Traitement du langage naturel

IFT6758 - Science des données

Sources:

http://demo.clab.cs.cmu.edu/NLP/

http://u.cs.biu.ac.il/~89-680/

And many more …

Mila

Université de Montréal

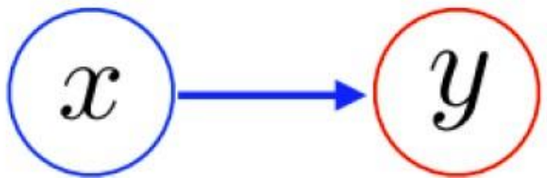
Annonce

- Corrections et notes disponible le 22 Novembre (normalement)
- **IFT 6758A:**
  - Lundi 18: aide pour l'étape 2
  - Présentation de l'étape 2 le 2 Décembre (+ heures de bureau sur l'étape 3 en parallèle)

Rappel
(probabilité 101)

- Joint probability $p(x, y)$
- Conditional probability $p(x|y)$
- Marginal probability $p(x)$ and $p(y)$

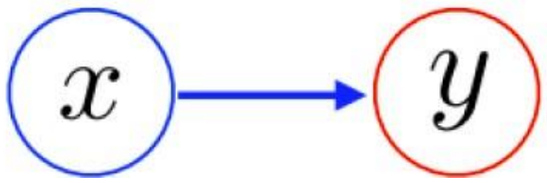
- They are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$



Rappel
(probabilité 101)

- Joint probability $p(x, y)$
- Conditional probability $p(x|y)$
- Marginal probability $p(x)$ and $p(y)$

- They are related by $p(x, y) = p(x|y)p(y) = p(y|x)p(x)$



Chain Rule:

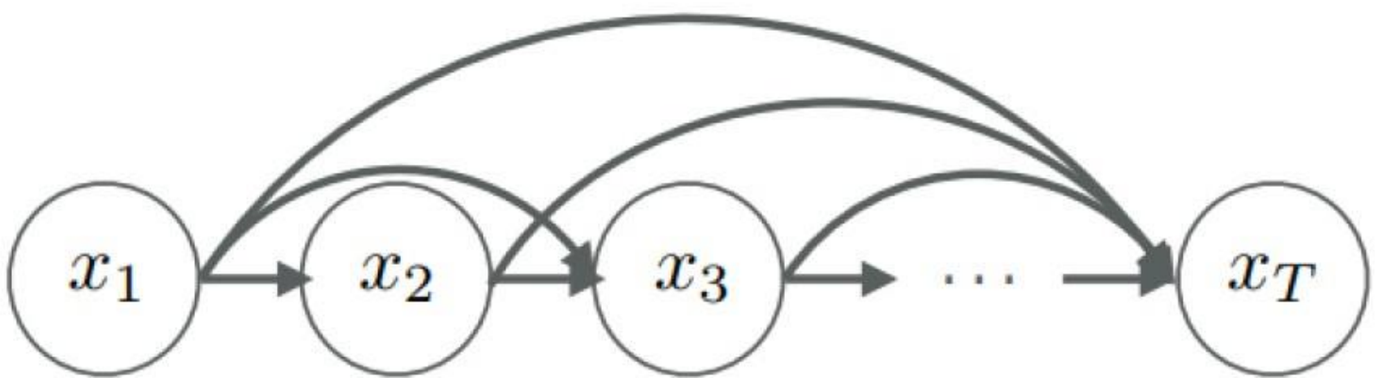
$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

Modèles de langage probabilistes

- Rewrite $p(x_1, x_2, \ldots, x_T)$ into

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

- Graphically,



Example

p(the,mouse,ate,the,cheese)=p(the)
p(mouse|the)
p(ate|the,mouse)
p(the|the,mouse,ate)
p(cheese|the,mouse,ate,the).

Température:

i=1,…,L. $x_i \propto p(x_i|x_{1:i-1})^{1/T}$ (La somme des probabilités est 1)

- T=0: Déterministe (échantillone le mot le plus probable)
- T=1: “Normal”.
- T=∞: Distribution uniforme.

Example: p(cheese)=0.4, p(mouse)=0.6

$p_{T=0.5}$(cheese)=0.31, $p_{T=0.5}$(mouse)=0.69

$p_{T=0.2}$(cheese)=0.12, $p_{T=0.2}$(mouse)=0.88

$p_{T=0}$(cheese)=0, $p_{T=0}$(mouse)=1

Mila

Université de Montréal

Différents modèles autorégressif

1) Modèles non-parametriques

(gros tableaux, n-grammes)

2) Modèles paramétriques

(gros réseaux de neurones)

Mila

Université de Montréal

Modèle de langage N-gramme

Hypothèse de Markov: le mot présent ne dépend que des **n-1** mots précédents. Exemple de n+1-gramme:

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

$$\approx \prod_{t=1}^{T} p(x_t | x_{t-n}, \ldots, x_{t-1})$$

Comment calculer de telles probabilités: (comptage)

Bigrammes: $$P(x_t | x_{t-1}) = \frac{\text{count}(x_{t-1}, x_t)}{\text{count}(x_{t-1})}$$

Exemple de n-gramme

Pour les Bigrammes:

$$P\left(x_t|x_{t-1}\right)=\frac{\text{count}\left(x_{t-1},x_t\right)}{\text{count}\left(x_{t-1}\right)}$$

<s> I am sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I}\,|\,\texttt{<s>})=\tfrac{2}{3}=.67 \qquad P(\texttt{Sam}\,|\,\texttt{<s>})=\tfrac{1}{3}=.33 \qquad P(\texttt{am}\,|\,\texttt{I})=\tfrac{2}{3}=.67$$
$$P(\texttt{</s>}\,|\,\texttt{Sam})=\tfrac{1}{2}=0.5 \qquad P(\texttt{Sam}\,|\,\texttt{am})=\tfrac{1}{2}=.5 \qquad P(\texttt{do}\,|\,\texttt{I})=\tfrac{1}{3}=.33$$

Effets de n dans la performance

- Ex) $p(\text{i}, \text{would}, \text{like}, \text{to}, \ldots, ., \langle/\text{s}\rangle)$
- Unigram Modelling

  $p(\text{i})p(\text{would})p(\text{like}) \cdots p(\langle/\text{s}\rangle)$
- Bigram Modelling

  $p(\text{i})p(\text{would}|\text{i})p(\text{like}|\text{would}) \cdots p(\langle/\text{s}\rangle\,|.)$
- Trigram Modelling

  $p(\text{i})p(\text{would}|\text{i})p(\text{like}|\text{i}, \text{would}) \cdots$

  ⋮

word	unigram	bigram	trigram	4-gram
i	6.684	3.197	3.197	3.197
would	8.342	2.884	2.791	2.791
like	9.129	2.026	1.031	1.290
to	5.081	0.402	0.144	0.113
commend	15.487	12.335	8.794	8.633
the	3.885	1.402	1.084	0.880
rapporteur	10.840	7.319	2.763	2.350
on	6.765	4.140	4.150	1.862
his	10.678	7.316	2.367	1.978
work	9.993	4.816	3.498	2.394
.	4.896	3.020	1.785	1.510
</s>	4.828	0.005	0.000	0.000
average	8.051	4.072	2.634	2.251
perplexity	265.136	16.817	6.206	4.758

Évaluation : Quelle est la qualité d'un modèle de langage ?

- **Évaluation extrinsèque :** tester un modèle entraîné sur une collection de test:
  - Essayez de prédire chaque mot
  - Plus un modèle peut prédire les mots avec précision, meilleur est le modèle

- **Évaluation intrinsèque** : Perplexité
  - La perplexité est la probabilité inverse de l'ensemble de tests, normalisée par le nombre de mots:

.

– Étant donné P(.) et un texte test de longueur N

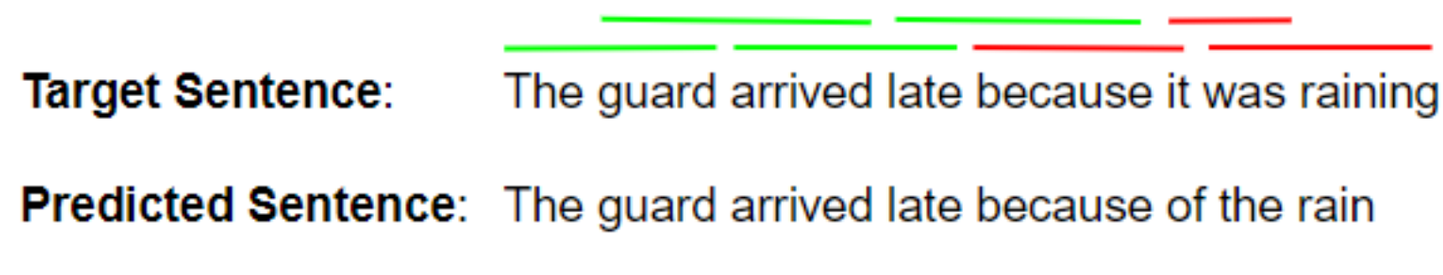
– Plus c'est bas, mieux c'est!

$$Perplexité = 2^{-\frac{1}{N}\sum_{i=1} \log_2 P(w_i)}$$

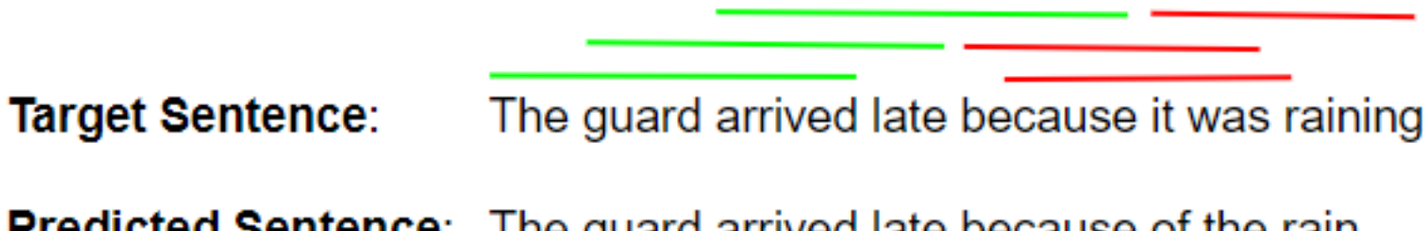
Évaluation: BLEU score

Precision 1-gram (p_1) = 5 / 8

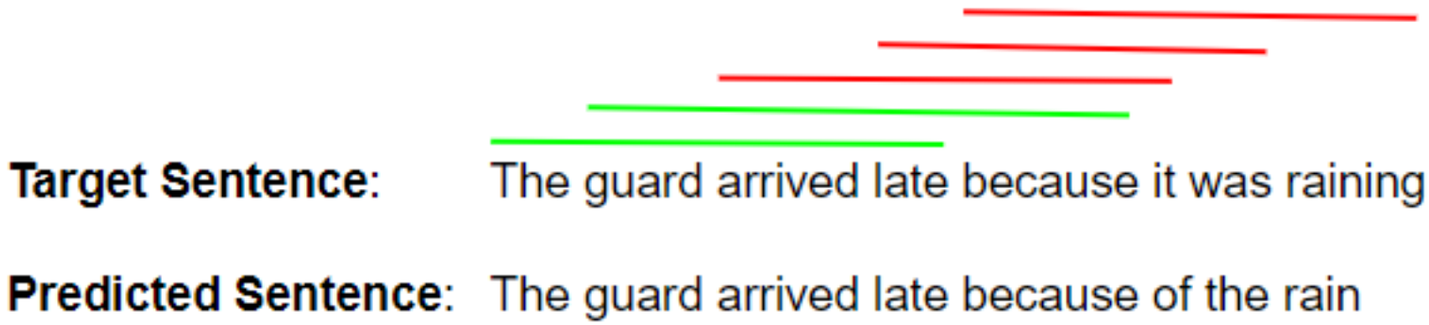
Precision 2-gram (p_2) = 4 / 7



Precision 3-gram (p_3) = 3 / 6



Precision 4-gram (p_4) = 2 / 5



Source: https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b

Quelle est l'astuce?

Precision 1-gram (p_1) = 1

Target Sentence: The

Predicted Sentence: The guard arrived late because of the rain

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

Mila

Université de Montréal

Évaluation: BLEU score

$$
\begin{aligned}
Geometric\ Average\ Precision\ (N) &= exp(\sum_{n=1}^{N} w_n\ log\ p_n) \\
&= \prod_{n=1}^{N} {p_n}^{w_n} \\
&= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}}
\end{aligned}
$$

$$
Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}
$$

$$
Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N)
$$

Source: https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b

Mila

Université de Montréal

Modèle de langage paramétrique

~~Non-parametric estimator~~ $\longrightarrow$ parametric estimator

$$
\begin{aligned}
P\left(x_t | x_{t-n}, \ldots, x_{t-1}\right) &= \frac{\cancel{\text{count}\left(x_{t-n}, \ldots, x_t\right)}}{\cancel{\text{count}\left(x_{t-1}, \ldots, x_{t-1}\right)}} \\
&= f_\Theta\left(x_{t-n}, \ldots, x_{t-1}\right)
\end{aligned}
$$

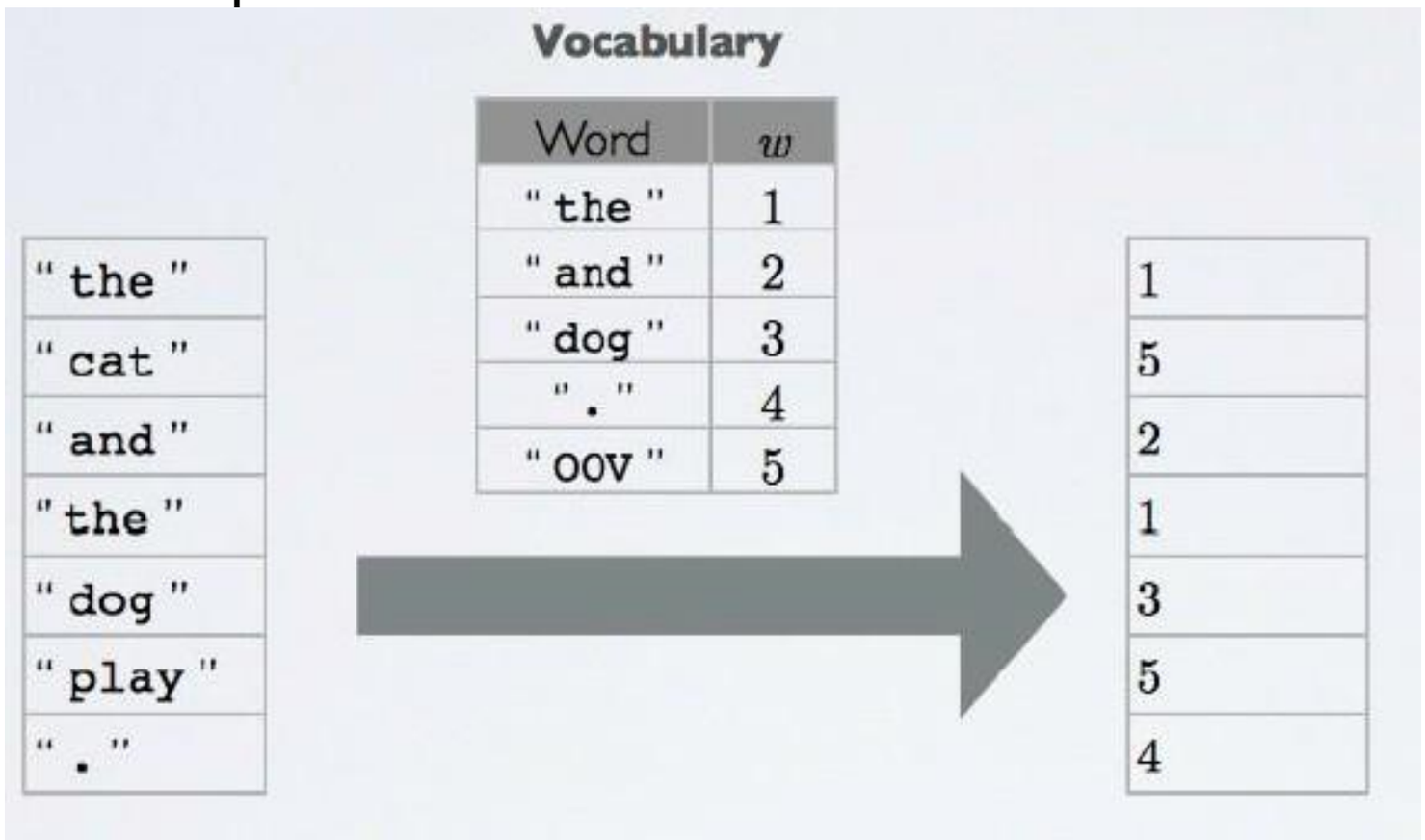
Plongement lexical

- **plongement**: fonction entre un espace avec une dimension par unité linguistique (caractère, morphème, mot, phrase, paragraphe, phrase, document) à un espace vectoriel continu avec une dimension beaucoup plus faible.
- Représentation vectorielle : la « signification » de l'unité linguistique est représentée par un vecteur de nombres réels.

$$\text{good} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Encodage à chaud (one-hot)

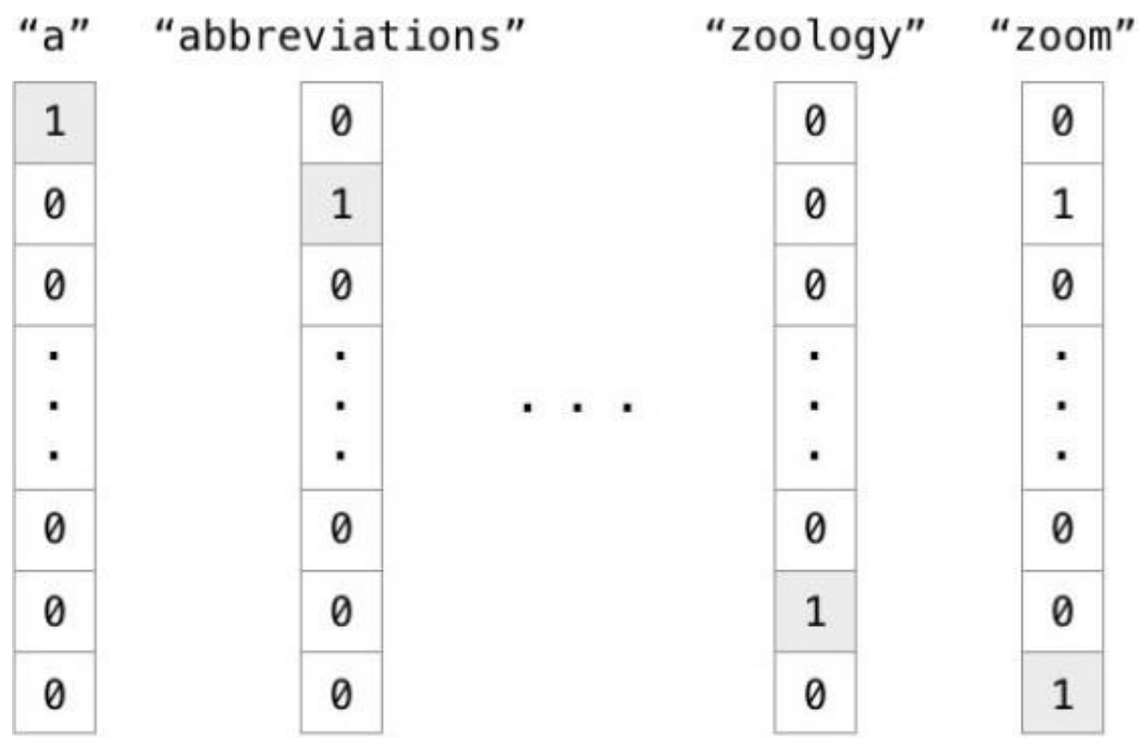
- Technique d'incorporation de mots naïve et simple : associer chaque mot à un identifiant unique



- Les tailles de vocabulaire typiques varient entre 10k et 250k.

Encodage à chaud

- Chaque token à un index
- Étape nécessaire dans tout les cas.
- Le vecteur ‘one hot’ est un vecteur rempli de 0, à l’exception d’un 1 à la position associée à l’ID. Par exemple, pour la taille du vocabulaire D = 10, le vecteur à une chaleur du token (w) ID = 4 est
  e(w) = [0 0 0 1 0 0 0 0 0 0]
- Un encodage à chaud unique ne fait **aucune hypothèse** sur la similitude des tokens et tous les tokens sont également similaires / différents les uns des autres



Encodage à chaud

- **Avantages:** Rapide et simple

- **Inconvénients:** Taille des échelles vectorielles d'entrée avec taille du vocabulaire. Doit prédéterminer la taille du vocabulaire**.**
  - Impossible d'évoluer vers des vocabulaires plus grands ou infinis (loi de Zipf!)
  - Coûteux en calcul - un vecteur d'entrée volumineux entraîne beaucoup trop de paramètres à apprendre.
  - Problème de « **hors-vocabulaire** » (OOV) : Comment géreriez-vous les mots invisibles dans l'ensemble de test ? (Une solution est d'avoir un symbole « UNKNOWN » qui représente des mots à basse fréquence ou invisibles)
  - Aucune relation entre les mots : chaque mot est un vecteur unitaire indépendant

D("cat", "refrigerator") = D("cat", "cats")

D("spoon", "knife") = D("spoon", "dog")

N-grammes

- Vocabulaire = ensemble de tous les n-grammes dans le corpus
- Document = compte des n-grammes dans le document relatifs au vocabulaire.
- Pour le bigramme :

Phrase 1: "The cat sat on the hat"

Phrase 2: "The dog ate the cat and the hat”

Vocab = { the cat, cat sat, sat on, on the, the hat, the dog, dog ate, ate the, cat and, and the}

Phrase 1: { 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}
Phrase 2 : { 1, 0, 0, 0, 0, 1, 1, 1, 1, 1}
Bigrammes: : { 2, 1, 1, 1, 1, 1, 1, 1, 1, 1}

N-grammes

- **Avantages:**
  - Incorpore l’ordre des mots
  - Simple et rapide
- **Inconvénients:**
  - Vocabulaire très large
  - Parcimonie des données

Parcimonie des données

Data sparsity:# of all possible n-grams: $|V|^n$, where $|V|$ is the size of the vocabulary. Most of them never occur.

Training Set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Set:

- ... denied the offer
- ... denied the loan

$$P(\text{offer}|\text{denied the}) = 0$$

N-grammes

- **Avantages:**
  - Incorpore l’ordre des mots
  - Simple et rapide
- **Inconvénients:**
  - Vocabulaire très large
  - Parcimonie des données
  - Fausse présomption d’indépendance

Fausse présomption d'indépendance

On suppose que chaque mot dépend seulement des n-1 précédents

False independence assumption: Because in an n-gram language model we assume that each word is only conditioned on the previous n-1 words

False conditional independence assumption

"The dogs chasing the cat bark". The tri-gram probability $P(\text{bark}|\text{the cat})$ is very low (not observed in the corpus by the model, because the cat never barks and the plural verb "bark" has appeared after singular noun "cat"), but the whole sentence totally makes sentence.

'bark' n'apparaît pas souvent avec comme mots précédent 'the cat' (distracteurs entre 'dogs' et 'bark')

N-grammes

- **Avantages:**
  - Incorpore l’ordre des mots
  - Simple et rapide
- **Inconvénients:**
  - Vocabulaire très large
  - Parcimonie des données
  - Fausse présomption d’indépendance
  - Ne peut capturer la similarité syntaxique/sémantique

L’hypothèse distributionnelle

- L’hypothèse distributionnelle : les mots qui apparaissent dans les mêmes contextes ont tendance à avoir des significations similaires (Harris, 1954)

- “You shall know a word by the company it keeps” (Firth, 1957)

L’hypothèse distributionnelle

Distributional Semantics (*Count*)

- Used since the 90’s
- Sparse word-context PMI/PPMI matrix
- Decomposed with SVD

Word Embeddings (*Predict*)

- Inspired by deep learning
- `word2vec` *(Mikolov et al., 2013)*
- GloVe *(Pennington et al., 2014)*

Underlying Theory: **The Distributional Hypothesis** *(Harris, ’54; Firth, ‘57)*

“Similar words occur in similar contexts”

Apprentissage de plongement denses

Mila

Université de Montréal

Rappel: Modèles de langage probabilistes

- **Objectif :** Calculer la probabilité d'une phrase ou de séquences de mots

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n)$$

- Tâche connexe : probabilité d'un mot à venir :

.

$$P(w_5 \mid w_1, w_2, w_3, w_4)$$

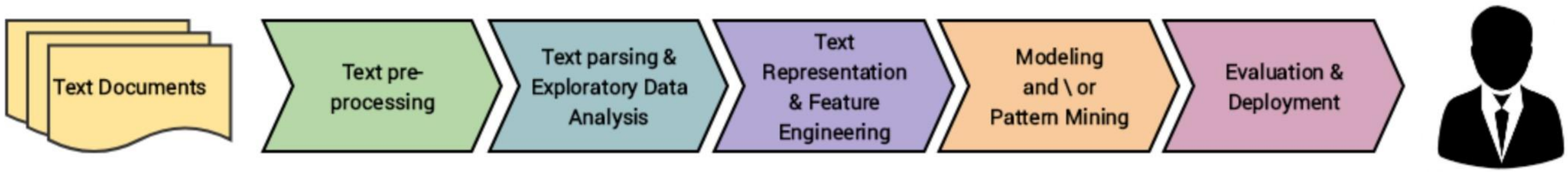
- Un modèle qui calcule l'un des éléments ci-dessus est appelé **modèle de langage.**

Rappel: Modèle de langage paramétrique

~~Non-parametric estimator~~ $\longrightarrow$ parametric estimator

$$
\begin{aligned}
P\left(x_t | x_{t-n}, \ldots, x_{t-1}\right) &= \frac{\cancel{\text{count}}\left(x_{t-n}, \ldots, x_t\right)}{\cancel{\text{count}}\left(x_{t-1}, \ldots, x_{t-1}\right)} \\
&= f_\Theta\left(x_{t-n}, \ldots, x_{t-1}\right)
\end{aligned}
$$

Représentation

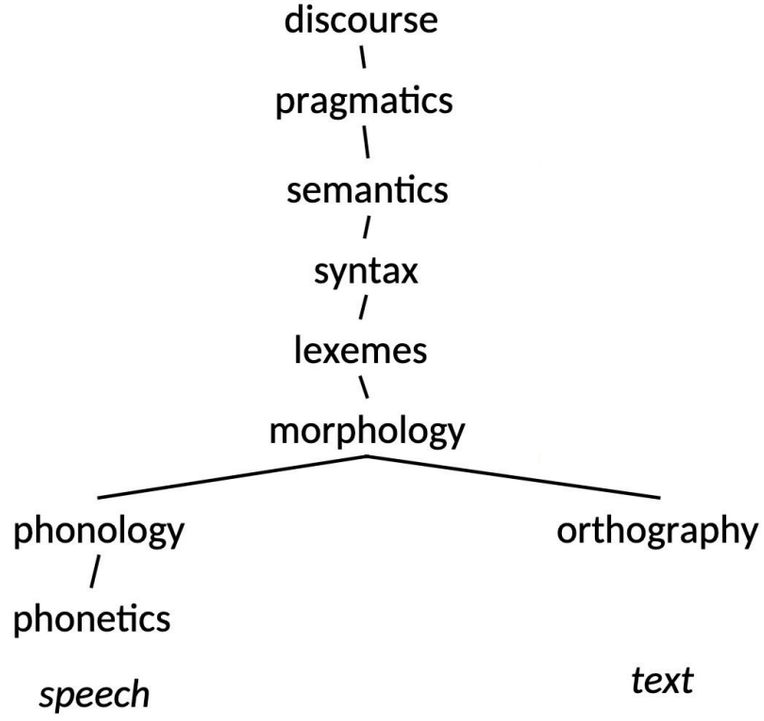


A high-level standard workflow for any NLP project

- - Nous pouvons représenter des objets dans différents niveaux hiérarchiques :
  - -Documents
  - Phrases
  - Mots

- Nous voulons que la représentation soit interprétable et facile à utiliser

- **La représentation vectorielle répond à ces exigences**



Représentation Classique des mots en NLP

- Problèmes avec la représentation vectorielle classique:

  - **Énorme –** chacun de dimension | V| (la taille du vocabulaire ~)
  - **Pas de similarité entre les mots**

- We want our vectors to be small and dense:

~~[0 1 0 0 0 0 0 0 0 0]~~ [0.315 0.136 0.831]

- **Les mots similaires ont des vecteurs similaires:** Capturez la similitude sémantique et morphologique de sorte que les caractéristiques des mots « similaires » sont « similaires » (= leurs vecteurs sont proches les uns des autres dans l'espace vectoriel)

Apprentissage de plongements denses

Matrix Factorization

Factorize word-context matrix.

	$Context_1$	$Context_1$	....	Context k
$Word_1$				
$Word_2$				
⋮				
$Word_n$				

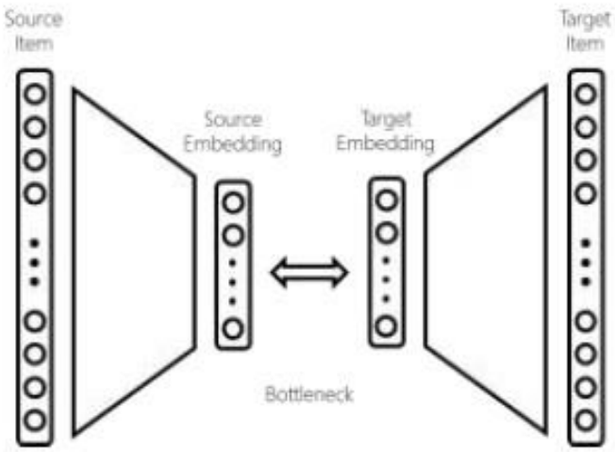
E.g.,

LDA (Word-Document),

GloVe (Word-NeighboringWord)

Neural Networks

A neural network with a bottleneck, word and context as input and output respectively.



E.g.,

Word2vec (Word-NeighboringWord)

Deerwester, Dumais, Landauer, Furnas, and Harshman, Indexing by latent semantic analysis, JASIS, 1990.
Pennington, Socher, and Manning, GloVe: Global Vectors for Word Representation, EMNLP, 2014.
Mikolov, Sutskever, Chen, Corrado, and Dean, Distributed representations of words and phrases and their compositionality, NIPS, 2013.

Plongement de mots via SVD

Word2Vec

Deux représentations

On représente à quelle fréquence un mot apparaît dans un document:

- Matrice mots-documents (ou mot-contexte)

Ou alors à quelle fréquence une paire de mots apparaît:

- Matrice mot-mot (ou co-occurrence)

Matrice mot-document

Documents

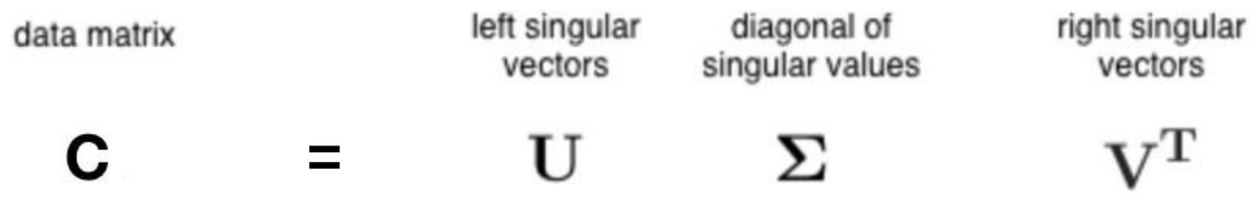
Words

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

- Cette matrice est la base pour calculer la **similitude** entre les documents et les mots
- Cette représentation est utilisée dans la recherche d'informations pour calculer la similarité entre les requêtes et les documents

Matrice mot-document

- Nous allons décomposer la matrice mot-document en un produit de matrices.



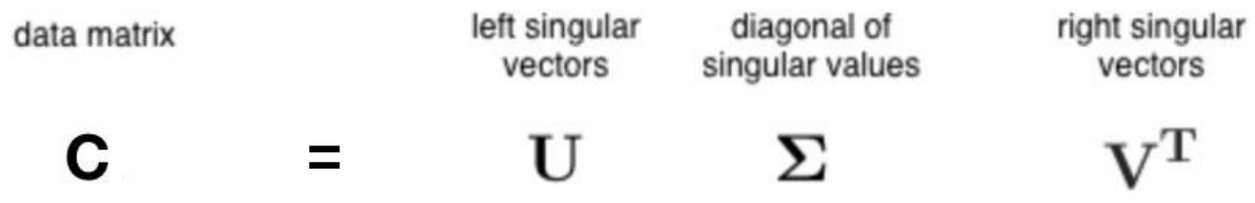
- La décomposition particulière que nous utiliserons : décomposition en valeurs singulières (SVD)

- Nous utiliserons ensuite la SVD pour calculer une nouvelle matrice de document terminologique améliorée C’.

- Nous obtiendrons de meilleures valeurs de similarité de C’ (par rapport à C).

Matrice mot-document

- Nous allons décomposer la matrice mot-document en un produit de matrices.

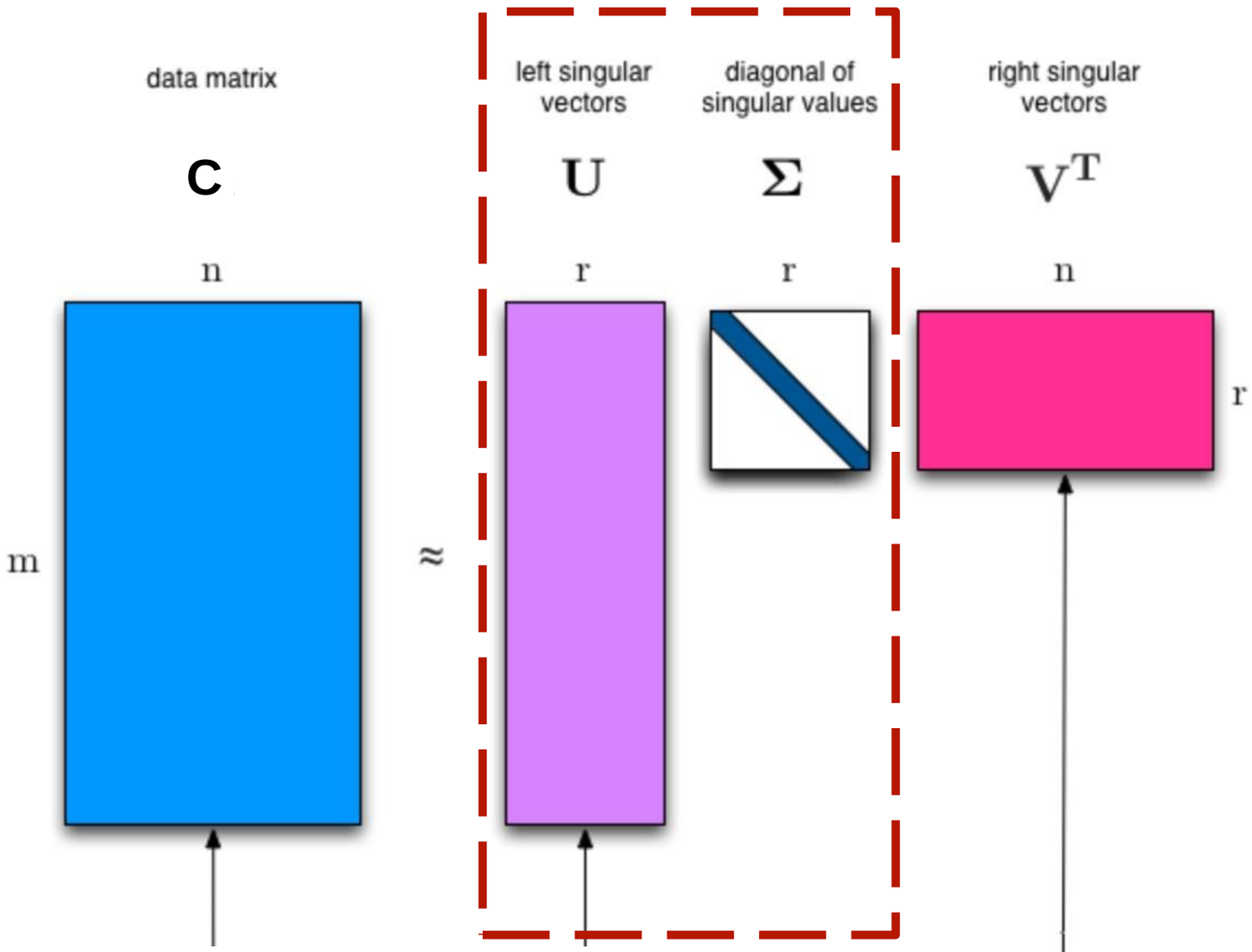


- Interpretation: U correspond à l'ACP de la matrice de similitude des mots:

$[CC^T]_{ij}=<C_{i:},C_{j:}>$ = #document avec le mot i et le mot j (covariance des mots)

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

Décomposition en valeurs singulières



Mila

Université de Montréal

Décomposition SVD

Matrice de plongement de mots

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Mesure de la similarité

Étant donné deux représentations u et v, on mesure leur similarité en calculant leur produit scalaire:

$$\text{dot-product}(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

- Important de s'assurer que les vecteurs u et v sont unitaires.
- Proche 1 quand les représentations sont similaires
- Proche de 0 quand les représentations sont différentes (mais pas opposées)
- Proche de -1 quand les représentations sont opposées

La similitude est préservée

- Étant donné une matrice mot-document C, nous pouvons obtenir une décomposition SVD de C.
- Pour cela on peut garder uniquement les plus grandes valeurs singulières de Σ

$$
\begin{aligned}
CC^T &= (U\Sigma V^T)(U\Sigma V^T)^T \\
&= (U\Sigma V^T)(V\Sigma U^T) \\
&= U\Sigma\Sigma^T U^T \quad (\because V^T V = I) \\
&= U\Sigma(U\Sigma)^T
\end{aligned}
$$

$[CC^T]_{ij} = \langle C_{i:}, C_{j:} \rangle = = \langle (U\Sigma)_{i:}, (U\Sigma)_{j:} \rangle$ = #document avec le mot i et le mot j

Décomposition SVD

Matrice de plongement de mots

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

$\times$

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Plus petite représentation en bleu

Nombre de composantes sélectionnées

Plongement de mots avec la DVS

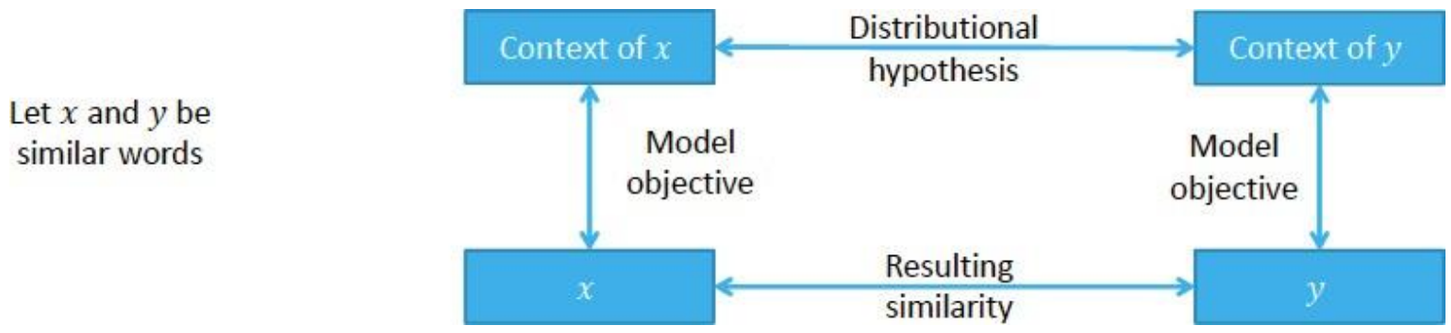
- **Avantages:**
  - Représentation vectorielle dense
  - Préserve la similitude entre les mots
  - Réduction de la dimensionnalité (peut mieux généraliser)
- **Inconvénients:**
  - Déséquilibre dans la fréquence des mots à traiter.
  - Coût de calcul (Nécessite une matrice de la taille du vocabulaire!)

Plongement via SVD

Word2Vec

Word2Vec

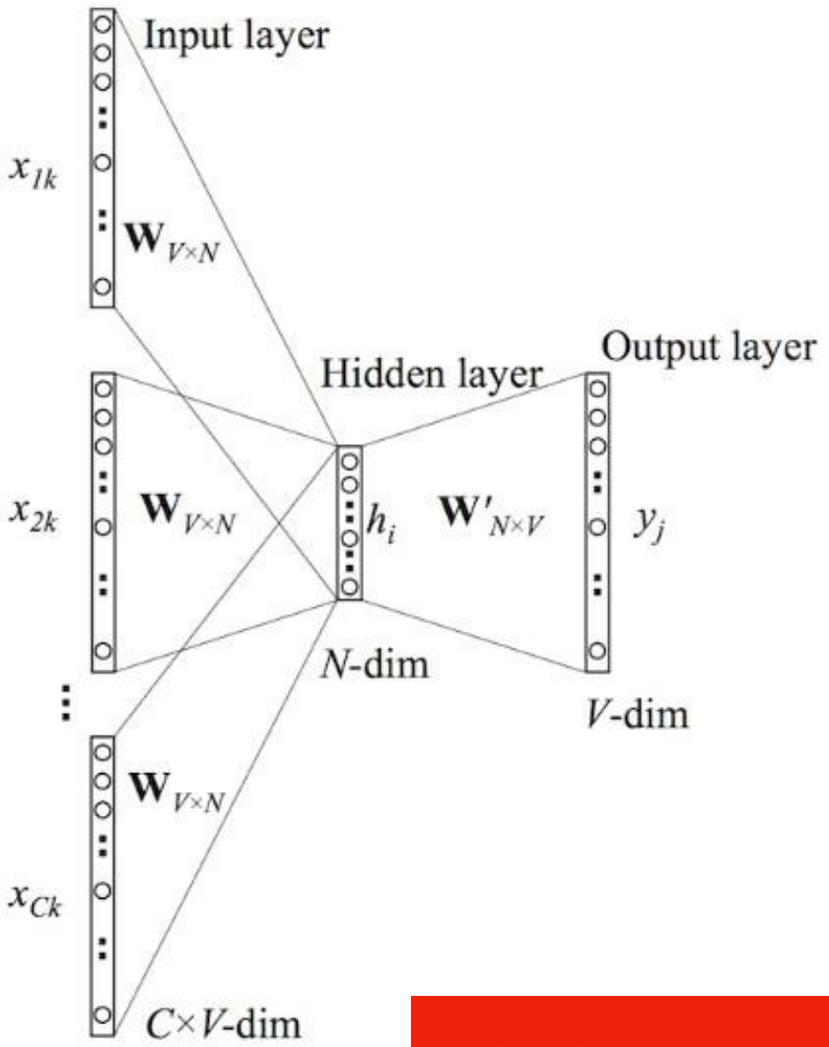
- Modèle pour créer efficacement des plongements de mots
- Rappelez-vous: notre hypothèse est que des **mots similaires apparaissent avec un contexte similaire**
- Intuition : deux mots qui partagent des contextes similaires sont associés à des vecteurs proches l'un de l'autre dans l'espace vectoriel



Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, 2013. Distributed representations of words and phrases and theircompositionality. In Advances in neural information processing systems.

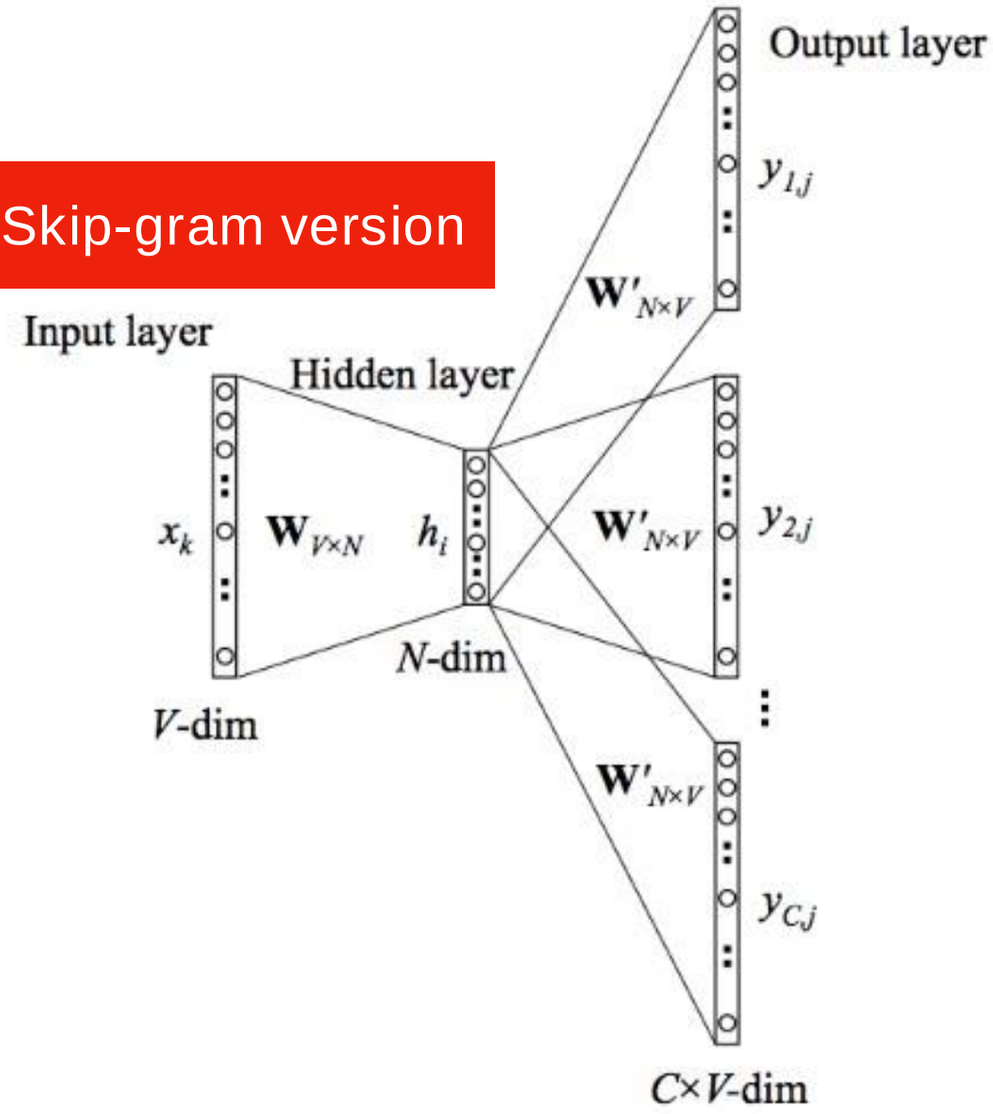
Word2Vec Embedding methods



CBOW version

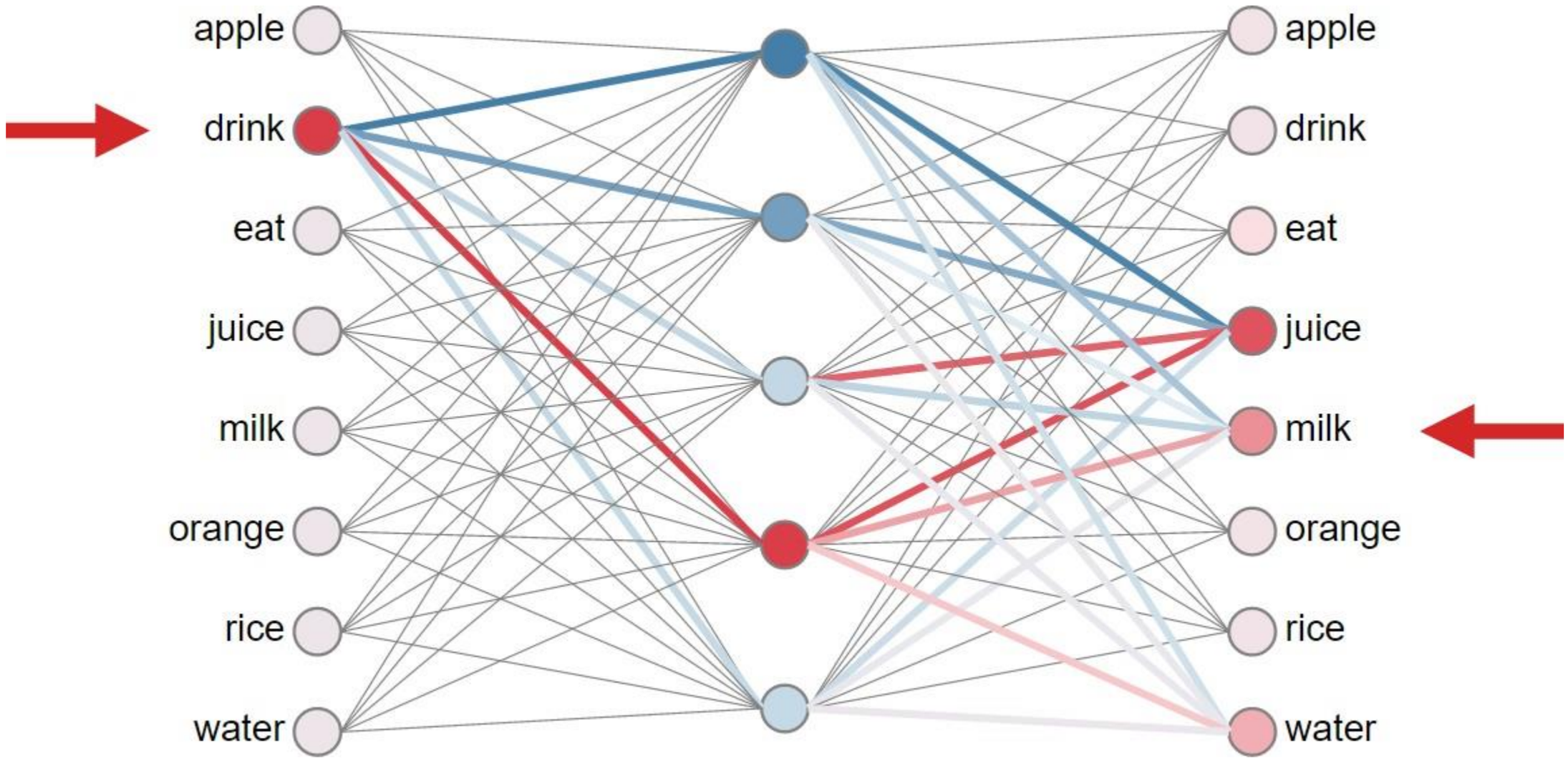
CBoW version: predict center word from context

Skip-gram version



Skip-gram version: predict context from word

Main Goal

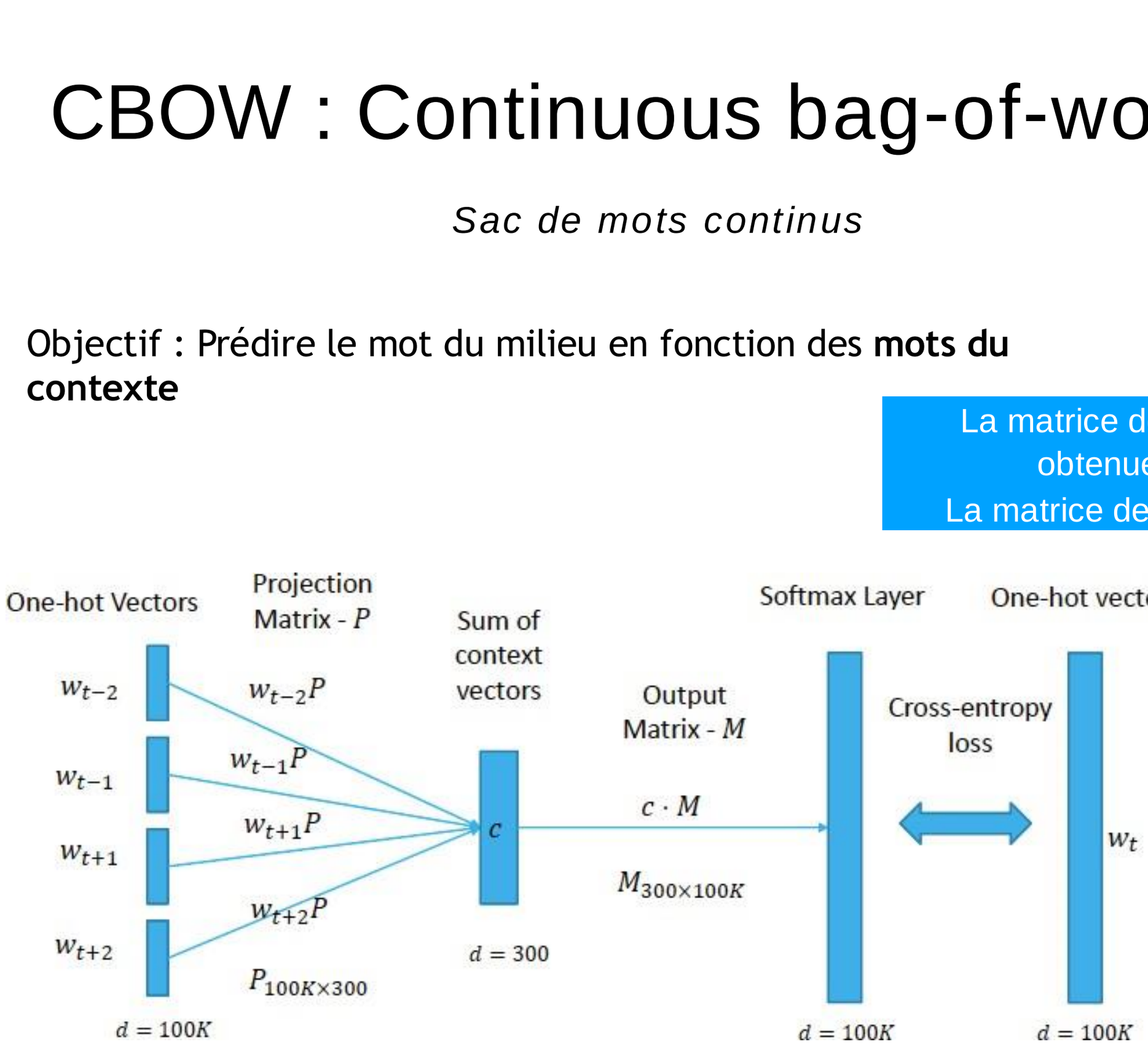

apple
drink
eat
juice
milk
orange
rice
water
apple
drink
eat
juice
milk
orange
rice
water

CBOW : Continuous bag-of-word

Sac de mots continus

- Objectif : Prédire le mot du milieu en fonction des **mots du contexte**

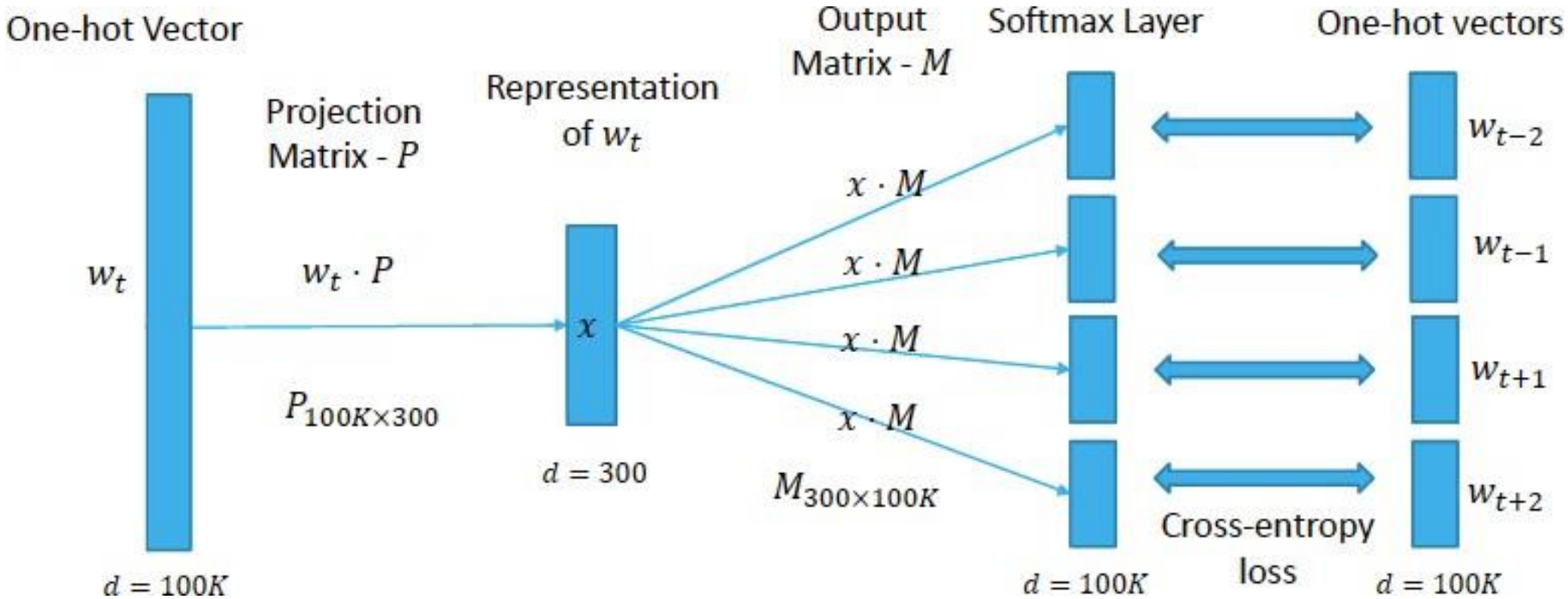
La matrice de projection obtenue P est
La matrice de plongement



Skip-gram (haut niveau)

- Objectif : Prédire les mots du contexte en fonction du mot du milieu

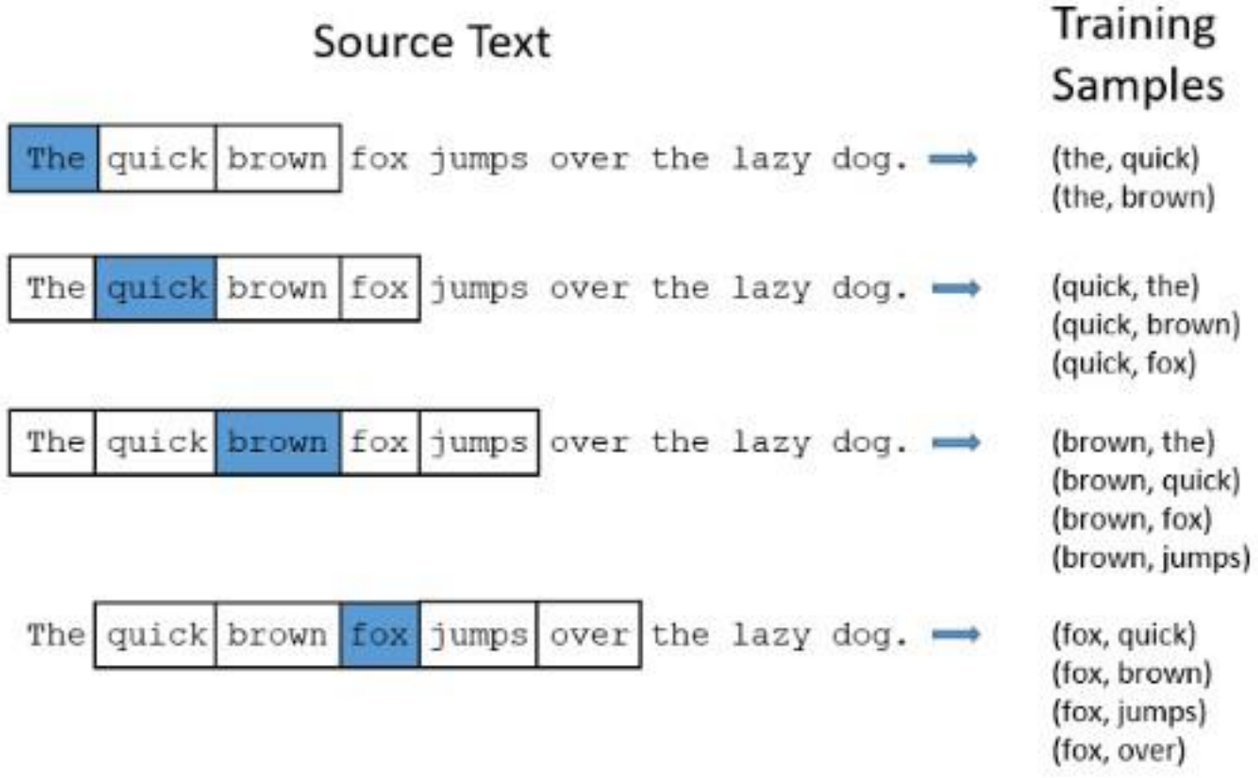
La matrice de projection résultante P est
La matrice de plongement



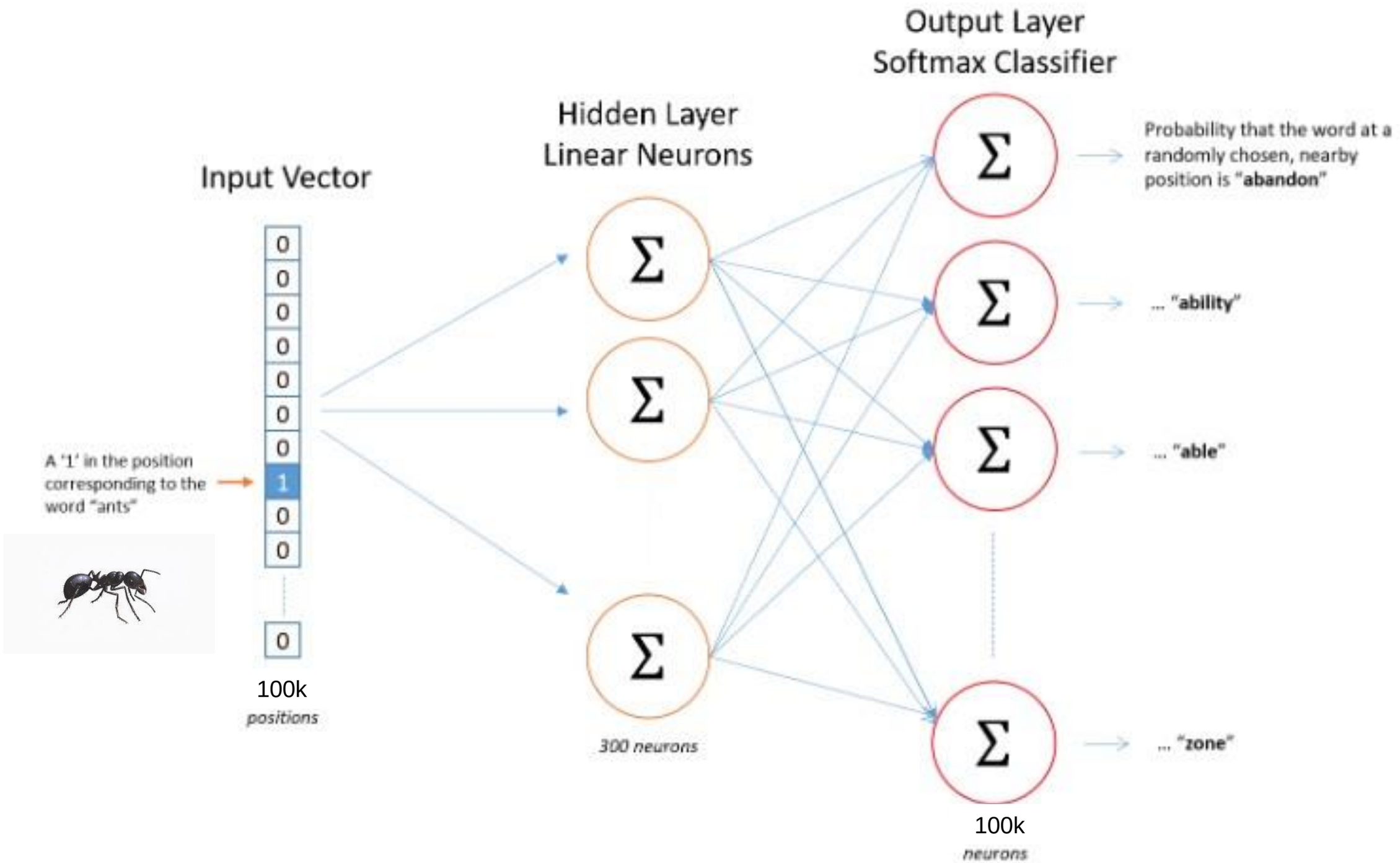
Données d’entraînement

- La création de paires de mots contextuels et cibles dépend de la taille de la fenêtre que vous prenez.
- Pour créer les paires, vous devez regarder à gauche et à droite du mot de contexte pour trouver autant de mots que de la taille d’une fenêtre.

e.g.,
window size = 2

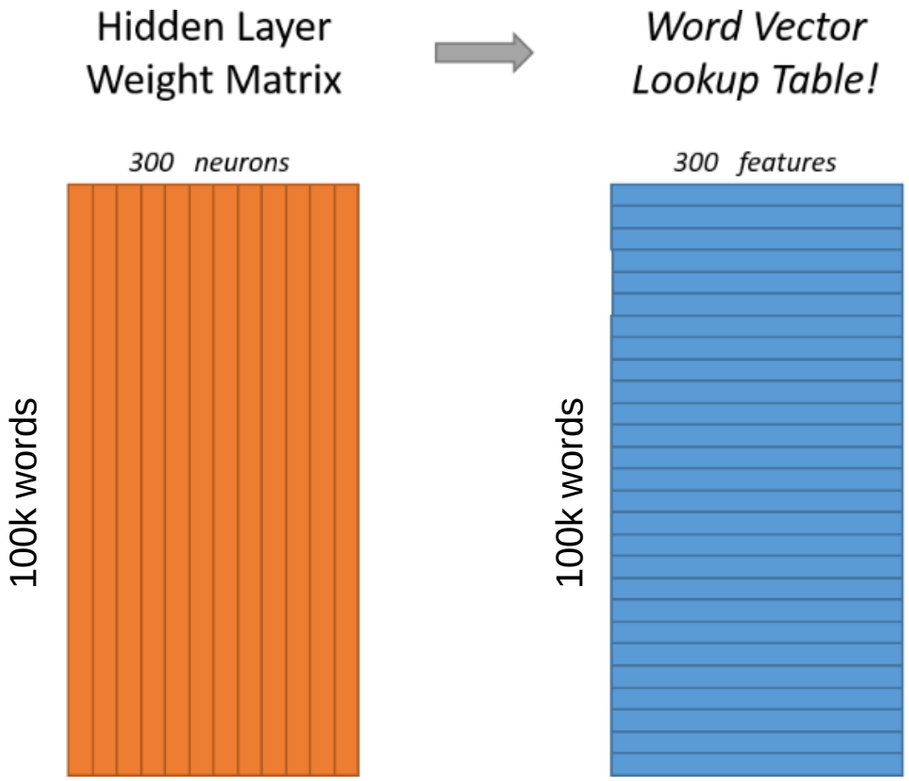


Architecture



Extraire le plongement de mots

Les lignes de la matrice de poids / projection de couche cachée sont en fait les vecteurs de mots (incorporation de mots).



$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

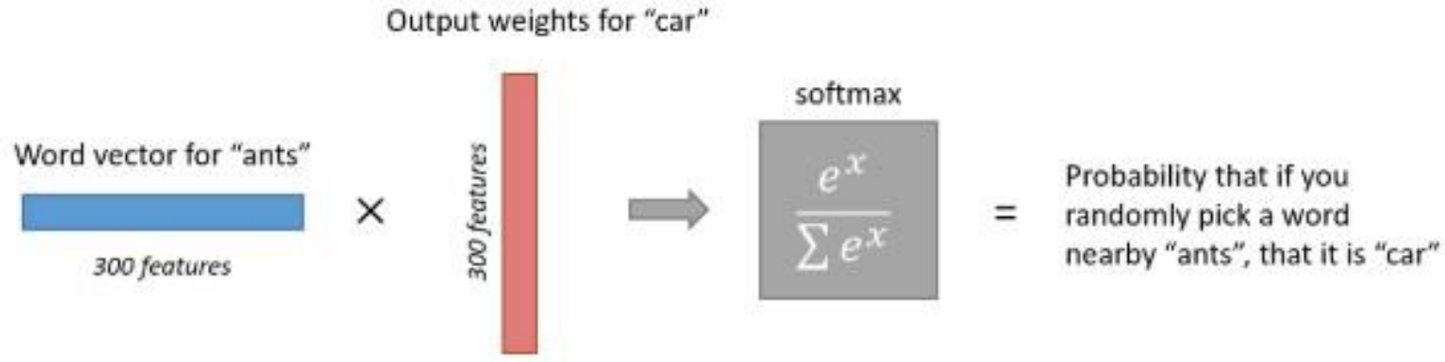
Vecteur unique d'un mot

Couche caché

Plongement

Comment apprendre cela?

- Il s'agit d'un apprentissage semi-supervisé car nous n'avons pas les étiquettes directes associées aux mots, mais nous utilisons les mots voisins (d'un mot de contexte dans une phrase) comme étiquettes. Comment ça marche?



- **Softmax**: est une fonction qui prend comme entrée un vecteur de K nombres réels, et la normalise en une distribution de probabilité constituée de K probabilités proportionnelles aux exponentielles des nombres d'entrée

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

Skip-gram: Formal definition

- Les représentations vectorielles seront utiles pour prédire les mots environnants.
- Formellement: Étant donné une séquence de mot. $w_1, w_2, ...w_T$
  L'objectif du modèle Skip-gram est de maximiser la probabilité logarithmique moyenne:

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

- La formulation de base Skip-gram définit $p(w_{t+j}|w_t)$ avec un softmax

$$p(w_{t+j}|w_t) = \frac{\exp(v'_{w_{t+j}} v_{w_t})}{\sum_{i=1}^{V} \exp(v'_{w_i} v_{w_t})}$$

v - input vector representations
v' - output vector representations

Échantillonnage négatif (détails techniques)

- Rappelons que pour Skip-gram, nous voulons maximiser la probabilité logarithmique moyenne:

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

- Ce qui équivaut à minimiser la **perte d'entropie croisée**

$$L=-\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)=-\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log\frac{\exp(v'_{w_{t+j}}v_{w_t})}{\sum_{i=1}^{V}\exp(v'_{w_i}v_{w_t})}$$

- C'est extrêmement **coûteux en calcul**, car nous devons mettre à jour tous les paramètres du modèle pour chaque exemple d'apprentissage...

Échantillonnage négatif (détails techniques)

- Lorsque l'on examine la perte obtenue à partir d'un seul exemple d'entraînement, nous obtenons :

$$-\log p(w_{t+j}|w_t) = -\log \frac{\exp(v'_{w_{t+j}} v_{w_t})}{\sum_{i=1}^{V} \exp(v'_{w_i} v_{w_t})} = -v'_{w_{t+j}} v_{w_t} + \log \sum_{i=1}^{V} \exp(v'_{w_i} v_{w_t})$$

"positive" pair — "negative" pair

- Lors de l'utilisation de l'échantillonnage négatif, au lieu de passer en revue tous les mots du vocabulaire pour les paires négatives, nous échantillonnons une quantité modeste de k mots (environ 5-20). L'objectif exact utilisé :

$$\log \sigma(v'_{w_{t+j}} v_{w_t}) + \sum_{1=1}^{k} \log \sigma(-v'_{w_i} v_{w_t})$$

Replaces the term: $\log p(w_{t+j}|w_t)$ for each word in the training

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

Sous-échantillonnage des mots fréquents (détails techniques)

- Afin d'éliminer l'effet négatif de mots très fréquents tels que « dans », « le », etc. (qui ne sont généralement pas informatifs), une approche simple de sous-échantillonnage est utilisée.
- Chaque mot w dans l'ensemble d'apprentissage est écarté avec probabilité:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}(f(w_j)^{3/4})}$$

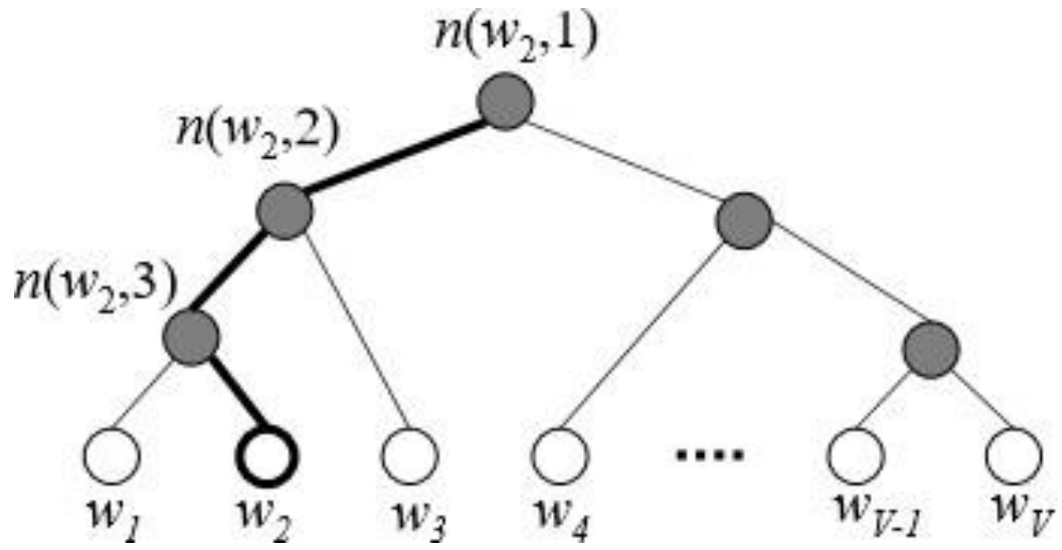
où 3/4 est la valeur obtenue en prenant des expériences; f(w) est la fréquence du mot dans le corpus. De cette façon, les mots fréquents sont jetés plus souvent.

- Cette méthode améliore la vitesse d'entraînement et rend les représentations de mots beaucoup plus précises

Softmax hiérarchique (Détails techniques

- La taille du vocabulaire peut être assez grande –varie entre 30K, 82K et 1M. s'il est implémenté de manière naïve, fait de la couche de sortie un goulot d'étranglement.

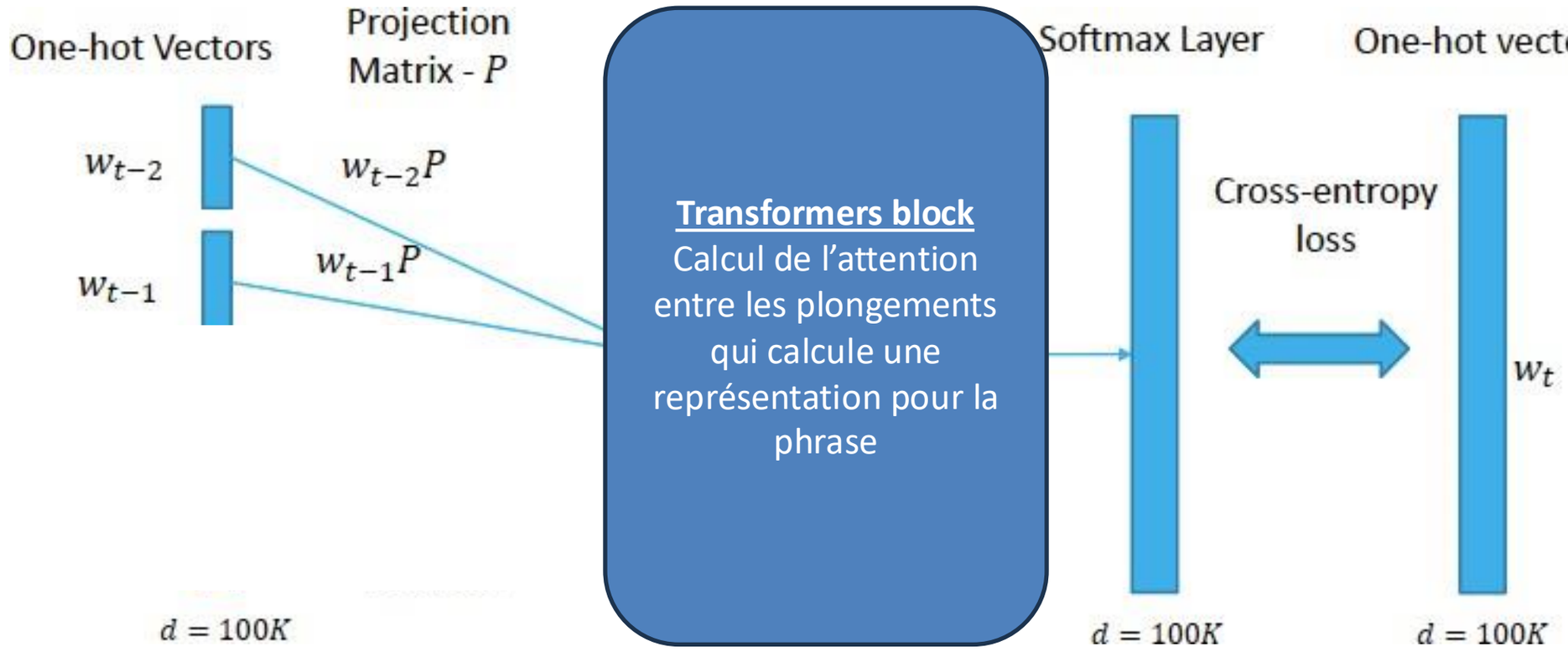
- Une option consiste donc à utiliser le softmax hiérarchique, représentant le vocabulaire sous la forme d'un arbre binaire de Huffman (les mots les plus courants étant plus proches de la racine), ce qui réduit la complexité à O(log(V)).



- Il existe un chemin unique du nœud racine vers un nœud feuille.
- En effet, la probabilité de prédire un mot est déduite du chemin unique correct du nœud racine à ce mot.

CBOW, Skip-Gram, LLMs

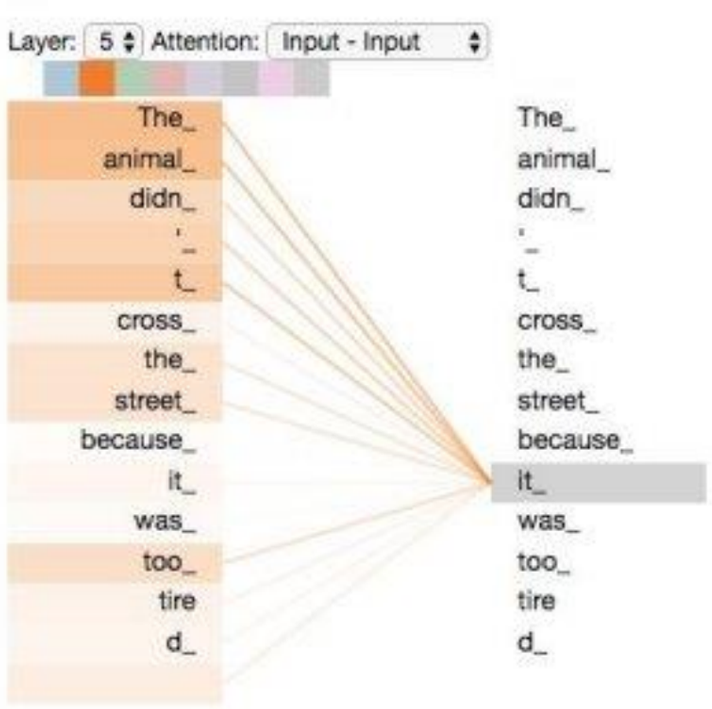
Les gros modèles de languages ont un fonctionnement très proche de celui de CBOW (juste plus profond et avec de l'attention)

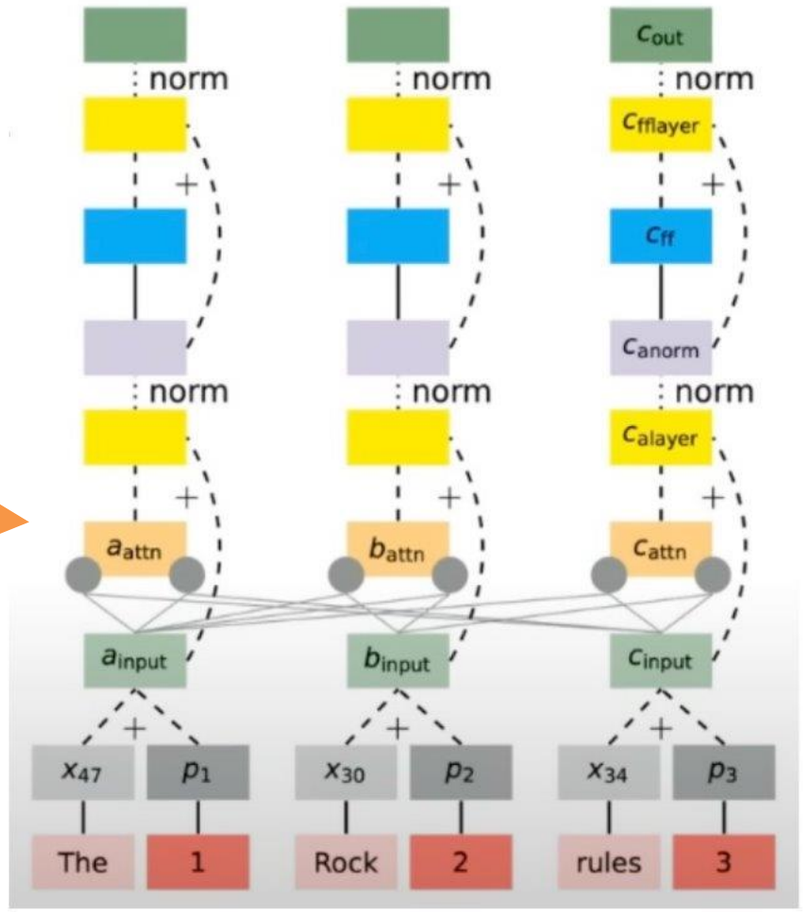


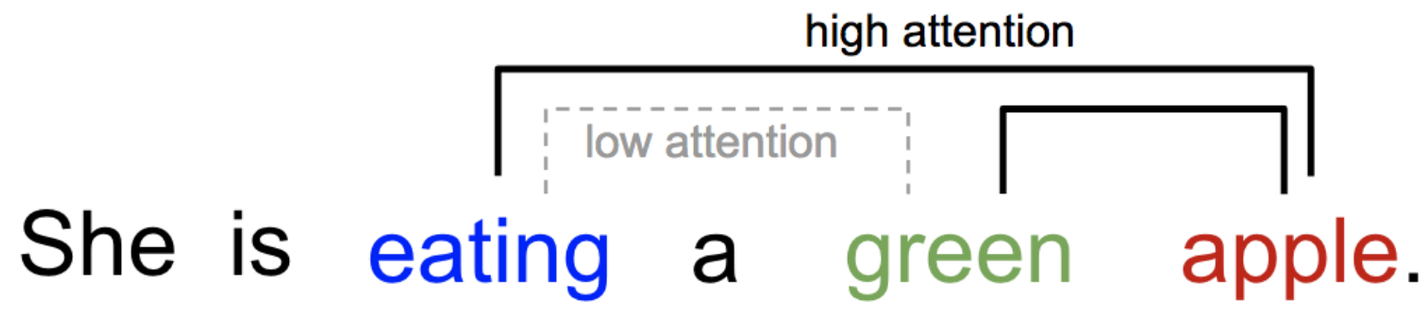
Mila

Université de Montréal

Transformers







Mila

Université de Montréal

Attention lorsque l'on génère

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

Word2Vec

- **Avantages:**
  - Fournir un vecteur dense
  - Plus rapide à entraîner que GolVe
  - Extraire la similitude sémantique entre les mots
  - Des modèles pré-entraînés sont disponibles en ligne.
- **Inconvénients:**
  - Impossible de capturer le contexte global
  - Il n’est pas clair comment représenter une phrase / un document avec word2vec (contrairement au sac de mots)

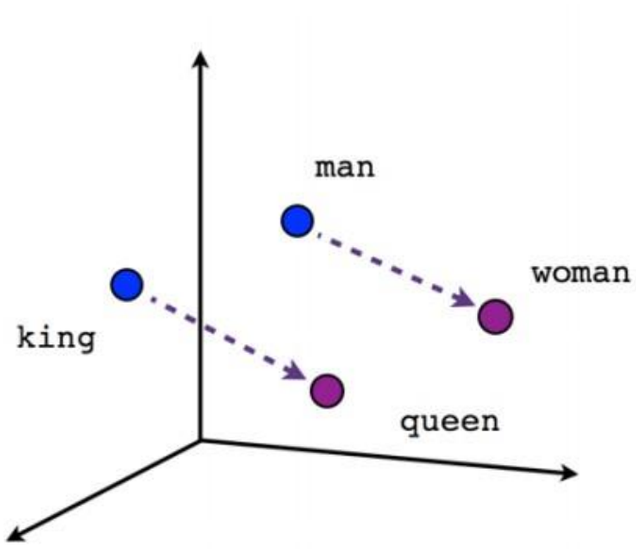
Méthodes de plongements de mots

- **Avantages:**
    - Apprend les caractéristiques de chaque mot à partir d'un corpus de texte.
    - Aucun prétraitement lourd n'est nécessaire, juste un corpus.
    - Les vecteurs de mots peuvent être utilisés comme caractéristiques pour de nombreuses tâches supervisés
    - Capture les similitudes et les relations linéaires entre les vecteurs de mots.

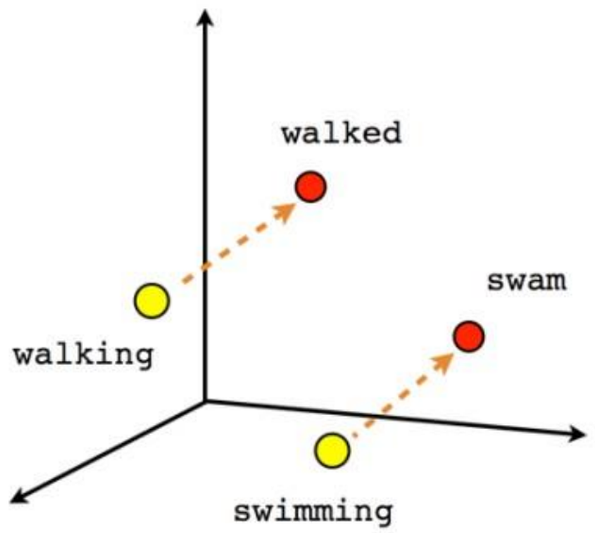
D’autres approches de plongement de mots?

- **Static word embeddings**
  - Word2Vec (Google): https://code.google.com/archive/p/word2vec/
  - GloVe (Stanford): https://nlp.stanford.edu/projects/glove/
  - FastText: https://fasttext.cc/
- **Dynamic word embeddings**
  - ELMO: https://allennlp.org/elmo
  - FlairEmbeddings: https://github.com/zalandoresearch/flair
  - BERT: https://pypi.org/project/bert-embedding/

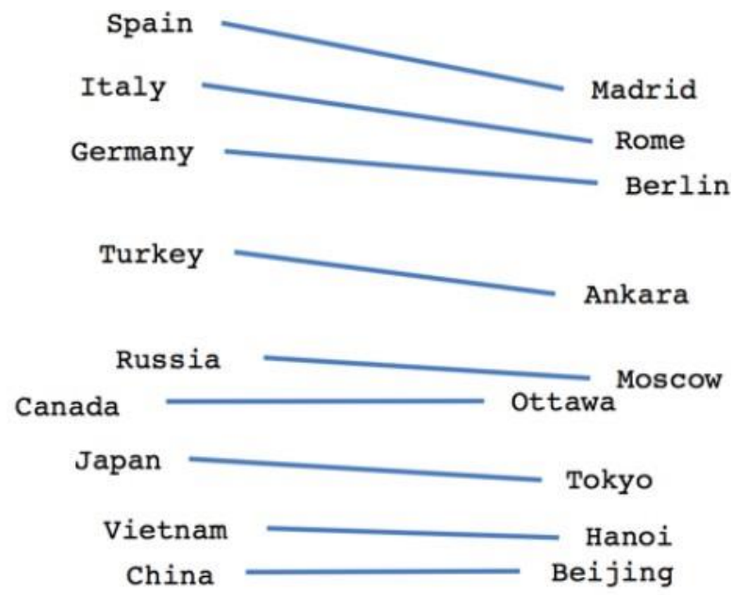
Régularités



Male-Female



Verb tense



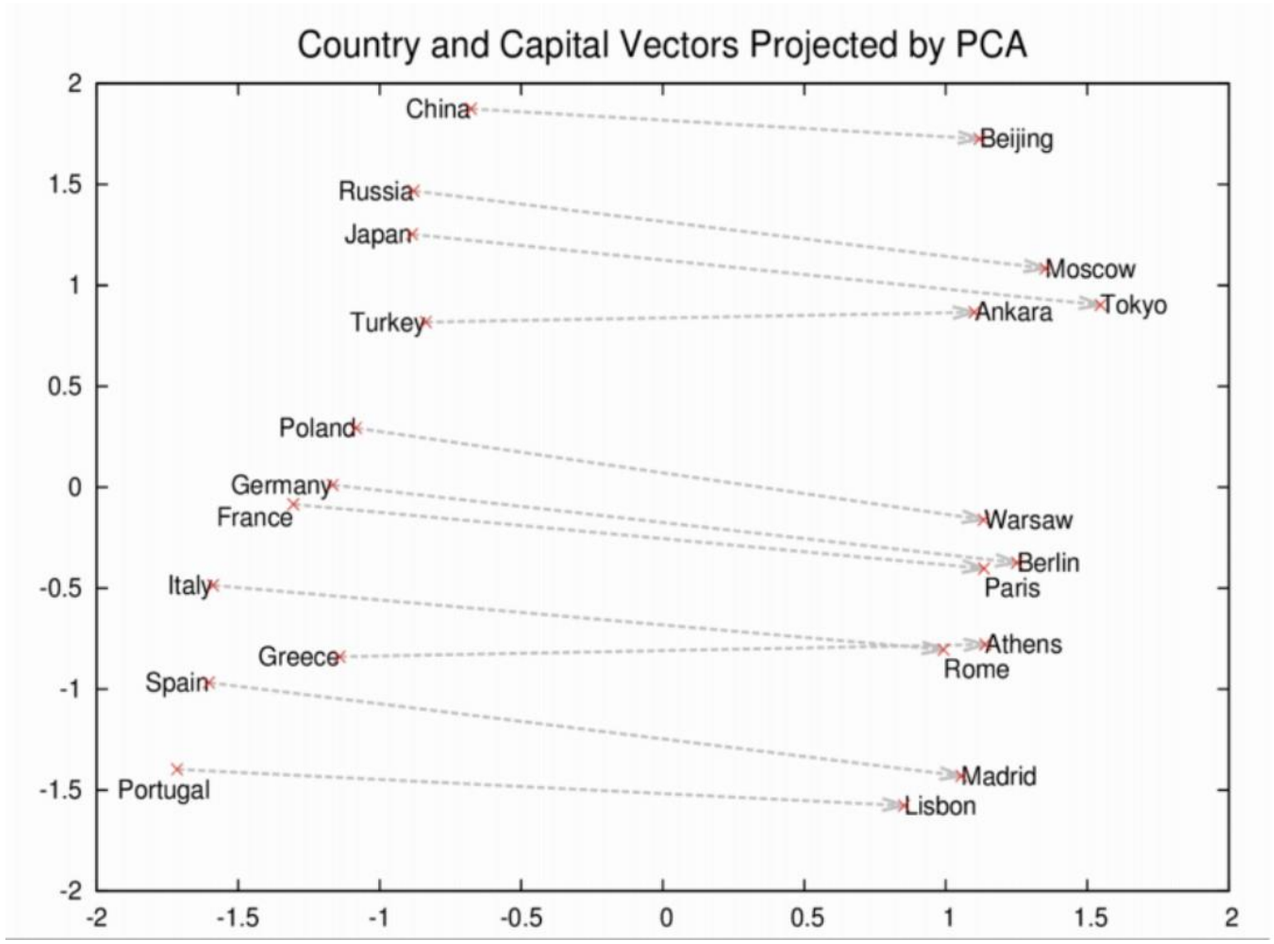
Country-Capital

vector[Queen] = vector[King] - vector[Man] + vector[Woman]

Régularités

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Visualisation dans l'espace des mots



Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Biais dans les plongements

Les professions ayant obtenu les scores les plus élevés à l'égard des femmes (à gauche) et les scores les plus élevés à l'égard des hommes (à droite) :

Highest female bias

occupation	bias	occupation	bias
maid	59.2	librarian	20.1
waitress	52.5	obstetrician	16.9
midwife	50.9	secretary	13.7
receptionist	50.2	socialite	12.1
nanny	47.7	therapist	10.2
nurse	45.4	manicurist	10.1
midwives	43.8	hairdresser	9.7
housekeeper	36.6	stylist	8.6
hostess	32	homemaker	6.9
gynecologist	31.6	planner	5.8

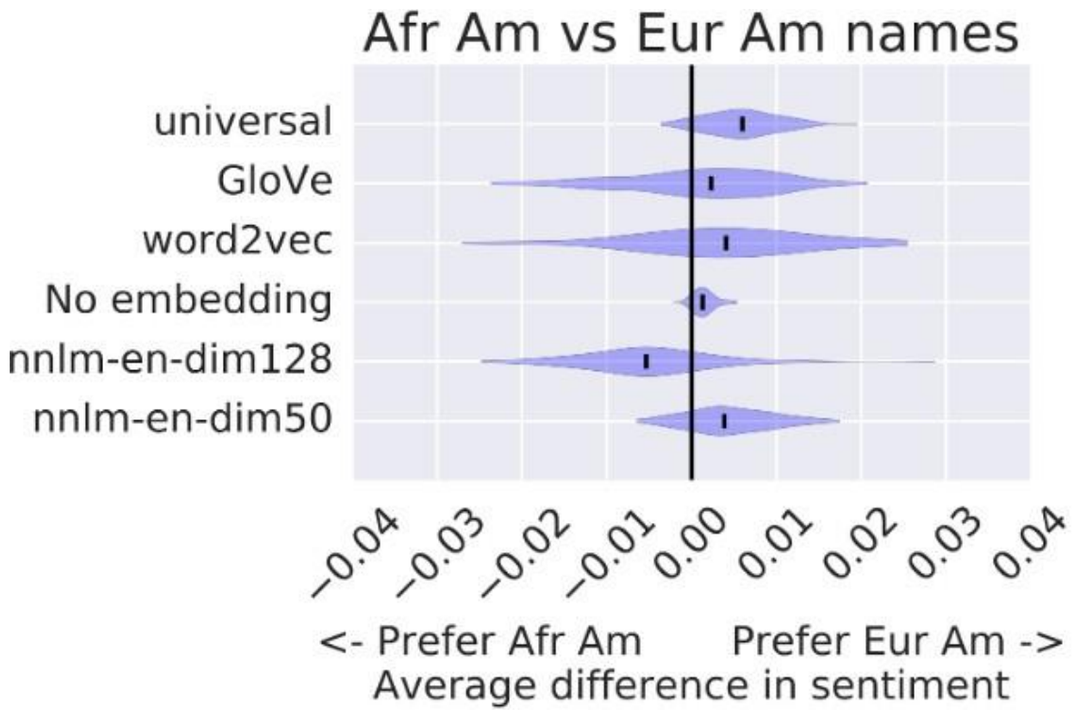
Highest male bias

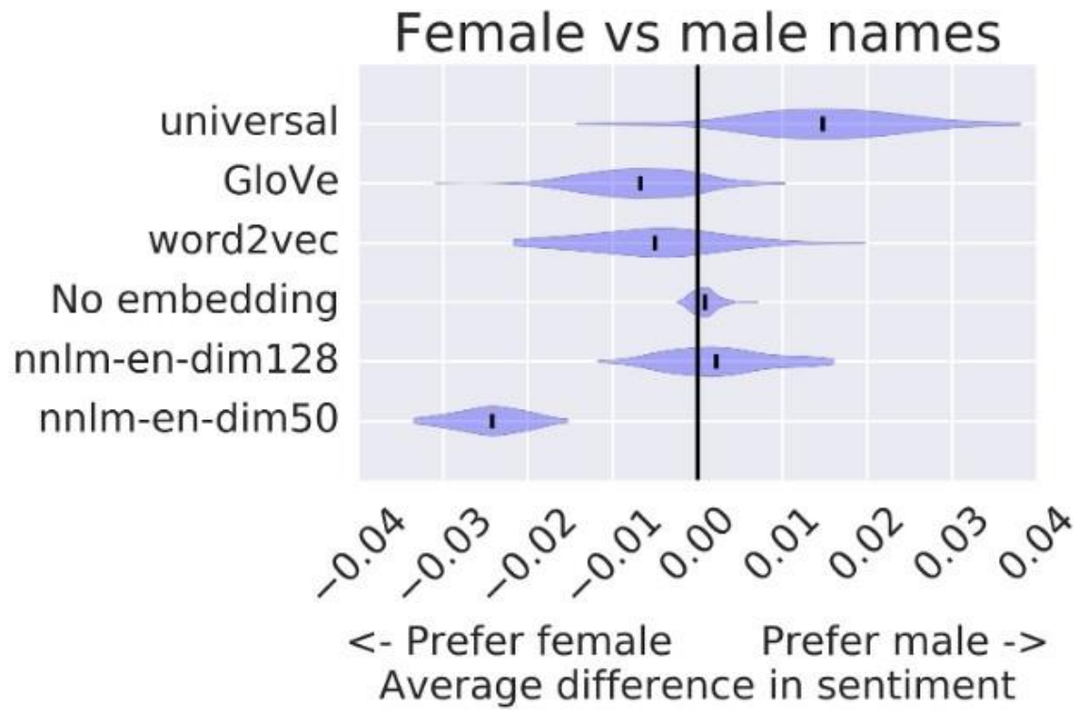
occupation	bias	occupation	bias
undertaker	-73.4	captain	-53.4
janitor	-62.3	announcer	-51.1
referee	-60.7	architect	-50.7
plumber	-58	maestro	-50.6
actor	-56.9	drafter	-46.7
philosopher	-56.2	usher	-46.6
barber	-55.4	farmer	-45.4
umpire	-54.3	broadcaster	-45.2
president	-54	engineer	-45.1
coach	-53.8	magician	-44.8

https://developers.googleblog.com/2018/04/text-embedding-models-contain-bias.html

Biais dans les plongements

Différence dans les scores de sentiment moyens :

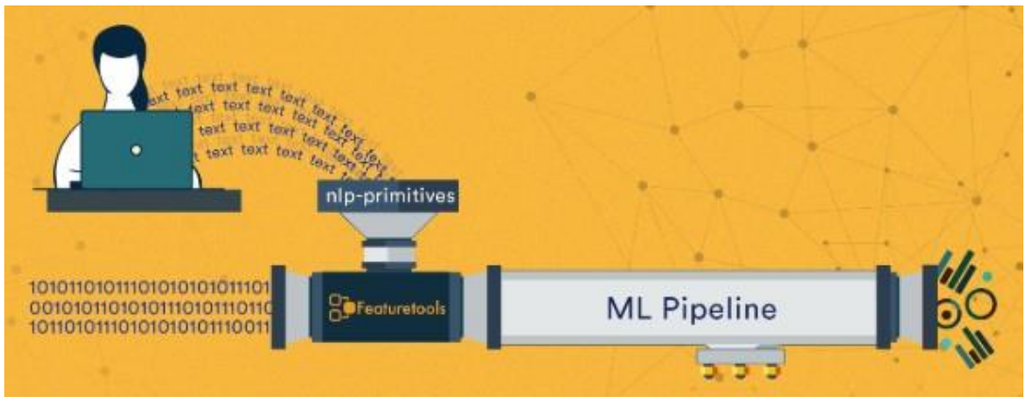




https://developers.googleblog.com/2018/04/text-embedding-models-contain-bias.html

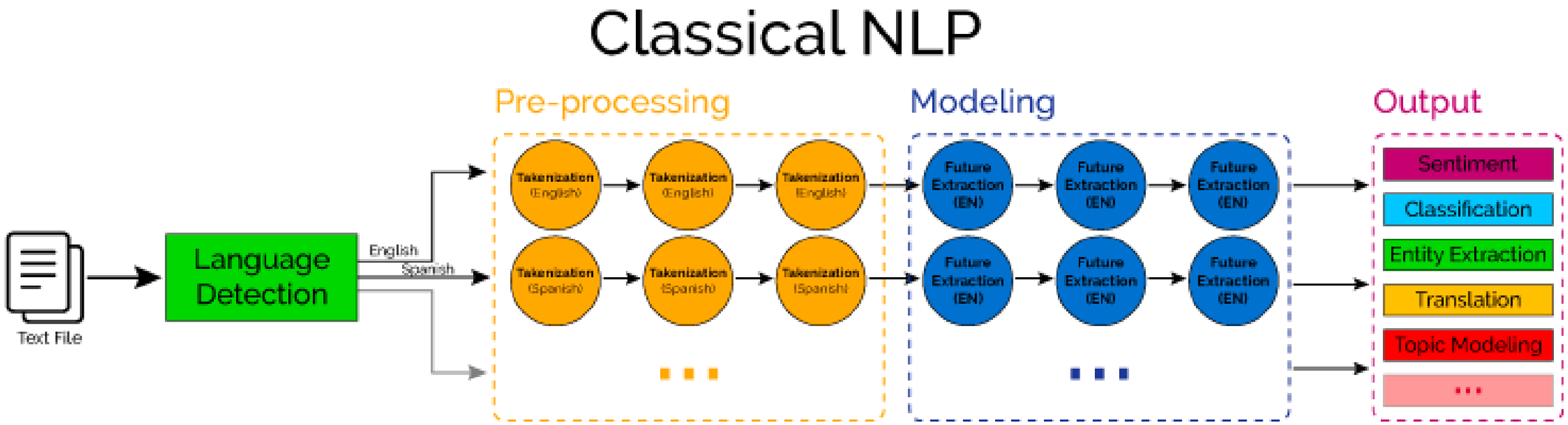
Apprentissage des représentations

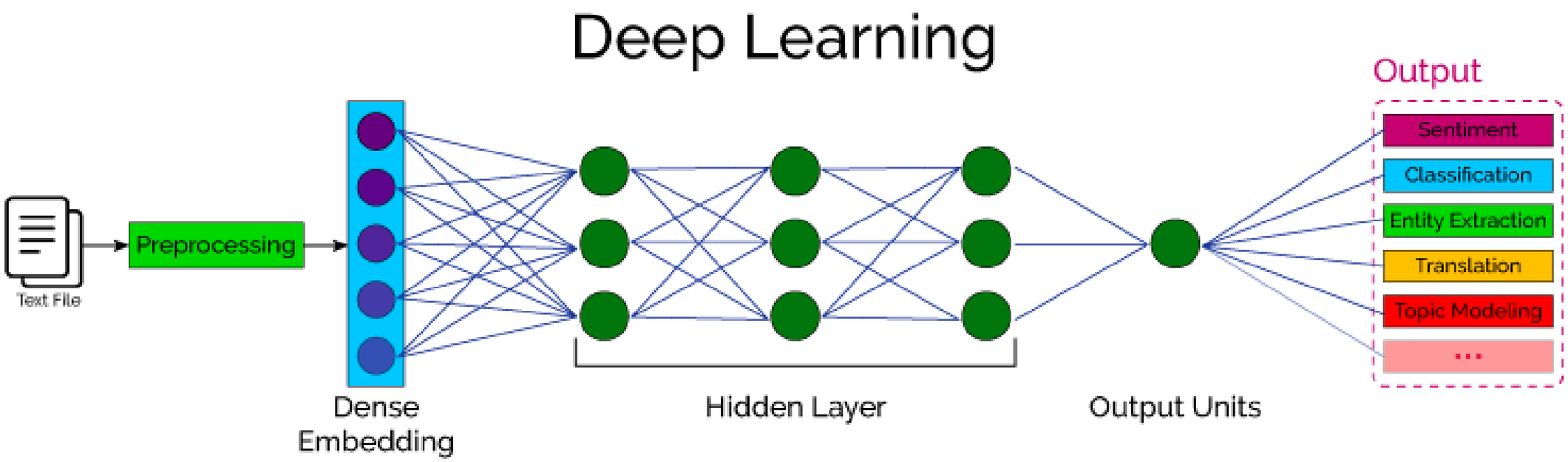
- L’apprentissage de la représentation est un ensemble de techniques qui apprennent une fonctionnalité : une transformation de l’entrée de données brutes en une représentation qui peut être exploitée efficacement dans les tâches d’apprentissage automatique.



- Fait partie de l’ingénierie / apprentissage des fonctionnalités.
- Débarrassez-vous des caractéristiques et de la représentation « conçues à la main »
- Apprentissage non supervisé des fonctionnalités - évite l’ingénierie manuelle des fonctionnalités

Différence entre le NLP classiques et le NLP d'apprentissage profond





Ressources

- Live demo: https://ronxin.github.io/wevi/
- Language modeling: https://web.stanford.edu/class/cs124/lec/languagemodeling.pdf
- C. Manning- Human Language & vector words: http://videolectures.net/deeplearning2015_manning_language_vectors/
- K. Cho - Deep Natural Language Understanding: http://videolectures.net/deeplearning2016_cho_language_understanding/
- John Arevalo, Language modeling and word embeddings

Conferences focusing on NLP

- Natural Language Processing
  ACL, NAACL, EACL, EMNLP, CoNLL, Coling, **TACL**

- Machine learning
  ICML, NIPS, ECML, AISTATS, ICLR, **JMLR**, **MLJ**

- Artificial Intelligence
  AAAI, IJCAI, UAI, **JAIR**