Formation

Introduction au Deep Learning

Séquence 8

"Données creuses et embedding, données séquentielles et réseaux de Neurones Récurrents (RNN)"

FIDLE

FIDLE

Formation

Introduction au Deep Learning

https://fidle.cnrs.fr

Powered by CNRS CRIC,
and UGA DGDSI of Grenoble, thanks !

- Course materials (pdf)
- Practical work environment (docker)
- Corrected notebooks
- Videos (YouTube)

(*) Procedure via Docket or pip
Remember to get the latest version !

Formation

Introduction au Deep Learning

Questions and answers :

https://fidle.cnrs.fr/q2a

Accompanied by :
IA Support (dream) Team of IDRIS

Directed by :
Agathe, Baptiste et Yanis - UGA/DAPI
Thibaut, Kamel - IDRIS

FIDLE

Formation

Introduction au Deep Learning

FIDLE https://fidle.cnrs.fr/listeinfo

Fidle information list

Agoria http://fidle.cnrs.fr/agoria

AI exchange list

agoria@grenoble.cnrs.fr

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations

FIDLE

Formation

Introduction au Deep Learning

https://listes.services.cnrs.fr/wws/info/devlog
List of ESR* « Software developers » group

GROUPECALCUL

https://listes.math.cnrs.fr/wws/info/calcul
List of ESR* « Calcul » group

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations

FIDLE

SAISON 23/24

Bases, Concepts et Enjeux

1 ✓
History and Fundamental Concepts

2 ✓ New !
Data, models and representation's hell
Data and models

3 ✓ New !
Demonstration Illustration
LLM / Text to Image

4 ✓
AI, Law, Society and **Ethics**

L'IA comme un outil,

5 ✓ New !
Mathematics, gradients everywhere !

6 ✓ New !
Learning methodology

7 ✓
Convolutional models
CNN

8
Sparse (text) and sequences data
Embedding, RNN

9
«Attention is All You Need»
Transformers

10
Graph Neural Network
GNN

11
Autoencoder networks
AE

12
Variational Antoencoder
VAE

13
Generative Adversarial Networks
GAN

14
Diffusion Model
Text to image

15
Deep Reinforcement Learning
RL

16
Physics-Informed Neural Networks
PINNS

Acteur de l'IA

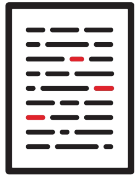
17
Learning faster and cheaper,
Eco-Friendly

18
Jean-Zay
GPU acceleration

19 New !
New models
VLM, SM, Multimodal, ...

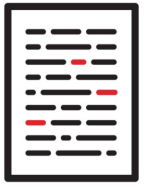
20 New !
Case Study
Experience feedback

8



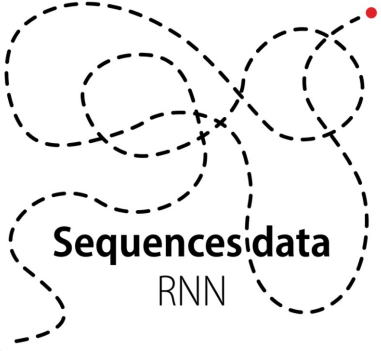
Sparse (text) and sequences data
Embedding, RNN

Part 1



Sparse data (text)
Embedding

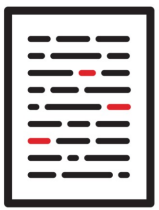
Part 2



9

«Attention is All You Need»
Transformers

8

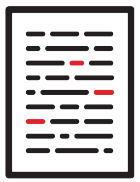


Sparse data (text)
Embedding

Part 1

1. Text **encoding**
   - From text to tensor
   - One hot encoding
   - Embedding
2. Example 1 : IMDB1
   - Sentiment analysis with **one-hot encoding**
3. Example 2 : IMDB2/3/4
   - Sentiment analysis with **embedding**

8



Sparse (text) and sequences data

Embedding, RNN

Part 1

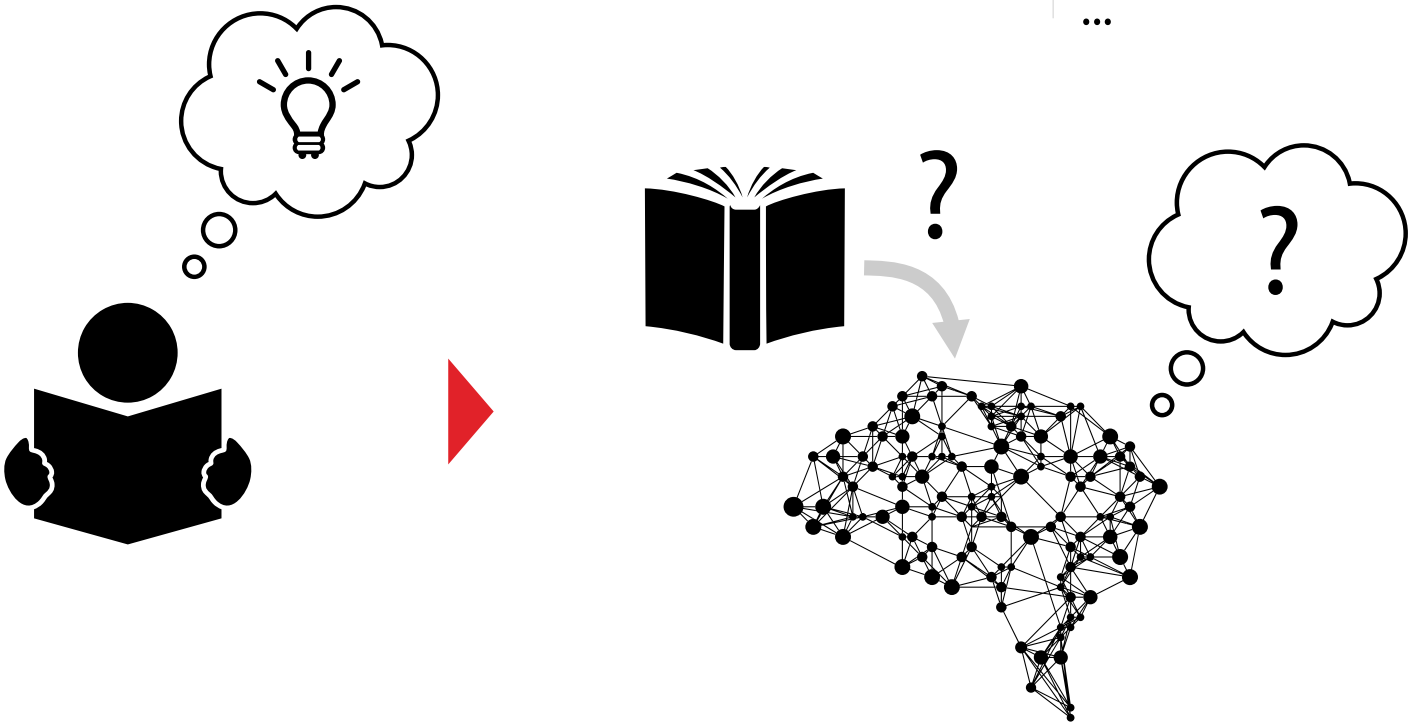
1 **Text encoding**
- From text to tensor
- One hot encoding
- Embedding

2 **Example 1 : IMDB1**
- Sentiment analysis with **one-hot encoding**

3 **Example 2 : IMDB2/3/4**
- Sentiment analysis with **embedding**

How to feed a neural network with text ?

Variable size
Large size space
Conceptual contents ;-)
...



How to feed a neural network with text ?

« I've never seen a movie like this before. »

How to build a descriptor for this kind of data (text, DNA, ...)?

One hot encoding

« I've never seen a movie like this before. »

Tokenization
(words)

["i've", 'never', 'seen', 'a', 'movie', 'like', 'this', 'before']

Dictionary

0	a
1	before
2	fantastic
3	i’ve
4	is
5	like
6	movie
7	never
8	seen
9	this

The word order in the dictionary may have some meaning. For example according to their rate of use.

One hot encoding

« I've never seen a movie like this before. »

Tokenization (words)

Dictionary

0	a
1	before
2	fantastic
3	i’ve
4	is
5	like
6	movie
7	never
8	seen
9	this

["i've", 'never', 'seen', 'a', 'movie', 'like', 'this', 'before']

[3, 7, 8, 0, 6, 5, 9, 1]

Vectorization

The values associated with each word are meaningless: words with a contiguous subscript are typically unrelated.

One hot encoding

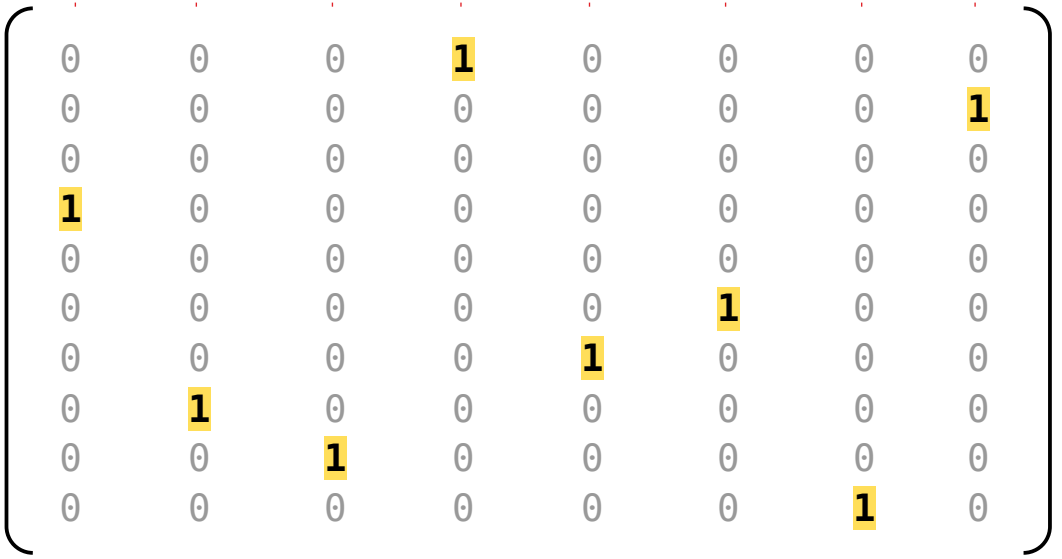
« I've never seen a movie like this before. »

["i've", 'never', 'seen', 'a', 'movie', 'like', 'this', 'before']

[3, 7, 8, 0, 6, 5, 9, 1]

Dictionary

0	a
1	before
2	fantastic
3	i've
4	is
5	like
6	movie
7	never
8	seen
9	this



Tokenization (words)

Vectorization

001000

One hot encoding

One hot encoding

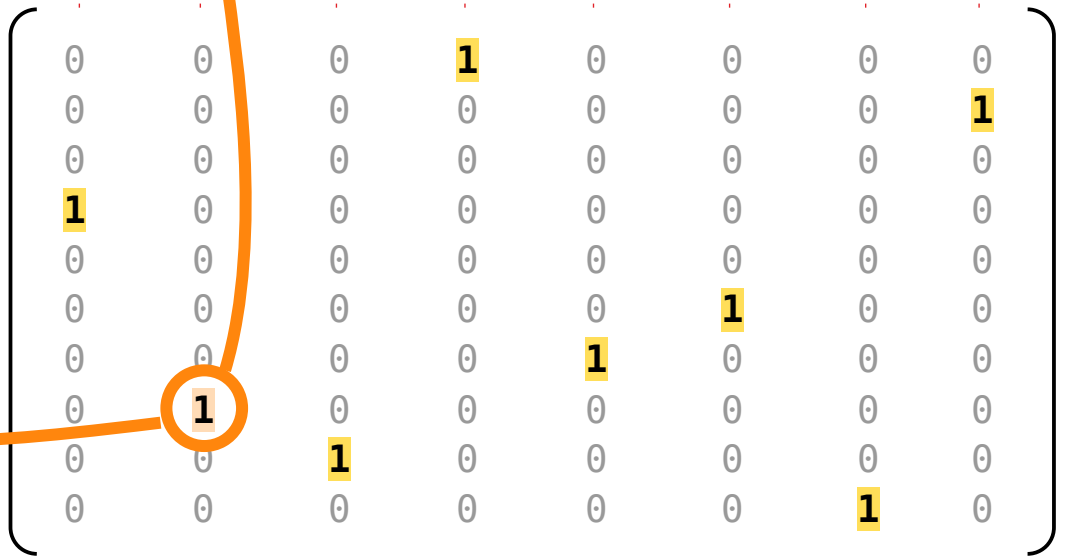
« I've never seen a movie like this before. »

["i've", 'never', 'seen', 'a', 'movie', 'like', 'this', 'before']

[3, 7, 8, 0, 6, 5, 9, 1]

Dictionary

0	a
1	before
2	fantastic
3	i've
4	is
5	like
6	movie
7	never
8	seen
9	this



Tokenization (words)

Vectorization

001000
One hot encoding

Each vector is independent. Each word has its **own dimension**.

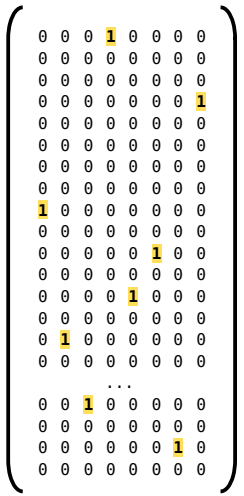
One hot encoding

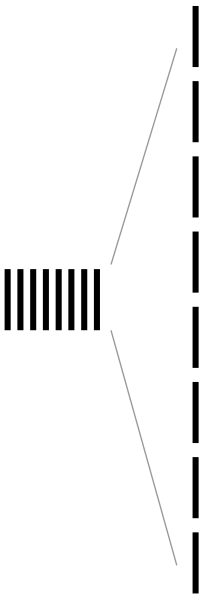
Limits of full one-hot encoding : Size !

Example :

Dictionary = 80 000 words

Sentence = 300 words





Need :

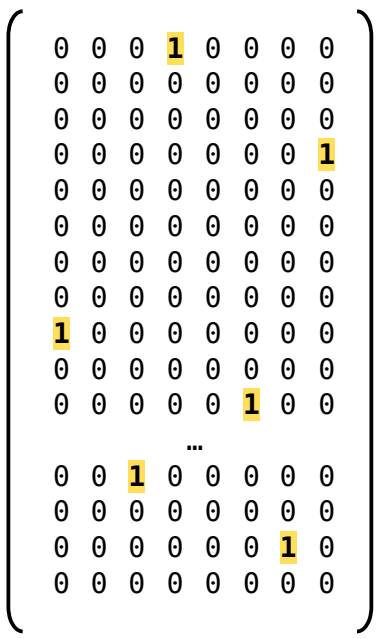
24 10^6

Parameters !

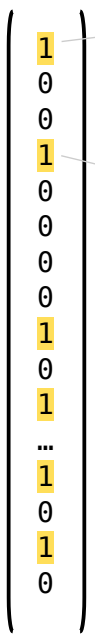
2 answers : { One hot vector ; Embedding

One hot vector encoding

Solution 1 : one hot vector



Logical OR
on each line



1 This word is **present** in the sentence

0 This word is **not present** in the sentence

Vector size = Dictionary size

The size no longer depends on the length of the sentence.
The size is drastically reduced, but we lose the words order.

Solution 2 : Using embedding

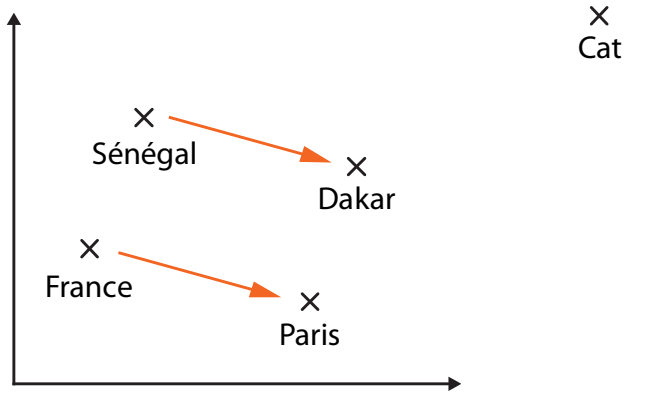
$(Word_i)$ $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

(W_i) $\begin{pmatrix} 0.7 \\ 4.2 \\ 1.6 \end{pmatrix}$

Short dense vector, whose value carries a semantic meaning of the word.

Long sparse vector, whose value just indicates membership in the dictionary

Semantic meaning, allows to perform calculations, such as :



dist('France', 'Cat') >> dist('France', 'Sénégal')

vect('France', 'Paris') ≈ vect('Sénégal', 'Dakar')

'Paris' = 'France' + vect('Sénégal', 'Dakar')

« Paris est à la France ce que Dakar est au Sénégal »

Embedding

Solution 2 : Using embedding

$(Word_i)$ $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ ▶ (W_i) $\begin{pmatrix} 0.7 \\ 4.2 \\ 1.6 \end{pmatrix}$

Short dense vector, whose value carries a semantic meaning of the word.

Long sparse vector, whose value just indicates membership in the dictionary

Number of parameters required :

Example :

Embedding size : 200

Sentence size : 300 words

Need only : 60.000 Parameters :-)

Embedding techniques :

- Contextual embedding
  - Keras embedding
- Global embedding
  - CBOW,
  - SG,
  - GloVe,
  - etc.

Embedding layers in Keras

Usable as a **simple layer**

This layer will constitute a **dictionary of vectors** which will be **optimized during the learning process**, according to the expected result and not to the pure semantics.

Keras embedding is therefore adapted to **classification**, but will not be able to take into consideration, for example, semantic similarities (such as identifying 2 sentences with the same semantic meaning).

The **output** of the layer is a **set of vectors**.

Embedding layers in Keras

(french)

Utilisable comme une **simple couche**

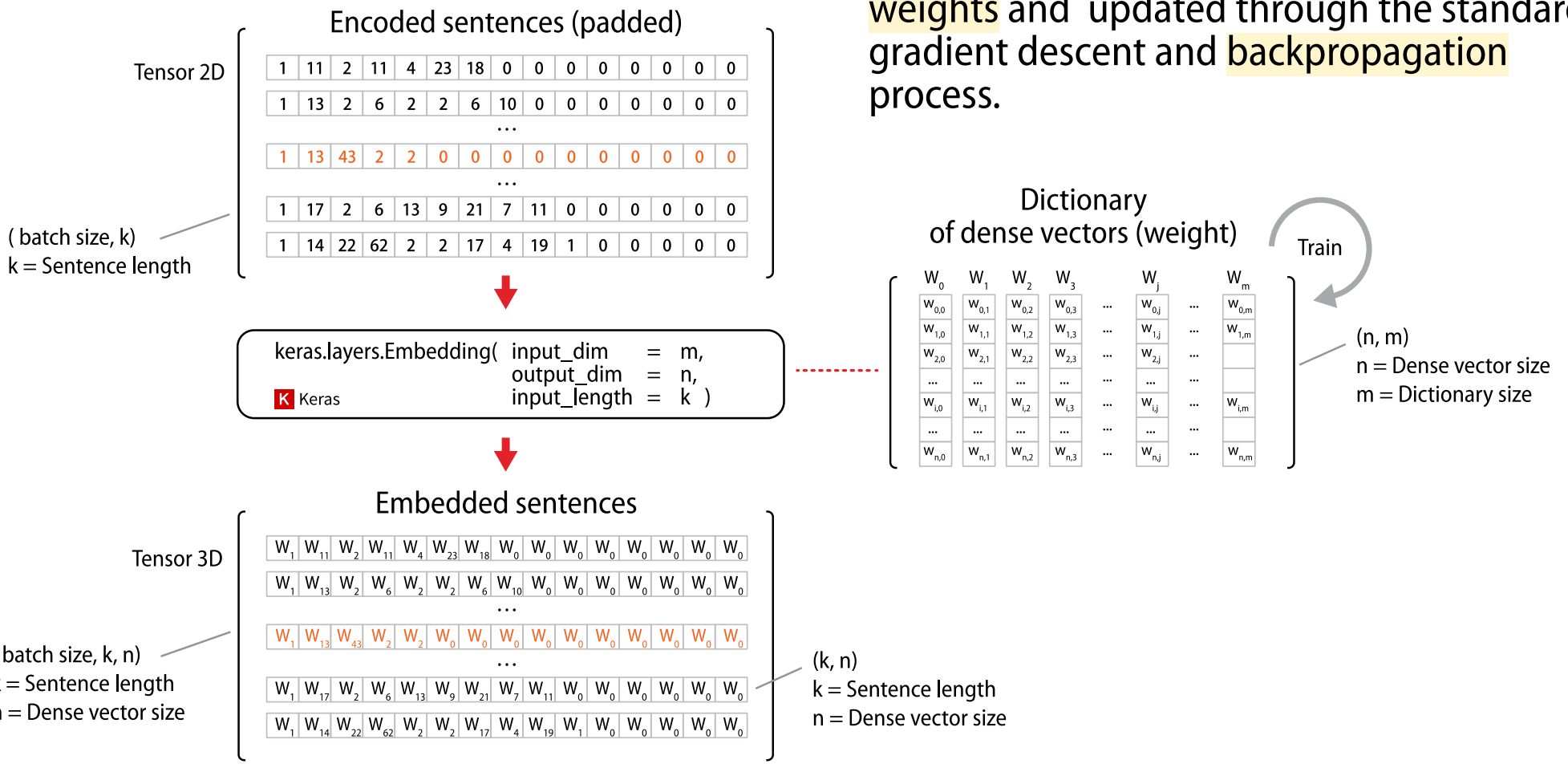
Cette couche va constituer un **dictionnaire de vecteurs** qu'elle optimisera au cours de l'apprentissage, en **fonction du résultat** attendu et non de la sémantique pure.

L'embedding Keras est donc adapté à la **classification**, mais ne rendra pas compte, par exemple, de similarités sémantiques (comme identifier 2 phrases ayant un même sens sémantique).

La **sortie** de la couche est un **ensemble de vecteurs**

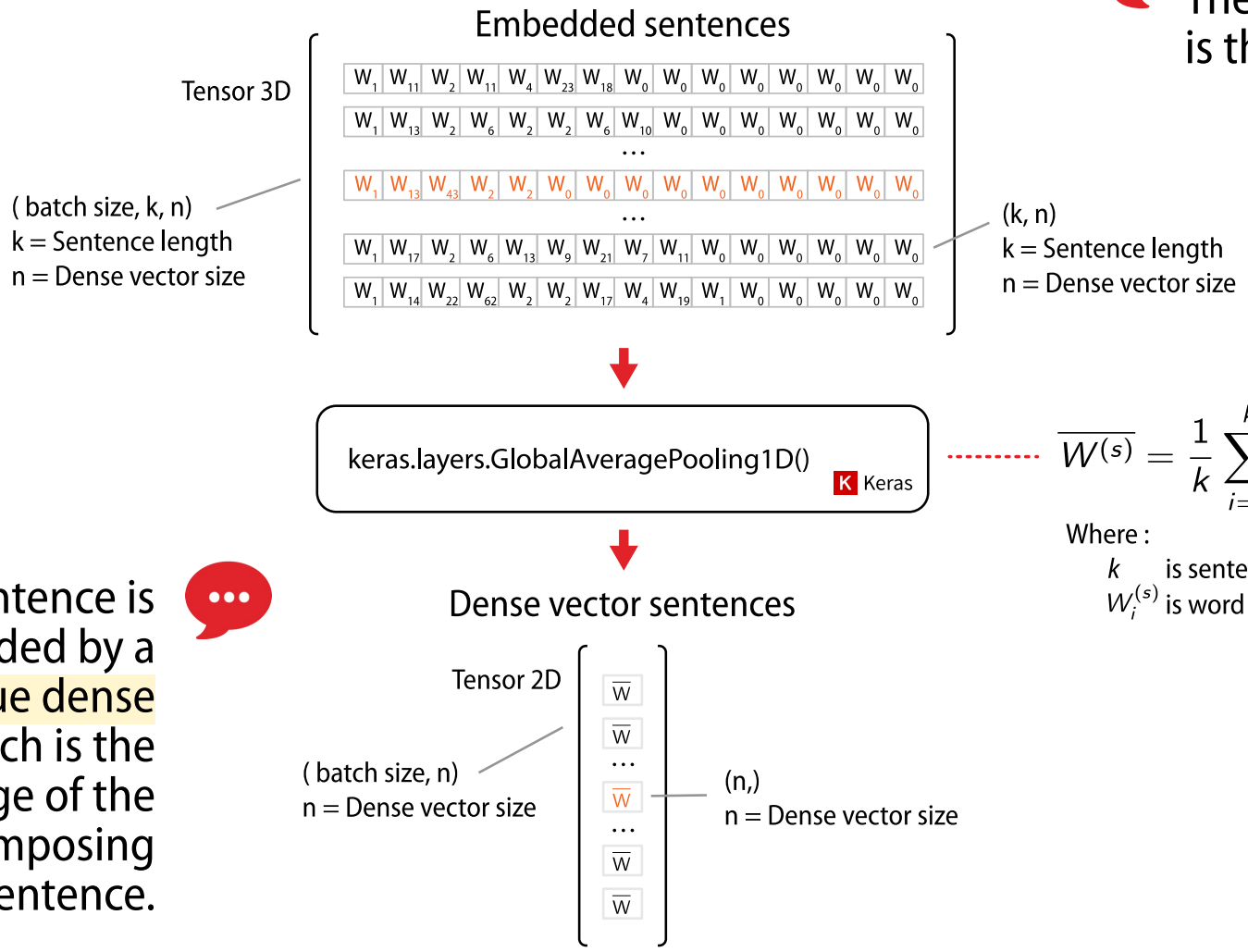
Keras embedding layer

Dense vector components are managed as weights and updated through the standard gradient descent and backpropagation process.



Keras embedding layer

The order of the words is therefore ignored !



$$\overline{W^{(s)}} = \frac{1}{k}\sum_{i=1}^{k} W_i^{(s)}$$

Where :

k is sentence length

$W_i^{(s)}$ is word vector i of sentence (s)

Each sentence is here encoded by a unique dense vector which is the average of the words composing the sentence.

Word2Vec

This approach aims to build **dictionaries** whose vector representation of words is based on **context** and therefore on **semantics**.

Dictionaries built from large corpora are available.

Two models:

- **Continuous Bag-of-Words** (CBOW),
- **Skip-Gram** (SG).

These approaches are historically interesting, but now a bit outdated... ;-)

Efficient Estimation of Word Representations in Vector Space
Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, 2013
https://doi.org/10.48550/arXiv.1301.3781

CBOW : Continuous Bag of Words - Embedding based on the prediction of the word according to its context.
SG : Skip-gram - Embedding based on context prediction from the word.

Word2Vec

Approche ayant pour objectif de constituer des **dictionnaires** *(french)* dont la représentation vectorielle des mots est basée sur le **contexte** et donc de la **sémantique**.

Des dictionnaires construits à partir de gros corpus sont disponibles.

Deux modèles :

- **Continuous Bag-of-Words** (CBOW),
- **Skip-Gram** (SG).

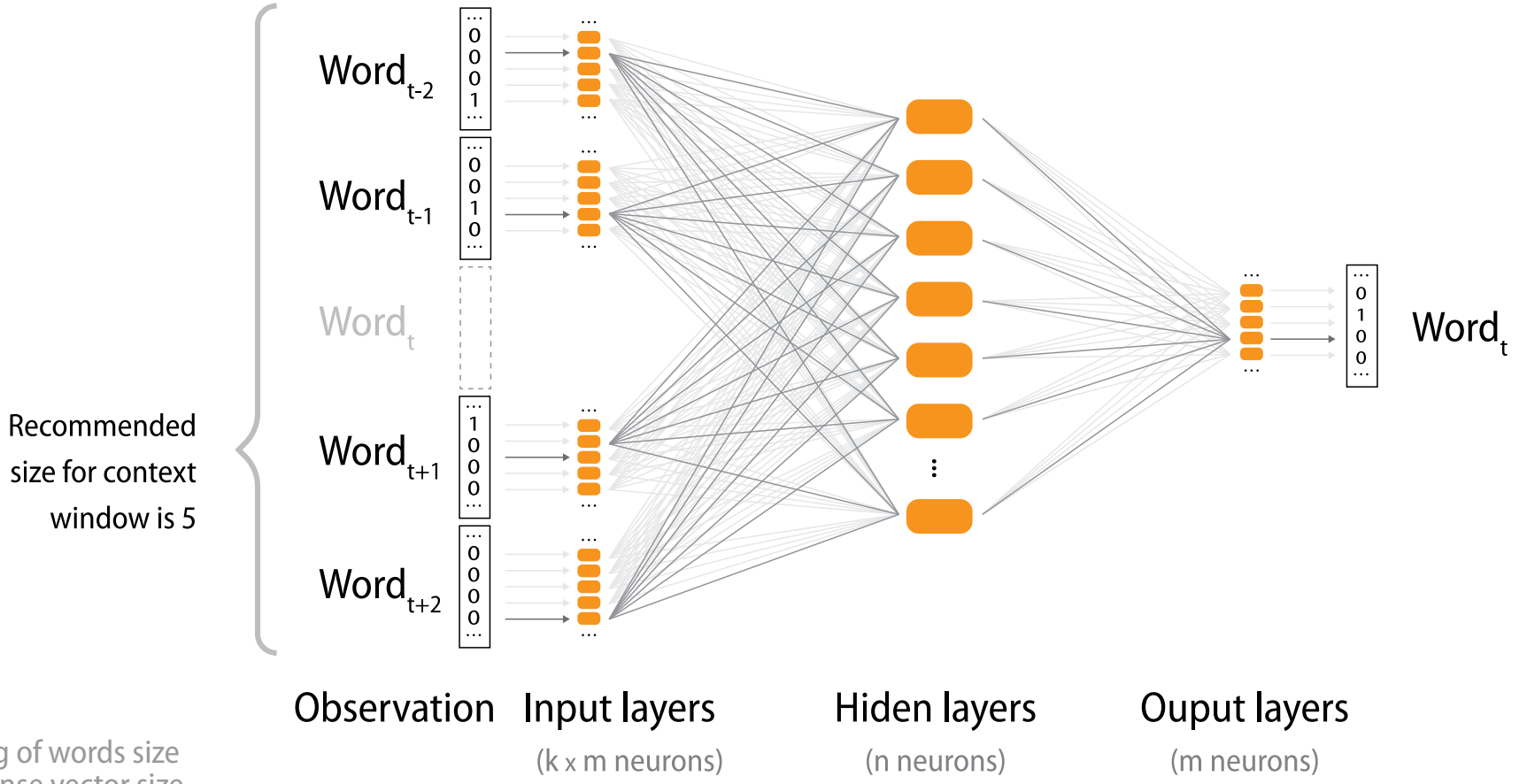
Ces approches sont historiquement intéressantes, mais désormais un peu dépassées... ;-)

Efficient Estimation of Word Representations in Vector Space
Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, 2013
https://doi.org/10.48550/arXiv.1301.3781

CBOW : Continuous Bag of Words - Embedding based on the prediction of the word according to its context.
SG : Skip-gram - Embedding based on context prediction from the word.

Continous Bag-of-Words

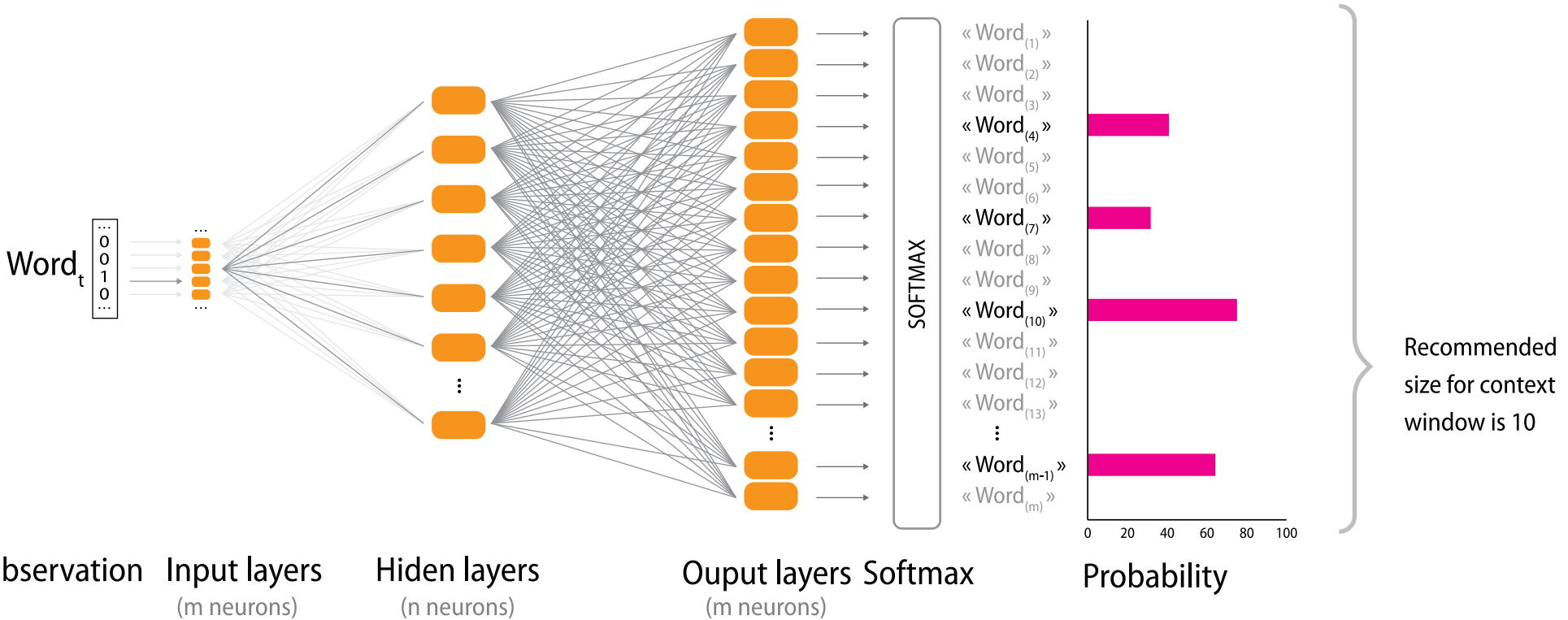
The objective of training is to (re)find a word from its context.



k = Bag of words size
n = Dense vector size
m = Dictionary size

Skip-Gram

The objective of training is to (re)find the context from the word.



k = Bag of words size
n = Dense vector size
m = Dictionary size

(Flau)BERT

2018

BERT est une représentation du langage proposée par Google en 2018, permettant de prendre en compte la dimension contextuelle du langage (« Avocat », peut être un fruit ou un juriste..).

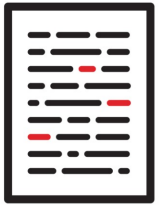
FlauBERT[2] est une adaptation de l'algorithme au français.

To be continued !

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
https://arxiv.org/abs/1810.04805

FlauBERT: Unsupervised Language Model Pre-training for French.
Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux,
Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé,
Laurent Besacier, Didier Schwab
https://arxiv.org/abs/1912.05372

8



Sparse data (text)
Embedding

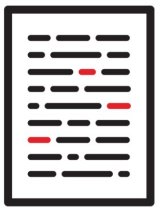
Part 1

1 **Text encoding**
- From text to tensor
- One hot encoding
- Embedding

2 **Example 1 : IMDB1**
- Sentiment analysis with **one-hot encoding**

3 **Example 2 : IMDB2/3/4**
- Sentiment analysis with **embedding**

8



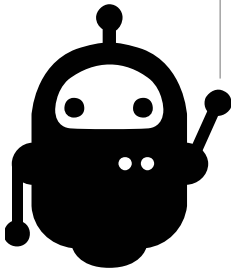
Sparse data (text)
Embedding

Part 1

1. Text **encoding**
   - From text to tensor
   - One hot encoding
   - Embedding
2. Example 1 : IMDB1
   - Sentiment analysis with **one-hot encoding**
3. Example 2 : IMDB2/3/4
   - Sentiment analysis with **embedding**

One-Hot encoding with IMDB

Notebook : [K3IMDB1]



Objective :
Guess whether a film review is positive or not based on the analysis of the text, using One-Hot encoding.

Dataset :
The IMDB dataset is composed of 50,000 film reviews from the site of the same name.

https://www.imdb.com/

K3IMDB1

<60 s

Import and init

Understanding hot-one encoding

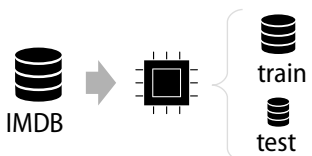
About our dataset

Build the model

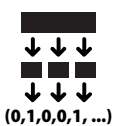
Evaluate

START → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → END

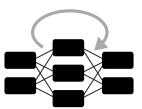
Retrieve data



One-hot vector encoding



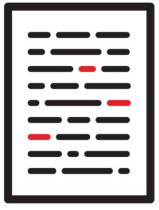
Train the model



Objectives :

Analysis of film reviews with One-Hot encoding

8



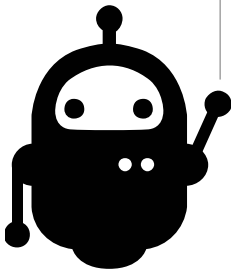
Sparse data (text)
Embedding

Part 1

1. Text **encoding**
   - From text to tensor
   - One hot encoding
   - Embedding
2. Example 1 : IMDB1
   - Sentiment analysis with **one-hot encoding**
3. Example 2 : IMDB2/3/4
   - Sentiment analysis with **embedding**

Text embedding with IMDB

Notebook : [K3IMDB2-4]

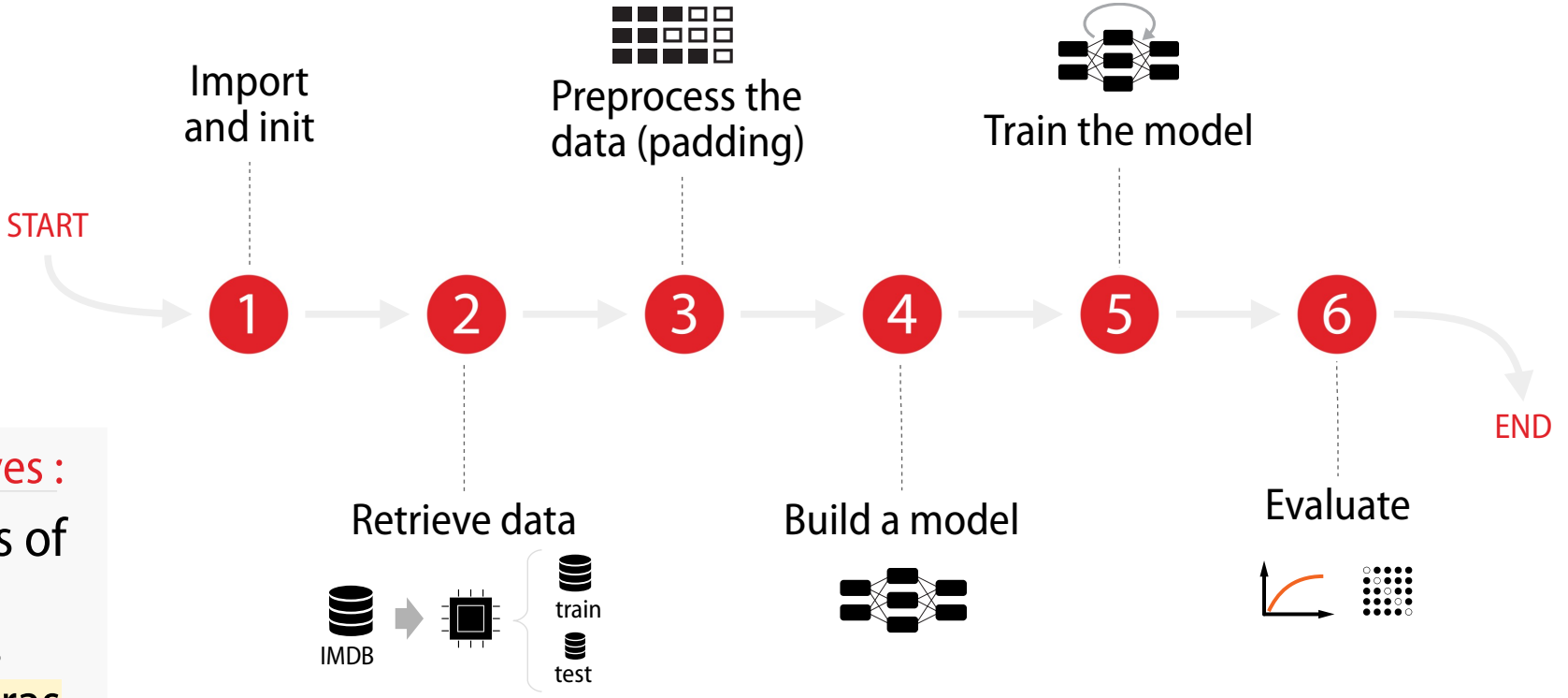


Objective :
Guess whether a film review is positive or not based on the analysis of the text, using embedding.

Dataset :
The IMDB dataset is composed of 50,000 film reviews from the site of the same name.

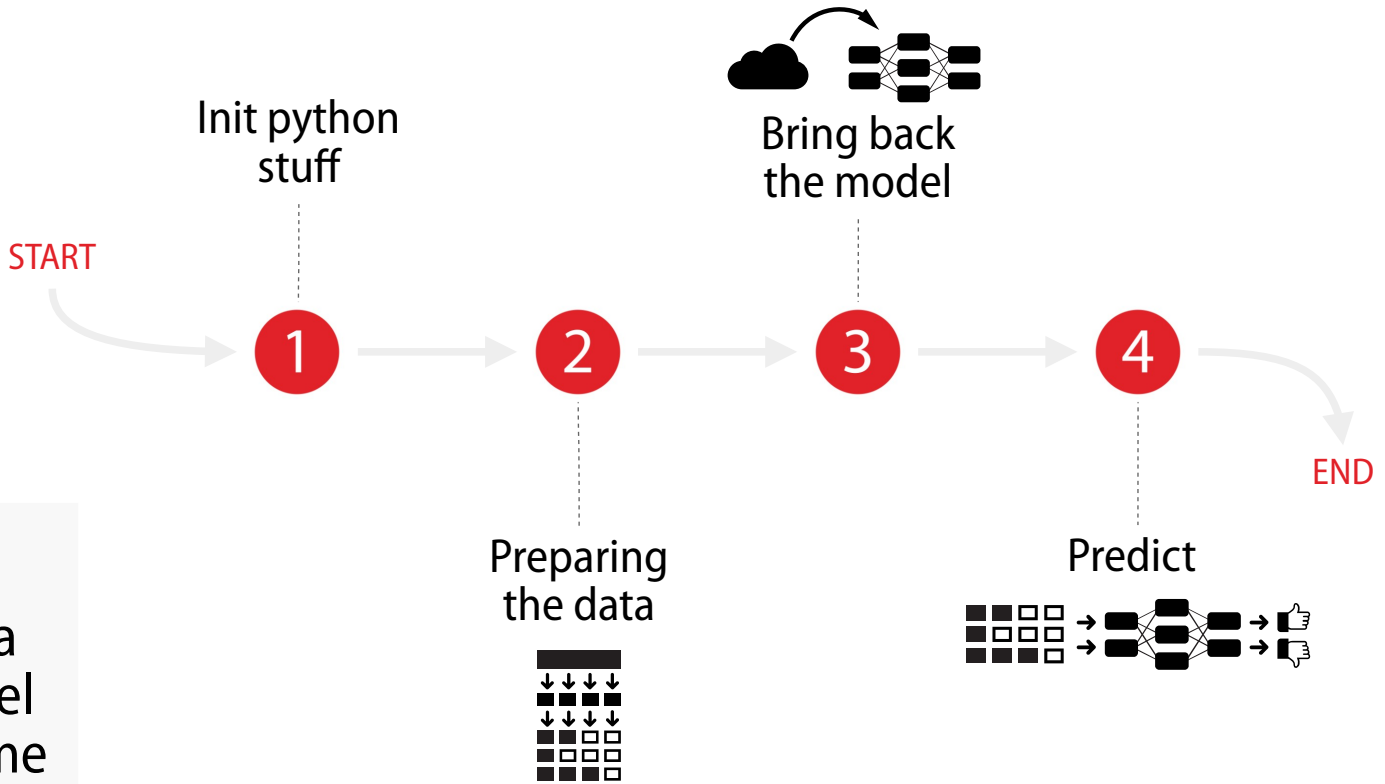
https://www.imdb.com/

<60 s



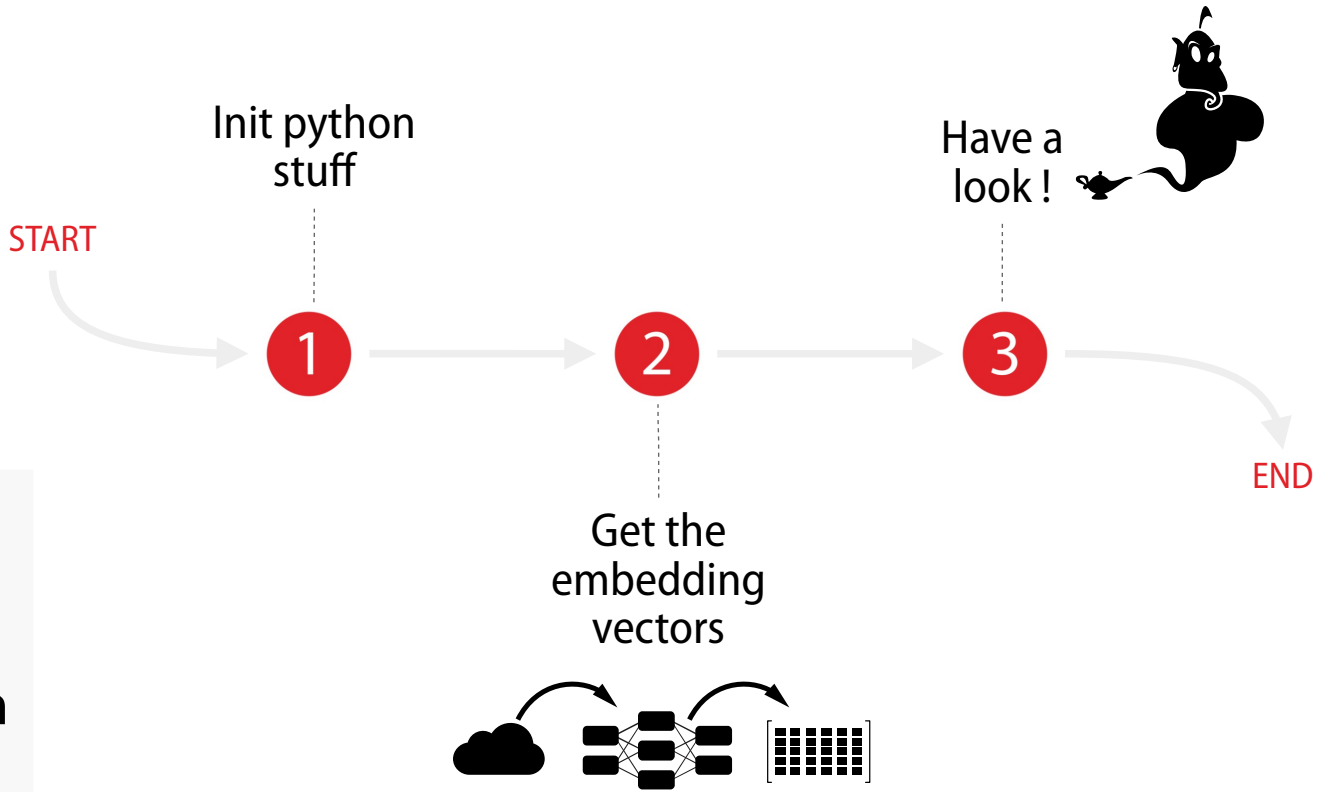
Objectives :

Analysis of film reviews with Keras embedding



Objectives :

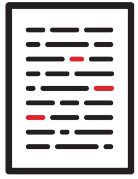
Retrieving a saved model to treat some new reviews



Objectives :

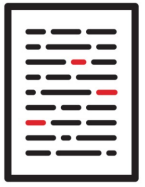
Retrieving a saved model and play with embedded vectors

8



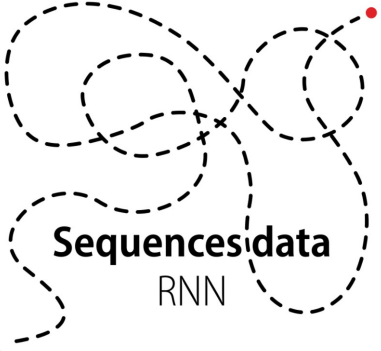
Sparse (text) and sequences data
Embedding, RNN

Part 1



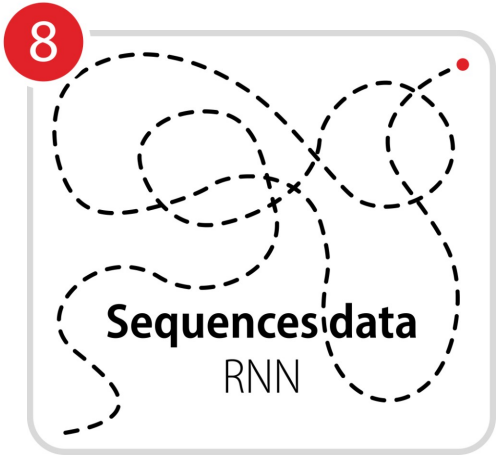
Sparse data (text)
Embedding

Part 2



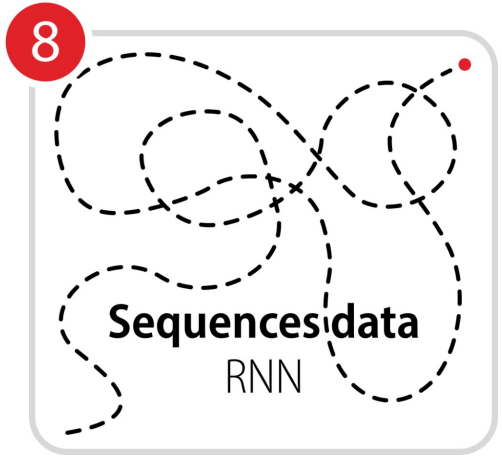
9

«Attention is All You Need»
Transformers



Part 2

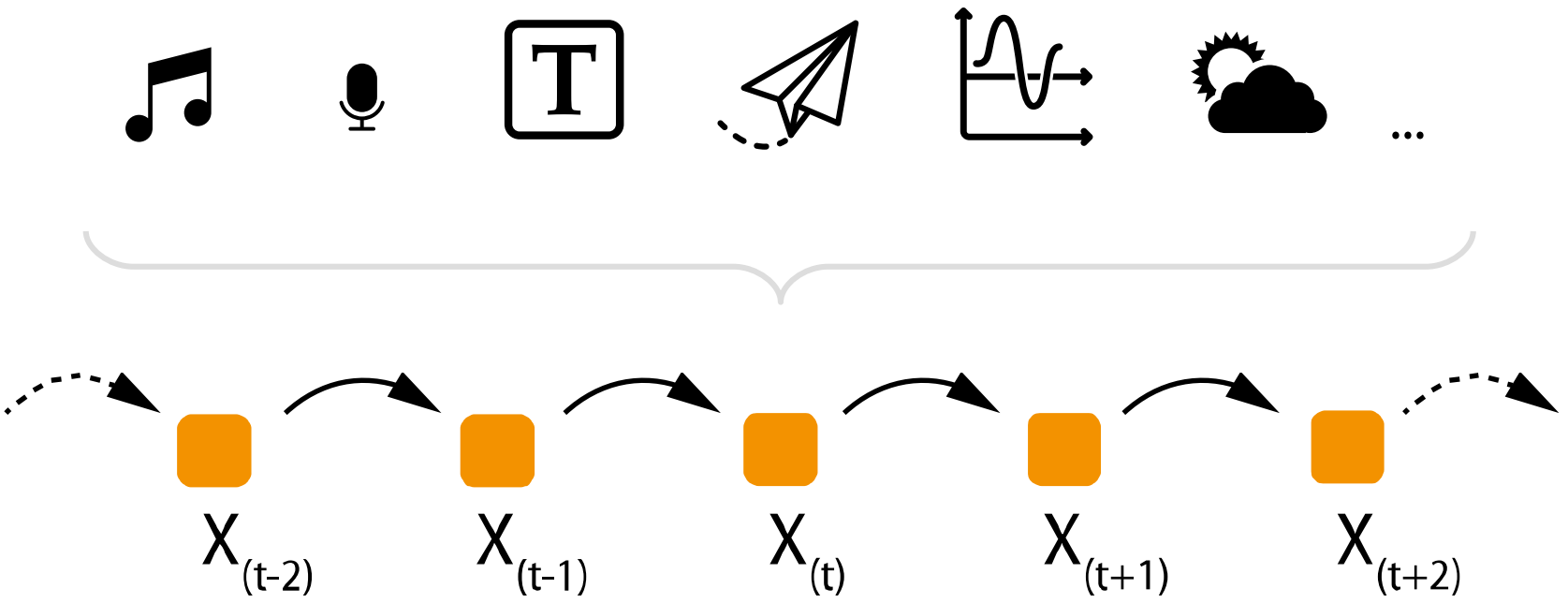
1. **Sequences** data
   - Recurrent Neural Network
   - LSTM and GRU
2. Example 1 : LadyBug
   - Prediction of a virtual trajectory



Part 2

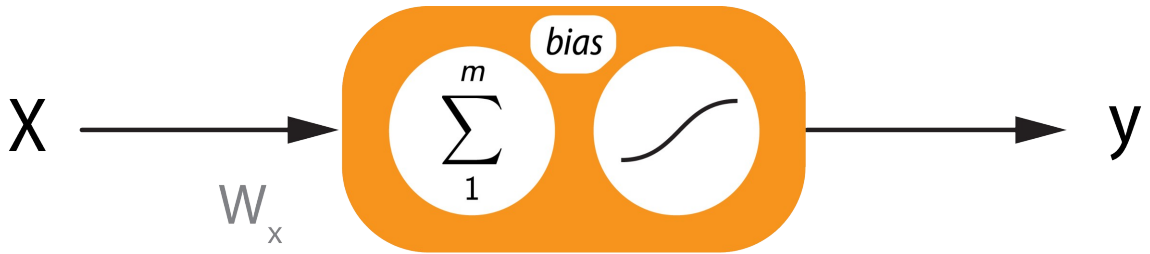
1. **Sequences** data
   - Recurrent Neural Network
   - LSTM and GRU
2. Example 1 : LadyBug
   - Prediction of a virtual trajectory

What if the world was just one big sequence ?



From classic to recurrent neuron

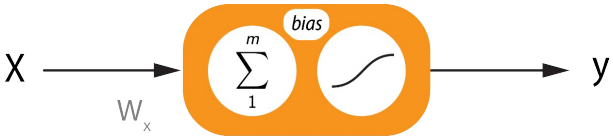
Classical neuron.



$$y = \sigma(W_x^T \cdot X + b)$$

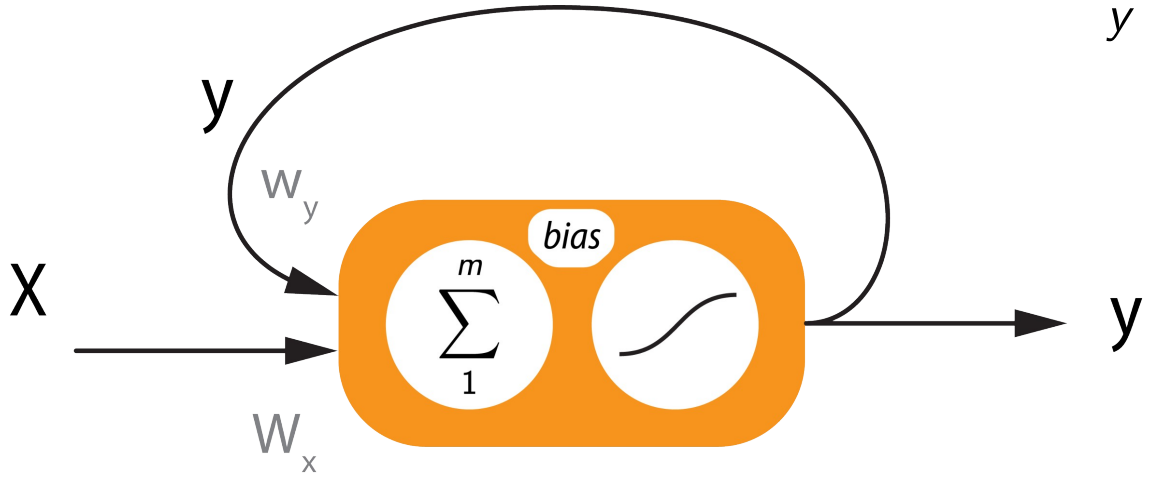
From classic to recurrent neuron

Classical neuron.



$$y = \sigma(W_x^T \cdot X + b)$$

Recurrent neuron.

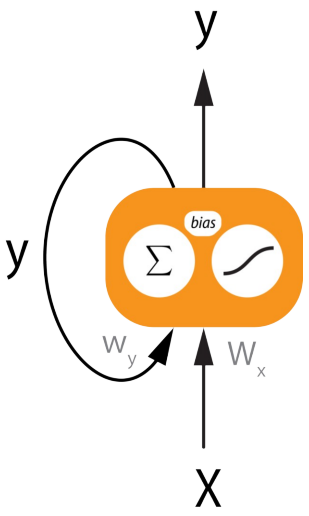


$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar

Simple reccurent neuron

Recurrent neuron.

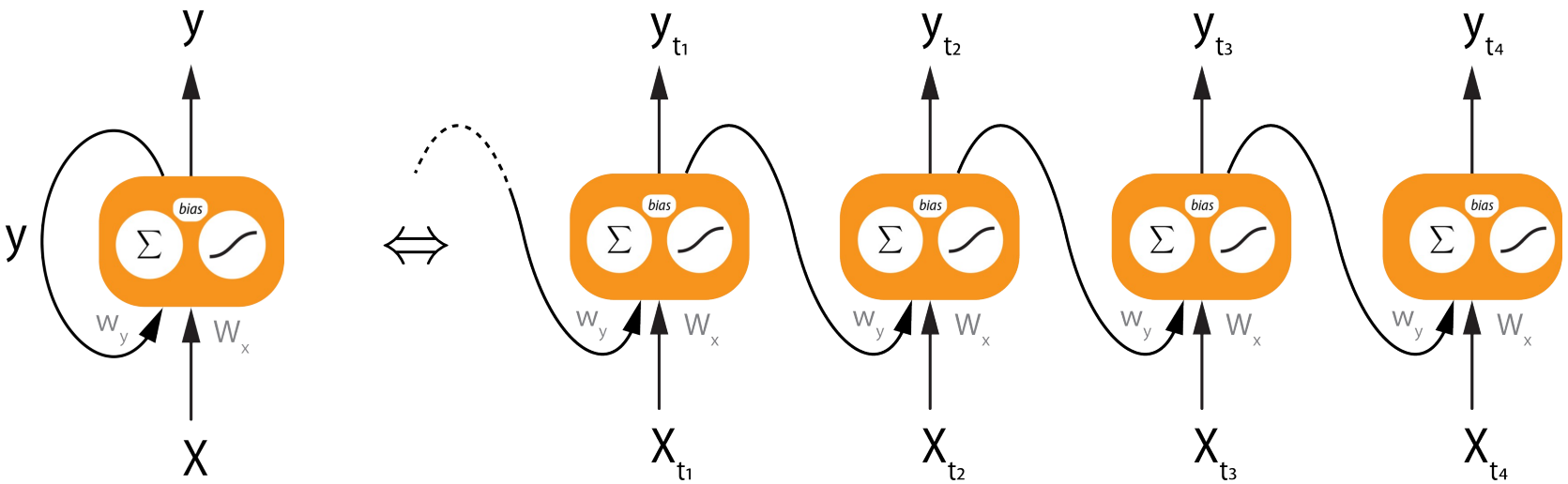


$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar

Simple reccurent neuron

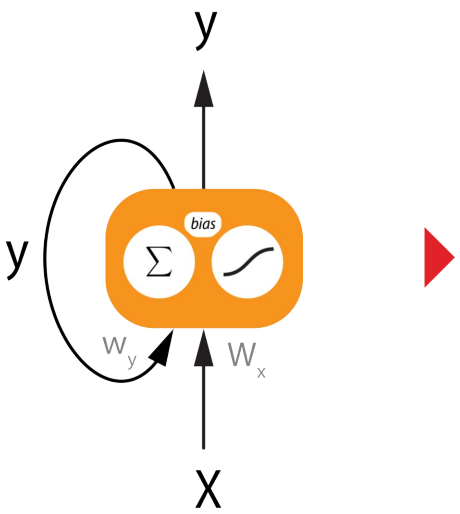
Recurrent neuron.



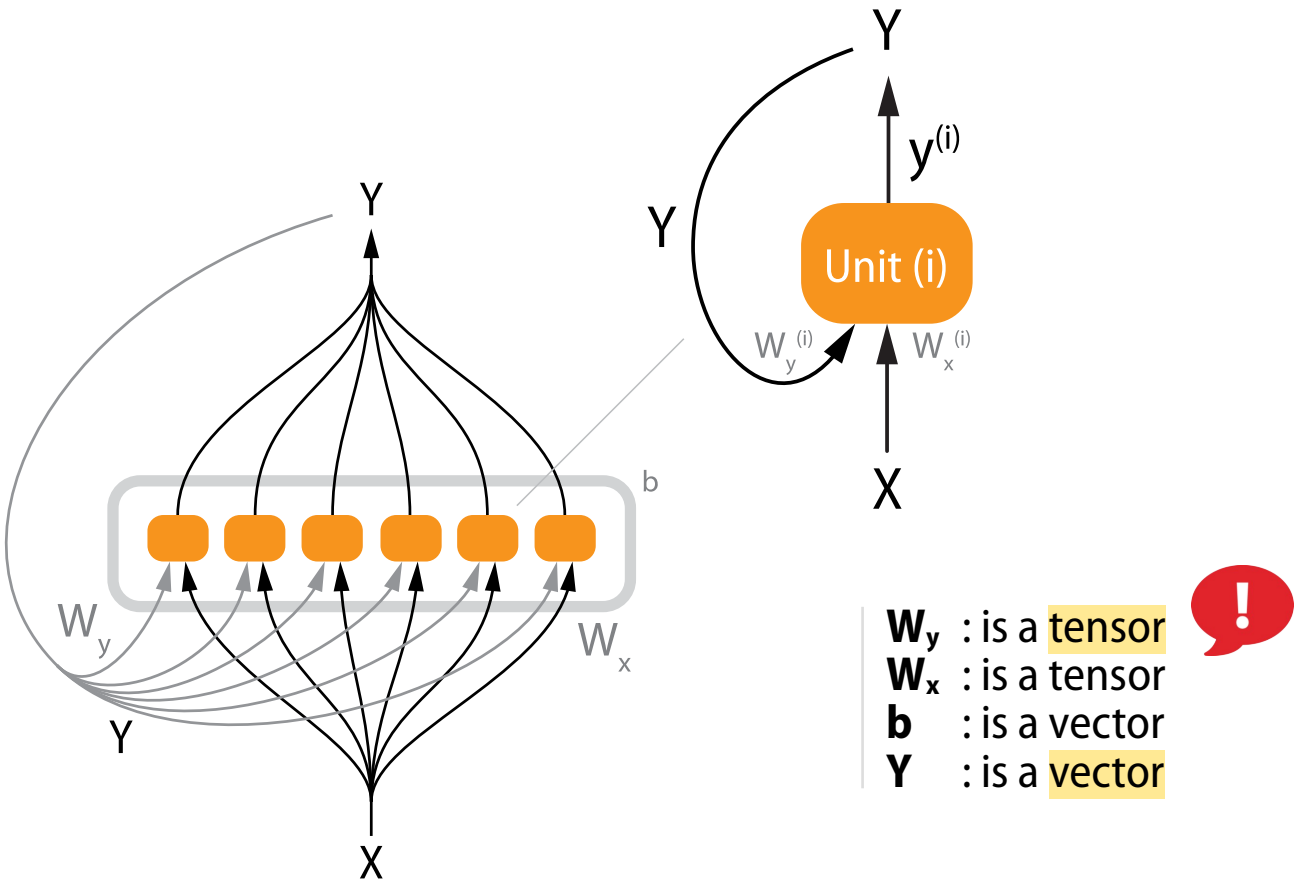
w_y : is a scalar
W_x : is a vector
b : is a scalar
y : is a scalar

$$y_{(t)} = \sigma(W_x^T \cdot X_{(t)} + w_y \cdot y_{(t-1)} + b)$$

Recurrent **Layer / Cell**



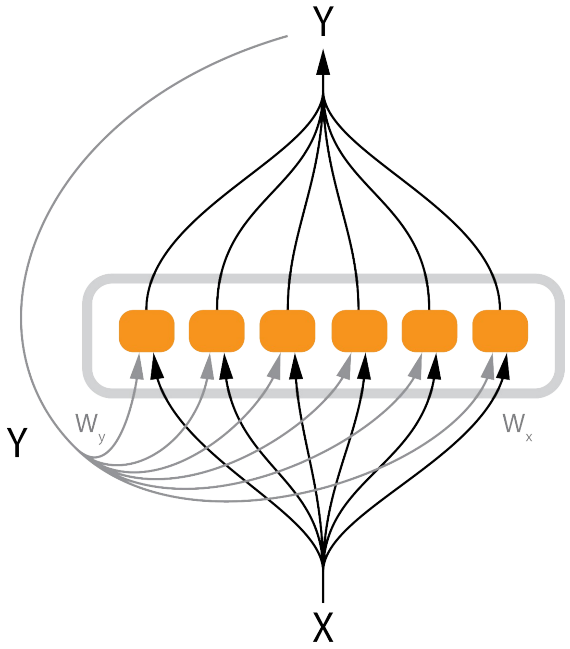
Recurrent **neuron**.



Recurrent **Layer / Cell**

W_y : is a tensor
W_x : is a tensor
b : is a vector
Y : is a vector

Recurrent **Layer / Cell**

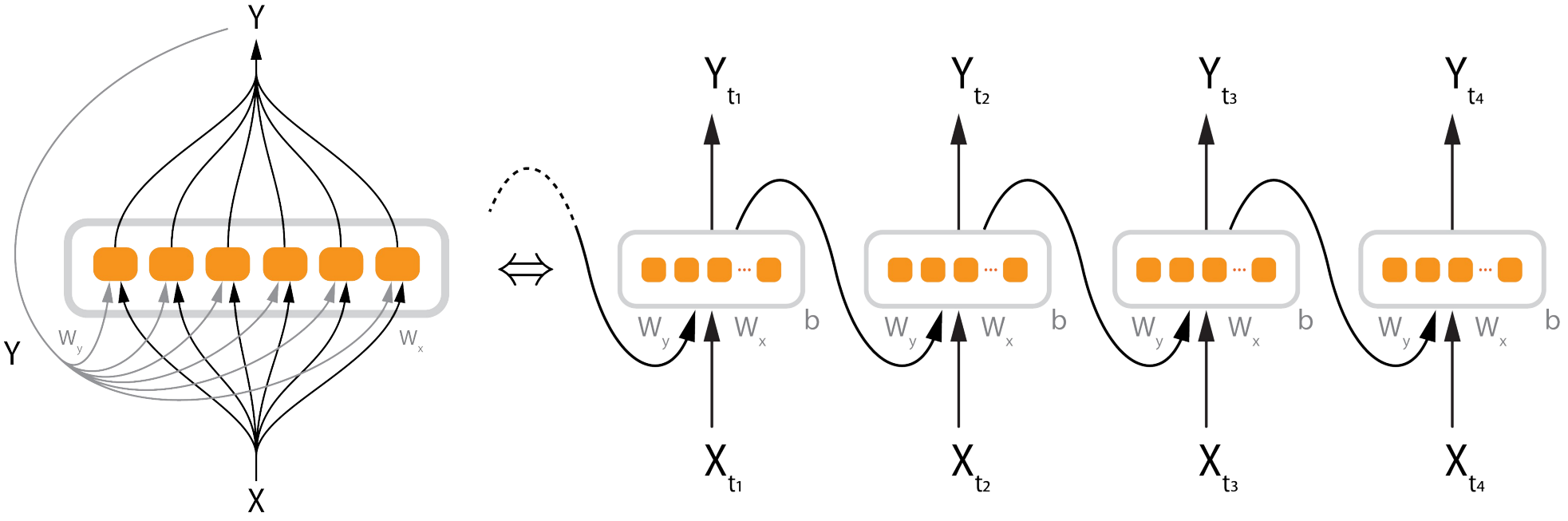


$$Y_{(t)} = \Phi\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right)$$

W_y : is a tensor
W_x : is a tensor
b : is a vector
Y : is a vector

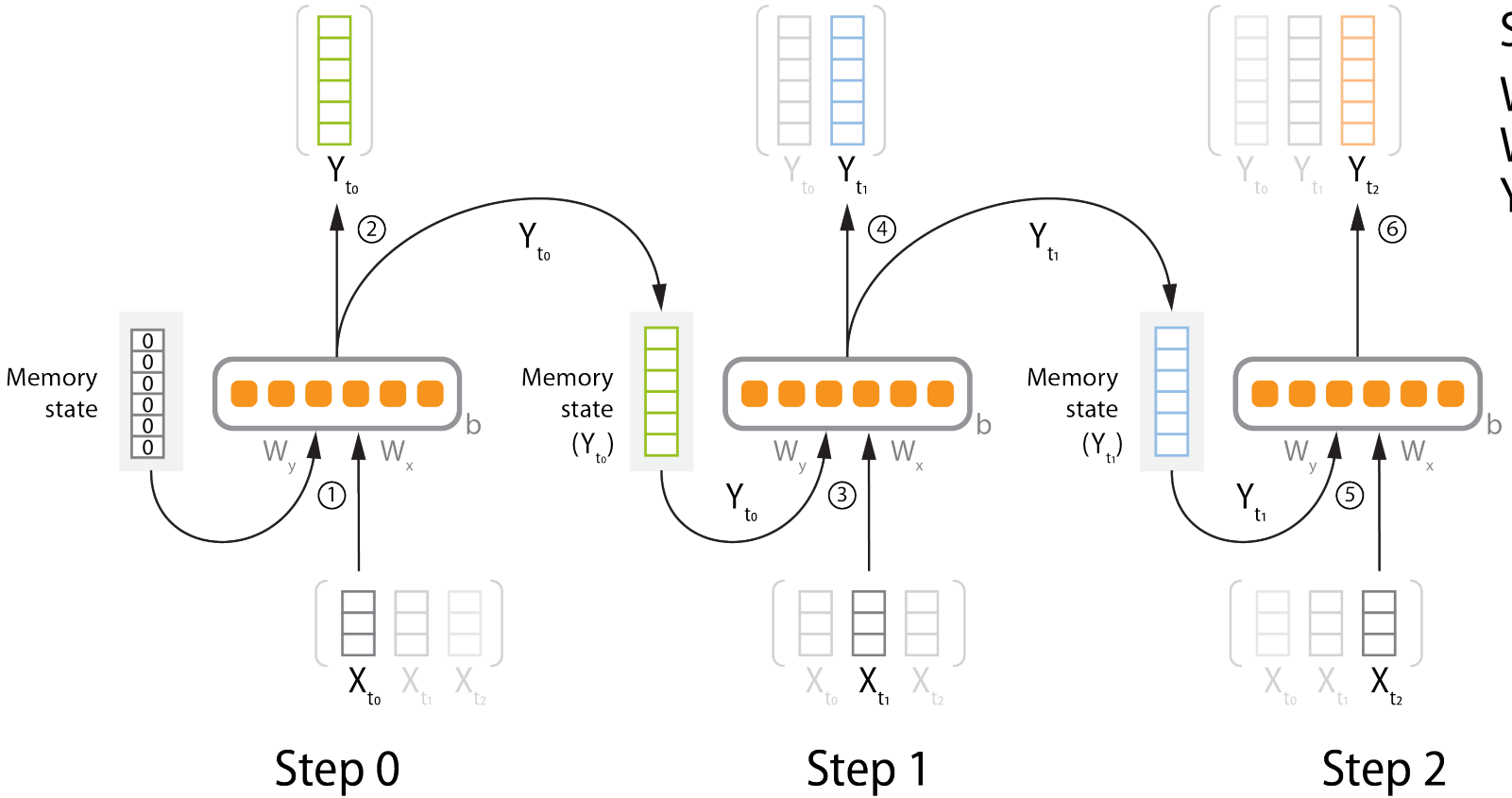
$\mathbf{W_y}$: is a tensor
$\mathbf{W_x}$: is a tensor
$\mathbf{b}$: is a vector
$\mathbf{Y}$: is a vector

Recurrent **Layer / Cell**



$$Y_{(t)} = \Phi\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right)$$

Recurrent Layer / Cell



Shapes :

W_x : (nb_{units}, X size)

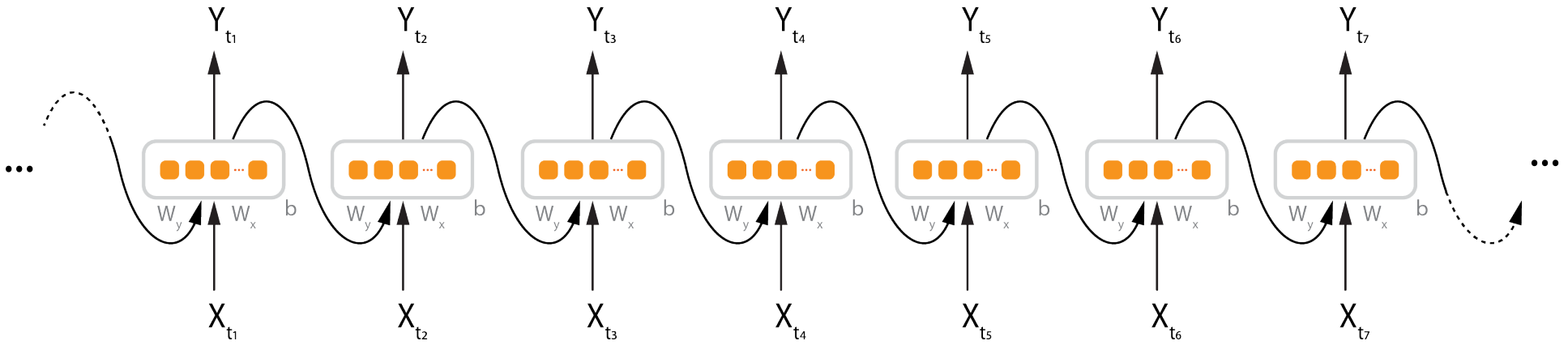
W_y : (nb_{units}, nb_{units})

Y : (nb_{units})

$$Y_{(t)} = \Phi\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right)$$

Simple RNN limits...

But....

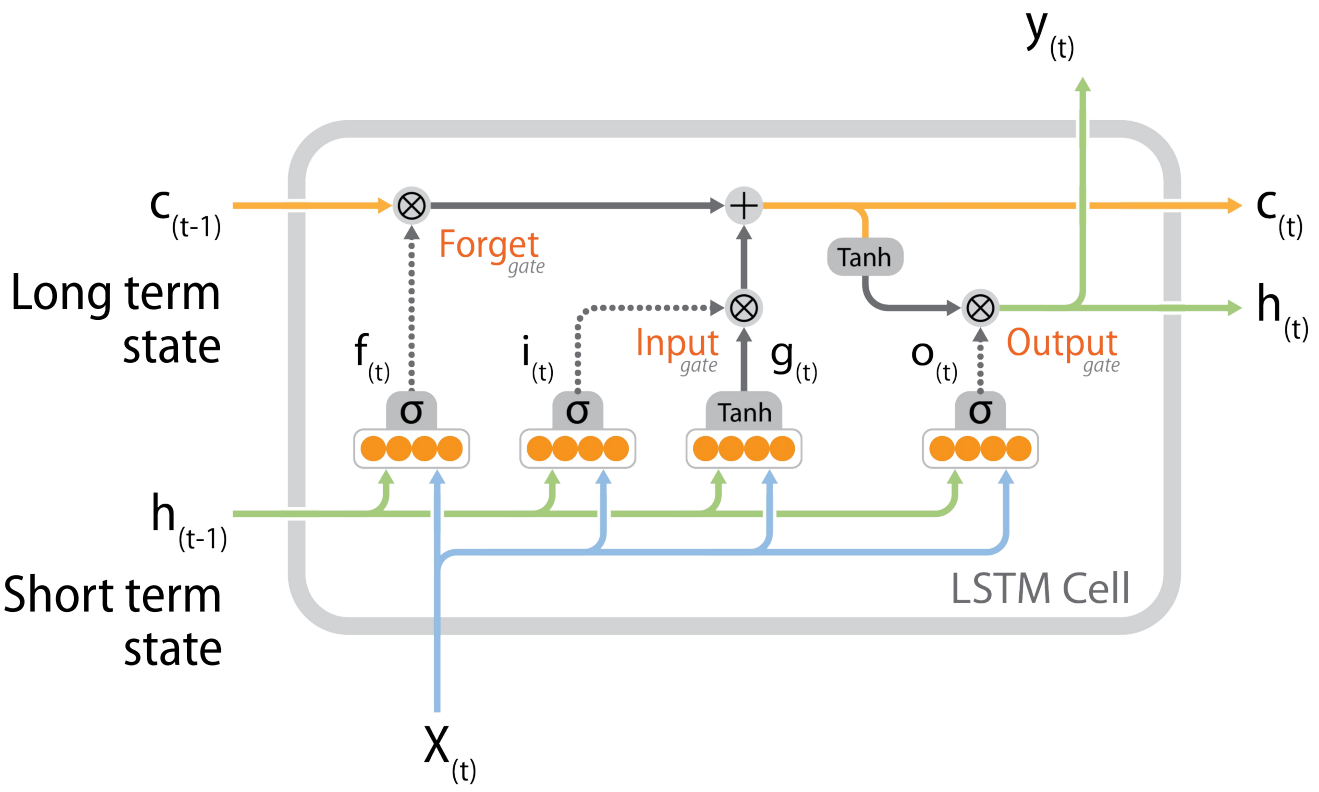


Slow convergence,
Short memory,
Vanishing / exploding gradients

⇒

In short...
it **doesn't work !**

Long Short-Term Memory (LSTM)



$$f_{(t)} = \sigma(W_{xf}^T X_{(t)} + W_{hf}^T h_{(t-1)} + b_f)$$
$$i_{(t)} = \sigma(W_{xi}^T X_{(t)} + W_{hi}^T h_{(t-1)} + b_i)$$
$$g_{(t)} = \tanh(W_{xg}^T X_{(t)} + W_{hg}^T h_{(t-1)} + b_g)$$
$$o_{(t)} = \sigma(W_{xo}^T X_{(t)} + W_{ho}^T h_{(t-1)} + b_o)$$
$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)}$$
$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh(c_{(t)})$$

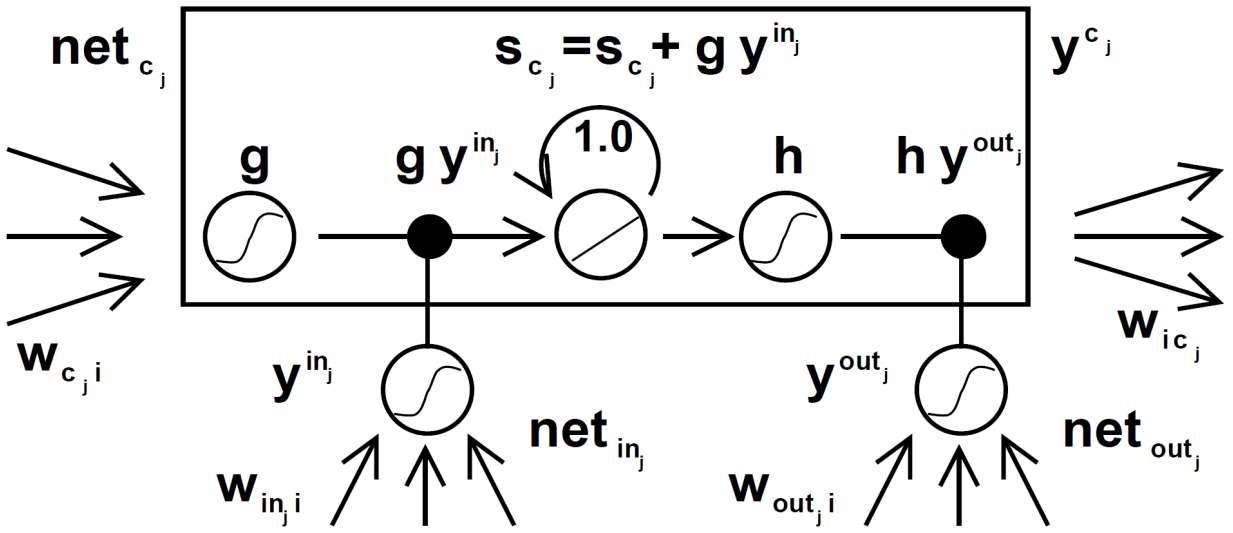
with :

$X_{(t)} \in \mathbb{R}^d$	input vector
$f_{(t)} \in \mathbb{R}^h$	forget gate's activation vector
$i_{(t)} \in \mathbb{R}^h$	input gate's activation vector
$o_{(t)} \in \mathbb{R}^h$	output gate's activation vector
$g_{(t)} \in \mathbb{R}^h$,	current entry vector
$h_{(t)}, y_{(t)} \in \mathbb{R}^h$	hidden state or output vector
$c_{(t)} \in \mathbb{R}^h$	cell state vector
$\otimes$	Hadamard product
σ	sigmoid function
W_k	weights matrix
b_k	bias vector

Long Short-Term Memory
Sepp Hochreiter, Jürgen Schmidhuber, 1998
https://doi.org/10.1162/neco.1997.9.8.1735

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, 2014
https://doi.org/10.48550/arXiv.1406.1078

Long Short-Term Memory (LSTM)



$$
\begin{aligned}
f_{(t)} &= \sigma(W_{xf}^T X_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \\
i_{(t)} &= \sigma(W_{xi}^T X_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \\
g_{(t)} &= \tanh(W_{xg}^T X_{(t)} + W_{hg}^T h_{(t-1)} + b_g) \\
o_{(t)} &= \sigma(W_{xo}^T X_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \\
c_{(t)} &= f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \\
y_{(t)} &= h_{(t)} = o_{(t)} \otimes \tanh(c_{(t)})
\end{aligned}
$$

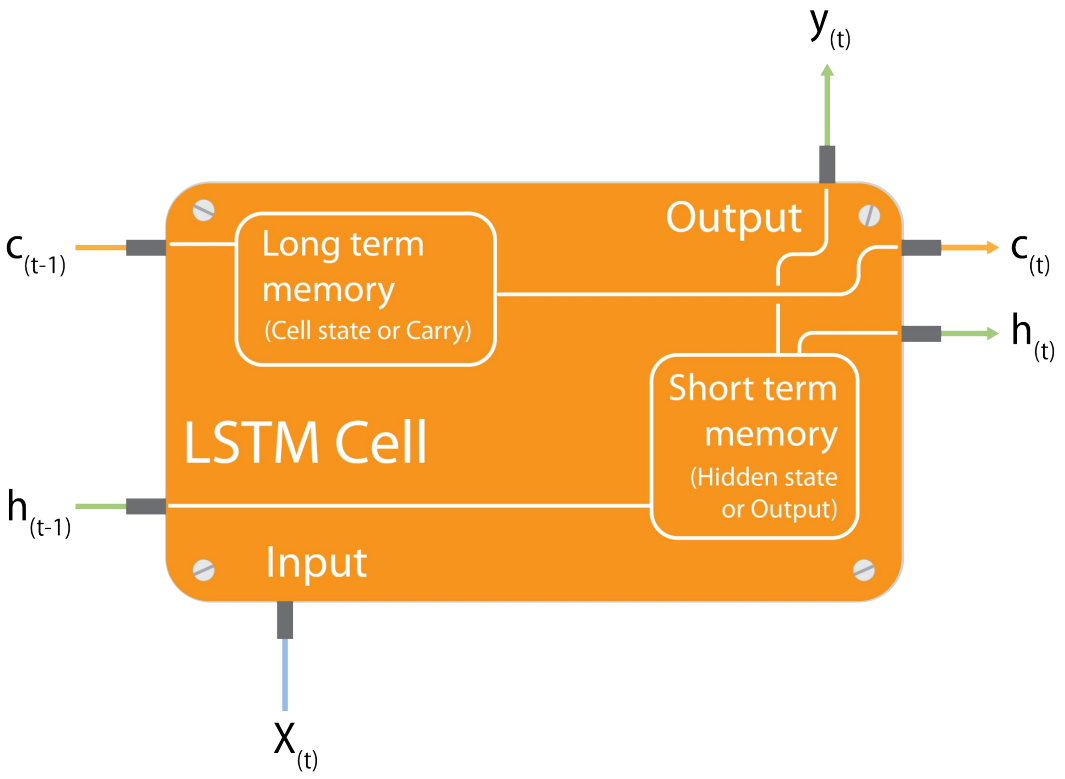
with :

$X_{(t)} \in \mathbb{R}^d$	input vector
$f_{(t)} \in \mathbb{R}^h$	forget gate's activation vector
$i_{(t)} \in \mathbb{R}^h$	input gate's activation vector
$o_{(t)} \in \mathbb{R}^h$	output gate's activation vector
$g_{(t)} \in \mathbb{R}^h$,	current entry vector
$h_{(t)}, y_{(t)} \in \mathbb{R}^h$	hidden state or output vector
$c_{(t)} \in \mathbb{R}^h$	cell state vector
$\otimes$	Hadamard product
σ	sigmoid function
W_k	weights matrix
b_k	bias vector

Long Short-Term Memory
Sepp Hochreiter, Jürgen Schmidhuber, 1998
https://doi.org/10.1162/neco.1997.9.8.1735

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, 2014
https://doi.org/10.48550/arXiv.1406.1078

Long Short-Term Memory (LSTM)



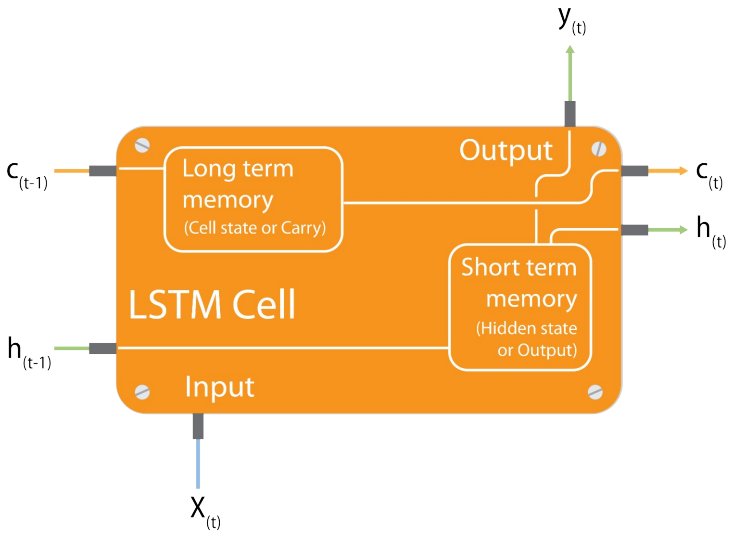
Keras

PyTorch

Long Short-Term Memory
Sepp Hochreiter, Jürgen Schmidhuber, 1998
https://doi.org/10.1162/neco.1997.9.8.1735

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, 2014
https://doi.org/10.48550/arXiv.1406.1078

Long Short-Term Memory (LSTM)



Serie to vector

```
input   = random.normal([32, 20, 8])

lstm    = keras.layers.LSTM(16)
output  = lstm(input)
```

```
Inputs shape is :  (32, 20, 8)
Output shape is :  (32, 16)
```

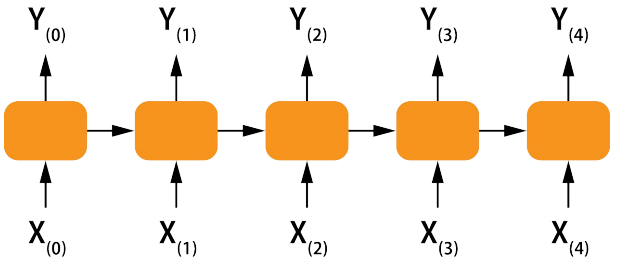
Serie to vector

```
lstm = keras.layers.LSTM(18, return_sequences=True, return_state=True)
output, memory_state, carry_state = lstm(input)
```

```
Output shape :  (32, 20, 18)
Memory state :  (32, 18)
Carry  state :  (32, 18)
```

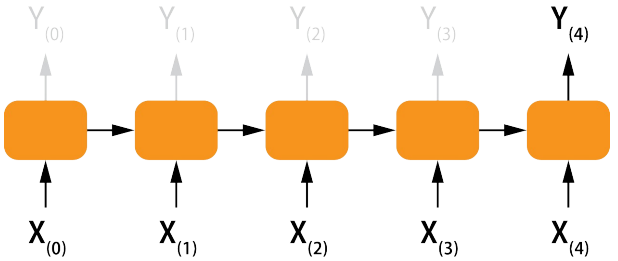
More about :
https://keras.io/api/layers/recurrent_layers/lstm/

Reccurent Neural Network (RNN)



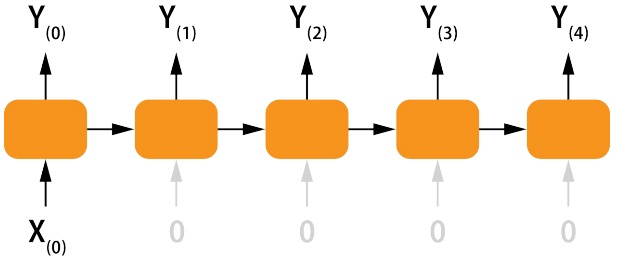
Serie → serie

Example : Time serie prediction



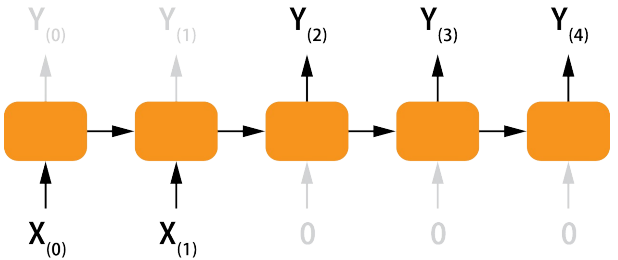
Serie → vector

Example : Sentiment analysis



Vector → serie

Example : Image annotation



Encoder → decoder

Example : Language Translation

How to predict a sequence ?

Known sequence :

$[X_1, X_2, X_3, X_4, \ldots, X_n, X_{n+1}]$

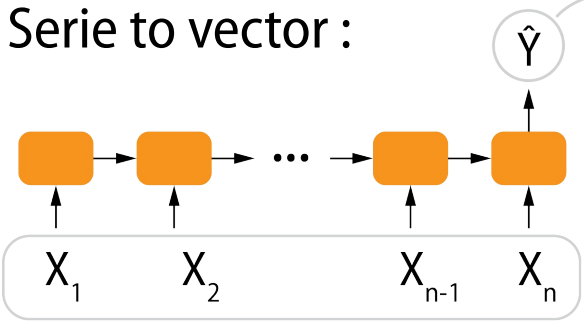
Input Sequence : $X = [X_1, X_2, \ldots, X_n]$

Expected output : $Y = [X_{n+1}]$

Objective :

Train our RNN network to **predict** the **n+1** vector of our input sequence :

Serie to vector :



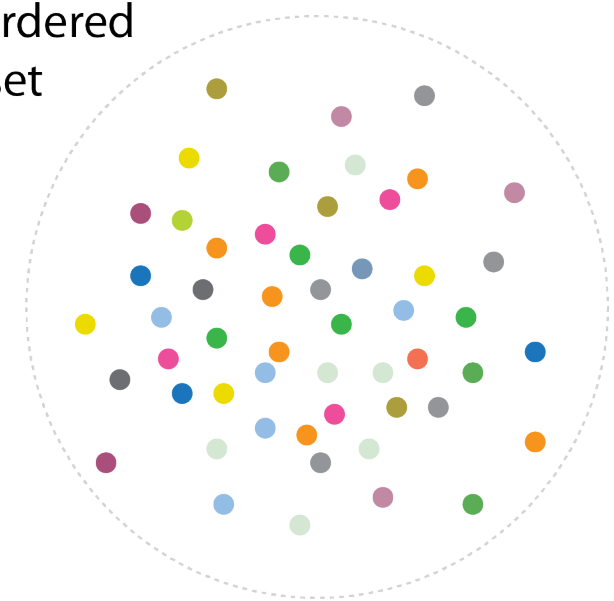
Predicted output que l'on comparera à X_{n+1}

Input sequence $[X_1, X_2, ..., X_{n-1}, X_n]$

Preparation of sequence data

The distribution of data between trains and test series must be comparable.

Not ordered
Dataset



Shuffle

Train set

Test set

Preparation of sequence data

Can the past
explain the future ?

Ordered dataset

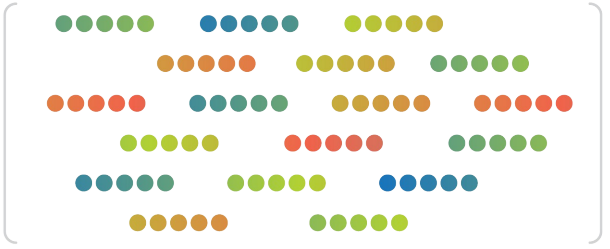
Preparation of sequence data

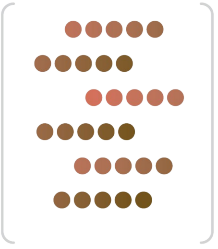
Ordered dataset

Past | Future

Train data

Test data

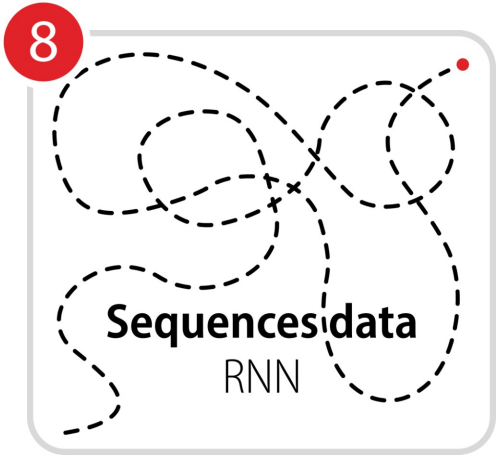




Train set

Test set

Note that a data generator should be very useful !

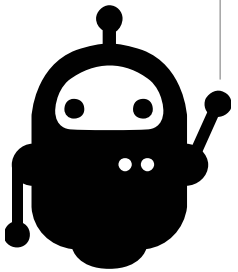


Part 2

1. **Sequences** data
   Recurrent Neural Network
   LSTM and GRU
2. Example 1 : LadyBug
   → Prediction of a virtual trajectory

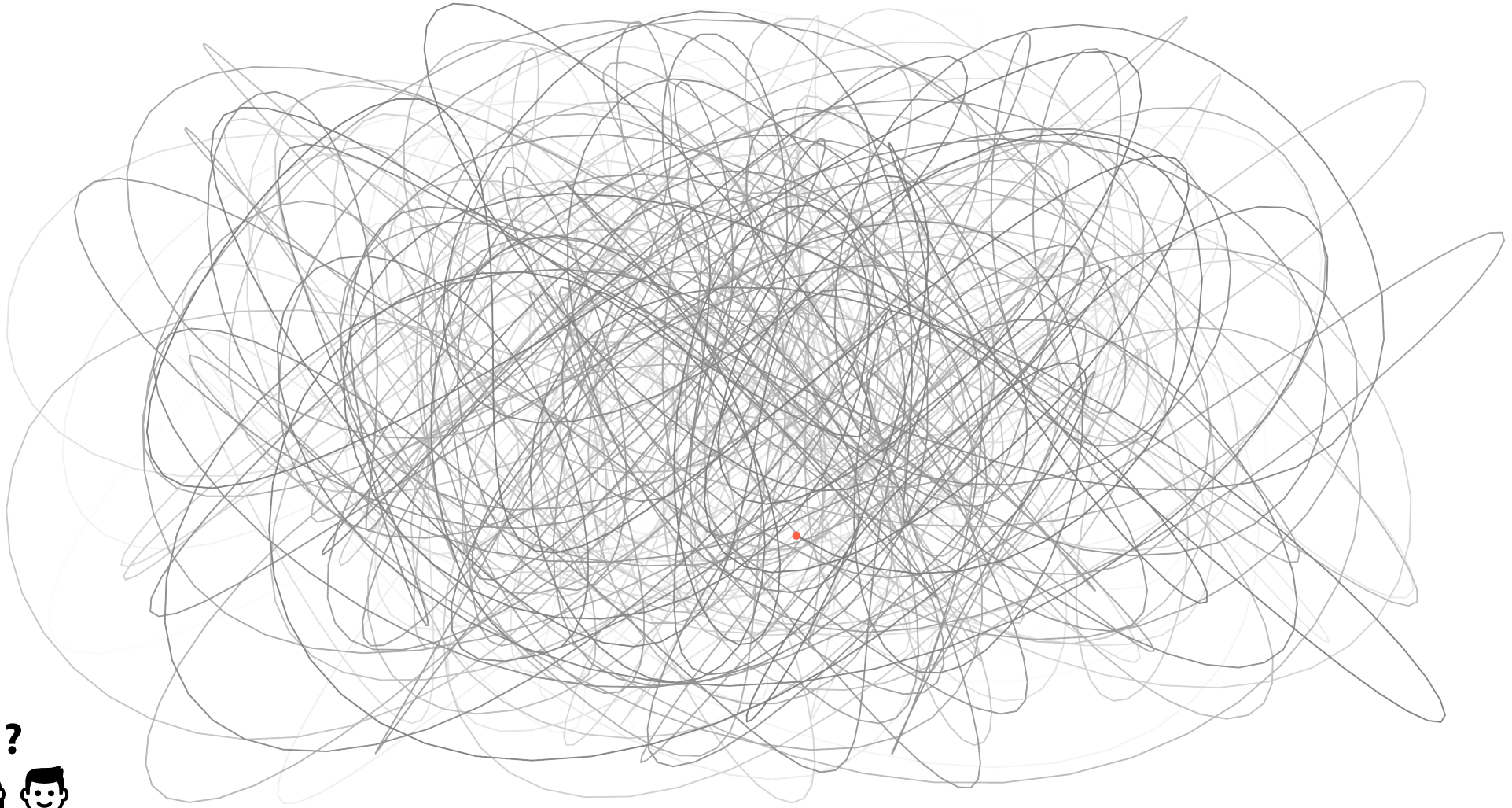
Time series with RNN

Notebook : [LADYB1]



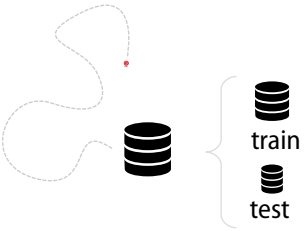
Objective :
Prediction of a 2D ladybug trajectory with a RNN

Dataset :
Generated trajectory



LADYB1

<60 s



Import and init

Generation of a dataset

Compile and train

START

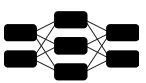
1 → 2 → 3 → 4 → 5 → END

Parameters

```
sequence_len = 20
predict_len  = 5

Scale        = 0.2
train_prop   = .8
batch_size   = 32
epochs       = 5
```

Create a model



Try a trajectory prediction

Objectives :

Prediction of a 2D ladybug* trajectory with a RNN

*Disclaimer : No ladybugs were harmed during the making of this notebook.

Little things and concepts to keep in mind

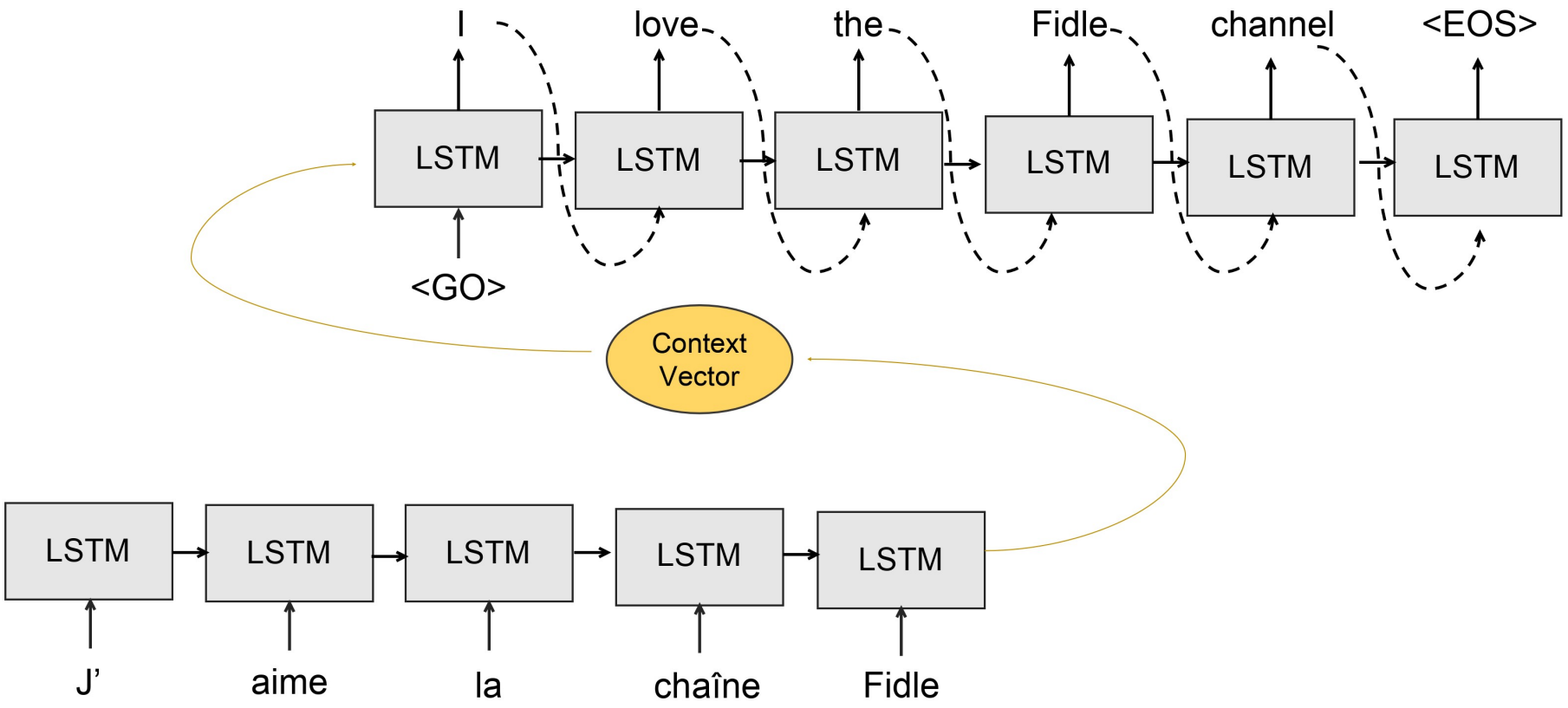
- Can the past explain the future ?
- Understand the data, again and again !
- Beware of overfiting
- Remember that Pandas is good for you !
- The json files are cool, too
- Preparing your data can cost 70% of the work
- Think about data generators
- Matplotlib are also very good for you !
- There is a lot of sequential data
- Not everything can uses GPUs...

But the story isn't over...
...and this was just the beginning!

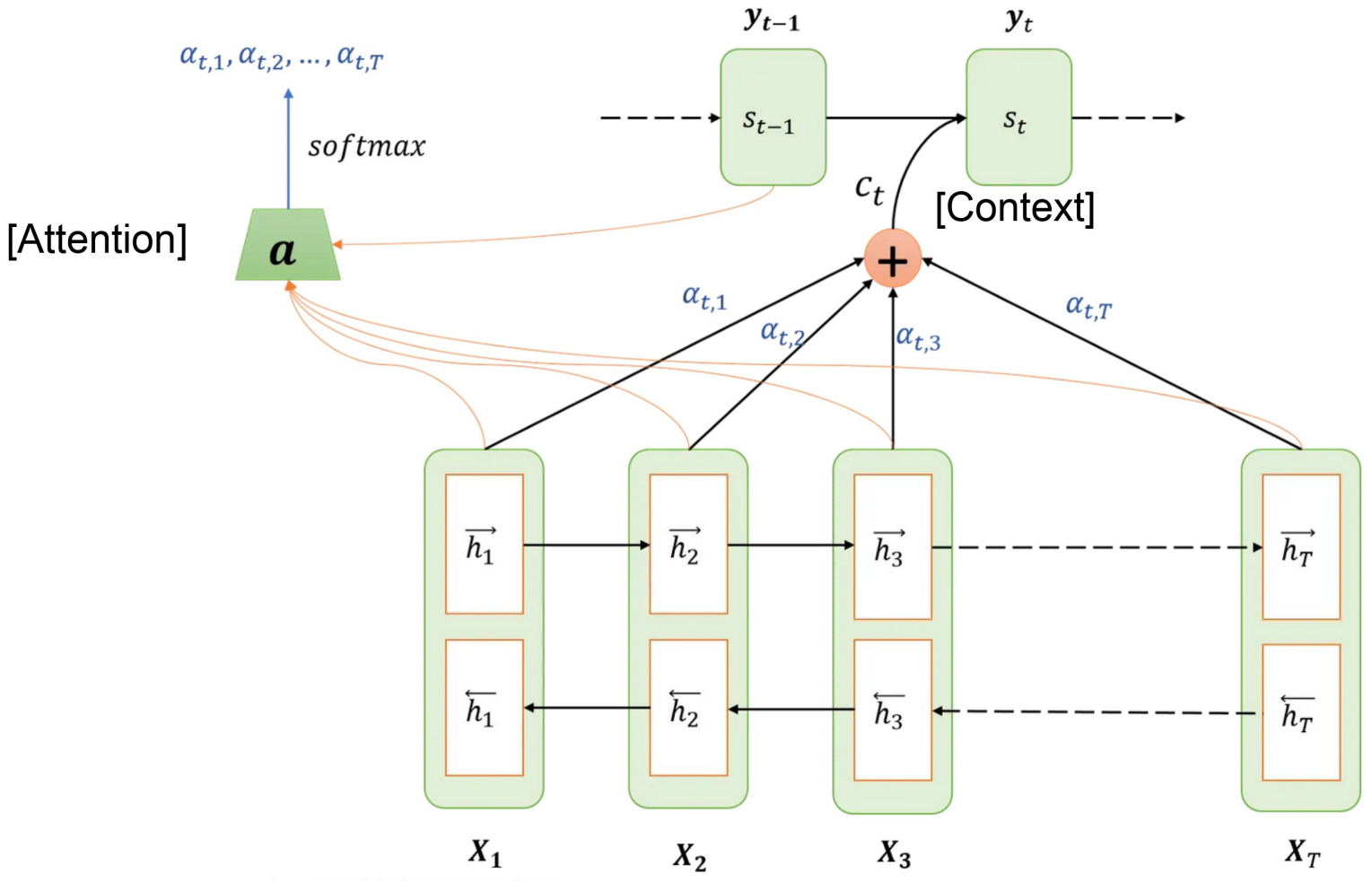
...لكن القصة لم تنته بعد
!و هذه كانت البداية فقط...

하지만 이야기는 끝나지 않았습니다 ...
... 그리고 이것은 시작에 불과합니다 !

Simple translator, sequence to sequence

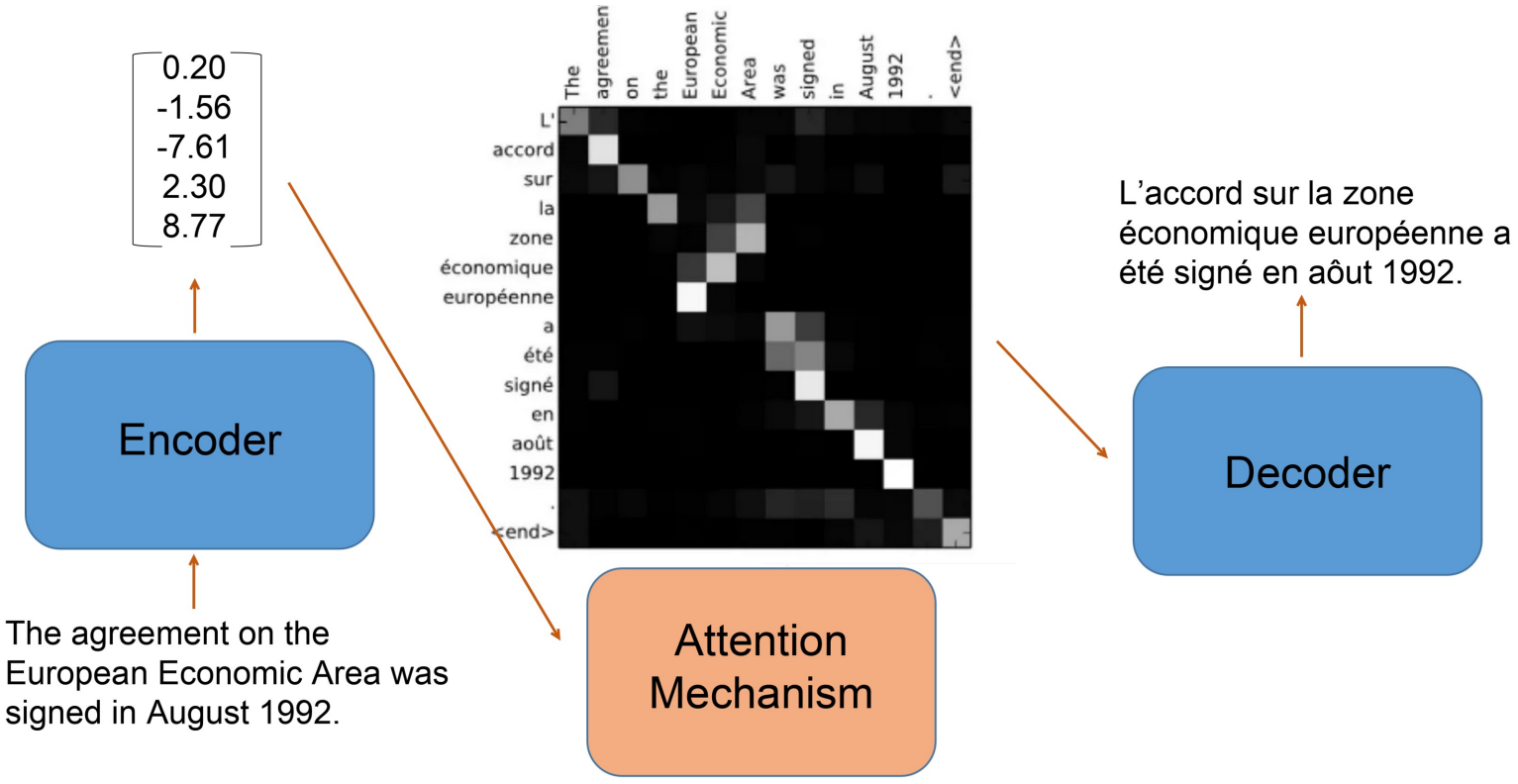


Attention Mechanism



https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b

Attention Mechanism



https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b

FIDLE

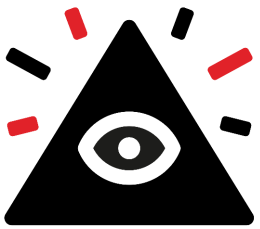
https://youtube.com/@CNRS-FIDLE
https://fidle.cnrs.fr

Jeudi
08
Février
à 14h00

Next, on Fidle :

L'IA comme un outil

9



«Attention is All You Need»

Transformers

Réalisation, technique et mise en scène :
Agathe, Baptiste et Yanis - UGA/DAPI
Thibaut, Kamel - IDRIS