

# Cours 1 : Représentations vectorielles

François HU - 13/10/2020

Data Scientist au DataLab de la Société Générale Assurances

Doctorant à l'ENSAE-CREST

Les cours se trouvent ici : <https://curiousml.github.io/>

# Sommaire

## 1. Quelques motivations

## 2. Words Embeddings

- Représentations vectorielles des mots :  
*one-hot-encoding* ?
- word embedding
- word2vec
- doc2vec

## 3. Topic modeling

- Représentation vectorielle des documents / topics
- Latent Semantic Analysis (**LSA**)
- Probabilistic LSA (**PLSA**)
- Latent Dirichlet Allocation (**LDA**)

Introduction

Représentations vectorielles

Deep Learning pour NLP

Active Learning

# Quelques motivations

## Contextes **non-supervisé**

Représenter **numériquement des mots** de sorte que la relation suivante ait un sens :

$$\text{vect}(\text{king}) - \text{vect}(\text{man}) + \text{vect}(\text{woman}) = \text{vect}(\text{queen})$$

Représenter **numériquement des thèmes (topics) cachés** et les **documents** :

- créer des **systèmes de recommandation**  
(utilisés par les e-commerçants, les moteurs de recherche, ...)
- **categorisation** de textes
- processus d'**exploration des données**
- en bio-informatique : **extraire des connaissances cachées** des données biologiques (molécules d'ADN)

## **2. Word Embeddings**

# Représentation vectorielle des mots : *one-hot-encoding* ?

objectif

texte à apprendre : mon client est content de son assurance → 

texte à prédire : l'assuré est satisfait de notre contrat → ?

# Représentation vectorielle des mots : *one-hot-encoding* ?

objectif

texte à apprendre : mon client est content de son assurance → 

texte à prédire : l'assuré est satisfait de notre contrat → ?

## 1. définir le vocabulaire

```
V = {'a' : 1, ...  
     'assurance' : 59, ...  
     'assuré' : 62, ...  
     'client' : 698, ...  
     'content' : 772, ...  
     'contrat' : 899, ...  
     'satisfait' : 8 201, ...  
     'zoo' : 9 999, ...  
     <UNK> : 10 000 }
```

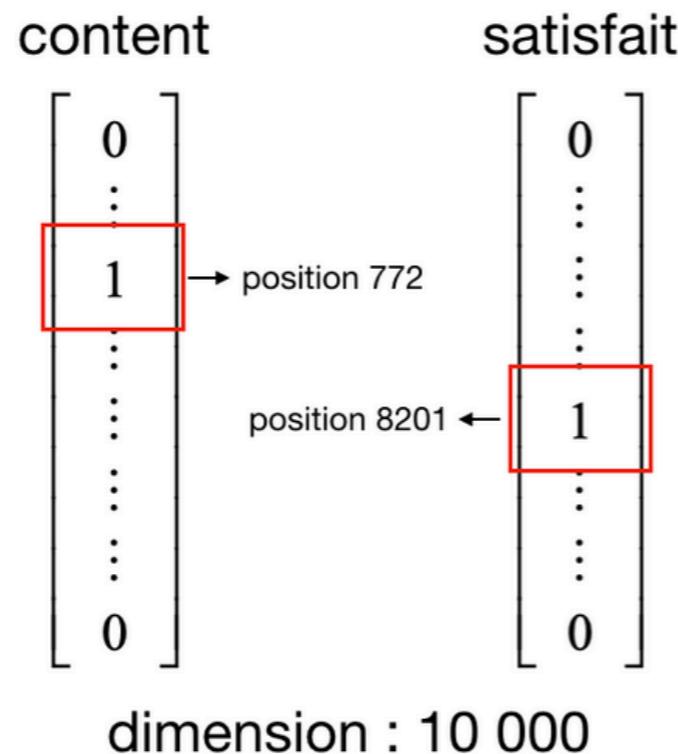
# Représentation vectorielle des mots : *one-hot-encoding* ?



1. définir le vocabulaire

2. one-hot encoding

```
V = {'a' : 1, ...
      'assurance' : 59, ...
      'assuré' : 62, ...
      'client' : 698, ...
      'content' : 772, ...
      'contrat' : 899, ...
      'satisfait' : 8 201, ...
      'zoo' : 9 999, ...
      <UNK> : 10 000 }
```



Approche naïve de vectorisation :

approche one-hot

approche TF-IDF

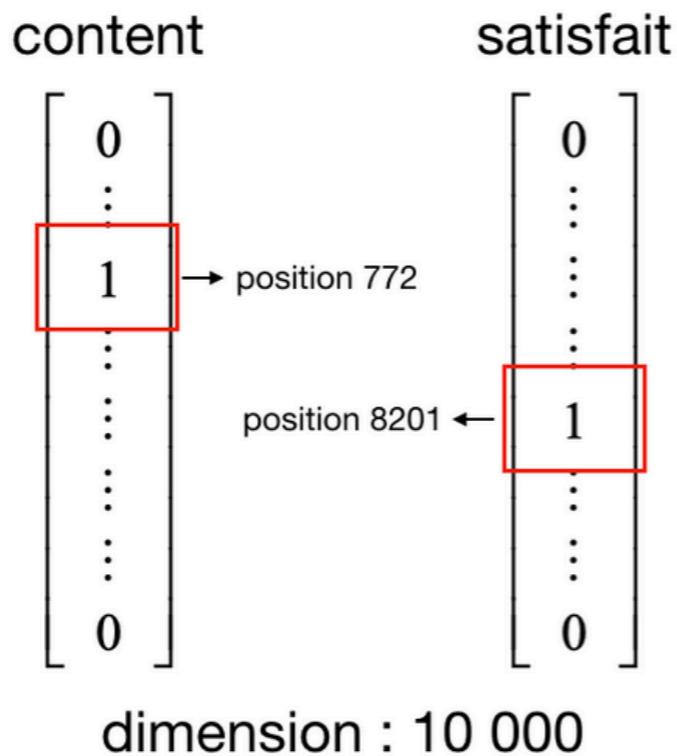
# Représentation vectorielle des mots : one-hot-encoding ?



## 1. définir le vocabulaire

```
V = {'a' : 1, ...
      'assurance' : 59, ...
      'assuré' : 62, ...
      'client' : 698, ...
      'content' : 772, ...
      'contrat' : 899, ...
      'satisfait' : 8 201, ...
      'zoo' : 9 999, ...
      <UNK> : 10 000 }
```

## 2. one-hot encoding



## Approche naïve de vectorisation :

approche one-hot

approche TF-IDF

## Problèmes :

vecteurs creux de grandes dimensions

sauvegardent des mots sans prendre en compte le contexte :

vect (« content »)



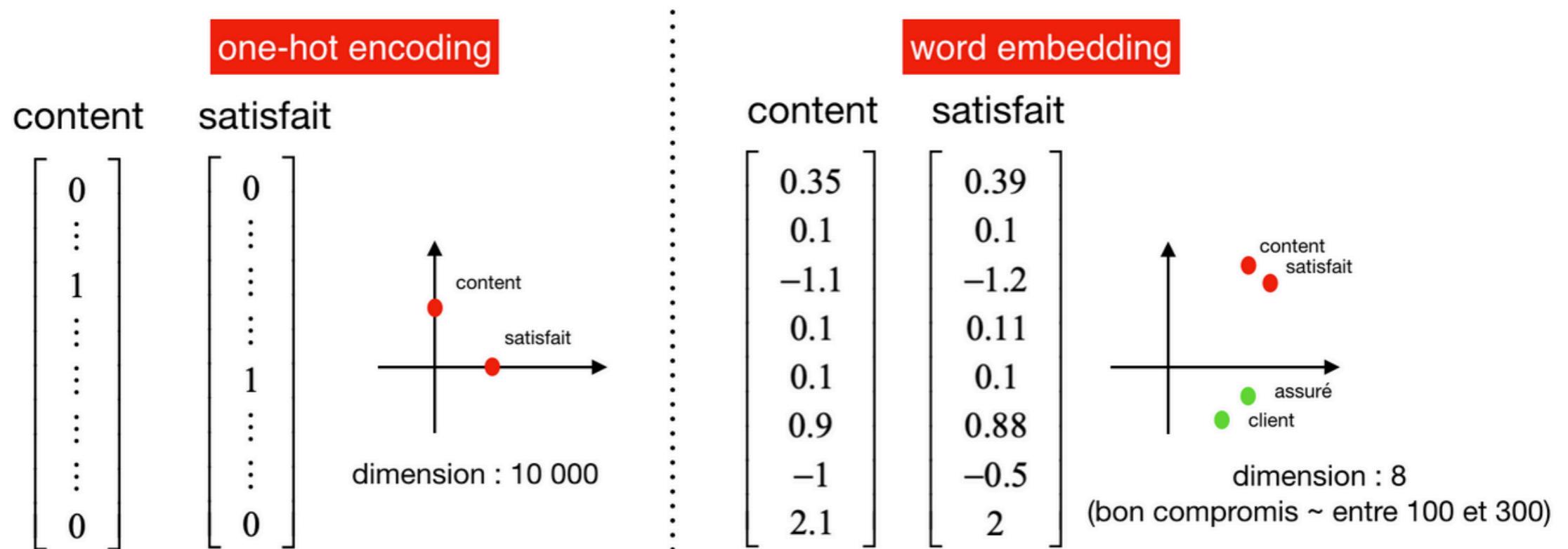
vect (« satisfait »)

## Solution :

approche word embedding

# Word Embedding

**Word embedding (plongement de mot en français)** : vectorisation des mots de sorte que les mots apparaissant dans des contextes similaires ont des significations apparentées



**Vecteurs denses** de plus petites dimensions

$$\text{mot1} \approx \text{mot2} \implies \text{word\_embedding}(\text{mot1}) \approx \text{word\_embedding}(\text{mot2})$$

Word embedding populaire : **word2vec**

# Word2vec

**Principe :** utiliser les réseaux de neurones pour construire ces vecteurs

**Deux architectures** de word2vec : **Continuous Bag-Of-Words** (CBOW) et **Skip-Gram**

**Exemple de texte :**

notre client est content de son assurance automobile

notre client est content de son assurance automobile  
client content assurance automobile

# Word2vec

**Principe :** utiliser les réseaux de neurones pour construire ces vecteurs

**Deux architectures** de word2vec : **Continuous Bag-Of-Words** (CBOW) et **Skip-Gram**

**Exemple de texte :**

notre client est content de son assurance automobile

notre client est content de son assurance automobile

client content assurance automobile

**Générer des observations** pour word2vec pour une fenêtre de taille 2

	cible	contexte
observation 1 :	client	content assurance automobile
observation 2 :	client	content assurance automobile
observation 3 :	client	content assurance automobile
observation 4 :	client	content assurance automobile

# Word2vec

**Principe :** utiliser les réseaux de neurones pour construire ces vecteurs

**Deux architectures** de word2vec : **Continuous Bag-Of-Words** (CBOW) et **Skip-Gram**

**Exemple de texte :**

notre client est content de son assurance automobile  
notre client est content de son assurance automobile  
client content assurance automobile

**Générer des observations** pour word2vec pour une fenêtre de taille 2

	cible	contexte
observation 1 :	client	content assurance automobile
observation 2 :	client	content assurance automobile
observation 3 :	client	content assurance automobile
observation 4 :	client	content assurance automobile

**Modèle CBOW :**

cherche à prédire un mot à partir du contexte



réseaux de neurones

prédire assurance

**Modèle skip-gram :**

cherche à prédire les mots du contexte à partir d'un mot

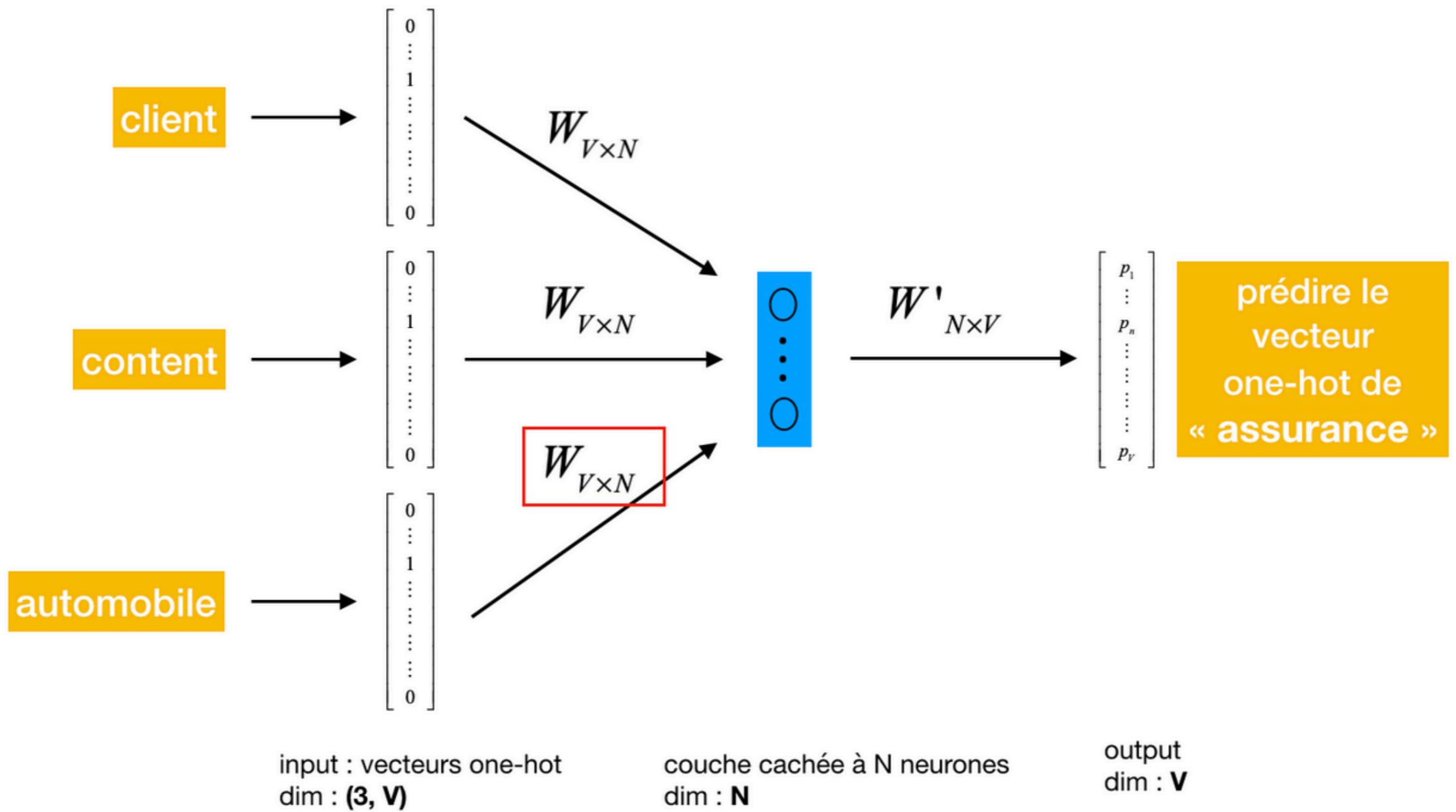


réseaux de neurones

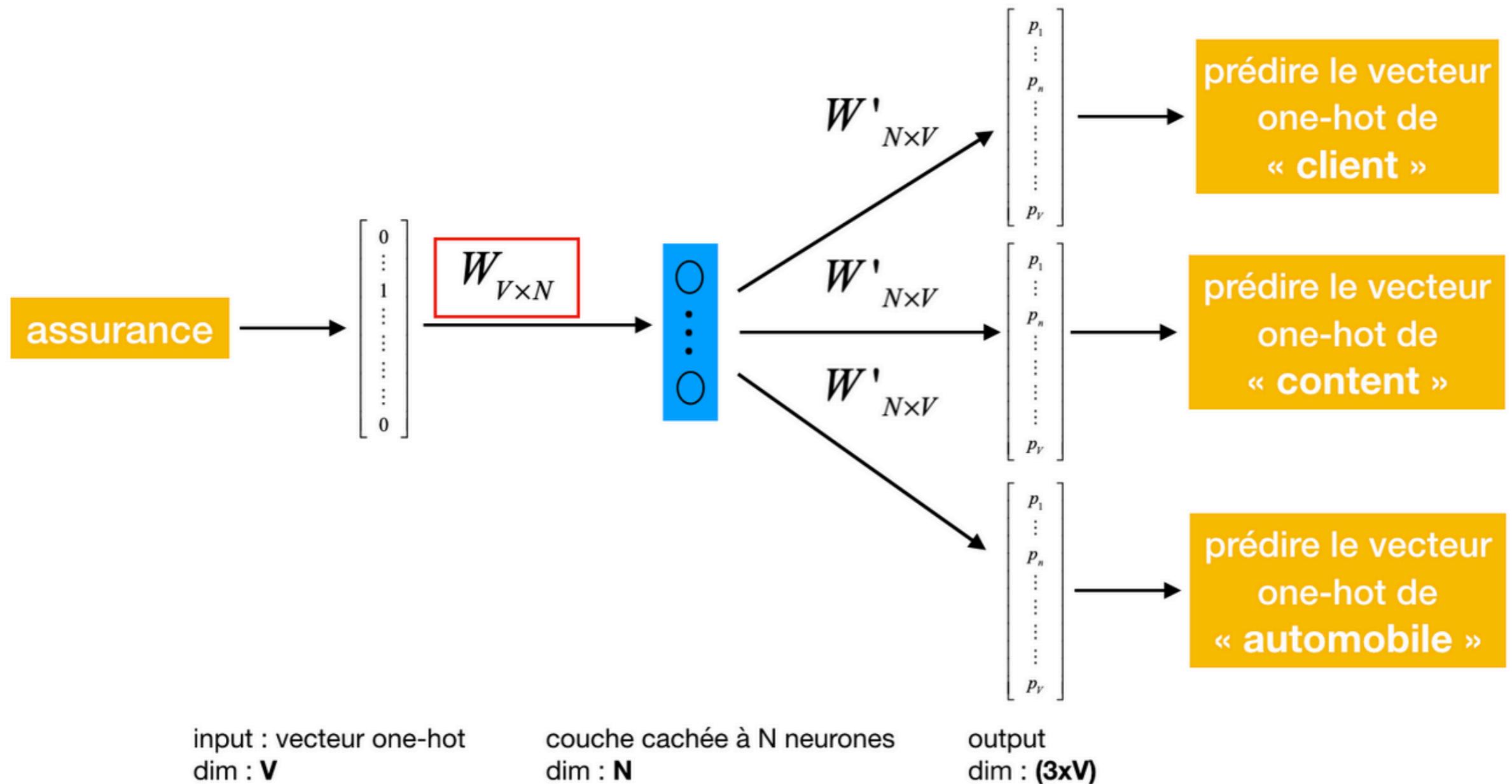
prédire : client content

automobile

# Word2vec : modèle CBOW



# Word2vec : modèle Skip-Gram



# Skip-Gram : phase d'entraînement

- prédire les **mots du contexte** sachant le mot central :

$$p(w_{t-h}, \dots, w_{t+h} | w_t; \theta) = \prod_{-h \leq k \leq h, k \neq 0} p(w_{t+k} | w_t; \theta)$$

- calculer (et maximiser) la proba du paramètre sachant les mots : **vraisemblance**

$$L(\theta) = \prod_{t=1, \dots, T} p(w_{t-h}, \dots, w_{t+h} | w_t; \theta) = \prod_{t=1, \dots, T} \prod_{-h \leq k \leq h, k \neq 0} p(w_{t+k} | w_t; \theta)$$

- **log-vraisemblance** :

$$\mathcal{L}(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1, \dots, T} \sum_{-h \leq k \leq h, k \neq 0} \log p(w_{t+k} | w_t; \theta)$$

- **softmax et paramètre**:

$$p(w_O | w_I) = \frac{e^{<v'_w_O, v_{w_I}>}}{\sum_{w \in V} e^{<v'_w, v_{w_I}>}}, \quad \theta = [v_{w_1}, \dots, v_{w_{|V|}}, v'_{w_1}, \dots] \in \mathbb{R}^{2 \cdot N \cdot |V|}$$

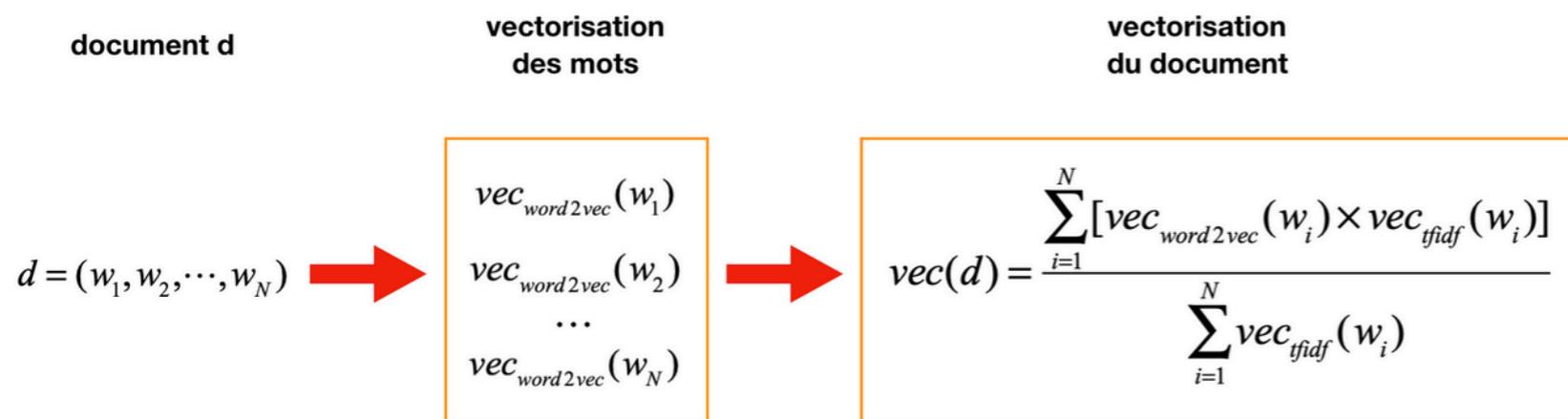
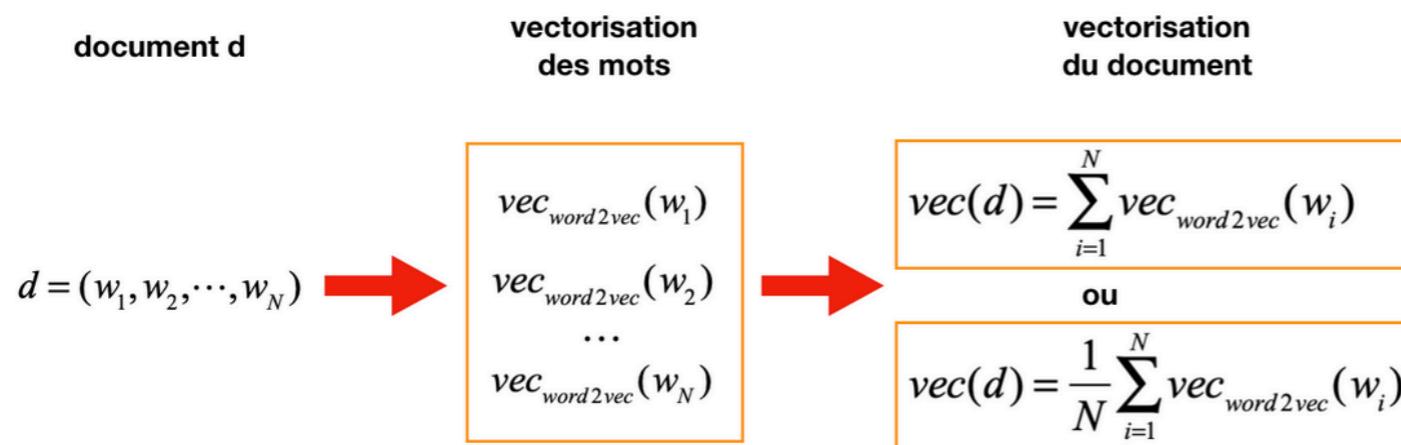
- algorithme de **descente de gradient** pour entraîner notre modèle

- **problème** : le calcul de softmax est trop long

- **solutions** : softmax hierarchique ou negative sampling

# classification de textes avec word2vec

agrégation puis classification classique :



doc2vec puis classification classique : voir [section suivante](#) pour le modèle doc2vec

classification avec les réseaux de neurones : voir [cours 2](#) pour les modèles séquentiels

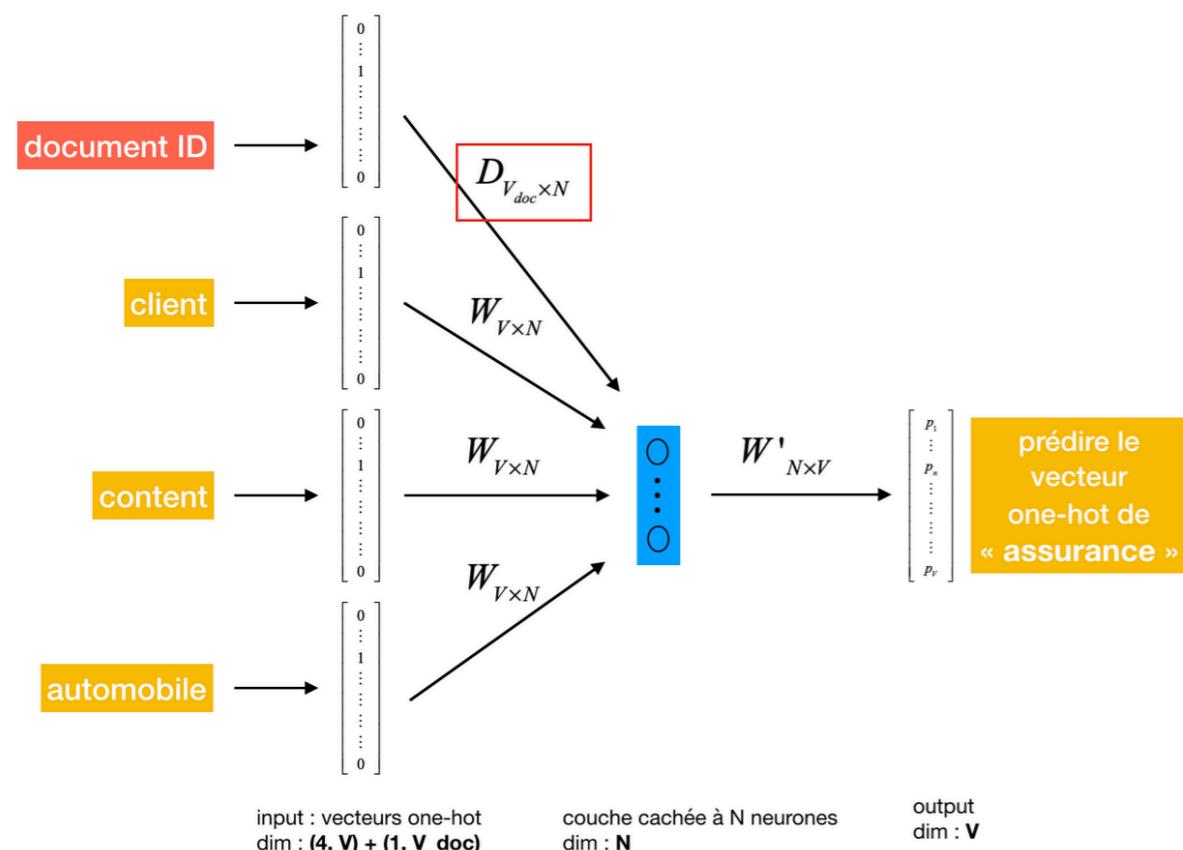
# classification de textes avec doc2vec

## doc2vec

vectorisation des **documents** (resp. des mots) de sorte que les **documents** (resp. les mots) apparaissant dans des contextes similaires ont des significations apparentées

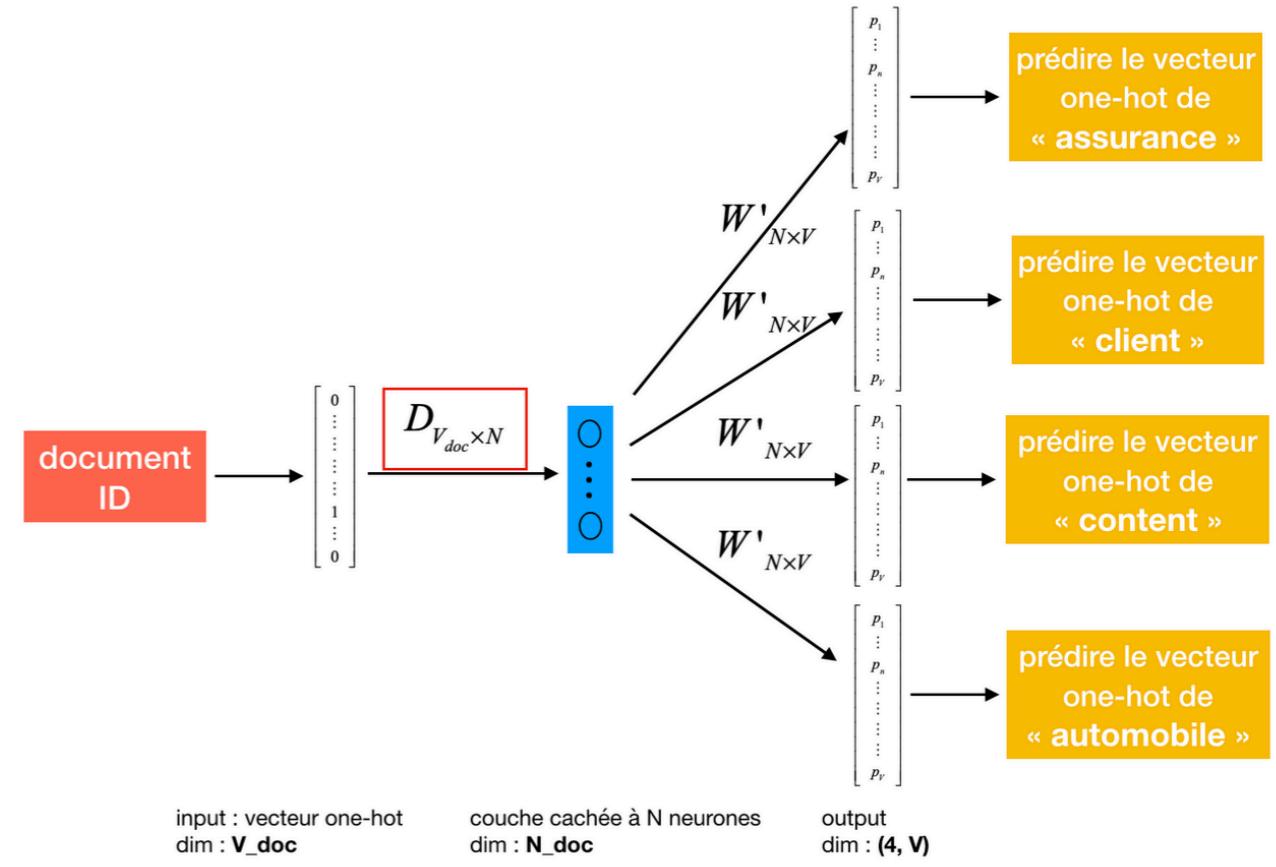
Distributed Memory (DM) :

$$p(w_i \mid w_{i-h}, \dots, w_{i+h}, d)$$

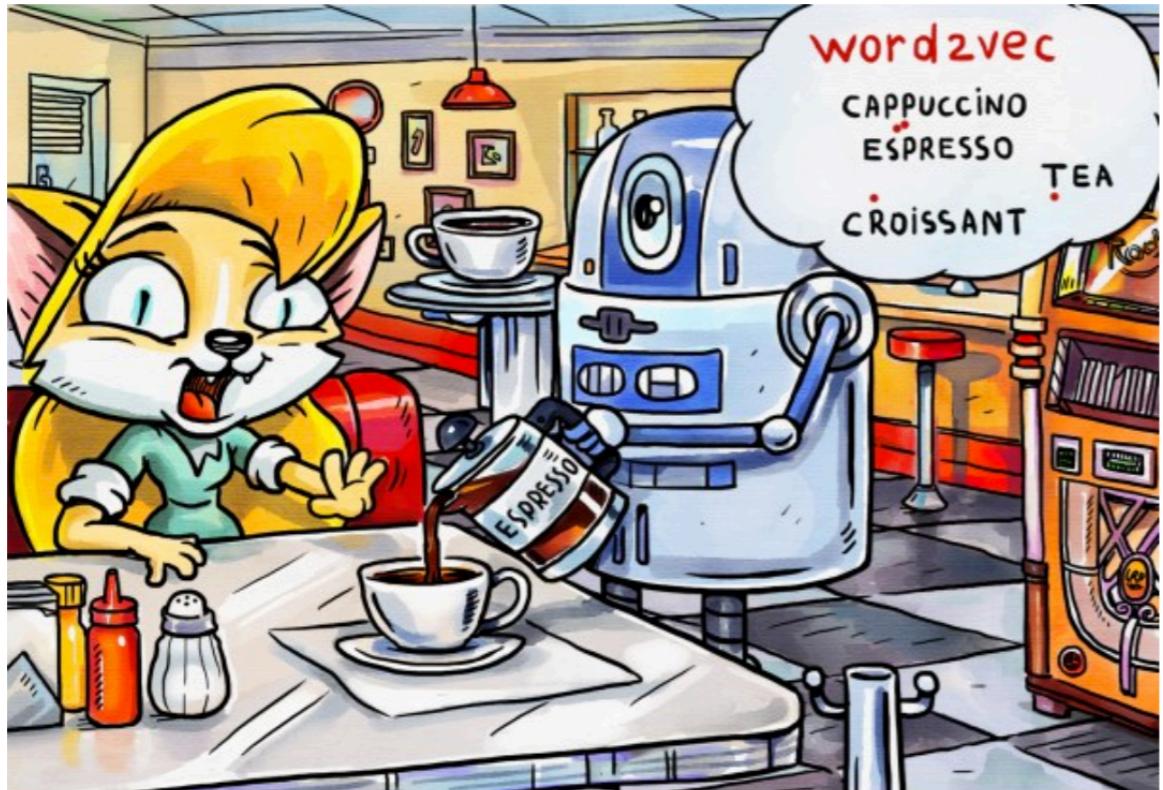


Distributed Bag-Of-Words (DBOW) :

$$p(w_{i-h}, \dots, w_{i+h} \mid d)$$



# Résumé



- Espresso? But I ordered a cappuccino!  
- Don't worry, the cosine distance between them is so small  
that they are almost the same thing.

**word embedding :**  
• **word2vec :**  
**CBOW, skip-gram**

**doc embedding :**  
• **doc2vec :**  
**DM, DBOW**

**classification de textes avec  
ces embeddings**

auteur : Dmitry Malkov, data scientist et dessinateur de BD chez Data Monsters

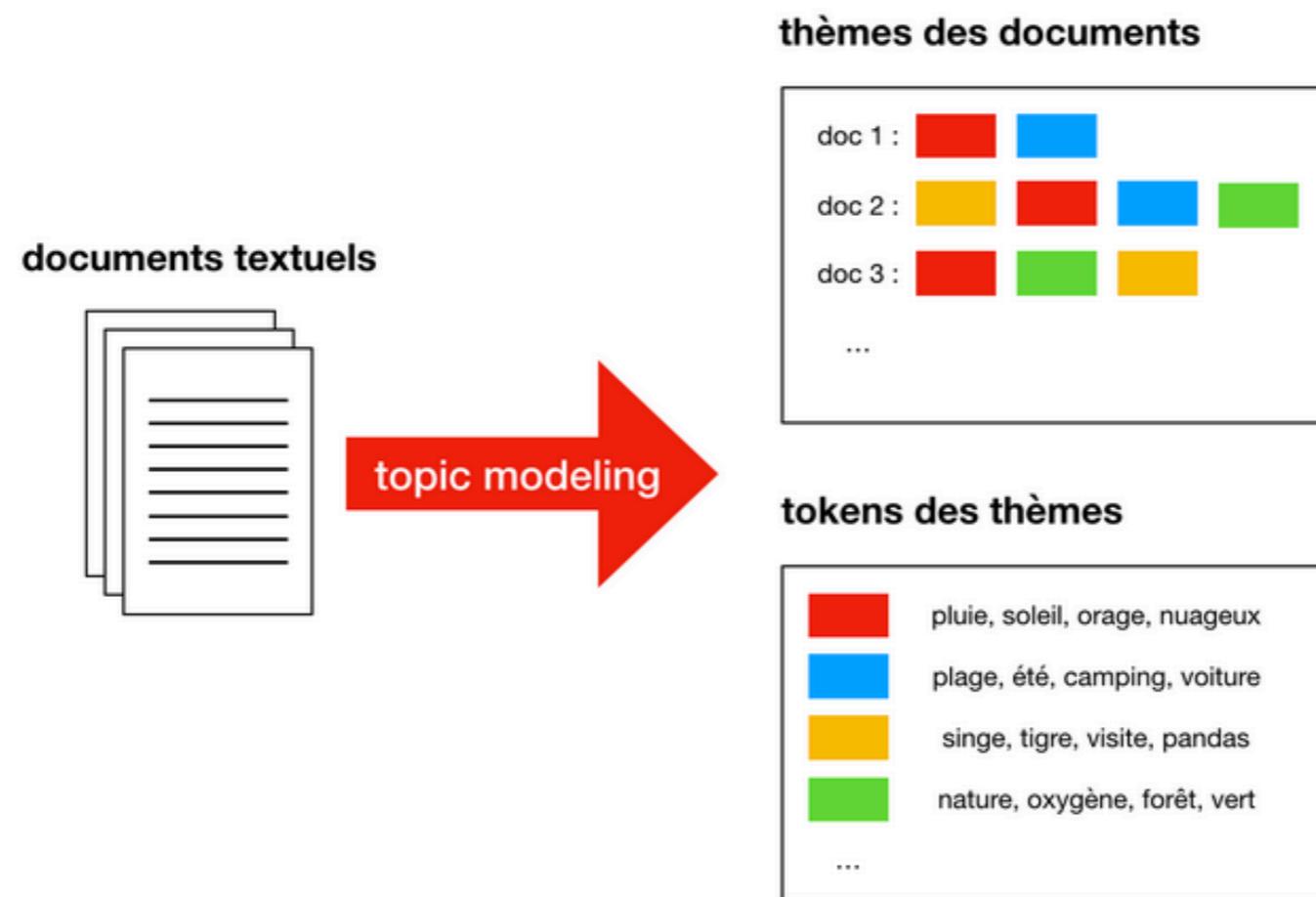
### **3. Topic modeling**

# Représentation vectorielle des documents / topics

**Topic modeling** (modèle thématique en français) : processus d'identification des topics (ou thèmes) permettant de décrire un ensemble de documents

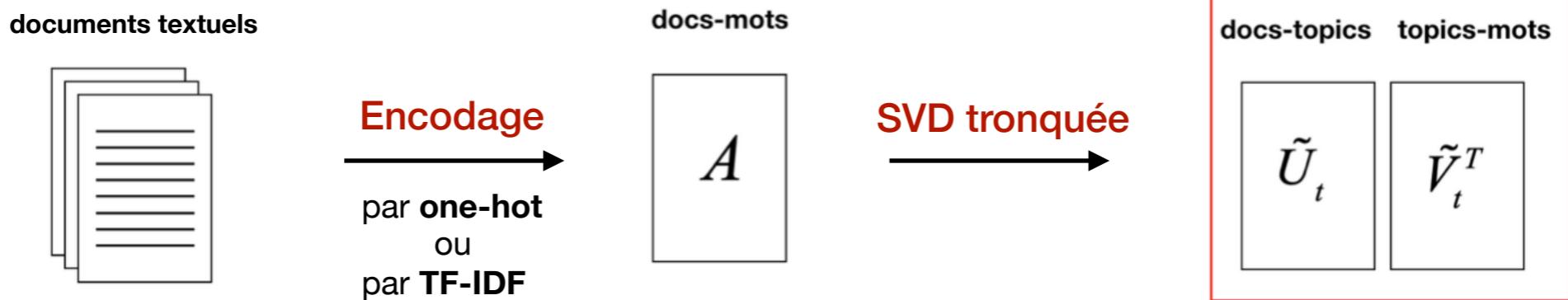
**Hypothèses :**

- chaque **document** est vu comme un **mélange de topics**
- chaque **topic** est vu comme une **collection de mots**

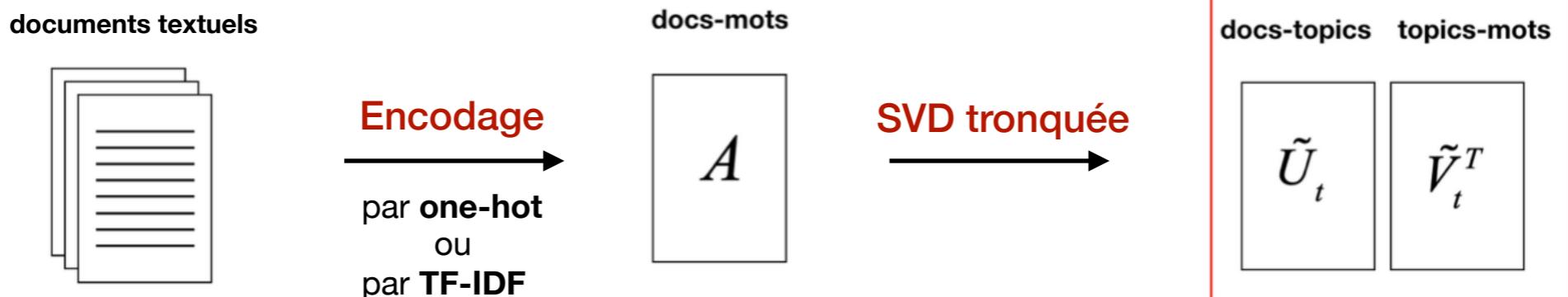


**Topic models** les plus populaires : **LSA, PLSA, LDA**

# Latent Semantic Analysis (1/2)



# Latent Semantic Analysis (1/2)



## Décomposition en Valeurs Singulières (SVD) :

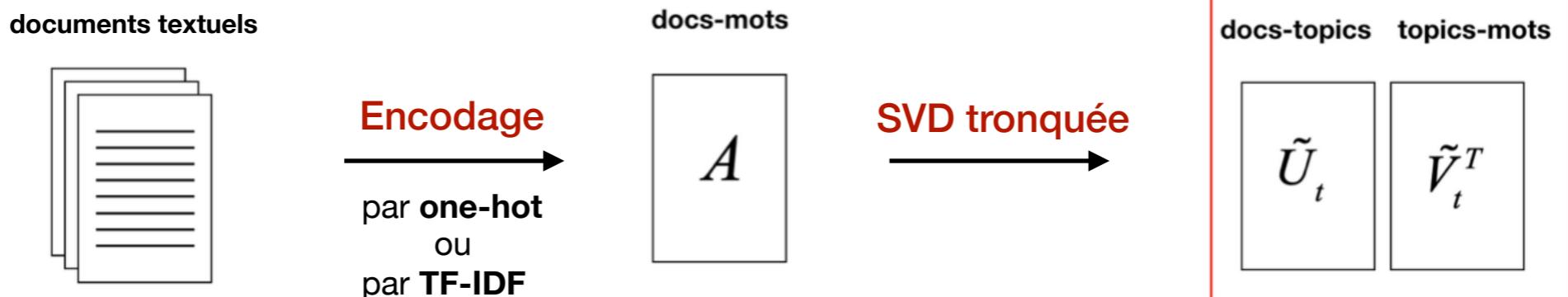
factorisation matricielle

$$A = U \times S \times V^T$$

matrice unitaire      matrice diagonale  
(valeurs singulières de A)  
ps : on les range par ordre décroissant

Dimensions des matrices :  
 $A: n \times d$ ,  $U: n \times n$ ,  $S: n \times d$ ,  $V^T: d \times d$  (matrice unitaire)

# Latent Semantic Analysis (1/2)



## Décomposition en Valeurs Singulières (SVD) :

factorisation matricielle

$$A = U \times S \times V^T$$

matrice unitaire      matrice diagonale  
(valeurs singulières de A)  
ps : on les range par ordre décroissant

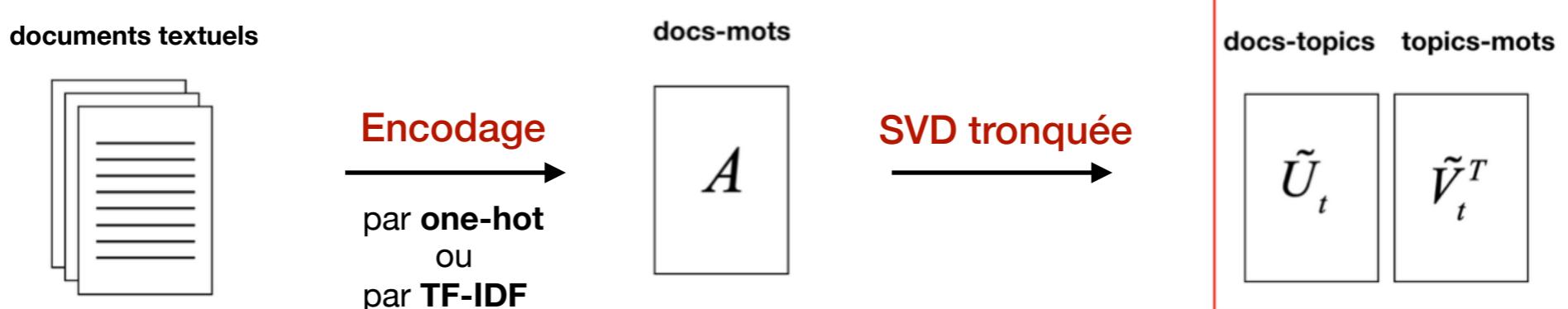
Dimensions:  $A: n \times d$ ,  $U: n \times n$ ,  $S: n \times d$ ,  $V^T: d \times d$

## SVD tronquée :

$$A \approx U_t \times S_t \times V_t^T$$

Dimensions:  $A: n \times d$ ,  $U: n \times n$ ,  $S: n \times d$ ,  $V^T: d \times d$

# Latent Semantic Analysis (1/2)



## Décomposition en Valeurs Singulières (SVD) :

factorisation matricielle

$$A = U \times \begin{matrix} \dots \\ S \end{matrix} \times V^T$$

$A$

$n \times d$

$U$

$n \times n$

matrice unitaire

$S$

$n \times d$

matrice diagonale  
(valeurs singulières de  $A$ )

$V^T$

$d \times d$

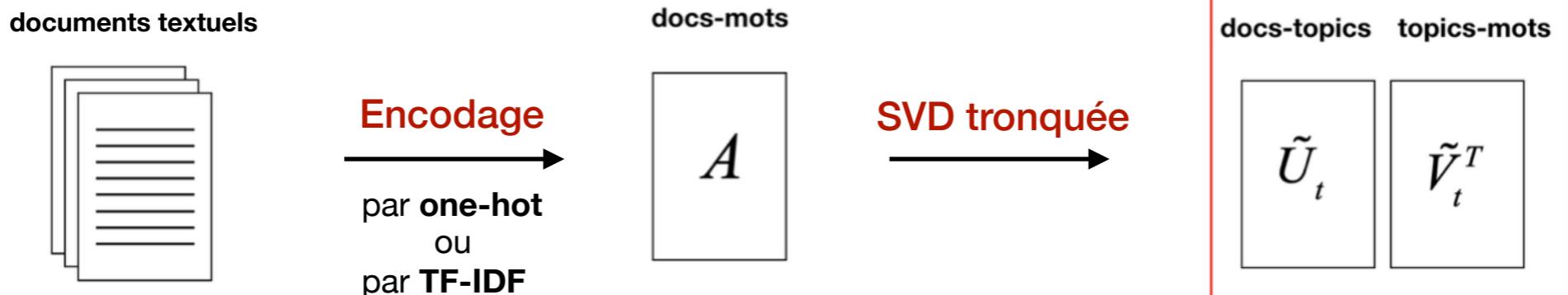
matrice unitaire

# SVD tronquée :

$$A \approx U_{n \times t} \times S_{t \times t} \times V_t^T_{t \times d}$$

## Comment choisir $\tilde{U}_t$ et $\tilde{V}_t^T$ ?

# Latent Semantic Analysis (1/2)



## Décomposition en Valeurs Singulières (SVD) :

factorisation matricielle

$$A = U \times S \times V^T$$

$n \times d$        $n \times n$        $n \times d$       matrice unitaire  
 matrice unitaire      matrice diagonale  
 (valeurs singulières de A)  
 ps : on les range par ordre décroissant

## SVD tronquée :

$$A \approx U_t \times S_t \times V_t^T$$

$n \times d$        $n \times t$        $t \times t$        $t \times d$   
 $\tilde{U}_t = U_t \times S_t$        $\tilde{V}_t^T = V_t^T$

Comment choisir  $\tilde{U}_t$  et  $\tilde{V}_t^T$  ?

$$A \approx U_t \times S_t \times V_t^T$$

$n \times d$        $n \times t$        $t \times t$        $t \times d$   
 $\tilde{U}_t = U_t \times \sqrt{S_t}$        $\tilde{V}_t^T = V_t^T \times \sqrt{S_t}$

# Latent Semantic Analysis (2/2)

## Remarques :

- approche **algèbre linéaire**
- évaluer la similarité entre les **documents** (similarité cosinus)
- évaluer la similarité entre les **mots** (similarité cosinus)

## Désavantages :

- vecteurs difficilement **interprétable**
- besoin d'un **grand ensemble de docs et de vocabulaires** pour obtenir une bonne précision

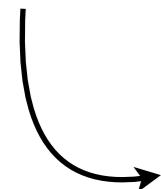
# Latent Semantic Analysis (2/2)

Remarques :

- approche **algèbre linéaire**
- évaluer la similarité entre les **documents** (similarité cosinus)
- évaluer la similarité entre les **mots** (similarité cosinus)

Désavantages :

- vecteurs difficilement **interprétable**
- besoin d'un **grand ensemble de docs et de vocabulaires** pour obtenir une bonne précision



**Solution : probabilistic LSA (PLSA)**

# Probabilistic LSA (1/3)

**Idée** : trouver un modèle probabiliste avec des topics latents qui peut générer les observations de matrice mots-docs

Formellement :

**Input** : collection de textes **bag-of-words** :

$n_{wd}$  = nombre d'occurrence du mot  $w$  dans le document  $d$

**Trouver** : probabilité que le **mot  $w$**  soit dans un **topic  $t$**  :

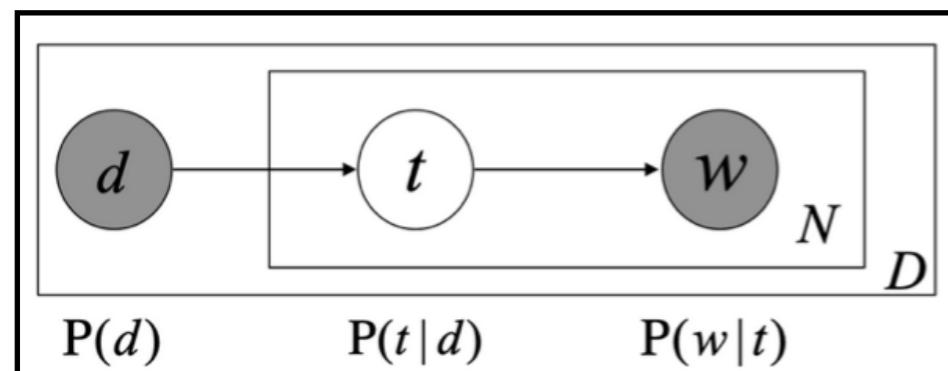
$$\phi_{wt} = p(w|t)$$

probabilité que le **topic  $t$**  soit dans le **document  $d$**  :

$$\theta_{td} = p(t|d)$$

comment est généré un document  $d$  d'après le modèle **PLSA** ?

un document  $d = (w_1, \dots, w_W)$  est généré de la manière suivante : pour tout  $w \in d$

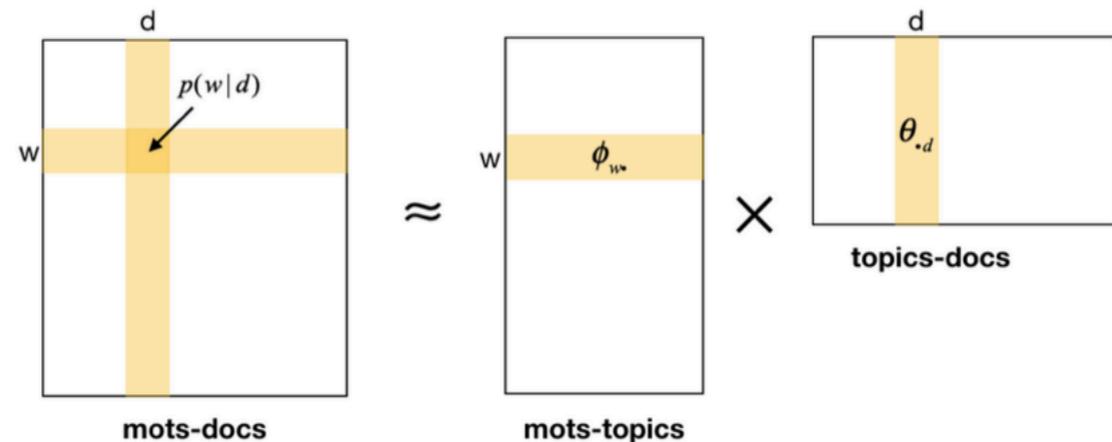


- $t \sim \text{Multinomial}(\theta_{\cdot d}) = \text{Multinomial}(p(t_1|d), p(t_2|d), \dots)$
- $w \sim \text{Multinomial}(\phi_{\cdot t}) = \text{Multinomial}(p(w_1|t), p(w_2|t), \dots)$

# Probabilistic LSA (2/3)

Ecriture probabiliste :  $p(w | d) = \sum_{t \in T} p(w | t, d) \cdot p(t | d) = \sum_{t \in T} p(w | t) \cdot p(t | d) = \sum_{t \in T} \phi_{wt} \cdot \theta_{td}$

Ecriture matricielle :

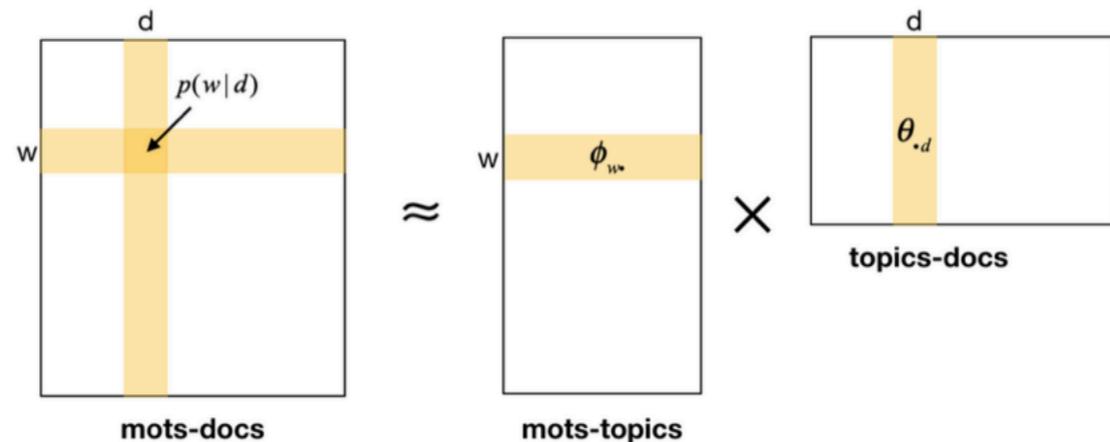


# Probabilistic LSA (2/3)

Ecriture probabiliste :  $p(w | d) = \sum_{t \in T} p(w | t, d) \cdot p(t | d) = \sum_{t \in T} p(w | t) \cdot p(t | d) = \sum_{t \in T} \phi_{wt} \cdot \theta_{td}$

Ecriture matricielle :

PLSA  
≈  
variante probabiliste  
de LSA

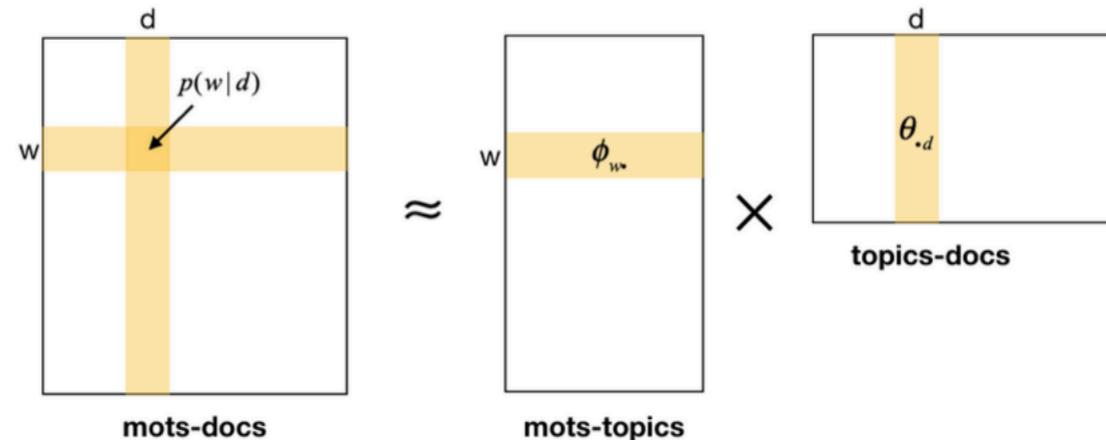


# Probabilistic LSA (2/3)

Ecriture probabiliste :  $p(w | d) = \sum_{t \in T} p(w | t, d) \cdot p(t | d) = \sum_{t \in T} p(w | t) \cdot p(t | d) = \sum_{t \in T} \phi_{wt} \cdot \theta_{td}$

Ecriture matricielle :

PLSA  
≈  
variante probabiliste  
de LSA



Entraînement du modèle PLSA : posons  $\Theta = [\dots \theta_{ij} \dots]$  et  $\Phi = [\dots \phi_{ij} \dots]$

- calcul de la **vraisemblance** (objectif maximiser ce terme) :

$$L(\Theta, \Phi) = \prod_d \prod_w p(d, w)^{n_{wd}} = \prod_d p(d) \prod_w p(w|d)^{n_{wd}}$$

- calcul de la **log-vraisemblance** (objectif maximiser ce terme) :

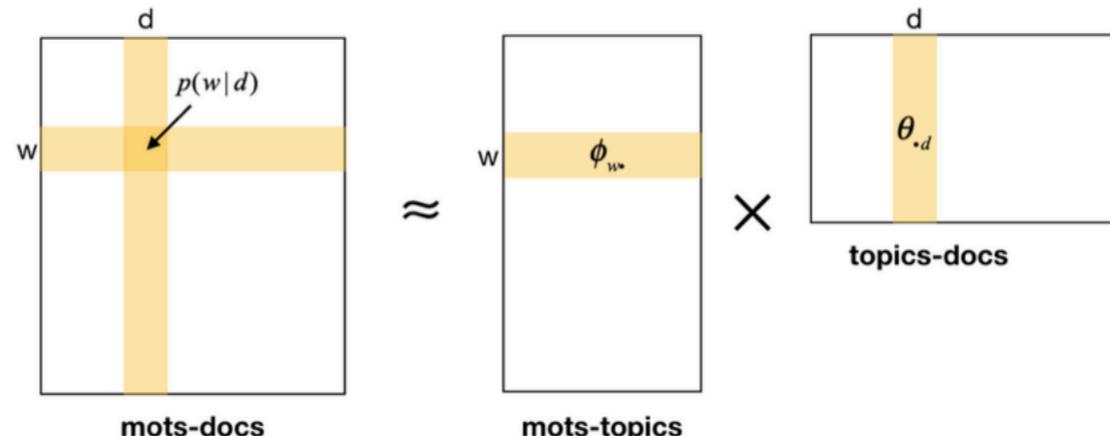
$$\mathcal{L}(\Theta, \Phi) = \sum_d \sum_w n_{wd} \log \sum_t \phi_{wt} \cdot \theta_{td}$$

# Probabilistic LSA (2/3)

Ecriture probabiliste :  $p(w | d) = \sum_{t \in T} p(w | t, d) \cdot p(t | d) = \sum_{t \in T} p(w | t) \cdot p(t | d) = \sum_{t \in T} \phi_{wt} \cdot \theta_{td}$

Ecriture matricielle :

PLSA  
≈  
variante probabiliste  
de LSA



Entraînement du modèle PLSA : posons  $\Theta = [\dots \theta_{ij} \dots]$  et  $\Phi = [\dots \phi_{ij} \dots]$

- calcul de la **vraisemblance** (objectif maximiser ce terme) :

$$L(\Theta, \Phi) = \prod_d \prod_w p(d, w)^{n_{wd}} = \prod_d p(d) \prod_w p(w|d)^{n_{wd}}$$

problème :  $\log \sum_t$  dur à maximiser

- calcul de la **log-vraisemblance** (objectif maximiser ce terme) :

$$\mathcal{L}(\Theta, \Phi) = \sum_d \sum_w n_{wd} \log \sum_t \phi_{wt} \cdot \theta_{td}$$

solution : [algorithme EM](#)

# Latent Dirichlet Allocation (1/2)

**PLSA** : topic modeling basé sur une inférence **fréquentiste** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **déterministes**

**LDA** : topic modeling basé sur une inférence **bayésienne** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **aléatoires** latentes de la distribution Dirichlet :

$$\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$$

$$Dir(\theta_d | \alpha) = \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \quad Dir(\phi_t | \beta) = \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1}$$

# Latent Dirichlet Allocation (1/2)

**PLSA** : topic modeling basé sur une inférence **fréquentiste** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **déterministes**

**LDA** : topic modeling basé sur une inférence **bayésienne** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **aléatoires** latentes de la distribution Dirichlet :

$$\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$$

$$Dir(\theta_d | \alpha) = \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \quad Dir(\phi_t | \beta) = \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1}$$



LDA  
≈  
extension du PLSA

# Latent Dirichlet Allocation (1/2)

**PLSA** : topic modeling basé sur une inférence **fréquentiste** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **déterministes**

**LDA** : topic modeling basé sur une inférence **bayésienne** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **aléatoires** latentes de la distribution Dirichlet :

$$\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$$

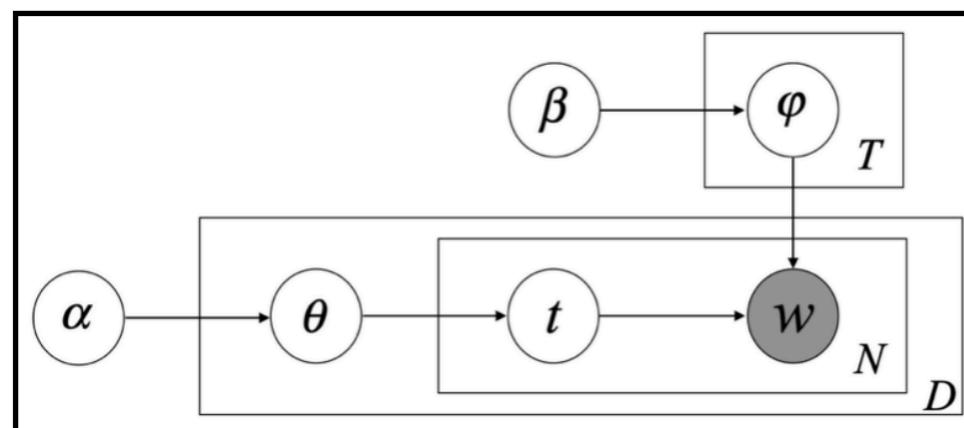
$$Dir(\theta_d | \alpha) = \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \quad Dir(\phi_t | \beta) = \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1}$$



**LDA**  
≈  
extension du PLSA

comment est généré un document  $d$  d'après le modèle **LDA** ?

un document  $d = (w_1, \dots, w_N)$  est généré de la manière suivante : pour tout  $w \in d$



- $\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$
- $t \sim Multinomial(\theta_{.d}), \quad w \sim Multinomial(\phi_{.t})$

# Latent Dirichlet Allocation (1/2)

**PLSA** : topic modeling basé sur une inférence **fréquentiste** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **déterministes**

**LDA** : topic modeling basé sur une inférence **bayésienne** :  $\phi_t = (\phi_{wt})_{w \in W}$  et  $\theta_d = (\theta_{td})_{t \in T}$  sont des variables **aléatoires** latentes de la distribution Dirichlet :

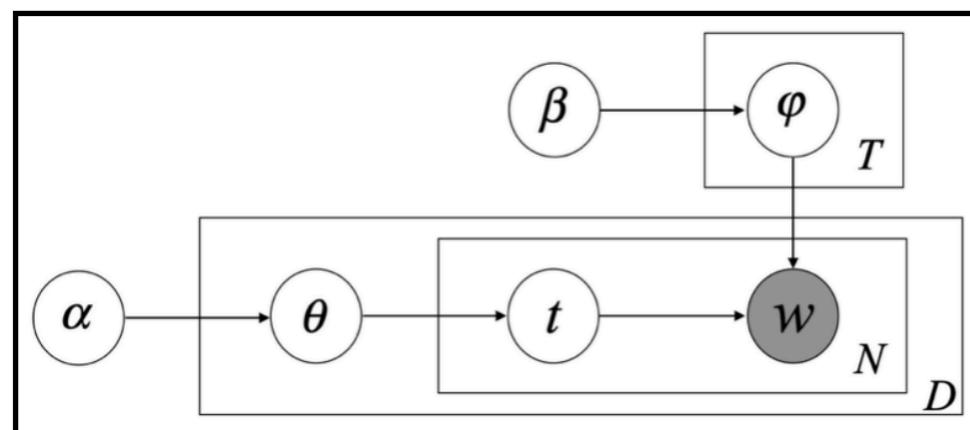
$$\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$$

$$Dir(\theta_d | \alpha) = \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_{td}^{\alpha_t - 1}, \quad Dir(\phi_t | \beta) = \frac{\Gamma(\sum_w \beta_w)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1}$$

LDA  
≈  
extension du PLSA

comment est généré un document  $d$  d'après le modèle **LDA** ?

un document  $d = (w_1, \dots, w_N)$  est généré de la manière suivante : pour tout  $w \in d$

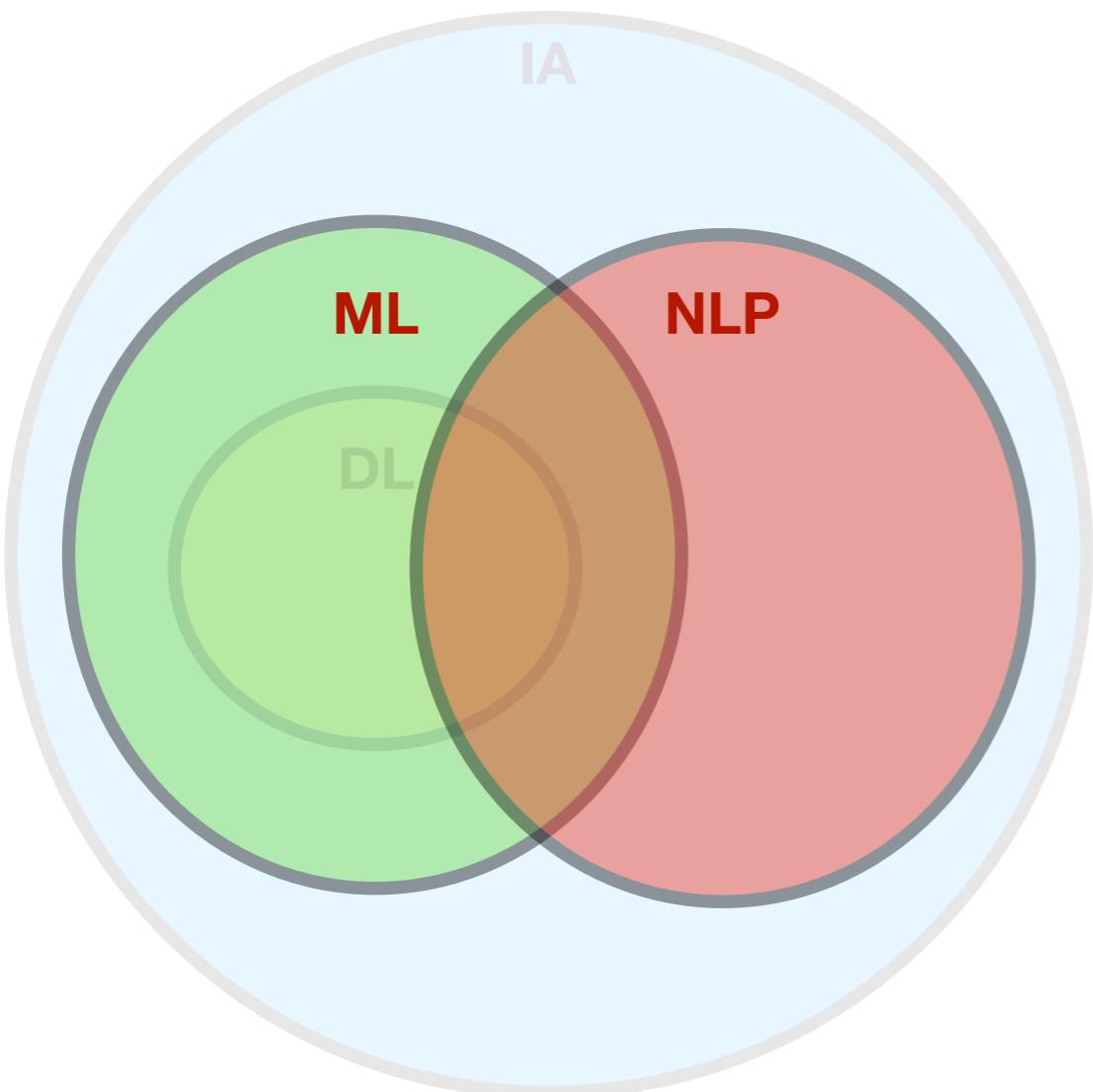


- $\theta_d \sim Dir(\alpha), \quad \phi_t \sim Dir(\beta)$
- $t \sim Multinomial(\theta_{.d}), \quad w \sim Multinomial(\phi_{.t})$

question : comment estimer les variables aléatoires  $\theta_d$  et  $\phi_t$  ?

réponse : [Gibbs sampling](#) (ou échantillonnage de Gibbs en français)

# Résumé



## Topics modeling

### Topic models :

- LSA,  
une approche algèbre linéaire
- PLSA,  
une inférence fréquentiste
- LDA,  
une inférence bayésienne

TP sur le topic modeling