
Word embeddings

Rappel *Embeddings* (pas *Word Embeddings*)

Embedding

```
CLASS torch.nn.Embedding(num_embeddings, embedding_dim, padding_idx=None, max_norm=None,  
                           norm_type=2.0, scale_grad_by_freq=False, sparse=False, _weight=None)
```

[\[SOURCE\]](#)

A simple lookup table that stores embeddings of a fixed dictionary and size.

This module is often used to store word embeddings and retrieve them using indices. The input to the module is a list of indices, and the output is the corresponding word embeddings.

Est une “lookup table”

Formalisme:

- Index d'un mot: w_i
- Table d'embeddings (lookup matrix): \mathbf{V}
- Embedding: \mathbf{e}_i
- $\mathbf{e}_i = \mathbf{V}(w_i)$

Représentation d'un mot

Différentes possibilités:

- Vecteur One-hot
 - *Chat*: [0,0,... 0,**1**,0,0,0,0,0,0,0,0,0,0...]
- Vecteur de contexte
 - *Chat*: [**1**,0,... 0,**0**,0,0,0,0,**1**,0,0,**1**,0,0...]
félin chat litière lait

Vecteurs de contexte

(15.1) A bottle of *tesgüino* is on the table.
Everybody likes *tesgüino*.
Tesgüino makes you drunk.
We make *tesgüino* out of corn.

Vecteurs de contexte

sugar, a sliced lemon, a tablespoonful of	apricot	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	pineapple	and another fruit whose taste she likened
well suited to programming on the digital	computer.	In finding the optimal R-stage policy from
for the purpose of gathering data and	information	necessary for the study authorized in the

Vecteurs de contexte

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Figure 15.4 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Vecteurs de contexte

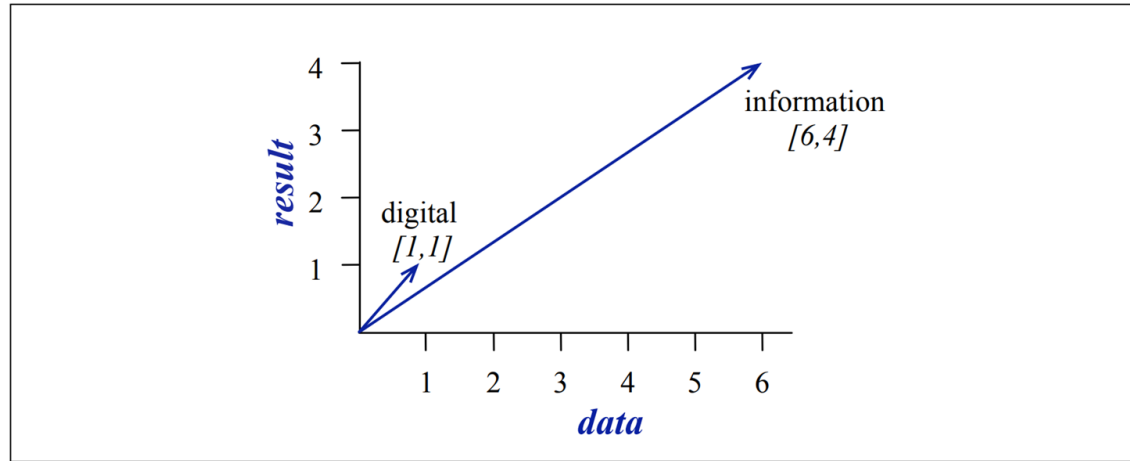


Figure 15.5 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

Vecteurs de contexte

- Vecteurs très grands (taille du vocabulaire)
- Contiennent beaucoup de 0
- **On cherche donc une manière de réduire la dimensionnalité pour:**
 - Efficacité en mémoire
 - Facile d'utilisation pour des classificateurs
 - Moins de paramètres
 - Des dimensions peuvent se recouper

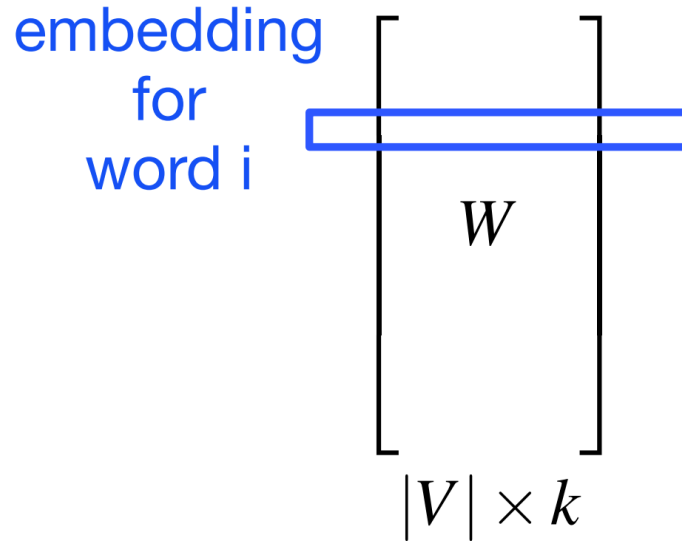
Décomposition en valeurs singulières

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times |V| \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix} \begin{bmatrix} C \\ |V| \times |V| \end{bmatrix}$$

On conserve les top k valeurs singulières

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix} \begin{bmatrix} C \\ k \times |V| \end{bmatrix}$$

On utilise ensuite seulement la matrice W



Méthodes à réseaux de neurones

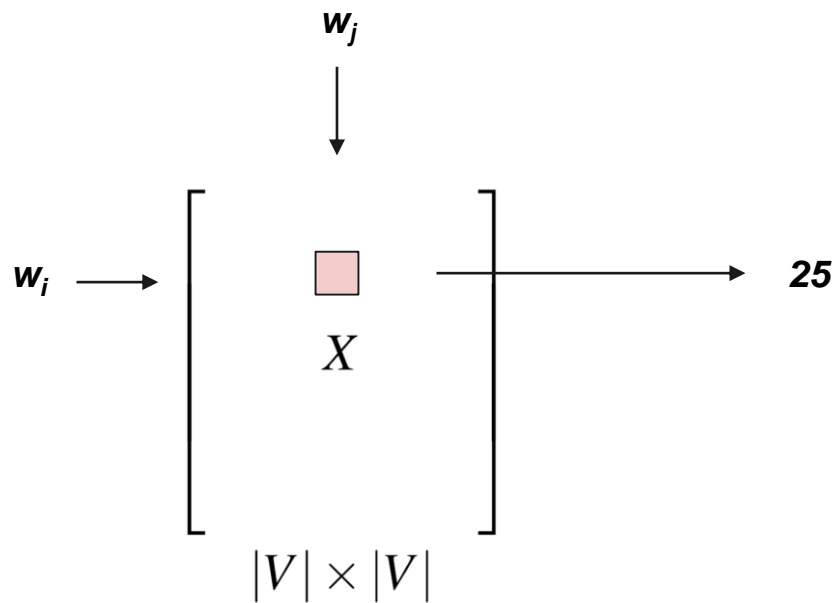
GloVe

“A weighted least squares regression model”

L'idée est de prédire le nombre de co-occurrences X_{ij} (ou le *log*) des mots w_i et w_j

S'apparente à Word2Vec (ou encore FastText)

GloVe



GloVe

$$\text{Perte} = \mathbf{v}(w_i) * \mathbf{v}(w_j) + b_i + b_j - \log(X_{ij})$$

GloVe

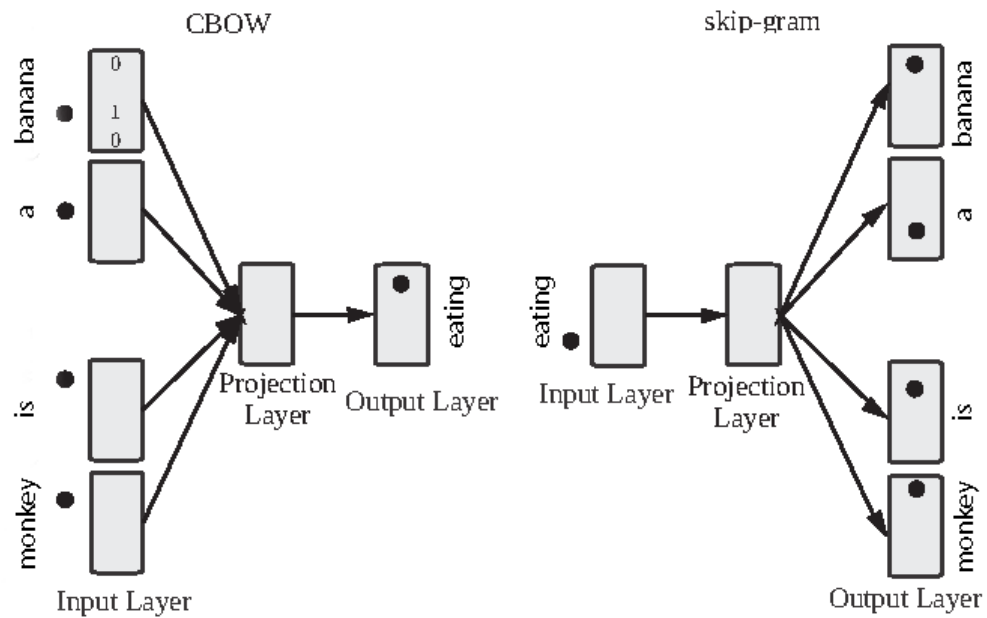
$$\text{Perte} = \mathbf{v}(w_i) * \mathbf{v}(w_j) + b_i + b_j - \log(25)$$

Word2Vec

2 algorithmes:

- Skip-Gram
- CBOW (Contextual Bag of Words)

Word2Vec



CBOW

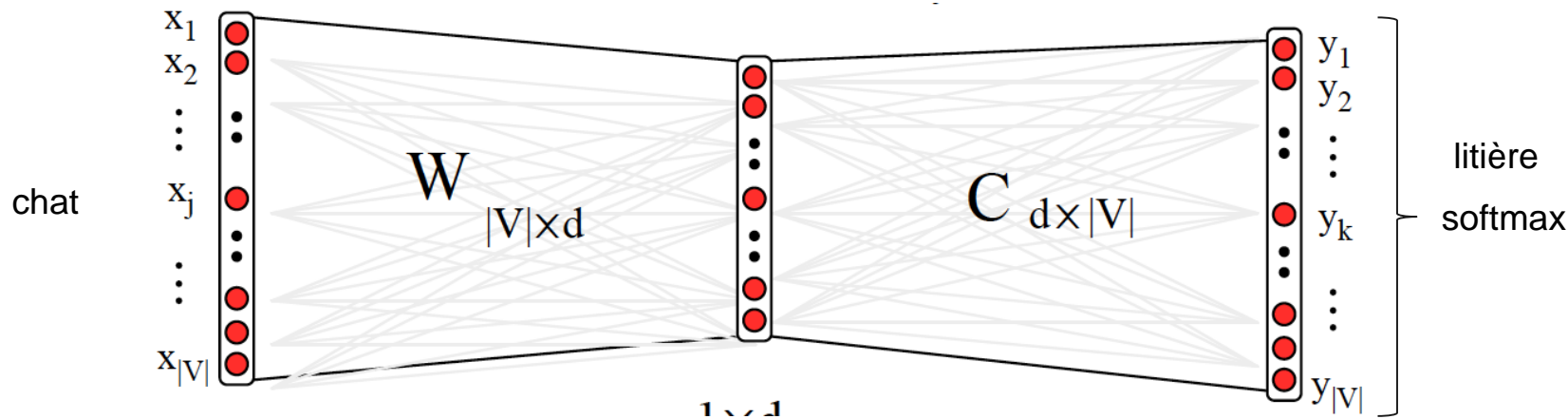
```
class CBOW(nn.Module):
    def __init__(self, vocab_size, embedding_dimension):
        super().__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dimension)
        self.projection_layer = nn.Linear(embedding_dimension, vocab_size)

    def forward(self, bow):
        embeddings = self.embeddings(bow)
        average_bow = torch.mean(embeddings, dim=1)
        logits = self.projection_layer(average_bow)
        return logits
```

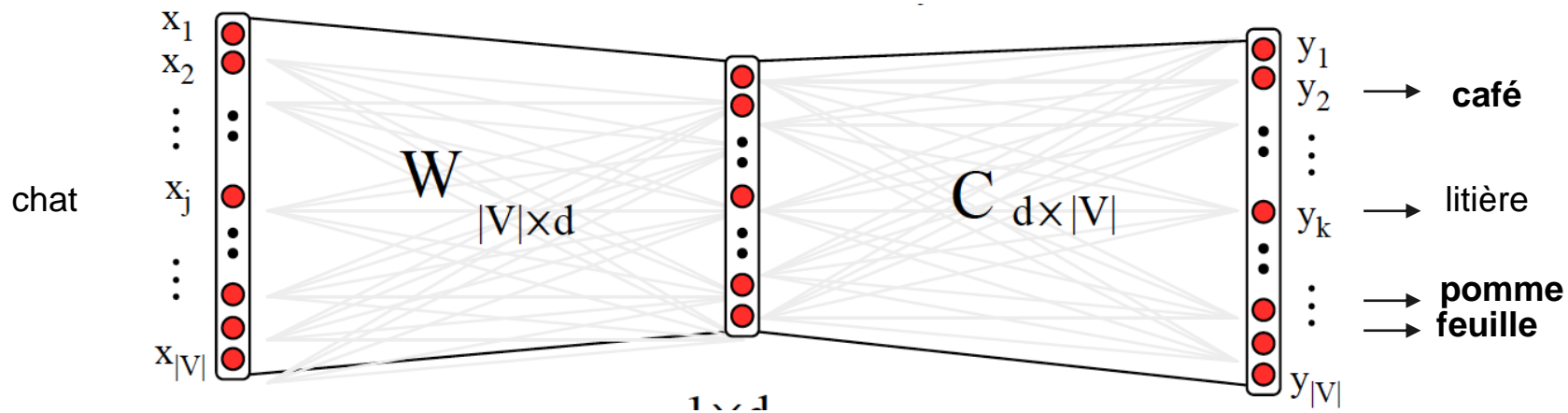
CBOW

$$\sum_{t=1}^T \log p(w_t | C_t) ,$$

CBOW - Negative Sampling



CBOW - Negative Sampling



CBOW - Negative Sampling

$$\log \left(1 + e^{-s(w,C)} \right) + \sum_{n \in N_C} \log \left(1 + e^{s(n,C)} \right),$$

score entre un mot **w** et un context **C**



CBOW - Comment obtenir un score

Produit vectoriel entre \mathbf{v}_C et \mathbf{v}_w

$$v_C = \sum_{p \in P} d_p \odot u_{t+p},$$

$$v_w + \frac{1}{|N|} \sum_{n \in N} x_n.$$

CBOW - Comment obtenir un score

$$v_w + \frac{1}{|N|} \sum_{n \in N} x_n \cdot \quad = \quad \langle \text{wh, whe, her, ere, re} \rangle, \langle \text{where} \rangle$$

CBOW - Phrase Representations

$v(\text{New}) + v(\text{York}) \approx \text{Boston?}$

CBOW - Phrase Representations

$v(\text{New}) + v(\text{York}) \approx \text{Issshhh?}$

CBOW - Phrase Representations

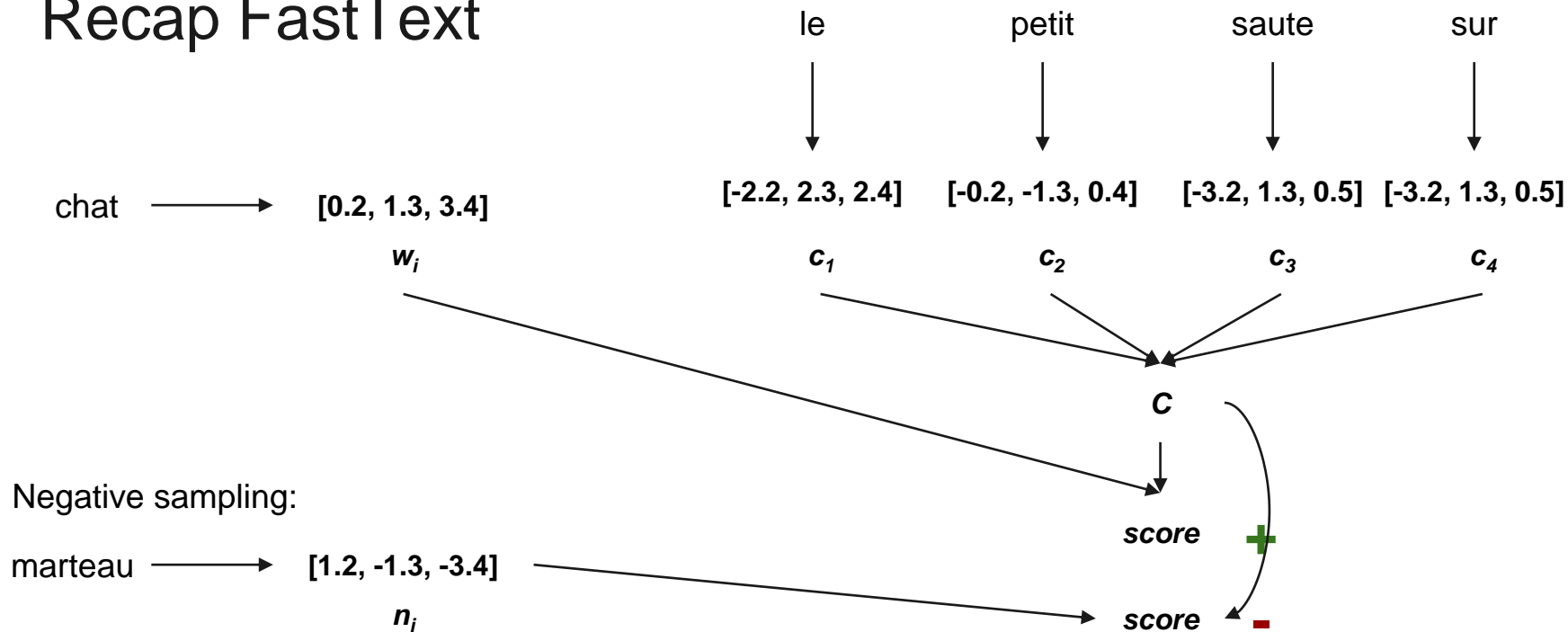
New York => New_York

Démo FastText

Recap FastText

... *le petit* **chat** *saute sur* ...

Recap FastText



ELMo

On le verra dans la section modèles de langue..!



Vecteurs de phrases

Comment obtenir la représentation d'une phrase?

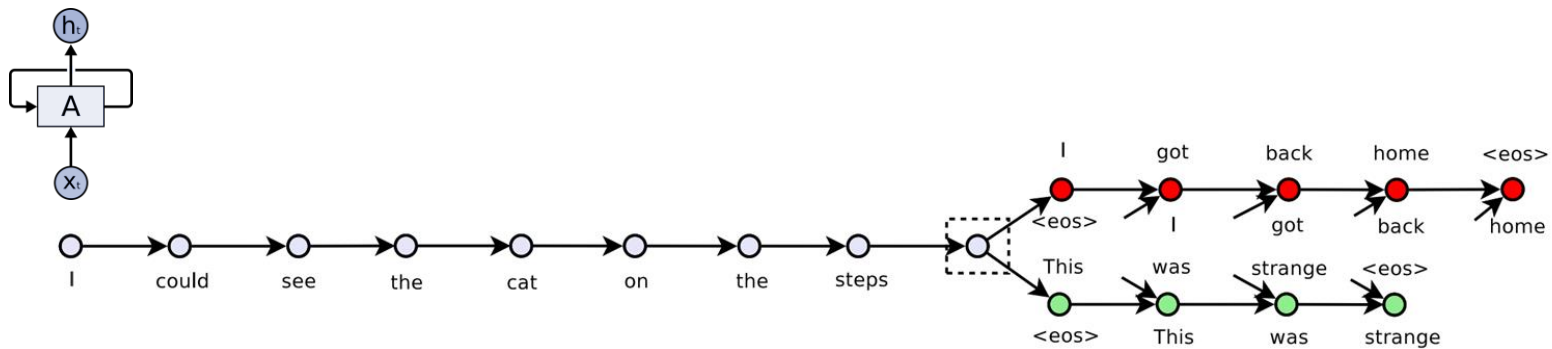
- Prendre la moyenne des embeddings de mots
- Utiliser une idée similaire à Skip-Gram!

Skip-Thought Vectors

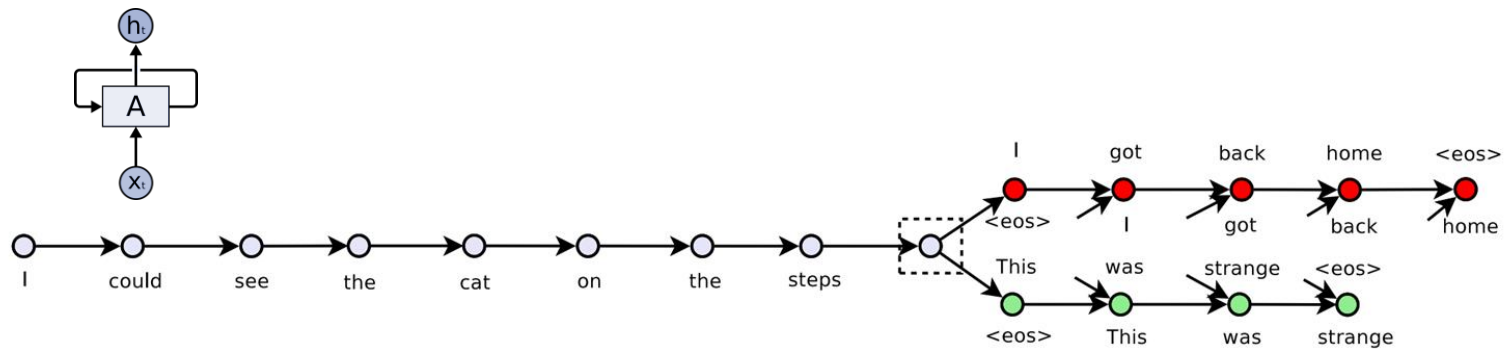
Idée de base:

- Étant donné un triplet de phrases (\mathbf{s}_{i-1} , \mathbf{s}_i , \mathbf{s}_{i+1})
 - Encoder la phrase \mathbf{s}_i
 - Générer les phrases \mathbf{s}_{i-1} et \mathbf{s}_{i+1}

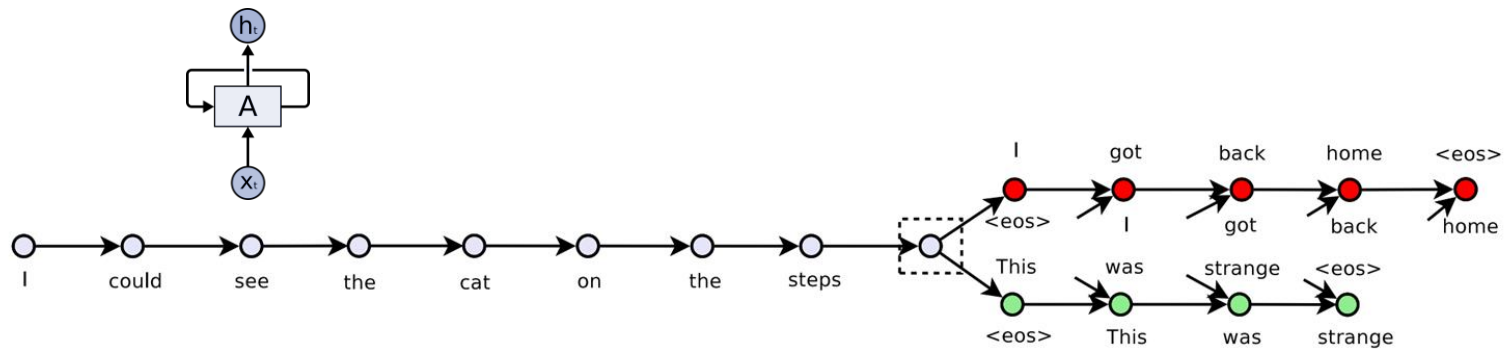
Skip-Thought Vectors



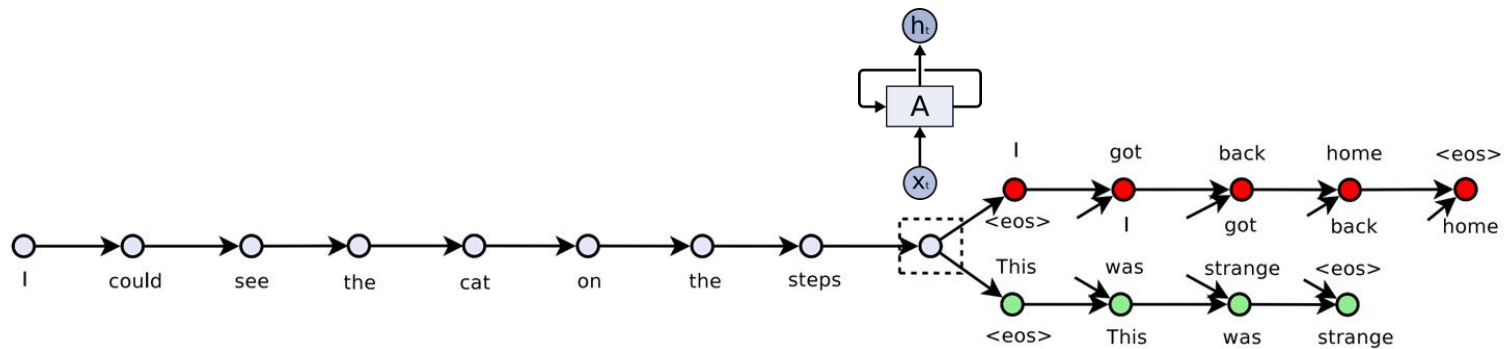
Skip-Thought Vectors



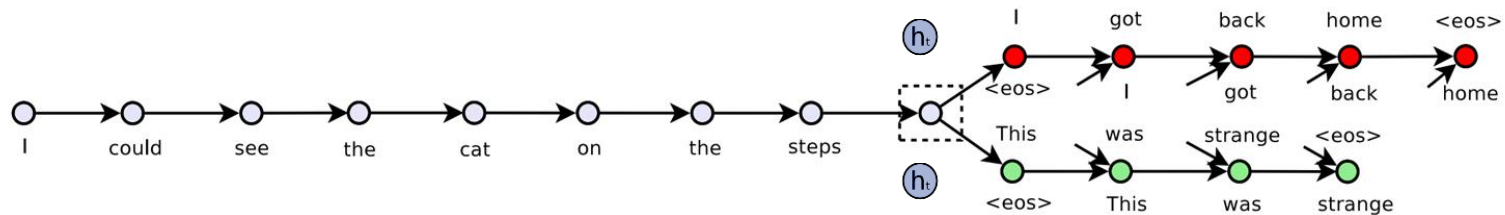
Skip-Thought Vectors



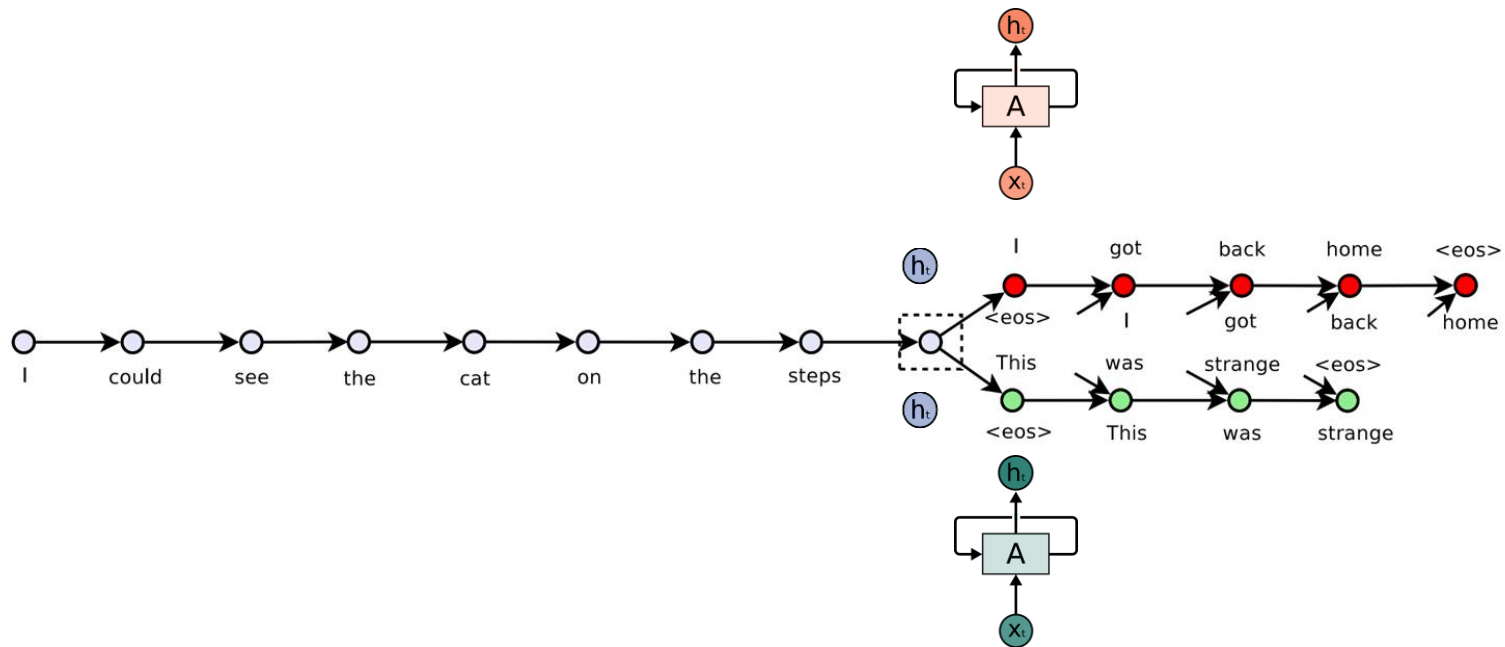
Skip-Thought Vectors



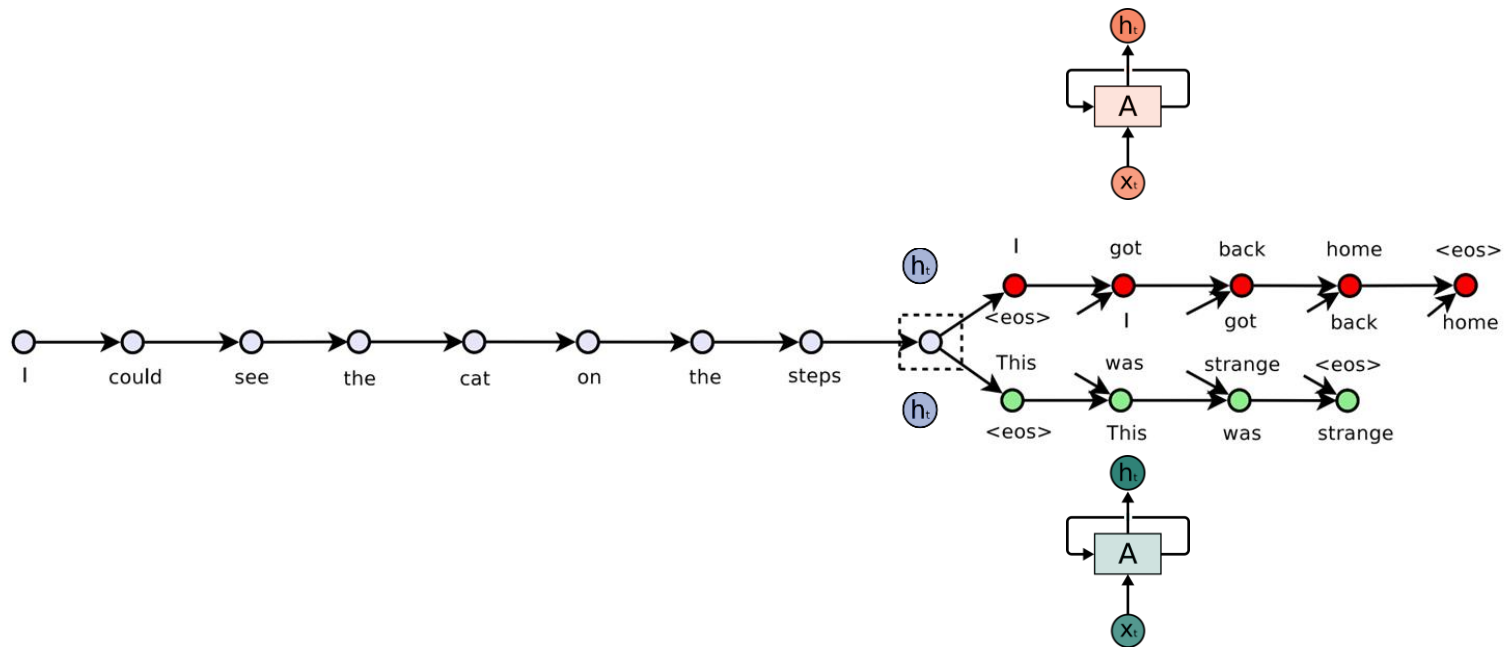
Skip-Thought Vectors



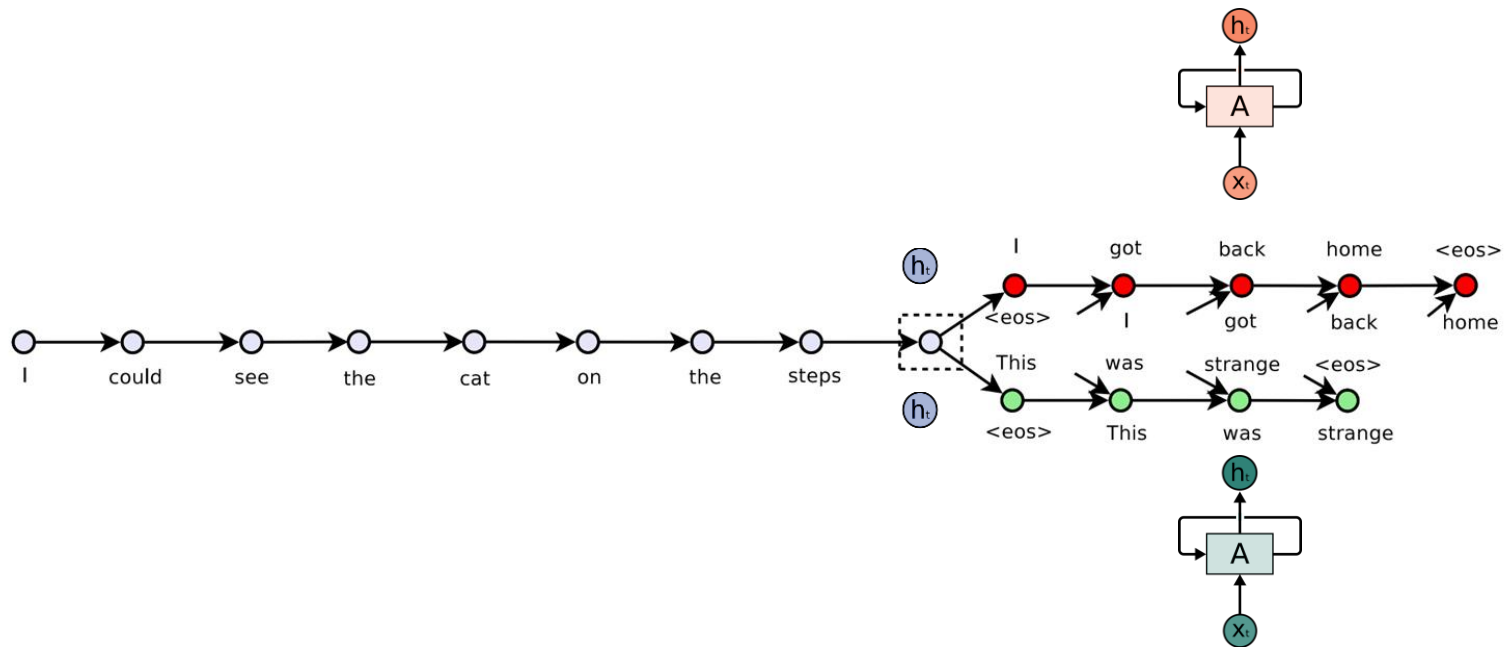
Skip-Thought Vectors



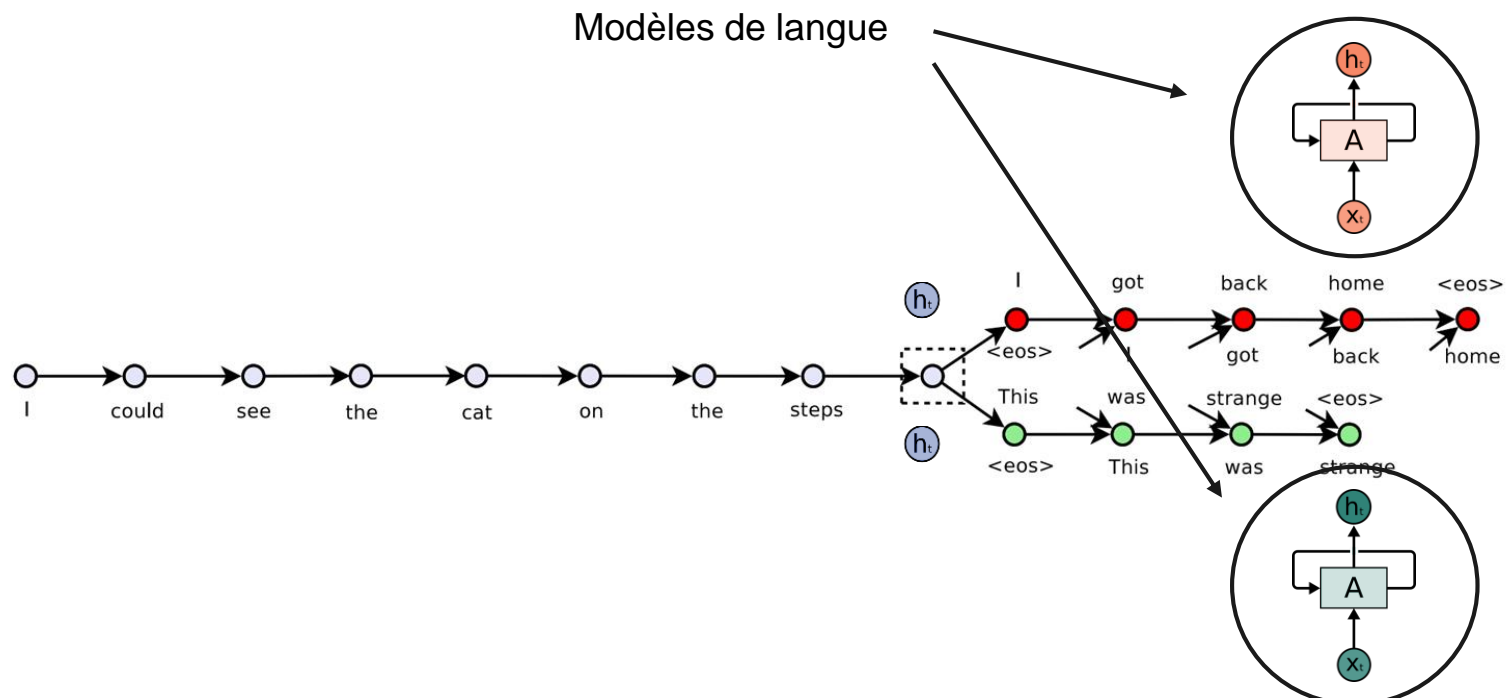
Skip-Thought Vectors



Skip-Thought Vectors

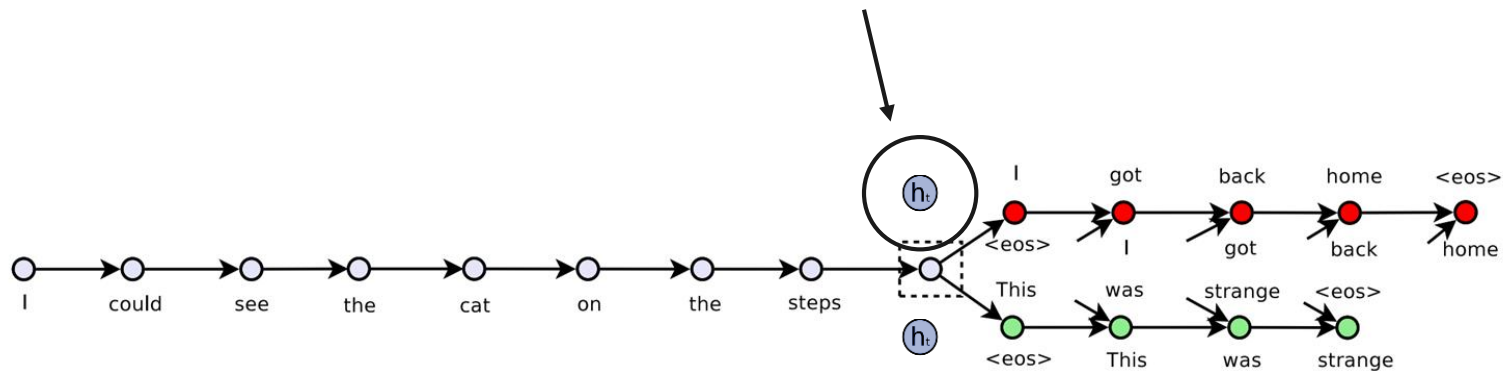


Skip-Thought Vectors



Skip-Thought Vectors

Au final, on se sert de ça!



Skip-Thought Vectors

Probabilité d'avoir généré la phrase suivante

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)$$

Probabilité d'avoir généré la phrase
précédente