

CYK Probabiliste

Lapraye & Lévêque & Viegas

Paris VII

1^{er} juillet 2016

L'Algorithme CYK

- Un algorithme de parsing ascendant
- Complexité $\mathcal{O}(|G|n^3)$
- Parsing tabulaire
- Extension aux grammaire hors-contexte probabilistes (PCFG)

Algorithm 1 CYK normal

```
function CKY( $w[1..n]$ ,  $G : \langle NT, T, P, \rho \rangle$ ,  $R[1..n, 1..n]$ )  
  for all  $i$  de 1 à  $n$  do  
    if  $N \rightarrow w_i$  then  
       $R[i, i+1] \leftarrow N$   
  for all  $k$  de 2 à  $n$  do  
    for all  $i$  de 1 à  $n - i + 2$  do  
      for all  $j$  de  $i+1$  à  $i+k$  do  
        for all  $A \in R[i, j]$  do  
          for all do  
            for all  $NT \in P$  do  
              if  $NT \rightarrow AB$  then  
                 $R[i, i+k] = R[i, i+k] \cup \{NT\}$   
  if  $S \in R[0, n]$  then return True
```

Exemple déroulé

La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

D				
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

D	N			
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

D	N	V		
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

D	N	V	D	
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

D	N	V	D	N
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

SN				
D	N	V	D	N
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

SN			SN	
D	N	V	D	N
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

		SV		
SN			SN	
D	N	V	D	N
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

Exemple déroulé

S				
		SV		
SN			SN	
D	N	V	D	N
La	Fille	Mange	Une	Pomme

S -> SN SV

SN -> D N

SV -> V SN

V -> mange

D -> la | une

N -> fille | pomme

- Les CFG : un quadruplet (Σ, V, S, P)

Les PCFG

- Les CFG : un quadruplet (Σ, V, S, P)
- Les CFG pondérées : ajout d'une fonction de poids
 $f : p \mapsto \alpha, w \in W, \alpha \in \mathbb{R}$

Les PCFG

- Les CFG : un quadruplet (Σ, V, S, P)
- Les CFG pondérées : ajout d'une fonction de poids
 $f : p \mapsto \alpha, w \in W, \alpha \in \mathbb{R}$
- Les CFG probabilistes : les poids correspondent à des probabilités pour une réécriture donnée.

$$f : p \mapsto \alpha, p \in P, \alpha \in [0, 1]$$

$$\forall X \in V, \sum_{X \rightarrow \alpha} p(X \rightarrow \alpha) = 1$$

Les PCFG

- Les CFG : un quadruplet (Σ, V, S, P)
- Les CFG pondérées : ajout d'une fonction de poids
 $f : p \mapsto \alpha, w \in W, \alpha \in \mathbb{R}$
- Les CFG probabilistes : les poids correspondent à des probabilités pour une réécriture donnée.

$$f : p \mapsto \alpha, p \in P, \alpha \in [0, 1]$$

$$\forall X \in V, \sum_{X \rightarrow \alpha} p(X \rightarrow \alpha) = 1$$

- Les CFG probabilistes représentent un modèle de prédiction déduit à partir du corpus dont elles sont extraites.

Les PCFG

- Les CFG : un quadruplet (Σ, V, S, P)
- Les CFG pondérées : ajout d'une fonction de poids
 $f : p \mapsto \alpha, w \in W, \alpha \in \mathbb{R}$
- Les CFG probabilistes : les poids correspondent à des probabilités pour une réécriture donnée.

$$f : p \mapsto \alpha, p \in P, \alpha \in [0, 1]$$

$$\forall X \in V, \sum_{X \rightarrow \alpha} p(X \rightarrow \alpha) = 1$$

- Les CFG probabilistes représentent un modèle de prédiction déduit à partir du corpus dont elles sont extraites.
- Extraction des PCFG

La forme normale de Chomsky (CNF)

- l'axiome S est inaccessible
- Les règles de production adoptent une des formes suivantes, avec ε la production vide, $A, B, C, D \in V$, et $e \in \Sigma$:

$$A \rightarrow BC$$

$$D \rightarrow e$$

$$S \rightarrow \varepsilon$$

Transformer la grammaire en CNF

- 1 Faire en sorte que l'axiome n'apparaisse plus dans les parties droites de règles
- 2 Supprimer les règles d'effacement (c'est à dire de la forme $A \rightarrow^* \varepsilon$) pour les non-terminaux autres que l'axiome.
- 3 Faire en sorte que tout les terminaux apparaissent uniquement dans la partie droite de règles unaires
- 4 Remplacer les règles de production n-aire par des règles binaires équivalentes.
- 5 Supprimer les productions singulières de non-terminaux, c'est à dire les règles de la forme $A \rightarrow B$ avec $A, B \in V$

Transformer la grammaire en CNF

- 1 Faire en sorte que l'axiome n'apparaisse plus dans les parties droites de règles
- 2 Supprimer les règles d'effacement (c'est à dire de la forme $A \rightarrow^* \varepsilon$) pour les non-terminaux autres que l'axiome.
- 3 Faire en sorte que tout les terminaux apparaissent uniquement dans la partie droite de règles unaires
- 4 Remplacer les règles de production n-aire par des règles binaires équivalentes.
- 5 Supprimer les productions singulières de non-terminaux, c'est à dire les règles de la forme $A \rightarrow B$ avec $A, B \in V$

Le corpus Sequoia

- Un corpus diversifié
- Des phrases de longueur variable
- Extraction de la grammaire



L'Algorithme CYK

Algorithm 2 CYK probabiliste (argmax)

```
function CYK( $w[1..n]$ ,  $G :< NT, T, P, \rho >$ ,  $R[1..n, 1..n]$ )      ▷  $w$  : mot ;  $G$  :  
Grammaire ;  $C$  : charte  
  for all  $k \leftarrow 2, |n|$  do                                     ▷ Boucle gérant l'empan  
    for all  $i \leftarrow k - 2, 0$  do  
      for all  $nt \in NT$  do                                       ▷ Boucle faisant l'argmax  
         $best = 0$ ;  
        for all  $nt \rightarrow nt^1 nt^2 \subset P$  do  
          for all  $j \leftarrow i + 1, k - 1$  do  
             $t1 = R[i, j][nt^1]$   
             $t2 = R[j, k][nt^2]$   
             $candidate = t1 * t2 * \rho(nt \rightarrow nt^1 nt^2)$ ;  
            if  $candidate > best$  then  
               $best = candidate$ ;  
           $R[i, k][nt] = best$ ;
```

Notre implémentation du CYK

- Un algorithme gourmand en temps et en mémoire

Notre implémentation du CYK

- Un algorithme gourmand en temps et en mémoire
- Optimiser la recherche des réécritures possibles

Notre implémentation du CYK

- Un algorithme gourmand en temps et en mémoire
- Optimiser la recherche des réécritures possibles
- Le backtracking

- Une évaluation problématique.


- Une évaluation problématique.
- La mesure dite de PARSEVAL

- Une évaluation problématique.
- La mesure dite de PARSEVAL
- Précision, Rappel et f-mesure


Résultats


	Étiqueté	Non-étiqueté
Précision	0.829	0.870
Rappel	0.828	0.868
F-mesure	0.829	0.869

Références

 Brian Roark, Richard Sproat.
Computational Approaches to Morphology and Syntax.
Oxford University Press, 2007.

 Mariana Romanyshyn, Vsevolod Dyomkin.
The Dirty Little Secret of Constituency Parser Evaluation, 2014.
<http://tech.grammarly.com/blog/posts/The-Dirty-Little-Secret-of-Constituency-Parser-Evaluation.html>

 Martin Lange, Hans Leiss
« To CNF or not to CNF : An Efficient Yet Presentable Version of the CYK Algorithm », 2009
Informatica Didactica N° 8

 E. Black, S. Abney et al.
« Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars »
1991, DARPA Speech and Natural Language Workshop