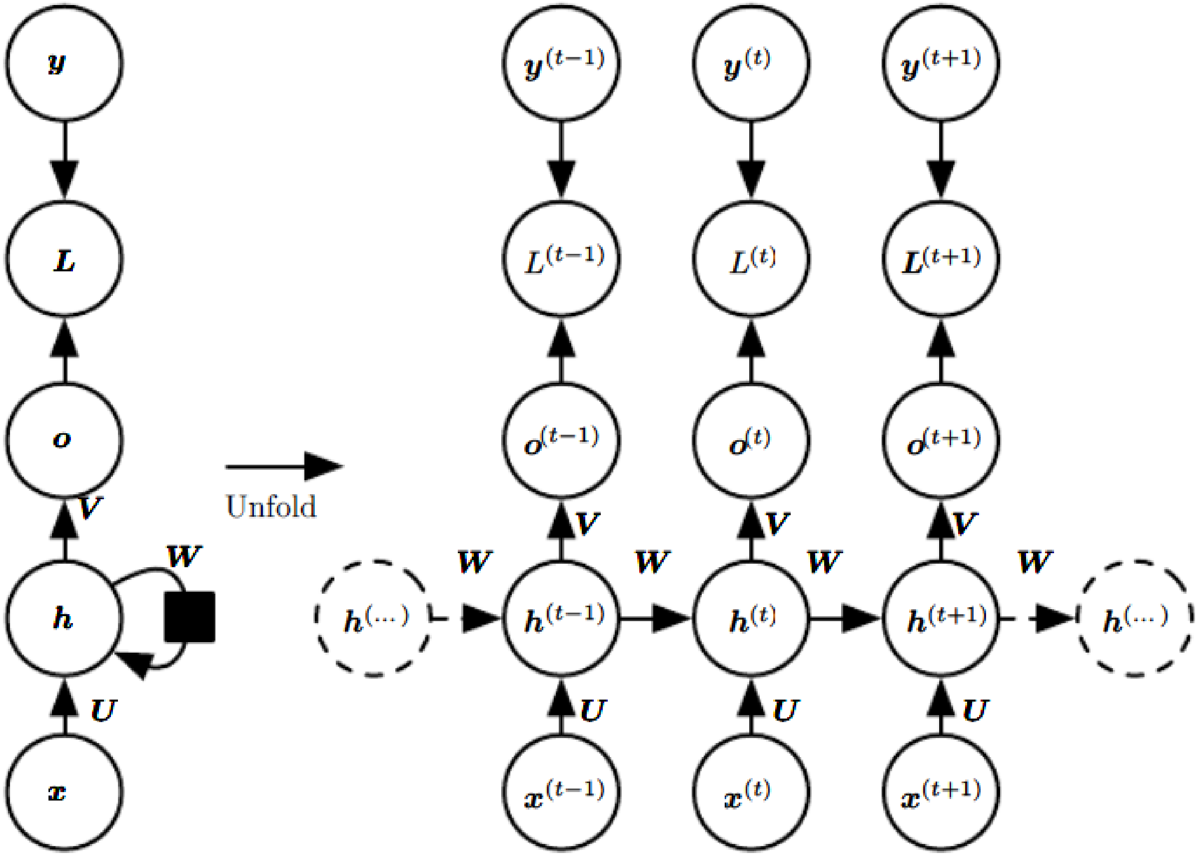
e.g.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

Weights reused at each time:

- learn patterns independently of position
- reduction of number of parameters

RNN. One output per step, recurrence between hidden nodes



Parameters U, W, V

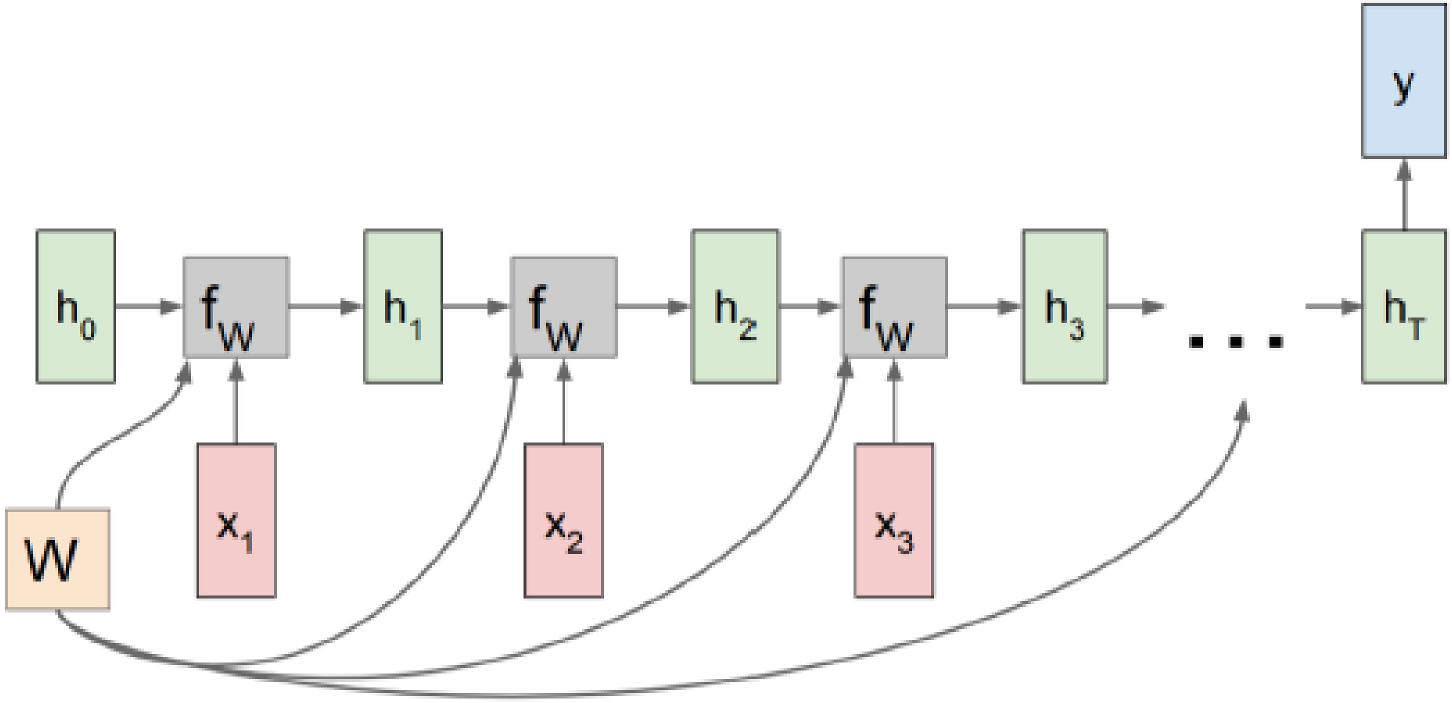
For example,

$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}$$

$$\begin{aligned}
& L\left(\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(\tau)}\}\right) \\
&= \sum_t L^{(t)} \\
&= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(t)}\}\right)
\end{aligned}$$

RNN: many to one example

Assigning sentiment (-,+) to a tweet



Training: Backprop through time

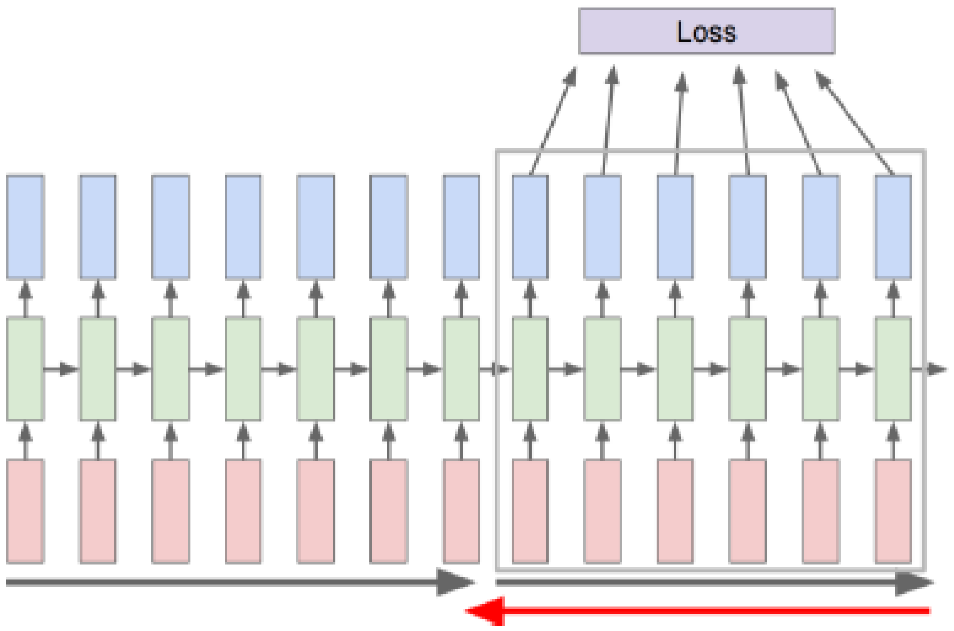
Unfolding the (computational) graph

Applying backprop

Limiting steps back for stability:

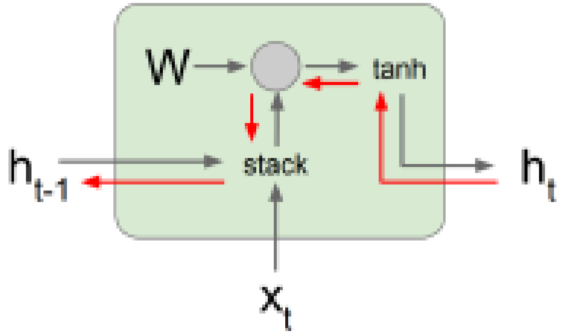
Truncated backprop

SGD or Adam or …



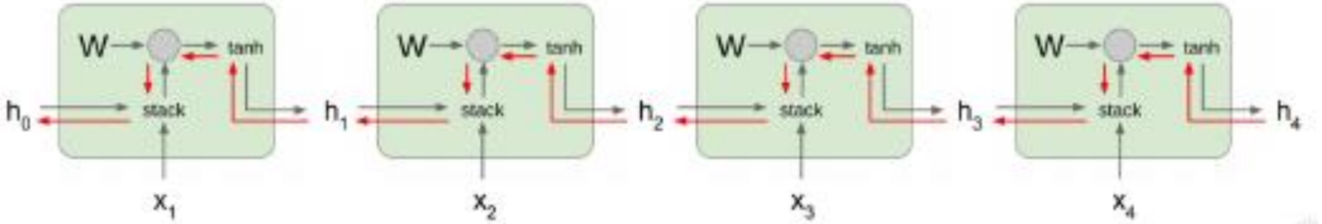
Problem with Elman's model....

Backprop one step back



$$
\begin{aligned}
h_t &= \tanh(W_{hh} h_{t-1} + W_{xh} x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

Over time



Repeated multiplications by W.

- If biggest eigenvalue>1, gradient explosion (gradient clipping)
- If biggest eigenvalue<1, gradient vanishing (LSTM, GRU)

$$\boldsymbol{W} = \boldsymbol{V} \mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{V}^{-1}$$

$$\boldsymbol{W}^t = \left(\boldsymbol{V} \mathrm{diag}(\boldsymbol{\lambda}) \boldsymbol{V}^{-1}\right)^t = \boldsymbol{V} \mathrm{diag}(\boldsymbol{\lambda})^t \boldsymbol{V}^{-1}$$

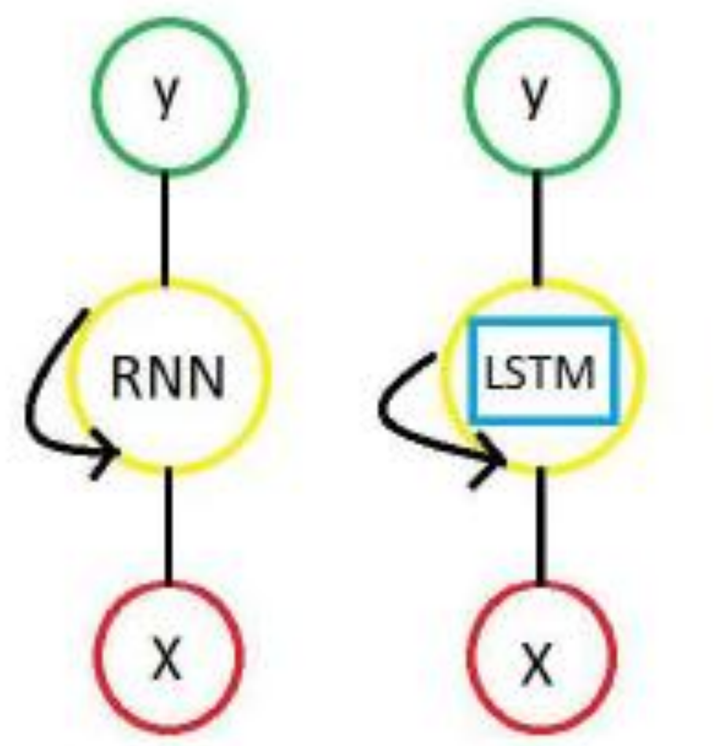
Long Short-term memory (LSTM) NNs

Hochreichter, Schmidhuber

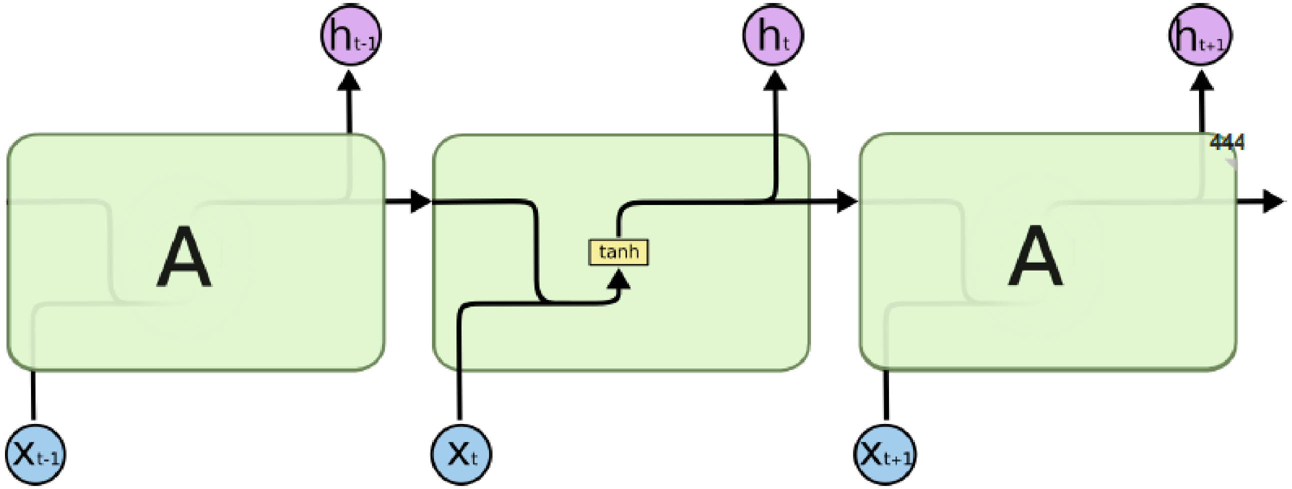
LSTM

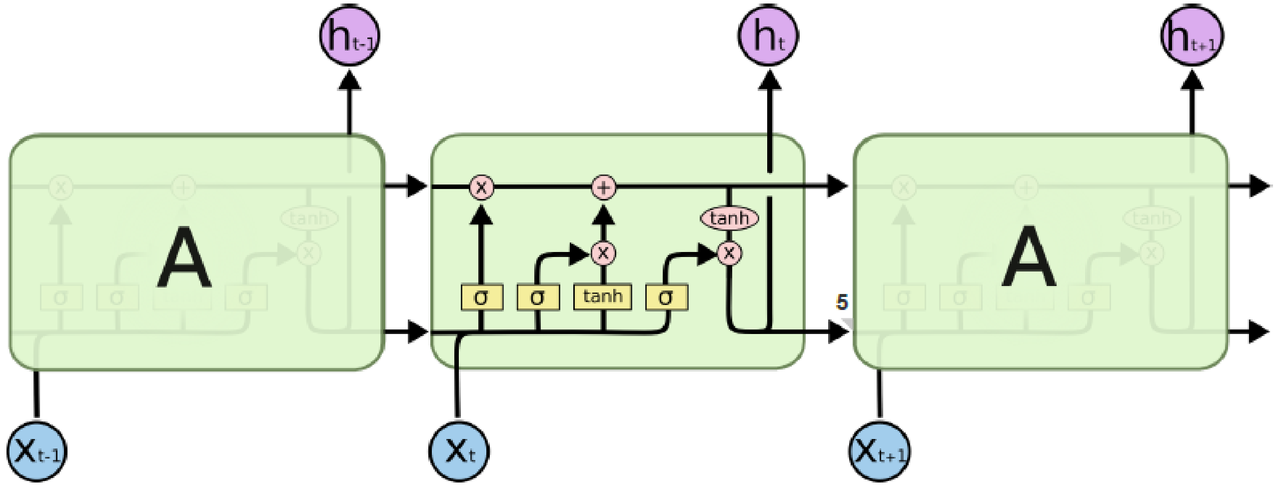
- Introduced by Hochreiter y Schmidhuber in 1997 but only used (a lot!!!) in last decade for NLP
- Hidden cells substituted by LSTM cells mitigating

vanishing and explosion



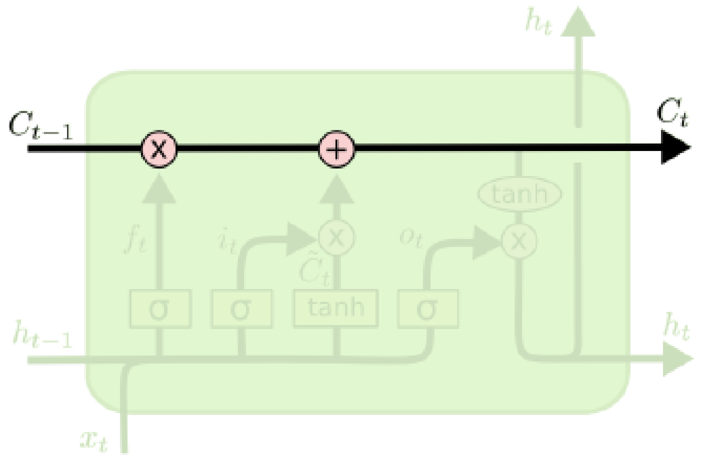
From RNNs to LSTMs



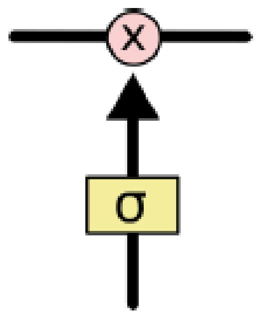


Basic ingredients

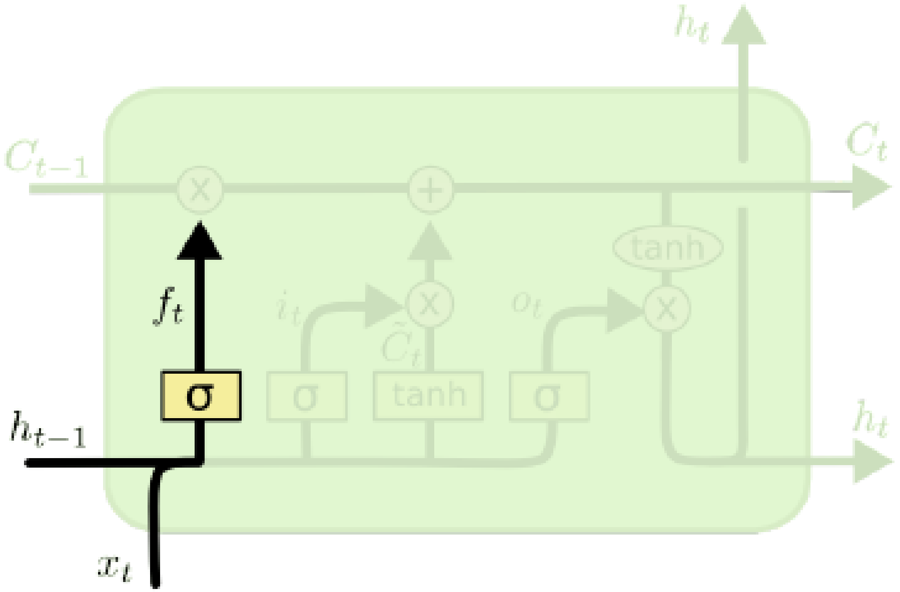
Cell state



Gate

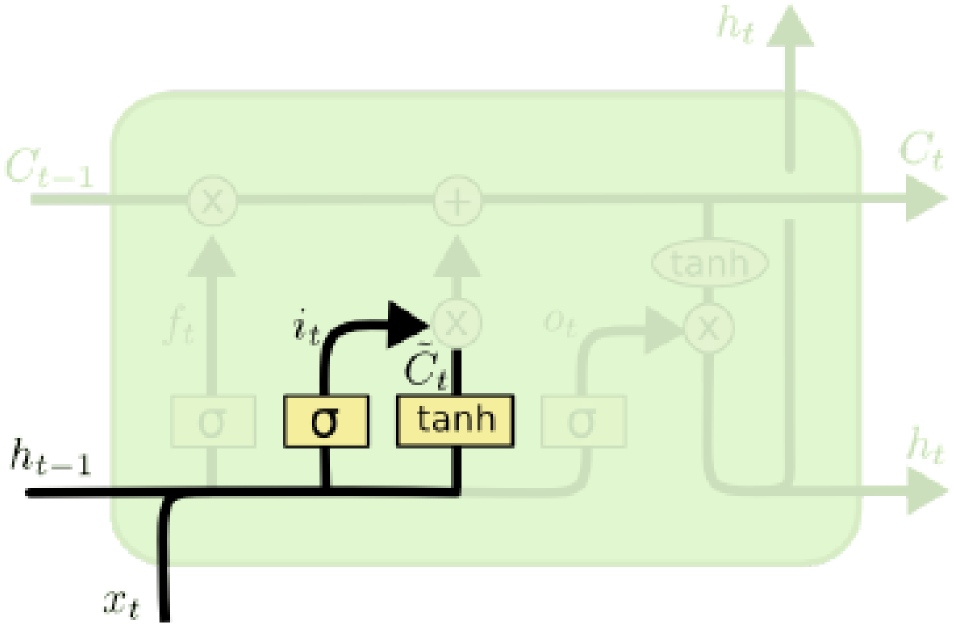


I) Info to be forgotten. Forget gate



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

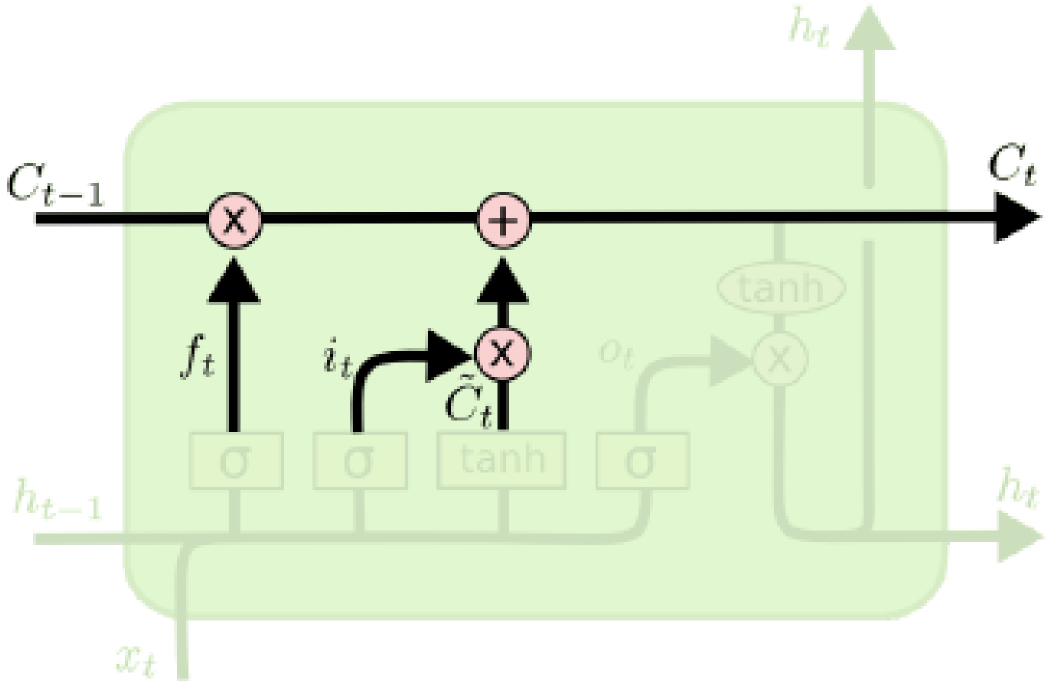
ii) Info to be stored in cell state



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

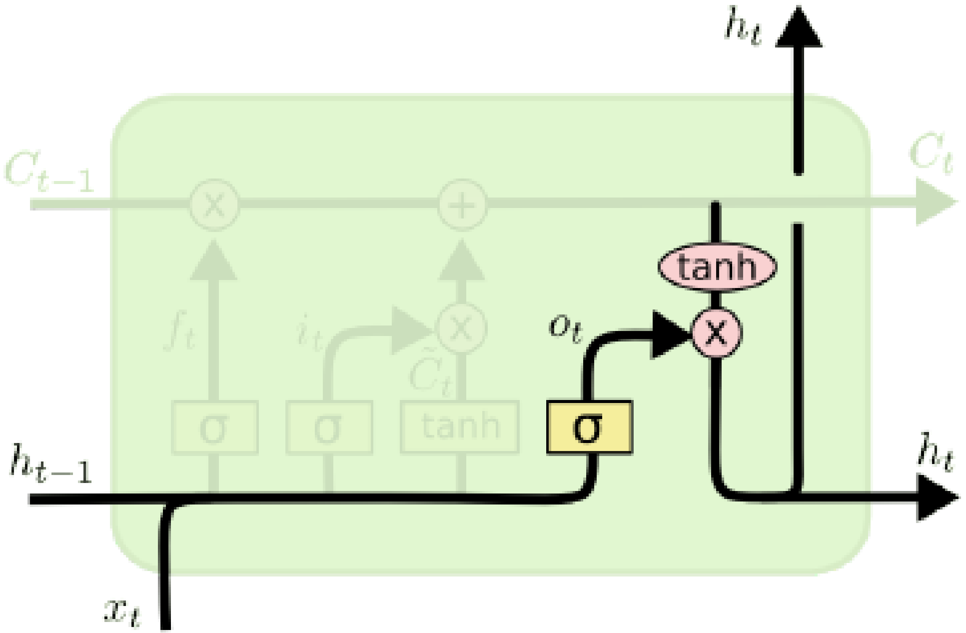
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

iii) Update cell state



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

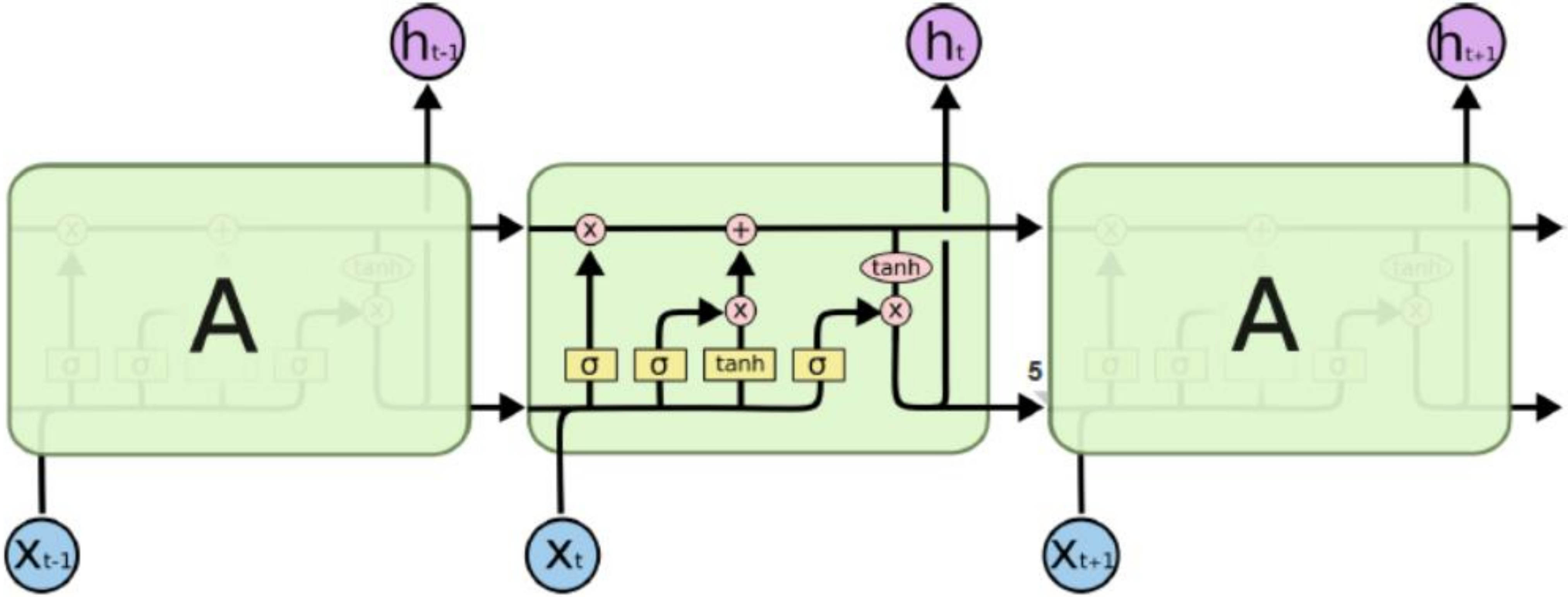
iv) Decide output



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

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

LSTM Global scheme

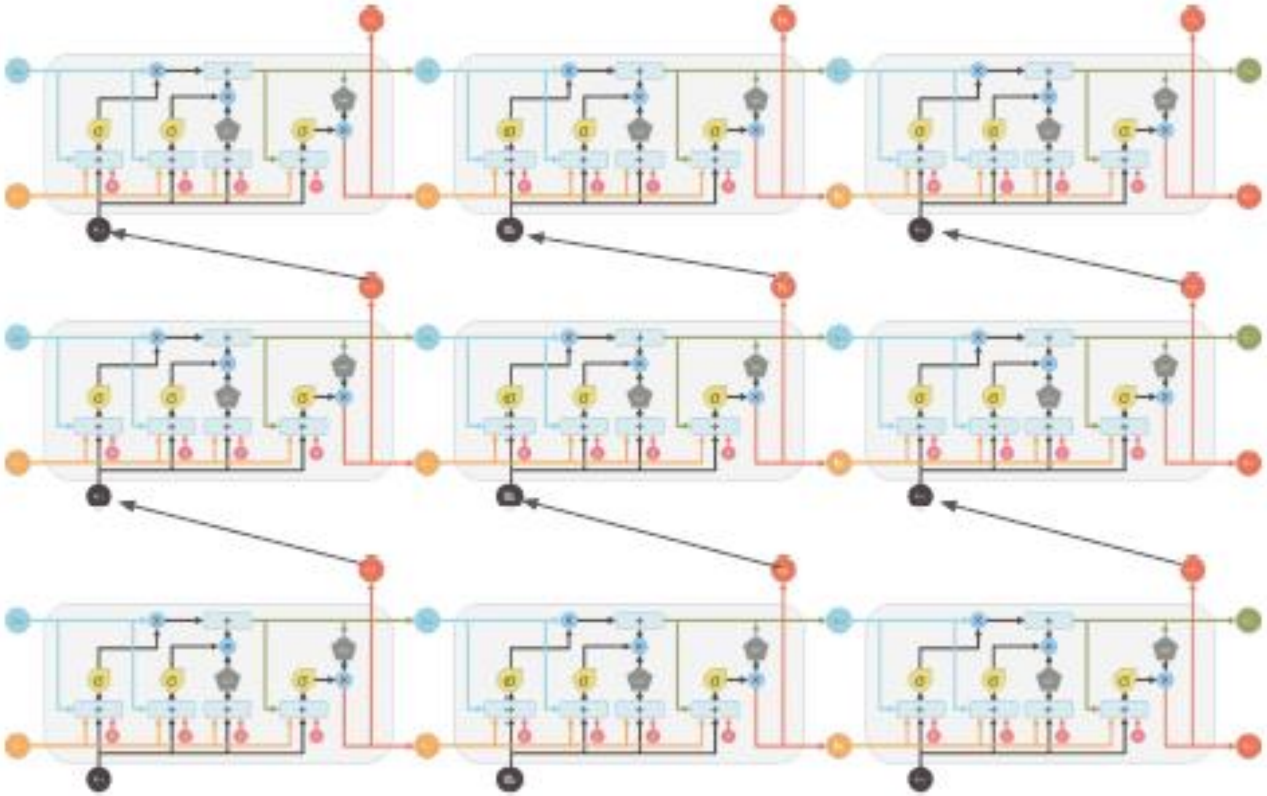


Deep LSTMs

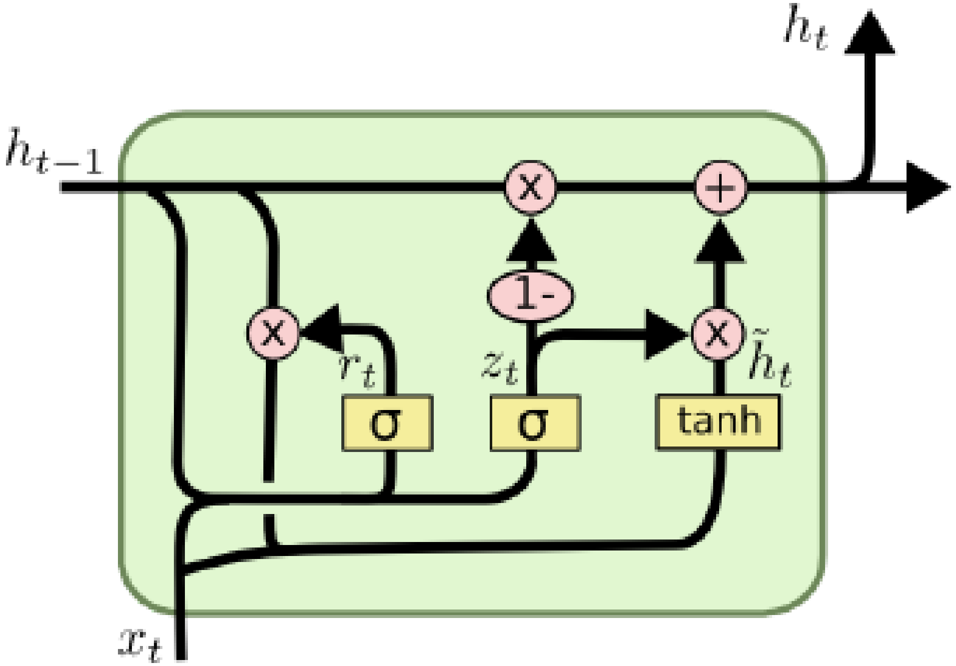
- Deep LSTMs can be created by stacking multiple LSTM layers vertically, with the output sequence of one layer forming the input sequence of the next (in addition to recurrent connections within the same layer)

- Increases the number of parameters - but given sufficient data, performs significantly better than single-layer LSTMs (Graves et al. 2013)

- Dropout usually applied only to non-recurrent edges, including between layers



Gate recurrent unit. GRU



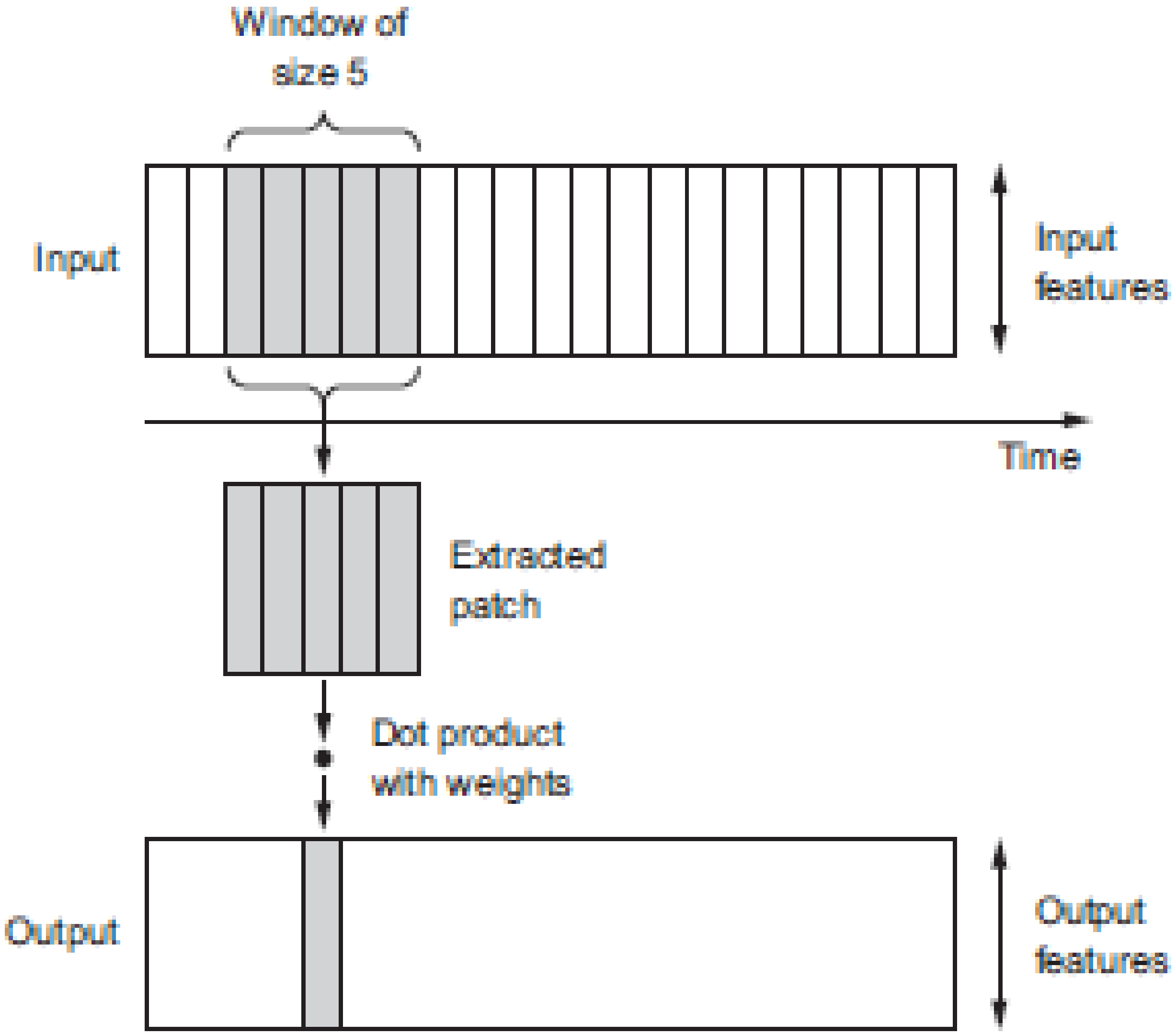
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

1-D convnets



+

maxpooling
or
average pooling

Transformers and LLMs

Transformers and Large Language Models

- https://www.youtube.com/watch?v=SZorAJ4I-sA Basic intro
- https://www.youtube.com/watch?v=_UVfwBqcnbM Detailed non-tech intro
- https://www.youtube.com/watch?v=S27pHKBEp30 Contextual intro

Attention is all you need

https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

http://nlp.seas.harvard.edu/2018/04/03/attention.html

Formal algos for transformers

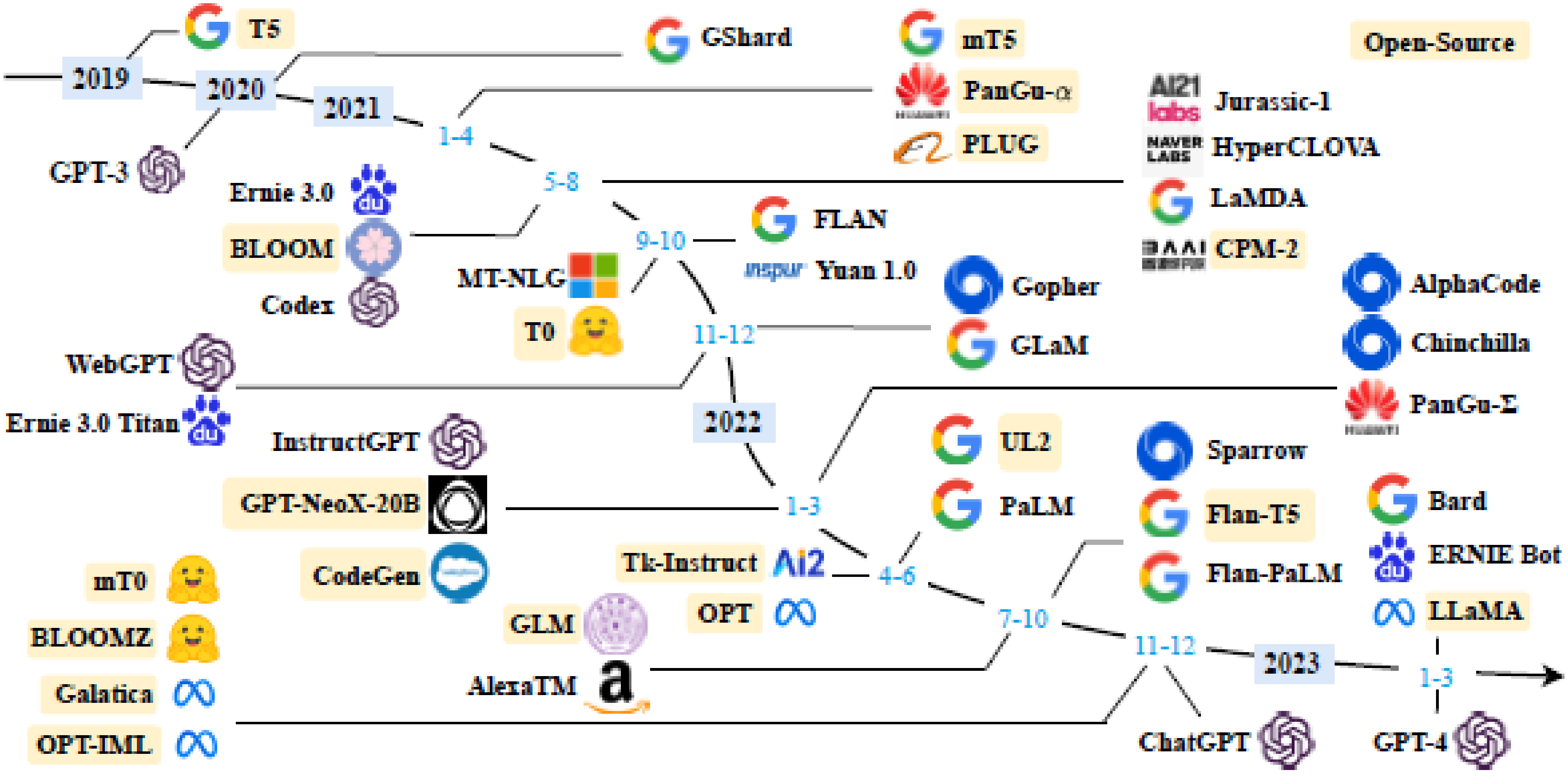
https://arxiv.org/pdf/2207.09238.pdf

A survey of large language models

https://arxiv.org/pdf/2303.18223.pdf

Eight things to know about large language models

Arxiv:2304.00612


T5
2019
2020
2021
GPT-3
GShard
1-4
mT5
PanGu-α
PLUG
Open-Source
Jurassic-1
HyperCLOVA
5-8
Ernie 3.0
BLOOM
Codex
LaMDA
CPM-2
9-10
FLAN
Yuan 1.0
MT-NLG
T0
Gopher
GLaM
AlphaCode
Chinchilla
11-12
WebGPT
Ernie 3.0 Titan
PanGu-Σ
2022
InstructGPT
UL2
Sparrow
GPT-NeoX-20B
1-3
PaLM
Flan-T5
Bard
CodeGen
Tk-Instruct
4-6
Flan-PaLM
ERNIE Bot
mT0
OPT
LLaMA
GLM
7-10
BLOOMZ
11-12
2023
1-3
AlexaTM
Galatica
OPT-IML
ChatGPT
GPT-4

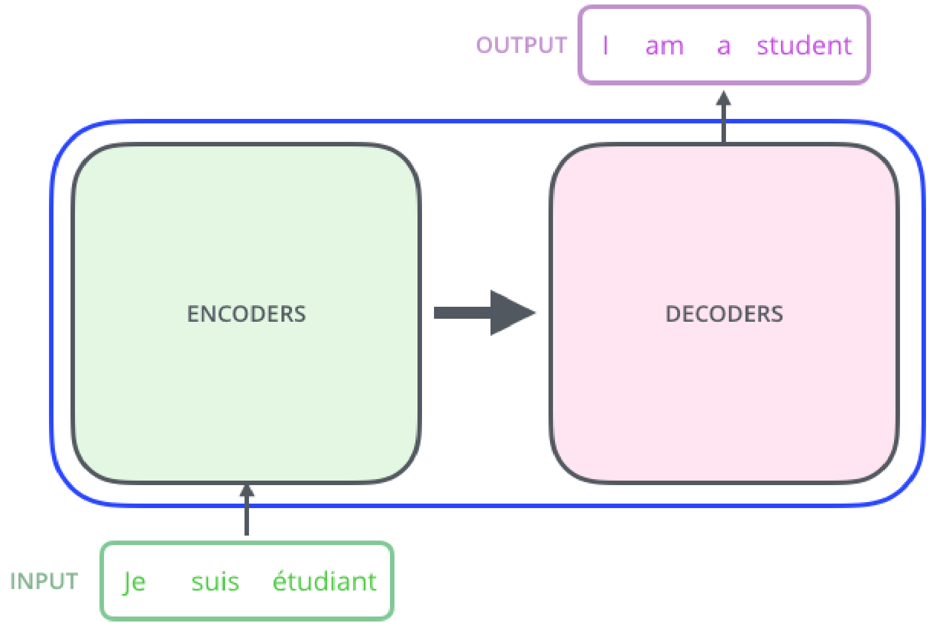
Transformers

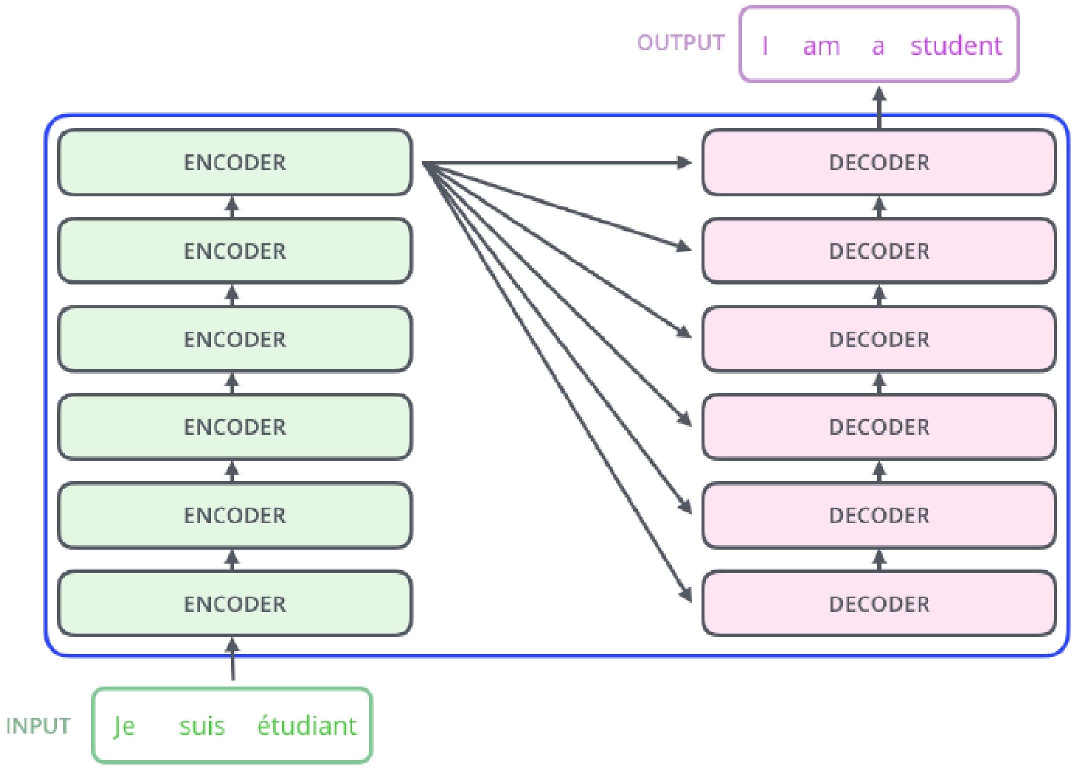
Seq2Seq with attention : https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Transformer : https://jalammar.github.io/illustrated-transformer/

BERT (and transfer learning) : https://jalammar.github.io/illustrated-bert/

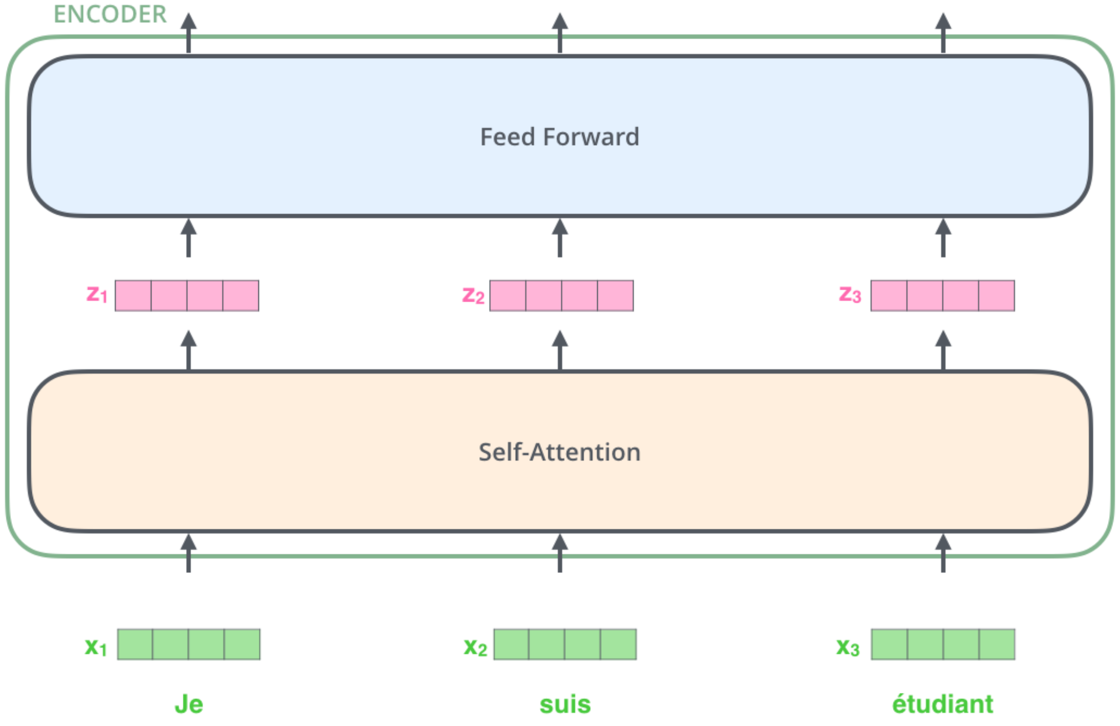
Transformer

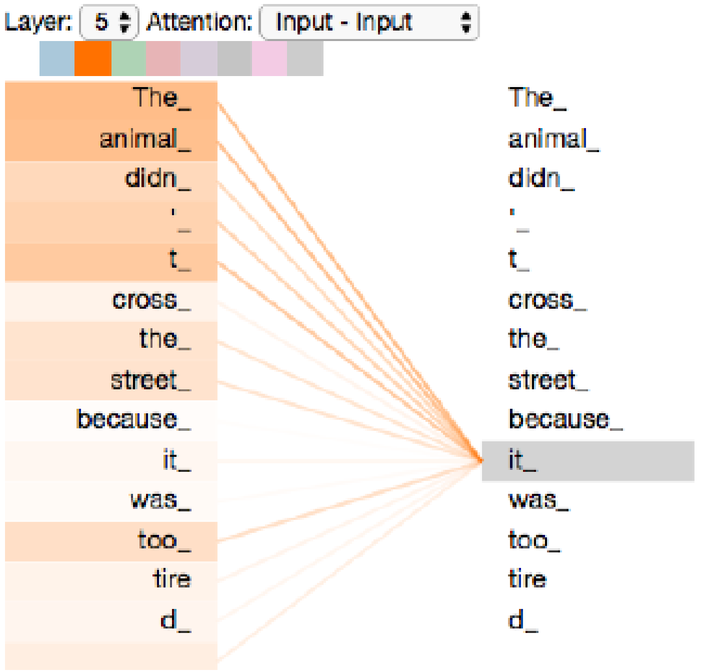




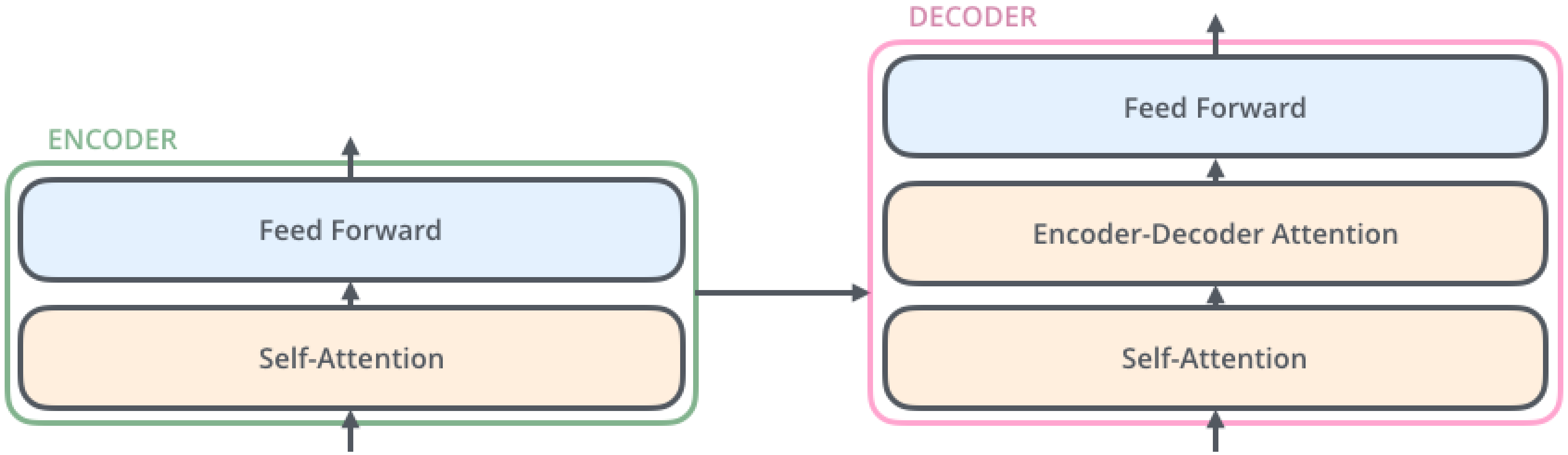
Transformer. Encoder

"The animal didn't cross the street because it was too tired"





Transformer-Decoder



Transformer. Self-attention

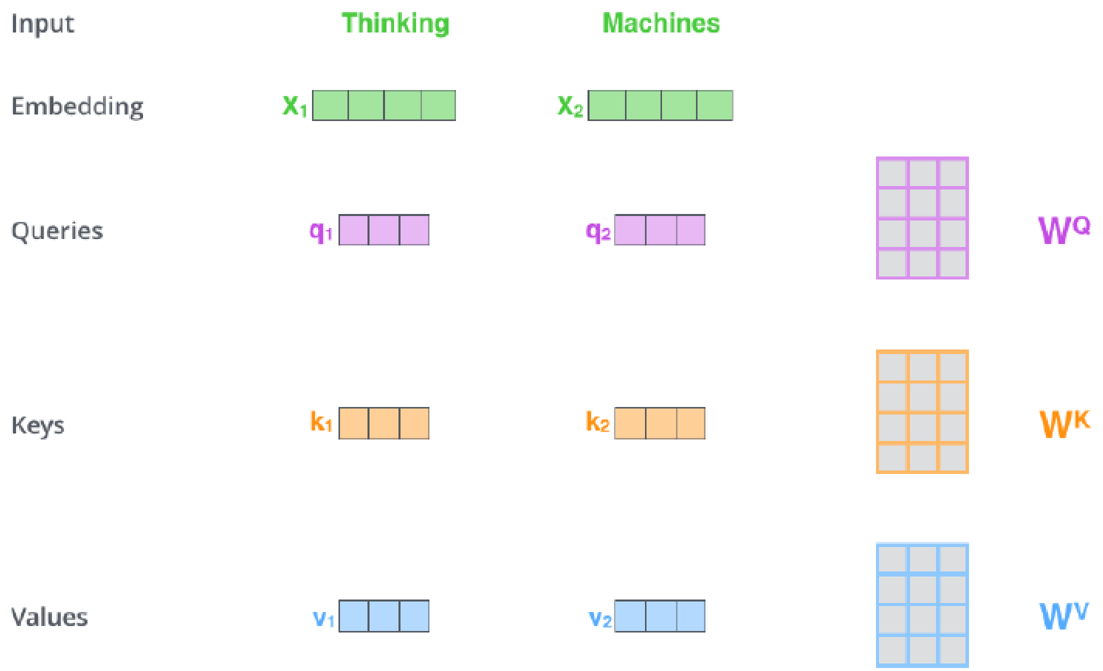
Embedding of i-th token x_i

Query vector $q_i = x_i' \gamma_q$

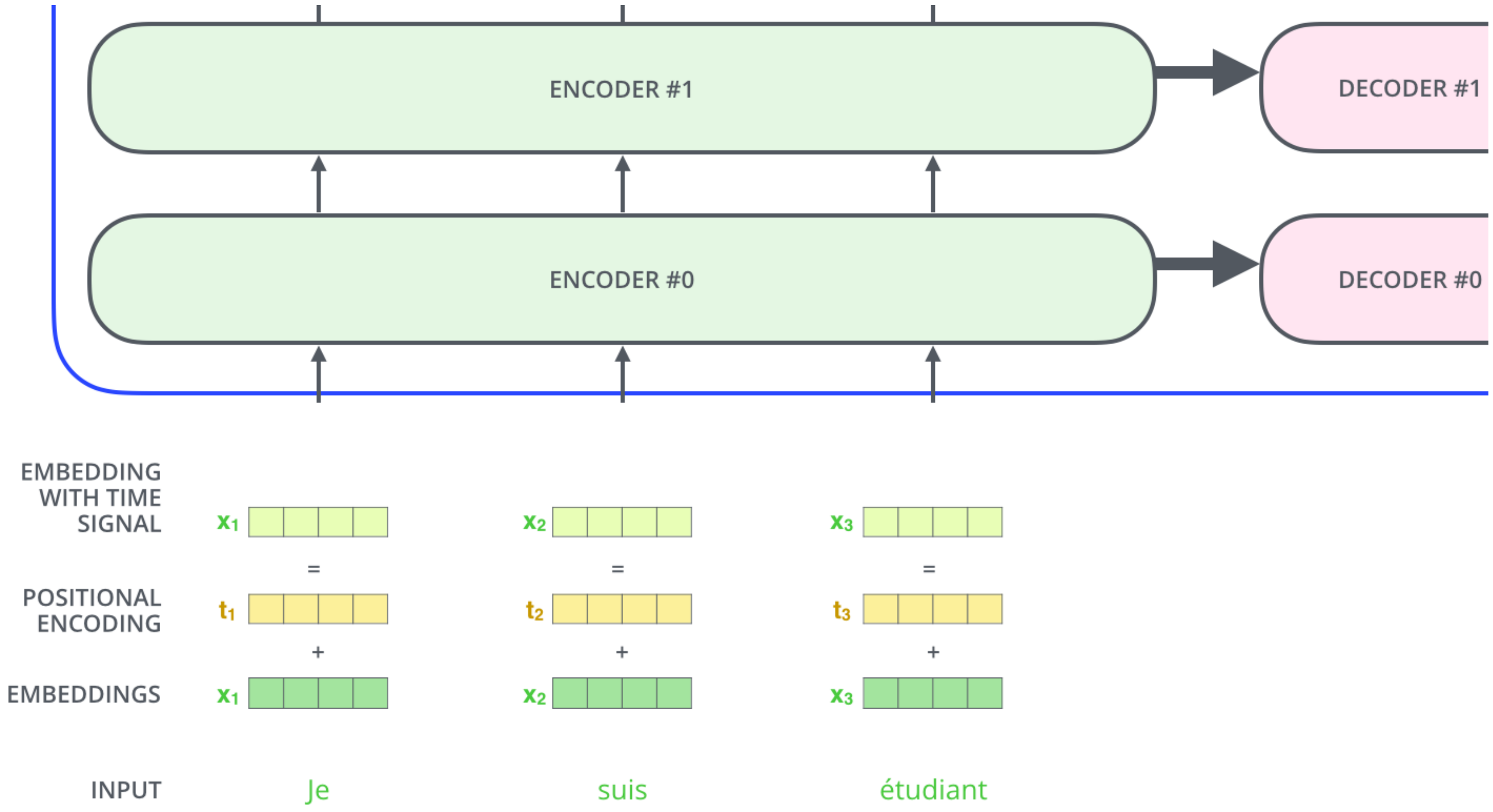
Key vector $k_i = x_i' \gamma_k$

Value vector $v_i = x_i' \gamma_v$

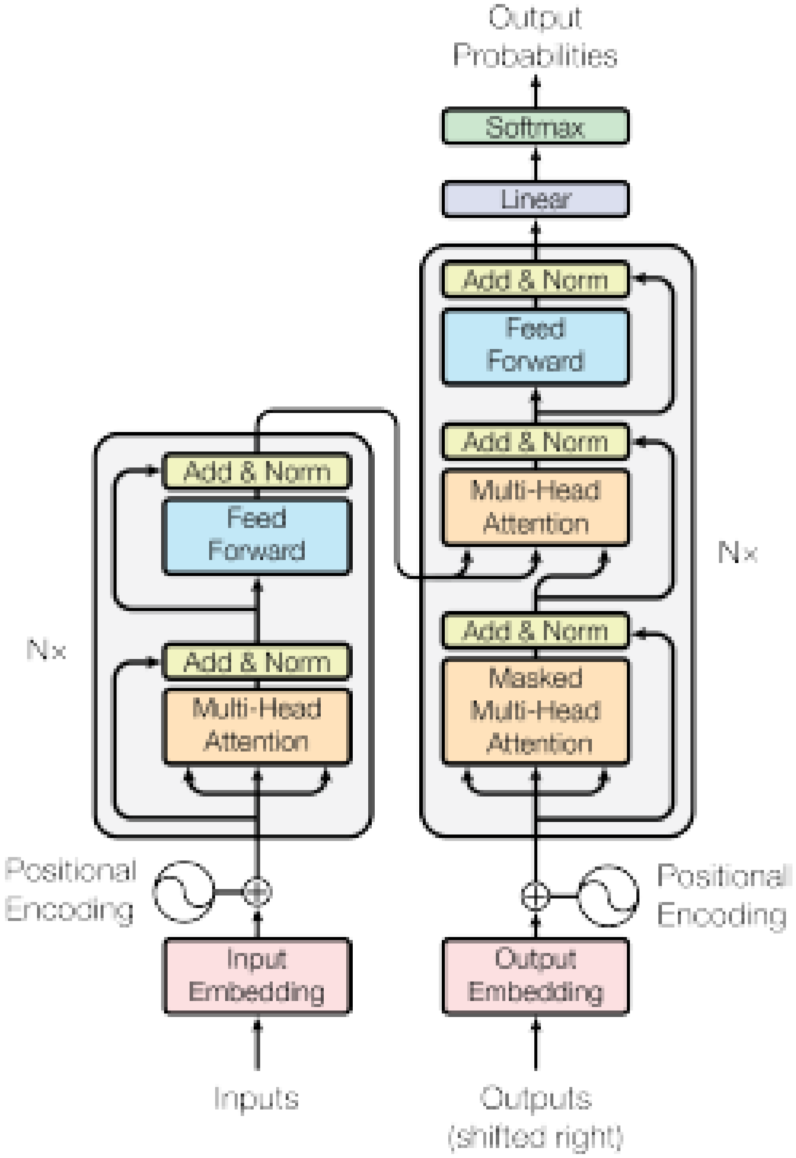
Output $\text{softmax}\left(\frac{qk'}{\sqrt{d_k}}\right) v$



Positional encoding



Transformer. Architecture



Final comments

RNNs in Keras. From Lab

```
model <- keras_model_sequential()
model %>%
  layer_embedding(input_dim = vocab_size, output_dim = 4) %>%
  layer_lstm(4, return_sequences = TRUE, go_backwards=TRUE) %>%
  layer_global_average_pooling_1d() %>%
  layer_dense(units = 4, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
model %>% compile(  optimizer = 'adam',  loss = 'binary_crossentropy',  metrics = list('accuracy'))
history <- model %>% fit(  partial_x_train,  partial_y_train,  epochs = 25,  batch_size = 512,  validation_data = list(x_val,
y_val))
```

RNNs in Keras (R)

layer_simple_rnn(….)
layer_lstm(…)
layer_gru(…)
bidirectional(layer_lstm(…))
layer_conv_1d(…)

For Transformers get from Hugging Face
https://blogs.rstudio.com/ai/posts/2020-07-30-state-of-the-art-nlp-models-from-r/
https://huggingface.co/docs/transformers/index

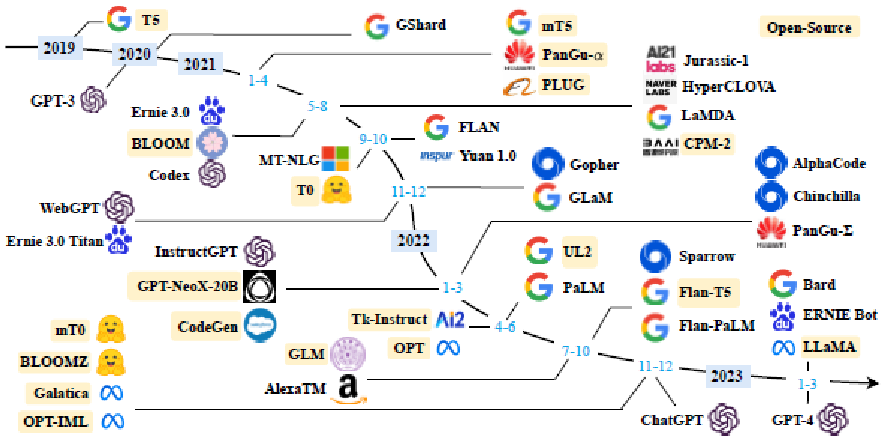
OHE text_tokenizer
Layer_embedding(input_dim,output_dim)
Word2vec, GloVe

From fully connected NNs to RNNs for sequences

From LSTMs to Transformers

Other NN paradigms next (AEs, VAEs, GANs) as part of unsupervised learning

This evolves rapidly!!!



Some pointers to stay tuned

https://www.reddit.com/r/learnmachinelearning/

https://www.reddit.com/r/MachineLearning/

https://medium.com/topic/machine-learning

See you next week

introml@icmat.es

Stuff at

https://datalab-icmat.github.io/courses_stats.html