Apprentissage profond neuronal, pourquoi et comment ?

Éric de la Clergerie
<Eric.De_La_Clergerie@inria.fr>

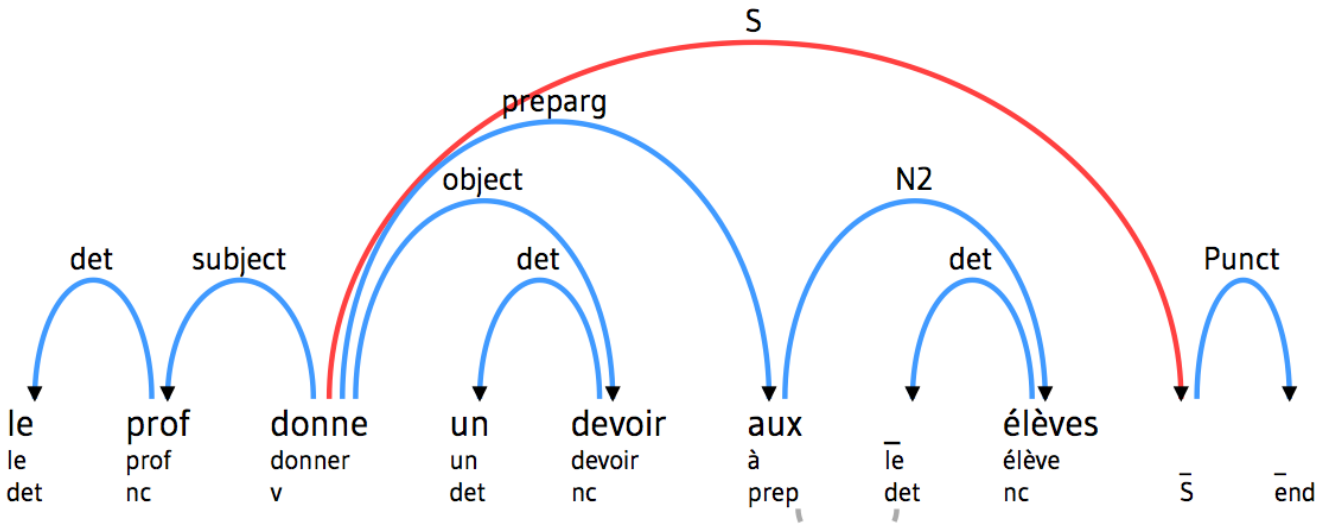
http://alpage.inria.fr
INRIA Paris-Rocquencourt

Journée ISN
Rocquencourt – 28 Mars 2018

Expliquer ce que je fais !

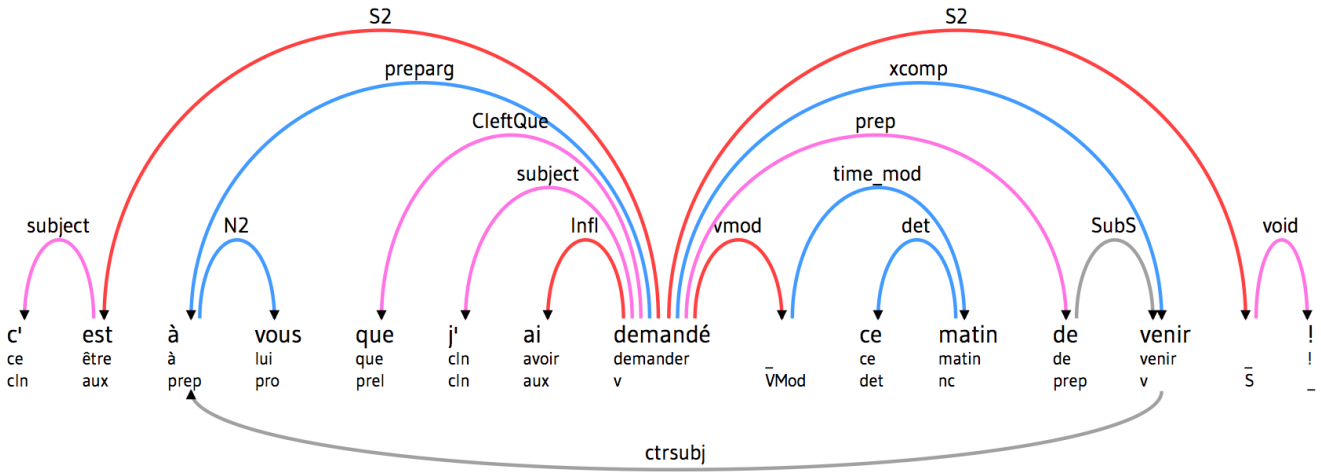
De l'analyse syntaxique,
une composante du Traitement Automatique des Langues (**TAL**).

L'ordinateur en classe de primaire,
objectif: Retrouver les fonctions et relations des mots dans la phrase



Vue par **dépendances**
en ligne sur `http://alpage.inria.fr/frmgwiki` (FRMG)

Mais c'est plus compliqué . . .



- de nombreuses constructions syntaxiques
- des rôles obscurs ou ambiguës dans la phrase
  il observe une maman avec ses jumelles
- . . .

En fait, beaucoup plus compliqué

Paul, je t'ai dit que François Flore est sorti furieux de chez son banquier car celui-ci lui avait ex abrupto refusé son prêt pour sa future maison ?

Pragmatique: contexte & connaissances
référants: celui-ci=banquier, lui=son=sa=François, t'=Paul
structures argumentatives: refus explique furieux
scénarios, implicites

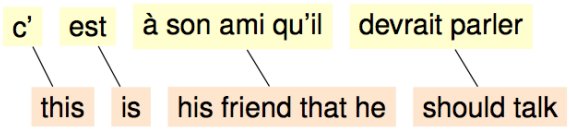
Sémantique: sens des énoncés et des mots
structure prédicatives, rôle des actants (agent, patient, ...)
`refuser(agent=celui-ci,patient=lui,theme=prêt)`

Syntaxe: structure des phrases et relations entre mots
fonctions syntaxiques (sujet, objet, ...) : **celui-ci**=sujet, **prêt**=objet, **lui**=obj indirect de **refusé**

Morphologie: les mots et leur structure (**lubéronisation**)
découpage du texte en mots, catégories syntaxiques:
`celui/pro -ci/adj lui/cld avait/aux ex_abrupto/adv ...`
flexion (conjugaison) : **avait**=`avoir+3s+Ind+Imparfait`
entités nommées (personnes, lieux, ...) : (François Flore) `PERSON_m`

Beaucoup d'applications potentielles . . .

2010: Google translate

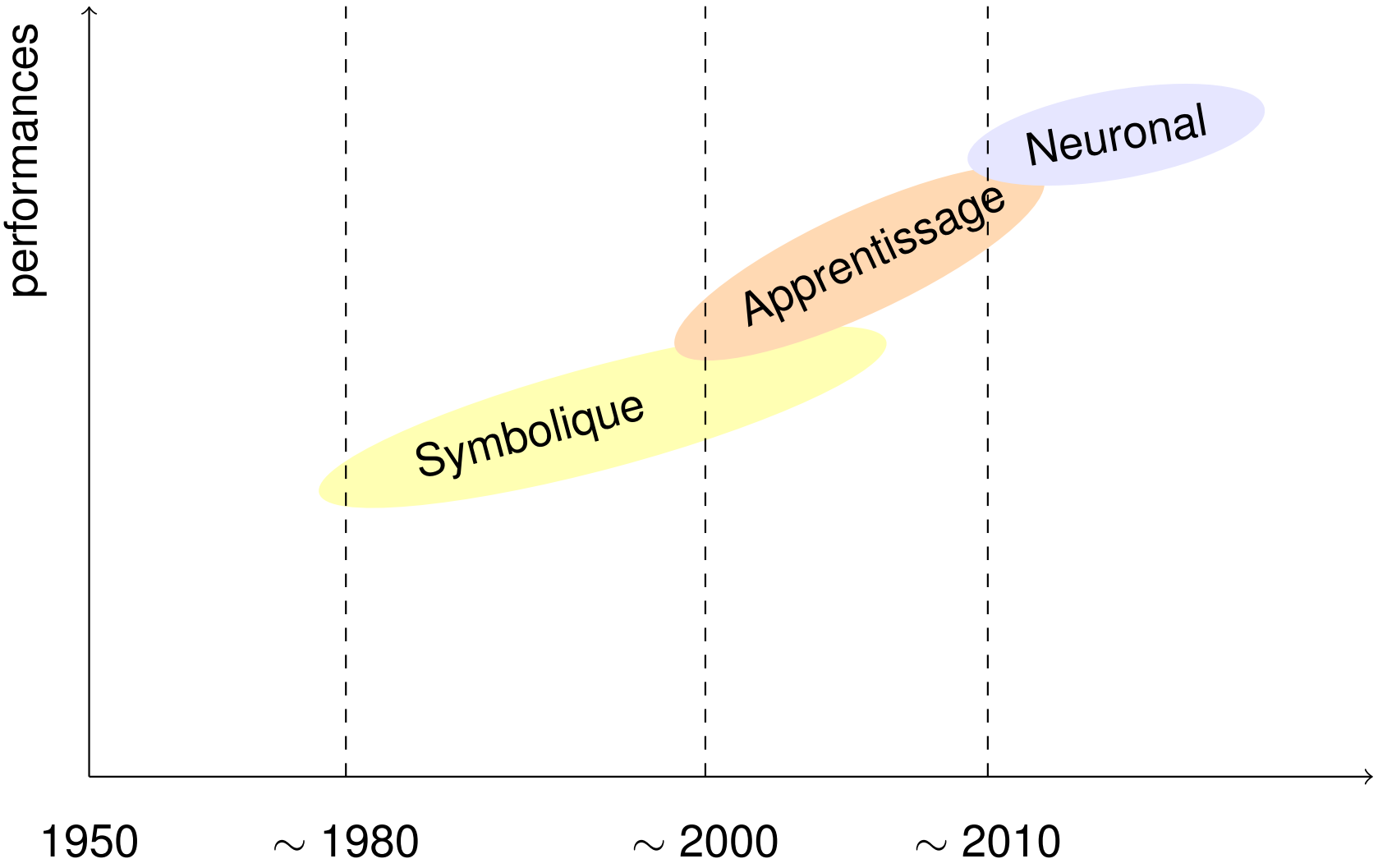


2011: IBM Watson gagne à Jeopardy

2011: Assistants vocaux Siri, Google Now, Cortana

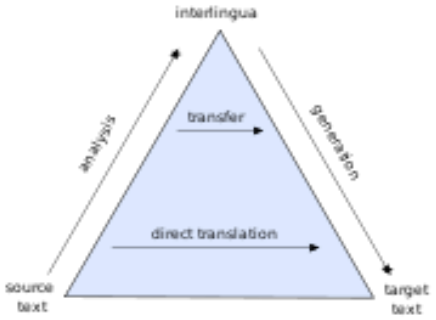
- **recherche** d'information, veille, extraction, questions-réponses
- **accès à l'information**: traduction, résumés, annotations/liens sémantiques
- analyse d'**opinion**, e-réputation
- **génération** (articles journaux, rapports, ...)
- plagiat, *authoring*, détection spams & faux commentaires,
- aide à la **rédaction**: correction grammaticale, stylistique; saisie prédictive
- aide **pédagogique**: apprentissage des langues; tutorat; évaluation
- **interaction**: agents conversationnels, chatbots, assistants numériques,

Une chronologie simplifiée



Traduction Automatique

Triangle de transfert

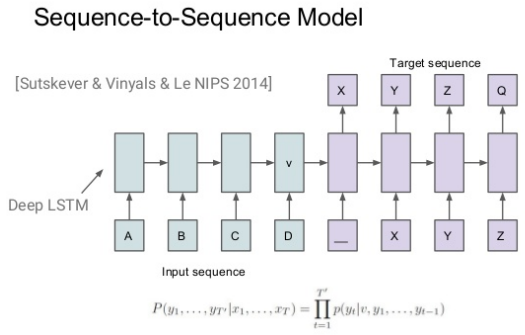


2010: Google translate

c' est à son ami qu'il devrait parler

this is his friend that he should talk

2014: seq2seq (avec attention)



- évolution vers une traduction de plus dirigée par les données de moins en moins de linguistique et de sémantique explicite !
  $\rightsquigarrow$ gain important en qualité
- mais requiert des volumes très importants de corpus alignés et des temps énormes d'apprentissage (semaines !)

malgré les quantités délirantes de données utilisées pour les entraîner, plus que ce qu'un humain pourrait voir en plusieurs vies, les erreurs faites par ces systèmes montrent qu'ils ne captent pas vraiment le sens commun, c'est-à-dire la compréhension générale du monde qui nous entoure.

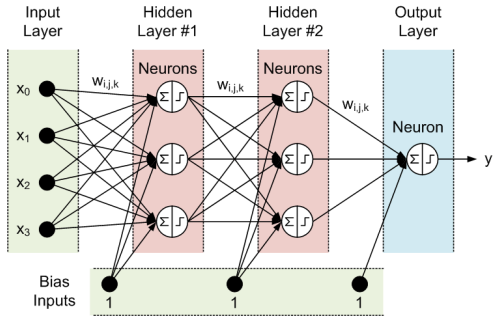
Yoshua Bengio (Le Monde)

Les données au coeur des nouvelles IA !

Masses de données textuelles (et autres)

Des algo d'apprentissage de + en + sophistiqués

deep learning



De la puissance de calcul

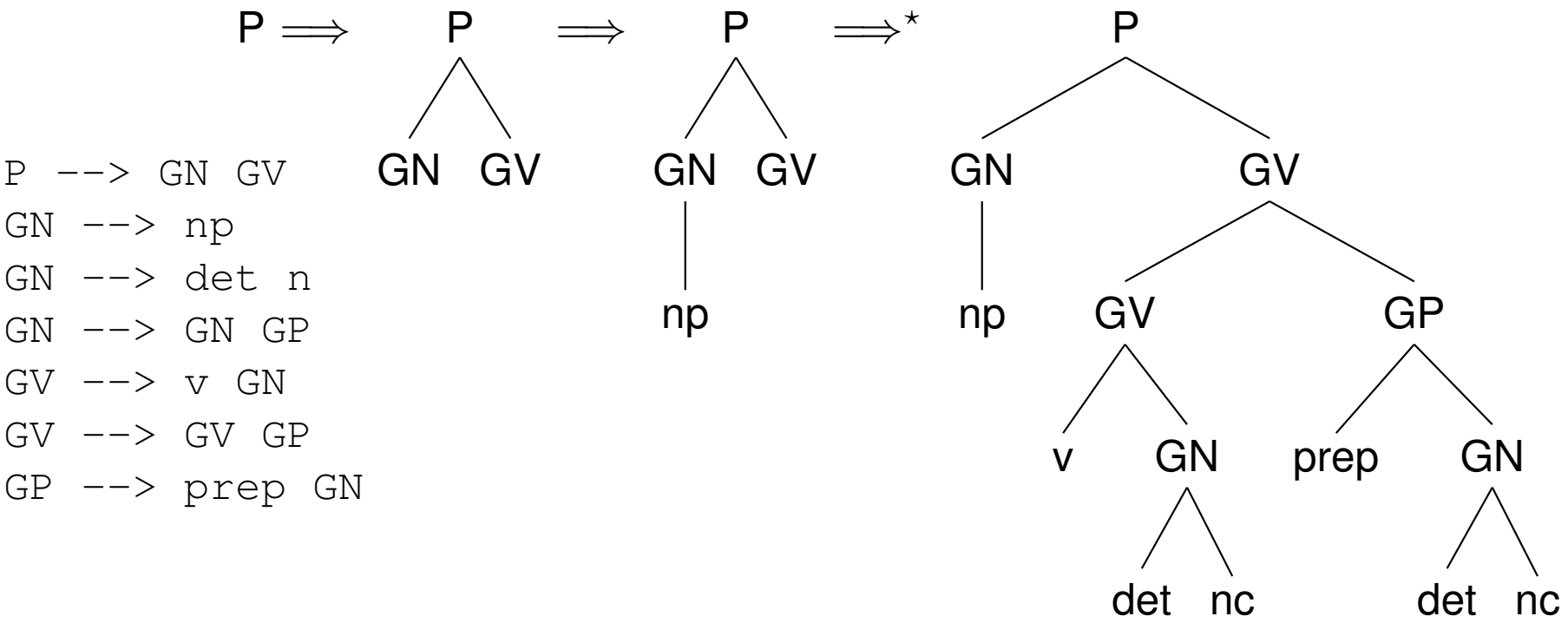
Mais on peut/doit discuter cette vue de l'IA ! raisonnement, déduction, induction, analogie, . . .

Outline

CFG & Langues naturelles

Les CFG utilisées pour réécrire des non-terminaux par substitution et obtenir des arbres d'analyse

Paul voit un homme avec un télescope




```
P --> GN GV
GN --> np
GN --> det n
GN --> GN GP
GV --> v GN
GV --> GV GP
GP --> prep GN
```

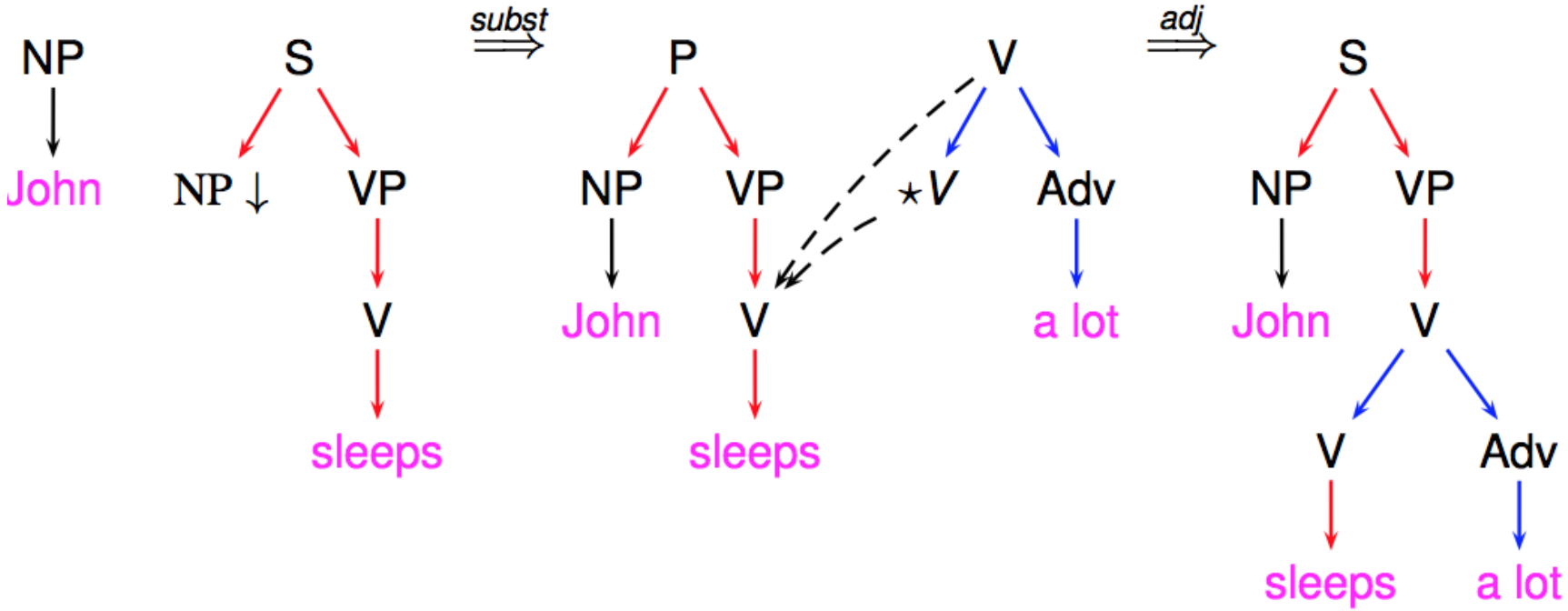
Les CFG suffisent pour de nombreux phénomènes syntaxiques

Grammaire d'Arbres Adjoints (TAG)

Introduites par Joshi, les TAG
s'appuient sur des arbres et 2 opérations pour les combiner

- la substitution d'une feuille par un arbre
- l'adjonction d'un arbre au sein d'un autre arbre

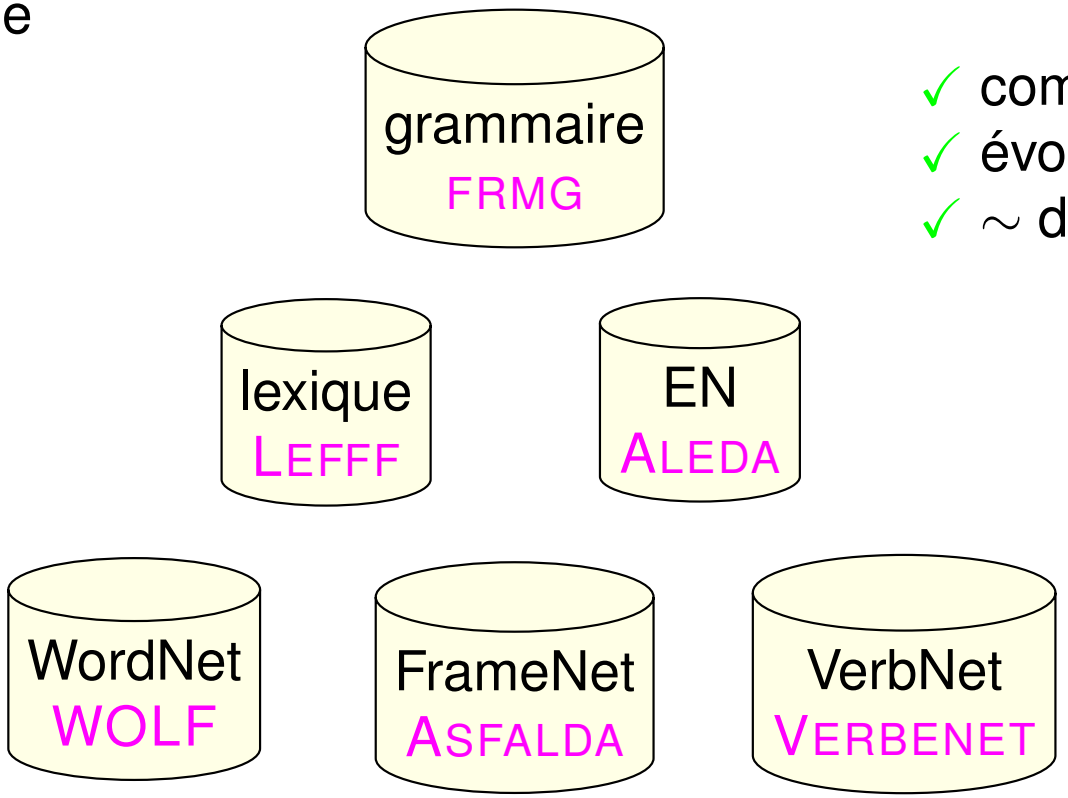
Aravind Joshi



Aproches symboliques (pro & cons)

Ecosystèmes (complexes) de ressources linguistiques

- ⊠ expertise linguistique
- ⊠ taille & complexité
- ⊠ faible couverture
- ⊠ trop fines
- ⊠ non probabilisées



- ✓ compréhensibles
- ✓ évolutives
- ✓ ∼ dev. logiciel

Outline

Une formulation probabiliste générique

Étant donnée une observation x, trouver la meilleure structure (cachée) y expliquant x (décodage)

- une séquence d'étiquettes $y = t_1 \cdots t_n$ for chaque mot dans $x = w_1 \cdots w_n$
- un arbre syntaxique T couvrant $w_{1:n}$

Approches discriminatives: fondées sur des configurations

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(y|x)$$

Approches génératives: fondées sur la combinaison de structures

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(y|x) = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, \frac{p(x|y)p(y)}{p(x)} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, \overbrace{p(x|y)}^{\textit{likehood}} \overbrace{p(y)}^{\textit{prior}}$$

Étiquetage syntaxique

Tâche: Assigner une catégorie syntaxique à chaque mot, problème des ambiguïtés

La/DET belle/N ferme/V le/DET voile/N
La/DET belle/ADJ ferme/N le/CL voile/V

Mais aussi:
La/CL belle/ADJ ferme/V le/CL voile/N
La/DET belle/N ferme/V le/DET voile/V
La/N belle/N ferme/V le/DET voile/V

$\rightsquigarrow$ Un nombre exponentiel de séquences d'étiquettes $O(|T|^n)$

Methodes discriminantes

Des probabilités conditionnelles pour étiqueter $x = w_1 \cdots w_n$ par $y = t_1 \cdots t_n$

$$\hat{y} = \underset{y \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(y|x) = \underset{\hat{t}_{1:n} \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(t_1|x).p(t_2|x,t_1).\cdots.p(t_n|x,t_{1:n-1})$$

chaque $x, t_1, \cdots t_k$ est vu comme une configuration c_{k-1}

$$\hat{y} = \underset{\hat{t}_{1:n} \in \mathcal{Y}(x)}{\operatorname{argmax}}\, p(t_1|c_0).p(t_2|c_1).\cdots.p(t_n|c_{n-1})$$

Mais pas assez de données pour calculer des probabilités fiables $p(.|c)$ pour chaque configuration c

Abstraire par des traits

Une configuration c résumée par un ensemble de propriétés (trait – *features*)

$$p(.|c) \sim p(.|f(c))$$

La configuration c

Paul/PN pense/V **que le chat dort**

abstraite par

f_1	le mot courant est **que**
f_2	le mot précedent est **pense**
f_3	le mot à -2 est **Paul**
f_4	l'étiquette du mot à -1 est `v`
f_5	le mot à -2 commence par une majuscule
⋮	⋮
f_{93}	les 2 étiquettes précédentes sont `pn v`
f_{100}	les 2 mots précédents sont **Paul pense**
⋮	⋮

Pondérer les traits

On approche $p(t|c)$ à partir de poids $\lambda_{i,t}$ associés aux traits f_i de c
Méthode MaxEnt (ou régression logistique)

$$p(t|c) = \frac{e^{\Sigma_i \lambda_{i,t} f_i}}{Z}$$

avec (normalisation)

$$Z = \Sigma_t e^{\Sigma_i \lambda_{i,t} f_i}$$

- Décodage (*glouton – greedy*): à chaque mot w_k et configuration c_{k-1}

$$t_k = argmax_t p(t|c_{k-1})$$

- Décodage par faisceau (*beam*), on garde les b meilleurs configurations c_k à chaque étape

Calcul des paramètres: perceptron

Comment calculer les paramètres $\lambda_{i,t}$ pour les traits f_i ?

Apprentissage possible sur des corpus annotés par perceptron (rétroaction)
méthode itérative avec mise à jour des $\lambda_{i,t}$ en fonction des succès et des échecs

- on initialise $\lambda_{i,t}^{(0)}$ (aléatoirement ou par décompte)
- si pour une configuration c, on choisit $t = \text{argmax}_t\, p(t|c)$ en place de la bonne étiquette s
  alors
  - on pénalise les paramètres pour t par μ

$$\lambda_{i,t}^{(m+1)} = \lambda_{i,t}^{(m)} - \mu$$

  - on favorise les paramètres pour s par μ

$$\lambda_{i,s}^{(m+1)} = \lambda_{i,s}^{(m)} + \mu$$

En pratique, convergence des paramètres

Limites des approches discriminantes

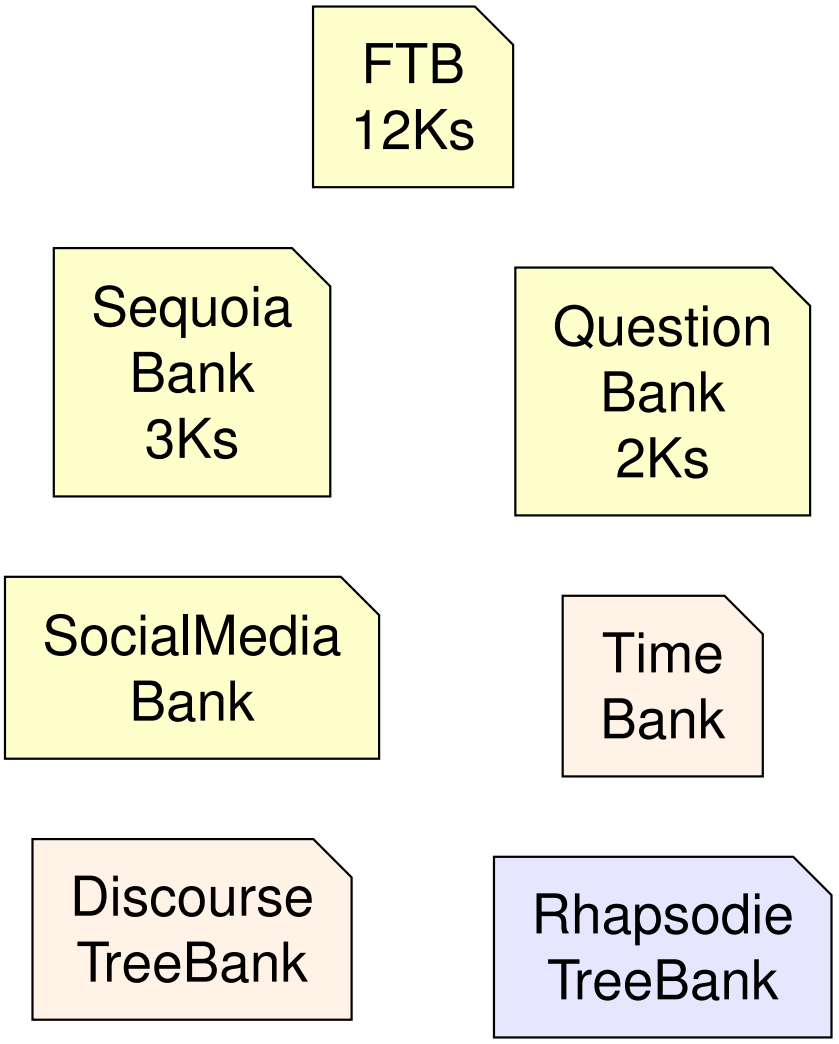
Les approches discriminantes (Perceptron, MaxEnt, SVM, CRF, . . .) donnent d'excellents résultats.

Mais

- elles nécessitent un effort important d'identification du bon jeu de traits.
  $\Longrightarrow$ en pratique, on considère d'immense jeux
  (plusieurs millions à dizaines de millions de traits)
- des corpus annotés de grande taille (surtout si de nombreux traits)
  Brown Corpus: 1Mmots
- des soucis pour des mots inconnus/rares dans les corpus annotés
  (loi de Zipf)

Succès et limites des approches supervisées

Courant majoritaire: données annotées et apprentissage supervisé

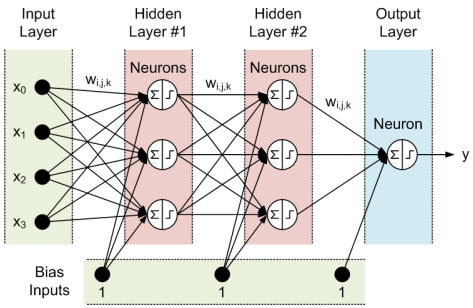


- ✓ efficace
- ✓ découplage
- ✓ robustesse
- ✓ évaluation
- ✓ autonome

- ⊠ coût humain
- ⊠ fastidieux
- ⊠ peu évolutif
- ⊠ sensibilité domaine
- ⊠ **expertise traits**
- ⊠ boite noire

Outline

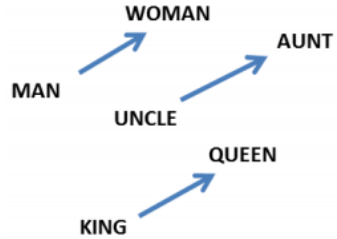
IA ? On y est presque :-)



Réseaux de Neurones: le retour !
Buzz sur Deep Learning et word embeddings

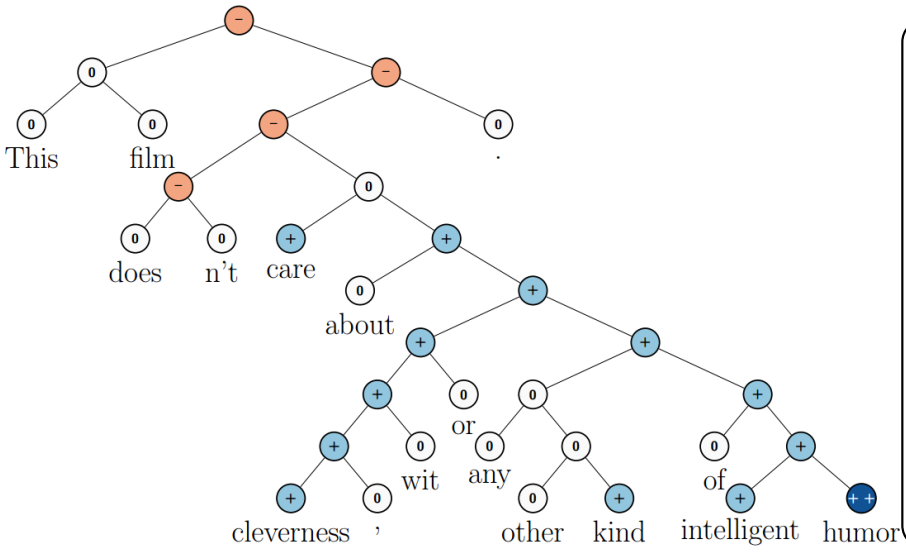
2013: Word embeddings

analogies ≡ calcul vectoriel



Mikolov et al.

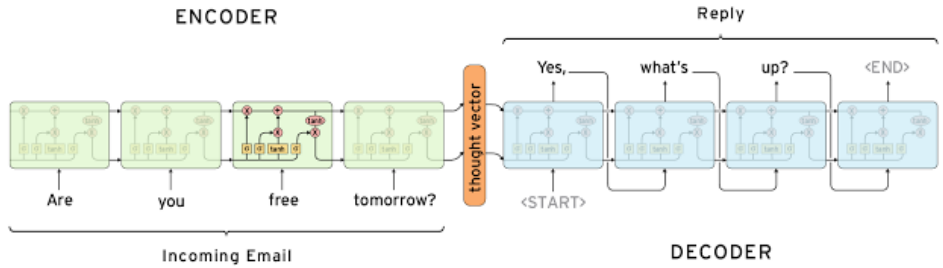
2014: Analyse d'opinions



Socher et al.

2015: Google SmartReply

suggérer des réponses aux mails



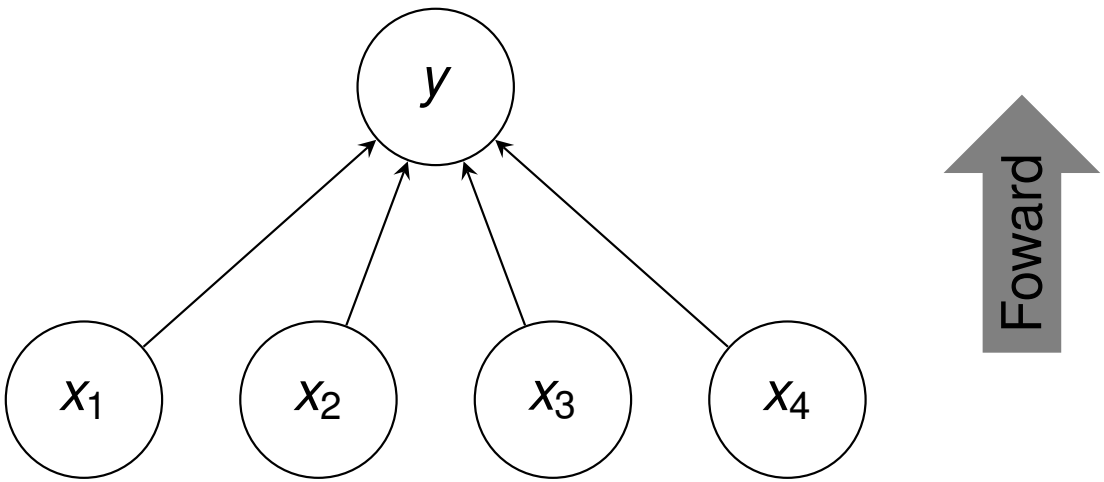
http://goo.gl/d8R2LI

- DeepMind (Google) annonce Neural Turing Machine
- Labo Facebook Paris
- Toolkit Google TensorFlow libéré
  nombreux autres toolkits

Bases

Modélisation des neurones biologiques:

- les neurones portent des vecteurs (de réels) de dimension d entre 100 à 500
- les vecteurs x_i associés à des neurones d'entrées sont combinés pour fournir un vecteur y à un neurone de sortie



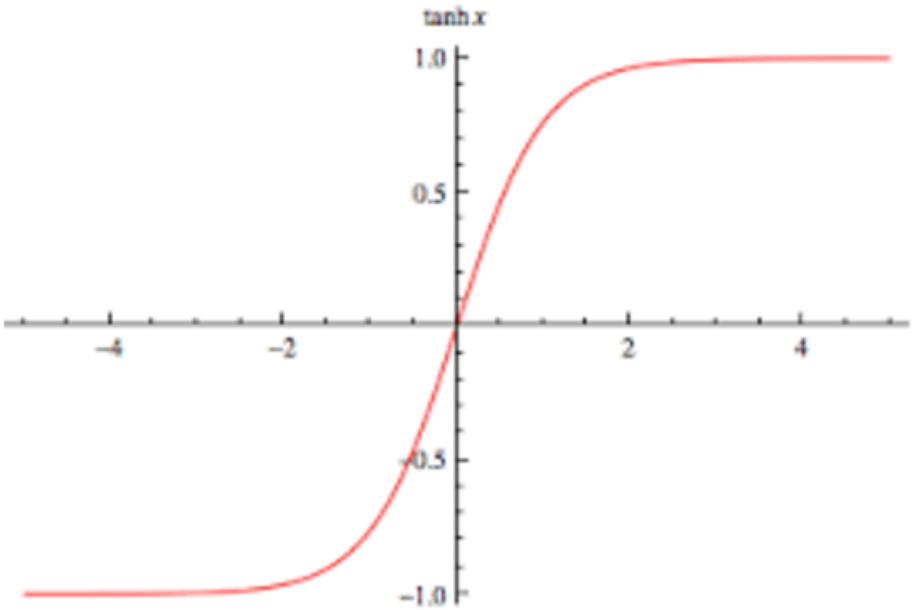
$$y = \sigma(\Sigma_i A_i x_i) \text{ avec } A_i \text{ matrice}$$

Fonction d'activation σ: en générale non linéaire
elle doit faire basculer l'état du neurone de sortie vers *oui* ou *non*

Fonctions d'activation

Utilisation de fonctions proches d'une bascule *oui*/*non* mais dérivables

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \tanh'(x) = 1 - \tanh^2(x)$$

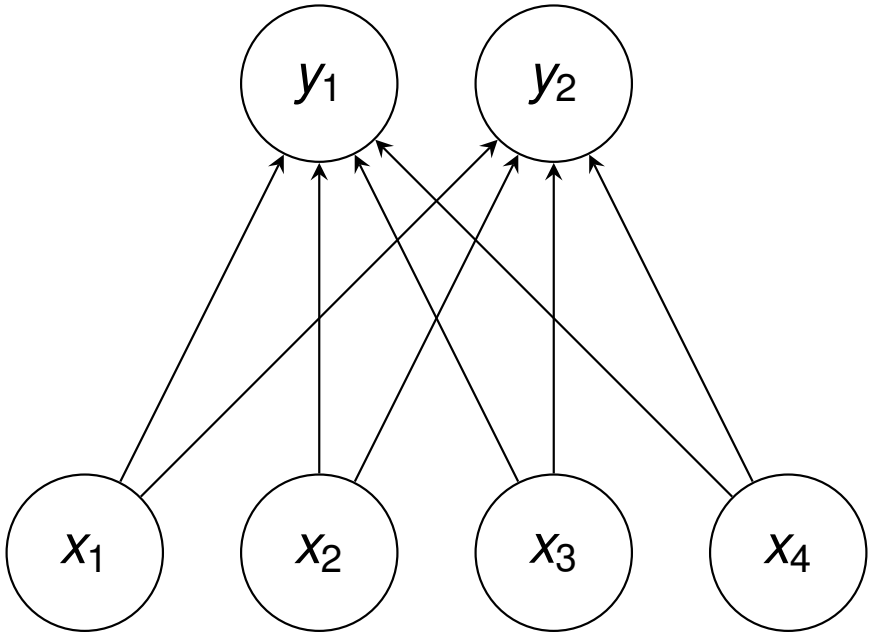


tangente hyperbolique `tanh`

D'autres fonctions sont aussi utilisées (*softmax*, *sigmoïde*)

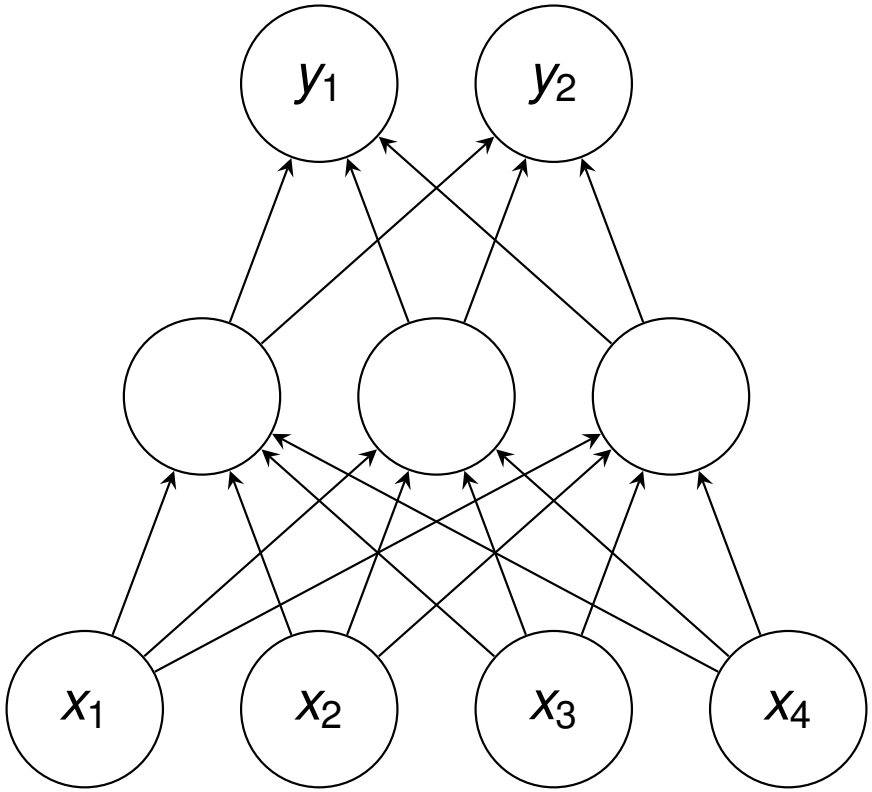
Sorties multiples

On peut avoir plusieurs neurones de sortie



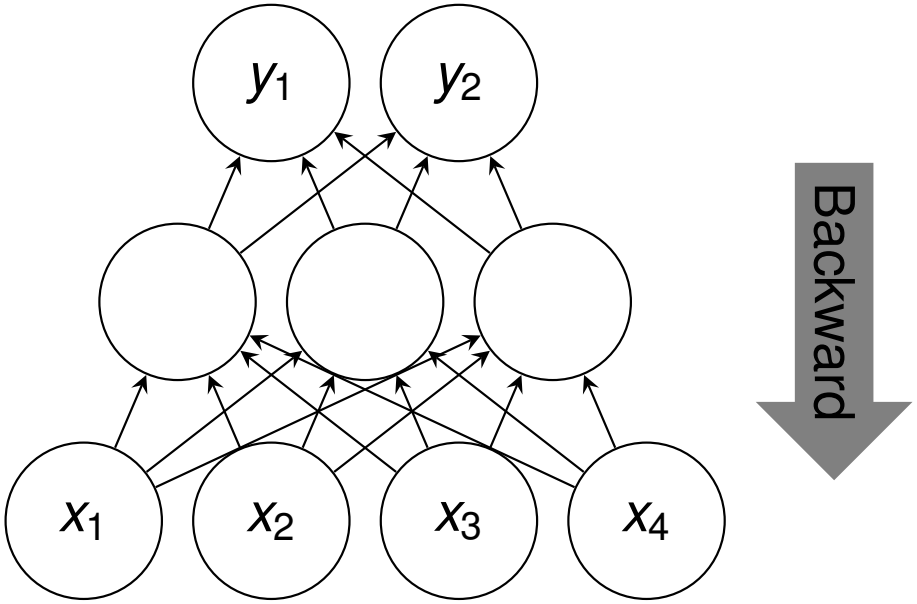
Couches cachées

On peut intercaler des couches cachées intermédiaires



Les couches permettent de progressivement abstraire les informations des neurones d'entrée $\rightsquigarrow$ **traits** implicites (difficiles à interpréter)

Apprentissage: *backpropagation*



Similaire au perceptron

- on fait descendre les erreurs des neurones de sortie vers les neurones d'entrée (*backpropagation*)
- mise à jour des paramètres W_i par descente de gradient

$$W_i^{(t+1)} = W_i^{(t)} - \mu * \frac{\partial y}{\partial W_i}$$

Mais trop de couches $\Longrightarrow$ évaporation des gradients (proches zero)

Enchassement de mots

En TAL, on part de mots $\Longrightarrow$ associer des vecteurs de réels comme entrées des réseaux (*enchassement* ou *word embeddings*):

- On peut partir de vecteurs initialisés au hasard et les faire évoluer par apprentissage
  mais quid des mots inconnus ou rare (pas ou mauvais vecteur)

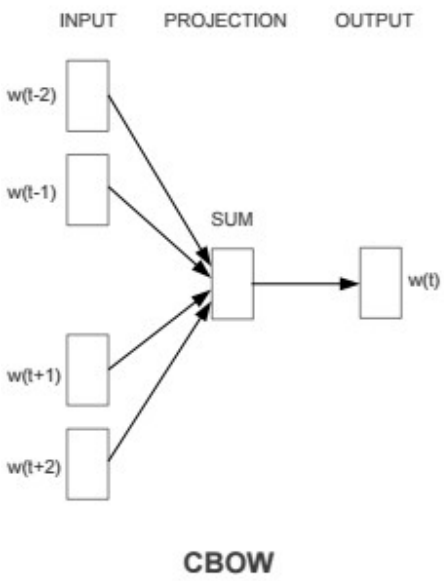
- ou utiliser des vecteurs appris sur de très gros corpus en exploitant l'hypothèse distributionnelle (Harris)

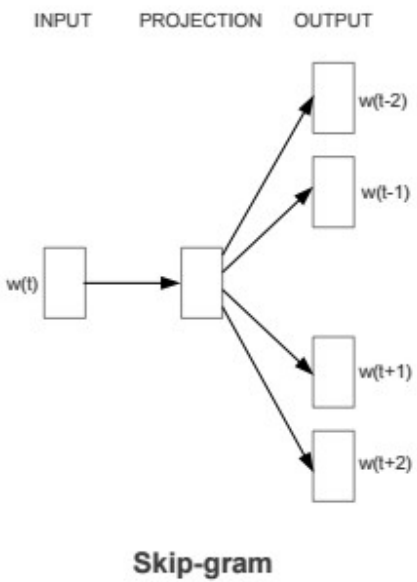
  > *des mots apparaissant dans des contextes similaires tendent à être proches sémantiquement.*

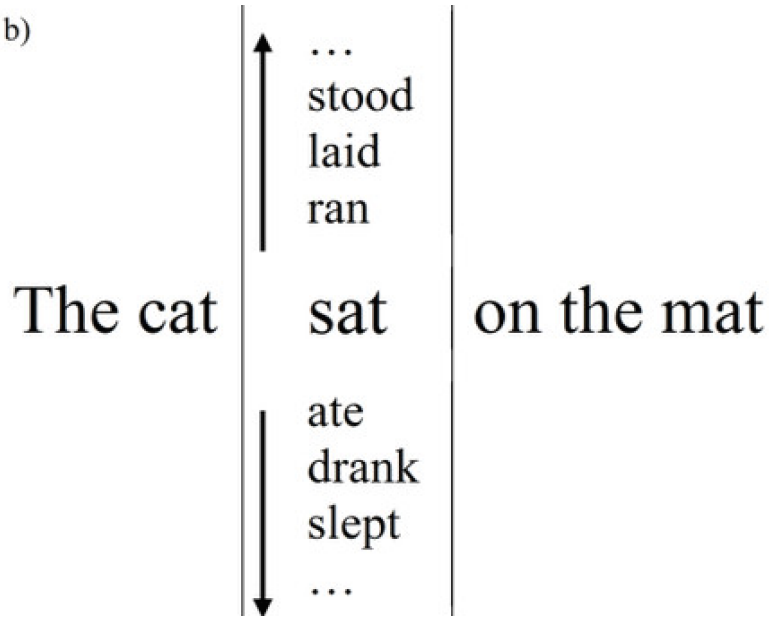
`word2vec` (Mikolov) produit des vecteurs tels que des mots proches (sémantiquement, syntaxiquement, . . .) ont des vecteurs proches (similarité *cosinus*).

2 vues duales de la notion de contexte pour produire les vecteurs

- prédire le mot courant, étant donné les *n* mots de contexte autour – **CBOW**
- prédire les *n* mots autour, étant donné le mot courant en contexte – **SKIP-GRAM**



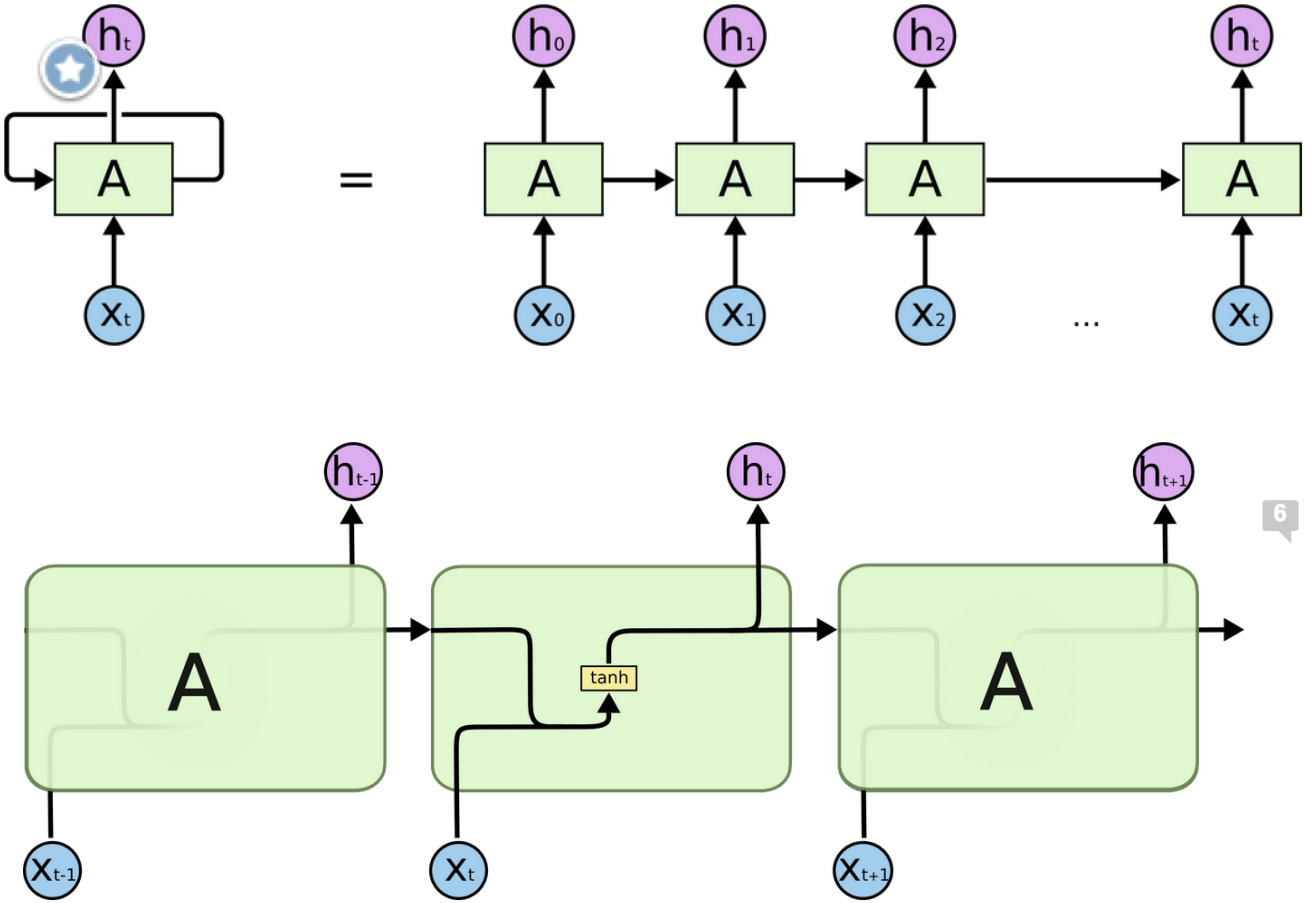




Note: Des vecteurs pour le français (produits avec DEPGLOVE)
`http://alpage.inria.fr/depglove/process`

Réseaux récurrents

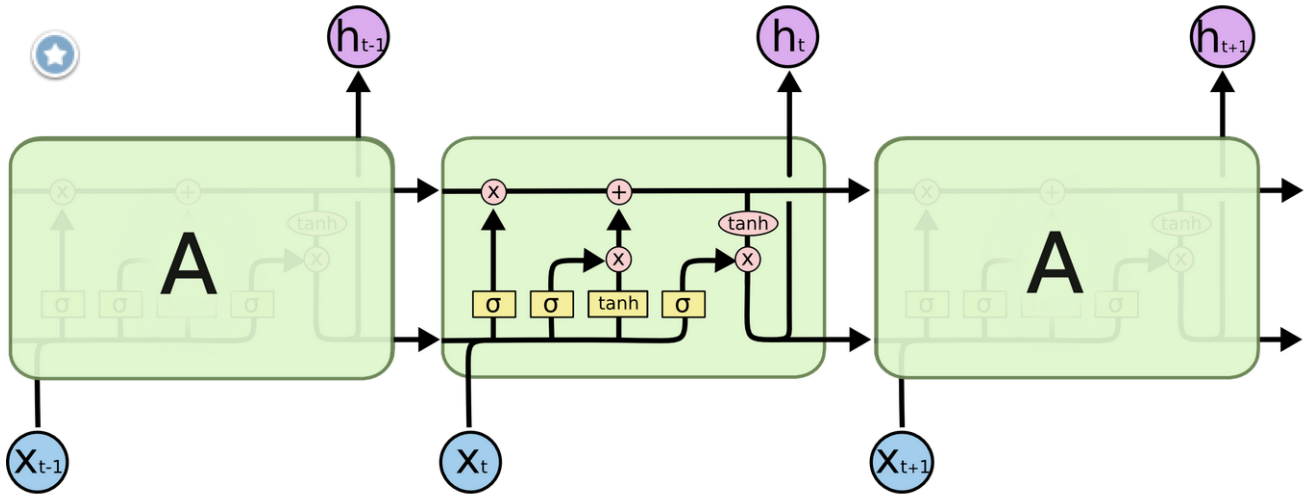
Pour le TAL, on veut des structures de réseaux adaptées au traitement de séquences de mots $\Longrightarrow$ Réseaux récurrents
la sortie de la config à i est utilisée comme entrée pour calculer $i + 1$



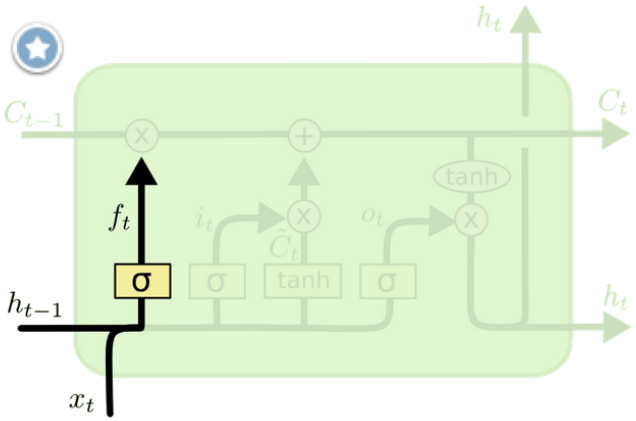
(crédit illustration: `http://colah.github.io/posts/2015-08-Understanding-LSTMs`)

LSTM

Encore mieux: des réseaux récurrents avec mémoire à long terme (et oubli)
Long-Short Term Memory – LSTM
État de l'art actuellement en TAL (bi-LSTM) : gauche vers droite et droite vers gauche

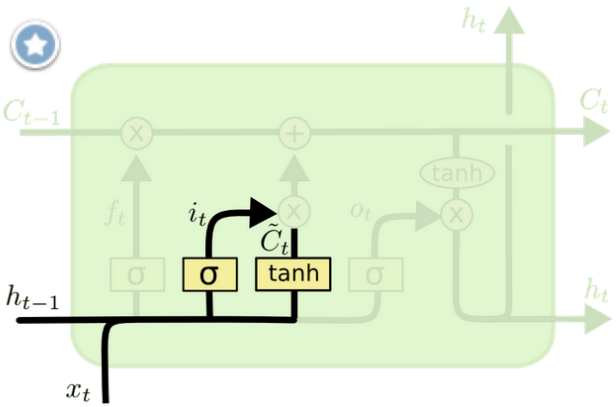


LSTM: oublier des info



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

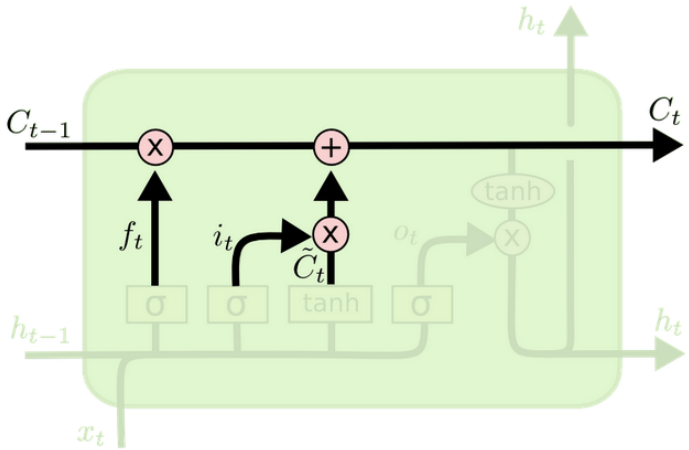
LSTM: ajouter des info



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

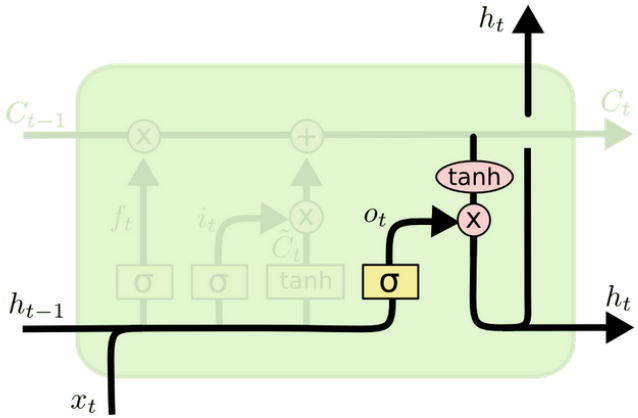
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM: mettre à jour la mémoire



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM: avancer

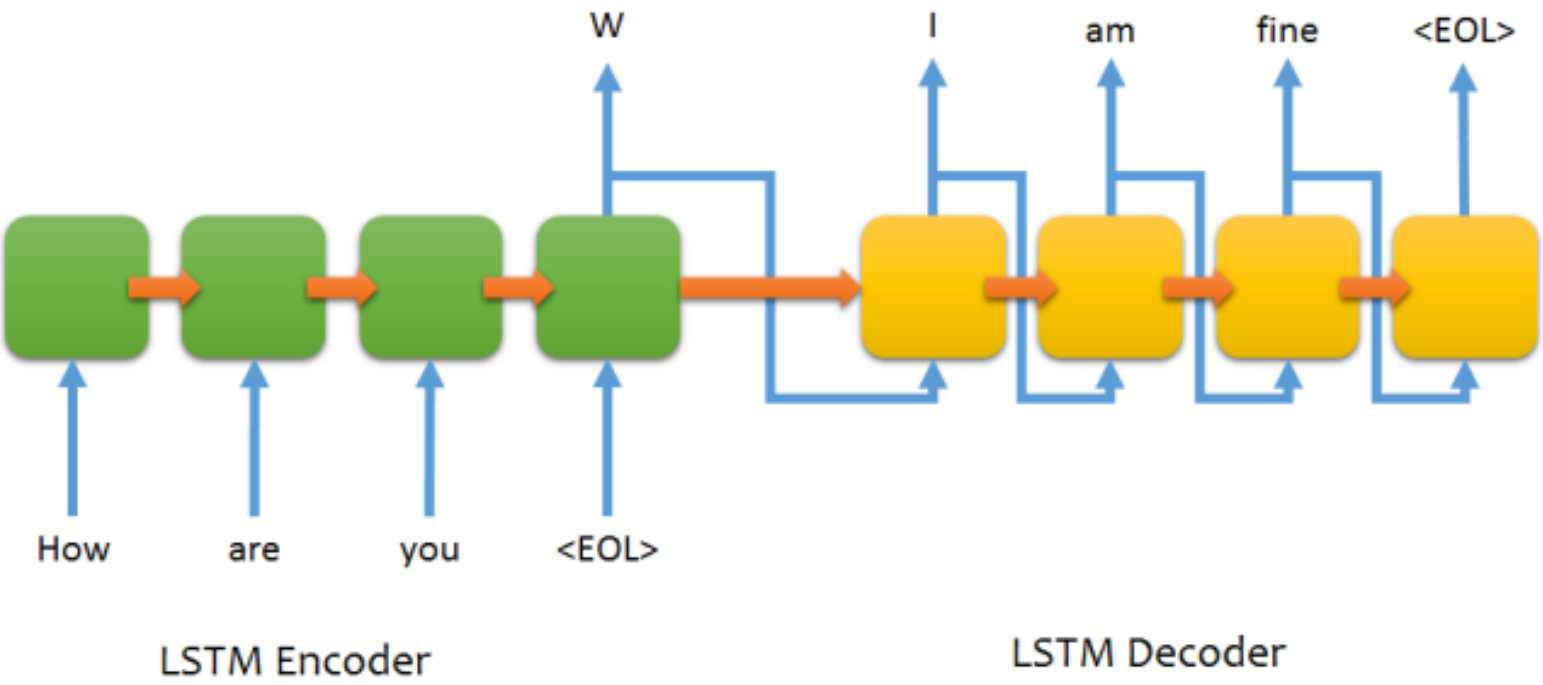


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Modèle séquence à séquence

On peut combiner des LSTM de manière à lire une séquence pour écrire une nouvelle séquence (**traduction**)

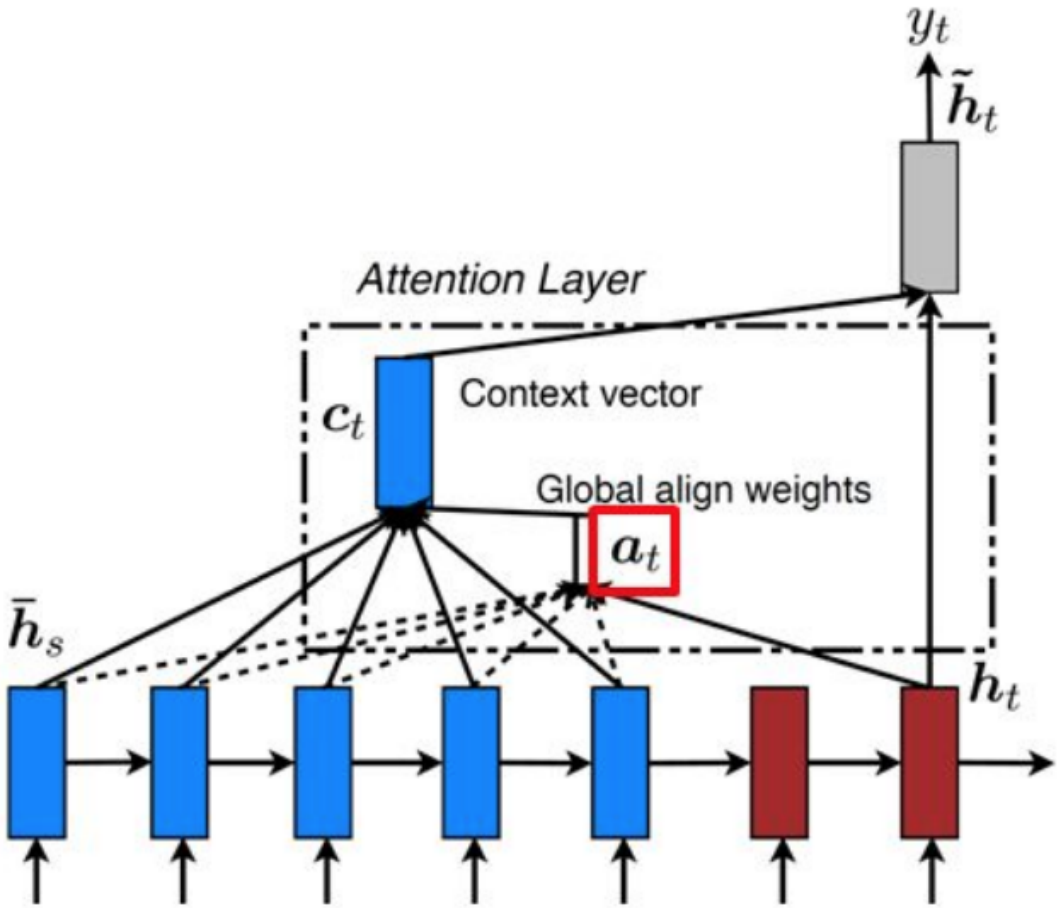


Bons résultats mais pas excellents !

- meilleurs en lisant la phrase à traduire de droite à gauche
- et surtout avec un méchanisme d'attention

Méchanisme d'attention

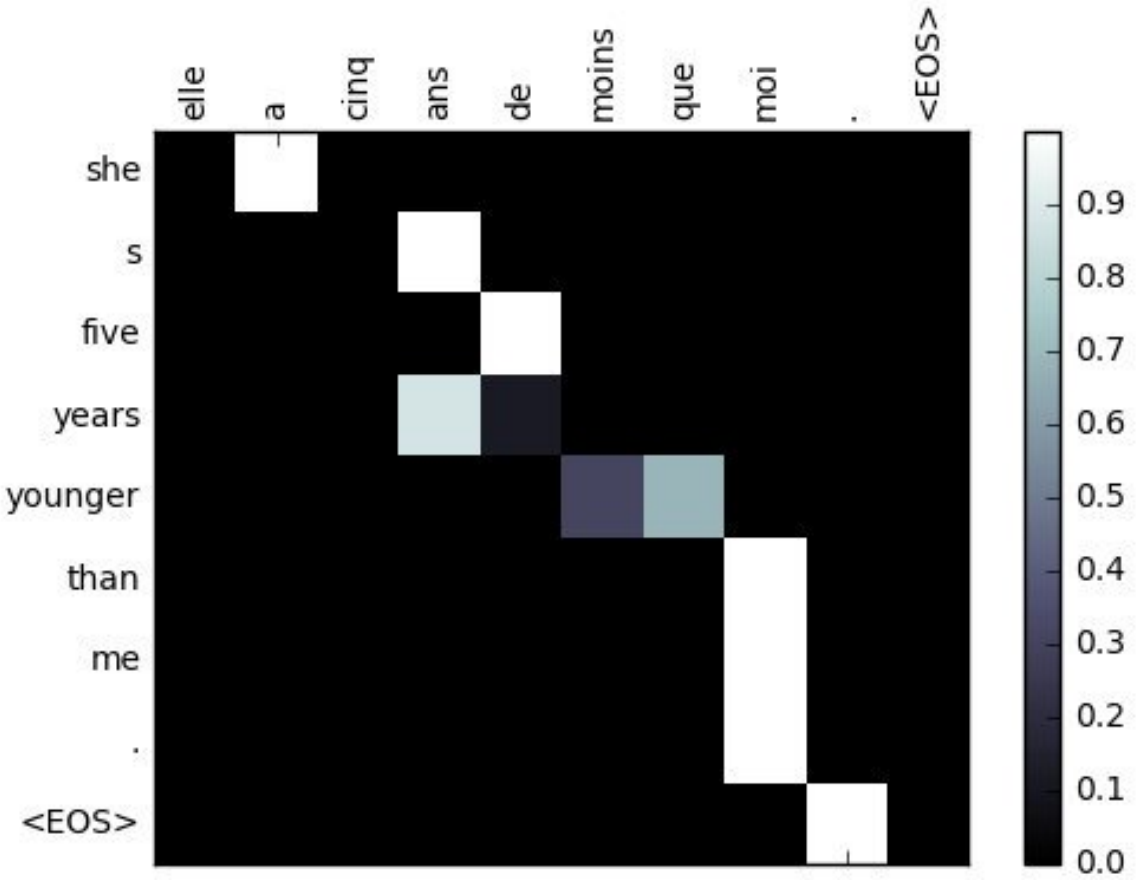
Focaliser l'attention sur les mots de la séquence d'entrée pouvant servir à produire le prochain mot



L'idée d'attention est de plus en plus utilisée

Interprétation

De plus, l'attention ouvre des perspectives d'interprétation



Pour aller plus loin !?

Des pistes en apprentissage profond, pas ou peu utilisées en TAL

AlphaGo

StarCraft

Succès récents et non anticipés pour les jeux

- approches neuronales adversairiales et par ré-enforcement
- découverte de stratégies innovantes par les IA
- éventuellement sans données

Mais, cadres très contraints,
même si énormes espaces de recherche
$\rightsquigarrow$ possibilité de jouer IA vs IA et de calculer les gains

Difficilement transposable dans les interactions humaines

Pour conclure

- L'approche Apprentissage Profond état de l'art en TAL (comme en vision, signal, . . .)
  mais encore en compétition avec d'autres approches (traits)
- besoin de beaucoup de données et de temps d'apprentissage
- pb sur la robustesse, la généralisation, la capacité à apprendre à apprendre
- $\rightsquigarrow$ IA neuronales encore très spécialisées !
  IA surhumaine peu probable à court terme

Plusieurs toolkits disponibles pour démarrer et s'amuser !

- page wikipedia: `https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software`
- TensorFlow (Google), Theano, Keras, PyTorch, Dynet, Gensym, Caffe
- API Python masquant les détails, essentiellement utilisation de briques pour agencer son réseau (mais beaucoup de paramètres possibles)