Introduction au traitement automatique des langues

RCP 217

Serge Rosmorduc
serge.rosmorduc@lecnam.net
Conservatoire National des Arts et Métiers
Cédric, équipe Vertigo

2020–2021

Introduction

- le traitement automatique des langues s'intéresse à l'analyse par un ordinateur des textes, essentiellement des textes écrits.
- plusieurs appellations
  - TAL
  - Natural Language Processing (NLP)
  - Computational Linguistics
  - Linguistiques computationnelles
  - ... qui correspondent à différentes approches

La linguistique

- Science qui étudie les langues
- Réflexion sur la langue très ancienne
  - tablettes de grammaires du sumérien !
  - grammaire du Sanskrit (Pāṇini)
  - grammairiens grecs, latins, arabes...
  - philologie
- naissance de la linguistique "moderne" fin 18e

Le structuralisme et ses descendants

- Saussure
  - langage : capacité générale de l'être humain
  - langue : système utilisé localement pour communiquer
  - parole : mise en ?uvre de ce système par des individus particuliers
  - langue vue comme un système d'oppositions : la langue est un système où tout se tient
- Chomsky
  - Linguistique générative
  - propose un modèle formel de la langue
  - en opposition à l'approche behaviouriste et statisticienne
  - important pour les grammaires formelles
- D'autres approches existent
  - socio-linguistique
  - linguistique comparative
  - linguistique quantitative

Grammaire Prescriptive et grammaire descriptive

- Grammaire prescriptive : dit l'usage "correct" de la langue. C'est la grammaire de l'Académie française et de l'école ;
- Grammaire descriptive : c'est la grammaire des linguistes. Décrit la langue telle qu'elle est réellement pratiquée

Le TAL, un domaine multiforme

Le TAL regroupe plusieurs communautés de chercheurs

- Linguistique formelle (souvent "linguistique computationelle")
- lexicométrie
- édition électronique
- Ingéniérie de la langue : visée pratique, la représentation de la langue n'étant pas un but mais éventuellement un moyen.

Histoire du TAL

- Naissance : fin des années 50
- But originel : traduction automatique
- années 60 : premiers succès, puis premières déceptions
- années 80 : IA formelle, systèmes experts, etc. développement important du TAL
- mais ça ne passe pas à l'échelle
- années 90-2010 : approche plus pragmatique ; utilisation de modèles statistiques de plus en plus perfectionnés
- 2010- : approche connexionniste

tâches du TAL

- Annotation
  - étiquetage
  - Reconnaissance d'entités nommées
- Analyse syntaxique
  - Arborescentes (grammaires de constituants)
  - Structure de graphe (grammaires de dépendances)
- Classification de textes
  - anti-spam
  - Sentiment Analysis
- Réécriture
  - Traduction automatique
  - Résumé
  - correction automatique
- Agent Conversationnel
- Auxiliaire d'autres procédés
  - OCR
  - Reconnaissance de la parole

Difficultés rencontrées

- La langue est ambiguë et mouvante (C. Hagège : *le locuteur est un démiurge*) ;
- homonymes, homographes, homophones ;
- problèmes d'échantillonnage ; mots inconnus/nouveaux/nom propres...
- ambiguïté de la grammaire
  - *The Duke yet lives that Henry shall depose (Shakespeare, Henry VI)*
    - Le duc que Henry renversera vit encore ;
    - Le duc qui renversera Henry vit encore.
  - mais plus simplement : *il mange son riz avec des baguettes/il mange son riz avec des champignons*.
- plus on essaie d'être « couvrant », plus on introduit d'ambiguïté. Ex :
  - « la » n'est pas ambigu dans une petite base de règle : article défini ;
  - en étendant la base, on ajoute le nom de la note de musique « la » !

Sondage !

Dans l'expression « le chat mange la souris », combien de mot sont ambigus (relativement à leur partie du discours : nom, verbe, article)

1. 1
2. 2
3. 3

Sondage

Et au niveau du *sens*, les noms « chat » et « souris » ont :

1. un sens bien définit chacun ;
2. un seul sens pour chat, deux pour souris ;
3. ça dépend du dictionnaire !

Loi de Zipf

- Si on ordonne les mots par fréquence décroissante, le mot de rang k a une fréquence proportionnelle à 1/k.
- c'est assez approximatif dans les faits
- reliée par Benoit Mandelbrot à la théorie de Shannon
- l'important : il y a un petit groupe de mots très fréquents et un groupe très étendu de mots très rares
- donc, quand on a un dictionnaire, on est sûr de tomber sur des mots qui n'y sont pas.

Exemple : mots dans *Notre Dame de Paris*

rang	mot	occurrences	rang * occurrences
1	de	8158	8158
2	la	5380	10760
26	pas	1116	29016
43	comme	708	30444
62	roi	297	18414
72	quasimodo	246	17712

Statistiques et séquences de mots

On utilise souvent les **n-grams** : des décomptes des séquences de mots dans un corpus.

- bi-gram : une séquence de deux mots ;
- tri-gram : séquence de trois mots ;
- etc...
- `https://books.google.com/ngrams`

Problème pour les statistique :

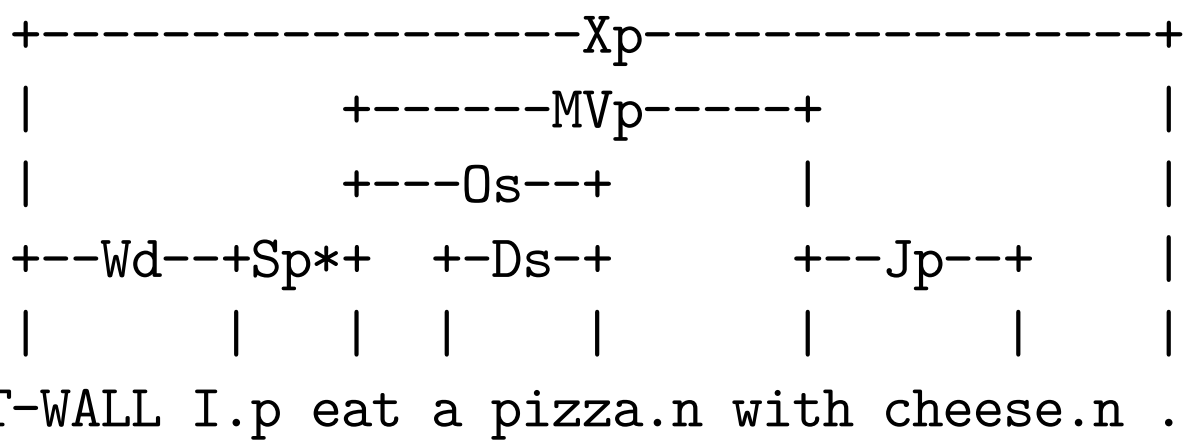
- pour deux mots n1 et n2, on a facilement compte(n1,n2) $= 0$;
- nécessiter de lisser les données (smoothing)

Formalismes grammaticaux

- Grammaires de constituants
  - Décrivent les groupes de mots
  - Grammaires hors-contexte (Context Free)
  - Lexical-Functional Grammar
  - Head-driven phrase structure grammar
  - Tree-adjoining grammar
- Grammaires de dépendances
  - Décrivent les relations entre les mots
  - Link grammar
- Demandent beaucoup d'expertise pour être écrites
- généralement plusieurs analyses possibles
- dans les années 80, construction de modèles sémantiques : fonctionnent seulement avec des micro-mondes
- depuis : modèles statistiques

Un exemple : les Link-grammar

Dans une *link-grammar*, pour chaque mot, on décrit les relations qu'il peut entretenir avec d'autres mots.
L'analyse est alors un graphe :



Lien : relation entre mots. Par exemple : Sp = sujet 1ere personne.

Forme des link grammar

Pour chaque mot, on décrit les relations qu'il peut entretenir à droite (+) et à gauche (-) :

```
the  :D+ ;
nice :A+ ;
boat :D- & A- ;
```

Permet l'analyse :

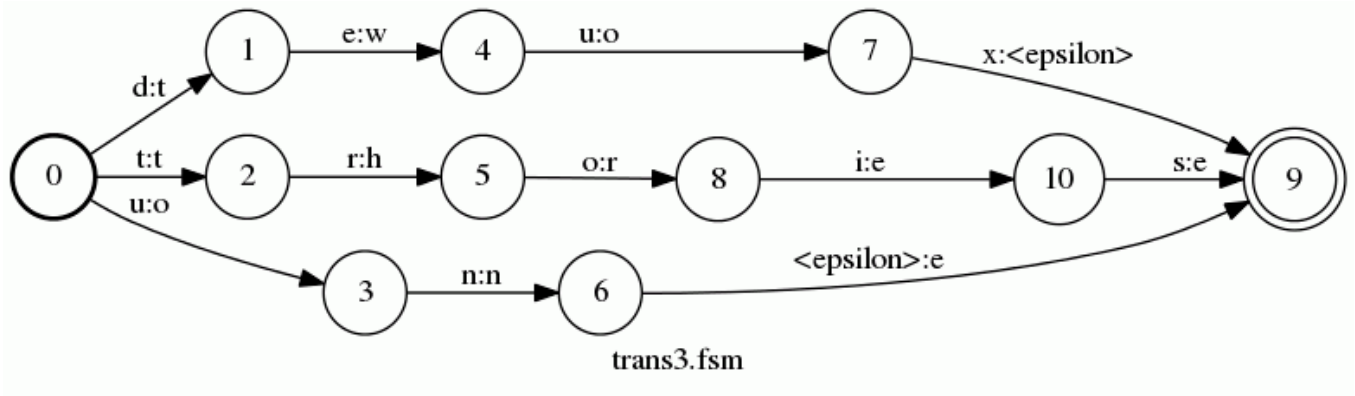
```
 +-----D------+
 |      +---A--+
 |      |      |
the nice   boat
```

Automates finis

- Grammaire de Chomsky de type 3
- équivalentes aux expressions régulières
- pouvoir expressif limité
- assez facile à utiliser pour décrire des fragments de la langue : expressions temporelles, etc...
- analysables en O(n)

Transducteurs finis

- Un transducteur, c'est en gros un automate qui réécrit
- algorithmes de traitement efficaces.



Un transducteur qui traduit *un, deux, trois* en anglais. *d'après F. Barthélémy, http ://deptinfo.cnam.fr/ barthe/NFP108/tp-transducteurs/*

Bibliographie : Roches et Schabes éditeurs, *Finite-State Language Processing* (1997)

Conclusion partielle

- L'approche formelle "pure" est possible dans une certaine mesure
- pose des problèmes d'ingéniérie de la connaissance
- gère mal les irrégularité de la langue
- gère mal les problèmes de sémantique
- mais il est possible de probabiliser certaines de ses approches

Approches probabilistes

Un exemple : les réseaux de Markov cachés.

L'annotation des parties du discours

Étant donné le texte *il la porte* annoter chaque mot avec sa partie du discours (Nom, Verbe, Déterminant...)

il		la		porte
PRO	:PER	PRO	:PER	VER

Problème : l'ambiguïté

- la : DET :ART ou PRO :PER ou NOM (note de musique) ;
- porte : NOM ou VER

Tâche généralement assez facile : en prenant l'annotation la plus fréquente pour un mot donné, 90% de succès (pas ici).
Utilité : pré-traitement pour l'indexation, pour l'analyse syntaxique, création de corpus annotés.

Chaînes de Markov

- Modélisation probabiliste d'une séquence d'événements ;
- On veut estimer $P = P(X_1 = v_1, X_2 = v_2, \ldots X_n = v_n)$;
- Si on considère simplement une séquence A,B,C,D par définition des probabilités conditionnelles, on a :

$$P(A, B, C, D) = P(A|B, C, D)P(B, C, D) \quad (1)$$
$$P(A, B, C, D) = P(A|B, C, D)P(B|C, D)P(C, D) \quad (2)$$
$$P(A, B, C, D) = P(A|B, C, D)P(B|C, D)P(C|D)P(D) \quad (3)$$

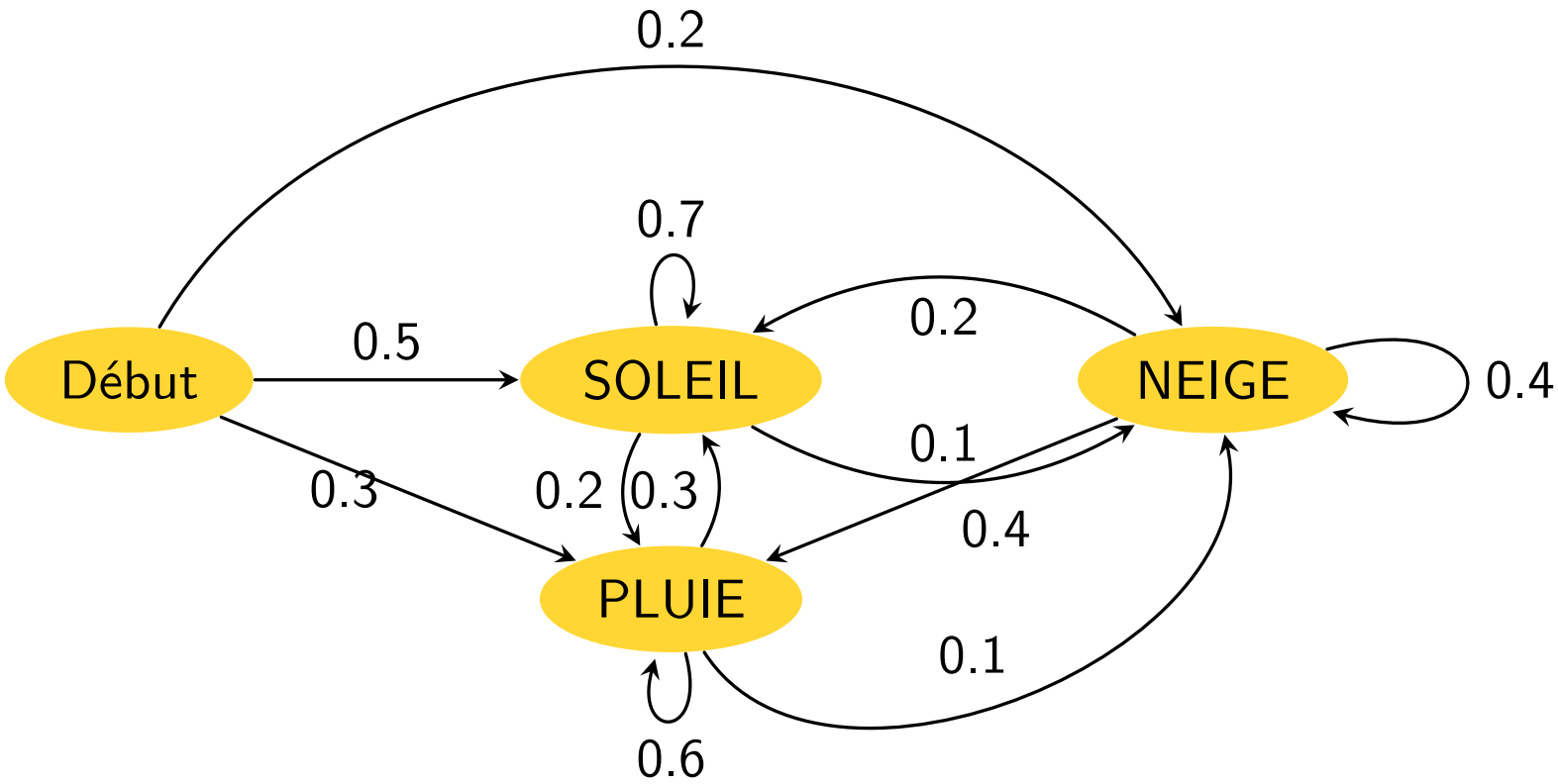
- Si la chaîne vérifie la *propriété de Markov*, la probabilité d'un événement ne dépend que de la valeur de l'événement précédent :

$$P(X_n|X_{n-1}X_{n-2}...X_1) = P(X_n|X_{n-1})$$

(ni des événements antérieurs au précédent, ni de l'indice n)

- on peut donc représenter toutes les possibilités par les couples de transition entre états.

Exemple



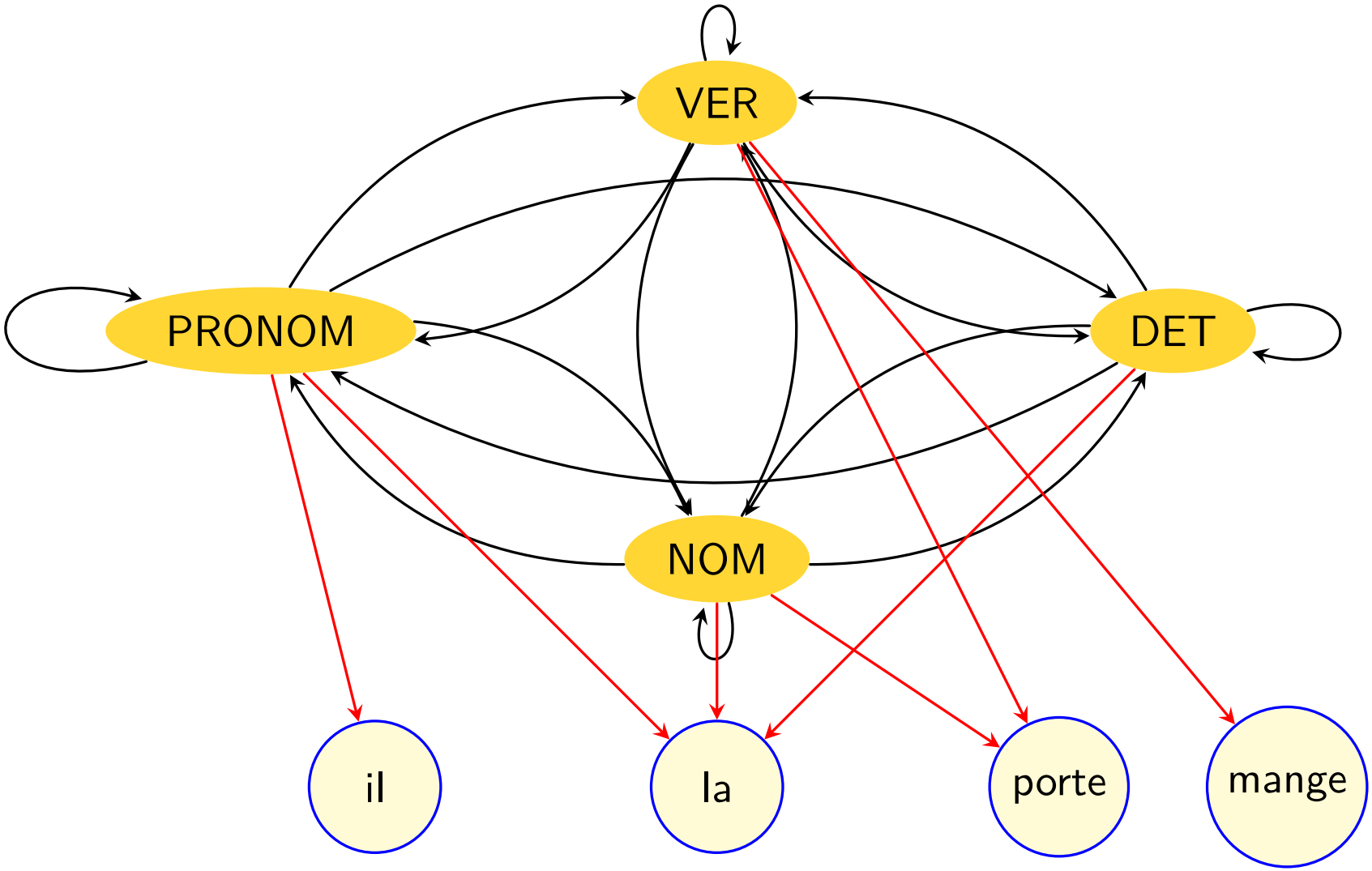
Noter que

$$\forall j, \sum_i P(X^{(i)} | Y^{(j)}) = 1$$

Réseau de markov caché

- on suppose qu'on a une suite d'états générée par un processus de Markov ;
- on n'observe pas directement les états ;
- mais chaque état émet aléatoirement une sortie (qui ne dépend que de l'état) ;
- on observe la séquence des sorties.

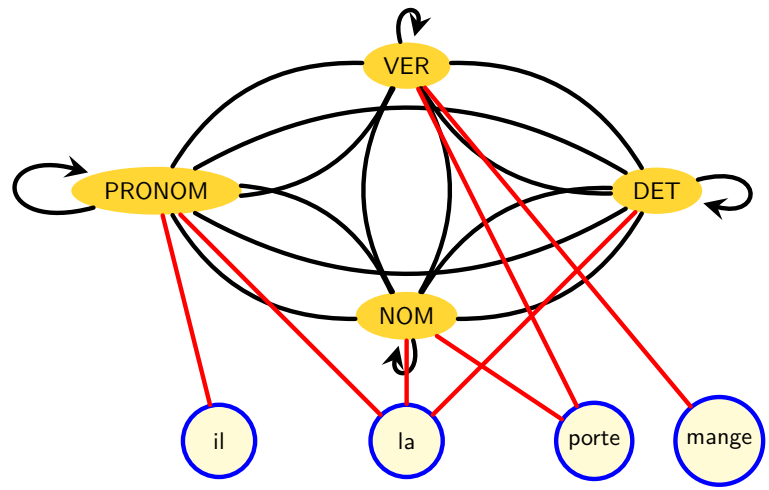
Markov caché



Sondage

Pourquoi ne pas inverser les mots et les étiquettes ? ce serait plus logique ?

1. on a beaucoup plus de séquences de mots que d'étiquettes ;
2. le corpus visible est composé de mots ;



Markov caché

On veut trouver la séquence T_i qui maximise :

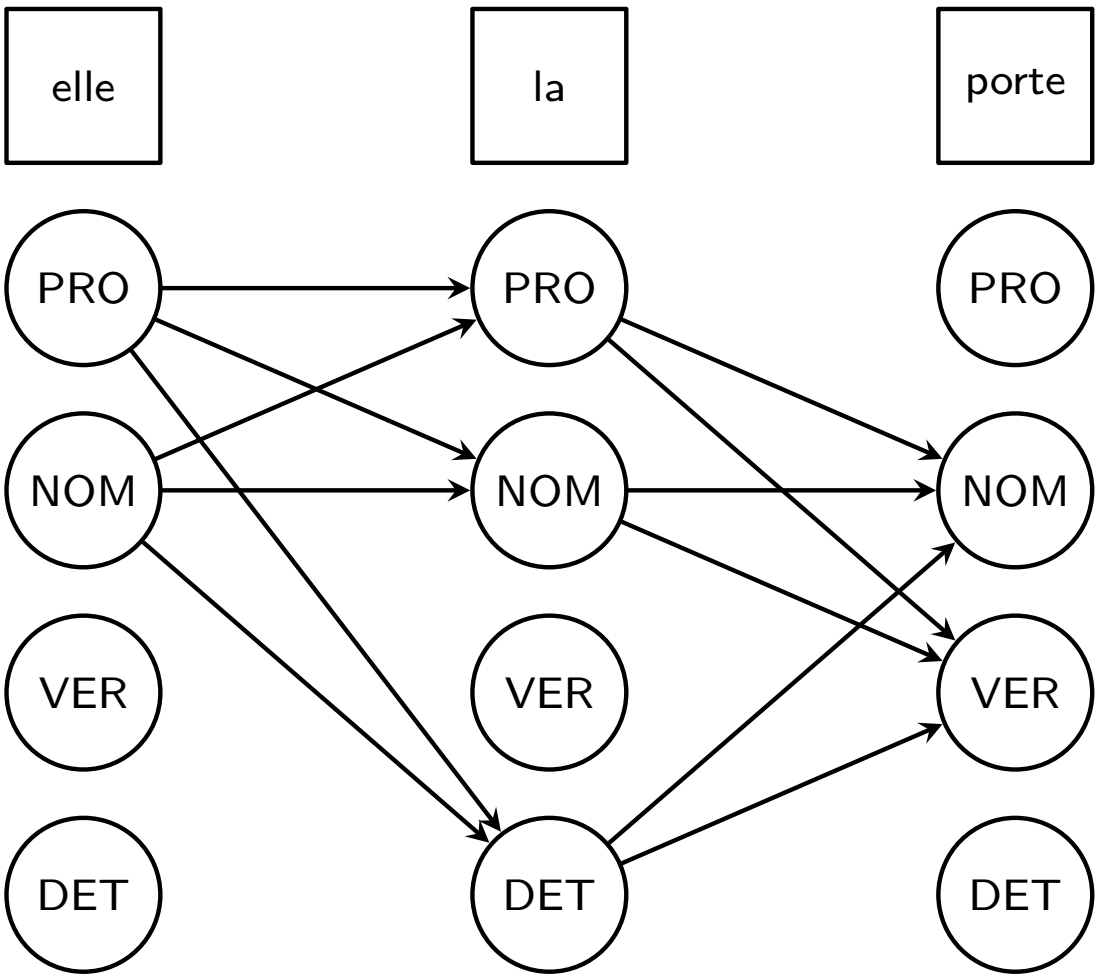
$$P(T_1, T_2, \ldots T_n | X_1, X_2, \ldots X_n) = \frac{P(X_1, X_2, \ldots X_n, T_1, T_2, \ldots T_n)}{P(X_1, X_2, \ldots X_n)}$$

Les X_i sont observés, ça revient donc à maximiser

$$P(X_1, X_2, \ldots X_n, T_1, T_2, \ldots T_n) = \prod_{i=1}^{n} P(T_i | T_{i-1}) P(X_i | T_i)$$

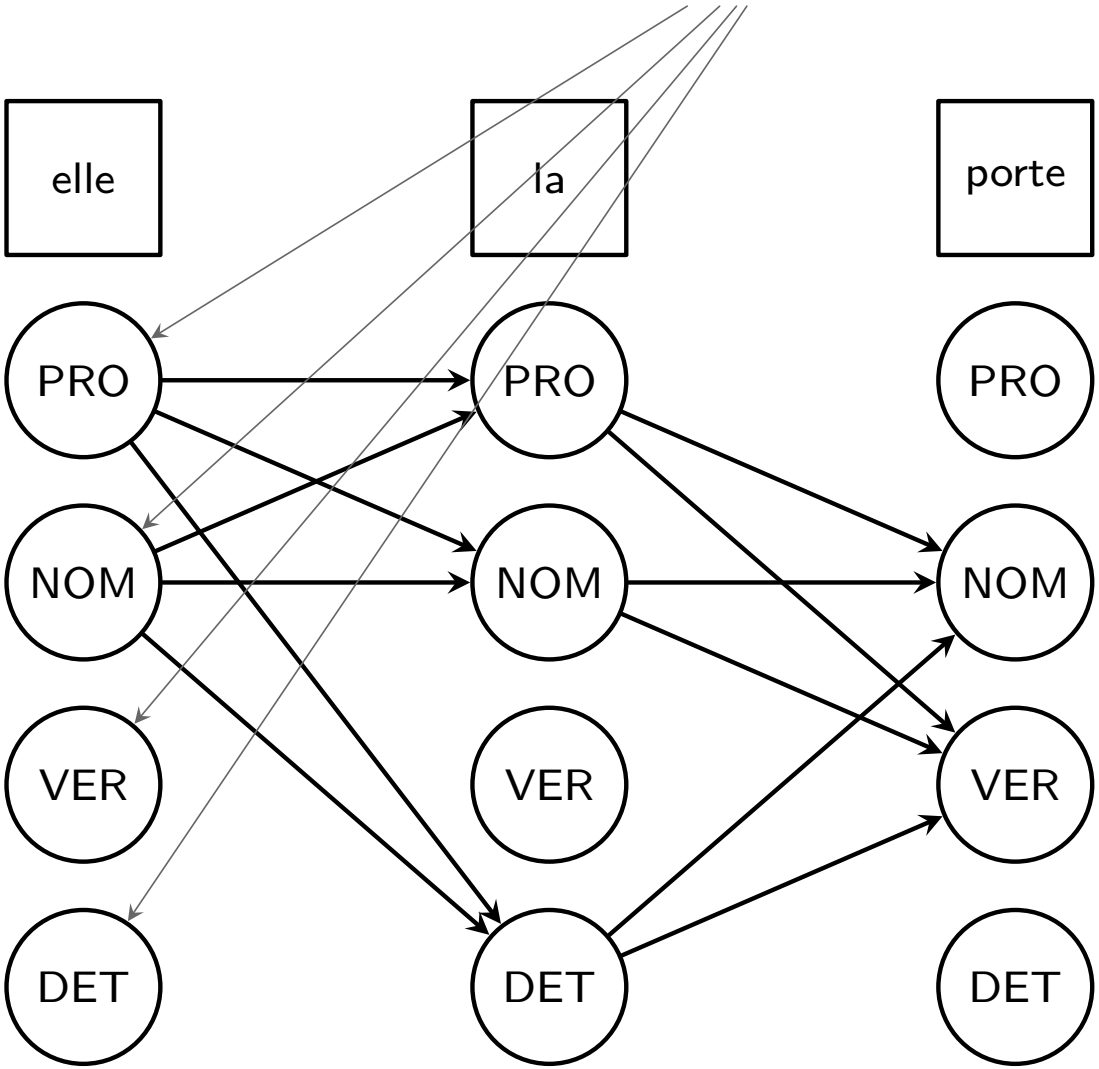
(on introduit un état 0, sans production, pour démarrer)
Les $P(T_i | T_{i-1})$ et les $P(X_i | T_i)$ peuvent s'estimer par simple *comptage* sur un corpus annoté.

Utilisation : algorithme de Viterbi



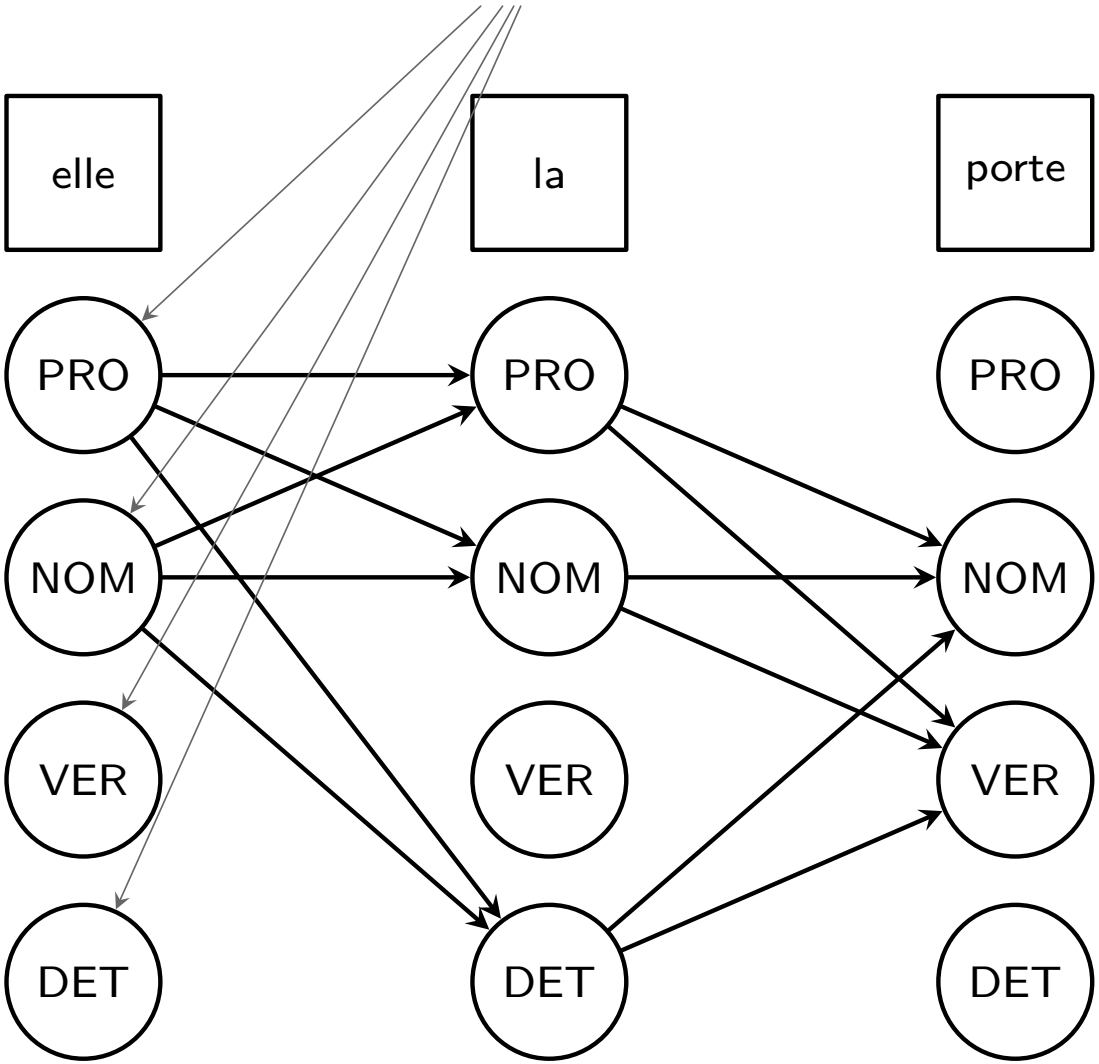
Utilisation : algorithme de Viterbi

on estime pour chaque label sa probabilité : $P(T_1 = t | T_0 = start, X_1 = elle)$



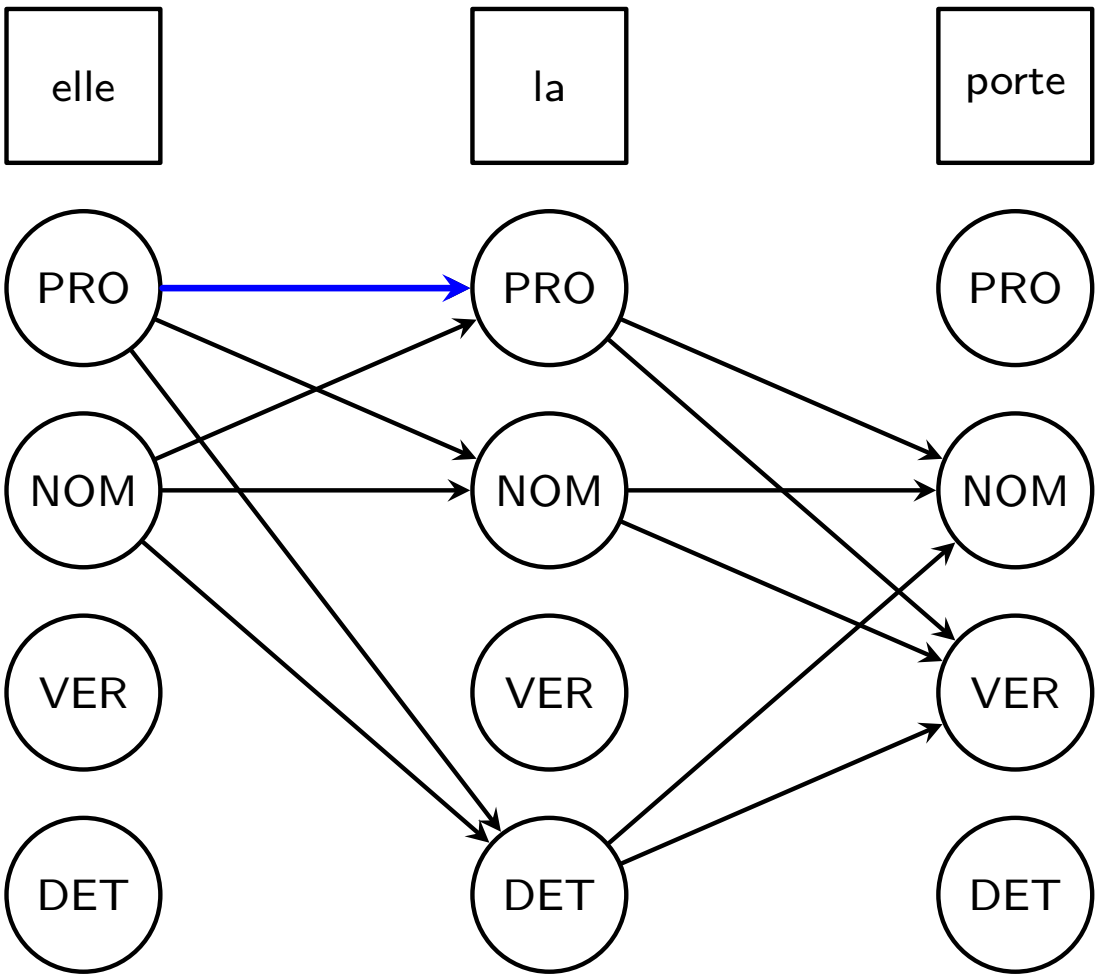
Utilisation : algorithme de Viterbi

$$soit\ P(T = t | T_{prev} = start) P(elle | T = t)$$



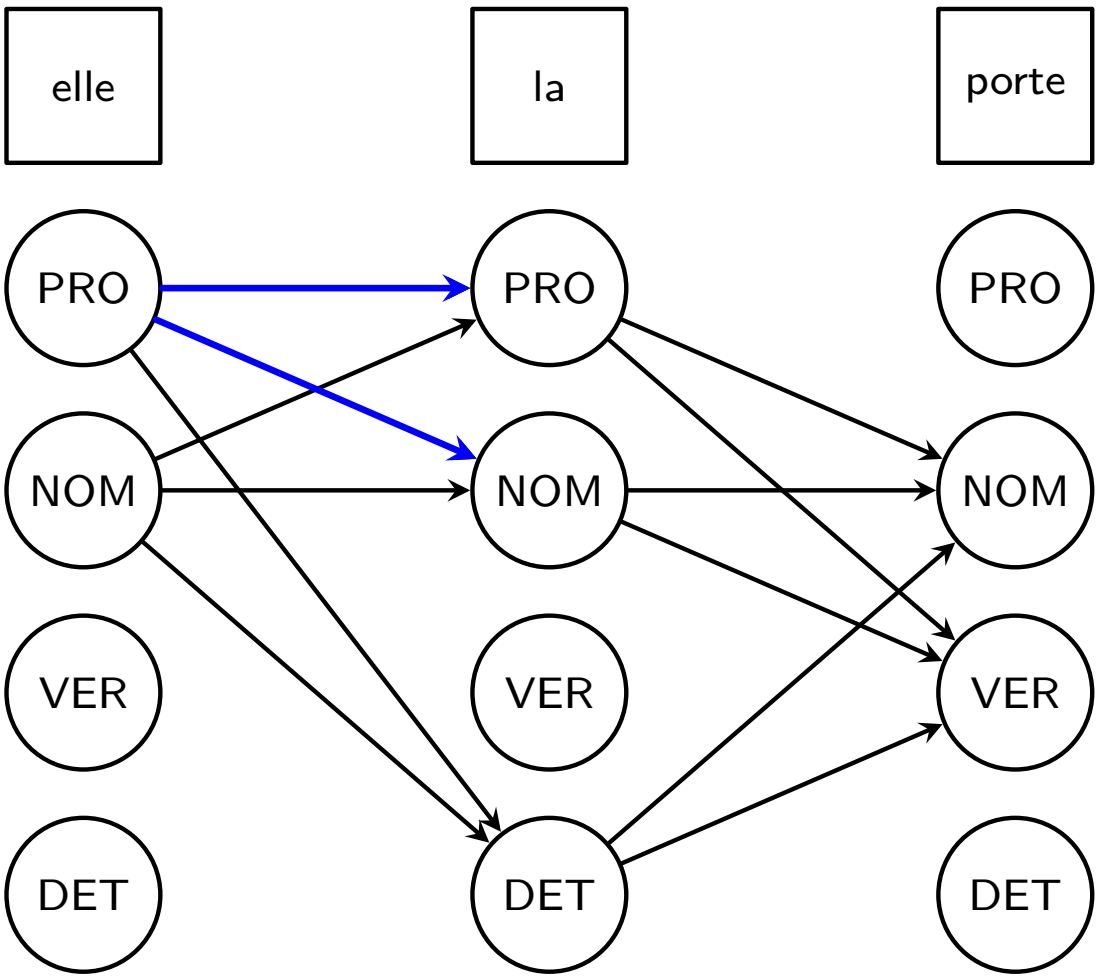
Utilisation : algorithme de Viterbi

Calcul des meilleurs chemins pour « la »



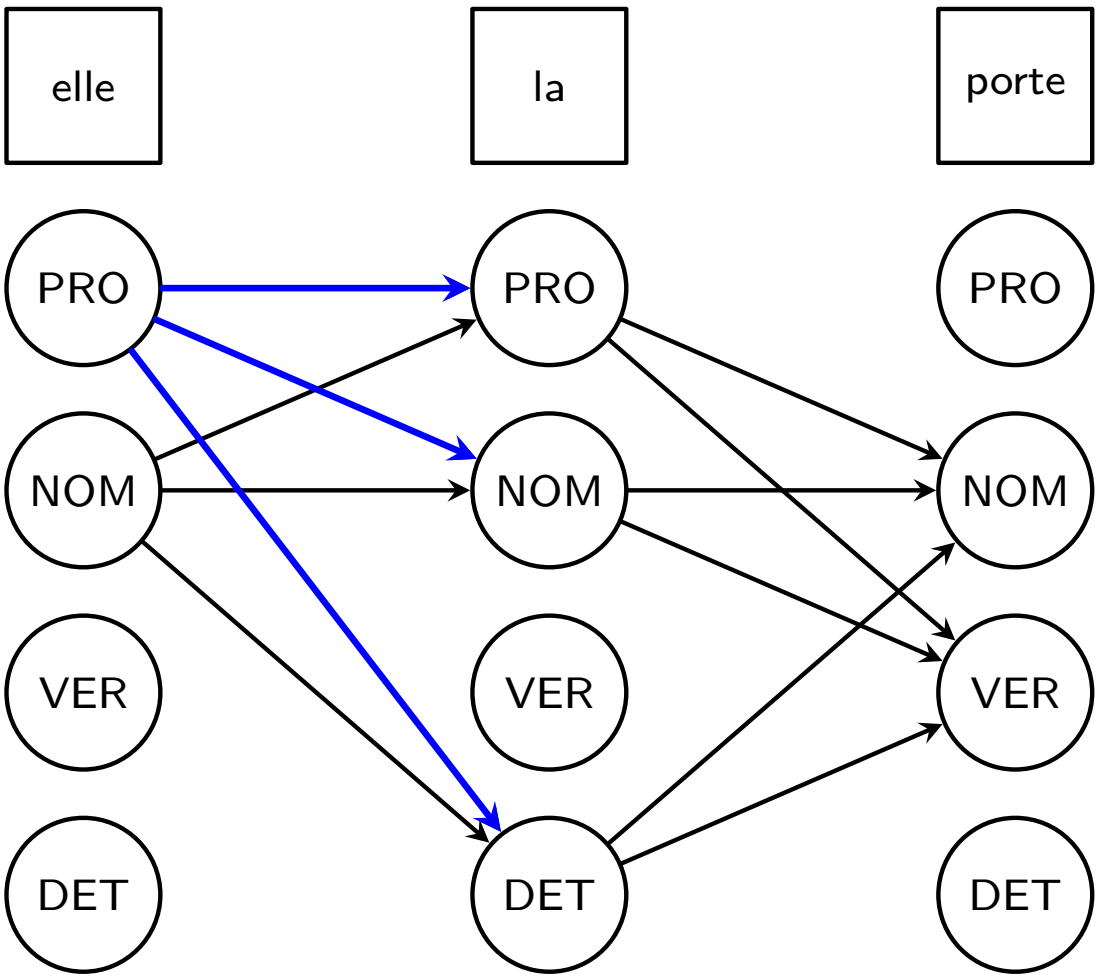
Utilisation : algorithme de Viterbi

Calcul des meilleurs chemins pour « la »



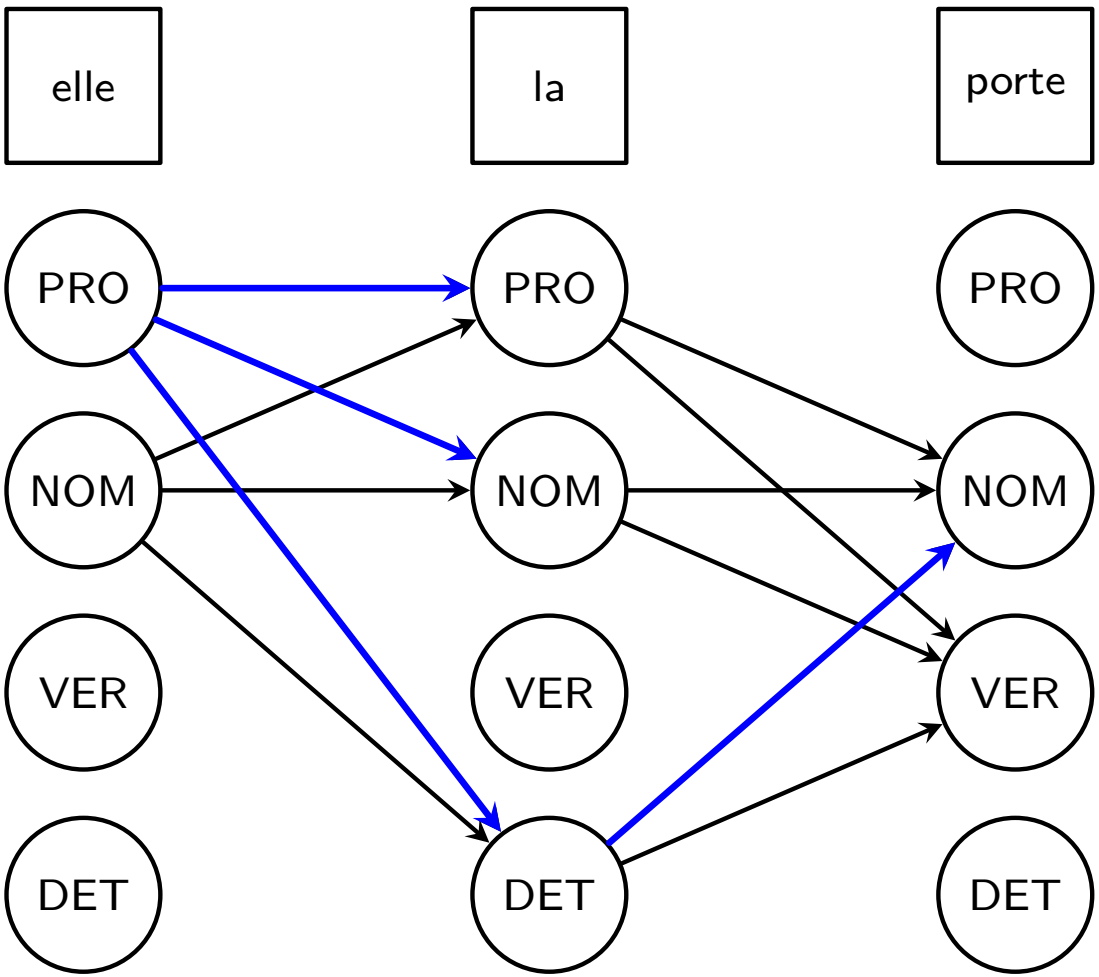
Utilisation : algorithme de Viterbi

Calcul des meilleurs chemins pour « la »



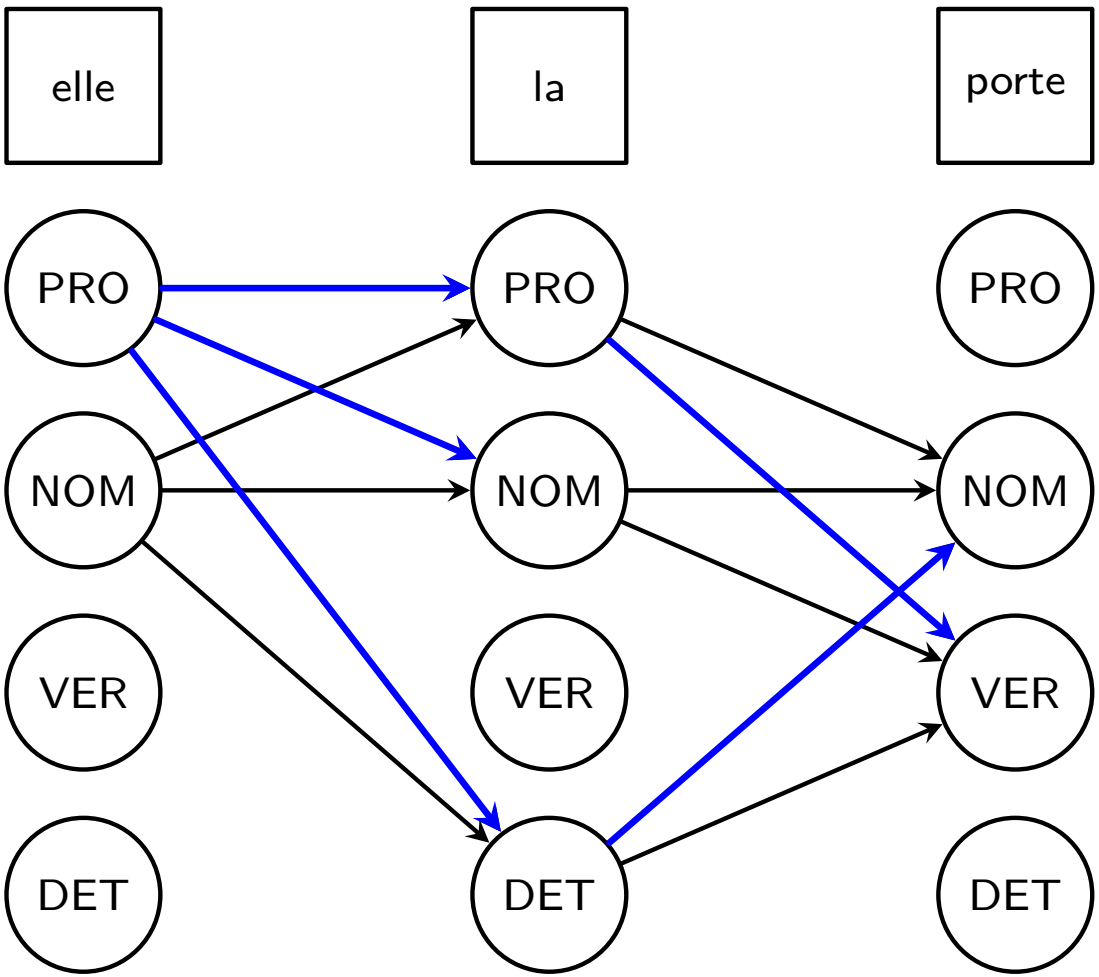
Utilisation : algorithme de Viterbi

Calcul des meilleurs chemins pour « porte »



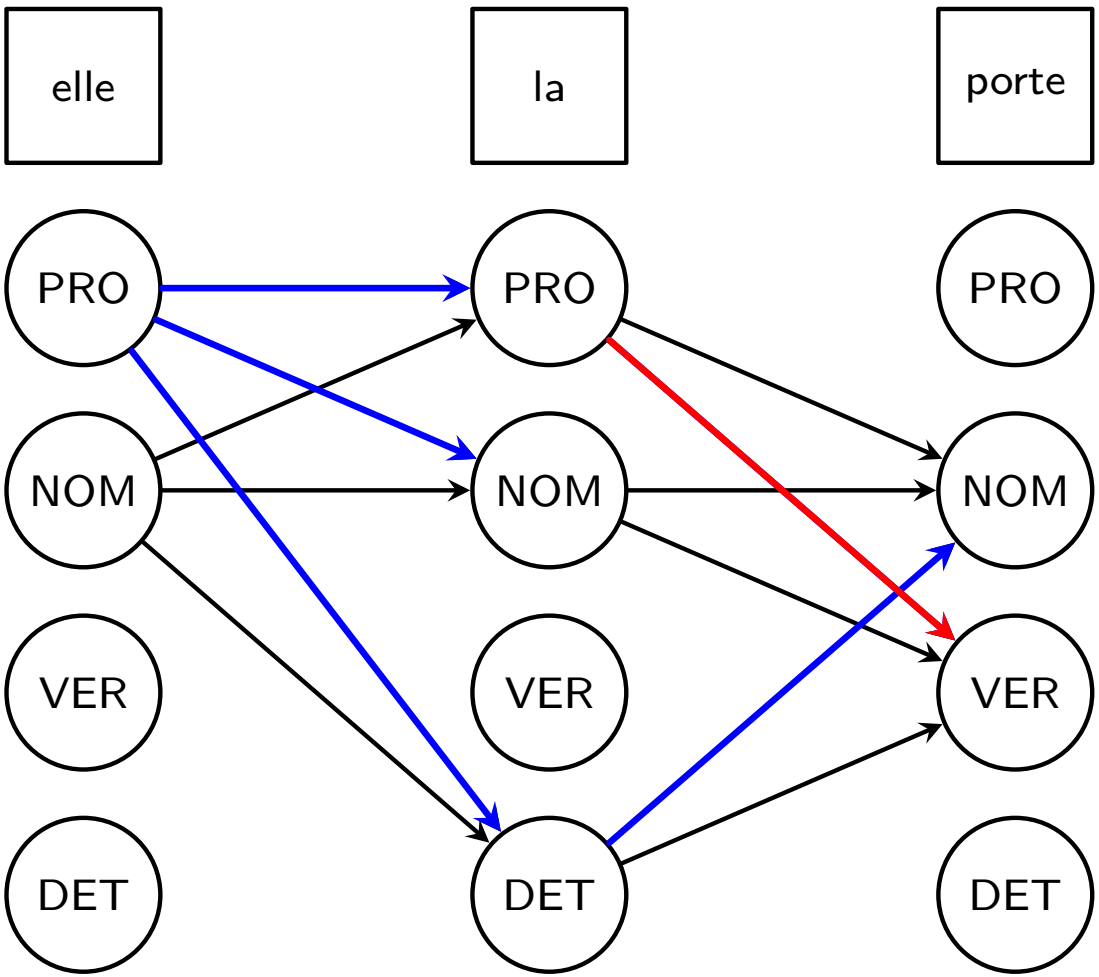
Utilisation : algorithme de Viterbi

Calcul des meilleurs chemins pour « porte »



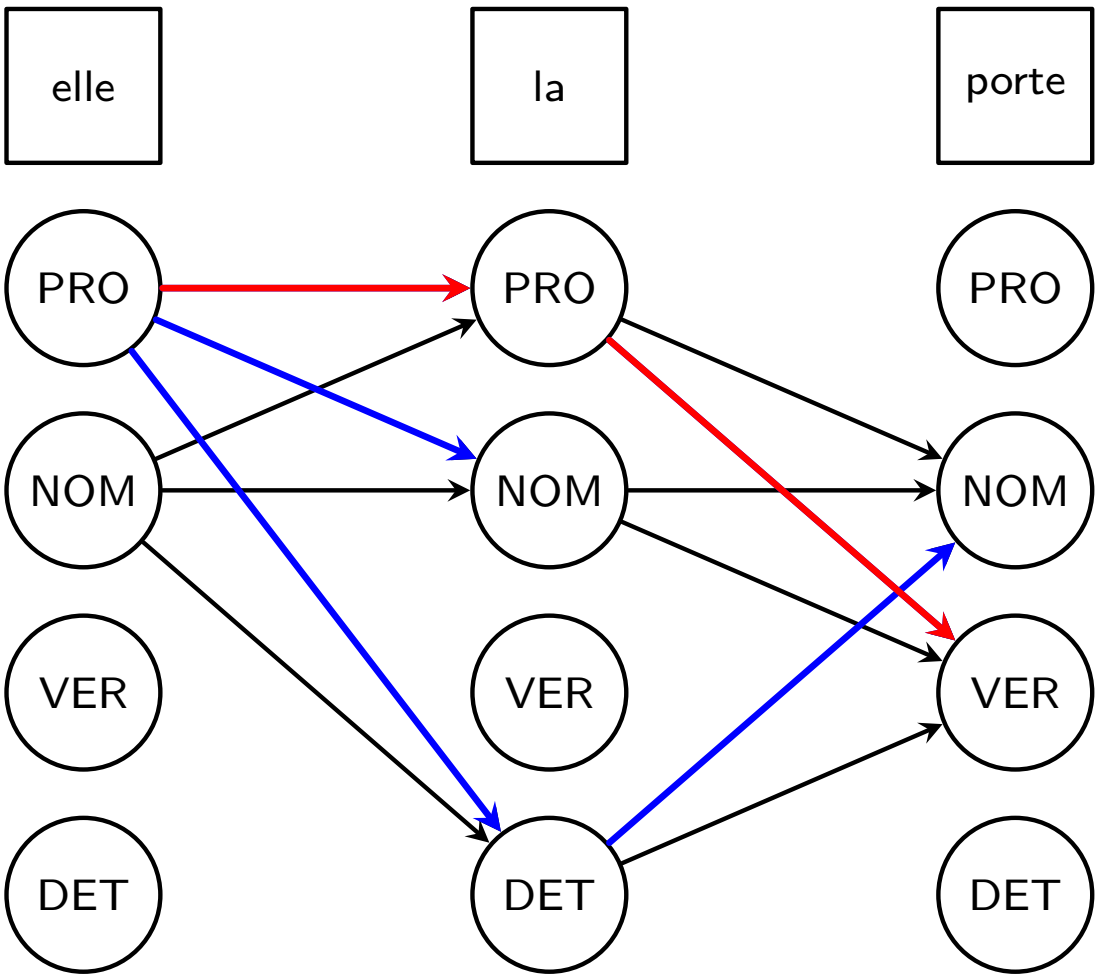
Utilisation : algorithme de Viterbi

On part de l'élément de probabilité max, et on remonte...



Utilisation : algorithme de Viterbi

On part de l'élément de probabilité max, et on remonte...



Algorithme de Viterbi

```
m  :nombre d'annotations possibles ; n  :longueur du texte
TS  :matrice des transition entre états ;
Out  :matrice des transition état-mot
X  :tableau des mots  ;
res  ≔tableau de taille m ; prev  ≔tableau de taille m

pour i  ≔0 à m faire  :
  prev[i].meilleurChemin  ≔[]
  prev[i].score  ≔1 si i = 0 sinon 0.0

pour i  ≔1 à n faire  :
  pour j  ≔1 à  m faire  :
    meilleurPrec = 0
    meilleurScore = 0
    pour k  ≔0 à  m faire  :
      score = prev[k].meilleurScore * TS[k,j] * Out[j,X[j]]
      si score > meilleurScore
        meilleurScore = score
        meilleurPrec = k
    res[j].score = meilleurScore
    res[j].meilleurChemin = prev[meilleurPrec].meilleurChemin + [j]
  prev = res
```

Problèmes pratiques

- on a beaucoup de multiplications : on les remplace par des additions en utilisant le log ;
- surtout : si une séquence d'étiquettes n'est pas observée en apprentissage, sa probabilité est 0 ;
- si on a un mot inconnu, que faire ?
- problème de *lissage* (smoothing) : *good turing estimate*

Représentation des mots

Qu'est-ce qu'un mot

- Une entrée dans le dictionnaire ? (et les mots inconnus ?)
- une suite de lettre ?
  - Sprachverarbeitungsprozesse (Sprach/verarbeitungs/prozesse = traitement des langues)
  - pomme de terre
  - aujourd'hui
  - S.N.C.F.
- sens des mots

WordNet

- Graphe hiérarchisé de mots librement disponible ; très utilisé en TAL « pré-connexionniste » ;
- Une entrée dans WordNet est un "synset" - un ensemble de quasi-synonymes
- Wordnet fournit des relations entre les mots : sorte-de, partie-de, etc.
- Reste assez rigide - comment représenter un "glissement" sémantique ?
- Les mots rentrent-ils toujours dans une hiérarchie ?

La représentation des mots, « one-hot »

Comment représenter un mot dans un réseau de neurones ?

- solution immédiate : on associe à chaque mot un indice entier arbitraire ;
- on préfère un codage binaire :

0	1	2	...	i -1	i	i+1	...	n_V
0	0	0	0	0	1	0	0	0

- fonctionne...
- le mot i et le mot $i+1$ du dictionnaire n'ont a priori aucun rapport entre eux...
- mais la distance entre deux mots est toujours de $\sqrt{2}$... on aimerait que $||\text{chat}, \text{chien}|| < ||\text{chat}, \text{calculer}||$
- représentation creuse « sparse ».

Marqueurs supplémentaires

- Dans le vecteur one-hot, on a des "pseudo" mots
- <UNKNOWN> pour les mots inconnus
- <PADDING> pour le padding
- si le vocabulaire n'est pas tiré du corpus d'entraînement, on aura naturellement des occurrence de UNKNOWN dans l'entraînement ;
- sinon, les simuler, en remplaçant aléatoirement des mots peu fréquents du corpus par UNKNOWN.
- éventuellement, un marqueur de début de phrase et un marqueur de fin de phrase

Sémantique distributionnelle

- *You shall know a word by the company it keeps* (Firth, J. R. 1957 :11) ;
- Le sens d'un mot, ce sont les **contextes** dans lesquels il peut apparaître ;
- notion de *concordance* chez les linguistes et les philologues ;
- idée : trouver une représentation du mot qui soit fonction de ses contextes ;
- premier essais 1992, Brown *et al*, « Class-based n-gram models of natural language », Computational Linguistics, 18(4) :467–479.

Word Embedding

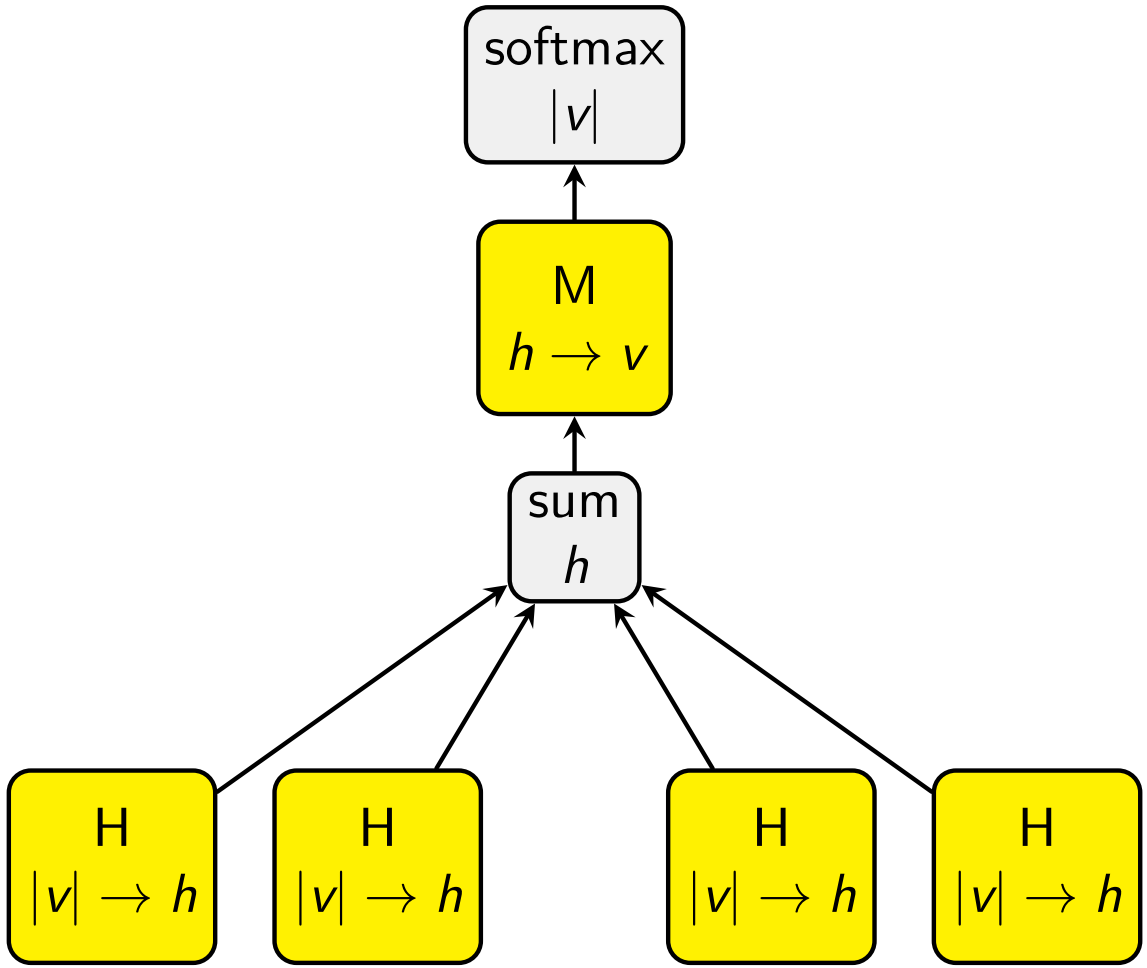
- Représentation vectorielle "pleine" et de petite dimension (50-300) d'un mot ;
- on espère que deux mots sémantiquement proches auront des représentations proches ;
- premières approches (non connexionnistes) : *Latent Semantic Analysis* ;

Word2Vect

- Apprentissage *non supervisé* de représentation vectorielles de mots ;
- Tâche : prédire un mot connaissant son environnement (cbow, « continuous bag of words »)
- algorithme alternatif : prédire un contexte à partir d'un mot (skip-gram)
- cbow fonctionne bien pour les mots fréquents, skip-gram pour les mots rares ;
- on a seulement besoin d'un corpus de textes.
- Structure du réseau : prédit le mot n en connaissant les mots $n-2$, $n-1$, $n+1$, $n+2$.
- les matrices H partagent les mêmes poids.

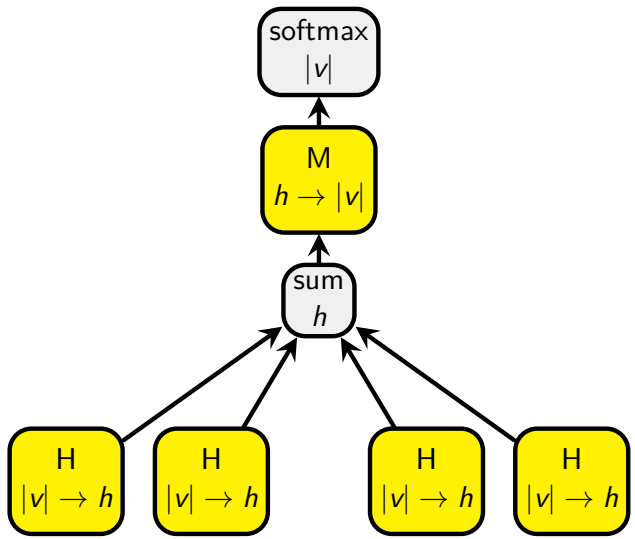
CBOW

Continuous Bag of Words

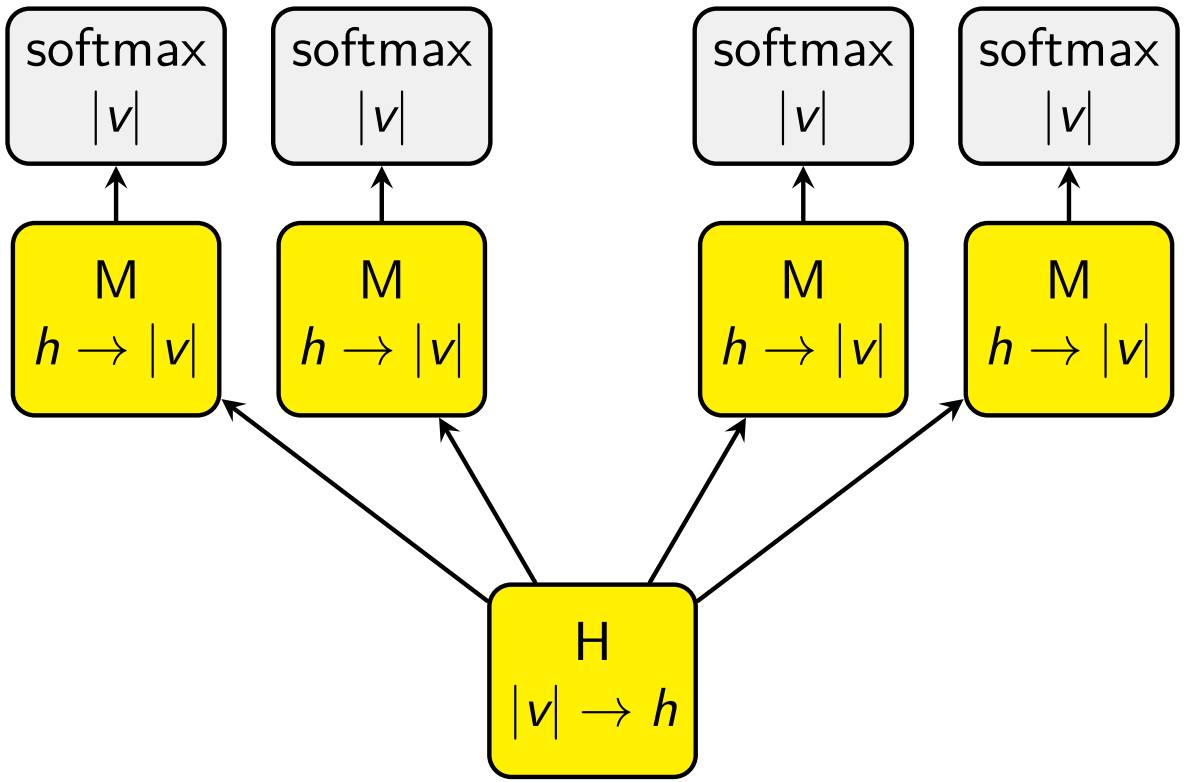


CBOW

- 4 vecteurs en entrées de taille v, taille du vocabulaire ;
- la dimension cachée h sera celle de l'embedding final ;
- l'embedding est $H + M^T$
- pour l'indice i d'un mot du vocabulaire V, l'embedding produit un vecteur de dimension h.

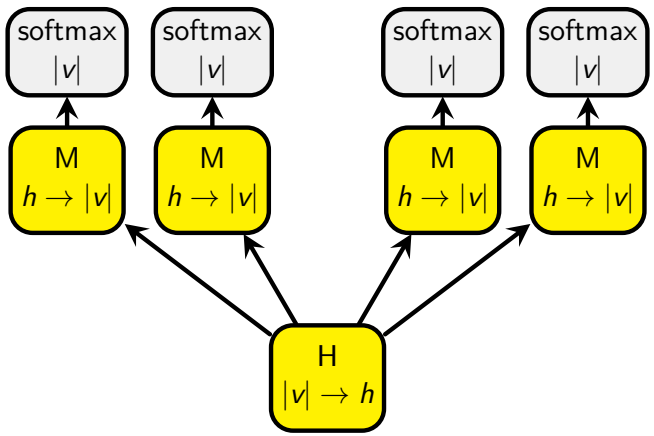


SKIP-GRAM


softmax
$|v|$
softmax
$|v|$
softmax
$|v|$
softmax
$|v|$
M
$h \rightarrow |v|$
M
$h \rightarrow |v|$
M
$h \rightarrow |v|$
M
$h \rightarrow |v|$
H
$|v| \rightarrow h$

Skip-Gram

- à partir du mot, on essaie de prédire $n = 4$ mots
- la fenêtre peut être plus large (d'où **skip**-gram) ;
- on peut considérer cela comme n tâches successives ;
- apprend plus lentement que **CBOW,** mais fonctionne mieux avec mots rares.



« Distance » Cosinus

- soient v_1 et v_2 les vecteurs associés à deux mots ;
- norme euclidienne dépend trop de la fréquence des mots ;
- l'angle entre v_1 et v_2 est un bon critère ;
- en pratique, on utilise le cosinus de cet angle ;
- plus il est proche de 1, plus v_1 et v_2 sont proches ;
- calcul : $\frac{v_1 . v_2}{|v_1||v_2|}$ (autrement dit $\frac{v_1^T v_2}{|v_1||v_2|}$)

Caractéristiques géométriques de l'embedding

Empiriquement, on constate :

- que les mots « proches » au sens du cosinus sont sémantiquement proches ;
- que la somme de plusieurs mots a un sens : « china + river = Yang Tse »
- qu'on peut travailler par analogie : « woman + (king - man) = queen »
- des phénomènes comme le pluriel, les contraires, le féminin, le superlatif sont représentés.

Glove

Global Vectors for Word Representation

Part des *propriétés souhaitées* sur les sommes et les différences de vecteurs ;
Si $P(a|b)$ est la probabilité de rencontrer le mot a dans le contexte du mot b, à votre avis :

1. $P(solide|glace)/P(solide|vapeur)$ est grand,
   $P(gas|glace)/P(gas|vapeur)$ est grand, et
   $P(chat|ice)/P(chat|vapeur)$ est proche de 0 ;
2. $P(solide|glace)/P(solide|vapeur)$ est proche de 1,
   $P(gas|glace)/P(gas|vapeur)$ est proche de 1, et
   $P(chat|ice)/P(chat|vapeur)$ est petit ;
3. $P(solide|glace)/P(solide|vapeur)$ est grand,
   $P(gas|glace)/P(gas|vapeur)$ est petit, et
   $P(chat|ice)/P(chat|vapeur)$ est proche de 1 ;

Glove

Global Vectors for Word Representation

	k = solid	k = gas	k = water	k = fashion
$P(k\|ice)$	$1.9\,10^{-4}$	$6.6\,10^{-5}$	$3\,10^{-3}$	$1.7\,10^{-5}$
$P(k\|steam)$	$2.2\,10^{-5}$	$7.8\,10^{-4}$	$2.2\,10^{-3}$	$1.8\,10^{-5}$
$P(k\|ice)/P(k\|steam)$	8.9	$8.5\,10^{-2}$	1.36	0.96

- idée : ce qui donne le sens des mots, ce sont les *rapports* entre les probabilités conditionnelles ;
- on définit la représentation w_i du mot i en essayant d'avoir une relation raisonnablement simple entre $w_i - w_j$, w_k, et $\frac{P(k|i)}{P(k|j)}$;
- les auteurs proposent *une* solution possible ;
- cette solution est calculée en minimisant une fonction de coût.
- cette solution est *globale* (pas de « fenêtre » comme dans word2vect) : capture mieux les informations **sémantiques** ;

Fasttext

- Similaire à Word2Vect
- Mais pour chaque mot on a dans le vecteur "one"-hot
  - une entrée pour le mot
  - une entrée pour chaque tri-gramme du mot
  - pour le mot *where*, on aura donc : `<wh`, `whe`, `her`, `ere`, `re>`, et `where`.
- (le vecteur n'est donc pas spécialement **one**-hot) ;
- plus robuste pour les mots inconnus ;
- capture aussi des informations de racine ;

Récapitulation sur l'embedding

- cbow fonctionne bien pour les mots fréquents, skip-gram pour les mots rares ;
- Glove fonctionne plutôt mieux que les deux précédents ;
- fasttext est plus robuste face aux mots inconnus ;

Représentation des mots caractère par caractère ?

- Représentation envisageable dans certains cas : le réseau « apprendra » sa propre notion de mot si nécessaire ;
- pas de problème de mots inconnus ;
- robustesse / fautes d'orthographe ;
- allonge l'entrée ; des problèmes liés à des mots proches (dans la phrase) deviennent des problèmes de dépendance à distance ;
- solution intermédiaire : découper le texte en syllabes ou en n-uplets de signes.

Bibliographie

Les livres un peu anciens (avant 2015 ?) ne contiennent presque rien sur le connexionnisme.

- Jurafsky, Daniel, et James H. Martin. *Speech and Language Processing.* 2 édition, 2008. (Draft de la 3e édition, 2020)
- Eisenstein, Jacob. *Introduction to natural language processing. Adaptive computation and machine learning.* Cambridge, Massachusetts : The MIT Press, 2019. preprint
- Bird, Steven, Ewan Klein, et Edward Loper. *Natural Language Processing with Python.* O'Reilly, Inc., 2009. (inclus dans nltk)
- Mikolov, Tomas, Kai Chen, Greg Corrado, et Jeffrey Dean. « Efficient Estimation of Word Representations in Vector Space ». arXiv :1301.3781 [cs], 2013
- Pennington, Jeffrey, Richard Socher, et Christopher D. Manning. « GloVe : Global vectors for word representation ». In *Empirical methods in natural language processing (EMNLP)*, 1532-43, 2014. lien

Webographie

- Atala : site de l'Association pour le Traitement Automatique des Langues
- Computational Linguistics revue internationale dans le domaine ;
- nlpprogress site listant l'état de l'art pour chaque type de problème classique ; donne aussi accès aux corpus de référence ;
- NLP à Stanford : site d'une des équipes les plus reconnues du domaine ;
- Cours de Stanford 2020 archives complètes du cours 2020 (et des précédents)
- CoreNLP : bibliothèque de logiciels de stanford ;
- Hugging Face implémentations de la plupart des architectures classiques ;
- Spacy : une bibliothèque puissante et multilingue, mais peu adaptée à l'expérimentation architecturale ;