


## Clase 2: Pre-procesamiento de textos

Marcelo Errecalde<sup>1,2</sup>

<sup>1</sup>Universidad Nacional de San Luis, Argentina 

<sup>2</sup>Universidad Nacional de la Patagonia Austral, Argentina 



Curso: Minería de Textos  
Facultad de Informática - Universidad Nacional de La Plata  
23 al 27 de Septiembre de 2019

## Resumen

- 1 **Partición del texto**
- 2 **Filtrado de palabras**
- 3 **Normalización de palabras**
- 4 **Etiquetado de palabras**

## Algunas técnicas de pre-procesamiento

- 1 **Partición** del texto
- 2 **Filtrado** (“stop-words”, baja frecuencia)
- 3 **Normalización** (mayúsculas, variaciones de uso)
- 4 Truncado (“**stemming**”) y lematización (“**lemmatization**”)
- 5 **Etiquetado** de las palabras
  - De **Partes de la Oración** (Part of Speech (**POS**) Tagging)
  - Desambiguación del Significado de las Palabras (**WSD**)
  - Reconocimiento de Entidades Nombradas (**NER**)

## Partición del texto

Constituyen el pre-procesamiento **básico** en la mayoría de las aplicaciones de TM.

- ① **Partición**: proceso que convierte el texto crudo en componentes **significativas** (de acuerdo al caso):
  - ① capítulos
  - ② secciones
  - ③ párrafos
  - ④ sentencias
  - ⑤ palabras (e incluso sílabas)

## Partición del texto “crudo”

- 1 **Partición**: proceso que convierte el texto crudo en componentes **significativas** (de acuerdo al caso):
  - 1 capítulos
  - 2 secciones
  - 3 párrafos
  - 4 sentencias
  - 5 palabras (e incluso sílabas)
- 2 La forma más común y básica de partición es el proceso que convierte una **secuencia de caracteres** en una **secuencia de palabras** (o **tokens**) usualmente referida como **tokenización**.

## Tokenización

- 1 **Tokenización**: tarea de **separar** (cortar/dividir) una **cadena de caracteres** en las mínimas **unidades lingüísticas identificables** (**tokens**)

- 2 **Ejemplo**, dada la siguiente sentencia:

*After sleeping for four hours, he decided to sleep for another four.*

El **proceso de tokenización** produciría:

{“After” “sleeping” “for” “four” “hours” “he” “decided”  
“to” “sleep” “for” “another” “four”}.

- 3 Este proceso suele requerir **conocimiento de dominio** substancial sobre el **lenguaje específico** en cuestión (límites de las palabras ambiguos debido a las particularidades de la **puntuación** en diferentes idiomas).

## Tokens/Palabras

- 1 **Token**: secuencia de caracteres de un texto que es tratado como una unidad indivisible para su procesamiento.
- 2 Cada mención de la misma palabra en un documento es tratada como un token separado.
- 3 Por lo tanto, la ocurrencia de la misma palabra **tres veces** creará los correspondientes **tres tokens**.
- 4 **Regla simple** de tokenización:  
*Usar los “espacios en blanco” (caracteres de espacio, tabs y newlines) como separadores una vez que los símbolos de puntuación han sido removidos.*
- 5 Varias situaciones **dificultan** este proceso simple de reconocimiento.

## Dificultades para reconocer límites de palabras (tokens)

- ❶ Falta (olvido) de **espacio en blanco** luego de puntuación.
- ❷ **Excepciones** al uso de símbolos de puntuación como separadores:
  - ❶ El **punto** (.) y la **coma** (,) para delimitar **decimales**.
  - ❷ Los **dos puntos** (:) en notación de **horarios** (ej. 8:20 PM)
- ❸ Similar al separador ‘/’ en fechas (ej. 16/11/2018).
- ❹ Puntos en acrónimos y abreviaturas (“Dr.”, “e.g.”, “p.ej.”)
- ❺ Guiones (-) en nros. de teléfono, de seguridad social, **emoticones** ( :) ), **hashtags** (#nlproc), etc.
- ❻ Direcciones de e-mail, URL’s, nros. de patentes, citas bibliográficas, etc



## Dificultades para reconocer límites de palabras (tokens)

- ❶ Expansión de contracciones **clíticas** marcadas con apóstrofes (')
- ❷ **clíticas**: palabras que sólo ocurren en combinación con otra palabra (ejemplo 'm en I'm.)
  - ❶ **what're** ⇒ se expande a dos tokens **what are**
  - ❷ **we're** ⇒ se expande a dos tokens **we are**
- ❸ Expresiones **multi-palabras** tratadas como **un sólo token** (**New York, Barack Obama, a priori**)
- ❹ Estas últimas requieren diccionarios multi-palabras y/o interacción con la **detección de entidades nombradas**.
- ❺ En la práctica (dado que la tokenización debe ejecutarse antes de cualquier otro procesamiento de lenguaje), es importante que sea **muy rápido**.
- ❻ El método estándar es utilizar **algoritmos determinísticos** basados en **expresiones regulares** que se compilan en autómatas de estado finito muy eficientes.

## Expresiones regulares

- Se pueden usar para tokenizar texto y tener mucho más control sobre el proceso.
- Una **expresión regular** (ER) es una **notación algebraica** (un **lenguaje**) para caracterizar un **conjunto de cadenas**.
- Permiten buscar un **patrón** en un texto o colección de textos (corpus).
- Una función de **búsqueda** de ERs buscará a través del corpus y devolverá todos los textos que coincidan con el patrón.
- Ejemplo, el comando **grep** de Unix toma una ER y devuelve cada línea del documento de entrada que coincide con la expresión.

# Meta-caracteres de expresiones regulares básicas

Operator	Behavior
.	Wildcard, matches any character
^abc	Matches some pattern <i>abc</i> at the start of a string
abc\$	Matches some pattern <i>abc</i> 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. <i>a*</i> , <i>[a-z]*</i> (also known as <i>Kleene Closure</i> )
+	One or more of previous item, e.g. <i>a+</i> , <i>[a-z]+</i>
?	Zero or one of the previous item (i.e. optional), e.g. <i>a?</i> , <i>[a-z]?</i>
{n}	Exactly <i>n</i> repeats where <i>n</i> is a non-negative integer
{n, }	At least <i>n</i> repeats
{, n}	No more than <i>n</i> repeats
{m, n}	At least <i>m</i> and no more than <i>n</i> repeats
a(b c)+	Parentheses that indicate the scope of the operators

## Algunos símbolos de expresiones regulares

Symbol	Function
\b	Word boundary (zero width)
\d	Any decimal digit (equivalent to [0-9])
\D	Any non-digit character (equivalent to [^0-9])
\s	Any whitespace character (equivalent to [ \t\n\r\f\v])
\S	Any non-whitespace character (equivalent to [^ \t\n\r\f\v])
\w	Any alphanumeric character (equivalent to [a-zA-Z0-9_])
\W	Any non-alphanumeric character (equivalent to [^a-zA-Z0-9_])
\t	The tab character
\n	The newline character

## Separando tokens usando espacios en blanco

**In [1]:**

```
import re
```

```
raw = """'When I'M a Duchess,' she said to herself, (not in a very  
hopeful tone though), 'I won't have any pepper in my kitchen AT  
ALL. Soup does very well without--Maybe it's always pepper that  
makes people hot-tempered,'"""
```

```
print(raw.split()) #usando split como herramienta  
print(re.split(r' ', raw)) #con expresiones regulares (ojo)
```

```
['"When"', 'I'M"', 'a', 'Duchess,', '"', 'she', 'said', 'to', 'herself,', '  
'(not', 'in', 'a', 'very', 'hopeful', 'tone', 'though)', '"', '"I',  
"won't", 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT',  
'ALL.', 'Soup', 'does', 'very', 'well', 'without--Maybe', "it's",  
'always', 'pepper', 'that', 'makes', 'people', "hot-tempered,'"']
```

```
['"When"', 'I'M"', 'a', 'Duchess,', '"', 'she', 'said', 'to', 'herself,', '  
'(not', 'in', 'a', 'very', '\nhopeful', 'tone', 'though)', '"', '"I',  
"won't", 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT',  
'\nALL.', 'Soup', 'does', 'very', 'well', 'without--Maybe', "it's",  
'always', 'pepper', 'that', '\nmakes', 'people', "hot-tempered,'"']
```

## ... mejorando la ER que separa tokens...

**In [1]:**

```
print(re.split(r'[\t\n]+', raw))
```

```
['"When"', 'I\'M"', 'a', 'Duchess,', '"', 'she', 'said', 'to', 'herself,',  
'(not', 'in', 'a', 'very', 'hopeful', 'tone', 'though),', '"I",  
"won\'t"', 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT',  
'ALL.', 'Soup', 'does', 'very', 'well', 'without--Maybe', "it's",  
'always', 'pepper', 'that', 'makes', 'people', "hot-tempered,"']
```

o usando la abreviatura \s (cualquier caracter de espacio en blanco)

**In [1]:**

```
print(re.split(r'\s+', raw))
```

```
['"When"', 'I\'M"', 'a', 'Duchess,', '"', 'she', 'said', 'to', 'herself,',  
'(not', 'in', 'a', 'very', 'hopeful', 'tone', 'though),', '"I",  
"won\'t"', 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT',  
'ALL.', 'Soup', 'does', 'very', 'well', 'without--Maybe', "it's",  
'always', 'pepper', 'that', 'makes', 'people', "hot-tempered,"']
```

## ... otras variantes de separación ...

Cosas que no sean **caracteres de palabras** (CPs):

In [1]:

```
print(re.split(r'\W+', raw))
```

```
['', 'When', 'I', 'M', 'a', 'Duchess', 'she', 'said', 'to', 'herself',  
'not', 'in', 'a', 'very', 'hopeful', 'tone', 'though', 'I', 'won',  
't', 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT', 'ALL',  
'Soup', 'does', 'very', 'well', 'without', 'Maybe', 'it', 's', 'always  
'pepper', 'that', 'makes', 'people', 'hot', 'tempered', '']
```

o buscando las palabras en lugar de los separadores:

In [1]:

```
print(re.findall(r'\w+', raw))
```

```
['When', 'I', 'M', 'a', 'Duchess', 'she', 'said', 'to', 'herself',  
'not', 'in', 'a', 'very', 'hopeful', 'tone', 'though', 'I', 'won',  
't', 'have', 'any', 'pepper', 'in', 'my', 'kitchen', 'AT', 'ALL',  
'Soup', 'does', 'very', 'well', 'without', 'Maybe', 'it', 's',  
'always', 'pepper', 'that', 'makes', 'people', 'hot', 'tempered']
```

## ... y buscando patrones más complejos ...

CPs **o** algo que **no** es un espacio en blanco seguido de CPs:

**In [1]:**

```
print(re.findall(r'\w+|\S\w*', raw))
```

```
['"When", 'I', '"M", 'a', 'Duchess', ',', ',', '"', 'she', 'said', 'to',  
'herself', ',', ',', '(not', 'in', 'a', 'very', 'hopeful', 'tone', 'though',  
)', ',', ',', '"I", 'won', 't', 'have', 'any', 'pepper', 'in', 'my',  
'kitchen', 'AT', 'ALL', '., 'Soup', 'does', 'very', 'well', 'without',  
'-', '-Maybe', 'it', '"s', 'always', 'pepper', 'that', 'makes', 'peopl',  
'hot', '-tempered', ',', ',', '"']
```

e incluyendo (entre otras cosas) palabras unidas por -:

**In [1]:**

```
print(re.findall(r"\w+(?:[-']\w+)*|'[-.()]+|\S\w*", raw))
```

```
['"', 'When', 'I"M", 'a', 'Duchess', ',', ',', '"', 'she', 'said', 'to',  
'herself', ',', ',', '(', 'not', 'in', 'a', 'very', 'hopeful', 'tone',  
'though', ')', ',', ',', '"', 'I', 'won't', 'have', 'any', 'pepper', 'in',  
'my', 'kitchen', 'AT', 'ALL', '., 'Soup', 'does', 'very', 'well',  
'without', '--', 'Maybe', 'it's', 'always', 'pepper', 'that', 'makes',  
'people', 'hot-tempered', ',', ',', '"']
```



## Filtrado

Se **remueven palabras** (tokens) de determinado tipo o que no cumplen ciertas condiciones:

- **Palabras de paro** (“stop-words”): palabras con escasa (o nula) información de contenido (artículos, conjunciones, preposiciones, etc).
  - En ciertas tareas no son eliminadas (ej: se requiere capturar **estilo**)
  - Existen diccionarios **específicos del lenguaje**
  - Se puede “aproximar” este efecto con eliminación de palabras **demasiado frecuentes** o el uso de **normalizaciones de pesos** que las **penalizan** (idf)
- **Umbrales** de frecuencia mínimo (ver métodos de filtrado en teoría de **reducción de dimensionalidad**).
- Símbolos o elementos **especiales** (símbolos de puntuación, números, etc)

## Normalizaciones (mayúsculas y variaciones de uso)

En este contexto, **normalizar** es llevar variaciones de tokens a una forma común.

- El uso de la **mayúscula** puede definir el significado de un término. **Bob** puede ser un **verbo** usado al comienzo de una sentencia o el nombre de una persona.
- Regla usual: palabras en el comienzo de sentencias, títulos y encabezados de sección se **convierten a minúscula**. El resto se mantiene como está.
- Otros casos que suelen normalizarse: pequeñas variaciones del mismo token que refieren a la misma palabra/concepto: **colour/color**, **naive/naïve**, **US/USA**.
- Solución: tablas con posibles variaciones y su forma estandarizada.

## Lematización y truncado

- Se pueden ver como otra forma de **normalización / standarización**
- Buscan reducir las **formas infleccionales** y **derivadas** de las palabras, obteniendo las **formas bases** comunes.

### Métodos de *lematización*

- Mapean palabras a su forma **base** o de **diccionario** (lema).
- En formas verbales busca el **infinitivo** y en sustantivos su forma **singular**.
- Requieren del uso de rotuladores (p. ej. POS tagger)
- Ejemplo: **saw** ⇒ **see** o **saw** (dependiendo de si es reconocido como sustantivo o verbo)

## Lematización y truncado

- Se pueden ver como otra forma de **normalización / standarización**
- Buscan reducir las **formas infleccionales** y **derivadas** de las palabras, obteniendo las **formas bases** comunes.

### Métodos de *truncado* (“stemming”)

- Considerados por algunos como una **lematización básica**.
- Más sencillo y económico que la obtención de lemas.
- Obtiene formas básicas de las palabras a partir de la *poda* de sufijos (“s” en sustantivos, “ing” en verbos, etc.)
- El “stem” o raíz de la palabra intenta representar palabras con igual (o similar significado).

## Ejemplos de truncado

El algoritmo de stemming (para inglés) más conocido es el alg. de **Porter**. Sitio oficial:

<http://tartarus.org/martin/PorterStemmer/>

### Ejemplos con Porter

- Ejemplo 1: Porter convierte: **operate operating operates operation operative operatives operational** en **oper**.
- Ejemplo 2: Ojo, Porter podría convertir **saw**  $\Rightarrow$  **s**.
- Buena comparación de stemming y lemmatization:  
<http://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>

## Ejemplo completo con Porter

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things—names and heights and soundings—with the single exception of the red crosses and the written notes.

produce la siguiente salida con palabras truncadas

Thi wa not the map we found in Billi Bone s  
chest but an accur copi complet in all thing  
name and height and sound with the singl  
except of the red cross and the written note

## Comparación truncado (nltk) vs lematización(spacy)

**In [1]:**

```
import spacy
import nltk

# cargar el modelo del lenguaje inglés de spacy
en_nlp = spacy.load('en')
doc = u"I saw there some saws to cut the tree"
# instanciar el "stemmer" de Porter de nltk
stemmer = nltk.stem.PorterStemmer()
# tokenizar documento con spacy
doc_spacy = en_nlp(doc)
# imprimir lemas encontrados por spacy
print("Lematización:")
print([token.lemma_ for token in doc_spacy])
# imprimir tokens obtenidos con el stemmer de Porter
print("Truncado:")
print([stemmer.stem(token.norm_.lower()) for token in doc_spacy])

Lematización:
['-PRON-', 'see', 'there', 'some', 'saw', 'to', 'cut', 'the', 'tree']
Truncado:
['i', 'saw', 'there', 'some', 'saw', 'to', 'cut', 'the', 'tree']
```

## Etiquetado de las Categorías Gramaticales (ECG)

Las palabras de una oración pueden ser etiquetadas/categorizadas de acuerdo al **rol** que tienen en el contexto de una sentencia: **sustantivo**, **verbo**, **adjetivo**, **adverbio**, etc.

**Ejemplo**, en:

*Si usted **tapa** la olla azul, necesitará una **tapa** grande.*

El primer uso de **tapa** corresponde a un **verbo**, y el segundo a un **sustantivo**.

Estas categorías de una palabra de acuerdo al contexto de uso se las conoce como **categorías gramaticales**, **clases de palabras** o **partes de la oración** (en inglés **Part-of-Speech (POS)**)



## Etiquetado de las Categorías Gramaticales (ECG) (II)

### Definición

El **Etiquetado de las Categorías Gramaticales (ECG)** (en inglés **Part-of-Speech (POS) Tagging**) es la tarea de asignar a las palabras distintas categorías gramaticales de acuerdo al rol que tienen en las sentencias en que aparecen.

Ejemplo: las palabras de la sentencia

**The grand jury commented on a number of other topics.**

podrían ser etiquetadas con las siguientes categorías por un sistema de ECG:

**The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.**

## Etiquetado de las Categorías Gramaticales (ECG) (III)

The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN**  
a/**DT** number/**NN** of/**IN** other/**JJ** topics/**NNS** ./.

- **DT** (determinador), **JJ** (adjetivo), **NN** (sustantivo), **IN** (preposición), **VBD** (verbo en pasado) y **NNS** (sustantivo plural) son etiquetas usuales en una sentencia.
- Sin embargo, considerando distintas categorías y subcategorías, se han propuesto distintos **tagsets**:
  - 1 **Penn Treebank** (45 etiquetas)
  - 2 **Brown corpus** (87 etiquetas)
  - 3 **C7 tagset** (146 etiquetas)

## Etiquetas para el inglés (Penn Treebank)

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+, %, &amp;</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VCN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>' or "</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>' or "</i>
PRP	Personal pronoun	<i>I, you, he</i>	(	Left parenthesis	<i>[, (, {, &lt;</i>
PRP\$	Possessive pronoun	<i>your, one's</i>	)	Right parenthesis	<i>], ), }, &gt;</i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>. ! ?</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>: ; ... --</i>
RP	Particle	<i>up, off</i>			

## Ejemplo 1: POS tagging (Python)

**In [1]:**

```
from nltk import pos_tag, wordpunct_tokenize
text = "The old building was demolished. Tomorrow, they will begin
building a new one"
pos_tag(wordpunct_tokenize(text))
```

**Out [1]:**

```
[('The', 'DT'),
 ('old', 'JJ'),
 ('building', 'NN'),
 ('was', 'VBD'),
 ('demolished', 'VBN'),
 ('.', '.'),
 ('Tomorrow', 'NNP'),
 (',', ','),
 ('they', 'PRP'),
 ('will', 'MD'),
 ('begin', 'VB'),
 ('building', 'VBG'),
 ('a', 'DT'),
 ('new', 'JJ'),
 ('one', 'CD')]
```

## Ejemplo 2: POS tagging (Python)

**In [1]:**

```
text = "The grand jury commented on a number of other topics."  
pos_tag(wordpunct_tokenize(text))
```

**Out [1]:**

```
[('The', 'DT'),  
 ('grand', 'JJ'),  
 ('jury', 'NN'),  
 ('commented', 'VBD'),  
 ('on', 'IN'),  
 ('a', 'DT'),  
 ('number', 'NN'),  
 ('of', 'IN'),  
 ('other', 'JJ'),  
 ('topics', 'NNS'),  
 ('.', '.')]
```

## ECG (POS tagging) - consideraciones finales

### Observaciones

- El **ECG (POS tagging)** también es llamado **etiquetado gramatical** o **desambiguación de la categoría gramatical**.
- Determinan estas categorías en base al *contexto* en que se encuentra la palabra.
- La mayoría de los ECG caen en dos grandes grupos: a) basados en **reglas** (p.ej. **EngCG**) y b) enfoques estocásticos (p.ej. **HMM** taggers)

### Material complementario

- Capítulo 8 (Part of Speech Tagging) de [2]
- Capítulo 5 (Categorizing and Tagging Words) de [1]

## Desambiguación del Significado de las Palabras

### Definición

En inglés **Word Sense Disambiguation (WSD)** trata de resolver la ambigüedad en el significado de las palabras en base al contexto en que éstas aparecen.

### Ejemplo, la palabra “banco” ...

Frase		Significado
Perdí la mañana en el <b>banco</b> pagando impuestos.	⇒	Institución Financiera
Desde el barco vi el <b>banco</b> de peces.	⇒	Cardumen
Sentado en un <b>banco</b> suspiraba.	⇒	Mueble
Las donaciones se están recibiendo en el <b>banco</b> de sangre.	⇒	Establecimiento Médico

## La Ambigüedad en el PLN

- El **lenguaje natural** se distingue de los **lenguajes artificiales** por su **riqueza** y **flexibilidad**.
- Sin embargo, si bien es bueno para la **comunicación humana** introduce problemas como la **ambigüedad**.
- Como se ha expresado en otros trabajos:

*“ Lo que son ventajas para la comunicación humana se convierten en problemas a la hora de un tratamiento computacional, ya que implican conocimiento y procesos de razonamiento que son difíciles de formalizar.”*
- Entre esos problemas, la **ambigüedad** es uno de los principales en el PLN y se da a distintos **niveles** (**palabras** polisémicas, **oraciones** con  $\neq$  interpretaciones), etc.



## Algunos tipos de *ambigüedad*

- Ambigüedad **léxica**: una **palabra**  $\Rightarrow \neq$  **categorías gramaticales**. Ejemplo: “**para**” (¿preposición o verbo?).
- Ambigüedad **sintáctica** (o **estructural**): una **oración**  $\Rightarrow \neq$  **interpretaciones**. Ejemplo: “Juan vio a su hermana con unos **prismáticos**”.
- Ambigüedad **semántica**: incluye
  - Ambigüedad debido a **palabras polisémicas**: Ejemplo: “**banco**”
  - Ambigüedad **referencial**: Ejemplo: “El jamón está en el armario. **Sácalo**. **Ciérralo**”.

En cierto sentido, distintas tareas de PLN (POS tagging, ENR, categorización) podrían ser consideradas  $\neq$  formas de **desambigüación**

## Word Sense Disambiguation (WSD)

### Desambigüación (Word Sense Disambiguation (WSD))

Procedimiento utilizado para decidir los significados de las palabras a partir del contexto que las rodea.

#### WSD en PLN

- **Traducción Automática:** “I’m in bed because I have a cold”
  - 1 “Estoy en cama porque estoy resfriado”
  - 2 “Estoy en cama porque tengo un frío”, No!!!!.
- **Recuperación de la Información:** Encontrar todas las páginas Web que hablen del “pato”. (¿El deporte o el ave?)
- **Búsqueda de respuestas:** ¿Qué opina George Miller sobre el control de armas?. (¿El psicólogo o el congresista?)
- **Adquisición de Conocimiento:** Añadir a la KB: Herb Bergson es el alcalde de Duluth (¿Minnesota o Georgia?)

## Clasificación de Sistemas de WSD

(de acuerdo a la principal fuente de conocimiento utilizada)

- Métodos basados en **Conocimiento** (o en **Diccionario**):  
utilizan diccionarios, tesauros y KB léxicas (como **Wordnet**)
  - ① solapamiento contextual entre definiciones de un diccionario (Algoritmo **Lesk**)
  - ② Similitud Semántica
  - ③ Preferencias de selección
  - ④ Métodos heurísticos (MFS, 1SXD)
- Métodos basados en **Corpus**
  - ① **No supervisados**: trabajan con corpus **sin etiquetar**.
  - ② **Supervisados**: trabajan con corpus **etiquetados semánticamente**.
- Métodos **Híbridos**

## La ontología Wordnet

- Concebido como un diccionario electrónico.
- El contenido se organiza mediante una base de datos léxica, donde se agrupan conjuntos de palabras (nombres, verbos, adjetivos y adverbios) en grupos de sinónimos llamados **synsets**.
- Por lo tanto, un **synset** es un conjunto de todas las palabras que expresan un mismo concepto.
- Dentro de WN, cada synset se codifica con un **número único** que representa un **concepto distinto**.
- Entre los **synsets** existen conexiones que expresan **relaciones** semánticas, conceptuales o léxicas.

## Relaciones definidas en Wordnet

- **Sinonimia**: car  $\Rightarrow$  automobile
- **Antonimia**: clean  $\Rightarrow$  dirty
- **Hiponimia** (TIPOS DE): cat  $\Rightarrow$  domestic cat, cat  $\Rightarrow$  wildcat
- **Hiperonimia** (ES UN TIPO DE): pine  $\Rightarrow$  tree, cat  $\Rightarrow$  feline  $\Rightarrow$  carnivore
- **Meronimia** (PARTES DE): foot  $\Rightarrow$  toe.
- **Holonimia**: (ES UNA PARTE DE): toe  $\Rightarrow$  foot, eye  $\Rightarrow$  face
- **Troponimia**: es **hiponimia** a nivel verbos. swim  $\Rightarrow$  crawl
- **Entailment**: es la inferencia lógica. snore  $\models$  sleep

## Algunas estadísticas de Wordnet (3.0)

POS	Words	Synsets
<b>Noun</b>	117798	82115
<b>Verb</b>	11529	13767
<b>Adjective</b>	21479	18156
<b>Adverb</b>	4481	3621
<b>Totals</b>	155287	117659

## WSD y Wordnet

- La tarea de **WSD** es básicamente una de **clasificación**:  
*Dada una palabra, asignar un **sentido unívoco** a la misma, de acuerdo a su contexto (palabras que la rodean) en la oración.*
- Este **sentido unívoco** está dado por el identificador (unívoco) de un **synset** de la ontología Wordnet.
- Un **synset** (por **conjunto de sinónimos**) es un conjunto de sentidos/significados de palabras que representan cosas que son **sinónimos** (o casi).
- Así, un **synset** representa un **concepto**, aquel representado como la lista de los significados de palabras que se usan para expresar dicho concepto.

## Ejemplo: sentidos en WN de la palabra “bass”

The noun “bass” has 8 senses in WordNet.

1. bass<sup>1</sup> - (the lowest part of the musical range)
2. bass<sup>2</sup>, bass part<sup>1</sup> - (the lowest part in polyphonic music)
3. bass<sup>3</sup>, basso<sup>1</sup> - (an adult male singer with the lowest voice)
4. sea bass<sup>1</sup>, bass<sup>4</sup> - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass<sup>1</sup>, bass<sup>5</sup> - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass<sup>6</sup>, bass voice<sup>1</sup>, basso<sup>2</sup> - (the lowest adult male singing voice)
7. bass<sup>7</sup> - (the member with the lowest range of a family of musical instruments)
8. bass<sup>8</sup> - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

The adjective “bass” has 1 sense in WordNet.

1. bass<sup>1</sup>, deep<sup>6</sup> - (having or denoting a low vocal or instrumental range)  
    *“a deep voice”*; *“a bass voice is lower than a baritone voice”*;  
    *“a bass clarinet”*



## Sentidos y synsets

- *bass* tiene 8 sentidos como sustantivo y 1 como adjetivo
- Aquí se pueden identificar synsets como {*bass*<sup>1</sup>, *deep*<sup>6</sup>} y {*bass*<sup>6</sup>, *bassvoice*<sup>1</sup>, *basso*<sup>2</sup>}
- Con NLTK se pueden obtener todos los sentidos de una palabra en Wordnet:

In [1]:

```
from nltk.corpus import wordnet as wn  
wn.synsets('bass')
```

Out [1]:

```
[Synset('bass.n.01'),  
 Synset('bass.n.02'),  
 Synset('bass.n.03'),  
 Synset('sea_bass.n.01'),  
 Synset('freshwater_bass.n.01'),  
 Synset('bass.n.06'),  
 Synset('bass.n.07'),  
 Synset('bass.n.08'),  
 Synset('bass.s.01')]
```

## WSD (NLTK)

- Desambiguar el sentido de la palabra (WSD) no es más que dar el **lema** de la palabra, su **rol/categoría gramatical** y **el contexto** en que está usada.
- El resultado es el **synset/concepto** (unívoco) que corresponde a este uso del lema:
- **Ejemplo:** usando para WSD el algoritmo de **Lesk**

In [1]:

```
lesk(['I', 'went', 'to', 'the', 'bank', 'to', 'deposit', 'money', '.']  
     'bank', 'n')
```

Out [1]:

```
Synset('savings_bank.n.02')
```

## Reconocimiento de Entidades Nombradas (REN)

### Características

- Subtarea de Extracción de Información.
- Consiste en la **ubicación** y **clasificación** de secuencias de palabras dentro de un texto, en categorías tales como **nombres de personas**, **lugares**, **organizaciones**, **cantidades**, etc.
- El resultado de este proceso es un documento con etiquetas que identifican el comienzo y fin de las entidades nombradas.
- Ejemplo: “Jim bought 300 shares of Acme Corp. in 2006”  
⇒ <ENAMEX TYPE=“PERSON”>Jim</ENAMEX> bought  
<NUMEX TYPE=“QUANTITY”>300</NUMEX> shares of  
<ENAMEX TYPE=“ORGANIZATION”>Acme Corp.  
</ENAMEX> in <TIMEX TYPE=“DATE”>2006</TIMEX>.

## Entidades Nombradas (EN)

### Entidades nombradas

- **Secuencia de palabras** que refiere a una **entidad** específica del mundo real mediante un **nombre propio**: una **persona**, una **ubicación**, una **organización**.
- También incluye otras cosas que no son estrictamente entidades nombradas (**fechas**), **valores monetarios**, etc

### Tipos y ejemplos de EN genéricas

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	<b>Turing</b> is a giant of computer science.
Organization	ORG	companies, sports teams	The <b>IPCC</b> warned about the cyclone.
Location	LOC	regions, mountains, seas	The <b>Mt. Sanitas</b> loop is in <b>Sunshine Canyon</b> .
Geo-Political	GPE	countries, states, provinces	<b>Palo Alto</b> is raising the fees for parking.
Entity			
Facility	FAC	bridges, buildings, airports	Consider the <b>Tappan Zee Bridge</b> .
Vehicles	VEH	planes, trains, automobiles	It was a classic <b>Ford Falcon</b> .

## Dificultades para el REN

- **Ambigüedad** de la **segmentación**: ¿ Donde empieza y termina la EN?
- **Ambigüedad** del **tipo** de EN: ¿**JFK** refiere al **presidente**, el **aeropuerto** o alguna **escuela**, **punto** o **calle**?

Algunas **ambigüedades categóricas** comunes:

Name	Possible Categories
<i>Washington</i>	Person, Location, Political Entity, Organization, Facility
<i>Downing St.</i>	Location, Organization
<i>IRA</i>	Person, Organization, Monetary Instrument
<i>Louis Vuitton</i>	Person, Organization, Commercial Product

Ejemplos de **ambigüedades** de **tipo** con el nombre **Washington**

[*PERS* Washington] was born into slavery on the farm of James Burroughs.  
[*ORG* Washington] went up 2 games to 1 in the four-game series.  
Blair arrived in [*LOC* Washington] for what may well be his last state visit.  
In June, [*GPE* Washington] passed a primary seatbelt law.  
The [*FAC* Washington] had proved to be a leaky ship, every passage I made...

## REN en NLTK-Python

Se puede usar la función `nltk.ne_chunk()`, que toma como entrada una sentencia con los POS tags.

Usaremos una sentencia del corpus TreeBank

**In [1]:**

```
sent = nltk.corpus.treebank.tagged_sents()[22]  
sent
```

**Out [1]:**

```
[('The', 'DT'),  
 ('U.S.', 'NNP'),  
 ('is', 'VBZ'),  
 ('one', 'CD'),  
 ('of', 'IN'),  
 ('the', 'DT'),  
 ('few', 'JJ'),  
   ...  
]
```

## REN en NLTK-Python (II)

**In [1]:**

```
print(nltk.ne_chunk(sent))
```

**Out [1]:**

```
(S
  The/DT
  (GPE U.S./NNP)
  is/VBZ
  one/CD
  ....
  according/VBG
  to/TO
  (PERSON Brooke/NNP T./NNP Mossman/NNP)
  ...)
```

## Referencias y material de lectura

[1] Steven Bird, Ewan Klein, and Edward Loper, 2009 *Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

<http://www.nltk.org/book/>.

[2] Dan Jurafsky and James H. Martin, 2019- *Speech and Language Processing. 3rd edition draft*. <https://web.stanford.edu/~jurafsky/slp3/>

**Notebook asociada:**

[https://github.com/merrecalde/curso\\_la\\_plata\\_2019/blob/master/clase\\_2\\_pre\\_procesamiento.ipynb](https://github.com/merrecalde/curso_la_plata_2019/blob/master/clase_2_pre_procesamiento.ipynb)