SAISON 22/23

Formation

Introduction Deep Learning

Séquence 05b

Données creuses / textuelles (Embedding)

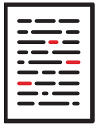
FIDLE

https://fidle.cnrs.f r

MIAI Grenoble Alpes
cnrs
UGA Université Grenoble Alpes
EFELIA - MIAI
ECOLE FRANÇAISE DE L'INTELLIGENCE ARTIFICIELLE
DEVLOG RÉSEAU
Resinfo
sari
ANITI
GRICAD
SIMaP

Sparse data – Case of textual data

5b



Sparse data (text)
Embedding

5.1 **Text encoding**

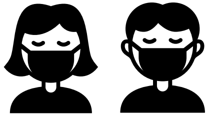
- → From text to tensor
- → One hot encoding
- → Embedding

5.2 **Example 1 : IMDB1**

- → Sentiment analysis with one-hot encoding

5.3 **Example 2 : IMDB2/3/4**

- → Sentiment analysis with embedding



Sparse data – Case of textual data

5b

Sparse data (text)
Embedding

5.1 **Text encoding**
- → From text to tensor
- → One hot encoding
- → Embedding

!

Please note that this is an extremely large subject and this is only an introduction !

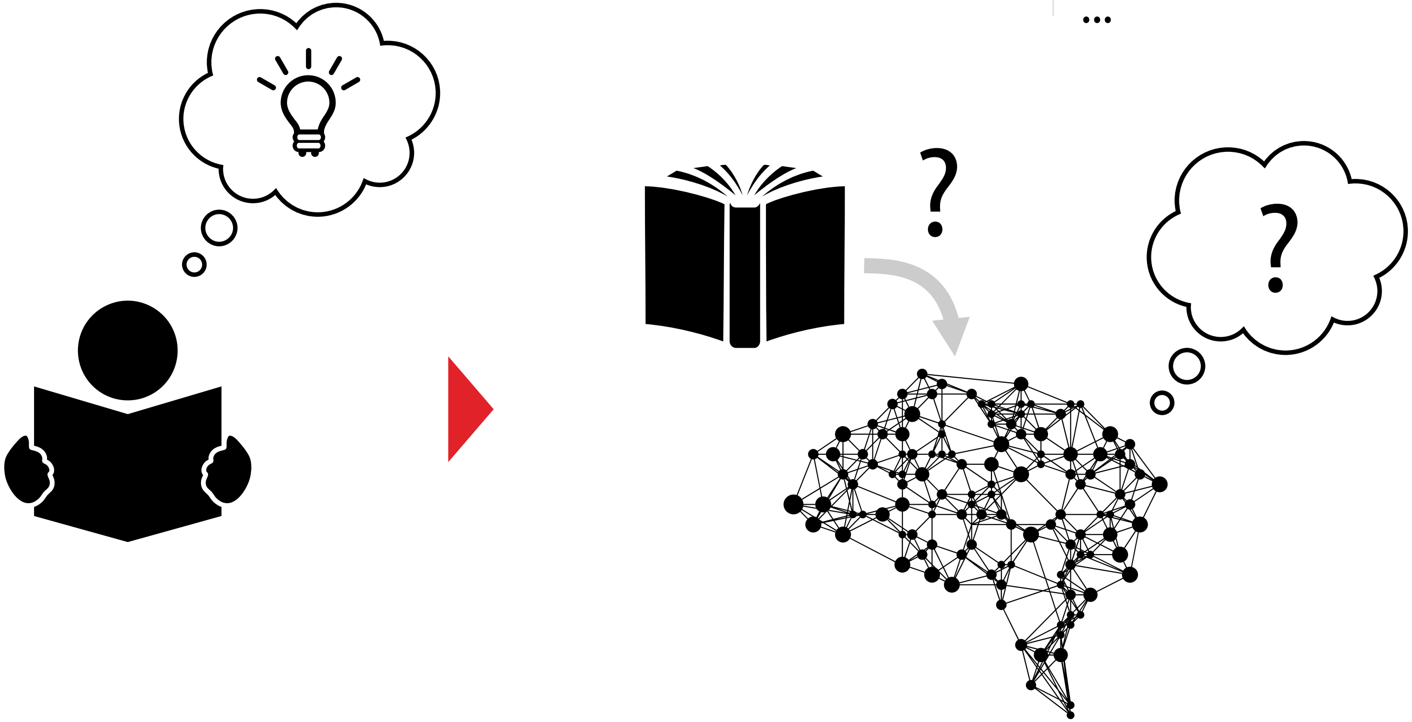
And mostly outdated, because progress is very fast ;-)

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

5.3 **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. »

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

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

Tokenization (words)

Vectorization

Dictionary

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

! 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

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Tokenization (words)

Vectorization

001000

One hot encoding

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

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

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
$$

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

$$\begin{pmatrix} 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ &&&\dots&&&&\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

Need :
24 10^6
Parameters !

2 answers :
- One hot vector
- Embedding

One hot vector encoding

Solution 1 : one hot vector

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & & & & \dots & & & \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \xrightarrow[\text{on each line}]{\text{Logical OR}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ \dots \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

1 This word is not present in the sentence

0 This word is 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.

Embedding

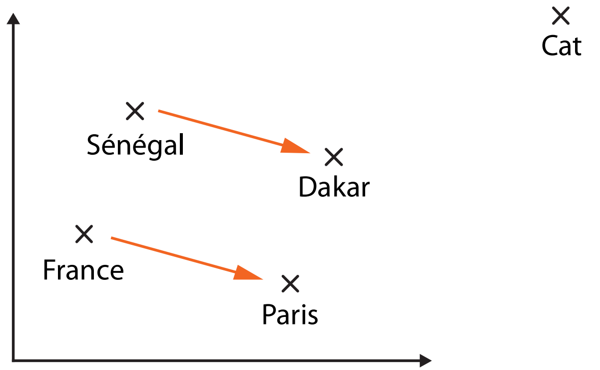
Solution 2 : Using embedding

$$(\text{Word}_i)\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\\dots\\0\\0\\0\\0\\0\end{pmatrix} \quad \blacktriangleright \quad (W_i)\begin{bmatrix}0.7\\4.2\\1.6\end{bmatrix}$$

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

$$(\text{Word}_i) \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \blacktriangleright \quad (W_i) \begin{bmatrix} 0.7 \\ 4.2 \\ 1.6 \end{bmatrix}$$

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

Encoded sentences (padded)

Tensor 2D

(batch size, k)
k = Sentence length

1	11	2	11	4	23	18	0	0	0	0	0	0	0	0
1	13	2	6	2	2	6	10	0	0	0	0	0	0	0
…														
1	13	43	2	2	0	0	0	0	0	0	0	0	0	0
…														
1	17	2	6	13	9	21	7	11	0	0	0	0	0	0
1	14	22	62	2	2	17	4	19	1	0	0	0	0	0

```
keras.layers.Embedding( input_dim    = m,
                        output_dim   = n,
                        input_length = k )
```

Keras

Dictionary of dense vectors (weight)

Train

W_0	W_1	W_2	W_3	…	W_j	…	W_m
$w_{0,0}$	$w_{0,1}$	$w_{0,2}$	$w_{0,3}$	…	$w_{0,j}$	…	$w_{0,m}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	…	$w_{1,j}$	…	$w_{1,m}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	…	$w_{2,j}$	…	
…	…	…	…	…	…	…	
$w_{i,0}$	$w_{i,1}$	$w_{i,2}$	$w_{i,3}$	…	$w_{i,j}$	…	$w_{i,m}$
…	…	…	…	…	…	…	
$w_{n,0}$	$w_{n,1}$	$w_{n,2}$	$w_{n,3}$	…	$w_{n,j}$	…	$w_{n,m}$

(n, m)
n = Dense vector size
m = Dictionary size

Embedded sentences

Tensor 3D

(batch size, k, n)
k = Sentence length
n = Dense vector size

W_1	W_{11}	W_2	W_{11}	W_4	W_{23}	W_{18}	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0
W_1	W_{13}	W_2	W_6	W_2	W_2	W_6	W_{10}	W_0	W_0	W_0	W_0	W_0	W_0	W_0
…														
W_1	W_{13}	W_{43}	W_2	W_2	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0
…														
W_1	W_{17}	W_2	W_6	W_{13}	W_9	W_{21}	W_7	W_{11}	W_0	W_0	W_0	W_0	W_0	W_0
W_1	W_{14}	W_{22}	W_{62}	W_2	W_2	W_{17}	W_4	W_{19}	W_1	W_0	W_0	W_0	W_0	W_0

(k, n)
k = Sentence length
n = Dense vector size

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

Keras embedding layer

Embedded sentences

Tensor 3D

(batch size, k, n)
k = Sentence length
n = Dense vector size

W_1	W_{11}	W_2	W_{11}	W_4	W_{23}	W_{18}	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0
W_1	W_{13}	W_2	W_6	W_2	W_2	W_6	W_{10}	W_0	W_0	W_0	W_0	W_0	W_0	W_0
…														
W_1	W_{13}	W_{43}	W_2	W_2	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0	W_0
…														
W_1	W_{17}	W_2	W_6	W_{13}	W_9	W_{21}	W_7	W_{11}	W_0	W_0	W_0	W_0	W_0	W_0
W_1	W_{14}	W_{22}	W_{62}	W_2	W_2	W_{17}	W_4	W_{19}	W_1	W_0	W_0	W_0	W_0	W_0

(k, n)
k = Sentence length
n = Dense vector size

keras.layers.GlobalAveragePooling1D()   K Keras

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

Where :
k is sentence length
$W_i^{(s)}$ is word vector i of sentence (s)

Dense vector sentences

Tensor 2D

(batch size, n)
n = Dense vector size

$\overline{W}$
$\overline{W}$
…
$\overline{W}$
…
$\overline{W}$
$\overline{W}$

(n,)
n = Dense vector size

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

The order of the words is therefore ignored !

Word2Vec[1]

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... ;-)

[1] Tomas Mikolov & all, (2013), [W3VEC]

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.

(french)

Word2Vec[1]

Approche ayant pour objectif de constituer des **dictionnaires** 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... ;-)

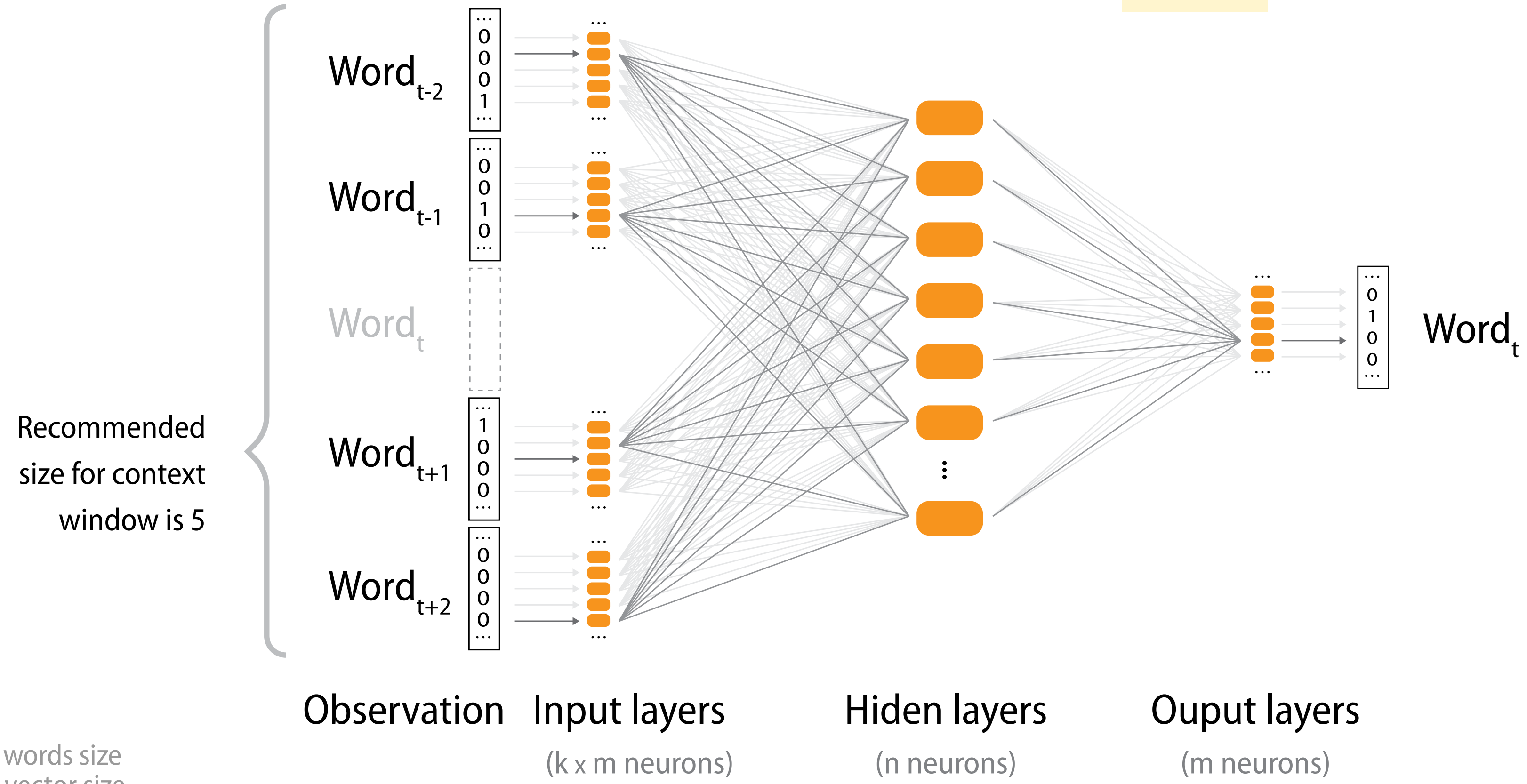
[1] Tomas Mikolov & all, (2013), [W3VEC]

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

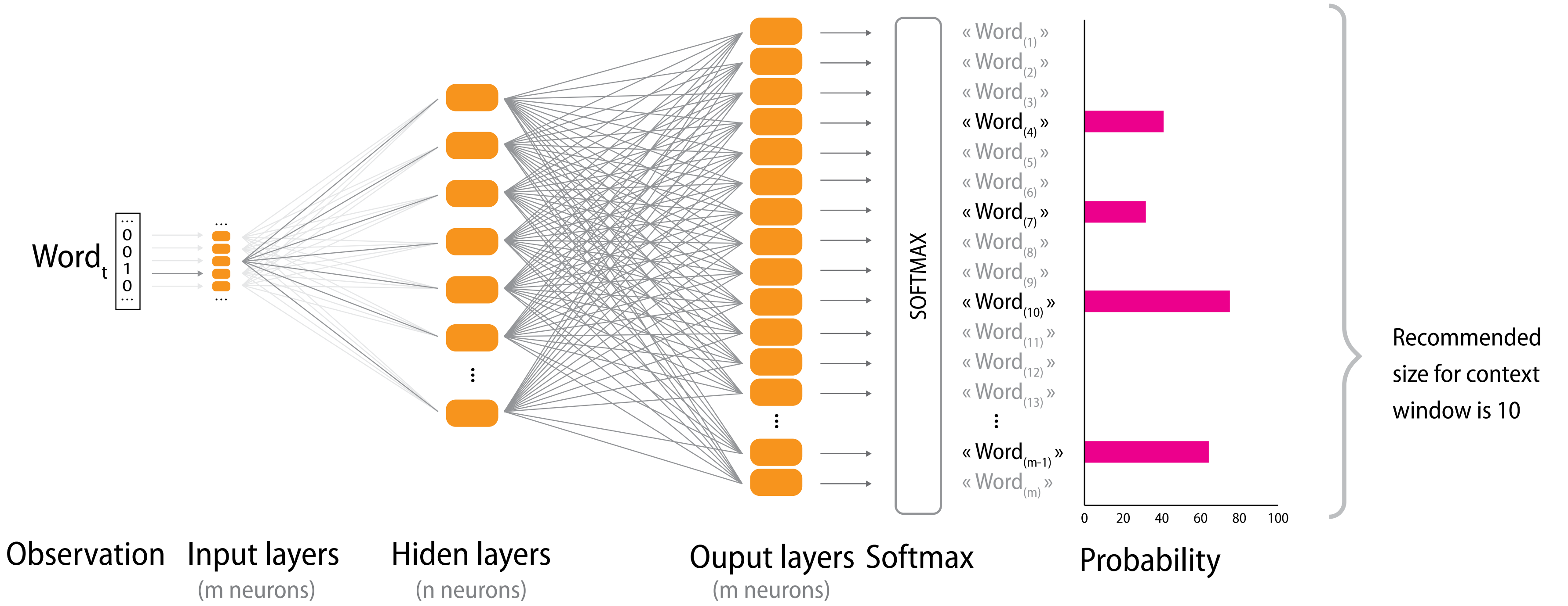
The objective of training is to (re)find a word from its context.



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

GloVe[1]

Contrairement à Word2vec, GloVe ne repose pas uniquement sur des statistiques locales (informations sur le contexte local des mots), mais intègre des **statistiques globales** (co-occurrence des mots) pour obtenir des vecteurs de mots.

[1] Jeffrey Pennington & all, (2014), [GLOVE]
Training is performed on aggregated global word-word co-occurrence statistics

(Flau)BERT[1]

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 !

1 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

2 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

Sparse data – Case of textual data

5b

Sparse data (text)
Embedding

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

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

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

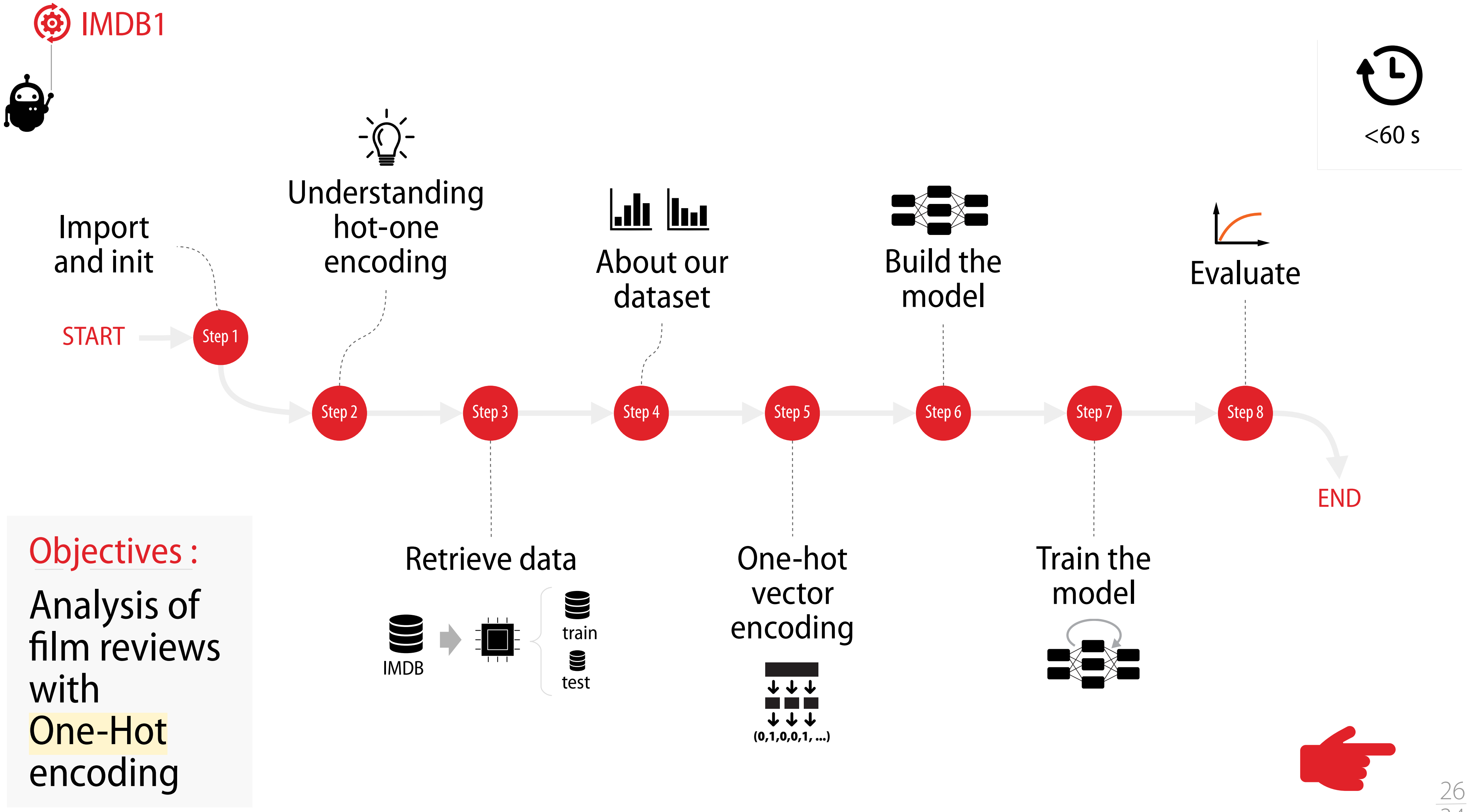
One-Hot encoding with IMDB

Notebook : [IMDB1]

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

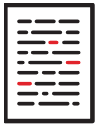
Dataset :
The IMDB dataset is composed of 50,000 film reviews from the site of the same name.

https://www.imdb.com/


IMDB1
<60 s
Import and init
START
Step 1
Understanding hot-one encoding
Step 2
Step 3
Retrieve data
IMDB
train
test
About our dataset
Step 4
Step 5
One-hot vector encoding
(0,1,0,0,1, ...)
Build the model
Step 6
Step 7
Train the model
Evaluate
Step 8
END
Objectives :
Analysis of film reviews with One-Hot encoding

Sparse data – Case of textual data

5b



Sparse data (text)
Embedding

5.1 **Text encoding**
- → From text to tensor
- → One hot encoding
- → Embedding

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

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

Sparse data – Case of textual data

5b

Sparse data (text)
Embedding

5.1 **Text encoding**

- → From text to tensor
- → One hot encoding
- → Embedding

5.2 **Example 1 : IMDB1**

- → Sentiment analysis with one-hot encoding

5.3 **Example 2 : IMDB2/3/4**

- → Sentiment analysis with embedding

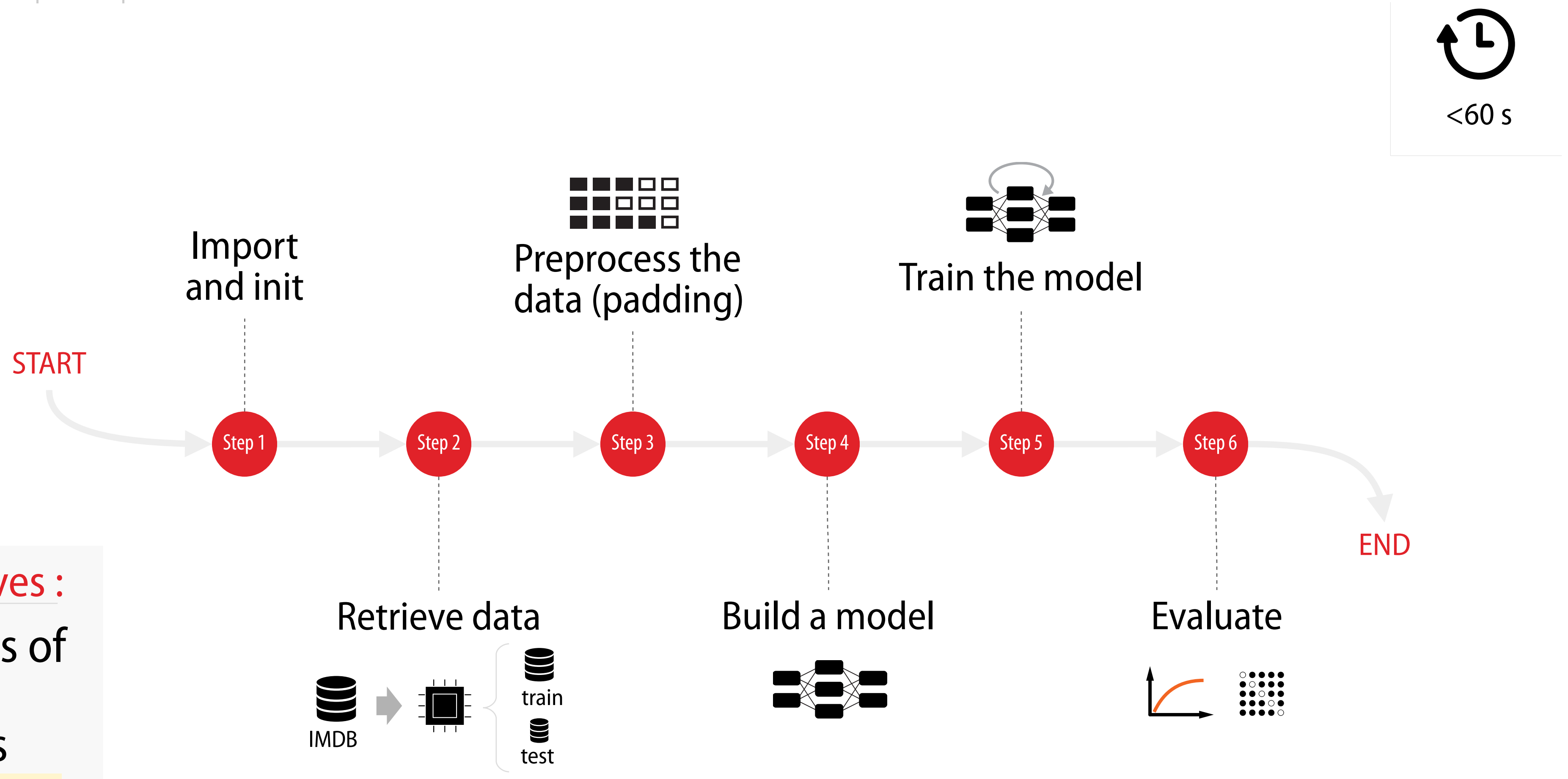
Text embedding with IMDB

Notebook : [IMDB2-4]

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

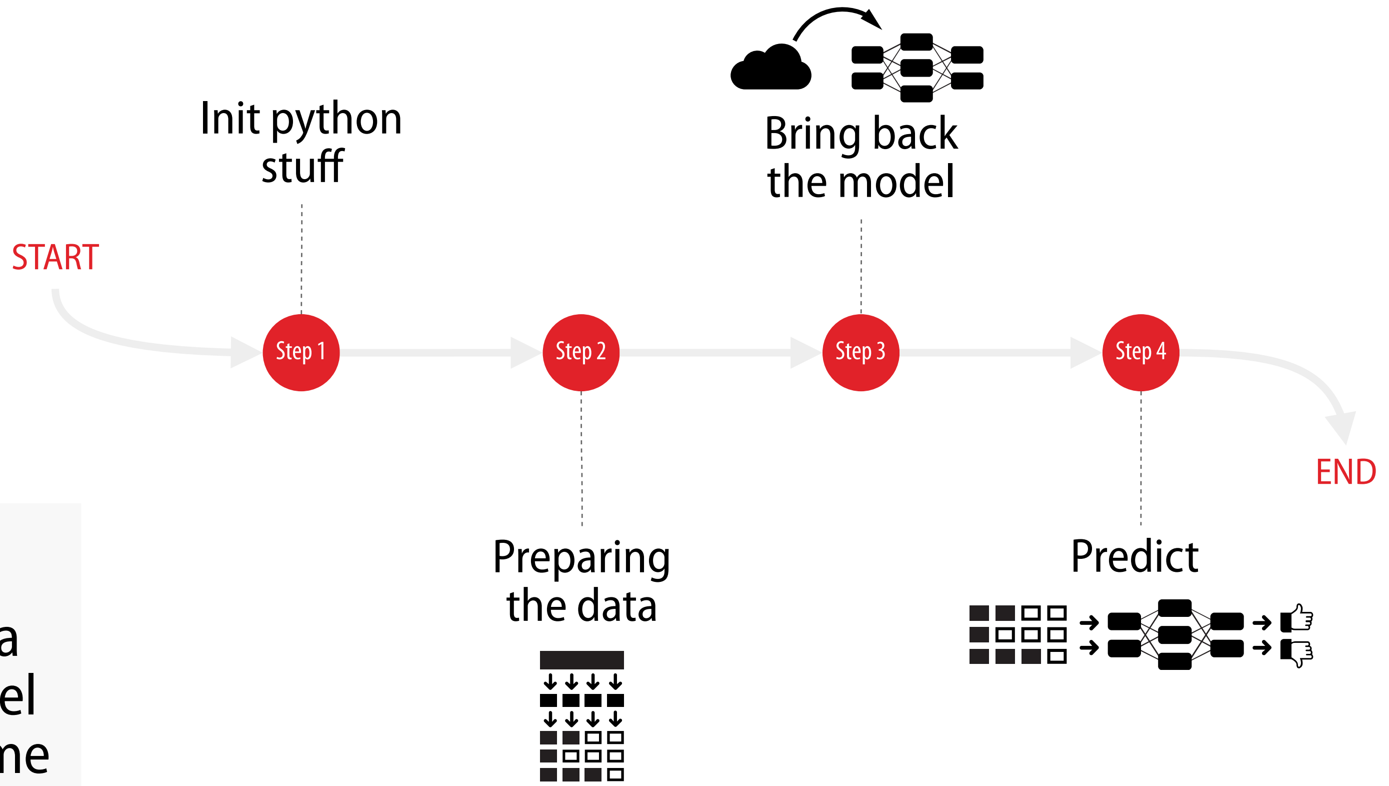
Dataset :
The IMDB dataset is composed of 50,000 film reviews from the site of the same name.

https://www.imdb.com/



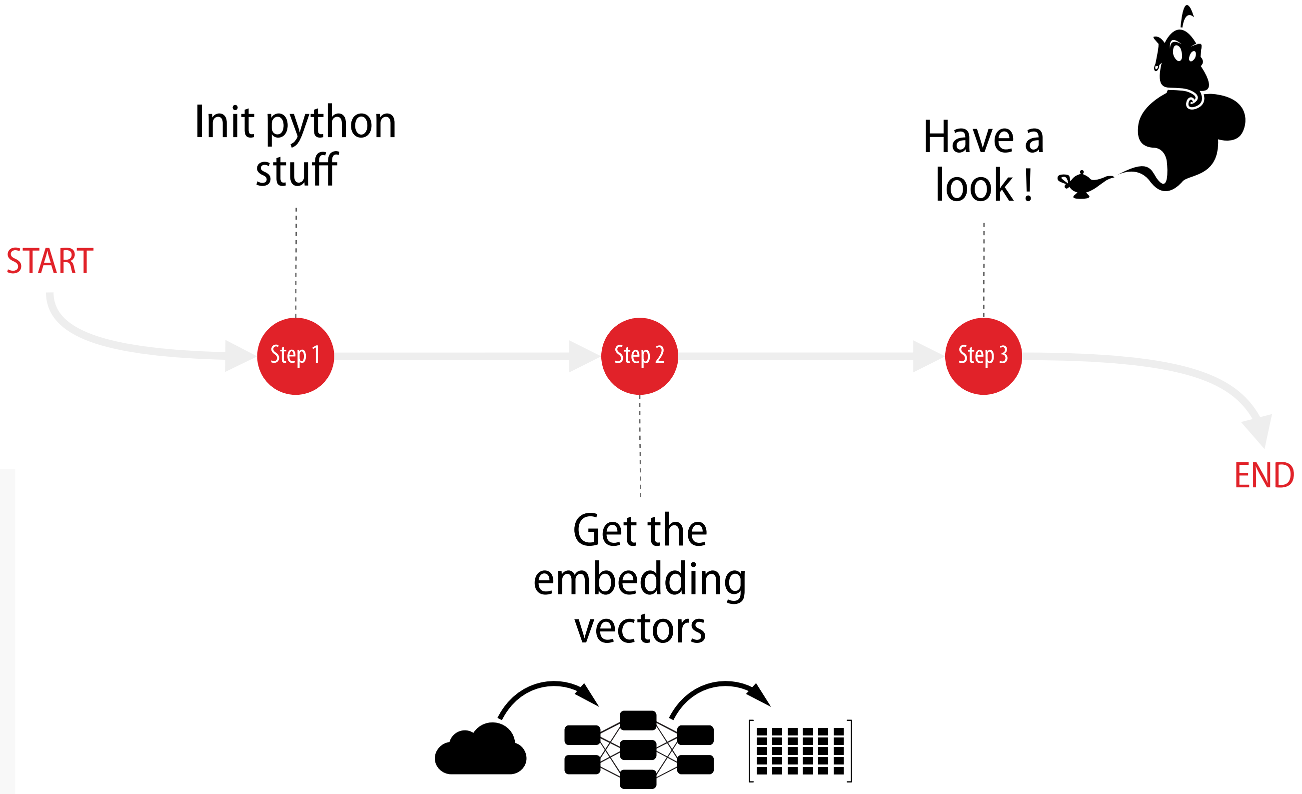
Objectives :

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

Next, on Fidle :

6

Sequences data
RNN

Jeudi 5 janvier,

Épisode 6 :

Quand les données sont des séquences...

Données séquentielles et réseaux récurrents (RNN)
RNN - LSTM - GRU - Spécificités des données séquentielles.

Exemples proposés :

Tentative de prédiction de la trajectoire d'une coccinelle (virtuelle) et de la météorologie, à partir de données réelles, à 3h et 12h.

Durée : 2h00

Next on Fidle :

6

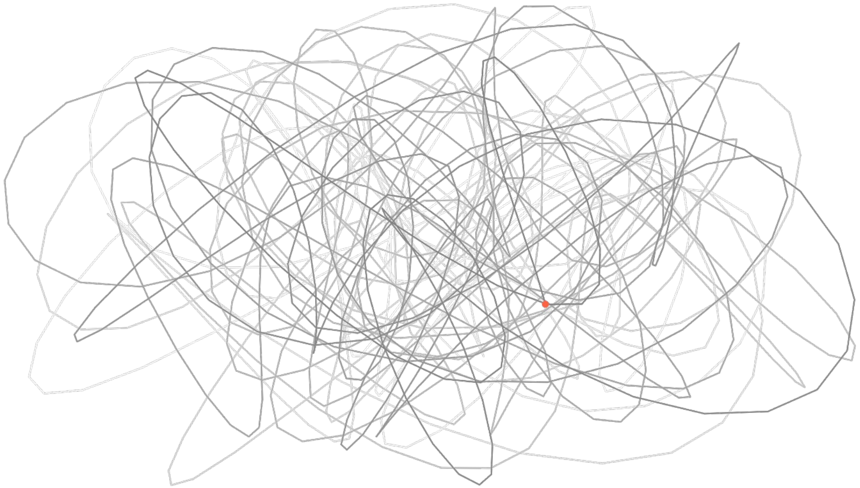
Sequence data

RNN

Jeudi 20 janvier,

Séquence 6 :

**Quand les données sont des séquences…
Données séquentielles et réseaux récurrents (RNN)**



To be continued...

Contact@fidle.cnrs.fr

FIDLE https://fidle.cnrs.fr

YouTube https://fidle.cnrs.fr/youtube

FIDLE