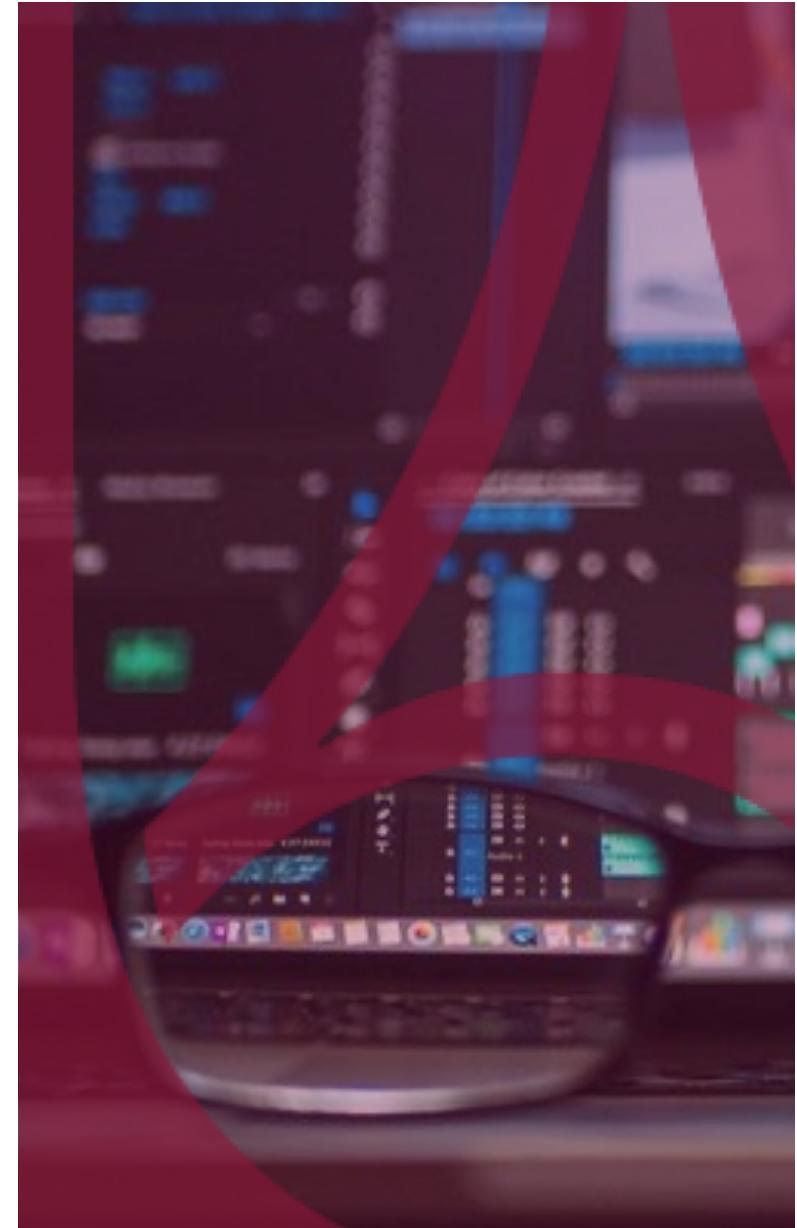


Traitement Automatique de la Langue Naturelle

M2 Data Science • Année 2020 / 2021

Marie CANDITO - Antoine SIMOULIN



Ressources



<https://moodle.u-paris.fr/course/view.php?id=11048>



<https://github.com/AntoineSimoulin/m2-data-sciences>



<m2midsunivdeparis.slack.com>

Icon made by [Freepik](#), [Becris](#), [Smashicons](#), [Pixel perfect](#), [srip](#), [Adib](#), [Flat Icons](#), [Vitaly Gorbachev](#) from www.flaticon.com

2

Séance #7



Rappel séances précédentes

Systèmes Symboliques vs Statistiques

Les systèmes statistiques d'appuient sur un **corpus d'exemples** pour apprendre la relation entre les labels recherchés et les représentations du texte.

Ils ne supposent pas d'expertise linguistique.

Ils permettent de résoudre une **multitude de cas d'usages** avec un pipeline de traitements unifié.

Ils permettent généralement d'obtenir de meilleures performances mais sont moins intelligibles.



Vectorisation du texte

Nous avons vu **deux méthodes de vectorisation** du texte. On représente un document $d \in D$ du corpus par un vecteur e de taille N la taille du vocabulaire. Chacune des coordonnées i du vecteur correspond à un mot w_i du vocabulaire.

Dans la représentation **Bag-Of-Word (BoW)**, chaque coordonnée du vecteur correspond à tf , la **fréquence du mot dans le document** $f(w_i) = tf(w_i)$.

La distribution des mots empiriques suit la **loi de Zipf**. Si on classe les mots en fonction de leur fréquence d'apparition dans le corpus, cette dernière décroît selon une loi $\propto \frac{1}{N}$.

Pour attribuer moins de poids au mots récurrents mais porteurs de peu d'information, on utilise le TF-IDF. chaque coordonnée du vecteur est données par $f(w_i) = tf(w_i) \times idf(w_i)$ avec $idf(w_i) = \log\left(\frac{N}{df_{w_i}}\right)$ et df_{w_i} le nombre de documents du corpus où apparaît le mot.



Classification

Une fois le texte vectorisé, il est possible d'utiliser tous les outils du machine learning pour exploiter les jeux de données.

De nombreux problèmes peuvent être ramenés à des **tâches de classification ou de régression** (détection de Spams dans les emails, analyse de sentiments dans les commentaires utilisateur ...)

Nous avons analysé les effets des pré-traitements et notamment de la **standardisation du texte** (suppression des accents, stemmatisation, lemmatisation, suppression des majuscules ...) et des **paramètres de la vectorisation** (taille du vocabulaire, filtres de fréquence, méthode de tokenization ...)



Exploration de Topics

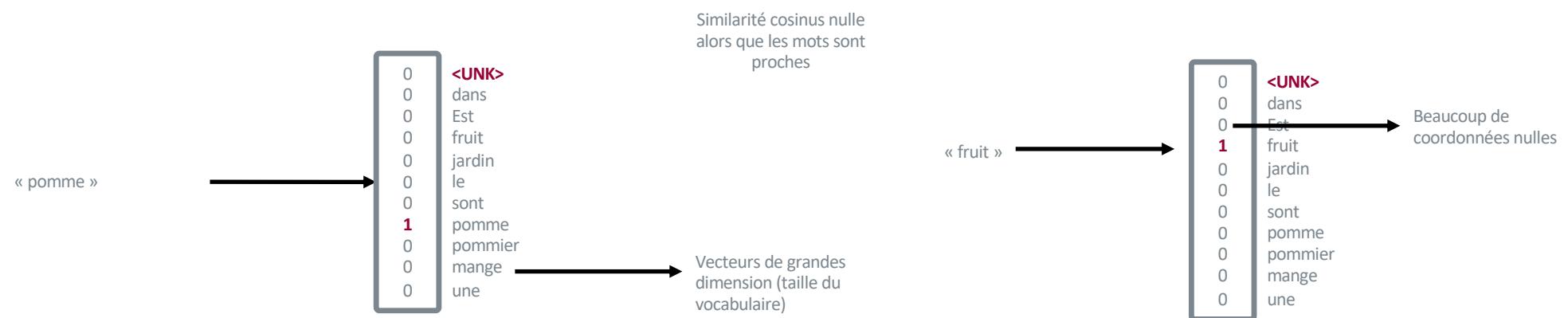
En traitement automatique du langage, on a souvent accès à **d'importants corpus mais souvent sans label** (par exemple des flux twitter)

L'exploration de topics permet de faire ressortir les thèmes récurrents dans un corpus. Nous avons en particulier détaillé l'algorithme de **Llatent Dirichlet Allocation (LDA)** qui introduit une variable latente : les topics. Les documents sont alors représentés comme une **distribution sur les topics**. Les topics sont eux mêmes représentés comme une distribution sur les mots. Ceci suppose **d'interpréter les topics** en fonction de la forme de la distribution.

Inconvénients des représentations BoW



Pour certaines applications, comme la traduction automatique ou la génération de texte, la modélisation BoW ou TF-IDF n'est pas assez fine. Elle ne prend en particulier pas en compte l'ordre des mots. On va s'intéresser maintenant à la **vectorisation des mots**.



Inconvénients des représentations BoW et TF-IDF



Les représentations BoW et TF-IDF sont des vecteurs de **grande dimension** : supérieure à 10 000.



En pratique, il peut être plus pratique d'utiliser une représentation de **dimension plus faible** comprise entre 50 et 300.

Ces vecteurs sont également « **sparse** », avec beaucoup d'entrées vides.



Ces nouvelles représentations seront des **vecteurs denses aussi appelés distribués**.

La sémantique des mots comme la **notion de synonyme ne sont pas exploitées**.



Par ailleurs, Obtenir une représentation qui conserve les **propriétés caractéristiques du « sens » des mots et une distance sémantique**.

Représentation des mots par leur contexte



Hypothèse de sémantique distributionnelle [1, 2]

Le sens d'un mot est déterminé par les mots qui apparaissent dans son contexte.



On cherche à créer des représentations vectorielles des mots qui assurent que deux mots avec un contexte similaire soient représentés par des vecteurs similaires. On va apprendre ces représentations sur la base d'importants corpus de texte dont on exploite la structure.

[1] Firth, J.R. (1957). *A synopsis of linguistic theory* 1930-1955. In *Studies in Linguistic Analysis*, pp. 1-32. Oxford: Philological Society. Reprinted in F.R. Palmer (ed.), *Selected Papers of J.R. Firth 1952-1959*, London: Longman (1968).

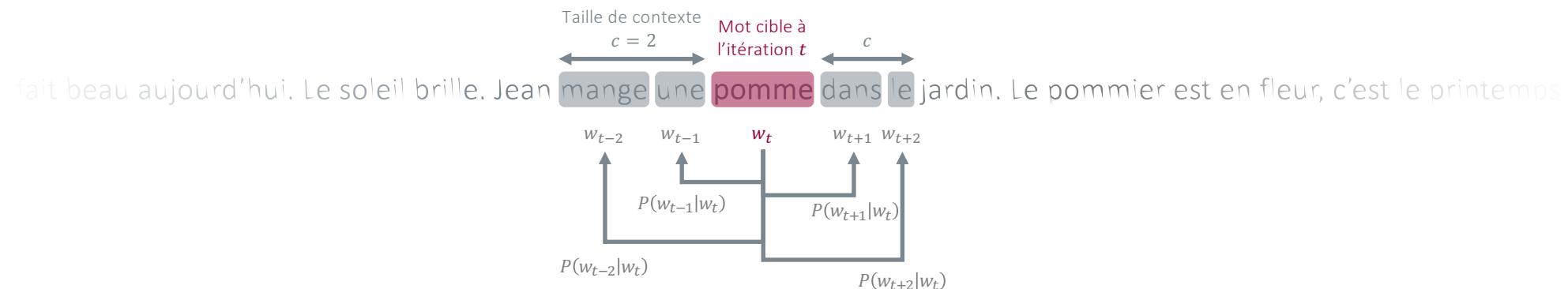
[2] Harris, Z. (1954). *Distributional structure*. *Word*, 10(23): 146-162.



Deux implémentations de Word2Vec : le modèle Skip-Gram

Principe Modèle Skip-Gram

Ensemble de modèles d'apprentissage des embeddings à partir du contexte des mots. En parcourant l'ensemble du corpus avec une fenêtre de taille fixe : on cherche à prédire la probabilité $P(c|w)$ d'un contexte c sachant le mot w (*modèle Skip-gram*)





Deux implémentations de Word2Vec : le modèle Skip-Gram

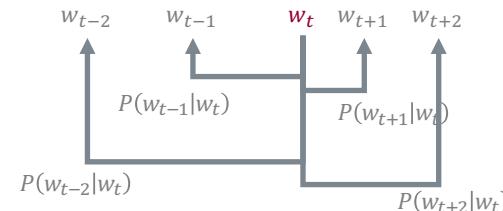
Principe Modèle Skip-Gram

Ensemble de modèles d'apprentissage des embeddings à partir du contexte des mots. En parcourant l'ensemble du corpus avec une fenêtre de taille fixe : on cherche à prédire la probabilité $P(c|w)$ d'un contexte c sachant le mot w (*modèle Skip-gram*)

Mot cible à
l'itération
 $t + 1$

c c

fait beau aujourd'hui. Le soleil brille. Jean mange une pomme dans le jardin. Le pommier est en fleur, c'est le printemps

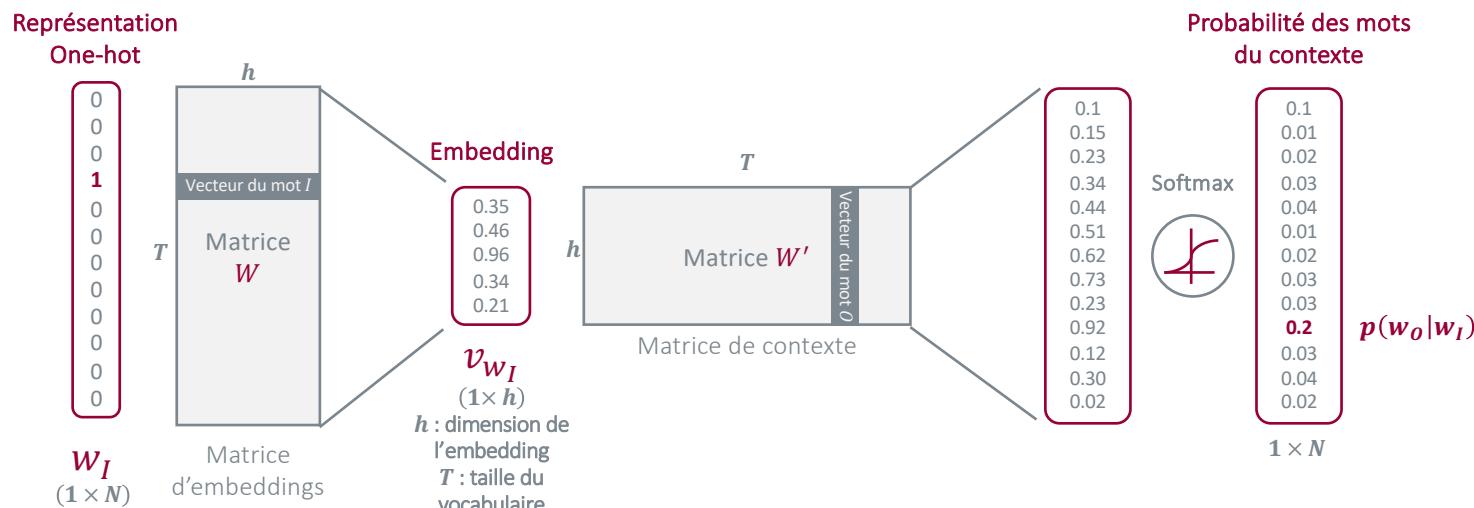




Le modèle Word2Vec

Architecture - Modèle de Deep-Learning avec une couche cachée pour :

- Prédire le contexte en fonction du vecteur one-hot de représentation du mot
- Prédire le mot en fonction des vecteurs one-hot du contexte



[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean: **Distributed Representations of Words and Phrases and their Compositionality**. NIPS 2013: 3111-3119

[2] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: **Efficient Estimation of Word Representations in Vector Space**. ICLR (Workshop Poster) 2013



La fonction de coût de Word2Vec (cas Skip-gram)

On cherchera à **maximiser la log-vraisemblance** par descente de gradient :

$$J(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T \sum_{-\mathbf{c} \leq j \leq \mathbf{c}, j \neq 0} \log p(w_{t+j} | w_t)$$

T le nombre de mot dans le corpus
c la taille de la fenêtre
 w_t le mot à l'emplacement t
 θ les paramètres du modèle à optimiser

Avec $p(w_{t+j} | w_t)$ définit par la fonction **softmax** :

$$p(w_O | w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^T \exp({v'_{w}}^T v_{w_I})}$$

v_w and v'_w sont les projections de w par les matrices \mathbf{W} et \mathbf{W}'



Hierarchical Softmax

On évalue $\log_2 V$ nœuds en moyenne au lieu de V (taille du vocabulaire).

Negative Sampling

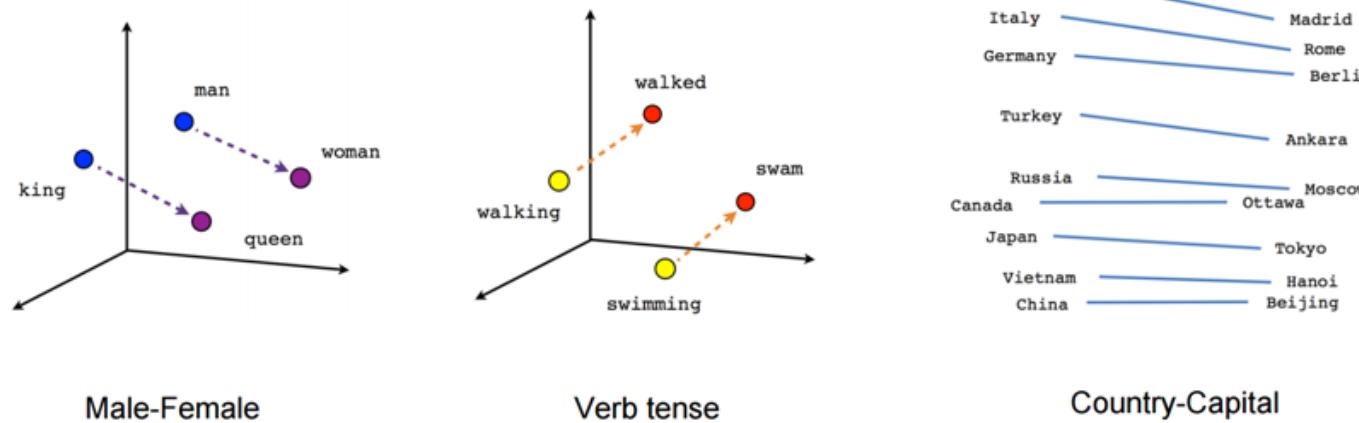
Au lieu de mettre à jour tous les poids, on met à jour les poids de l'exemple positif et de quelques exemples négatifs (typiquement 5, plus petit pour les gros datasets).

La probabilité de sélection dépend de la fréquence $P(w_i) = f(w_i)^{3/4} / \sum f(w_i)^{3/4}$

Subsampling

Chaque mot w_i est conservé avec une proba $P(w_i) = \frac{1 + \sqrt{f(w_i)/t}}{f(w_i)/t}$ où t est le seuil (typiquement 0,001) et f la fréquence d'apparition.

Objectifs

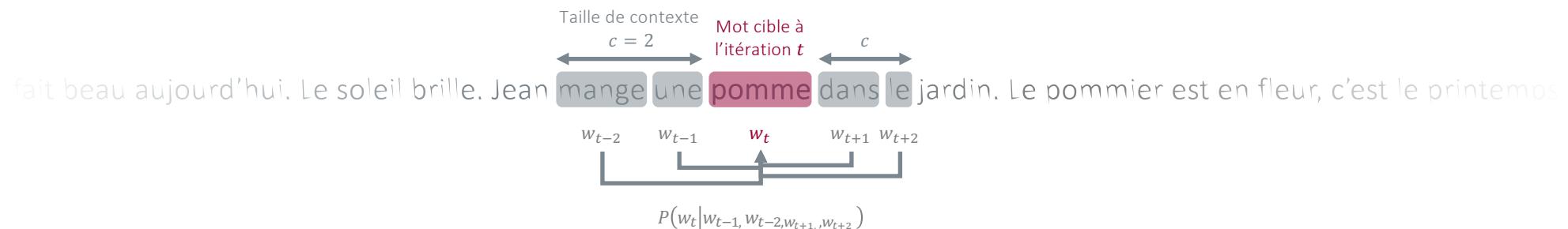




Deux implémentations de Word2Vec : le modèle CBow

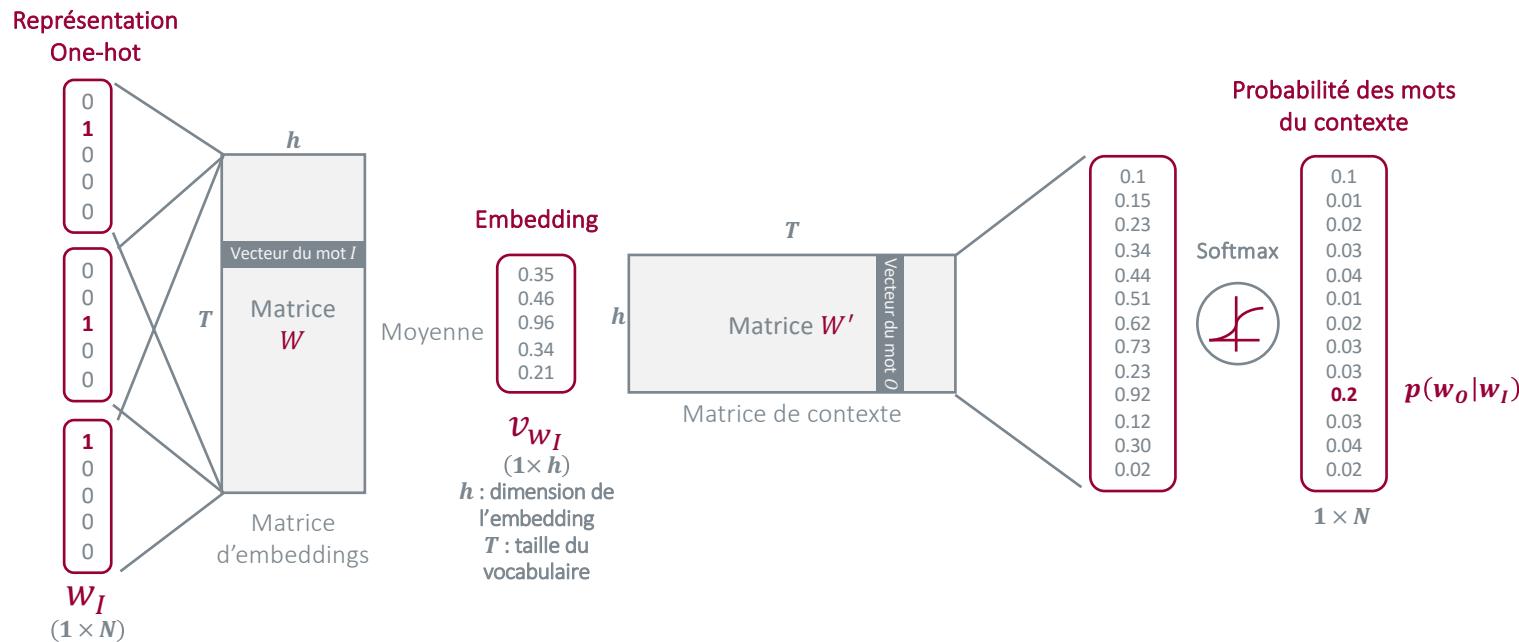
Principe Modèle CBow

Ensemble de modèles d'apprentissage des embeddings à partir du contexte des mots. En parcourant l'ensemble du corpus avec une fenêtre de taille fixe : on cherche à prédire la probabilité $P(c|w)$ d'un contexte c sachant le mot w (*modèle Skip-gram*)





Deux implémentations de Word2Vec : le modèle CBow



GloVe: Global Vectors



Le modèle **GloVe** ^[1] cherche à combiner une **factorisation de la matrice de cooccurrence** des mots et le modèle Skip-Gram.

Dans l'hypothèse de la sémantique distributionnelle, le sens des mots est caractérisé par leur fréquence de cooccurrence. On ne cherche plus à évaluer $p(w_o|w_I)$ mais plutôt à définir une **probabilité de cooccurrence** :

$$p_{co}(w_k|w_i) = \frac{C(w_i, w_k)}{C(w_i)}$$

Avec $C(w_i, w_k)$ une mesure de la cooccurrence des mots w_i et w_k .

[1] Jeffrey Pennington, Richard Socher, Christopher D. Manning: **Glove: Global Vectors for Word Representation**. EMNLP 2014: 1532-1543

Mesurer la distribution de co-occurrence des mots



On définit la **co-occurrence** comme le nombre de fois où **deux mots apparaissent ensemble dans une fenêtre de 5 mots.**

Jean mange une pomme
Les pommes sont sur le pommiers
Le pommier est dans le jardin
Camille s'assoit dans le jardin

→

	fruit	jardin	pomme	mange
fruit	-	0.1	0.2	0.1
jardin	0.5	-	0.1	0.3
pomme	0.2	0.1	-	0.25
mange	0.1	0.3	0.25	-

Embedding sémantique de « *jardin* » [1]

$X_{pomme,jardin}$

[1] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, Richard A. Harshman: **Indexing by Latent Semantic Analysis**. J. Am. Soc. Inf. Sci. 41(6): 391-407 (1990)

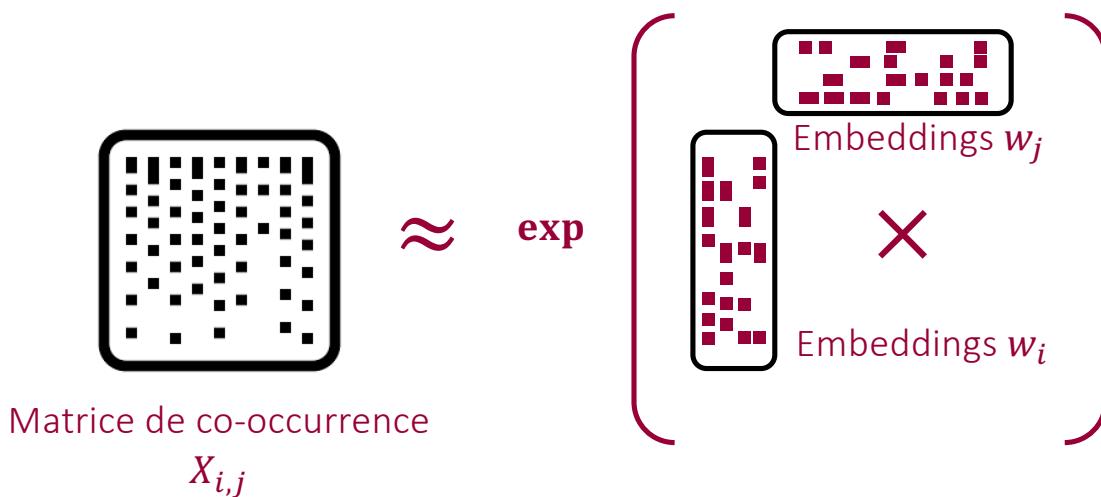
[2] Omer Levy, Yoav Goldberg: Linguistic Regularities in Sparse and Explicit Word Representations. CoNLL 2014: 171-180

Limites de cette approche : La fréquence des mots rare n'est pas caractéristique de l'association entre deux mots. Inversement, certains termes sont très fréquents comme « le » mais ne sont pas très discriminatifs. On peut employer d'autres métriques de co-occurrence comme la PPMI [2] Les représentations sont de grandes dimension et sparse, on peut donc les factoriser.

Méthodes avec statistiques de comptage



L'objectif est de trouver des matrices de dimension moindre où chaque ligne correspond au vecteur associé à chaque mot. En pratique on va « apprendre » les matrices en minimisant une fonction objectif. On choisit d'approcher les coordonnées de la matrice de cooccurrence tel que $p_{co}(w_j|w_i) \approx \exp(w_i^T w_j)$





Fonction de coût de GloVe (résumé)

On cherche ainsi à **minimiser la fonction objectif suivante** :

$$J(\boldsymbol{\theta}) = \sum_{i,j} f(X_{ij})(\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j - \log(X_{ij}))^2$$

b_i et b_j des termes de biais associés aux mots i et j respectivement.

$f(x) = \min\left(1, \left(\frac{x}{x_{max}}\right)^\alpha\right)$ une fonction de pondération pour corriger les termes de cooccurrence trop rares et donc bruités ainsi que ceux trop prépondérants (ex: « c' est »)

X_{ij} la matrice de cooccurrence des mots

Remarques sur Word2Vec



Les représentations sont obtenues à partir de **méthodes « semi-supervisées »** :

Il n'y a pas besoin de données labélisées, des représentations peuvent donc être entraînées à partir de n'importe quel corpus suffisamment conséquent !



Permet d'obtenir une représentation de **dimension plus faible** qui possède des propriétés caractéristiques du « sens » des mots



Les embeddings obtenus dépendent :

- De l'architecture employée : Skip Gram ou C bow
- Du corpus d'entraînement : la quantité de mots présent et la qualité
- De la taille de fenêtre choisie : 10 pour Skip-Gram et 5 pour C-bow généralement



Ces représentations sont **particulièrement adaptées pour les réseaux de neurones** : représentations vectorielles denses



Il existe des **implémentations particulièrement efficaces** des trois principales méthodes Word2Vec, FastText et Glove qui permettent d'obtenir rapidement des représentations compatibles pour toute application et réseau de neurones.

Les limites de Word2Vec



Les types de relations capturées entre les mots sont floues :

Par exemple **les synonymes et les antonymes auront tendance à avoir des représentations proches** car ils sont souvent présents dans un contexte proche. Exemple : « aimer » et « détester ».



Il sera difficile d'obtenir des embeddings pour les mots rares ou présentant des fautes d'orthographies car ceux-ci sont généralement affectés à « UNK » lors de la constitution du dictionnaire ou ceux-ci ne présentent pas assez d'exemple pour obtenir des embeddings de bonnes qualités



Une fois les embeddings entraînés il est impossible d'ajouter de nouveaux mots au dictionnaire, tout nouveau mot rencontré sera considéré comme un « UNK »



Les représentations obtenues **ne peuvent pas être composées**. Il est **difficile de construire une représentation pertinente de la phrase** en fonction des représentations des mots qui la constituent dans le cas général.



Chaque mot admet **une seule et unique représentation**. Par exemple le mot « avocat » qui peut désigner le fruit et la profession sera représenté par un unique vecteur dense, une sorte de moyenne de deux sens. Cette limitation est en partie adressée par la récente méthode « **ELMO** » qui **tient compte du contexte pour proposer une représentation**.

La similarité cosinus



A l'instar des vectorisation BoW ou TF-IDF, on peut Calculer une distance entre les documents en utilisant la similarité cosinus (« cosine similarity »). Pour rappel, en considérant deux vecteurs \vec{d}_1 et \vec{d}_2 formant un angle θ , leur produit scalaire peut s'écrire :

$$\vec{d}_1 \cdot \vec{d}_2 = \| \vec{d}_1 \| \| \vec{d}_2 \| \cos \theta$$

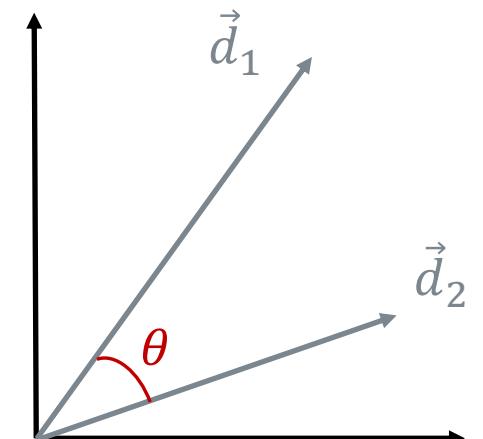
La similarité cosinus correspond à la valeur de $\cos \theta$

On a donc :

$$sim(d_1, d_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{\| \vec{d}_1 \| \| \vec{d}_2 \|} \in [-1, 1]$$

Si $|sim(d_1, d_2)| = 1$ les vecteurs sont colinéaires et donc « similaires »

Si $sim(d_1, d_2) = 0$ les vecteurs sont orthogonaux et donc « indépendants »





gensim

```
>>> from gensim.test.utils import common_texts, get_tmpfile
>>> from gensim.models import Word2Vec
>>>
>>> path = get_tmpfile("word2vec.model")
>>>
>>> model = Word2Vec(common_texts, size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```

```
>>> model = Word2Vec.load("word2vec.model")
>>> model.train([["hello", "world"]], total_examples=1, epochs=1)
(0, 2)
```

```
>>> vector = model.wv['computer'] # numpy vector of a word
```

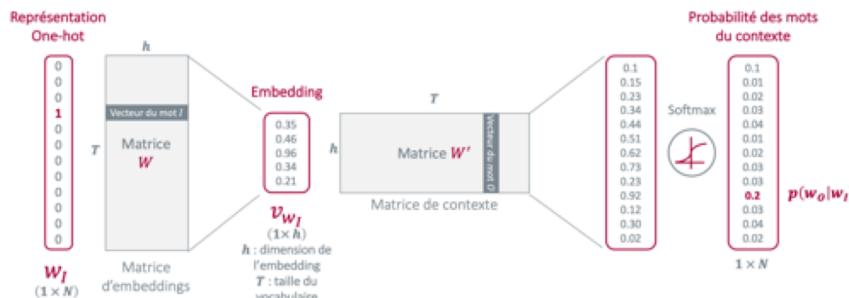


Modèle Skip-Gram

En fonction du mot cible, le modèle apprend à prédire chacun des mots du contexte un par un.

On cherche à prédire la probabilité $P(c|w)$ d'un contexte c sachant le mot w

- Taille de la fenêtre : autour de 10
- Plus lent
- Meilleurs résultats sur les mots rares
- Meilleurs avec des jeux de données de faible taille



Modèle C-Bow (Continuous Bag-of-Words)

Le modèle prédit le mot cible en fonction de l'ensemble des mots du contexte

On cherche à prédire la probabilité $P(c|w)$ d'un contexte c sachant le mot w

- Taille de la fenêtre : autour de 5
- Plus rapide
- Meilleure précision sur les mots fréquents

