# *Cours de Traitement Automatique du Langage*

*Chapitre* 03 : Modèles de Langues

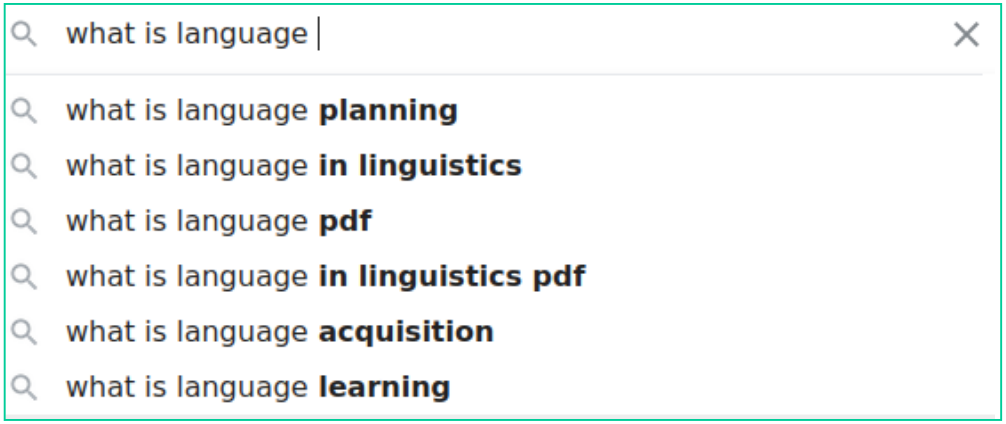
***DI5 – 2021-22***

Polytech ’Tours J.Y. RAMEL

# Introduction

## Modèle de langages

- La tâche centrale du TAL
- Objectif
  - Estimer les probabilités de séquences de mots




- Probabilité d'une phrase: P(S) = P(w1,w2,...,wn)
  - Auto-completion : P(traitement automatique de l'information)>P(traitement automatique de l'eau)
  - Traduction automatique : My tall brother → P(Mon grand frere) > P(Mon haut frere)
  - Correction des fautes grammaticales : P(Un objet qu'on puisse emporter) > P(Un objet qu'ont puisse emporter)
  - Reconnaissance de paroles : P(Jeudi matin) > P(Je dis matin)
  - …

# Introduction

## Approche probabiliste

- Probabilité d'occurrence d'un mot dans une phrase : $P(w_n|w_1,...,w_{n-1})$
- Formule des probabilités composées (décomposition)
  - $P(w_1...w_n) = P(w_1).P(w_2|w_1).P(w_3|w_1,w_2)...P(w_n|w_1,...,w_{n-1})$
  - $P(w_1, ..., w_n) = \prod_{i=1}^{n} P(w_i|w_{<i})$
- Les probabilités sont estimées en comptant les mots dans des corpus (apprentissage)

**Exemple de probabilité d'une phrase**

$P(je\ travaille\ à\ l'ESI) =$
$P(je)P(travaille|je)P(à|je\ travaille)\ldots P(ESI|je\ travaille\ à\ l')$
$P(ESI|je\ travaille\ à\ l') = \frac{C(je\ travaille\ à\ l'ESI)}{C(je\ travaille\ à\ l')}$
Où
$C$ est la fonction qui compte le nombre d'occurrences d'une expression dans un corpus d'entraînement

- Il faut un grand corpus pour estimer cette probabilité

# Introduction

## Approche probabiliste

- Il faut de grands corpus pour estimer ces probabilités !
- Mais :
  - Il y a trop de phrases possibles et pas assez de données !
  - Impossible d'avoir un corpus de texte contenant toutes les phrases.
  - Parmi les phrases qui n'apparaissent pas, certaines sont probables, d'autres non,
  - Cela s'aggrave lorsque les séquences deviennent plus longues.
  - Solution → Limiter leur longueur !

# Modèle n-gram

## Hypothèse de Markov (Markov assumption)

- Un état futur ne dépend que de l état présent
  - $P(x_n|x_1,...,x_{n-1}) \approx P(x_n|x_{n-1})$
- Cas général avec k états passés
  - $P(x_n|x_1,...,x_{n-1}) \approx P(x_n|x_{n-k},...,x_{n-1})$
- Estimation de probabilité en utilisant les N-gram (➔ k = n - 1)
  - $P(x_1,...,x_n) \approx \prod P(x_i|x_{i-k+1},...,x_{i-1})$

- Modèle uni-gram : $P(w_n|w_1,...,w_{n-1}) \approx P(w_n)$ → proba des mots dans un/des corpus
- Modèle bi-gram : $P(w_n|w_1,...,w_{n-1}) \approx P(w_n|w_{n-1})$ → proba du mot suivant sachant un mot
- Modèle tri-gram : $P(w_n|w_1,...,w_{n-1}) \approx P(w_n|w_{n-2},w_{n-1})$ →…

- Difficile d'aller bcp plus loin que le modèle Trigram…. Pourquoi ?
- Exemple Tri-grams

$$P(\text{What is language modeling}) \approx P(\text{What}) \times P(\text{is} \mid \text{What})$$
$$\times P(\text{language} \mid \text{What, is}) \times P(\text{modeling} \mid \text{is, language})$$

# Modèle n-gram

## Estimateur de maximum de vraisemblance sur bi-grammes

$C$ est le nombre d'occurrences des N-grammes dans le corpus

$$\text{Bi-grammes} : P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

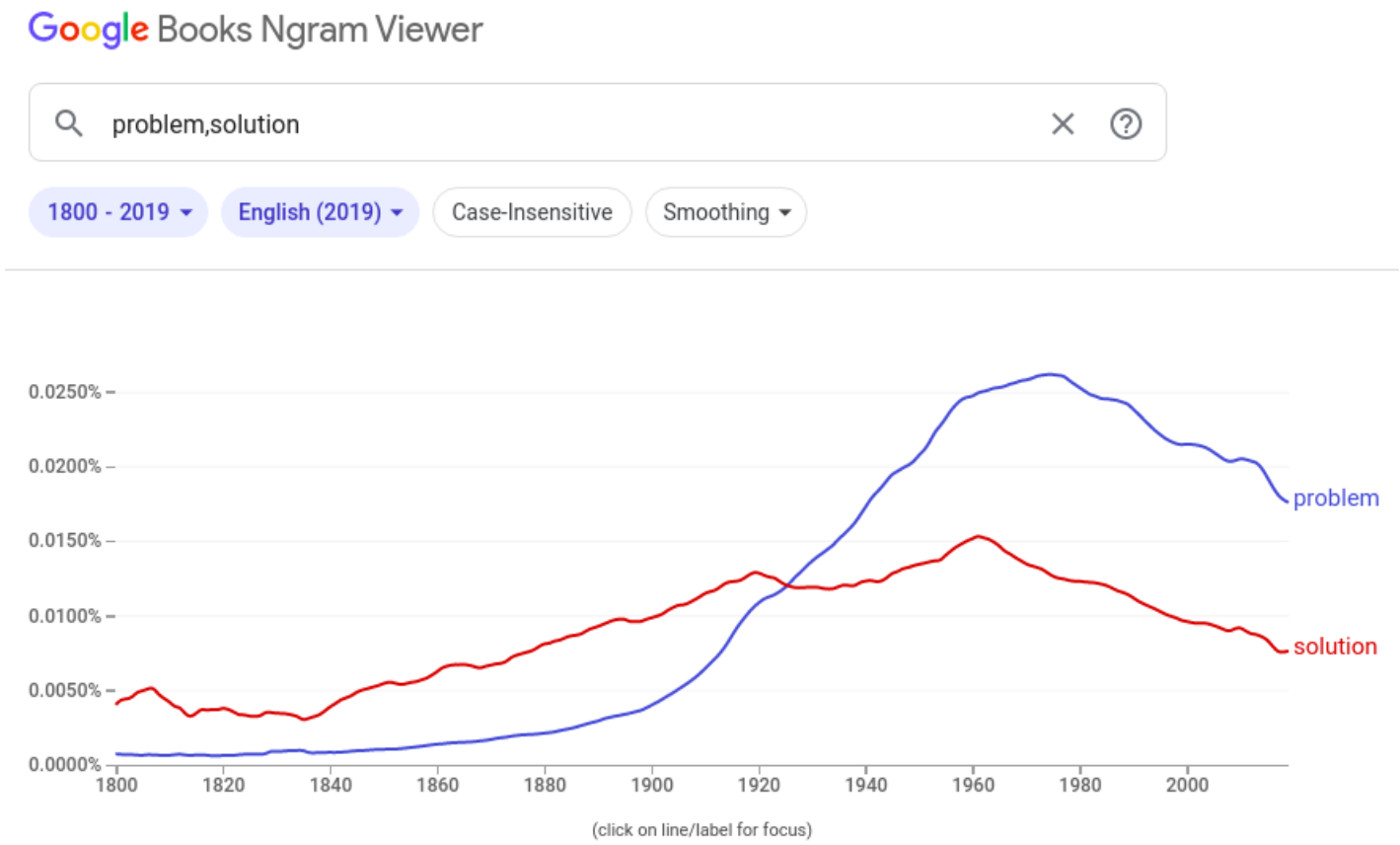
**Exemple d'un corpus d'entraînement**

- <s>un ordianteur peut vous aider </s>
- <s>il veut vous aider </s>
- <s>il veut un ordinateur </s>
- <s>il peut nager </s>

- $P(peut|il) =$ 
- $P(\text{<s>}il\ peut\ vous\ aider\ \text{</s>}) =$

# Modèle n-gram

- Google Books N-gram Viewer
  - https://books.google.com/ngrams
- Modèles prétraités a partir des livres :
  - https://storage.googleapis.com/books/ngrams/books/datasetsv3.html



# Modèle n-gram

Prenons un exemple : Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- ….
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Au total 9222 phrases dans ce corpus

# Modèle n-gram

Enumération des bigrams

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

# Modèle n-gram

Normalisation selon les unigrams:

- Result:

| i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| want | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| to | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| eat | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| chinese | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| food | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| lunch | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| spend | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

# Modèle n-gram

Estimation de la probabilité d'une phrase ?

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

# Modèle n-gram

## Quels interêts ?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

## Remarque pratique au passage

- On fait tous les calculs en Log() pour éviter le dépassement de capacité de précision et addition plus rapide que multiplication

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

# Modèle n-gram - Lissage (smoothing)

- **Problème des N-gram absents dans le corpus d'entraınement**
  - P(<s>il veut nager</s>) =P(il|<s>)P(veut|il)P(nager|veut)P(</s>|nager) =3/4.2/3.0/1.1/1=0
- Plus N est grand plus le risque augmente, MAIS
- Si on utilise des petits N-gram⇒ Perte de l'information
  - Les langues permettent des dépendances long terme
  - L'ordinateur que j'ai utilisé hier a Tours pendant la séance du cours a planté
- Si on utilise des grands N-gram ⇒ Complexité élevée du modèle
  - Il faut un corpus plus grand
  - Représentation des N-gram : $V^N$ ou V est la taille du vocabulaire et N est le nombre de grams
- **Techniques de lissage, 2 intuitions :**
  - emprunter une petite portion des probabilités des N-grams existants pour former une probabilité aux N-gram absents (Laplace)
  - Réduire la taille des N-grams pour estimer les proba des N-grams qui sont absents (back-off)

# Modèle n-gram

## Exemple de techniques de lissage

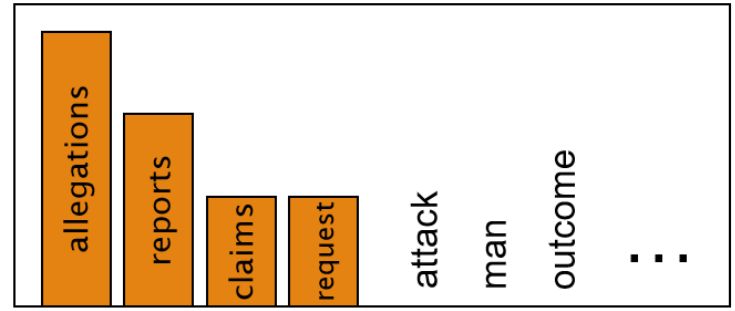
### Lissage de Lidstone (Bi-grammes comme exemple)

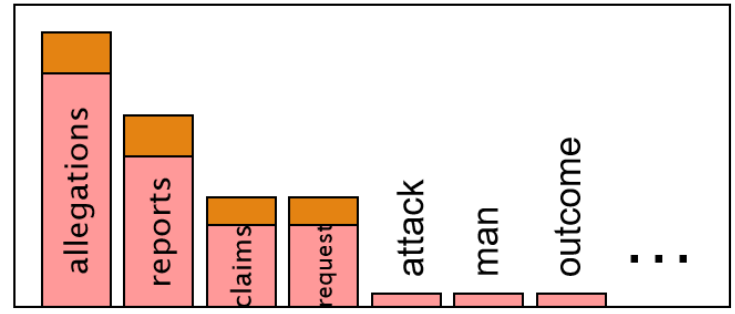
$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + \alpha}{C(w_{n-1}) + \alpha V}$$

Où $V$ est la taille du vocabulaire du modèle

$\alpha = 1$ : ***Lissage de Laplace***

$\alpha = 0.5$ : ***Loi de Jeffreys-Perks***




### Exemple : lissage de Laplace

- Le corpus contient 8 mots différents
- Il y a $8^2 = 64$ bi-grammes possibles → Prenons V ≈ 64
- $P(\text{<s>il veut nager </s>}) =$
  $P(il|\text{<s>})P(veut|il)P(nager|veut)P(\text{</s>}|nager) = \frac{3+1}{4+64}\frac{2+1}{3+64}\frac{0+1}{1+64}\frac{1+1}{1+64}$

# Modèle n-gram

## Exemple de techniques de lissage

$$P(\text{is language}) = 0 \;\Rightarrow\; P(\text{is language}) \approx P(\text{language})$$

### Interpolation (Tri-grammes comme exemple)

$$P_I(w_n|w_{n-2}w_{n-1}) = \lambda_3 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_1 P(w_n)$$

Où $\sum_i \lambda_i = 1$
$\lambda_3$, $\lambda_2$ et $\lambda_1$ sont estimés en utilisant un autre corpus de réglage

### Back-off de Katz (Tri-grammes comme exemple)

$$P_{BO}(w_n|w_{n-2}w_{n-1}) = \begin{cases} P^*(w_n|w_{n-2}w_{n-1}) & \text{si } C(w_{n-2}w_{n-1}w_n) > 0 \\ \alpha(w_{n-2}w_{n-1}) P_{BO}(w_n|w_{n-1}) & \text{sinon} \end{cases}$$

Où :
$P^*$ est la probabilité réduite (la réduction sera distribuée sur les probabilités des N-grammes de l'ordre inférieur)
$\alpha$ est une fonction qui distribue la réduction selon le contexte

# Modèle n-gram

## Reprenons notre exemple

- Lissage de Laplace

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

*Avec V = 1467*

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 6 | 828 | 1 | 10 | 1 | 1 | 1 | 3 |
| want | 3 | 1 | 609 | 2 | 7 | 7 | 6 | 2 |
| to | 3 | 1 | 5 | 687 | 3 | 1 | 7 | 212 |
| eat | 1 | 1 | 3 | 1 | 17 | 3 | 43 | 1 |
| chinese | 2 | 1 | 1 | 1 | 1 | 83 | 2 | 1 |
| food | 16 | 1 | 16 | 1 | 2 | 5 | 1 | 1 |
| lunch | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| spend | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.0015 | 0.21 | 0.00025 | 0.0025 | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want | 0.0013 | 0.00042 | 0.26 | 0.00084 | 0.0029 | 0.0029 | 0.0025 | 0.00084 |
| to | 0.00078 | 0.00026 | 0.0013 | 0.18 | 0.00078 | 0.00026 | 0.0018 | 0.055 |
| eat | 0.00046 | 0.00046 | 0.0014 | 0.00046 | 0.0078 | 0.0014 | 0.02 | 0.00046 |
| chinese | 0.0012 | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052 | 0.0012 | 0.00062 |
| food | 0.0063 | 0.00039 | 0.0063 | 0.00039 | 0.00079 | 0.002 | 0.00039 | 0.00039 |
| lunch | 0.0017 | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011 | 0.00056 | 0.00056 |
| spend | 0.0012 | 0.00058 | 0.0012 | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

# Modèle n-gram

## Reprenons notre exemple

- Reconstruction du tableau à partir des nouvelles proba

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

- Comparaison avec les données initiales

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

# Modèles neuronaux

## Idées principales → CF Word Embeddings

- Vecteurs de caractéristiques continus : Chaque mot est représenté par un vecteur caractéristique de dimension $d \ll |V|$
- Fonction de probabilité : La probabilité du mot suivant est exprimée en tant que fonction continue des caractéristiques du mot dans le contexte actuel
- Apprentissage conjoint : Les paramètres de la représentation du mot et de la fonction de probabilité sont appris conjointement.

## Continuité + hypothèse distributionnelle implique :

- qu'un petit changement dans le vecteur de caractéristiques induira un petit changement dans la probabilité.
- Par conséquent, les mises à jour causées par la présence de la phrase suivante dans les données d'apprentissage augmenteront la probabilité de toutes les phrases 'voisines'.

*A dog was running in a room*

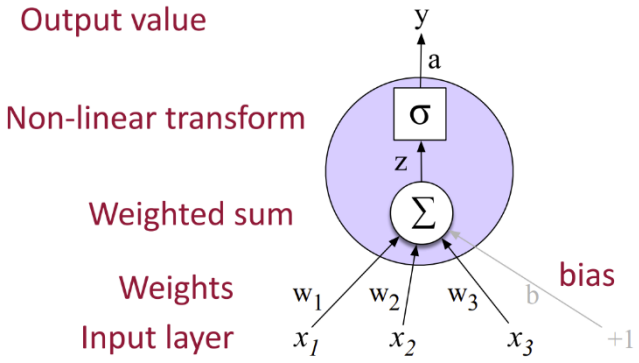
- Le fait d'avoir également la phrase suivante fera en sorte que les caractéristiques des mots *(cat, dog)* se rapprochent les unes des autres.

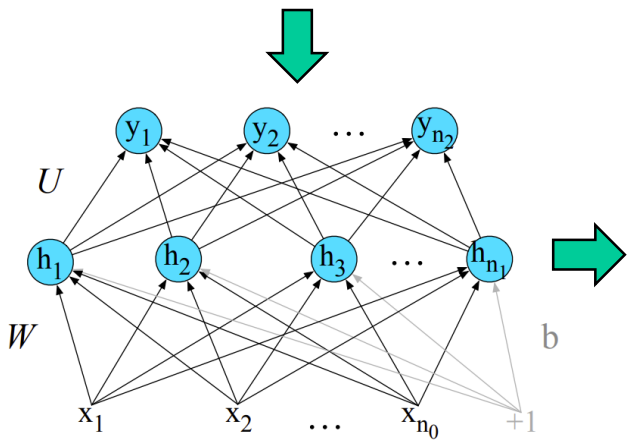
*A cat was running in a room*

# Modèles neuronaux

## Feedforward neural networks (RN à propagation avant / MLP - CF cours RN)

- Réseaux entièrement connectés → Fully-connected networks (W, U)
- Fenetre glissante de taille fixe (taille = N-Gram)
- Remplacement des “hand-crafted features” (inputs) par du “word embedding” (via E)




output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

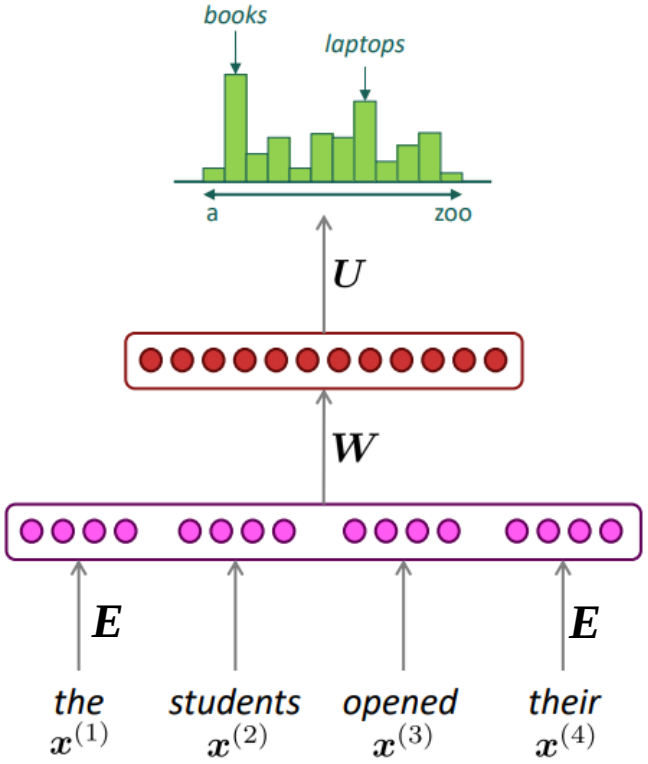
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

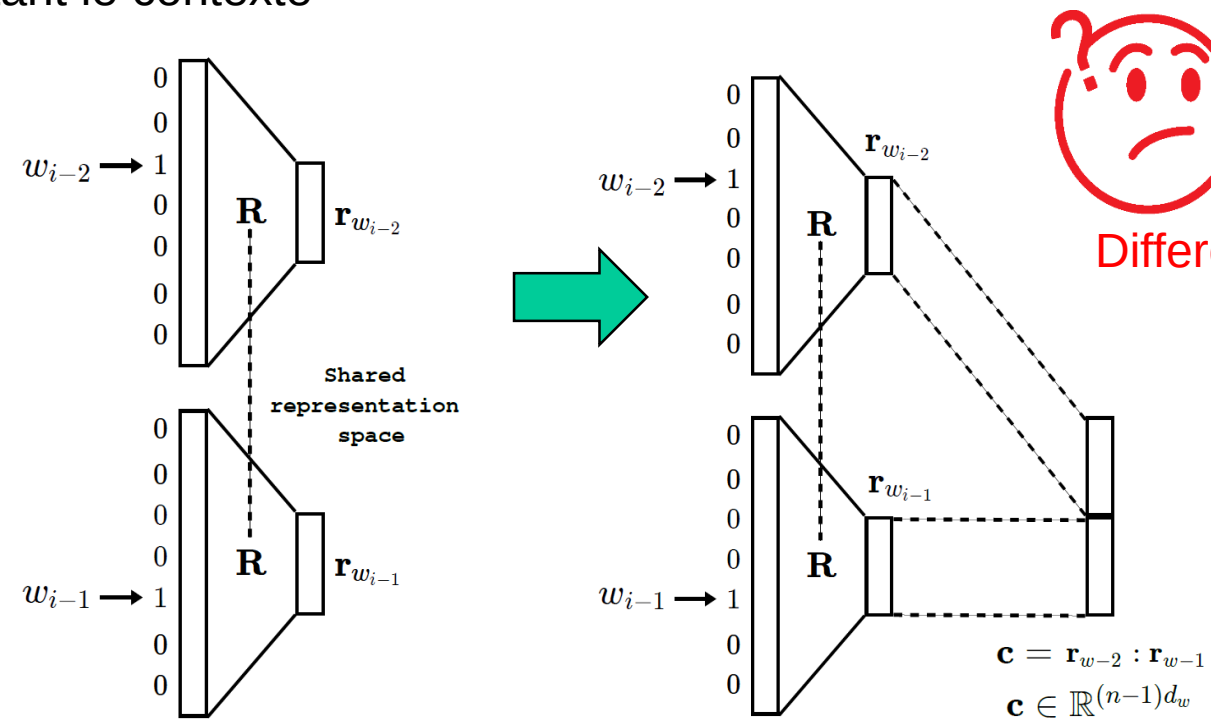
$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



# Modèles neuronaux

## 1/ Word Embeddings de N-grams

- L'entrée est un ensemble de *N-1* vecteurs one-shot du vocabulaire *V*
- Couche dense connectée à une couche plus petite, de dimension *dw*.
- *R* contient les paramètres du embedding de mots.
- Les embedding de mots **sont fusionnés (concaténés)** en un unique vecteur *c* representant le contexte






Difference avec Word2Vec ?

2 embedding pour des Tri-grams (N=3)

# Modèles neuronaux

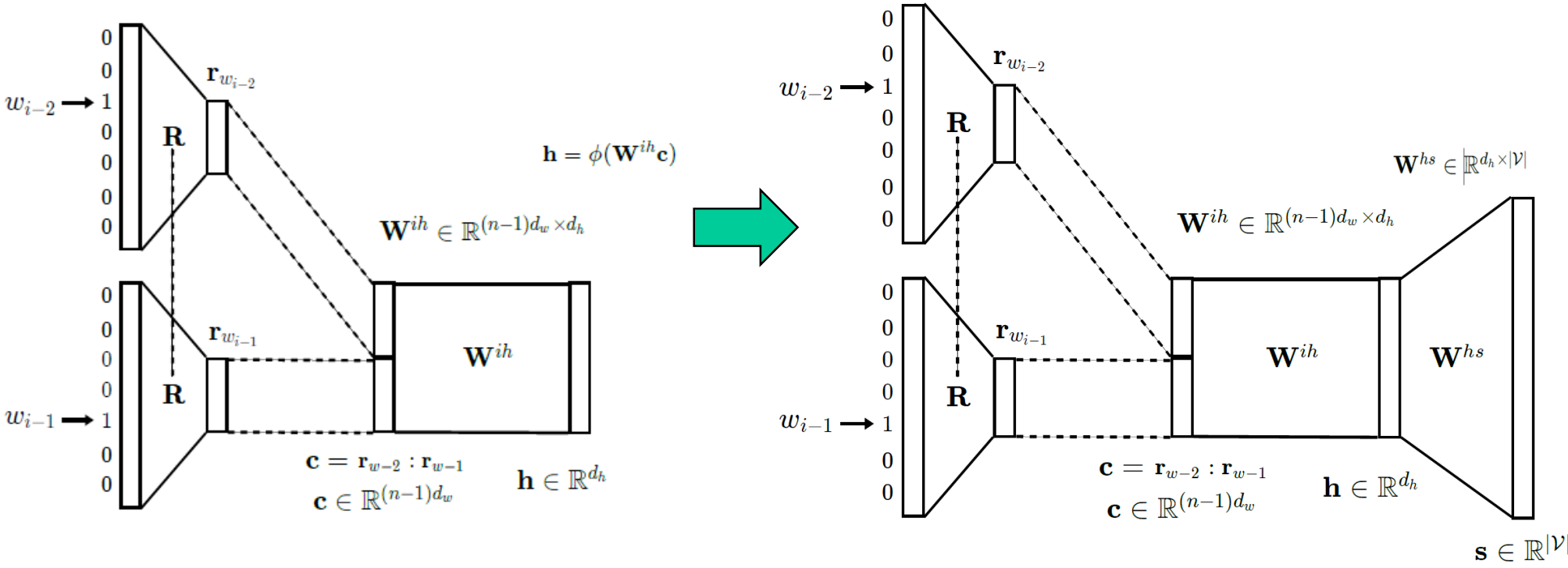
## 2/ Couches cachées → 3/ Output

- Étant donné la représentation du contexte, création d'1 représentation cachée

$$\boldsymbol{h = \phi(W^{ih}.c)}$$

- Etant donné h, obtention de scores pour tous les mots de V

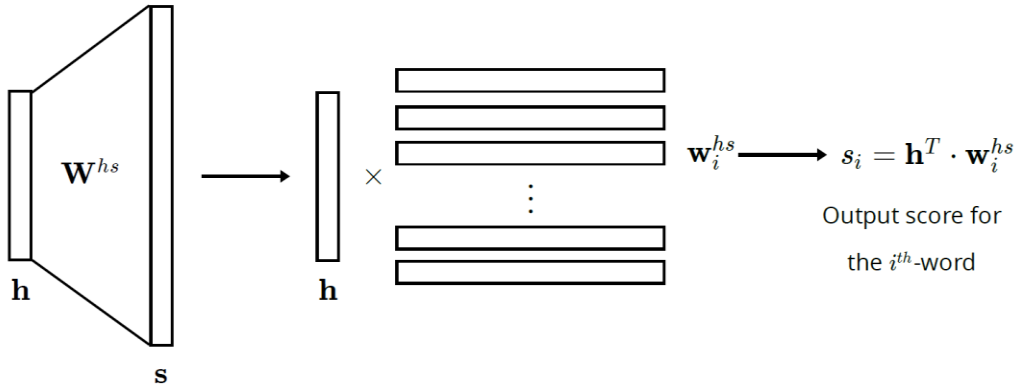
$$\boldsymbol{s = W^{hs}.h}$$



# Modèles neuronaux

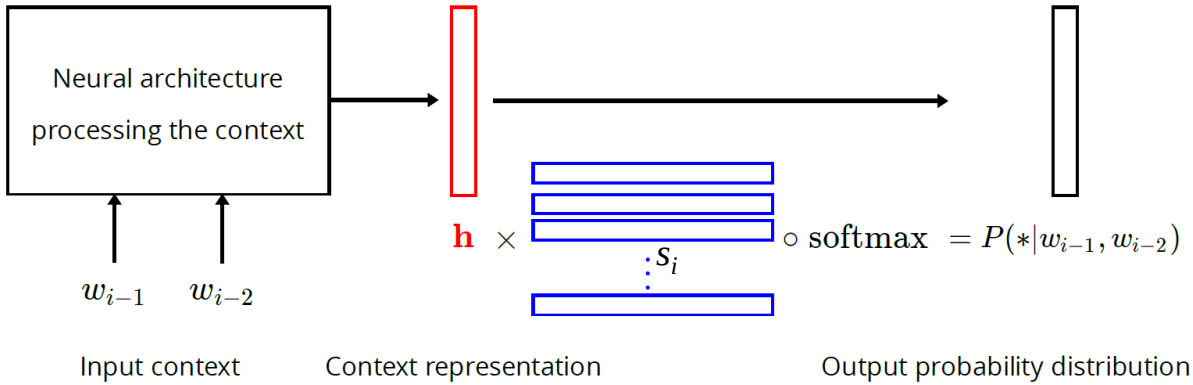
## 3/ Output

- La dernière couche peut être vue comme un produit scalaire entre *h* et un embedding des mots de sortie




- Ensuite, on estime les probabilités pour tous les mots d'indices $k \in V$ étant donnés $w_{i-1}$ ,$w_{i-2}$ **avec la fonction softmax** :

$$P(k|w_{i-1}, w_{i-2}) = \frac{\exp(\mathbf{s}_k)}{\sum_{l=1}^{|\mathcal{V}|} \exp(\mathbf{s}_l)} = \frac{\exp(\mathbf{h}_i^T \mathbf{w}_k^{hs})}{\sum_{l=1}^{|\mathcal{V}|} \exp(\mathbf{h}_i^T \mathbf{w}_l^{hs})}$$



# Modèles neuronaux

## 4/ Apprentissage

- $\theta = (R, W^{ih}, W^{hs})$ sont les paramètres du modèle **appris conjointement**
- Nous voulons que la distribution de probabilité de la sortie du modèle à chaque pas de temps (i) se rapproche de la vérité terrain
- En minimisant l'entropie croisée, à chaque étape, nous minimisons la log-vraisemblance négative de l'échantillon de données

$$d\left(\begin{bmatrix}0\\ \vdots \\ 1 \\ \vdots \\ 0\end{bmatrix}, \begin{bmatrix}P_\theta^{(i)}(1|w_{<i})\\ \vdots \\ P_\theta^{(i)}(w_i|w_{<i})\\ \vdots \\ P_\theta^{(i)}(|\mathcal{V}||w_{<i})\end{bmatrix}\right) ? \longrightarrow \begin{aligned}&\text{Cross-entropy}(P_*^{(i)}, P_\theta^{(i)}(*|w_{<i}))\\ &= -\sum_{k=1}^{|\mathcal{V}|} P_*^{(i)}(k) log(P_\theta^{(i)}(k|w_{<i}))\\ &= -log(P_\theta^{(i)}(w_i|w_{<i}))\ !\end{aligned}$$

- Pour tous les échantillons de données D = $(wi)_{i=1..N}$, , l'objectif est l'estimation de la plus grande vraisemblance (MLE).

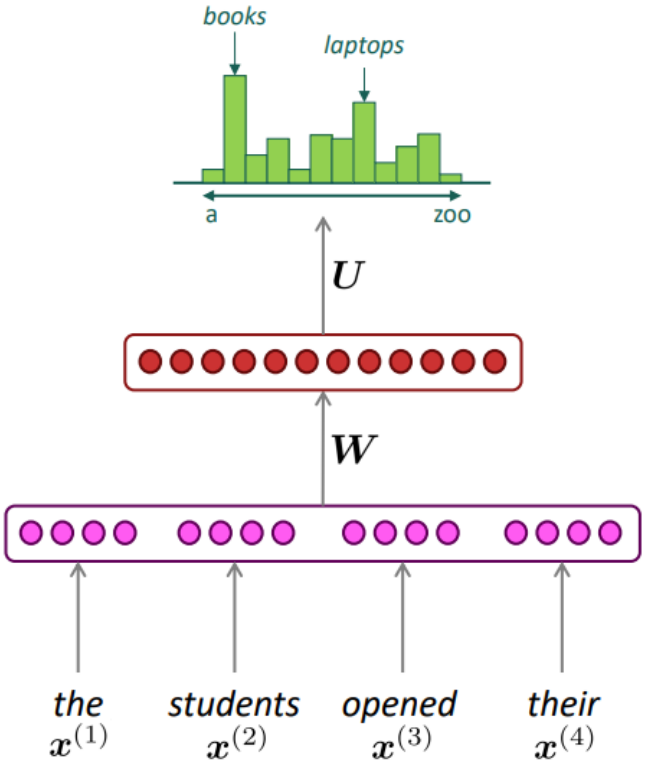
$$NLL(\theta) = -\sum_{i=1}^{N} \log(P_\theta^{(i)}(w_i|w_{<i}))$$

# Modèles neuronaux

## Bilan sur les Feedforward neural networks pour les ML

- Améliorations par rapport aux modèles n-gram :
  - Pas de problème de sparsité (lissage inutile)
  - Il n'est pas nécessaire de stocker tous les n-grams observés

- Problèmes restants :
  - La taille de fenêtre fixe est souvent trop petite
  - L'agrandissement de la fenêtre agrandit W
  - La fenêtre n'est jamais assez grande !
  - $x_1$ et $x_2$ sont multipliés par des poids complètement différents dans W
  - Aucune symétrie dans la façon dont les entrées sont traitées



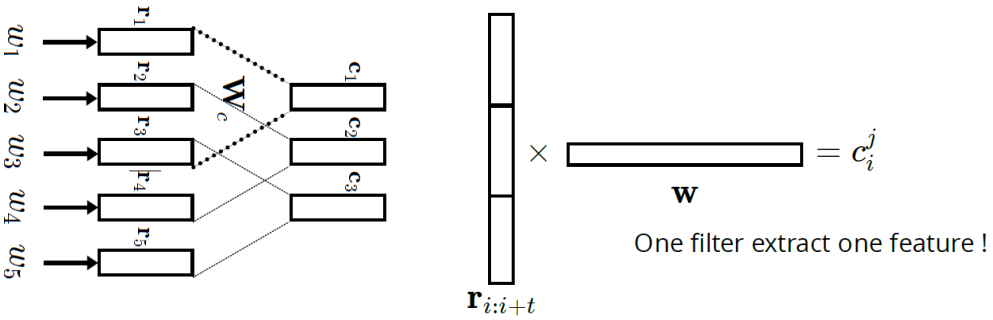

- Nous aurions besoin d'une architecture neuronale qui peut traiter n'importe quelle longueur de chaines en entrée.

Explications ?

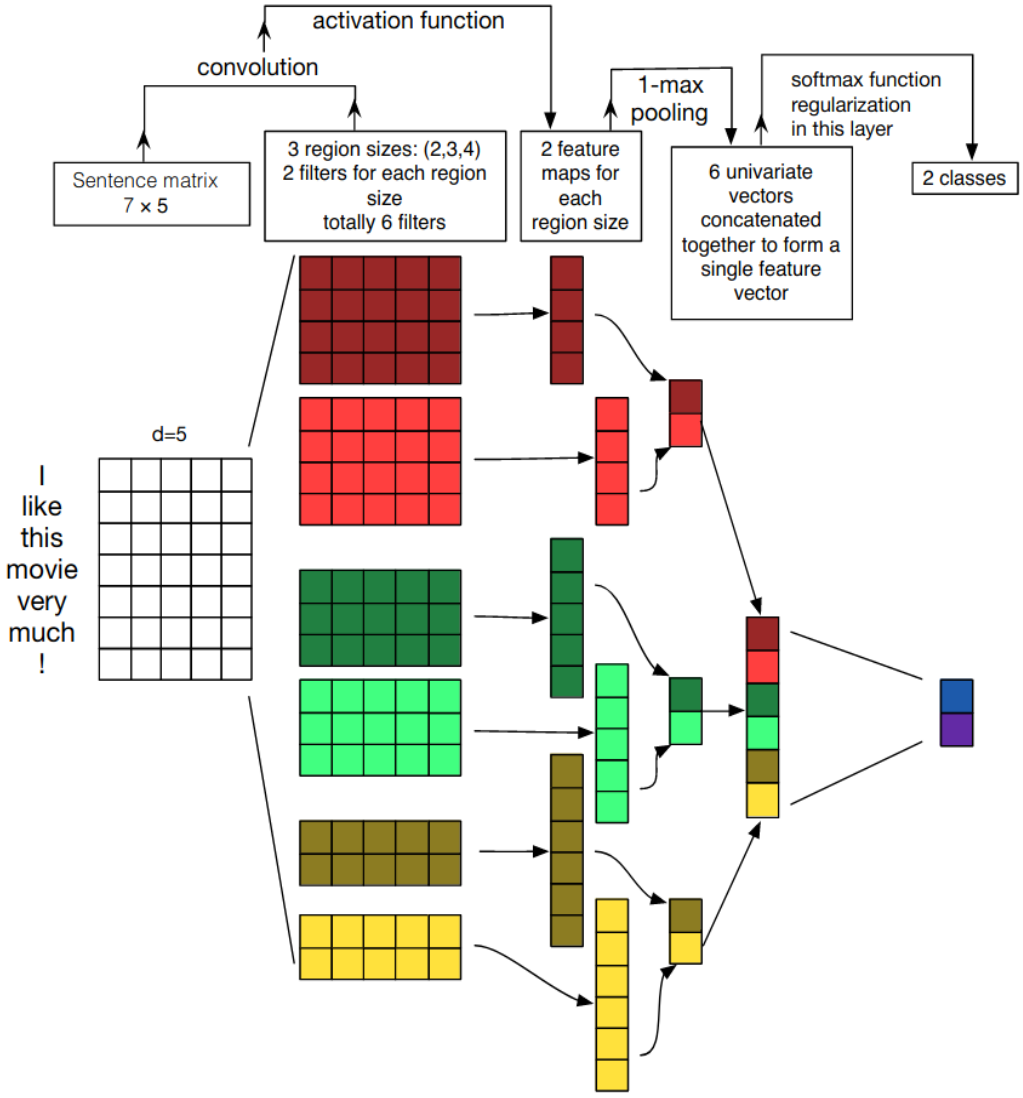
# Modèles neuronaux

## Approche avec Convolution et Pooling

- Les tâches de classification sur des séquences nécessitent une représentation globale
- L'idée est d'agglomérer consécutivement les caractéristiques obtenues sur des fenêtres de N-grams.
  - **Convolution** : Calculer plusieurs représentations pour chaque fenêtre (sous-séquence possibles d'une certaine longueur)




  - **Pooling :** Regrouper des éléments dans une représentation plus synthétique…
    → max, mean, …

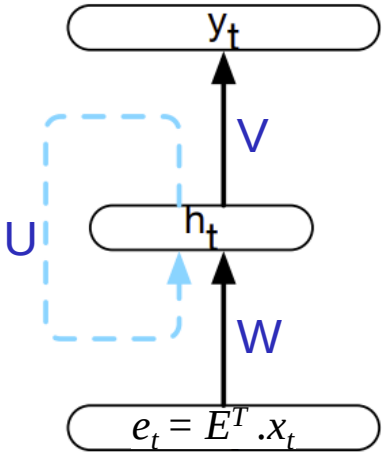



From "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification" (Ye Zhang, Byron Wallace, 2015)

# Modèles neuronaux

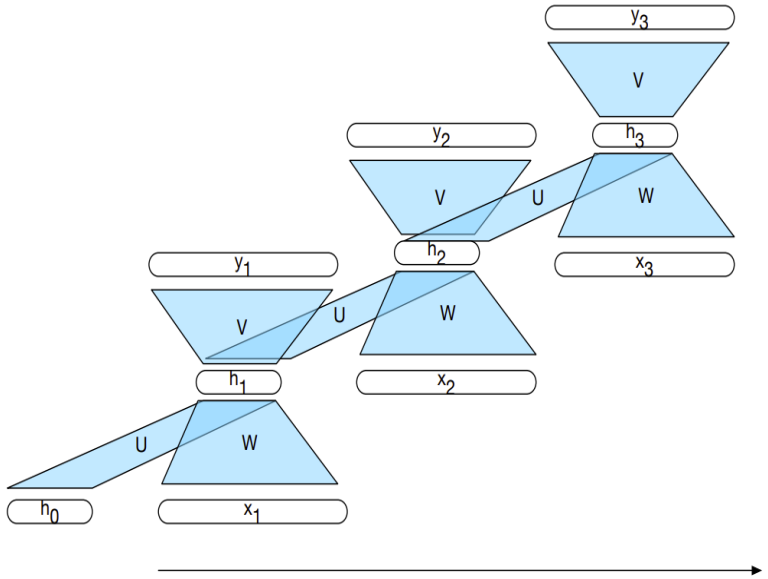
## RN Récurrents

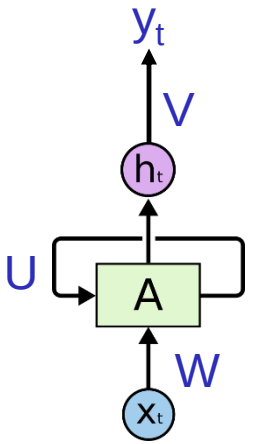
- Idée : Les poids W appliqués aux entrée restent fixes au cours du temps
- Présence de boucle dans les connexions entre neurones
- Les poids des connexions récurrentes (U) sont fixes également
- A un instant t, on calcule l'état $y_t$ en se basant sur l'état précédent $y_{t-1}$ et sur l'entrée actuelle $x_t$ → $y_{t-1}$ au lieu des mots contextes
- Ce type de RNN peut se représenter sous la forme d'un réseau FW adapté → utilisation des modes d'apprentissage classiques




$$
\begin{aligned}
e_t &= E^T x_t \\
h_t &= g(U h_{t-1} + W e_t) \\
y_t &= \text{softmax}(V h_t) \\
P(w_{1:n}) &= \prod_{i=1} P(w_i | w_{1:i-1}) \\
&= \prod_{i=1}^{n} y^i_{w_i}
\end{aligned}
$$

- *g() = sigmoid() ou tanh()*



# Modèles neuronaux

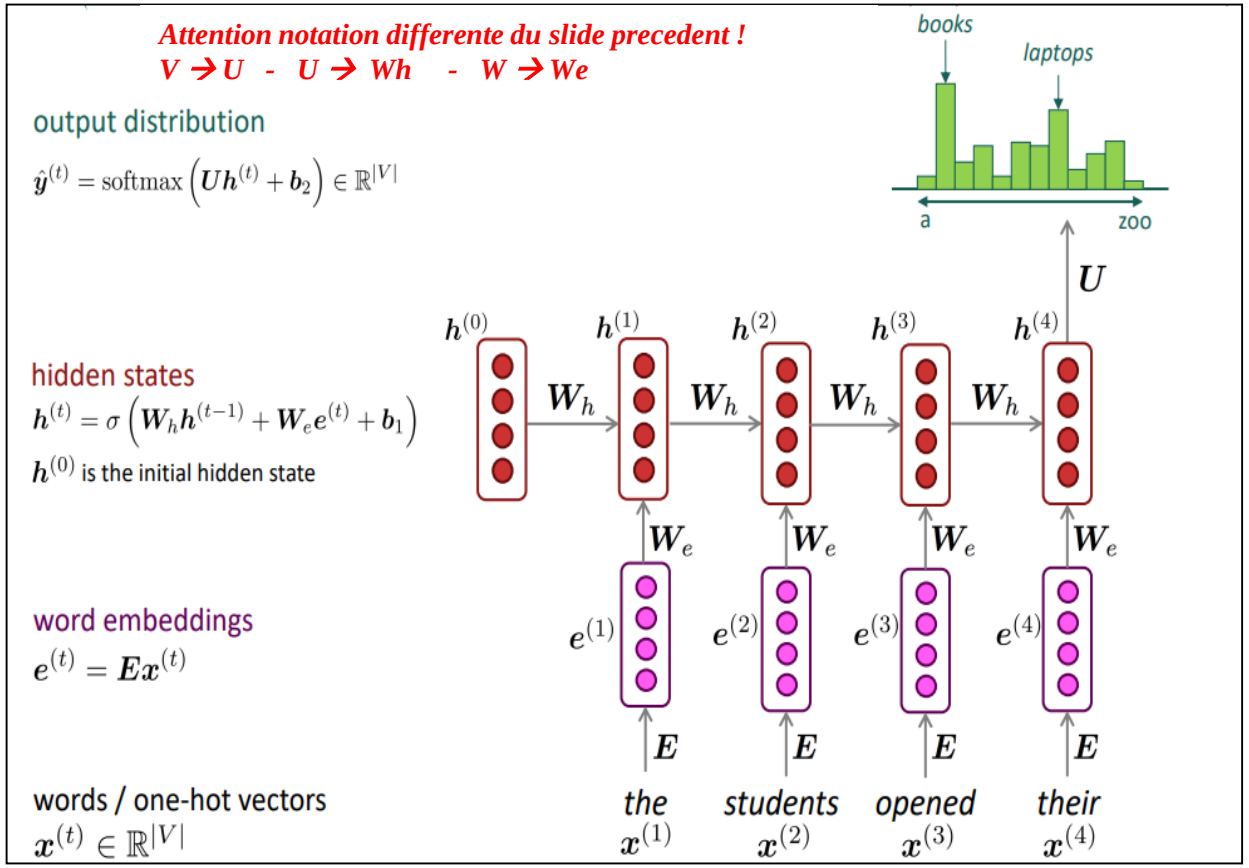
## RN Récurrents

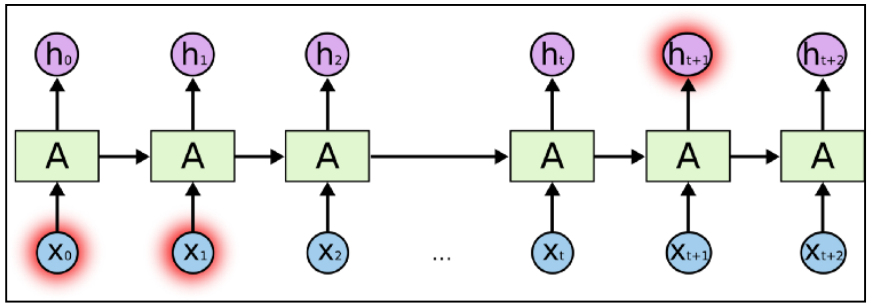
Avantages du RNN :

- Longueur d'entrée quelquonque
- La taille du modèle n'augmente pas pour contexte d'entrée plus long
- Les mêmes poids sont appliqués à chaque pas de temps → symétrie dans la façon dont les entrées sont traitées
- Le calcul à l'étape t peut (en théorie) utiliser des informations de plusieurs étapes passées

Inconvénients des RNN :

- Le calcul récurrent est lent
- Apprentissage plus difficile (descente de gradient)
- En pratique, il est difficile d'accéder à l'information de plusieurs étapes en arrière



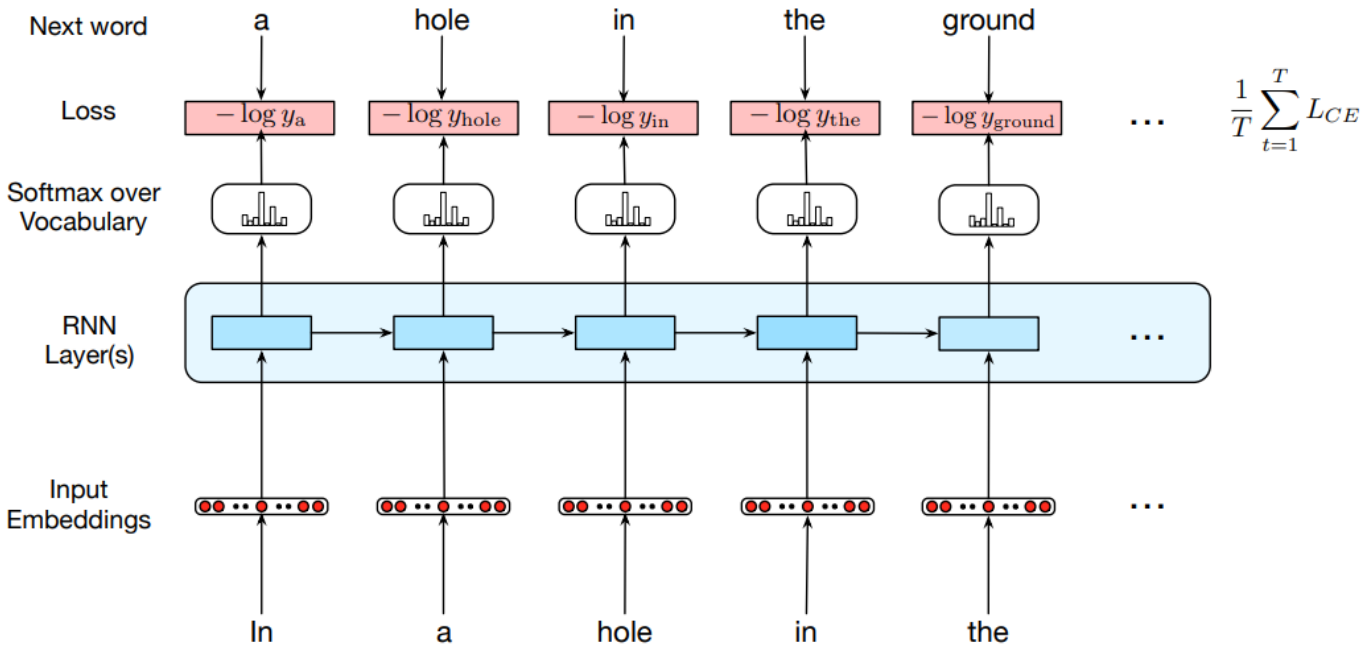



# Modèles neuronaux

## RN Récurrents

### Phase d'Apprentissage

- Loss = Moyenne des cross-entropies calculées sur une sequence



# Evaluation des modeles

- Mesure de Perplexité
  - Mesure l'incertitude : dans quelle mesure le modèle est-il surpris par de nouvelles données ?
  - Mesure la qualité de prédiction d'un modèle sur un corpus de test
  - Utilise la probabilité estimée sur chaque mot d'un corpus de test de taille T
  - Maximisation des probabilités de l'ensemble des mots du corpus
  - Le modèle avec une **perplexité minimale est le meilleur**

$$\text{perplexity} = \prod_{t=1}^{T} \left( \frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

  - Elle correspond à *exp( L(θ) )* *(Loss* cross-entropique sur le corpus de test – cf slide 23)

$$: \prod_{t=1}^{T} \left( \frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left( \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \textit{exp( L(θ) )}$$

# Evaluation des modeles

- Evaluation intrinsèque
  - Evaluer le modèle par rapport à sa représentation du langage
  - Exemple, comparer deux modèles en se basant sur leur capacité à bien représenter un dataset de test
  - Ne garantit pas une bonne performance du modele pour une autre tache

- Evaluation extrinsèque
  - Evaluer le modèle par rapport à une autre tache
  - Les modèles de langages sont très souvent utiliser pour évaluer des modèles conçus pour d'autres taches
  - Exemple, la qualité de traduction automatique en utilisant un modèle de langage
  - Evaluation souvent très couteuse

# Bilan ML

**Modèles de Langues**

- Tache centrale (prédiction mot suivant = classification multi-classes) pouvant servir de base pour de nombreuses autres taches de classification : POS, NER, Sentiment, …
- Modèles N-Grams
- Réseaux neuronaux à propagation avant, convolutionnels ou récurrents

**Limites des RN classiques pour la création de Modèles de Langues**

- Taille limitée du contexte dans les réseaux a propagation avant
- Problème de disparition du gradient dans les réseaux récurrents

  → Limiter la taille du contexte
- Mots hors vocabulaire

  → Limiter le vocabulaire et marquer le reste des mots par 〈UNK〉

**Utilisation d'architectures / réseaux plus avancés : LSTM, GRU, Transformers, …**

**→ CF suite du cours…**

# Bibliographie


- Eisenstein, J. (2018). Natural Language Processing.
- Jurafsky, D. (2020). Language modeling.Cours ”CS 124: From Languages to Information”. Stanford University.
- Mikolov, T., Karafi´at, M., Burget, L., Cernock`y, J., andKhudanpur, S. (2010). Recurrent neural network based language model.In In Proc. of Inter speech. http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf.
- Smith, N. (2020).Neural models. Cours ”Natural Language Processing (CSE 517)”. University of Washington.