Tópicos Especias em Sistemas Inteligentes II

Aula 2: PLN - Introdução

João C. P. da Silva

August 26, 2016

Introdução

- **Forma Linguística**
  - *Generative Syntax - Noam Chomsky (1957)* : regras que geram estruturas sintáticas.
- **Significado e Conteúdo**
  - como colocar "conteúdo" em forma linguística?
  - como extrair "conteúdo" da forma linguística?
- **Language Input**: Reconhecimento de Fala, Compreensão de Linguagem
- **Language Output**: Síntese de Fala, Geração de Linguagem Natural

Introdução

Conhecimento Envolvido

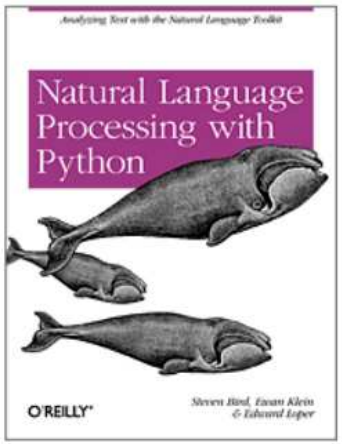
- **Fonética e Fonologia**: como as palavras são pronunciadas em termos de sequência de sons e como cada um destes sons são percebidos acusticamente.
- **Morfologia**: como as palavras se dividem em partes que carregam significados (singular / plural).
- **Sintaxe**
- **Semântica**: significado das palavras.
- **Composição Semântica**: Western Europe, Eastern Europe,...
- **Coreferência**

Aplicações

- Tradução de Máquina
- Web-based Question Answering
- Extração de Informação
- Word Sense Disambiguation

NLTK

- Conjunto de bibliotecas e programas para processamento de linguagem simbólica e análise estatística.
- Bem documentada e de fácil utilização. Considerada uma das melhores ferramentas livres para PLN.
- Site: `http://www.nltk.org/`

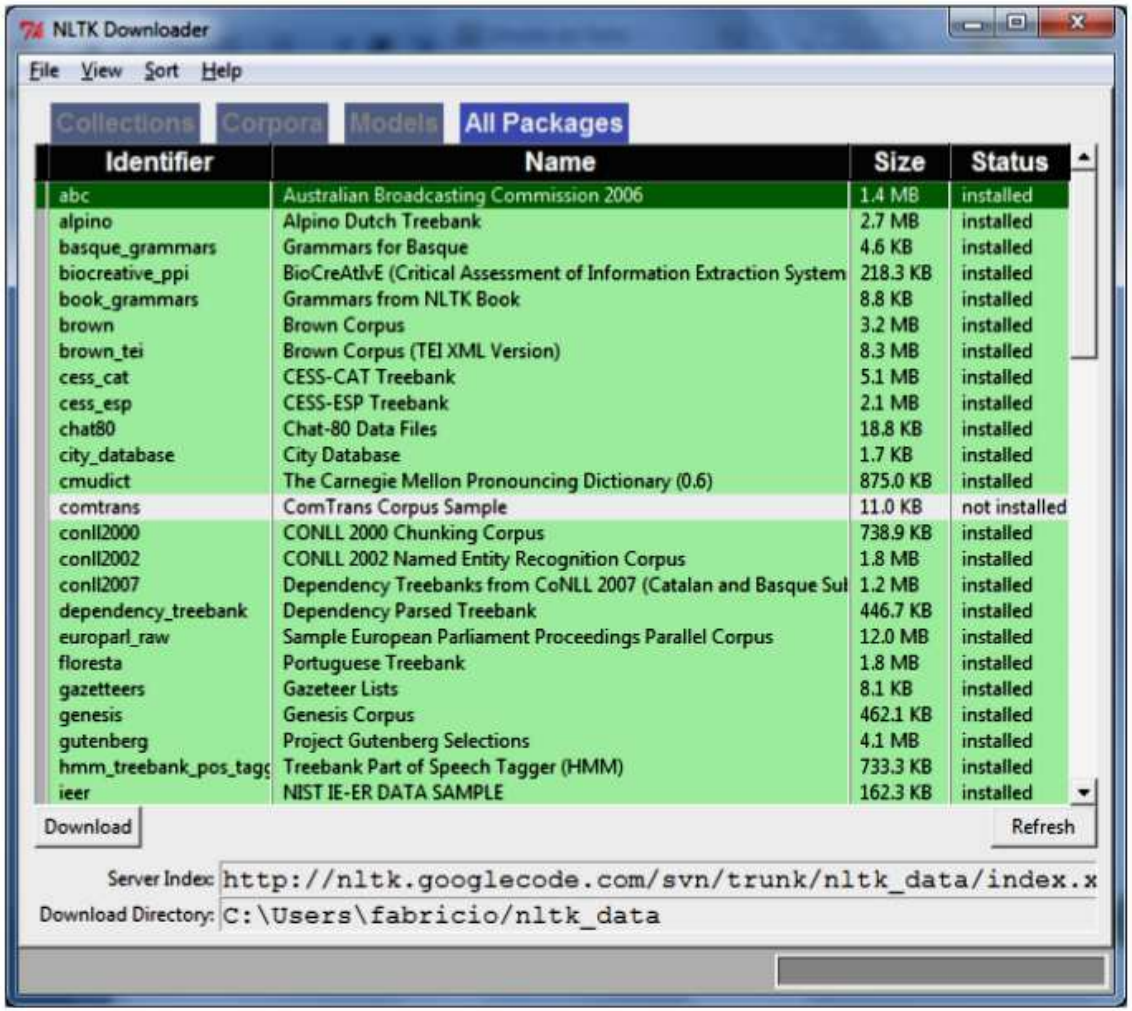


NLTK

Como Instalar

- Necessário ter Python 2.6 ou 2.7 instalado e a biblioteca PyYAML.
- Para geração de gráficos e processamento estatístico é importante ter as bibliotecas NUMPY e MATPLOTLIB instaladas.
- No site da biblioteca tem todos os links dos instaladores.
- Abra o Idle e digite: import nlkt
- Depois digite nltk.download() para aparecer a tela de instalação de adicionais.

NLTK


NLTK Downloader
File View Sort Help
Collections Corpora Models All Packages
Identifier Name Size Status
abc Australian Broadcasting Commission 2006 1.4 MB installed
alpino Alpino Dutch Treebank 2.7 MB installed
basque_grammars Grammars for Basque 4.6 KB installed
biocreative_ppi BioCreAtIvE (Critical Assessment of Information Extraction System 218.3 KB installed
book_grammars Grammars from NLTK Book 8.8 KB installed
brown Brown Corpus 3.2 MB installed
brown_tei Brown Corpus (TEI XML Version) 8.3 MB installed
cess_cat CESS-CAT Treebank 5.1 MB installed
cess_esp CESS-ESP Treebank 2.1 MB installed
chat80 Chat-80 Data Files 18.8 KB installed
city_database City Database 1.7 KB installed
cmudict The Carnegie Mellon Pronouncing Dictionary (0.6) 875.0 KB installed
comtrans ComTrans Corpus Sample 11.0 KB not installed
conll2000 CONLL 2000 Chunking Corpus 738.9 KB installed
conll2002 CONLL 2002 Named Entity Recognition Corpus 1.8 MB installed
conll2007 Dependency Treebanks from CoNLL 2007 (Catalan and Basque Sul 1.2 MB installed
dependency_treebank Dependency Parsed Treebank 446.7 KB installed
europarl_raw Sample European Parliament Proceedings Parallel Corpus 12.0 MB installed
floresta Portuguese Treebank 1.8 MB installed
gazetteers Gazeteer Lists 8.1 KB installed
genesis Genesis Corpus 462.1 KB installed
gutenberg Project Gutenberg Selections 4.1 MB installed
hmm_treebank_pos_tagg Treebank Part of Speech Tagger (HMM) 733.3 KB installed
ieer NIST IE-ER DATA SAMPLE 162.3 KB installed
Download
Refresh
Server Index: http://nltk.googlecode.com/svn/trunk/nltk_data/index.x
Download Directory: C:\Users\fabricio/nltk_data

The Stanford NLP Group

http://nlp.stanford.edu/

Processamento Linguagem Natural

Quais as tarefas envolvidas no Processamento de Linguagem Natural ?

Barack Hussein Obama II is the 44th and current President of the United States. He is the first African American to hold the office.

Born in Honolulu, Hawaii, Obama is a graduate of Columbia University and Harvard Law School, where he was president of the Harvard Law Review. He was a community organizer in Chicago before earning his law degree. He worked as a civil rights attorney in Chicago and taught constitutional law at the University of Chicago Law School from 1992 to 2004. He served three terms representing the 13th District in the Illinois Senate from 1997 to 2004, running unsuccessfully for the United States House of Representatives in 2000.

Usando NLTK

- **Tipos de Arquivos**: html, pdf, doc, $\cdots$
- Vamos trabalhar com arquivos do tipo txt.

```
>>> import nltk
>>> arquivo = open('obama') # abrindo arquivo
>>> raw = arquivo.read()  # lendo arquivo
>>> print raw
Barack Hussein Obama II (born August 4, 1961) is the 44th and
current President of the United States. He is the first
African American to hold the office.
Born in Honolulu, Hawaii, Obama is a graduate of Columbia
University and Harvard Law School, where he was president of
the Harvard Law Review. He was a community organizer in
Chicago before earning his law degree. He worked as a civil
rights attorney in Chicago and taught constitutional law at
the University of Chicago Law School from 1992 to 2004. He
served three terms representing the 13th District in the
Illinois Senate from 1997 to 2004, running unsuccessfully
for the United States House of Representatives in 2000.
```

Pré-Processamento

- **Tokenization**
- **Normalization**
- **Stemming / Lemmatization**
- **Part-of-Speech (POS) tagging**
- **Coreference Resolution**

Tokenization - NLTK

- Tem como objetivo detectar os limites de palavras e sentenças.

```
>>> raw = arquivo.read()  # lendo arquivo

>>> tokens = nltk.word_tokenize(raw) #tokenizando o texto

['Barack', 'Hussein', 'Obama', 'II', '(', 'born', 'August', '4',
',','1961', ')', 'is', 'the', '44th', 'and', 'current',
'President','of', 'the', 'United', 'States', '.', 'He', 'is',
'the', 'first','African', 'American', 'to', 'hold', 'the',
'office', '.', 'Born', 'in', 'Honolulu', ',', 'Hawaii', ',',
'Obama', 'is', 'a', 'graduate', 'of', 'Columbia', 'University',
'and', 'Harvard', 'Law', 'School', ',', 'where', 'he', 'was',
'president', 'of', 'the', 'Harvard', 'Law', 'Review', '.', 'He',
'was', 'a', 'community', 'organizer', 'in', 'Chicago', 'before',
'earning', 'his', 'law', 'degree', '.', 'He', 'worked', 'as', 'a',
'civil', 'rights', 'attorney', 'in', 'Chicago', 'and', 'taught',
'constitutional', 'law', 'at', 'the', 'University', 'of', ...
```

Tokenization - NLTK

Como definir o que é um token?

- nltk.word_tokenize("Barack Hussein Obama II") : ['Barack', 'Hussein', 'Obama', 'II']
- nltk.word_tokenize("Hewlett-Packard") : ['Hewlett-Packard']
- nltk.word_tokenize("c.p.f. 910.921.872-23") : ['c.p.f', '.', '910.921.872-23']
- nltk.word_tokenize("c. p. f. 910.921.872-23") : ['c.', 'p.', 'f.', '910.921.872-23']
- nltk.word_tokenize("I wouldn't be here if, time and again, the torch had not been passed to a new generation.") :

  ['I', 'would', "n't", 'be', 'here', 'if', ',', 'time', 'and', 'again', ',', 'the', 'torch', 'had', 'not', 'been', 'passed', 'to', 'a', 'new', 'generation', '.'].

Normalização

- Permite a definição de classes de equivalência dos termos.
  - UFRJ x U.F.R.J.
  - UFRJ x ufrj
  - US x us
  - 12/05/14 x 12 de maio de 2014
  - from 1997 to 2002 x (??/??/1997 - ??/??/2002)

Stemming / Lemmatization

Aplicados como um passo de normalização, mapeia variantes morfológicas a seu correspondente básico.

- foxes $\Rightarrow$ fox
- going $\Rightarrow$ go
- car, cars, car's, cars' $\Rightarrow$ car
- am, are, is $\Rightarrow$ be
- the boy's cars are different colors $\Rightarrow$ the boy car be differ color

Stemming / Lemmatization

Afixos : morfemas responsáveis por transformar uma palavra primitiva em uma outra, derivada desta.

- **Stemming**: processo heurístico que normalmente corta o final da palavra (remoção dos afixos de derivação).

  foxes ⇒ fox ; going ⇒ go ; car, cars, car's, cars' ⇒ car ; saw ⇒ s

- **Lemmatization**: utiliza análise morforlógica para identificar a forma básica da palavra que está no dicionário (lemma).

  am, are, is ⇒ be ; saw ⇒ see (verb) ou saw (noun)

Observação: *stemming* tem a ver com derivação (uma palavra de outra) e *lemmatization* tem a ver com flexão (verbal por exemplo)

Stemming - NLTK

- raw = "DENNIS: Listen, strange women lying in ponds distributing swords is no basis for a system of government. Supreme executive power derives from a mandate from the masses, not from some farcical aquatic ceremony."

- tokens = nltk.word_tokenize(raw)

  tokens = ['DENNIS', ':', 'Listen', ',', 'strange', 'women', 'lying', 'in', 'ponds', 'distributing', 'swords', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a', 'mandate', 'from', 'the', 'masses', ',', 'not', 'from', 'some', 'farcical', 'aquatic', 'ceremony', '.']

- **Porter Stemmer:**
  - porter = nltk.PorterStemmer()
  - [porter.stem(t) for t in tokens]
  - ['DENNI', ':', 'Listen', ',', 'strang', 'women', 'lie', 'in', 'pond', 'distribut', 'sword', 'is', 'no', 'basi', 'for', 'a', 'system', 'of', 'government.', 'Suprem', 'execut', 'power', 'deriv', 'from', 'a', 'mandat', 'from', 'the', 'mass', ',', 'not', 'from', 'some', 'farcic', 'aquat', 'ceremoni', '.']

Stemming - Algoritmo de Porter

- Mais usado em inglês
- Consiste de reduções dos seguintes tipos:

  Passo 1

Regra	Exemplo
SSES $\rightarrow$ SS	caresses $\rightarrow$ caress
IES $\rightarrow$ I	ponies $\rightarrow$ poni
SS $\rightarrow$ SS	caress $\rightarrow$ caress
S $\rightarrow$	cats $\rightarrow$ cat
(m $>$ 0) EED $\rightarrow$ EE	feed $\rightarrow$ fee ; agreed $\rightarrow$ agree
(*v*) ED $\rightarrow$	plastered $\rightarrow$ plaster
(*v*) ING $\rightarrow$	motoring $\rightarrow$ motor ; sing $\rightarrow$ sing

Stemming - Algoritmo de Porter

- Mais usado em inglês
- Consiste de reduções dos seguintes tipos:

Passo 1

Regra	Exemplo
AT $\rightarrow$ ATE	conflat(ed) $\rightarrow$ conflate
BL $\rightarrow$ BLE	troubl(ed) $\rightarrow$ trouble
IZ $\rightarrow$ IZE	siz(ed) $\rightarrow$ size
...	...
(*v*) Y $\rightarrow$ I	happy $\rightarrow$ happi ; sky $\rightarrow$ ski

Stemming - Algoritmo de Porter

- Mais usado em inglês
- Consiste de reduções dos seguintes tipos:

  Passos 2, 3, 4 e 5

Regra	**Exemplo**
(m $>$ 0) ATIONAL $\rightarrow$ ATE	relational $\rightarrow$ relate
...	...
(m $>$ 0) ALIZE $\rightarrow$ AL	formalize $\rightarrow$ formal
...	...
(m $>$ 1) ISM $\rightarrow$	communism $\rightarrow$ commun
...	...
(m $>$ 1) E $\rightarrow$	probate $\rightarrow$ probat

Stemming - NLTK

- **Lancaster Stemmer:**
  - tokens = ['DENNIS', ':', 'Listen', ',', 'strange', 'women', 'lying', 'in', 'ponds', 'distributing', 'swords', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a', 'mandate', 'from', 'the', 'masses', ',', 'not', 'from', 'some', 'farcical', 'aquatic', 'ceremony', '.']

    lancaster = nltk.LancasterStemmer()

    [lancaster.stem(t) for t in tokens]

    ['den', ':', 'list', ',', 'strange', 'wom', 'lying', 'in', 'pond', 'distribut', 'sword', 'is', 'no', 'bas', 'for', 'a', 'system', 'of', 'government.', 'suprem', 'execut', 'pow', 'der', 'from', 'a', 'mand', 'from', 'the', 'mass', ',', 'not', 'from', 'som', 'farc', 'aqu', 'ceremony', '.']

Lemmatization - NLTK

- tokens = ['DENNIS', ':', 'Listen', ',', 'strange', 'women', 'lying', 'in', 'ponds', 'distributing', 'swords', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a', 'mandate', 'from', 'the', 'masses', ',', 'not', 'from', 'some', 'farcical', 'aquatic', 'ceremony', '.']

  wnl = nltk.WordNetLemmatizer()

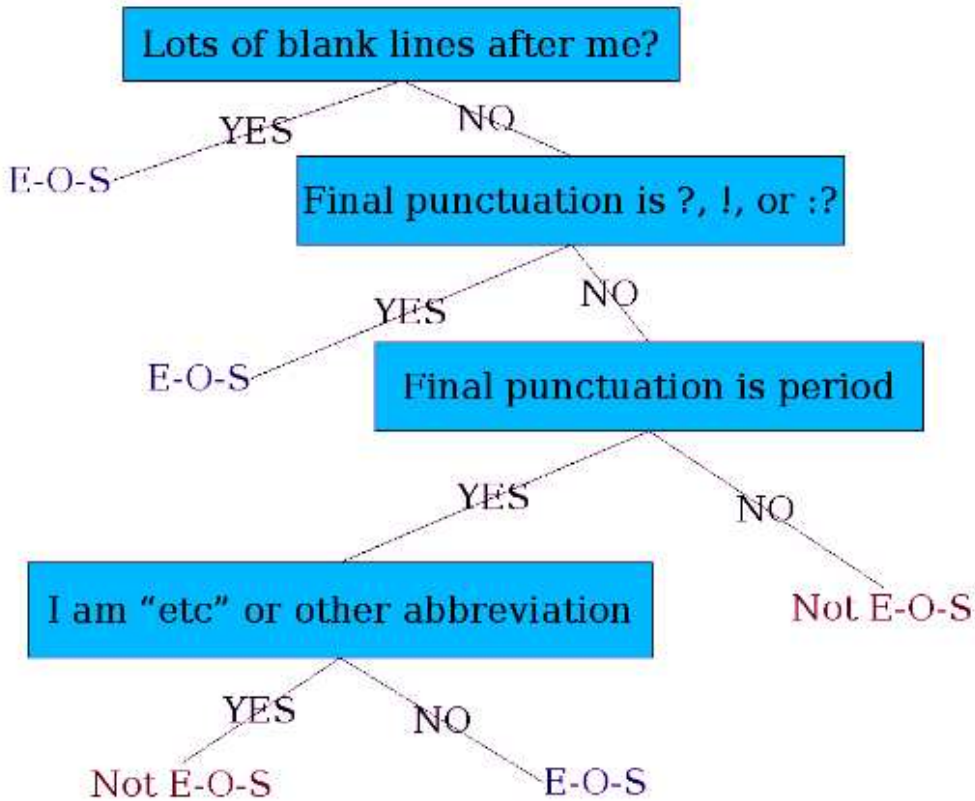
  [wnl.lemmatize(t) for t in tokens]

  ['DENNIS', ':', 'Listen', ',', 'strange', 'woman', 'lying', 'in', 'pond', 'distributing', 'sword', 'is', 'no', 'basis', 'for', 'a', 'system', 'of', 'government.', 'Supreme', 'executive', 'power', 'derives', 'from', 'a', 'mandate', 'from', 'the', 'mass', ',', 'not', 'from', 'some', 'farcical', 'aquatic', 'ceremony', '.']

Segmentação de Sentença

- Determinar os limites de uma sentença - . , ! , ?
  - This is a sentence.
  - Prof. , Sr. , $\cdots$
  - 3.14159 , 3!
- Utilizar um classificador para definir se temos ou não uma sentença.

Segmentação de Sentença - Árvore de Decisão



Segmentação de Sentença - NLTK

- **Punkt sentence segmenter**: tokenizador pre-treinado para inglês

  sent_tokenizer=nltk.data.load('tokenizers/punkt/english.pickle')

- raw = "In most countries of the world, the information revolution has altered many aspects of life significantly: commerce, employment, medicine, security, transportation, entertainment, and on and on. Consequently, information and communication technology (ICT) has affected, in both good ways and bad ways, community life, family life, human relationships, education, careers, freedom, and democracy (to name just a few examples). Computer and information ethics, in the present essay, is understood as that branch of applied ethics which studies and analyzes such social and ethical impacts of ICT."

- sentences = sent_tokenizer.tokenize(raw)

- print(sentences)

- ['"**In most countries of the world, the information revolution has altered many aspects of life significantly: commerce, employment, medicine, security, transportation, entertainment, and on and on.**","Consequently, information and communication technology (ICT) has affected, in both good ways and bad ways, community life, family life, human relationships, education, careers, freedom, and democracy (to name just a few examples).", "**Computer and information ethics, in the present essay, is understood as that branch of applied ethics which studies and analyzes such social and ethical impacts of ICT.**"]

Expressão Regular

Permite identificar padrões em textos.

- Modelo simples que é bastante utilizado no processamento de textos
- Recuperação de informação
- Processador de texto
- Usado como features na construção de classificadores (aprendizado de máquina)
- Úteis para capturar generalizações

Expressão Regular - Aplicação

ELIZA (Weizenbaum - 1966)

USER: Men are all alike.

ELISA: In What way.

USER: They' re always bugging us about something or other.

ELISA: Can you think of a specific example.

USER: Well, my boyfriend made me come here.

ELISA: Your boyfriend made you come here.

USER: He says I'm depressed much of the time.

ELISA:I am sorry to hear you are depressed.

USER: You are afraid of me.

ELISA: Does it please you to believe I am afraid of you.

Expressão Regular

- Importando módulo de expressões regulares: import re
- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.e.', w)]

Expressão Regular

- Importando módulo de expressões regulares: import re
- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.e.', w)]

    ['countries', 'revolution', 'altered', 'aspects', 'commerce', 'employment', 'medicine', 'security', 'entertainment', 'Consequently', 'technology', 'affected', 'relationships', 'careers', 'freedom', 'democracy', 'few', 'examples', 'Computer', 'present', 'understood', 'applied', 'studies', 'analyzes']
  - [w for w in tokens if re.search(r'^.e.$', w)]

Expressão Regular

- Importando módulo de expressões regulares: import re
- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.e.', w)]

    ['countries', 'revolution', 'altered', 'aspects', 'commerce', 'employment', 'medicine', 'security', 'entertainment', 'Consequently', 'technology', 'affected', 'relationships', 'careers', 'freedom', 'democracy', 'few', 'examples', 'Computer', 'present', 'understood', 'applied', 'studies', 'analyzes']
  - [w for w in tokens if re.search(r'^.e.$', w)]

    ['few']
  - Qualquer símbolo (.), início de palavra (^) e fim de palavra ($)

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.ion', w)]

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.ion', w)]

    ['information', 'revolution', 'transportation', 'information', 'communication', 'relationships', 'education', 'information']
  - [w for w in tokens if re.search(r'.ion$', w)]

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'.ion', w)]

    ['information', 'revolution', 'transportation', 'information', 'communication', 'relationships', 'education', 'information']
  - [w for w in tokens if re.search(r'.ion$', w)]

    ['information', 'revolution', 'transportation', 'information', 'communication', 'education', 'information']

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['email','e-mail']
  - [w for w in tokens if re.search(r'^e-?mail$', w)]

    [['email', 'e-mail']]
  - [w for w in tokens if re.search(r'^e-mail$', w)]

    ['e-mail']
  - O símbolo anterior a ocorrência de ? é opcional.

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'^[wr]', w)]

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'^[wr]', w)]

    ['world', 'revolution', 'ways', 'ways', 'relationships', 'which']
  - [w for w in tokens if re.search(r'^[wr].*[dh]$', w)]

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'^[wr]', w)]

    ['world', 'revolution', 'ways', 'ways', 'relationships', 'which']
  - [w for w in tokens if re.search(r'^[wr].*[dh]$', w)]

    ['world','which']
  - [w for w in tokens if re.search(r'^[A-Z][A-Z][A-Z]$', w)]

Expressão Regular

- Busca de padrões - **re.search(r'p', s)** verifica se o padrão **'p'** ocorre na string **s**.
  - tokens = ['In', 'most', 'countries', 'of', 'the', 'world', ',', 'the', 'information', 'revolution', 'has', 'altered', 'many', 'aspects', 'of', 'life', 'significantly', ':', 'commerce', ',', 'employment', ',', 'medicine', ',', 'security', ',', 'transportation', ',', 'entertainment', ',', 'and', 'on', 'and', 'on', '.', 'Consequently', ',', 'information', 'and', 'communication', 'technology', '(', 'ICT', ')', 'has', 'affected', ',', 'in', 'both', 'good', 'ways', 'and', 'bad', 'ways', ',', 'community', 'life', ',', 'family', 'life', ',', 'human', 'relationships', ',', 'education', ',', 'careers', ',', 'freedom', ',', 'and', 'democracy', '(', 'to', 'name', 'just', 'a', 'few', 'examples', ')', '.', 'Computer', 'and', 'information', 'ethics', ',', 'in', 'the', 'present', 'essay', ',', 'is', 'understood', 'as', 'that', 'branch', 'of', 'applied', 'ethics', 'which', 'studies', 'and', 'analyzes', 'such', 'social', 'and', 'ethical', 'impacts', 'of', 'ICT', '.']
  - [w for w in tokens if re.search(r'^[wr]', w)]

    ['world', 'revolution', 'ways', 'ways', 'relationships', 'which']
  - [w for w in tokens if re.search(r'^[wr].*[dh]$', w)]

    ['world','which']
  - [w for w in tokens if re.search(r'^[A-Z][A-Z][A-Z]$', w)]

    ['ICT','ICT']
- **Fecho de Kleene**: + e *

Expressão Regular

Table 3-3. Basic regular expression metacharacters, including wildcards, ranges, and closures

Operator	Behavior
`.`	Wildcard, matches any character
`^abc`	Matches some pattern *abc* at the start of a string
`abc$`	Matches some pattern *abc* at the end of a string
`[abc]`	Matches one of a set of characters
`[A-Z0-9]`	Matches one of a range of characters
`ed|ing|s`	Matches one of the specified strings (disjunction)
`*`	Zero or more of previous item, e.g., a*, `[a-z]*` (also known as *Kleene Closure*)
`+`	One or more of previous item, e.g., a+, `[a-z]+`
`?`	Zero or one of the previous item (i.e., optional), e.g., a?, `[a-z]?`
`{n}`	Exactly *n* repeats where *n* is a non-negative integer
`{n,}`	At least *n* repeats
`{,n}`	No more than *n* repeats
`{m,n}`	At least *m* and no more than *n* repeats
`a(b|c)+`	Parentheses that indicate the scope of the operators

Expressão Regular

Operadores	Descrição
\D	qualquer não dígitos
\w	qualquer caracter de palavra
\W	qualquer não-caracter de palavra
[∧aeiou]	qualquer símbolo que não seja vogal minúscula
[∧0-9]	qualquer símbolo que não seja dígito

Expressão Regular

- **re.findall()**: encontra todas as ocorrências (sem overlapping) que casam com a expressão regular dada.
  - >>> word = 'supercalifragilisticexpialidocious'
  - >>> re.findall(r'[aeiou]', word)
  - ['u', 'e', 'a', 'i', 'a', 'i', 'i', 'i', 'e', 'i', 'a', 'i', 'o', 'i', 'o', 'u']
  - >>> len(re.findall(r'[aeiou]', word))
  - 16

Part of Speech - (POS) tagging

- **Objetivo**: atribuir a cada token sua categoria sintática.
  - **nouns** : *people, animals, concepts, things.*
  - **verbs** : *events, states, actions, beliefs, attitudes, etc.*
  - **adjectives** : *describe properties of nouns.*
  - **proper nouns** : *individuals.*
  - **adverbs** : *description of manner.*
  - **prepositional phrases** : *spatio-temporal conditions.*
  - **pronouns** : *act like variables in that they refer to a person or thing that is somehow salient in the discourse context.*
  - **determiners** :*describe the particular reference of a noun.*

Part of Speech - (POS) tagging

As categorias de palavras são sistematicamente relacionadas por *processos morfológicos* como, por exemplo, a forma plural *dogs* formada da forma singular *dog*.
Este tipo de processo permite inferir propriedades sintáticas e semânticas de palavras novas (ainda não conhecidas).

Importância da Morfologia

- Inglês
  - Verbos regulares: 4 formas distintas
  - Verbos irregulares: no máximo 8 formas
- Finlandês
  - Os verbos possuem mais de 10.000 formas.

Part of Speech - (POS) tagging

Tipo de Processos Morfológicos

- **Inflexão**: modificações de uma raíz através de prefixos e sufixos que indicam diferenças gramaticais (singular-plural).

  Não altera significativamente a classe da palavra ou seu significado.

  Todas as formas inflexionais de uma palavra são frequentemente agrupadas como manifestações de um único lexema.

Type of inflection	Instances
number	singular, plural
gender	feminine, masculine, neuter
case	nominative (I,you,he,she,it,we,they), accusative (me, you, him, her, it, us, you, them), genitive (my, 's), dative (Ich schickte **dem** Mann das Buch.)

Part of Speech - (POS) tagging

Tipo de Processos Morfológicos

- **Derivação**: corresponde a uma mudança da categoria sintática e frequentemente envolve uma mudança de signifcado. Exemplos:
  - sufixo -**ly** transforma adjetivo em advérbio : *wide-ly*.
  - sufixo -**en** transforma adjetivos em verbos: *weak-en*, *soft-en*.
  - sufixo -**able** transforma verbos em adjetivos : *understand-able*, *accept-able*.
  - sufixo -**er** transforma verbos em substantivos : *teach-er*, *lead-er*.

Part of Speech - (POS) tagging

Tipo de Processos Morfológicos

- **Composição**: compõem duas ou mais palavras em uma nova palavra.

  Exemplos

  - *tea kettle*
  - *disk drive*
  - *college degree*
  - *mad cow disease*

Part of Speech - (POS) tagging

Palavras que acompanham substantivos (nouns)

- **Determiners**: descreve a referência particular a um substantivo.
- **Articles**: subtipo de determiners. Indica que estamos falando sobre alguém ou sobre alguma coisa que já conhecemos ou podemos determinar.
    - **Exemplos**: a, an, the, this, that.
- **Adjectives**: usados para descrever propriedades de substantivos.
- **Attributive ou Adnominal**: quando modificam um substantivo. Exemplo: many *intelligent* children, a *red* rose.
- **Predicative**: adjetivo usado como complemento do verbo **be**. Exemplo: The rose is *red*, The journey will be *long*.

Part of Speech - (POS) tagging

Modificações morfológicas de adjetivo

- -ly
- **Comparative**: *rich-er*, *trend-ier*
- **Superlative**: *rich-est*, *trend-iest*
- **Periphrastic forms**: formada com palavras auxiliares (*more*, *most*)

Quantificadores

- *all*, *many*, ...
- *some* e *any* são determiners que podem funcionar como quantificadores.
- **wh-determiner**: *what*, *which*
- **possessive wh-pronoun**: *whose*
- **objective wh-pronoun**: *whom*, *which*, *that*
- **nominative wh-pronoun**: *who*, *which*, *that*

Part of Speech - (POS) tagging

Verbos - Usados para descrever:

- **ações**: *She threw the stone*
- **atividades**: *She walked along the river*
- **estados**: *I have $50*

Formas morfológicas (verbos regulares)

- **the root or base form**: *walk*
- **the third singular present tense**: *walks*
- **the gerund and present participle**: *walking*
- **the past tense form and past/passive participle**: *walked*

Part of Speech - (POS) tagging

Verbos Irregulares: possuem diferentes formas para o past tense e past participle. Exemplos: drive - drove - driven e take - took - taken

Feature	Category	Instances
subject	number	singular, plural
subject	person	first (I walk), second (you walk), third (she walks)
tense		present tense, past tense, future tense
aspect		progressive, perfect
mood/modality		possibility, subjunctive, irrealis
participles		present participle (walking), past participle (walked)
voice		active, passive, middle

Table : Features commonly marked on verbs

Part of Speech - (POS) tagging

Adverbs, Prepositions, and Particles

- **Adverbs**: modificam um verbo da mesma forma que adjetivos modificam substantivos. Especificam lugar, tempo, forma ou grau.
  - Alguns advérbios não possuem sufixo *-ly*, como *often*.
  - Alguns advérbios podem também modificar adjetivos

    (*a very unlikely event* - *a shockingly frank exchange*)

    e outros advérbios

    (*She started her career off very impressively*).

Part of Speech - (POS) tagging

Adverbs, Prepositions, and Particles

- **Degree Adverbs ou Qualifiers**: advérbios especializados na função de modificar adjetivos e advérbios e não modificar verbos. Exemplo: *very*.
- **Preposição** : expressam relacionamento espacial.
- **Partículas**: subclasse das preposições que podem ter ligação forte com verbos na formação dos *prepositional verbs*.
- **Prepositional Verb**: combinações de um verbo e uma partícula (em geral uma preposição) que funciona sintática e semanticamente como uma única unidade. Exemplos:
  - The plane took off at Sam.
  - Don't give in to him.
  - It is time to take on new responsibilities.
  - He was put off by so much rudeness.

Part of Speech - (POS) tagging

Coordinating and Subordinating Conjunctions

- **Coordinating Conjunctions**: Coordena duas palavras ou frases de mesma categoria:
  - husband *and* wife [nouns]
  - She bought *or* leased the car. [verbs]
  - the green triangle *and* the blue square [noun phrases]
  - She bought her car, *but* she also considered leasing it. [sentences]

Part of Speech - (POS) tagging

Coordinating and Subordinating Conjunctions

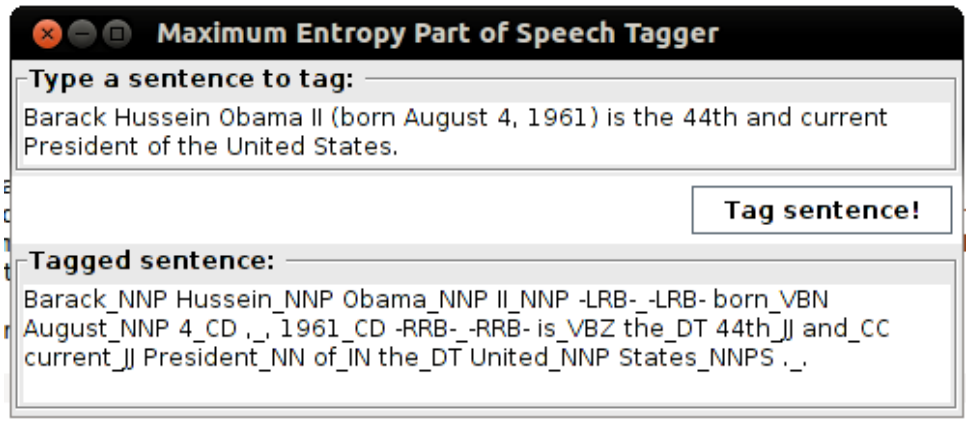
- **Subordinating Conjunctions**: Liga duas sentenças (ou *clauses*):
  - She said *that* he would be late. [proposition]
  - She complained *because* he was late. [reason]
  - I won't wait *if* he is late. [condition]
  - She thanked him *although* he was late. [concession]
  - She left *before* he arrived. [temporal]

Stanford POS-Tagger

"Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States."

Barack_NNP Hussein_NNP Obama_NNP II_NNP -LRB-_-LRB- born_VBN August_NNP 4_CD ,_, 1961_CD -RRB-_-RRB- is_VBZ the_DT 44th_JJ and_CC current_JJ President_NN of_IN the_DT United_NNP States_NNPS ._.

java -jar stanford-postagger.jar



Penn Treebank POS tagset

Corpus formado por mais de 4.5 mihões de palavras

Barack_NNP Hussein_NNP Obama_NNP II_NNP -LRB-_-LRB- born_VBN August_NNP 4_CD ,_, 1961_CD -RRB-_-RRB- is_VBZ the_DT 44th_JJ and_CC current_JJ President_NN of_IN the_DT United_NNP States_NNPS ._.

Sigla	Significado
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
IN	Preposition/subordinating conjunction
JJ	Adjective
NN	Noun, singular or mass
NNP	Proper noun, singular
NNPS	Proper noun, plural
VBN	Verb, past participle
VBZ	Verb, 3rd ps. sing. present

Part of Speech - (POS) tagging - NLTK

- text = nltk.word_tokenize("And now for something completely different")
- nltk.pos_tag(text)
- $[('And','CC'), ('now','RB'), ('for','IN'), ('something','NN'), ('completely','RB'), ('different','JJ')]$
- Para saber o significado das tags: nltk.help.upenn_tagset()

Part of Speech - (POS) tagging

Problema: Determinar qual o POS-tag de uma dada palavra.

- The back door. : The_DT back_JJ door_NN.
- On my back. : On_IN my_PRP$ back_NN.
- Win the voters back. : Win_VB the_DT voters_NNS back_RB.
- Promised to back the bill. : Promised_VBN to_TO back_VB the_DT bill_NN.

Acurácia: 95% a 97%

- muitas palavras não possuem ambiguidade
- baseline : 90%
  - "Taggear" toda palavras com a tag mais frequente
  - "Taggear" palavras desconhecidas como "nouns"

Análise Sintática

- **Chunking**
- **Phrase Structure**
- **Dependency Structure**

Chunking

- **Chunking / Shallow or Partial Parsing**: técnica de processamento superficial (expressões regulares) para agrupar palavras em constituintes sintáticos maiores e com um signifcado.
- **Exemplo**: `the exciting modern art museum`, tem `museum` como constituinte principal e as demais palavras tem a função de restringir seu significado.
- Não identificam relações gramaticais (sujeito, objeto, etc).
- **Ambiguidade Sintática**: *I saw the man on the hill with the telescope.*
- **Tarefa**: Aprendizado de ontologias

Phrase Structure

- The children slept.
  The_DT children_NNS slept_VBD.
- The children ate the cake.
  The_DT children_NNS ate_VBD the_DT cake_NN.
- The students ate the cake of the children in the mountains.
  The_DT students_NNS ate_VBD the_DT cake_NN of_IN the_DT children_NNS in_IN the_DT mountains_NNS.

Phrase Structure

- The_DT children_NNS slept_VBD.
- The_DT children_NNS ate_VBD the_DT cake_NN.
- The_DT students_NNS ate_VBD the_DT cake_NN of_IN the_DT children_NNS in_IN the_DT mountains_NNS.

Phrase Structure

- [The_DT children_NNS] slept_VBD.
- [The_DT children_NNS] ate_VBD [the_DT cake_NN].
- [The_DT students_NNS] ate_VBD [the_DT cake_NN] of_IN [the_DT children_NNS] in_IN [the_DT mountains_NNS].

[DT NN] e [DT NNS] : Agrupar como [NP]

NP é um constituinte (se comporta como uma unidade).

The students ate the cake of the children in the mountains.
The students in the mountains ate the cake of the children.
The students in the ate the cake of the children mountains.

Phrase Structure

- [NP] slept_VBD.
- [NP] ate_VBD [NP].
- [NP] ate_VBD [NP] of_IN [NP] in_IN [NP].

[DT NN] e [DT NNS] : Agrupar como [NP]

Phrase Structure

- [NP] slept_VBD.
- [NP] ate_VBD [NP].
- [NP] ate_VBD [NP] of_IN [NP] in_IN [NP].

Phrase Structure

- [NP] slept_VBD.
- [NP] ate_VBD [NP].
- [NP] ate_VBD [NP] [of_IN [NP]] [in_IN [NP]].

Phrase Structure

- [NP] slept_VBD.
- [NP] ate_VBD [NP].
- [NP] ate_VBD [NP] [of_IN [NP]] [in_IN [NP]].

[IN [NP]] : Agrupar como [PP]

Phrase Structure

- [NP] slept_VBD.
- [NP] ate_VBD [NP].
- [NP] ate_VBD [NP] [PP] [PP].

Phrase Structure

- [NP] [slept_VBD].
- [NP] [ate_VBD [NP]].
- [NP] [[[ate_VBD [NP]][PP]] [PP]].

[[VBD]] e [[VBD] [NP]] : Agrupar como [VP]

[[VP] [PP]] : Agrupar como [VP]

Phrase Structure

- [NP] [VP].
- [NP] [VP].
- [NP] [VP].

[[NP] [VP]] : Agrupar como [S]

Phrase Structure

- **Noun Phrase - NP** : parte da sentença na qual a informação sobre o substantivo (constituinte central) é agrupada.
- **Verb Phrase - VP** : iniciada por um verbo, organiza os elementos da sentença que dependem sintaticamente do verbo.
- **Prepositional Phrase - PP** : iniciado por uma preposição e contém um Noun Phrase.

Phrase Structure

$G = (T, N, S, L, R)$

- T : conjunto de símbolos terminais
- N : conjunto de símbolos não-terminais
- S : símbolo inicial ($S \in N$)
- L : conjunto de produções da forma $X \rightarrow x$ com $X \in N$ e $x \in T$ (lexicon)
- R : conjunto de produções da forma $X \rightarrow \gamma$ com $X \in N$ e $\gamma \in N^*$ (gramática)

Phrase Structure Grammars - Chomsky

$S \rightarrow NP\ VP$

$NP \rightarrow DT\ NNS | DT\ NN \mid NP\ PP$

$VP \rightarrow VP\ PP \mid VBD \mid VBD\ NP$

$PP \rightarrow IN\ NP$

$DT \rightarrow$ the

$NNS \rightarrow$ children | students | mountains

$VBD \rightarrow$ slept | ate | saw

$IN \rightarrow$ in | of | with

$NN \rightarrow$ cake | spoon

Phrase Structure Grammars - Chomsky

Derivações

$S \rightarrow NP\ VP$

$\rightarrow DT\ NNS\ VBD$

$\rightarrow$ the children slept

$S \rightarrow NP\ VP$

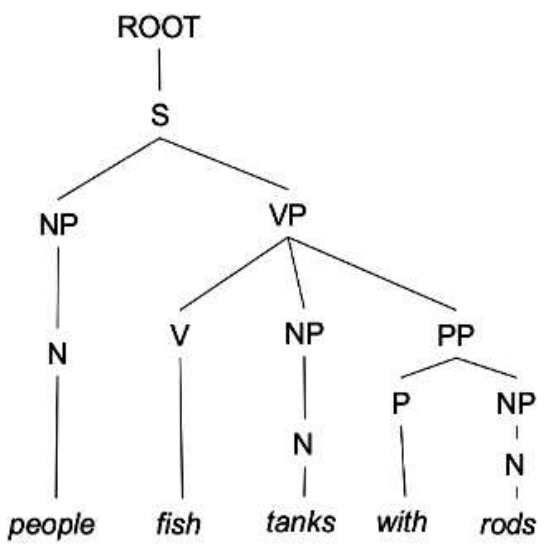
$\rightarrow DT\ NNS\ VBD\ NP$

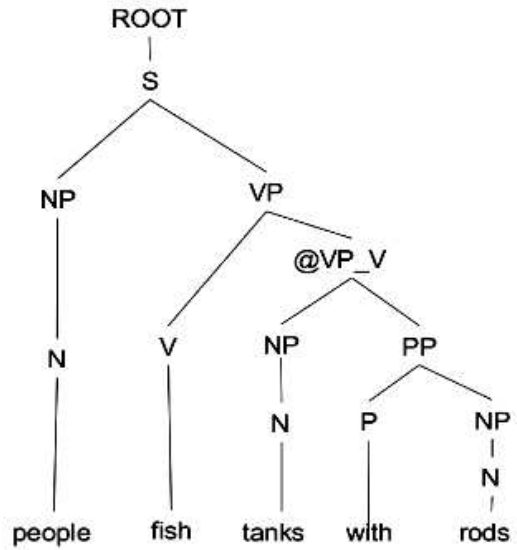
$\rightarrow DT\ NNS\ VBD\ DT\ NN$

$\rightarrow$ the children ate the cake

Phrase Structure Grammars - Chomsky

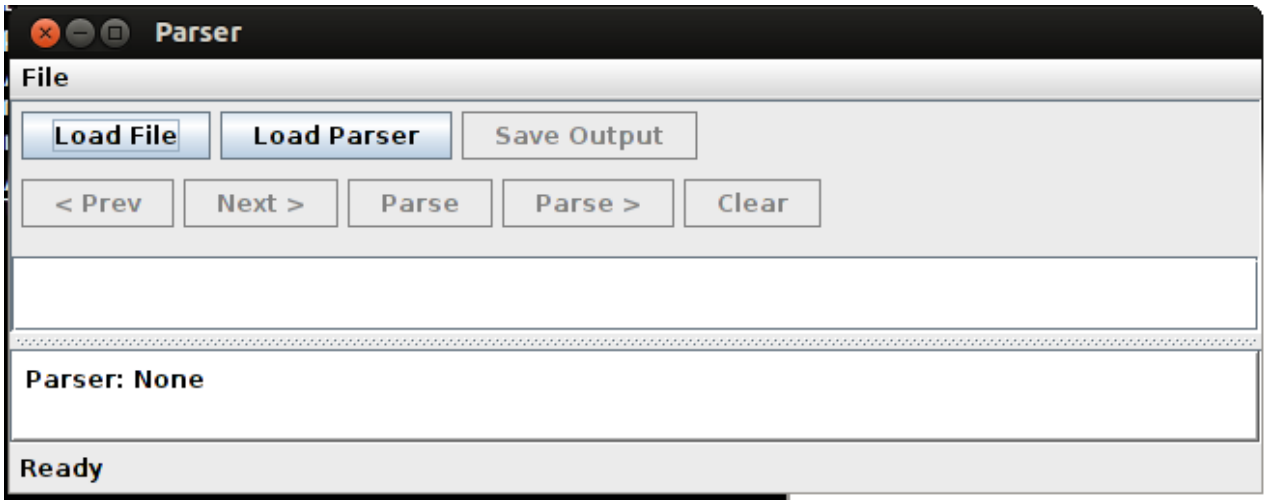
- **Forma Normal de Chomsky**: derivações de árvores binárias. Melhora a eficiência do parsing.



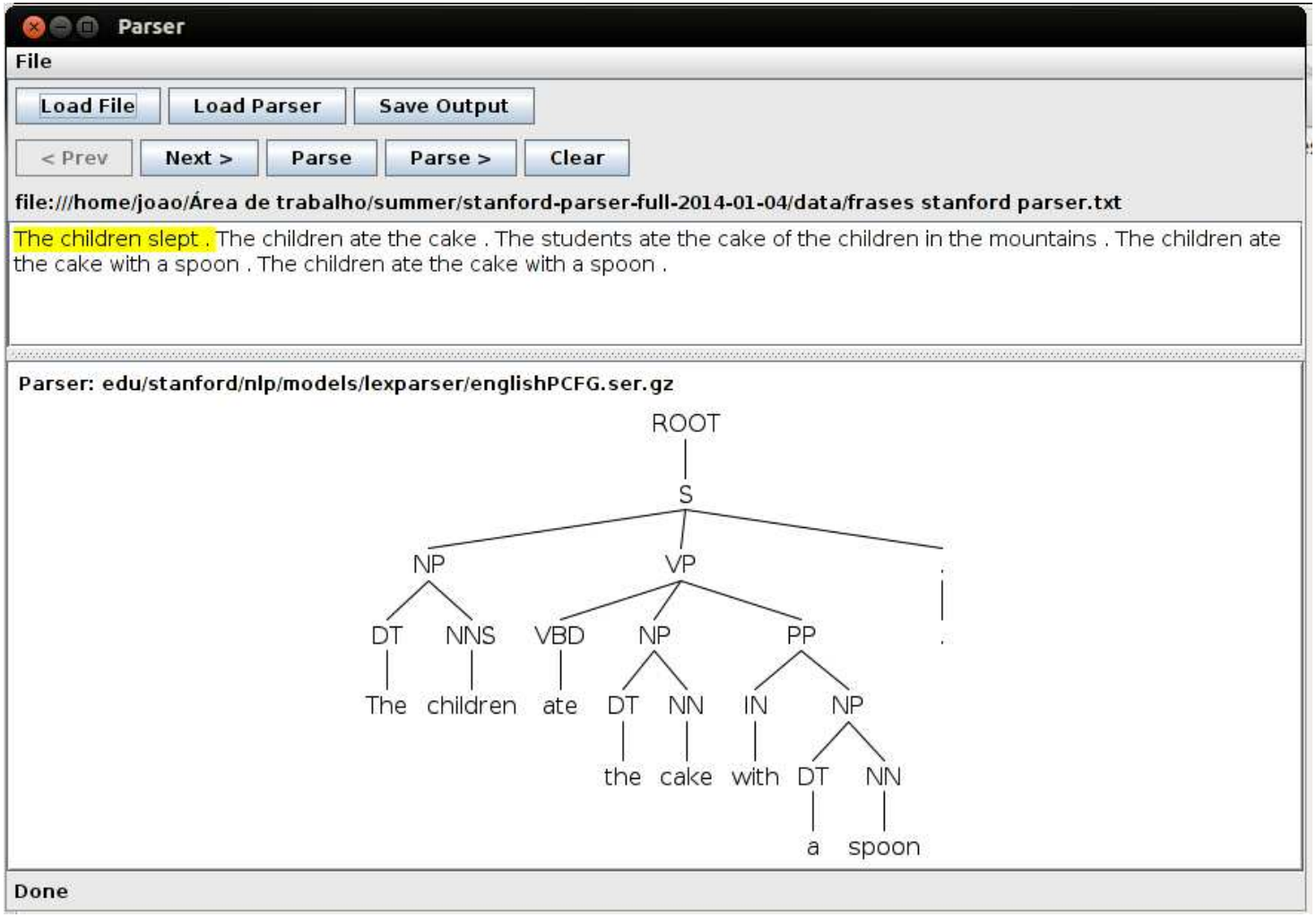


Stanford Parser

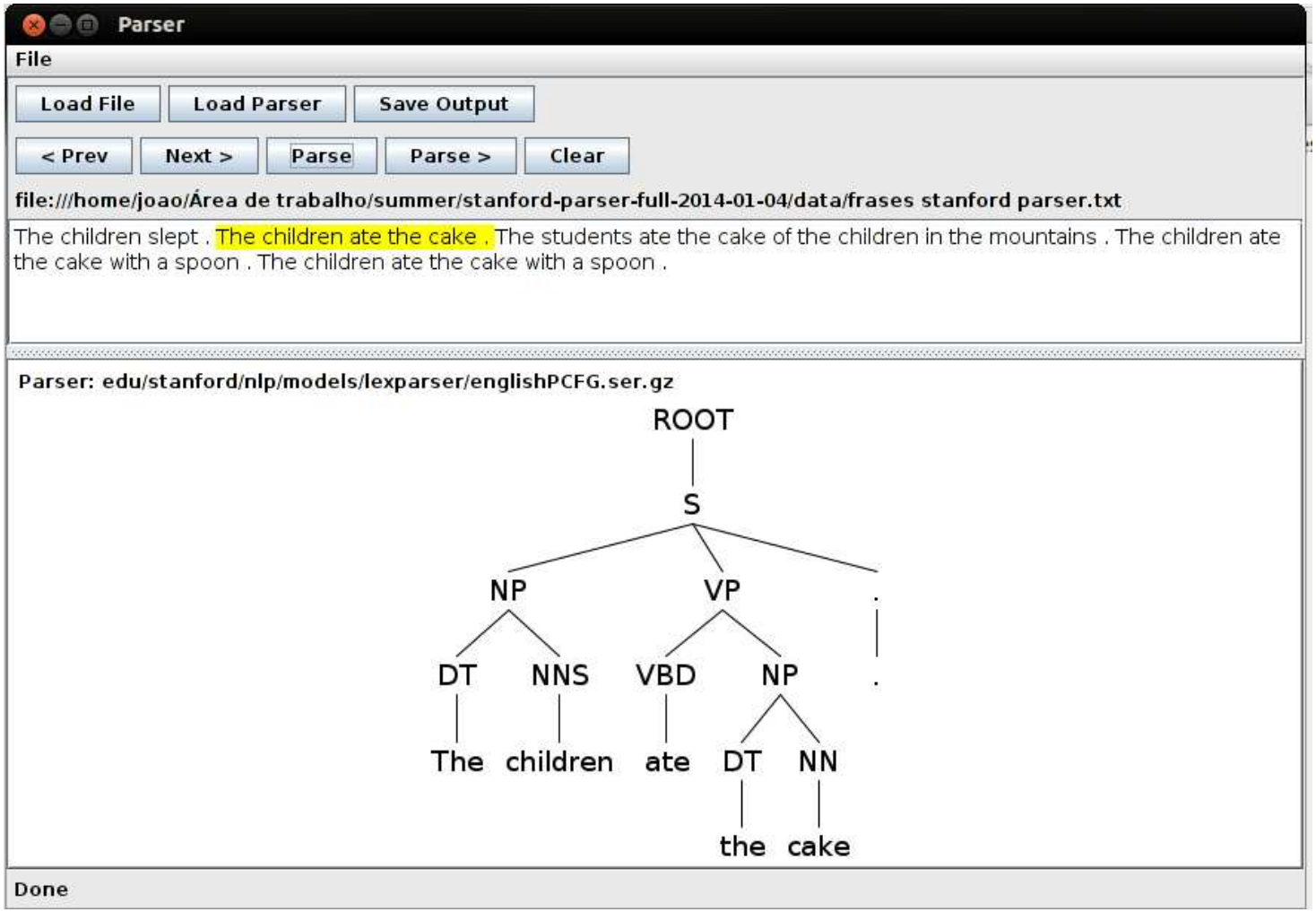
- java -jar stanford-parser.jar
- Selecionar o arquivo com o texto: *load file*
- Escolher o parser: *load parser* (stanford-parser-3.3.1-models.jar)



Stanford Parser

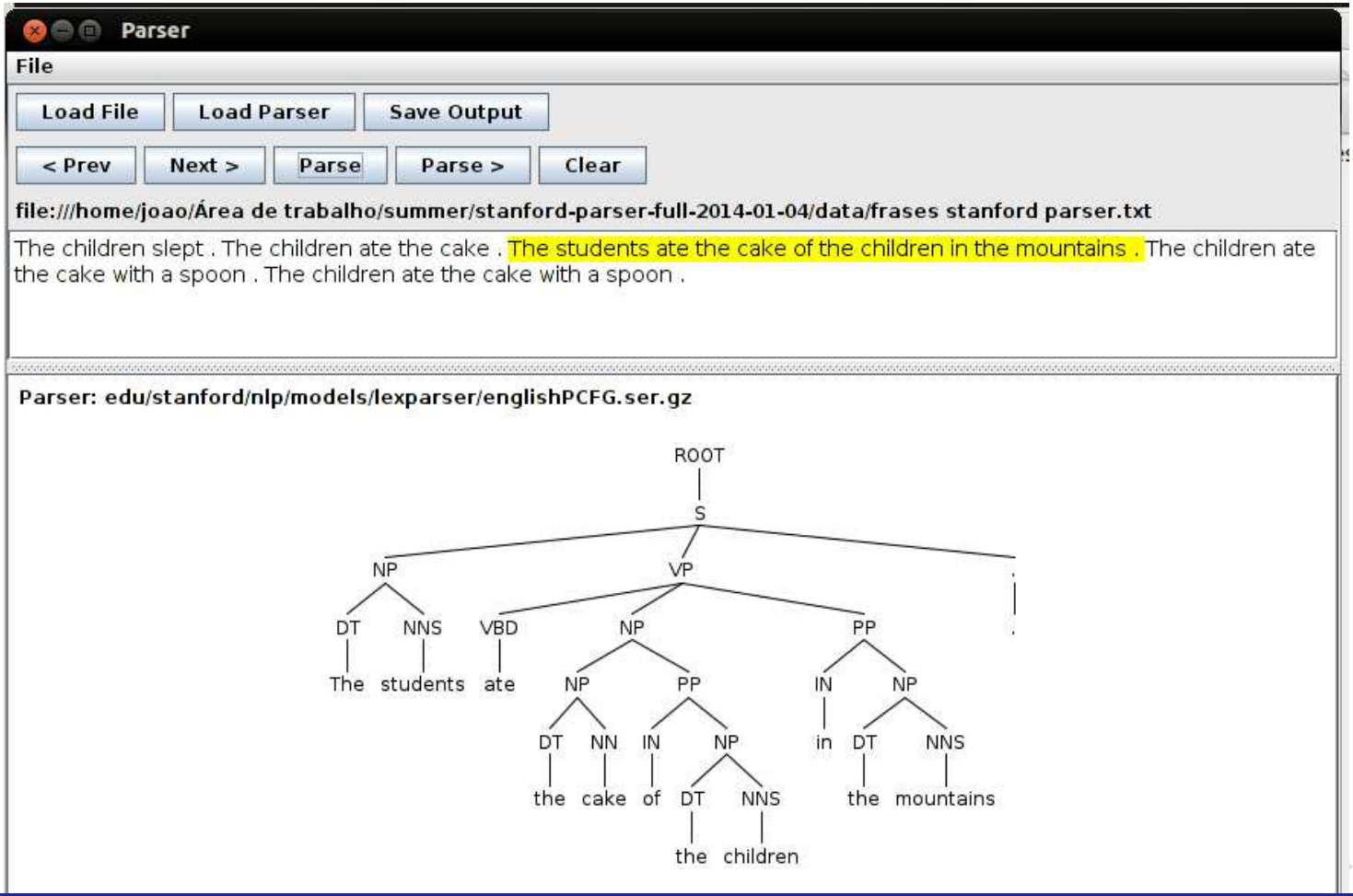


Stanford Parser



Stanford Parser

Ambiguidade



Parsing

Como resolver o problema de ambiguidade?

- Associado ao número de regras que a gramática possui
- **Gramática mais restritivas** $\Rightarrow$ sentenças que não possuem derivação nenhuma
- **Gramática menos restritivas** $\Rightarrow$ sentenças que possuem muitas derivações
- Sistema que seja flexível e que permita obter as derivações mais prováveis de uma sentença $\Rightarrow$ **Statistical Parsing**

Gramática Livre de Contexto Probabilística (PCFG)

$G = (T, N, S, L, R.P)$

- T : conjunto de símbolos terminais
- N : conjunto de símbolos não-terminais
- S : símbolo inicial ($S \in N$)
- L : conjunto de produções da forma $X \rightarrow x$ com $X \in N$ e $x \in T$ (lexicon)
- R : conjunto de produções da forma $X \rightarrow \gamma$ com $X \in N$ e $\gamma \in N^*$ (gramática)
- $P : R \rightarrow [0, 1]$: função de probabilidade

$$\forall X \in N \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$$

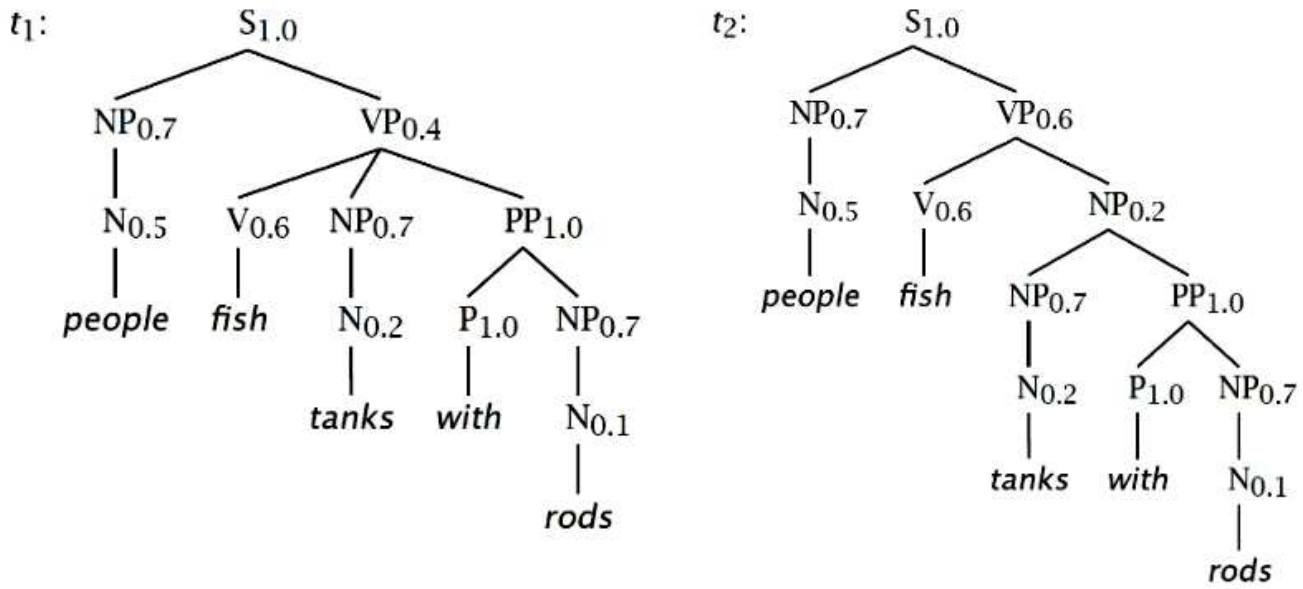
Gramática Livre de Contexto Probabilística (PCFG)

Exemplo

$S \rightarrow NP\ VP$	1.0	$N \rightarrow$ *people*	0.5
$VP \rightarrow V\ NP$	0.6	$N \rightarrow$ *fish*	0.2
$VP \rightarrow V\ NP\ PP$	0.4	$N \rightarrow$ *tanks*	0.2
$NP \rightarrow NP\ NP$	0.1	$N \rightarrow$ *rods*	0.1
$NP \rightarrow NP\ PP$	0.2	$V \rightarrow$ *people*	0.1
$NP \rightarrow N$	0.7	$V \rightarrow$ *fish*	0.6
$PP \rightarrow P\ NP$	1.0	$V \rightarrow$ *tanks*	0.3
		$P \rightarrow$ *with*	1.0

Gramática Livre de Contexto Probabilística (PCFG)

Exemplo



- Probabilidade de cada árvore : $P(t_1) = 0.0008232$ e $P(t_2) = 0.00024696$
- Probabilidade de um string s :
  $P(\text{people fish tanks with rods}) = P(t_1) + P(t_2) = 0.00107016$

The Penn Treebank Project

Bank of Linguistic Trees: Árvores anotadas com informações sintáticas e semânticas.

- Utilizado para treinamento de algoritmos de aprendizado de máquina
- Permite a construção de parsers melhores (mostra a flexibilidade da estrutura)
- Usado para testar hipóteses sobre linguagem
- Pode ser usado na avaliação de sistemas
- Existem diversos treebanks : diferentes linguagens e diferentes domínios

Tarefa

Para cada artigo contido no arquivo fornecido, faça um programa que:

- Separe os artigos em arquivos, de modo que cada arquivo será formado:
  - pelo texto referente a um artigo (armazenado como uma única string S);
  - uma lista de fatias da string S, onde cada fatia corresponde a uma linha do texto (segmentação);
  - uma lista de fatias da string S, onde cada fatia corresponde a uma entidade nomeada encontrada no texto.
- Aplique às sentenças de um dos arquivos gerados no item anterior os POS-taggers do NLTK.
- Gere um arquivo com as entidades nomeadas de cada artigo fornecido.

Tópicos Especias em Sistemas Inteligentes II

Aula 2: PLN - Introdução

João C. P. da Silva

August 26, 2016