Grammaires formelles : Algorithme de Cocke (CKY)

Karën Fort

karen.fort@sorbonne-universite.fr / https://members.loria.fr/KFort/

LETTRES SORBONNE UNIVERSITÉ

Quelques sources d'inspiration

par ordre d'importance décroissant

- cours de et interactions avec M. Cori
- cours d'A. Rozenknop (Paris 13)
- cours de D. Battistelli (Paris 3), grâce aux notes de C. Riquier (Master 2, Paris 4)

Analyses descendantes ou ascendantes

Deux types de méthodes, qui correspondent à deux philosophies de l'analyse (*parsing*), sont possibles :

1. une analyse descendante, qui consiste à postuler la forme que peuvent prendre les phrases, et à vérifier si l'énoncé à analyser entre dans l'une des formes de phrases possibles

→ construit des arbres inutiles dont la liste des feuilles entre en contradiction avec la phrase à analyser

2. une analyse ascendante, qui consiste à partir des unités qui constituent l'énoncé, et à vérifier si des regroupements de ces unités sont possibles. En effectuant des regroupements de plus en plus grands, on tente d'obtenir des phrases

→ construit des arbres inutiles qui n'entreront jamais dans un arbre dont la racine représente une phrase

Gestion de la multiplicité des solutions

Au cours du processus d'analyse, il peut apparaître différentes solutions (partielles) dont certaines ne sont pas correctes. On gère cela de plusieurs manières :

1. méthodes avec retour en arrière : on fait des hypothèses, et si ces hypothèses s'avèrent erronées à un moment donné, on les abandonne et on en essaye d'autres
2. méthodes en parallèle : on essaye les différentes hypothèses simultanément
3. méthodes déterministes : on emploie une stratégie qui permet de faire le bon choix au fur et à mesure de l'analyse. Il est clair que ce type de méthode ne permet pas de résoudre le cas des phrases réellement ambiguës

La petite histoire de l'algorithme CKY

Échange de mails avec C. Boitet, mémoire vivante du TAL (30/11/14)

D'après Martin Kay, qui y était [...], c'était un colloque de scientifiques qui s'intéressaient au domaine pas encore nommé de l'informatique.

Je me souviens que Cocke était en architecture des ordinateurs. Il y avait des astronomes aussi (c'était avant 1960, du temps où [...]).

Certains se sont mis à discuter du problème de l'analyse en utilisant des grammaires, et de la façon dont on pourrait éviter la fameuse explosion combinatoire.

[...]

Eh bien, ce M. Cocke a pris un bout de papier, et a dit qqch comme « si je comprends bien, vous voulez faire ceci », en dessinant la fameuse matrice et en y plaçant un arbre exemple. Et il proposa de la remplir ligne par ligne et de bas en haut. Ce fut le tout premier bloc de « programmation dynamique » de l'histoire du TALN. [...] Et M. Cocke n'a jamais publié ça, vu que c'était hors de son domaine.

Présentation de l'algorithme de Cocke

ou de Cocke-Younger-Kasami (CKY ou CYK)

L'algorithme CKY est un algorithme d'analyse ascendante sans retour en arrière (en parallèle)

Il s'agit d'un algorithme de programmation dynamique : les résultats intermédiaires sont stockés, afin d'éviter de les recalculer

Pré-requis

1. une grammaire indépendante du contexte $G = \{V_T, V_N, S, R\}$ sous forme normale de Chomsky
2. une suite $u_1 u_2 ... u_m$ de m formes lexicales (mots). À chaque forme lexicale est associé un ensemble de catégories.

Intuition

On examine :

1. tous les regroupements possibles de 2 formes lexicales
2. puis les regroupements possibles de 3 formes
3. jusqu'aux regroupements de $m-1$ formes
4. et à l'unique regroupement de m formes

La grammaire sert à sélectionner certains de ces regroupements.
Tout regroupement sélectionné est appelé un syntagme

 La phrase est acceptée si et seulement si il existe un syntagme construit à l'aide des m formes

Démonstration au tableau

Grammaire sous FNC :

$$P = \begin{cases} S \longrightarrow SN\ SV & (1) \\ SV \longrightarrow V\ SN & (2) \\ SV \longrightarrow PRO\ V & (3) \\ SN \longrightarrow D\ N & (4) \\ SN \longrightarrow SN\ ADJ & (5) \\ SN \longrightarrow D\ SN_1 & (6) \\ SN_1 \longrightarrow ADJ\ N & (7) \\ D \longrightarrow la & (8) \\ PRO \longrightarrow la & (9) \\ ADJ \longrightarrow belle \mid ferme & (10) \\ N \longrightarrow belle \mid ferme \mid porte & (11) \\ V \longrightarrow ferme \mid porte & (12) \end{cases}$$

Chaîne à analyser : la belle ferme la porte

Tableau

Démonstration en ligne

http://lxmls.it.pt/2015/cky.html

À tester avec "la belle ferme la porte" et 500 ms d'intervalle avec :

```
S -> SN SV
SV -> V SN
SV -> PRO V
SN -> D N
SN -> SN ADJ
SN -> D SN_1
SN_1 -> ADJ N
D -> la
PRO -> la
ADJ -> belle
ADJ ->  ferme
N -> belle
N -> ferme
N -> porte
V -> ferme
V -> porte
```

Sources

Analyse syntaxique

Pour finir

- CQFR : Ce Qu'il Faut Retenir
- TD

Parsing :

- définition
- algorithme CKY

Exercice 1

Exercez-vous à appliquer l'algorithme CKY sur la chaîne $w = abab$, avec la grammaire suivante :

$$P = \begin{cases} S \longrightarrow XY & (1) \\ T \longrightarrow ZT|a & (2) \\ X \longrightarrow TY & (3) \\ Y \longrightarrow YT|b & (4) \\ Z \longrightarrow TZ|b & (5) \end{cases}$$

Exercice 1

Exercez-vous à appliquer l'algorithme CKY sur la chaîne $w = abab$, avec la grammaire suivante :

$$P = \begin{cases} S \longrightarrow XY & (1) \\ T \longrightarrow ZT|a & (2) \\ X \longrightarrow TY & (3) \\ Y \longrightarrow YT|b & (4) \\ Z \longrightarrow TZ|b & (5) \end{cases}$$

Vous trouverez une vidéo détaillant une solution (en anglais) ici :
https://www.youtube.com/watch?v=t4AVU0jDD_A

Exercice 2

Exercez-vous à appliquer l'algorithme CKY sur la chaîne *bbabaa*, avec la grammaire suivante :

$$
P = \begin{cases}
S \longrightarrow AB & (1) \\
S \longrightarrow BS_1 & (2) \\
S \longrightarrow BV_a & (3) \\
S_1 \longrightarrow AB & (4) \\
A \longrightarrow BA & (5) \\
A \longrightarrow a & (6) \\
B \longrightarrow aa & (7) \\
B \longrightarrow V_a S_1 & (8) \\
B \longrightarrow S_1 V_a & (9) \\
B \longrightarrow S_1 S_1 & (10) \\
B \longrightarrow b & (11) \\
V_a \longrightarrow a & (12)
\end{cases}
$$

Exercice 2

Exercez-vous à appliquer l'algorithme CKY sur la chaîne *bbabaa*, avec la grammaire suivante :

$$P = \begin{cases} S \longrightarrow AB & (1) \\ S \longrightarrow BS_1 & (2) \\ S \longrightarrow BV_a & (3) \\ S_1 \longrightarrow AB & (4) \\ A \longrightarrow BA & (5) \\ A \longrightarrow a & (6) \\ B \longrightarrow aa & (7) \\ B \longrightarrow V_a S_1 & (8) \\ B \longrightarrow S_1 V_a & (9) \\ B \longrightarrow S_1 S_1 & (10) \\ B \longrightarrow b & (11) \\ V_a \longrightarrow a & (12) \end{cases}$$

Vous trouverez une vidéo détaillant une solution (en anglais) ici :
https://www.youtube.com/watch?v=b98Uyj7JHIU

Exercice 3

Soit la grammaire suivante :

$$P = \begin{cases} S \longrightarrow \text{NP VP} & (1) \\ NP \longrightarrow \text{DET N} & (2) \\ N \longrightarrow \text{N PP} \mid \text{N ADJ} & (3) \\ PP \longrightarrow \text{PREP N} & (4) \\ ADJ \longrightarrow \text{américain} & (5) \\ DET \longrightarrow \text{le} & (6) \\ N \longrightarrow \text{football} \mid \text{joueur} & (7) \\ PREP \longrightarrow \text{de} & (8) \\ VP \longrightarrow \text{arrive} & (9) \end{cases}$$

Détaillez l'algorithme de Cocke (ou CKY) appliqué à la phrase suivante (selon la grammaire ci-dessus) :

Le joueur de football américain arrive.

Donnez la conclusion de l'algorithme en la justifiant.