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: $\boldsymbol{w_i}$
- Table d’embeddings (lookup): $\mathbf{v}$
- Embedding: $\boldsymbol{e_i}$
- $\boldsymbol{e_i = \mathbf{v}(w_i)}$

Représentation d'un mot

Différentes possibilités:

- Vecteur One-hot
  - *Chat*: [0,0,... 0,**1**,0,0,0,0,0,0,0,0,0...]
- Vecteur de context
  - *Chat*: [**1**,0,... 0,**0**,0,0,0,0,**1**,0,0,**1**,0,0...]

félin **chat** litière lait

Vecteurs de contexte

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

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.

“Chap. 15: Vector Semantics.” Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

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

"Chap. 15: Vector Semantics." Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

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.

"Chap. 15: Vector Semantics." Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

Vecteurs de contexte

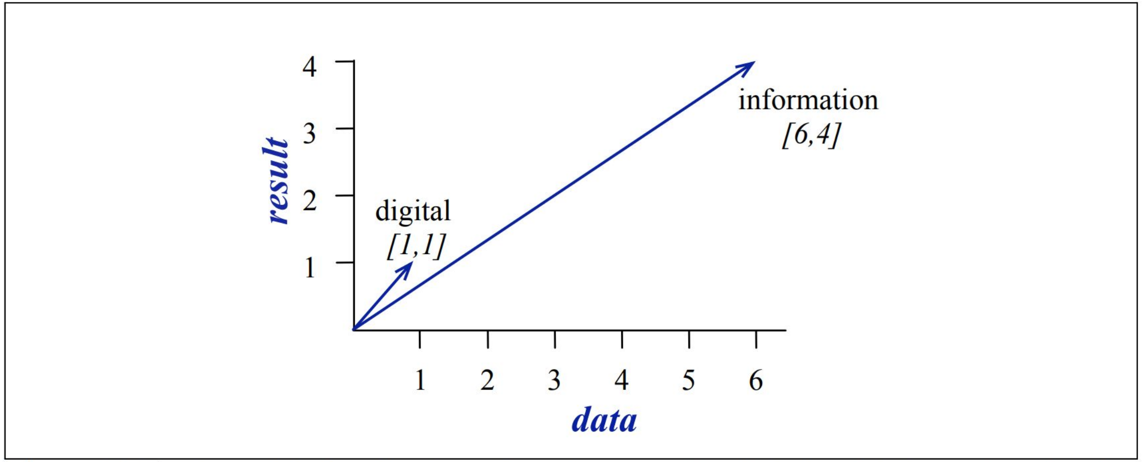
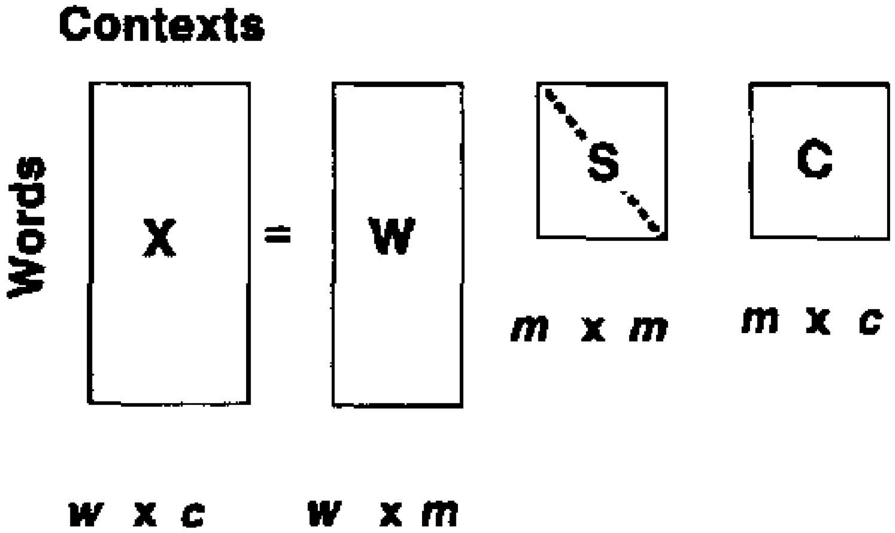


Figure 15.5 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

“Chap. 15: Vector Semantics.” Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

Une façon de réduire la dimensionnalité



Landuaer and Dumais 1997

Décomposition en valeurs singulières

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

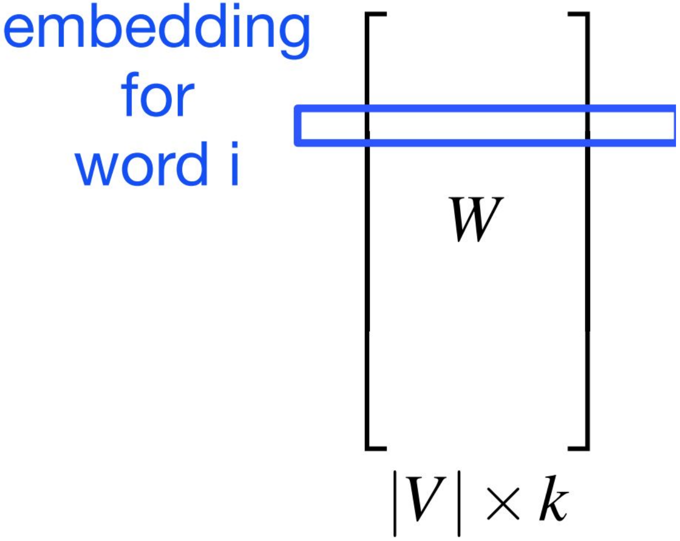
“Chap. 15: Vector Semantics.” Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

On conserve les top *k* valeurs singulières

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

"Chap. 15: Vector Semantics." Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

On utilise ensuite seulement la matrice *W*



“Chap. 15: Vector Semantics.” Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, by Dan Jurafsky and James H. Martin, Dorling Kindersley Pvt, Ltd., 2014.

Méthodes à réseaux de neurones

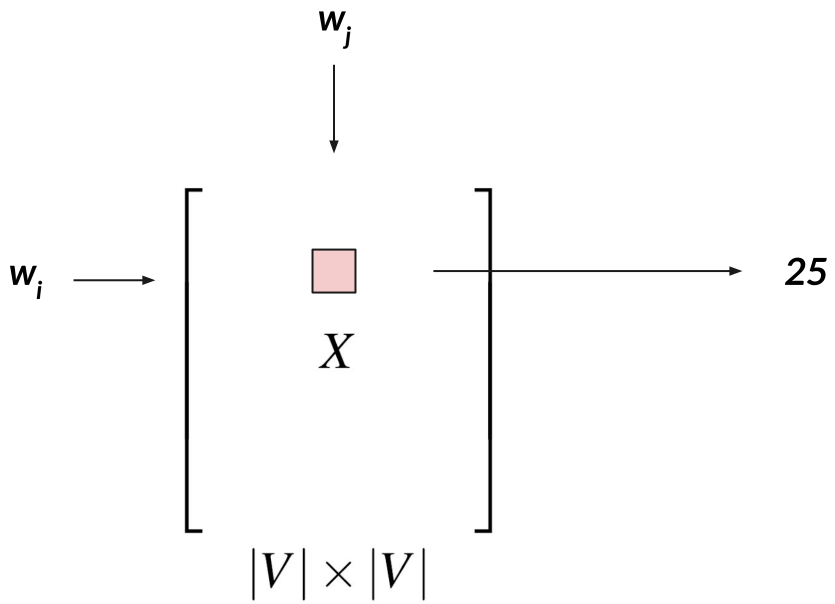
GloVe

“A weighted least squares regression model”

L’idée est de prédire le nombre de co-occurrences $\mathbf{X}_{ij}$ (ou le *log*) des mots $\mathbf{w}_i$ et $\mathbf{w}_j$

S’apparente à Word2Vec (ou encore FastText)

GloVe



GloVe

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

GloVe

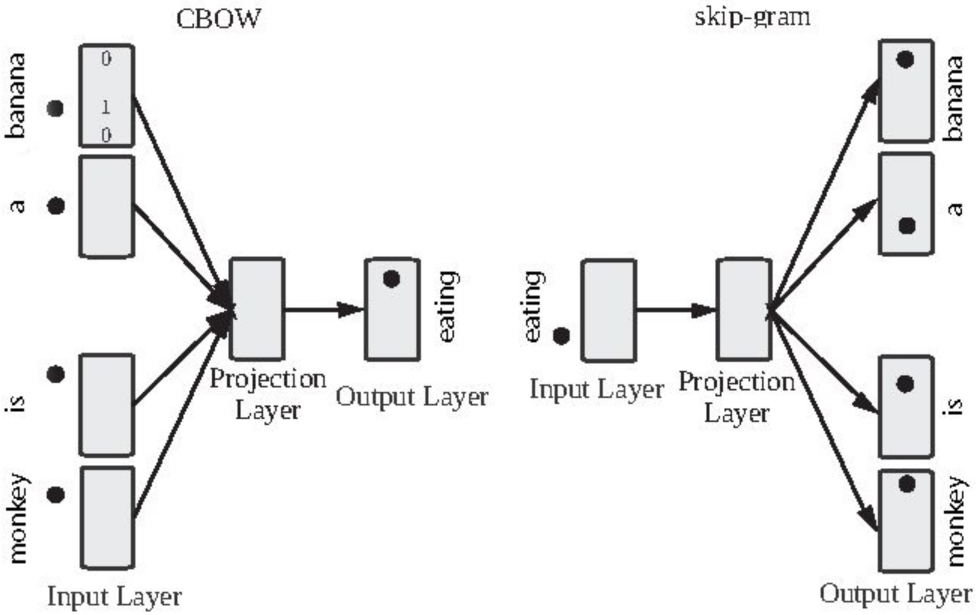
$$\text{Perte} = \mathbf{v}(\boldsymbol{w_i}) * \mathbf{v}(\boldsymbol{w_j}) + \mathbf{b_i} + \mathbf{b_j} - \log(\mathbf{25})$$

Word2Vec

2 algorithmes:

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

Word2Vec



CBOW
skip-gram
banana
a
is
monkey
0
1
0
Input Layer
Projection Layer
Output Layer
eating
eating
Input Layer
Projection Layer
banana
a
is
monkey
Output Layer

CBOW

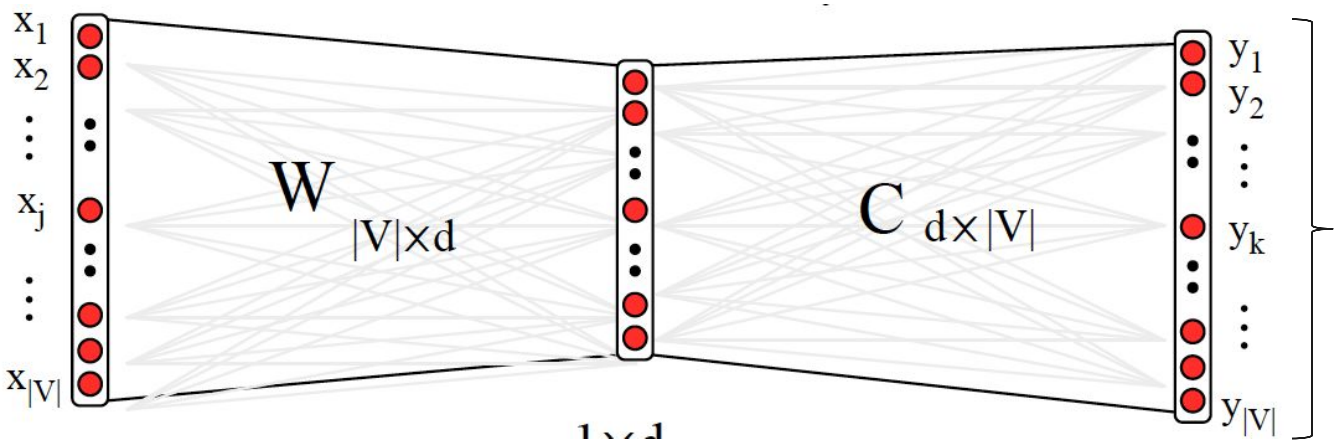
```
class CBOW(nn.Module):
    def __init__(self, vocab_size, embedding_dimension):
        super().__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dimension)
        self.projection_layer = nn.Linear(embedding_dimension, vocab_size)

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

CBOW

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

CBOW - Negative Sampling

chat

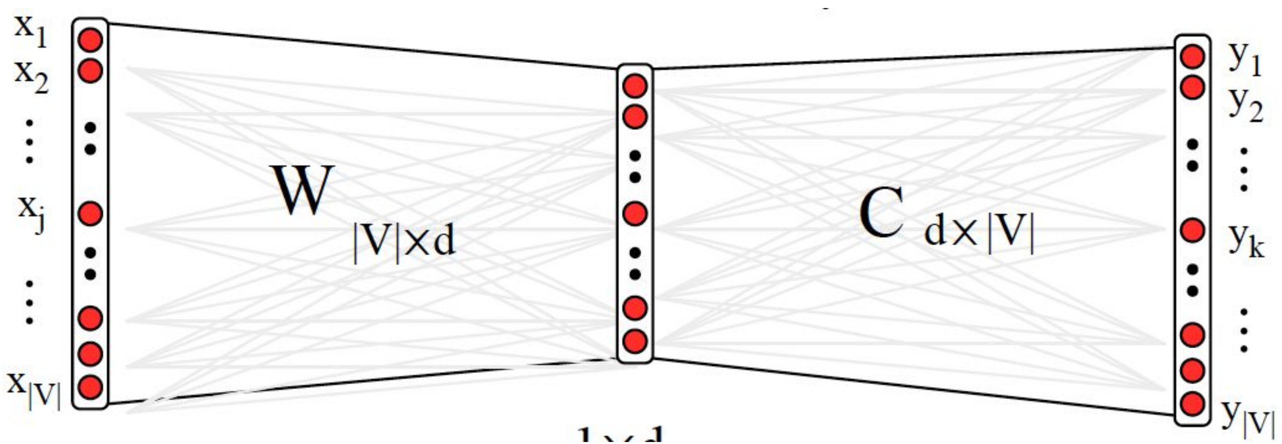


litière

softmax

CBOW - Negative Sampling

chat



café

litière

pomme

feuille

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 $\boldsymbol{v_C}$ et $\boldsymbol{v_w}$

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

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

CBOW - Comment obtenir un score

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

= <wh, whe, her, ere, re>, <where>

CBOW - Phrase Representations

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

CBOW - Phrase Representations

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

CBOW - Phrase Representations

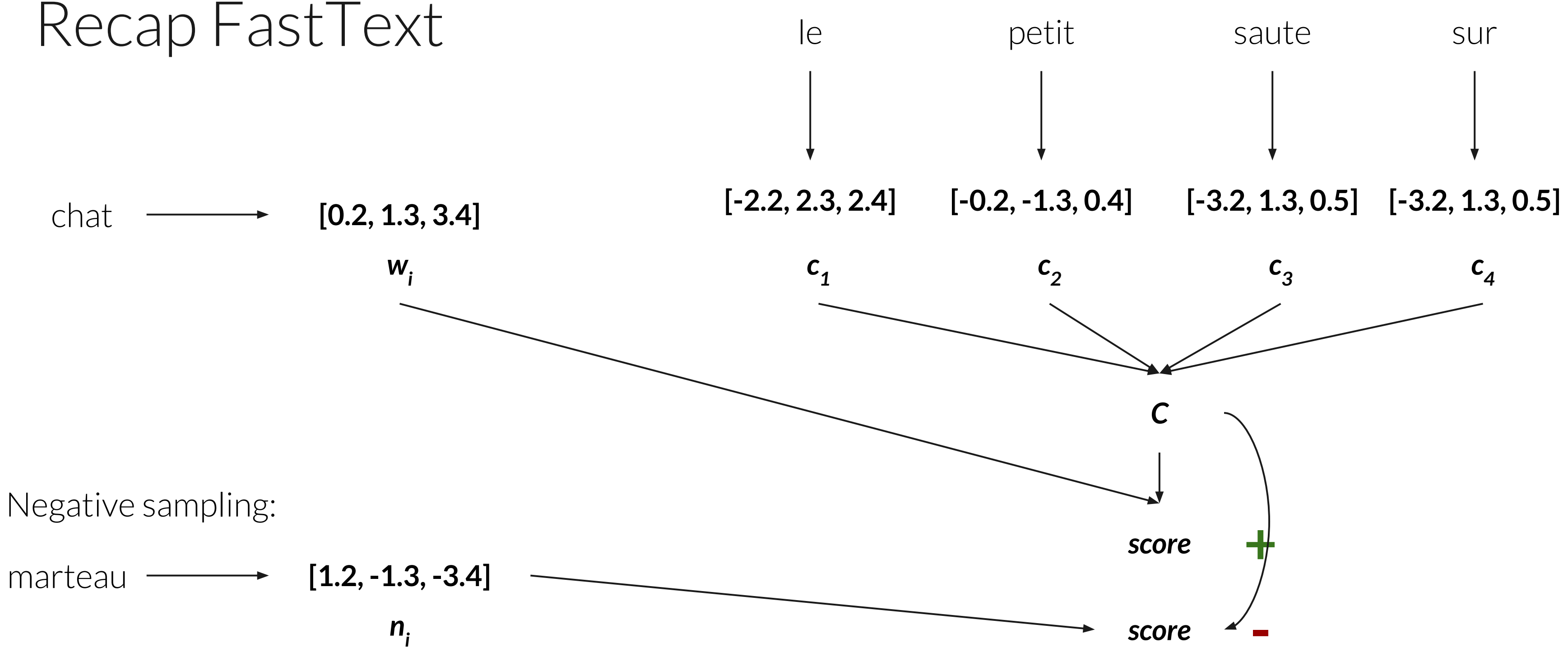
New York => New_York

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?

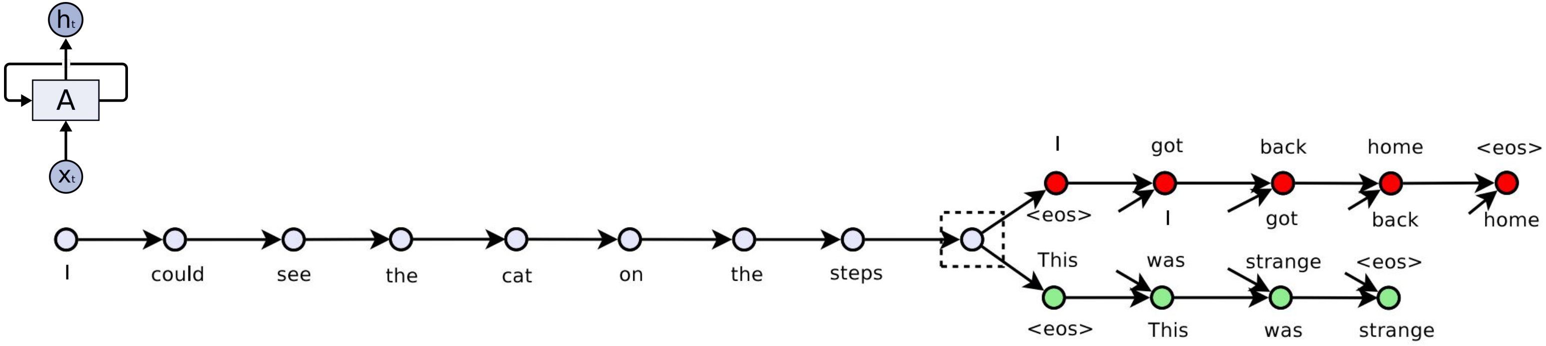
- Prendre la moyenne des embeddings de mots
- Utiliser une idée similaire à Skip-Gram!

Skip-Thought Vectors

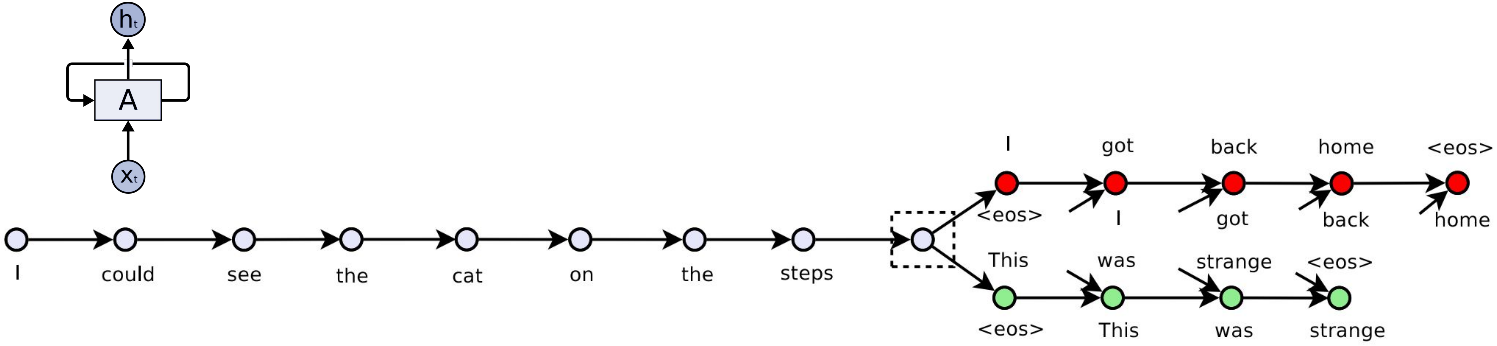
Idée de base:

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

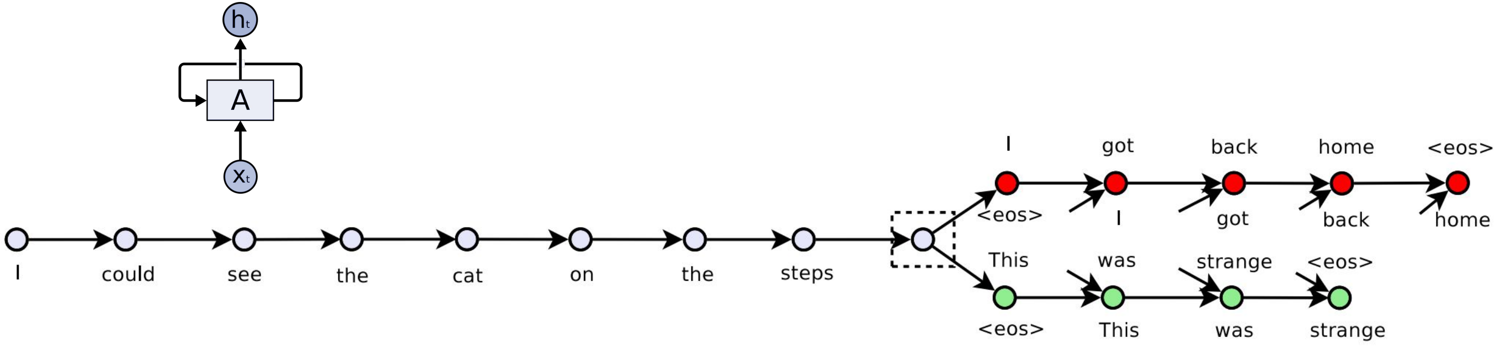
Skip-Thought Vectors



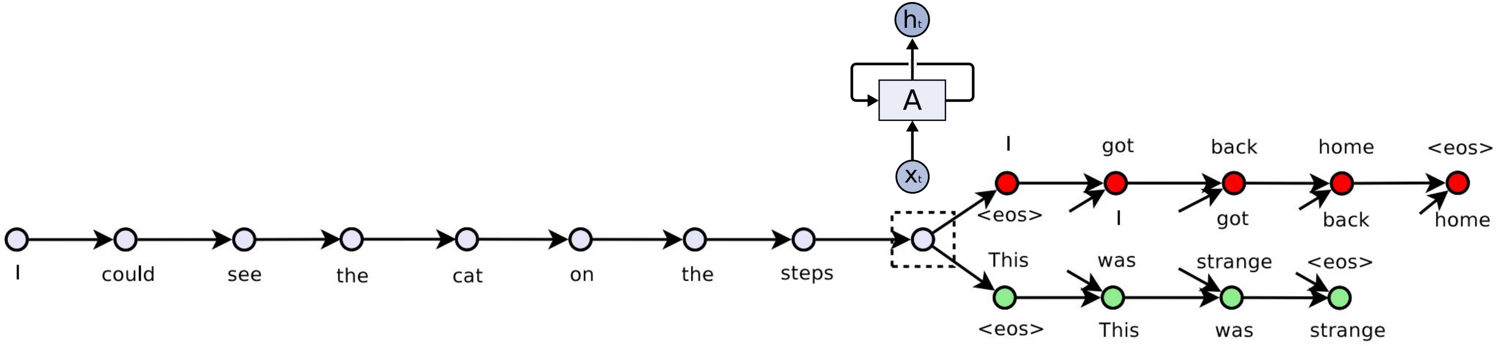
Skip-Thought Vectors



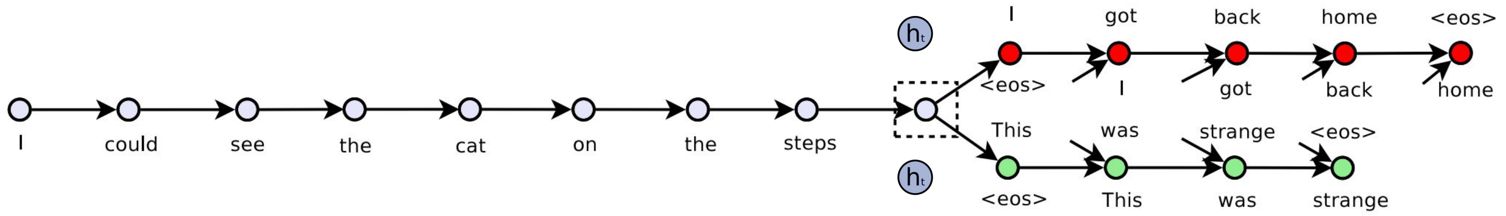
Skip-Thought Vectors



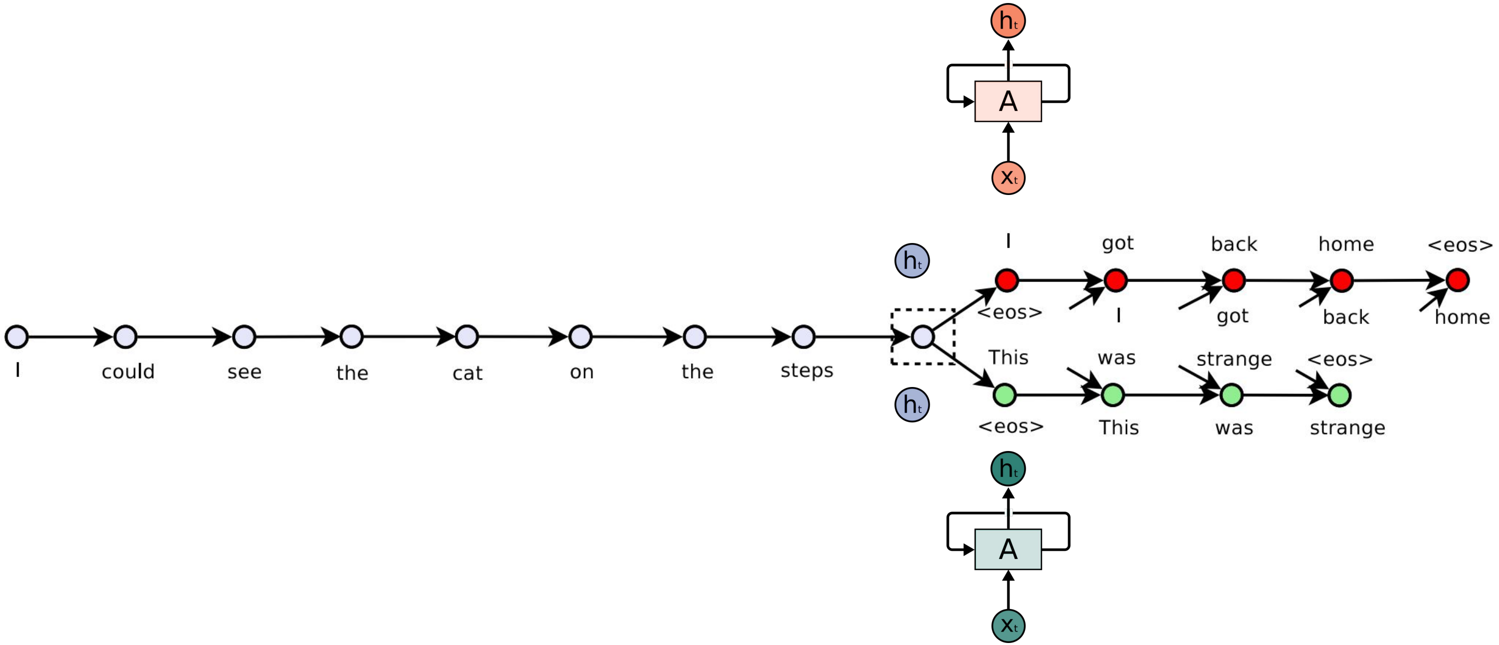
Skip-Thought Vectors



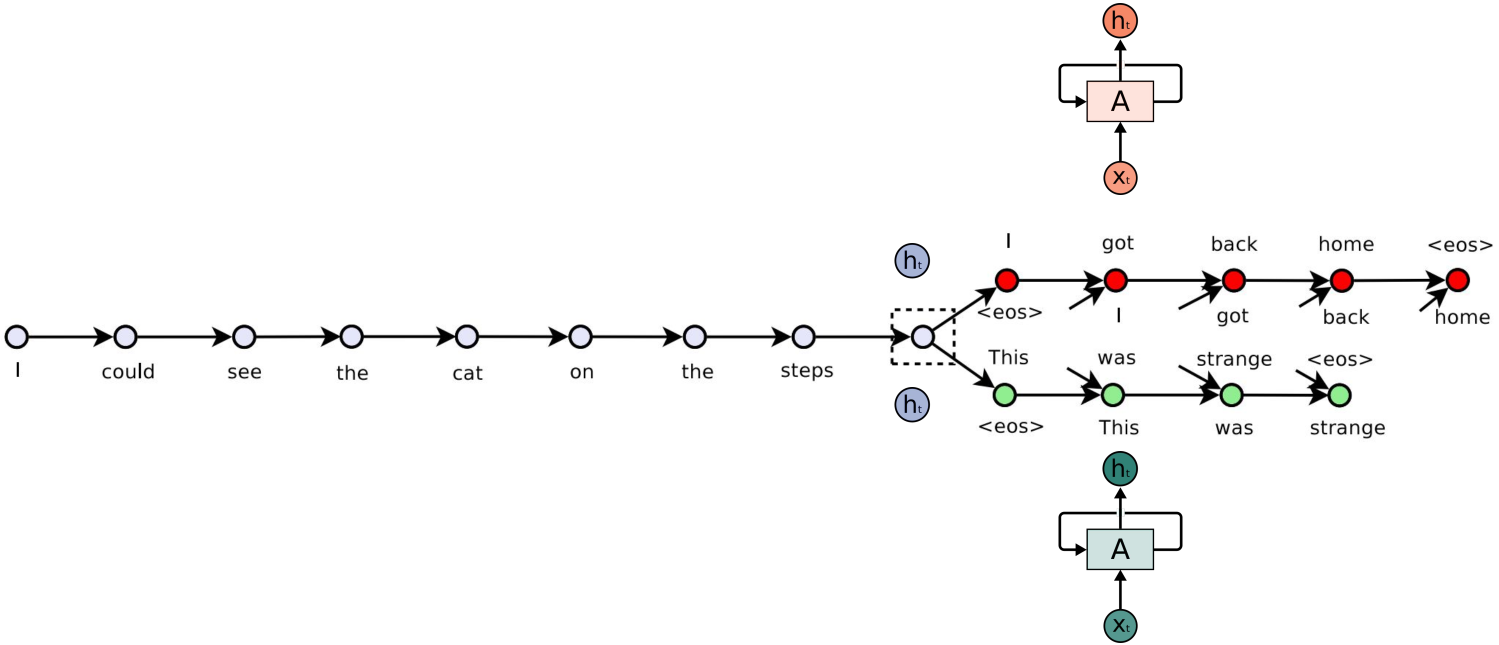
Skip-Thought Vectors



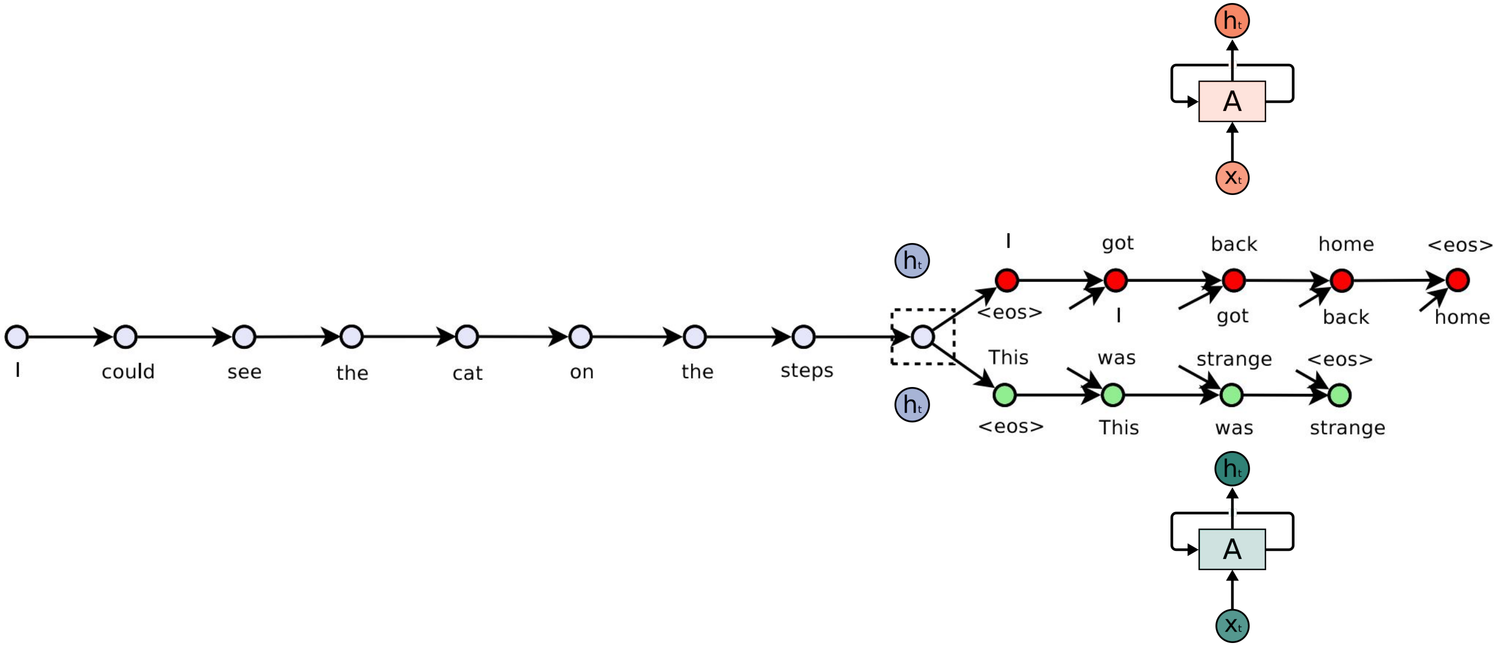
Skip-Thought Vectors



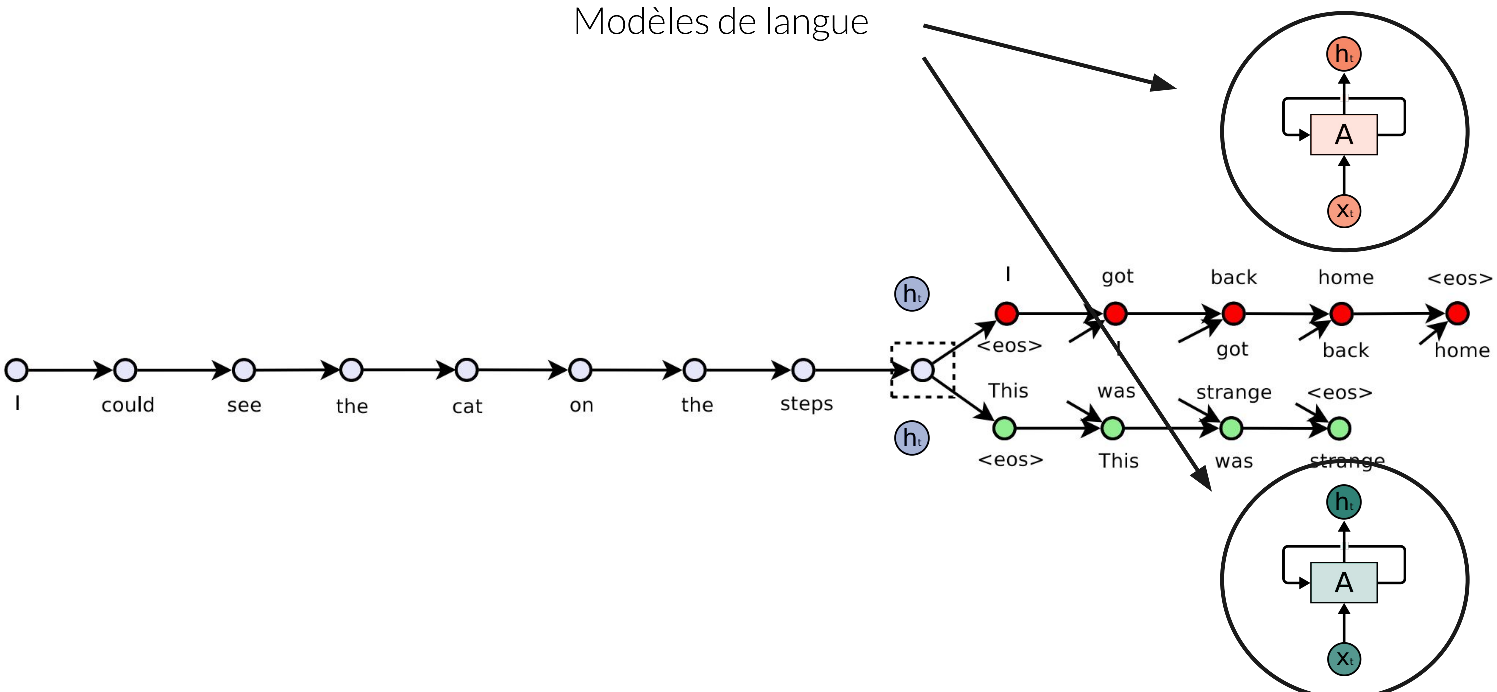
Skip-Thought Vectors



Skip-Thought Vectors

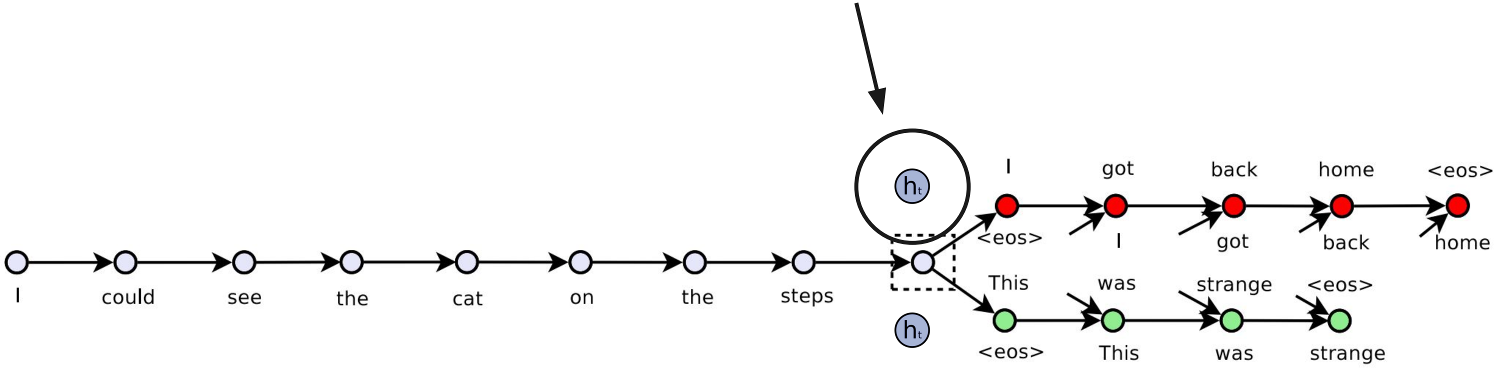


Skip-Thought Vectors



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