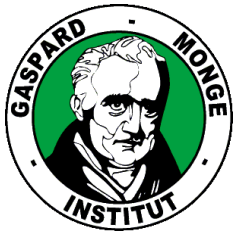
Master 1 Informatique – Université Marne-la-Vallée (IGM)
13/03/2014 – Cours 5
Ingéniérie Linguistique

Étiquetage grammatical et analyse syntaxique



UNIVERSITÉ
PARIS-EST
MARNE-LA-VALLÉE

Philippe Gambette

Sources du cours

- Cours de Matthieu Constant, *Ingéniérie Informatique 1*

http://igm.univ-mlv.fr/ens/Master/M1/2010-2011/IngenierieLinguistique1/cours.php

- Cours d'Anthony Sigogne, *Ingéniérie Informatique 1*, UPEMLV 2011-2012

Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Étiquetage grammatical

But

Associer à une séquence $w = w_1 \ldots w_n$ de mots, une séquence $e = e_1 \ldots e_n$ d'étiquettes appartenant à un jeu d'étiquettes J.

Exemple de sortie

Le/DET chien/NC mange/V ses/DET croquettes/NC ./PONCT

Point de vue probabiliste

Trouver la séquence d'étiquettes $\hat{e}$ qui maximise $P(e|w)$ parmi l'ensemble des séquences d'étiquettes possibles.

$\hat{e} = \text{argmax}_e P(e|w) = \text{argmax}_e P(w|e).P(e)$

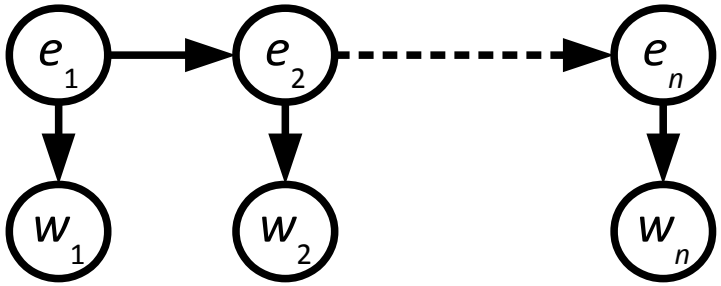
Modèle de Markov caché d'ordre *k*

Principe

Symbole visible : mot

Symbole caché à découvrir : étiquette

Hypothèses d'indépendance



Hypothèses de Markov pour le calcul de P(*w*|*e*).P(*e*)

• P(*w*|*e*) ? un symbole observé (mot) ne dépend que du symbole caché associé (étiquette)

• P(*e*) ? un symbole caché (étiquette) ne dépend que des *k* précédents

Modèle de Markov caché d'ordre *k*

Calcul de P(*w*|*e*)

$P(w|e) = P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$

Calcul de P(*e*) pour *k*=1 (modèle des bigrammes)

$P(e)=P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1})$

Calcul de P(*e*) pour *k*=2 (modèle des trigrammes)

$P(e)=P(e_1).P(e_2|e_1).P(e_3|e_2e_1)...P(e_n|e_{n-1}e_{n-2})$

Estimation des probabilités

Corpus d'apprentissage

Le corpus utilisé pour l'apprentissage des probabilités de base est composé de phrases annotées : chaque token est associé à une catégorie grammaticale.

Exemple

Le/DET chien/NC aboie/V ./PONCT

Le/DET cordon_bleu/NC est/V bon/ADJ ./PONCT

Marc/NPP a/V acheté/VPP un/DET nouveau/ADJ téléphone/NC ./PONCT

...

Estimation des probabilités

Calcul des probabilités d'émission $P(w_i|e_i)$

$$P(w_i|e_i) = \frac{\#occ(w_i,e_i)}{\#occ(e_i)}$$

Calcul des probabilités de transitions $P(e_i|e_{i-1})$

$$P(e_i|e_{i-1}) = \frac{\#occ(e_{i-1}\,e_i)}{\#occ(e_{i-1})}$$

Exemple (n = 2)

On suppose que J={X, Y} et que notre vocabulaire est {a, b, c}.

Probabilités des étiquettes P(e_i)

P(X)=6/10 P(Y)=4/10

Probabilités d'émission

	X	Y
a	1/10	6/10
b	4/10	3/10
c	5/10	1/10

Probabilités de transition

	X	Y
X	7/10	4/10
Y	3/10	6/10

Quelle est la séquence d'étiquettes du mot *w*=acba ?

Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général: grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Seconde solution (optimisation)

Application de l'algorithme de programmation dynamique : Viterbi.

=> temps de calcul réduit

En entrée de l'algorithme : treillis de la phrase.

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= P(e_1).\ P(e_2|e_1).\ P(e_3|e_2)\ \dots\ P(e_n|e_{n-1}).$$

$$P(w_1|e_1).\ P(w_2|e_2).\ P(w_3|e_3).\ \dots\ P(w_n|e_n)$$

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= \boxed{P(e_1).P(w_1|e_1).} \quad \boxed{P(e_2|e_1).P(w_2|e_2).} \quad \dots \quad \boxed{P(e_n|e_{n-1}).P(w_n|e_n)}$$

Décomposition en fonction du choix de l'étiquette e_i

soit e_1 = X soit e_2 = X soit e_n = X

soit e_1 = Y soit e_2 = Y soit e_n = Y

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= \boxed{P(e_1).P(w_1|e_1).} \boxed{P(e_2|e_1).P(w_2|e_2).} \dots \boxed{P(e_n|e_{n-1}).P(w_n|e_n)}$$

Décomposition en fonction du choix de l'étiquette e_i

soit e_1 = X

soit e_1 = Y

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= \boxed{P(e_1).P(w_1|e_1).} \boxed{P(e_2|e_1).P(w_2|e_2).} \ \dots \ \boxed{P(e_n|e_{n-1}).P(w_n|e_n)}$$

Décomposition en fonction du choix de l'étiquette e_i

soit e_1 = X → soit e_2 = X

soit e_1 = Y → soit e_2 = Y

Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

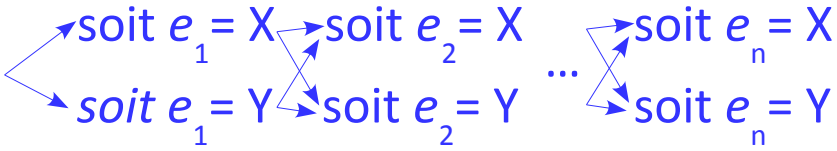
=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= \boxed{P(e_1).P(w_1|e_1).} \boxed{P(e_2|e_1).P(w_2|e_2).} \ ... \ \boxed{P(e_n|e_{n-1}).P(w_n|e_n)}$$

Décomposition en fonction du choix de l'étiquette e_i



Décodage

Décodage = trouver la séquence d'étiquettes la plus probable

Première solution (naïve)

Calculer la probabilité de chacune des séquences d'étiquettes.

=> temps de calcul exponentiel

Calcul de P(*e*).P(*w*|*e*) avec le modèle des bigrammes :

$$P(e).P(w|e) = P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1}).P(w_1|e_1).P(w_2|e_2)...P(w_n|e_n)$$

$$= \boxed{P(e_1).P(w_1|e_1).} \quad \boxed{P(e_2|e_1).P(w_2|e_2).} \quad \dots \quad \boxed{P(e_n|e_{n-1}).P(w_n|e_n)}$$

Décomposition en fonction du choix de l'étiquette e_i

soit e_1 = X, soit e_1 = Y → soit e_2 = X, soit e_2 = Y → ... → soit e_n = X, soit e_n = Y

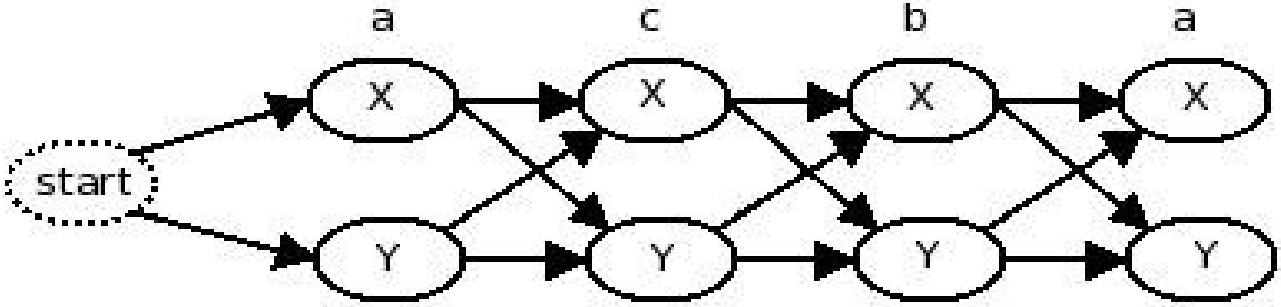
Maximisation par **programmation dynamique**

Treillis de la phrase (analyse lexicale)

Principe

Le treillis de la phrase représente sous la forme d'un graphe toutes les séquences d'étiquettes possibles, pour une séquence données de mots.
Un noeud correspond à une étiquette possible.

Exemple pour la séquence de mots *acba*



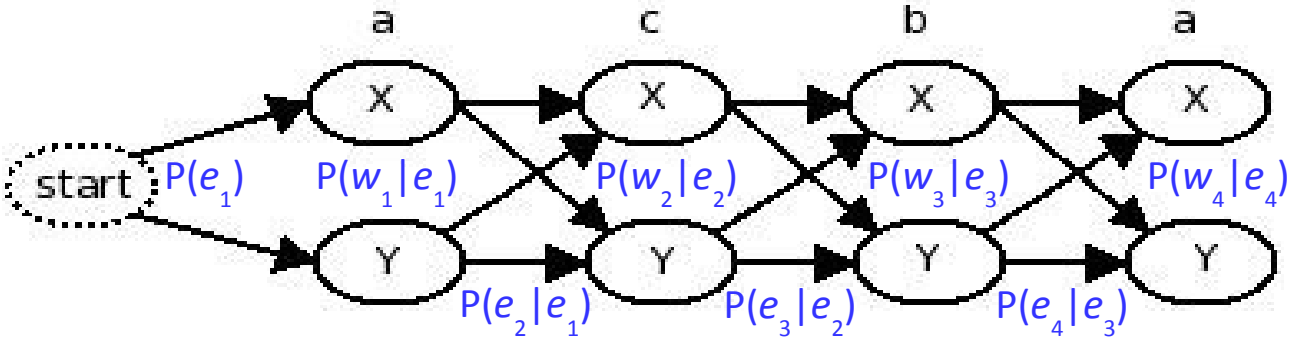
Etiquettes grammaticales

Les étiquettes attribuées aux mots peuvent provenir de différentes sources : le corpus d'apprentissage ou des ressources lexicales externes (dictionnaires, lexiques,...).

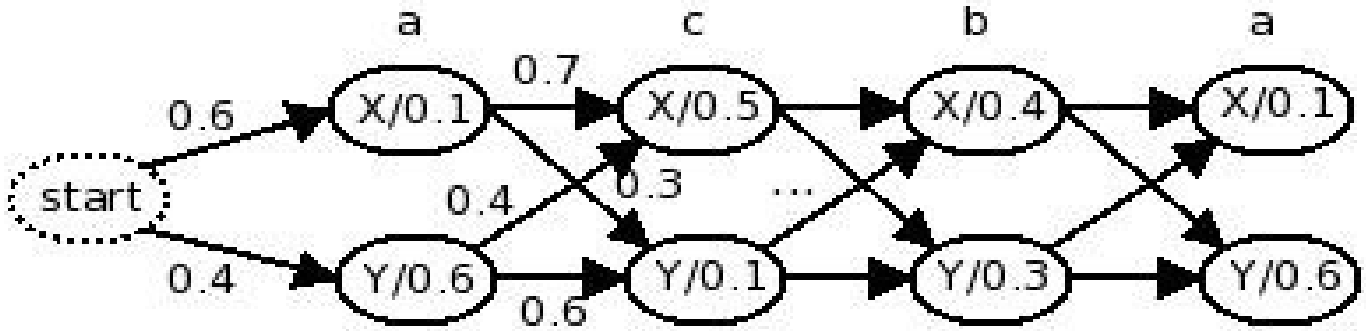
Pondération du treillis

=> Les noeuds et les arcs sont pondérés respectivement par les probabilités d'émission et de transition.

=> cas particulier: P(start) = 1, les arcs (start,Z) sont pondérés par P(Z).



Sur l'exemple :



Décodage par programmation dynamique

Trouver la séquence d'étiquettes la plus probable

=> La probabilité d'un chemin du treillis est le produit des poids du chemin.

=> Trouver le chemin le plus probable.

Algorithme de Viterbi

=> Algorithme itératif.

=> A chaque étape (mot *i*), on se sert des calculs faits à l'étape précédente (mot *i* – 1).

=> Remplissage itératif d'une matrice (ligne=étiquette ; colonne=mot) colonne par colonne en partant de la première et en terminant par la dernière.

Algorithme de Viterbi

Notations

- w_i mot à l'indice i (colonne i).
- e_j étiquette à la ligne j.

Proba[*j*,*i*] : probabilité maximale d'accéder au noeud (j,i) du treillis en partant du noeud *start*

Initialisation (*i* = 1) :

$$Proba[j,1] = P(e_j) * P(w_1 | e_j)$$

Récurrence pour *i* > 1 :

$$Proba[j,i] = \max_k \left(Proba[k,i-1] * P(e_j | e_k) * P(w_i | e_j)\right)$$

On garde en mémoire (dans *Back*[*j,i*]) l'étiquette e_k qui a permis de maximiser

Algorithme de Viterbi (suite)

Proba	a	c	...	a
X	p(X).p(a|X)	max(p(i-1).p(X|X).p(b|X), p(i-1).p(X|Y).p(b|X))	...	...
Y	p(Y).p(a|Y)	...	...	...

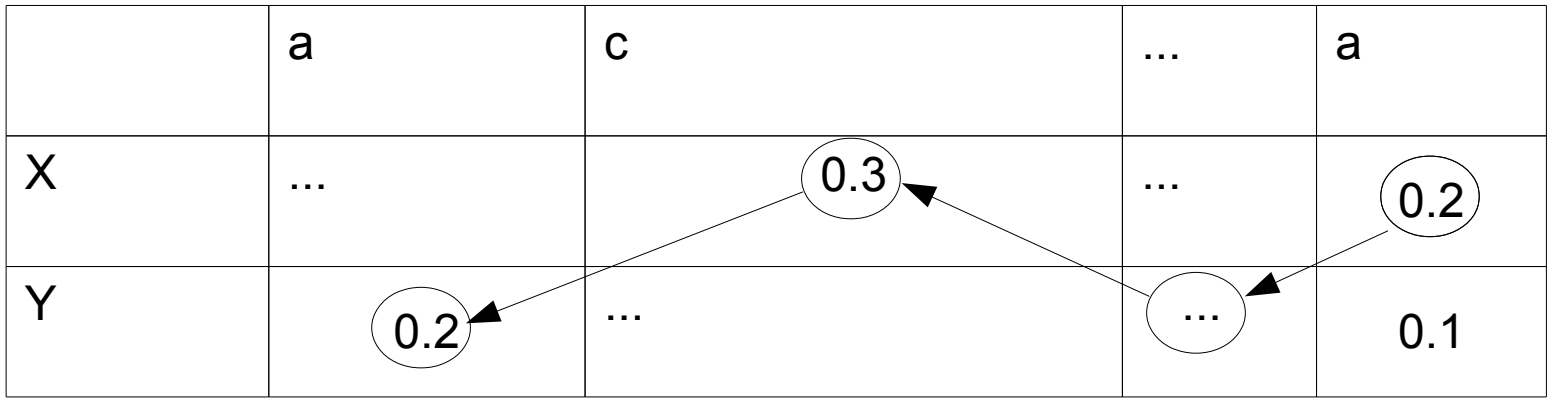
On garde en mémoire (dans *Back*[*j,i*]) l'étiquette e_k qui a permis de maximiser :

Back	a	c	...	a
X	...	0.3	...	0.2
Y	0.2	...	...	0.1

Algorithme de Viterbi (suite)

Procédure finale

Une fois la matrice remplie, on retrouve le chemin qui a donné la probabilité maximale dans la dernière colonne de *Proba* et on en déduit le chemin de l'automate des transitions correspondant grâce à la matrice *Back*.



Exercice

Trouver la séquence d'étiquettes la plus probable pour ***acba***
avec P(X) = 0.6 et P(Y) = 0.4

Probabilités d'émission P(X|a)=0.1, P(X|b)=0.4, P(X|c)=0.5
P(Y|a)=0.6, P(Y|b)=0.3, P(Y|c)=0.1

Probabilités de transitions P(X|X)=0.7, P(Y|X)=0.3, P(X|Y)=0.4, P(Y|Y)=0.6

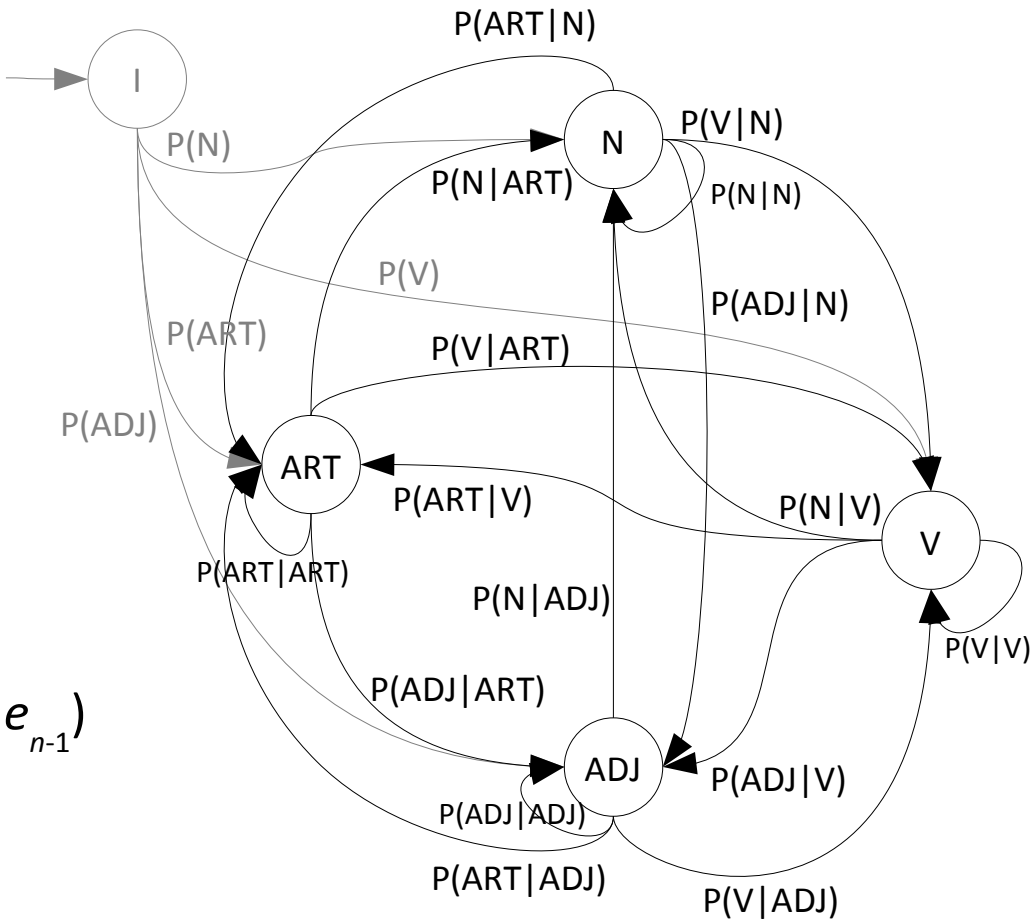
	a	c	b	a
X				
Y				

Pourquoi « modèle de Markov caché d'ordre *k* » ?

Calcul de P(*e*) pour *k*=1

$P(e)=P(e_1).P(e_2|e_1).P(e_3|e_2)...P(e_n|e_{n-1})$

→ Chaîne de Markov d'ordre 1
(automate avec transitions pondérées par des probabilités)

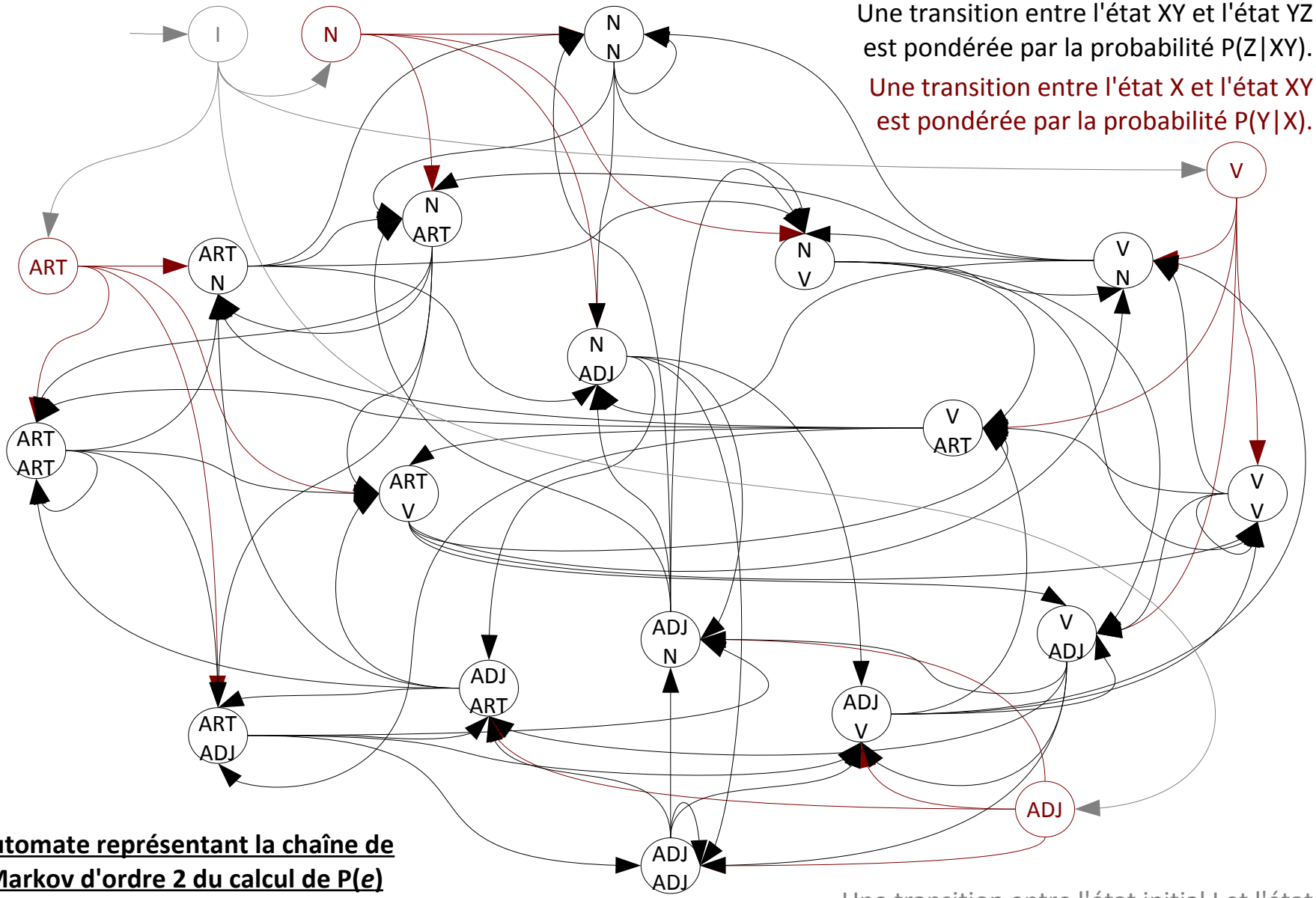


Automate représentant la chaîne de Markov d'ordre 1 du calcul de P(*e*) pour les étiquettes ART, ADJ, N, V

Pourquoi « modèle de Markov caché d'ordre *k* » ?

Une transition entre l'état XY et l'état YZ est pondérée par la probabilité P(Z|XY).

Une transition entre l'état X et l'état XY est pondérée par la probabilité P(Y|X).



Automate représentant la chaîne de Markov d'ordre 2 du calcul de P(*e*) pour les étiquettes ART, ADJ, N, V

Une transition entre l'état initial I et l'état X est pondérée par la probabilité P(X).

Pourquoi « modèle de Markov caché d'ordre *k* » ?

→ Chaîne de Markov d'ordre *k*
automate avec transitions pondérées par des probabilités

→ Modèle de Markov caché d'ordre k
automate avec transitions pondérées par des probabilités,
avec des « états cachés » = les étiquettes, qui sont inconnues

Autres modélisations probabilistes

• Les modèles de Markov sont performants et rapides.

• Cependant, il existe de nombreux modèles probabilistes dits “discriminants” permettant d'améliorer significativement les performances :

- Maximum d'entropie

- Champs conditionnels aléatoires

Performances sur le français

	Score
Markov ordre 2	96.5
Modèle discriminant	97.5

Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général: grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Désambiguisation symbolique

Principe

Application de règles de désambiguisation écrites manuellement sur l'automate afin de supprimer des ambiguités lexicales.

• en général, désambiguisation partielle (peu de règles ou règles inefficaces).

• utilisation d'heuristiques ou d'un modèle probabiliste pour déterminer la séquence d'étiquettes.

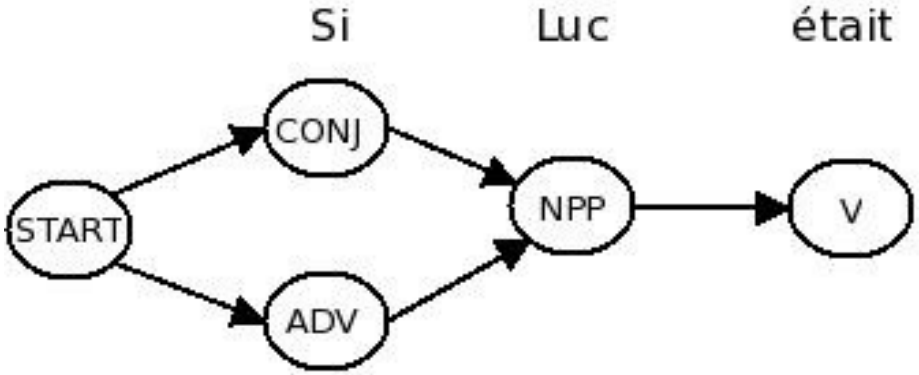
Plusieurs systèmes de règles

- Elag (intégré à Unitex) : http://igm.univ-mlv.fr/~unitex/.

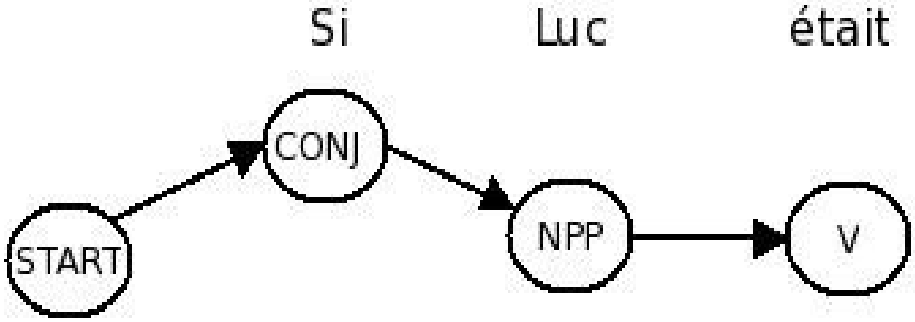
- EngCG développé pour le traitement de textes anglais.

Désambiguisation symbolique (suite)

Automate de la phrase



Automate après application de la règle Elag



Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

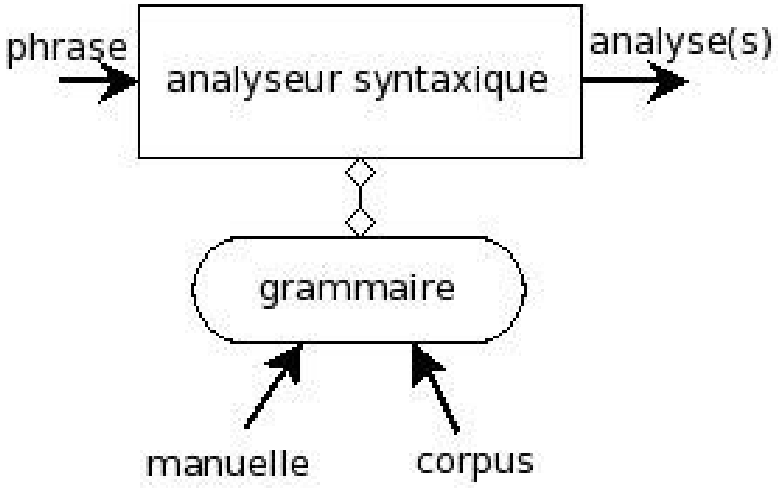
Plan

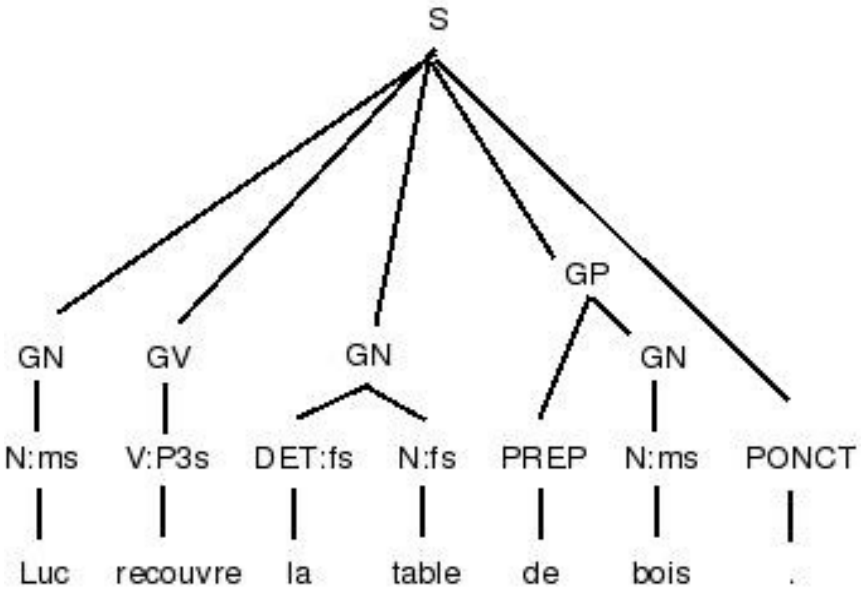
- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Analyse syntaxique : introduction

But :

Déterminer l'arbre syntaxique d'une phrase.





Les grammaires d'arbres

Grammaires hors contexte (Context-Free Grammars)

- extraction de règles à partir du corpus A → B

- deux types de règles :

=> règles lexicales, membre gauche = étiquette

dérivation = mot du lexique

=> règles contextuelles, membre gauche = constituant

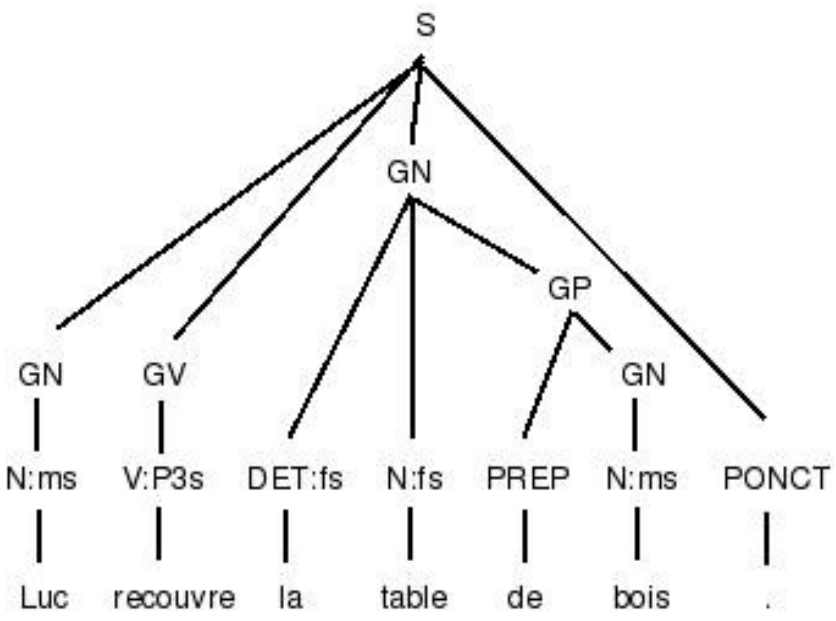
dérivation = constituant(s)/étiquette(s)

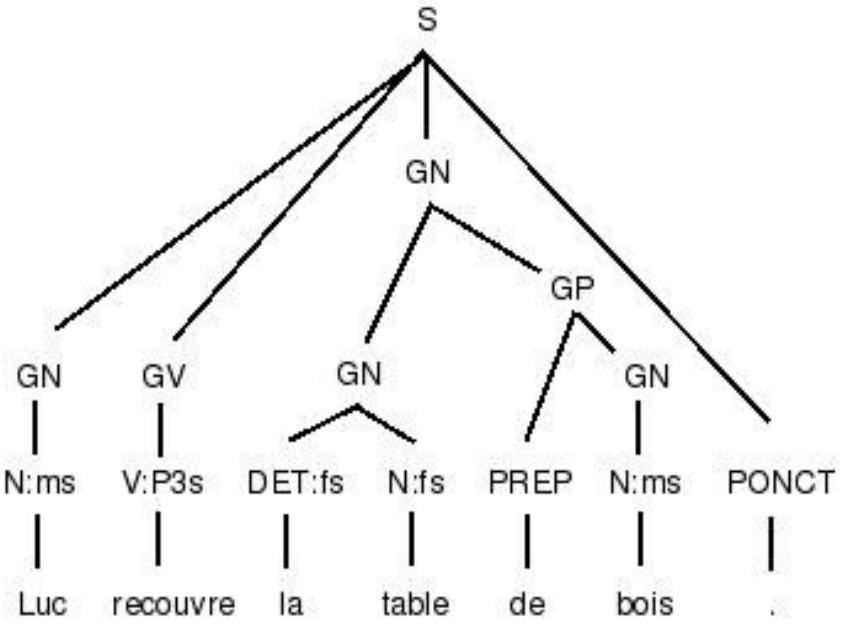
Règles contextuelles	Règles lexicales
S -> NP V	DET -> le
S -> NP V NP	DET -> la
NP -> DET NC	NPP -> Luc
NP -> NPP	V -> recouvre

Ambiguités de la langue

Ambiguités lexicales et syntaxiques

La langue est ambiguë tant au niveau lexical qu'au niveau syntaxique.





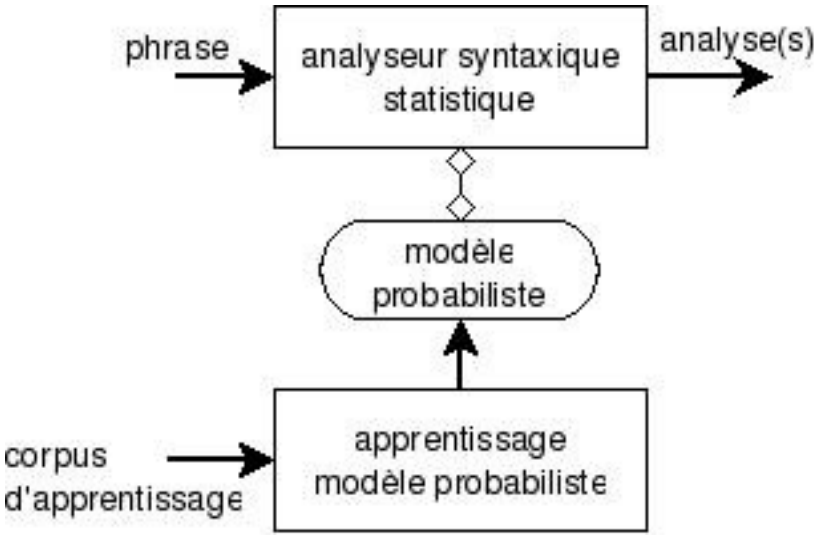
Plan

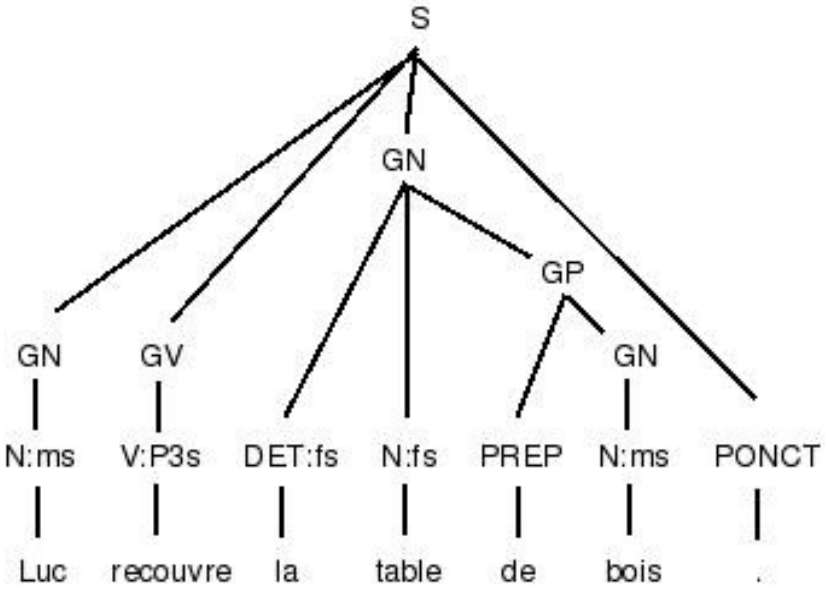
- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Solution: grammaire probabiliste

Point de vue probabiliste:

Déterminer l'arbre syntaxique d'une phrase à l'aide d'une grammaire probabiliste.





Grammaire hors contexte probabiliste

Principe

• extension des grammaires CFG

• chaque règle est associée à une probabilité :

$$P(A \rightarrow B) = \#occ(A \rightarrow B) / \#occ(A)$$

avec #occ(x) le nombre d'occurrences de x dans le corpus d'apprentissage.

• décodage : divers algorithmes de programmation dynamique à la Viterbi, CYK ou encore Earley.

Règles contextuelles		Règles lexicales	
S -> NP V	0.4	DET -> le	0.2
S -> NP V NP	0.6	DET -> la	0.2
NP -> DET NC	0.5	NPP -> Luc	1.0
NP -> NPP	0.5	V -> recouvre	1.0

Plan

- Étiquetage grammatical
  - Modèle de Markov caché
  - Décodage : algorithme de Viterbi
  - Désambiguisation symbolique
- Analyse syntaxique
  - Principe général : grammaires hors contexte
  - Grammaires probabilistes
  - Évaluation de la qualité des analyses

Evaluation des analyses produites

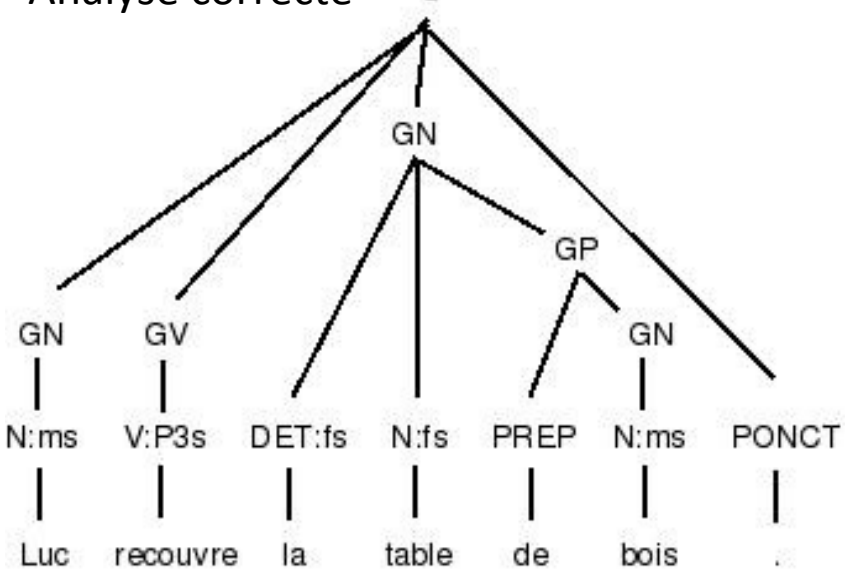
Plusieurs mesures

- rappel : nombre de constituants corrects sur le nombre total de constituants de l'analyse correcte

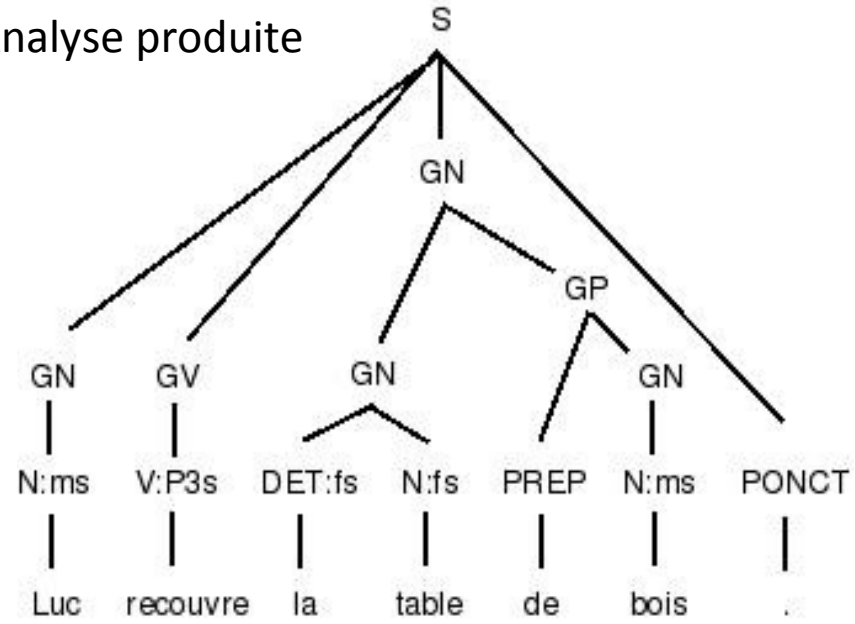
- précision : nombre de constituants corrects sur le nombre total de constituants de l'analyse produite

- f-mesure : $2.\frac{precision.rappel}{rappel + precision}$

Analyse correcte



Analyse produite



Exercice

1) Trouver les arbres de dérivations possibles à partir de la grammaire suivante pour la phrase: *le chien marron aboie*

Règles contextuelles		Règles lexicales			
S -> NP V	1.0	DET -> le	0.6	NC -> marron	0.2
NP -> DET NC NC	0.1	PRO -> le	0.4	A -> marron	0.8
NP -> DET A NC	0.3	NC -> chien	0.8	V -> aboie	1.0
NP -> DET NC A	0.6	A -> chien	0.2		

2) Déterminer l'arbre qui a la meilleure probabilité (multipliez simplement les probabilités associées aux règles).

3) Déterminer les valeurs de rappel, précision et f-mesure pour chaque arbre trouvé.