Os diferentes níveis de análise lingüística

- Quais são as palavras;
- Como elas são combinadas para formar sentenças;
- O que as palavras significam;
- Como o significado das palavras contribui para o significado da sentença, etc.

Os diferentes tipos de conhecimento para o processamento das línguas naturais (PLN)

1. **Conhecimento fonético**: como as palavras são relacionadas aos sons (sistemas baseados em fala);
2. **Conhecimento morfológico**: como as palavras são construídas a partir de unidades básicas, chamadas *morfemas*;
3. **Conhecimento sintático**: como as palavras são organizadas em sentenças, qual o papel estrutural de cada palavra e que frases são subpartes de outras;

4. **Conhecimento semântico**: qual o significado das palavras e como esses significados se combinam para formar o significado da sentença;
5. **Conhecimento pragmático**: como as sentenças são usadas em diferentes situações e como o seu uso afeta a interpretação da sentença;
6. **Conhecimento do discurso**: como as sentenças imediatamente precedentes afetam o significado da próxima sentença (pronomes e aspectos temporais);
7. **Conhecimento do mundo**: conhecimento geral sobre o mundo que os usários precisam para manter a conversação (o que usuário deve saber sobre as crenças e metas dos outros).

Exemplos

1. A linguagem é um dos aspectos fundamentais do comportamento humano, sendo um dos componentes cruciais das nossas vidas.
2. Sapos verdes tem nariz grande.
3. Idéias verdes tem nariz grande.
4. Grande tem idéias verdes nariz.

Compreensão de Texto

Gramáticas

- Linguagem: conjunto de sentenças
- Sentença: conjunto de um ou mais símbolos (palavras) do vocabulário
- Gramática: especificação formal e finita desse conjunto
- A maioria das linguagens comporta um número infinito de sentenças

Gramáticas de estrutura de frase ou gramáticas de produção

- T = vocabulário dos terminais (palavras) da linguagem sendo definida
- N = vocabulário dos não-terminais (diferentes dos terminais)
- V = T U N
- P = conjunto de produções da forma a $\rightarrow$ b
- S = símbolo inicial (membro de N)
- Operação básica: reescrever uma seqüência qualquer de símbolos que contenham o símbolo **a** como **b**

Exemplo

- N = {S}
- T = {a,b,c}
- P = {S $\rightarrow$ aSc; S $\rightarrow$ b}
- Linguagem definida pela gramática: ...

- Linguagem definida pela gramática:
  b, abc, aabcc, aa...abcc...c
- Posso escolher a ordem em que efetuo as produções

Hierarquia de Chomsky

1. Gramáticas de estrutura de frase não restritas (tipo 0)
2. Gramáticas sensíveis ao contexto (tipo 1)
3. Gramáticas livres de contexto (tipo 2)
4. Gramáticas regulares (tipo 3)

Gramáticas Regulares

Duas formas:

- $A \rightarrow Bt$; $A \rightarrow t$
- $A \rightarrow tB$; $A \rightarrow t$
- Exemplo:
  $S \rightarrow aA$
  $A \rightarrow bB$
  $B \rightarrow cA$
  $B \rightarrow d$
- Também chamadas de gramáticas de estado finito.

Gramáticas Regulares (cont.)

- Linguagem gerada: abcb...cbd (“ab” + n X “cb” + d)
- x (x) ((x)) (((x)))... (exemplo clássico)
- O homem (que feriu a menina (que estava aqui)) fugiu.

Gramáticas Livres de Contexto

- Produções da forma A $\rightarrow$ x,
  onde x = zero ou mais símbolos terminais e não terminais.
- Exemplo: x (x) ((x)) (((x)))...

- $\mathrm{S} \rightarrow (\mathrm{S})$
- $\mathrm{S} \rightarrow \mathrm{x}$

Gramáticas Livres de Contexto (cont.)

- SENTENCA → NOME_PROPRIO SINTAGMA_VERBAL
- NOME_PROPRIO → joao
- NOME_PROPRIO → maria
- SINTAGMA_VERBAL → VERBO NOME_COMUM
- VERBO → come
- VERBO → bebe
- NOME_COMUM → vinho
- NOME_COMUM → queijo

Gramáticas Livres de Contexto (cont.)

- Sentença: João come queijo.
- Derivação:

SENTENCA $\rightarrow$ NOME_PROPRIO SINTAGMA_VERBAL
$\rightarrow$ joao SINTAGMA_VERBAL
$\rightarrow$ joao VERBO NOME_COMUM
$\rightarrow$ joao come NOME_COMUM
$\rightarrow$ joao come queijo

Gramáticas Sensíveis ao Contexto

- Produções da forma X $\rightarrow$ y,
  onde o tamanho (número de símbolos) de y é maior ou igual ao tamanho de X.
- Exemplo:
  AB $\rightarrow$ CDE (OK)
  ABC $\rightarrow$ DE (NÃO!)
  xAZ $\rightarrow$ xyz

Duas etapas da compreensão

- Reconhecimento: Dada uma seqüência de palavras, decidir se a seqüência é uma "sentença" da gramática
- Análise sintática (parsing): construir a árvore de derivação se a seqüência for uma "sentença" da gramática

Exemplo de Gramática

- “a menina viu o menino”
- “a menina viu o menino com a bola”
- “a menina viu o menino com o telescópio”

Gramática

- Essência da linguagem

  SENTENCA $\rightarrow$ SINTAGMA_NOMINAL
  SINTAGMA_VERBAL
  SINTAGMA_NOMINAL $\rightarrow$ ARTIGO SUBSTANTIVO
  SINTAGMA_VERBAL $\rightarrow$ VERBO SINTAGMA_NOMINAL

Léxico

- Complementa a gramática e expressa o conhecimento da linguagem.
- Léxico:
  ARTIGO $\rightarrow$ o
  ARTIGO $\rightarrow$ a
  SUBSTANTIVO $\rightarrow$ menina
  SUBSTANTIVO $\rightarrow$ menino
  SUBSTANTIVO $\rightarrow$ telescopio
  SUBSTANTIVO $\rightarrow$ bola
  VERBO $\rightarrow$ viu

Gramática abreviada

- Gramática

  S $\rightarrow$ SN SV
  SN $\rightarrow$ ART SUBST
  SV $\rightarrow$ VERBO SN

Léxico abreviado

- Léxico:

  ART $\rightarrow$ o

  ART $\rightarrow$ a

  SUBST $\rightarrow$ menina

  SUBST $\rightarrow$ menino

  SUBST $\rightarrow$ telescopio

  SUBST $\rightarrow$ bola

  VERBO $\rightarrow$ viu

Processamento Descendente (Top-Down)

- Parte-se de hipóteses, convertendo-as en sub-hipóteses até que todas as sub-hipóteses tenham sido verificadas.
- Exemplo de análise top-down da sentença "a menina viu o menino".

  1) hipótese: S

  ("a menina viu o menino" é uma sentença)

  2) procuro produções do tipo

  $S \rightarrow \alpha$

  $S \rightarrow SN\ SV$

3) hipótese: SN SV

4) procuro produções do tipo
SN $\rightarrow \alpha$
SN $\rightarrow$ ART SUBST

5) hipótese: ART SUBST SV

6) procuro produções do tipo
ART $\rightarrow \alpha$
ART $\rightarrow$ o
ART $\rightarrow$ a

7) hipótese: o SUBST SV

8) verifico se “o” corresponde à primeira palavra da sentença: FALSO!

9) retorno ao último ponto de escolha (situação 6), escolhendo agora
ART $\rightarrow$ a

10) hipótese: a SUBST SV

11) verifico que a primeira palavra é um “a”

12) procuro produções do tipo SUBST $\rightarrow \alpha$
SUBST $\rightarrow$ menina (ESCOLHO)
SUBST $\rightarrow$ menino
SUBST $\rightarrow$ telescopio
SUBST $\rightarrow$ bola

13) hipótese: a menina SV

14) procuro produções do tipo
SV $\rightarrow \alpha$
SV $\rightarrow$ V SN (ESCOLHO)

15) procuro produções do tipo
V $\rightarrow \alpha$
V $\rightarrow$ viu

16) hipótese: a menina viu SN

17) procuro produções do tipo
SN $\rightarrow \alpha$
SN $\rightarrow$ ART SUBST

18) hipótese: a menina viu ART SUBST

19) procuro procuções do tipo
ART $\rightarrow \alpha$
ART $\rightarrow$ o (ESCOLHO)
ART $\rightarrow$ a

20) hipótese: a menina viu o SUBST

21) procuro produções do tipo
SUBST $\rightarrow \alpha$
SUBST $\rightarrow$ menina
SUBST $\rightarrow$ menino (ESCOLHO)

22) hipótese: "a menina viu o menino" PROVADA!

- Além do princípio top-down: os símbolos não terminais a ser substituídos foram escolhidos da esquerda para a direita, e a busca foi em profundidade.

Processamento Ascendente (Bottom-up)

- Parte-se dos dados
- Exemplo de análise bottom-up da sentença “a menina viu o menino”.

1) pegue o primeiro termo da seqüência e procure uma produção do tipo
X $\rightarrow$ a
ART $\rightarrow$ a

2) substitua o não terminal na seqüência
S: ART menina viu o menino
S: ART menina viu o menino

3) pegue o primeiro termo da seqüência e procure por produções do tipo
X $\rightarrow$ ART (NÃO ACHO)

4) pegue os dois termos da seqüência e procure por produções do tipo
X $\rightarrow$ ART menina (NÃO ACHO)

5) pegue os três termos da seqüência e procure por produções do tipo
X $\rightarrow$ ART menina viu (NÃO ACHO)

6) pegue os quatro termos da seqüência e procure por produções do tipo
X $\rightarrow$ ART menina viu o (NÃO ACHO)

7) pegue os cinco termos da seqüência e procure por produções do tipo
X $\rightarrow$ ART menina viu o menino (NÃO ACHO)

8) pegue o segundo termo da seqüência e procure por produções do tipo
X $\rightarrow$ menina
SUBST $\rightarrow$ menina
S: ART SUBST viu o menino

9) pegue o primeiro termo da seqüência e procure por produções do tipo
X $\rightarrow$ ART (NÃO ACHO)

10) pegue os dois termos da seqüência e procure por produções do tipo
X $\rightarrow$ ART SUBST
SN $\rightarrow$ ART SUBST

S: SN viu o menino

11) pegue o primeiro termo da seqüência e procure por produções do tipo
X $\rightarrow$ SN (NÃO ACHO)

12) pego os dois primeiros termos da seqüência e procuro por produções do tipo
X $\rightarrow$ SN viu (NÃO ACHO)
X $\rightarrow$ SN viu o (NÃO ACHO)
X $\rightarrow$ SN viu o menino (NÃO ACHO)

13) pegue o segundo termo da seqüência e procure por produções do tipo

X $\rightarrow$ viu

V $\rightarrow$ viu

S: SN V o menino

14) procure por produções do tipo

X $\rightarrow$ SN

X $\rightarrow$ SN V

X $\rightarrow$ SN V o

X $\rightarrow$ SN V o menino

X $\rightarrow$ V

X $\rightarrow$ V o

X $\rightarrow$ V o menino

X $\rightarrow$ o

ART $\rightarrow$ o

S: SN V ART menino

15) procure por produções do tipo
X → SN
X → SN V
X → SN V ART
X → SN V ART menino
X → V
X → V ART
X → V ART menino
X → ART
X → ART menino
X → menino
SUBST → menino

S: SN V ART SUBST

16) procure por produções do tipo
X $\rightarrow$ SN
X $\rightarrow$ SN V
X $\rightarrow$ SN V ART
X $\rightarrow$ SN V ART SUBST
X $\rightarrow$ V
X $\rightarrow$ V ART
X $\rightarrow$ V ART SUBST
X $\rightarrow$ ART
X $\rightarrow$ ART SUBST
SN $\rightarrow$ ART SUBST
S: SN V SN

S: SN V SN

17) procure por produções do tipo
X $\rightarrow$ SN
X $\rightarrow$ SN V
X $\rightarrow$ SN V SN
X $\rightarrow$ V
X $\rightarrow$ V SN
SV $\rightarrow$ V SN

S: SN SV

S: SN SV

18) procure por produções do tipo
X $\rightarrow$ SN SV
S $\rightarrow$ S (PROVADO!)

- Princípios usados neste exemplo: bottom-up, da esquerda para a direita
- bottom-up: X (obter) $\rightarrow \alpha$ (dado)
  É mais raro de ter alternativas
- top-down: X (dado) $\rightarrow \alpha$ (obter)
  Há mais alternativas

Abordagem Híbrida

- Pode-se combinar as abordagens bottom-up e top-down
- Bottom-up para converter palavras em categorias
  S: a menina viu o menino
  S: ART SUBST V ART SUBST
  e depois top-down

- Bottom-up direcionado

  S: ART menina viu o menino

  em lugar de procurar

  X $\rightarrow$ ART

  procurar

  X $\rightarrow$ ART Y (hipótese)

  X $\rightarrow$ ART SUBST

  SN $\rightarrow$ ART SUBST

  X $\rightarrow$ SN Y

Alguns Predicados em Prolog

- membro(X,[X|_]).
- membro(X,[_|Y] :- membro(X,Y).
- appendice([],L,L).
- appendice([X|L1],L2,[X|L3]) :- appendice(L1,L2,L3).

```
?- ['gram2.pl'].
% gram2.pl compiled 0.00 sec, -4 bytes

Yes
?- trace.

Yes
[trace]  ?- appendice([a,b],[c,d],X).
   Call: (8) appendice([a, b], [c, d], _G329) ?
   Call: (9) appendice([b], [c, d], _G407) ?
   Call: (10) appendice([], [c, d], _G410) ?
   Exit: (10) appendice([], [c, d], [c, d]) ?
   Exit: (9) appendice([b], [c, d], [b, c, d]) ?
   Exit: (8) appendice([a, b], [c, d], [a, b, c, d]) ?

X = [a, b, c, d]
```

```
[trace]  ?- membro(c,[a,b,c]).
   Call: (8) membro(c, [a, b, c]) ?
   Call: (9) membro(c, [b, c]) ?
   Call: (10) membro(c, [c]) ?
   Exit: (10) membro(c, [c]) ?
   Exit: (9) membro(c, [b, c]) ?
   Exit: (8) membro(c, [a, b, c]) ?
```

Como fazer parsing em Prolog

- Gramática

  s $\rightarrow$ sn, sv.

  sn $\rightarrow$ art, subst.

  sv $\rightarrow$ v, sn.

  art $\rightarrow$ o.

  art $\rightarrow$ a.

  subst $\rightarrow$ menino.

  subst $\rightarrow$ menina.

  v $\rightarrow$ viu.

Gramáticas em Prolog

```
s(X) :- append(Y,Z,X), sn(Y), SV(Z).
sn(X) :- append(Y,Z,X), art(Y), subst(Z).
sv(X) :- append(Y,Z,X), v(Y), sn(Z).
art([o]).
art([a]).
subst([menino]).
subst([menina]).
v([viu]).
```

Execução da Gramática

?- s([o,menino,viu,a,menina]).

1. A meta é s([o,menino,viu,a,menina]).
2. Decomponha a lista em 2 listas, Y e Z. As seguintes decomposições são posssíveis:
   Y=[], Z=[o,menino,viu,a,menina].
   Y=[o], Z=[menino,viu,a,menina].
   Y=[o,menino], Z=[viu,a,menina]).
   Y=[o,menino,viu], Z=[a,menina]).
   Y=[o,menino,viu,a], Z=[menina]
   Y=[o,menino,viu,a,menina], Z=[]

3. Escolha uma possibilidade para Y e veja se Y é um sn.

4. Se Y for um sn, então procure por um sv.
   Se não for, volte ao passo 3 e escolha uma nova possibilidade.
   Y=[], Z=[o,menino,viu,a,menina].
   Y=[o], Z=[menino,viu,a,menina].
   Y=[o,menino], Z=[viu,a,menina].

Trace

```
[trace]  ?- s([o,menino,viu,a,menina]).
   Call: (7) s([o, menino, viu, a, menina]) ?
   Call: (8) append(_L191, _L192, [o, menino, viu,
             a, menina]) ?
   Exit: (8) append([], [o, menino, viu, a, menina],
             [o, menino, viu, a, menina]) ?
   Call: (8) sn([]) ?
   Call: (9) append(_L227, _L228, []) ?
   Exit: (9) append([], [], []) ?
   Call: (9) art([]) ?
   Fail: (9) art([]) ?
   Redo: (9) append(_L227, _L228, []) ?
   Fail: (9) append(_L227, _L228, []) ?
   Fail: (8) sn([]) ?
   Redo: (8) append(_L191, _L192, [o, menino, viu, a,
```

```
          menina]) ?
Exit: (8) append([o], [menino, viu, a, menina],
          [o, menino, viu, a, menina]) ?
Call: (8) sn([o]) ?
Call: (9) append(_L248, _L249, [o]) ?
Exit: (9) append([], [o], [o]) ?
Call: (9) art([]) ?
Fail: (9) art([]) ?
Redo: (9) append(_L248, _L249, [o]) ?
Exit: (9) append([o], [], [o]) ?
Call: (9) art([o]) ?
Exit: (9) art([o]) ?
Call: (9) subst([]) ?
Fail: (9) subst([]) ?
Redo: (9) art([o]) ?
Redo: (9) append([o|_G394], _L249, [o]) ?
```

```
Fail: (9) append(_L248, _L249, [o]) ?
Fail: (8) sn([o]) ?
Redo: (8) append([o|_G391], _L192, [o, menino, viu,
          a, menina]) ?
Exit: (8) append([o, menino], [viu, a, menina],
          [o, menino, viu, a, menina]) ?
Call: (8) sn([o, menino]) ?
Call: (9) append(_L262, _L263, [o, menino]) ?
Exit: (9) append([], [o, menino], [o, menino]) ?
Call: (9) art([]) ?
Fail: (9) art([]) ?
Redo: (9) append(_L262, _L263, [o, menino]) ?
Exit: (9) append([o], [menino], [o, menino]) ?
Call: (9) art([o]) ?
Exit: (9) art([o]) ?
Call: (9) subst([menino]) ?
```

```
Exit: (9) subst([menino]) ?
Exit: (8) sn([o, menino]) ?
Call: (8) sv([viu, a, menina]) ?
Call: (9) append(_L351, _L352, [viu, a, menina]) ?
Exit: (9) append([], [viu, a, menina], [viu, a,
          menina]) ?
Call: (9) v([]) ?
Fail: (9) v([]) ?
Redo: (9) append(_L351, _L352, [viu, a, menina]) ?
Exit: (9) append([viu], [a, menina], [viu, a, menina]) ?
Call: (9) v([viu]) ?
Exit: (9) v([viu]) ?
Call: (9) sn([a, menina]) ?
Call: (10) append(_L408, _L409, [a, menina]) ?
Exit: (10) append([], [a, menina], [a, menina]) ?
Call: (10) art([]) ?
```

```
   Fail: (10) art([]) ?
   Redo: (10) append(_L408, _L409, [a, menina]) ?
   Exit: (10) append([a], [menina], [a, menina]) ?
   Call: (10) art([a]) ?
   Exit: (10) art([a]) ?
   Call: (10) subst([menina]) ?
   Exit: (10) subst([menina]) ?
   Exit: (9) sn([a, menina]) ?
   Exit: (8) sv([viu, a, menina]) ?
   Exit: (7) s([o, menino, viu, a, menina]) ?

Yes
[debug]  ?- s([o,menino,viu,a,menina]).
```

Uma maneira mais direta

- sn(X,Y) é verdade se → existe um sintagma nominal (sn) no começo da seqüência X e a parte que sobra depois do sintagma nominal é Y.
- ?- sn([o,menino,viu,a,menina],[viu,a,menina]).
- ?- sn([o,menino,caiu],[caiu]).

- sn(X,Y) :- art(X,Z), subst(Z,Y).
- art([o|X],X).
- subst([menino|X],X).

Nova Gramática

- ?- s([o,menino,viu,a,menina],[]).

```
s(S0,S) :- sn(S0,S1), sv(S1,S).
sn(S0,S) :- art(S0,S1), subst(S1,S).
sv(S0,S1) :- v(S0,S1), sn(S1,S).
art([o|S],S).
art([a|S],S).
subst([menino|S],S).
subst([menina|S],S).
v([viu|S],S).
```

Notação em Prolog

s → sn, sv.

sn → art, subst.

sv → v, sn.

art → [o].

art → [a].

subst → [menino].

subst → [menina].

v → [viu].

Tratamento de plural

- O menino viu a menina.
- Os meninos viram a menina.
- * Os meninos viu a menina.
- * O menino viram a menina.
- * Os menino viu a menina.
- * O meninos viu a menina.
- * Os menino viram a menina.

Nova gramática

s → s_singular.

s → s_plural.

sn → sn_singular.

sn → sn_plural.

s_singular → sn_singular, sv_singular.

sn_singular → art_singular, subst_singular.

sv_singular → v_singular, sn.

art_singular → [o].

art_singular → [a].

subst_singular → [menino].

subst_singular → [menina].

v_singular → [viu].

Finalmente...

s → s(X).
s(X) → sn(X), sv(X).
sn(X) → art(X), subst(X).
sv(X) → v(X), sn(Y).
art(singular) → [o].
art(plural) → [os].
subst(singular) → [menino].
subst(plural) → [meninos].
v(singular) → [viu].
v(plural) → [viram].

Árvore de Derivação

s(s(SN,SV)) → sn(SN), sv(SV).
sn(sn(ART,SUBST)) → art(ART), subst(SUBST).
sv(sv(VERBO,SN)) → v(VERBO), sn(SN).
art(art(o)) → [o].
art(art(a)) → [a].
subst(subst(menino)) → [menino].
subst(subst(menino)) → [menina].
v(v(viu)) → [viu].

Adicionando Testes Extras

- Convenção: tudo que estiver entre chaves "{" e "}" não deve ser traduzido pela DCG (deve permanecer como Prolog puro).
- Vamos melhorar o dicionário. No momento:

  subst(subst(banana),singular) → [banana].

  é traduzido em Prolog puro para:

  subst(subst(banana),singular),[banana|S],S).

Adicionando Testes Extras (cont.)

- Misturando regras gramaticais e Prolog puro.

- Informações gerais sobre os substantivos em uma regra gramatical e informações sobre quais palavras são substantivos em cláusulas Prolog.

  subst(subst(S),N) → [S], {e_subst(S,N)}.
  e_subst(banana,singular).
  e_subst(bananas, plural).
  e_subst(homem,singular).

Exercício

- Escrever uma gramática que gere duas árvores de derivação para as seguintes sentenças:

  [a)] O menino viu a menina com a bola.

  [b)] O menino viu a menina com o telescópio.

Exemplos a ser tratados no primeiro projeto

```
s([o,menino,que,viu,a,menina,caiu],[]).

Yes
?- s([o,menino,que,a,menina,viu,caiu],[]).

Yes
?- s([o,menino,que,a,menina,que,viu,o,cachorro,
      beijou,caiu],[]).
```