Conception et implémentation

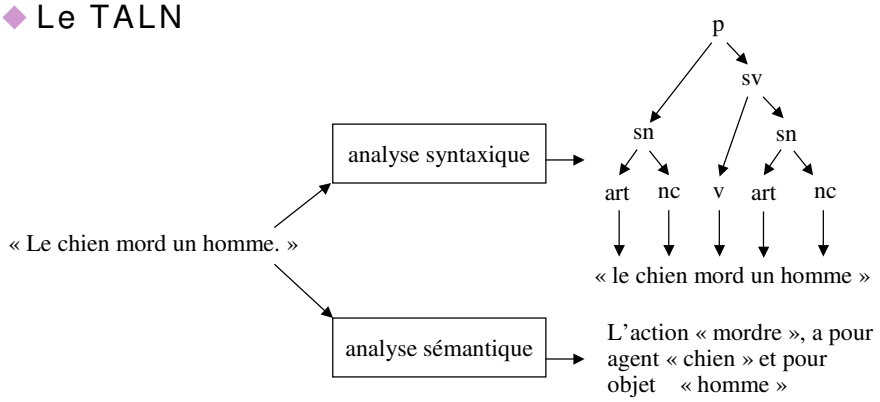
IFT-17586 Intelligence artificielle I - A2002

Séance 10

- Implémentation du TALN en Prolog

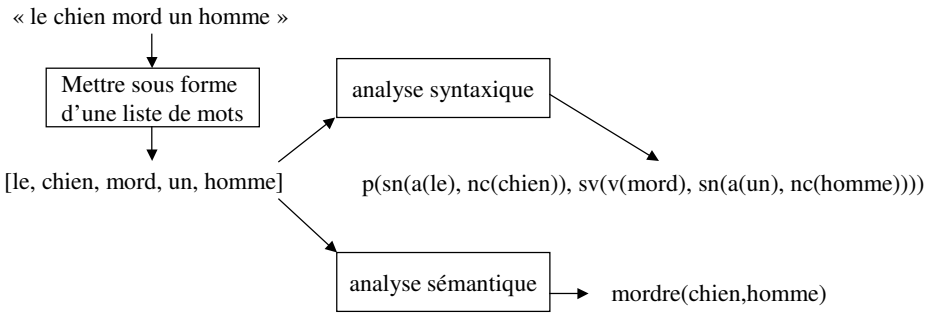
Introduction

- Le TALN



Introduction

- Le TALN avec PROLOG



TALN

- Besoins pour le traitement :

Phrase → Liste de mots

Règle de grammaire → Clause : fait ou règle

Symbole → Prédicat

⇒ **Idée : balayer la liste mot par mot**

TALN

```
ma_phrase(X, Z) :-
        syntagme_nominal(X, Y),
        syntagme_verbal(Y, Z).
```

- Analyse de la phrase :

```
?- ma_phrase([le, chien, mord, un, homme], [ ]).

X=[le, chien, mord, un, homme]
Z=[ ]
Y=[mord, un, homme]
```


TALN

- Grammaire #1
  - Ne fonctionne pas

```
p :- sn,  sv.
sn:- art, nc.
sv:- v, sn.
art:- le.
art:- un.
nc:- chien.
nc:- homme.
v:- mord.
```

```
| ?- p.
no
```

TALN

Grammaire #2

- Attribuer les symboles terminaux
- Fonctionne mais ne sert à rien

```
p :- sn,  sv.
sn:- art(A), nc(NC).
sv:- v(V), sn.
art( le ).
art( un ).
nc( chien ).
nc( homme ).
v( mord ).
```

```
| ?- p.
yes
```


TALN

Grammaire #3

- Attribuer les symboles terminaux et non-terminaux
- Fonctionne pour générer ou valider un arbre syntaxique

```
p( p(SN,SV) ) :- sn(SN), sv(SV).
sn( sn(A,NC) ) :- art(A), nc(NC).
sv( sv(V, SN ) ):- v(V), sn(SN).
art( a(le) ).
art( a(un) ).
nc( nc(chien) ).
nc( nc(homme) ).
v( v(mord) ).
```

```
| ?- p(P).
P = p(sn(a(le),nc(chien)),sv(v(mord),sn(a(le),nc(chien)))) ;
...
| ?- p(p(sn(a(le),nc(chien)),sv(v(mord),sn(a(le),nc(chien)))) ).
yes
```

TALN

Grammaire #4

- Fonctionne pour générer ou valider syntaxiquement une phrase sous forme de liste de mots

```
p( P ) :- sn(SN),  sv(SV), append(SN, SV, P).
sn( SN ) :- art(A), nc(NC), append(A, NC, SN).
sv( SV ):- v(V), sn(SN), append(V, SN,SV).
art( [le] ).
art( [un] ).
nc( [chien] ).
nc( [homme] ).
v( [mord] ).
```

```
| ?- p(P).
P = [le,chien,mord,le,chien] ;
...
| ?- p([le,chien,mord,un,homme]).
yes
| ?- p([le,chien,mord,homme]).
no
```


TALN

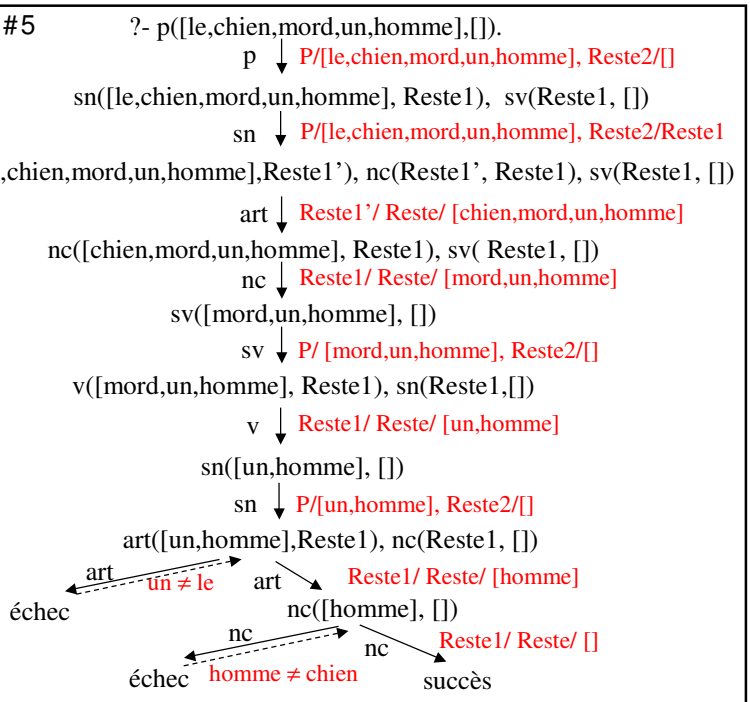
Grammaire #5

- Fonctionne pour générer ou valider syntaxiquement une phrase sous forme de liste de mots

```
p(P, Reste2) :- sn(P, Reste1),  sv(Reste1, Reste2).
sn(P,Reste2):- art(P,Reste1), nc(Reste1,Reste2).
sv(P,Reste2):- v(P,Reste1), sn(Reste1,Reste2).
art( [ le | Reste ], Reste ).
art( [ un | Reste ], Reste ).
nc( [ chien | Reste ], Reste ).
nc( [ homme | Reste ], Reste ).
v( [ mord | Reste ], Reste).
```

```
| ?- p(P,[]).
P = [le,chien,mord,le,chien] ;
...
| ?- p([le,chien,mord,un,homme],[]).
yes
| ?- p([le,chien,mord,homme],[]).
no
```

- Grammaire #5
  - exemple



L'opérateur DCG « --> »

- Remplace l'opérateur « :- »
- Permet de ne pas écrire les deux paramètres indispensables pour traiter les phrases
- Exemples :

```
ma_phrase(X,Z) :-
    syntagme_nominal(X,Y), syntagme_verbal(Y,Z).
ma_phrase -->
    syntagme_nominal, syntagme_verbal.
article( [le | R], R).
article --> [le].
```


TALN

Grammaire #6

- Même chose que #5, mais les 2 listes sont traitées par l'opérateur DCG (Definite Clause Grammar)
- L'appel de p/0 doit se faire comme s'il s'agit de p/2

```
p --> sn, sv.
sn --> a, nc.
sv --> v, sn.
a --> [le].
a --> [un].
nc --> [chien].
nc --> [homme].
v --> [mord].
```

```
| ?- p(P,[]).
P = [le,chien,mord,le,chien] ;
...
| ?- p([le,chien,mord,un,homme],[]).
yes
| ?- p([le,chien,mord,homme],[]).
no
```


TALN

Grammaire #7

- Même chose que #6, avec une grammaire attribuée pour l'analyse syntaxique
- L'appel se fait avec un argument de plus qu'avec la grammaire #6 : ?- p(ARBRE,PHRASE,[]).

```
p( p(SN,SV) ) --> sn(SN), sv(SV).
sn( sn(A,NC) ) --> a(A), nc(NC).
sv( sv(V,SN) ) --> v(V), sn(SN).
a( a(le) ) --> [le].
a( a(un) ) --> [un].
nc( nc(chien) ) --> [chien].
nc( nc(homme) ) --> [homme].
v( v(mord) ) --> [mord].
```

```
| ?- p(A,P,[]).
A = p(sn(a(le),nc(chien)),sv(v(mord),sn(a(le),nc(chien)))) ,
P = [le,chien,mord,le,chien] ;
...
| ?- p(A,[le,chien,mord,un,homme] ,[]).
A = p(sn(a(le),nc(chien)),sv(v(mord),sn(a(un),nc(homme))))
| ?- p(A,[le,chien,mord,homme] ,[]).
no
```

Analyse sémantique

- Analyse syntaxique ⇒ arbre syntaxique

```
ph( sn( art(le), nom(chien) ),
    sv(   v(mord),
          sn( art(un), nom(homme) )
    )
)
```

- Analyse sémantique ⇒ schéma sémantique

  mord(chien, homme)

  ACTION(AGENT, OBJET)

TALN

- Grammaire #8
  - même principe que #7, avec une grammaire attribuée pour le traitement sémantique

```
p( ACTION(AGENT, OBJET) ) --> sn(AGENT), sv(ACTION, OBJET).
sn( AGENT ) --> a, nc(AGENT).
sv( ACTION,OBJET ) --> v(ACTION), sn(OBJET).
a --> [le].
a --> [un].
nc( chien ) --> [chien].
nc( homme ) --> [homme].
v( mordre ) --> [mord].
```

```
| ?- p(A,P,[]).
A = mordre(chien,chien) , P = [le,chien,mord,le,chien]
...
| ?- p(A,[le,chien,mord,un,homme],[]).
A = mordre(chien,homme)
| ?- p(mordre(chien, chien),P,[]).
P = [le,chien,mord,le,chien]
```

Utilisation d'autres prédicats que les symboles

- Exemple avec l'analyse sémantique : Agent ≠ Objet

```
sém_phrase(Action(Agent, Objet) ) -->
    syntagme_nominal(Agent),
    syntagme_verbal(Action, Objet),
    Agent \== Objet.
```

- Erreur : les arités des prédicats sont différentes

```
sém_phrase(Action(Agent, Objet) ) -->
    syntagme_nominal(Agent),
    syntagme_verbal(Action, Objet),
    {Agent \== Objet}.
```


TALN

- Grammaire #9
  - Même chose que #8 avec une condition : agent \= objet
  - Pour spécifier cette condition, utiliser les { }

```
p( ACTION(AGNT, OBJ) ) --> sn(AGNT), sv(ACTION, OBJ), { AGNT \= OBJ }.
sn( AGENT ) --> a, nc(AGENT).
sv( ACTION,OBJET ) --> v(ACTION), sn(OBJET).
a --> [le].
a --> [un].
nc( chien ) --> [chien].
nc( homme ) --> [homme].
v( mordre ) --> [mord].
```

```
| ?- p(A,P,[]).
A = mordre(chien,homme) , P = [le,chien,mord,le,homme] ;
...
| ?- p(A,[le,chien,mord,un,homme],[]).
A = mordre(chien,homme)
| ?- p(A,[le,chien,mord,un,chien],[]).
no
| ?- p(mordre(chien, chien),P,[]).
no
```

TALN : grammaires attribuées et opérateur DCG

- Ajouter des paramètres et des procédures

Analyse syntaxique :

```
syn_phrase( ph(SN, SV) ) -->
        syntagme_nominal(SN), syntagme_verbal(SV).
article( art(le) ) --> [le].
```

Analyse sémantique :

```
sém_phrase( ACTION(AGENT, OBJET) ) -->
        syntagme_nominal(AGENT),
        syntagme_verbal(ACTION, OBJET).
verbe( mordre ) --> [mord].
```