UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

Seminario de análisis del discurso

Dr. César Antonio Aguilar
Facultad de Lenguas y Letras
23/09/2010

CAguilar@iingen.unam.mx

Análisis de corpus por computadora (1)

Siguiendo con el uso de herramientas para hacer análisis en corpus textuales, veamos de forma general una plataforma diseñada para hacer distintas tareas lingüísticas llamada **Natural Language Tool-Kit** (NLTK).

NLTK es una plataforma de herramientas programables, las cuales han sido diseñadas en lenguaje Python.

NLTK fue desarrollado básicamente por Steven Bird (Universidad de Melbourne), Ewan Klein (Universidad de Edinburg) y Edward Loper (Universidad de Pennsylvania).

www.nltk.org/

Steven Bird

Ewan Klein

Edward Loper

Análisis de corpus por computadora (2)

NLTK es un recurso diseñado para realizar diversas tareas relacionadas con el procesamiento de lenguaje natural, entre ellas, el análisis de corpus lingüísticos.

www.python.org

Para poder sacarle mejor provecho a NLTK, conviene primero aprender algunos principios básicos de programación en Python. Para esta clase, usaremos algunas **expresiones regulares** para resolver algunos problemas.

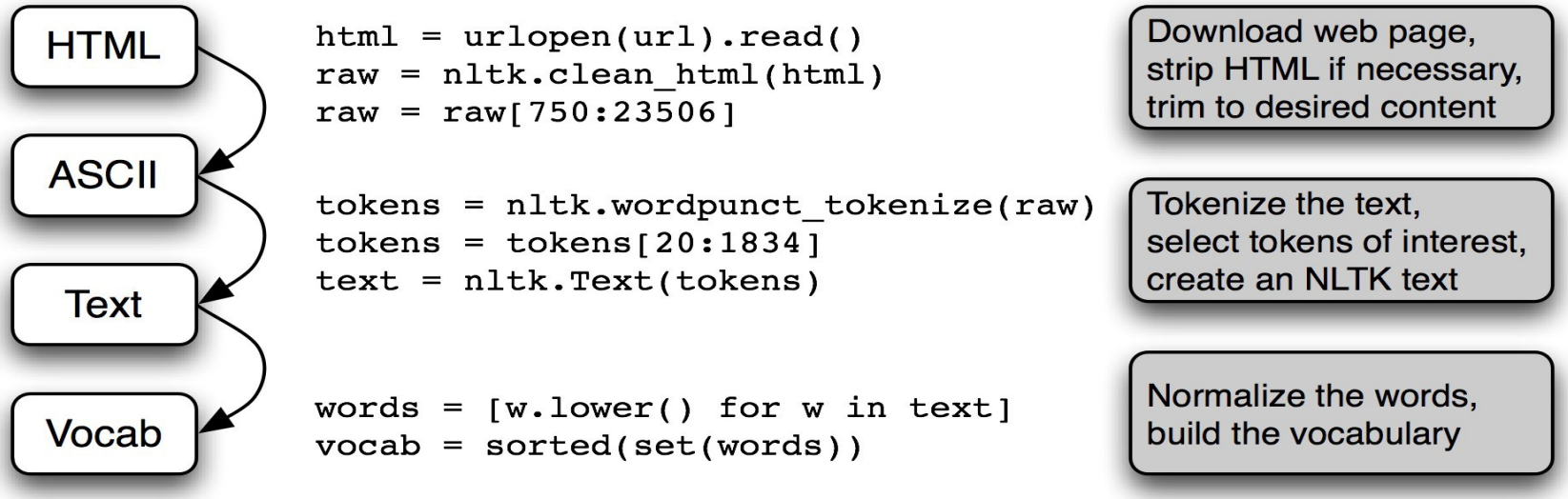
Para mayores detalles al respecto, pueden revisar el sitio oficial de Python, así como un manual para no-programadores (es posible encontrar también guías en español).

http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_2.6

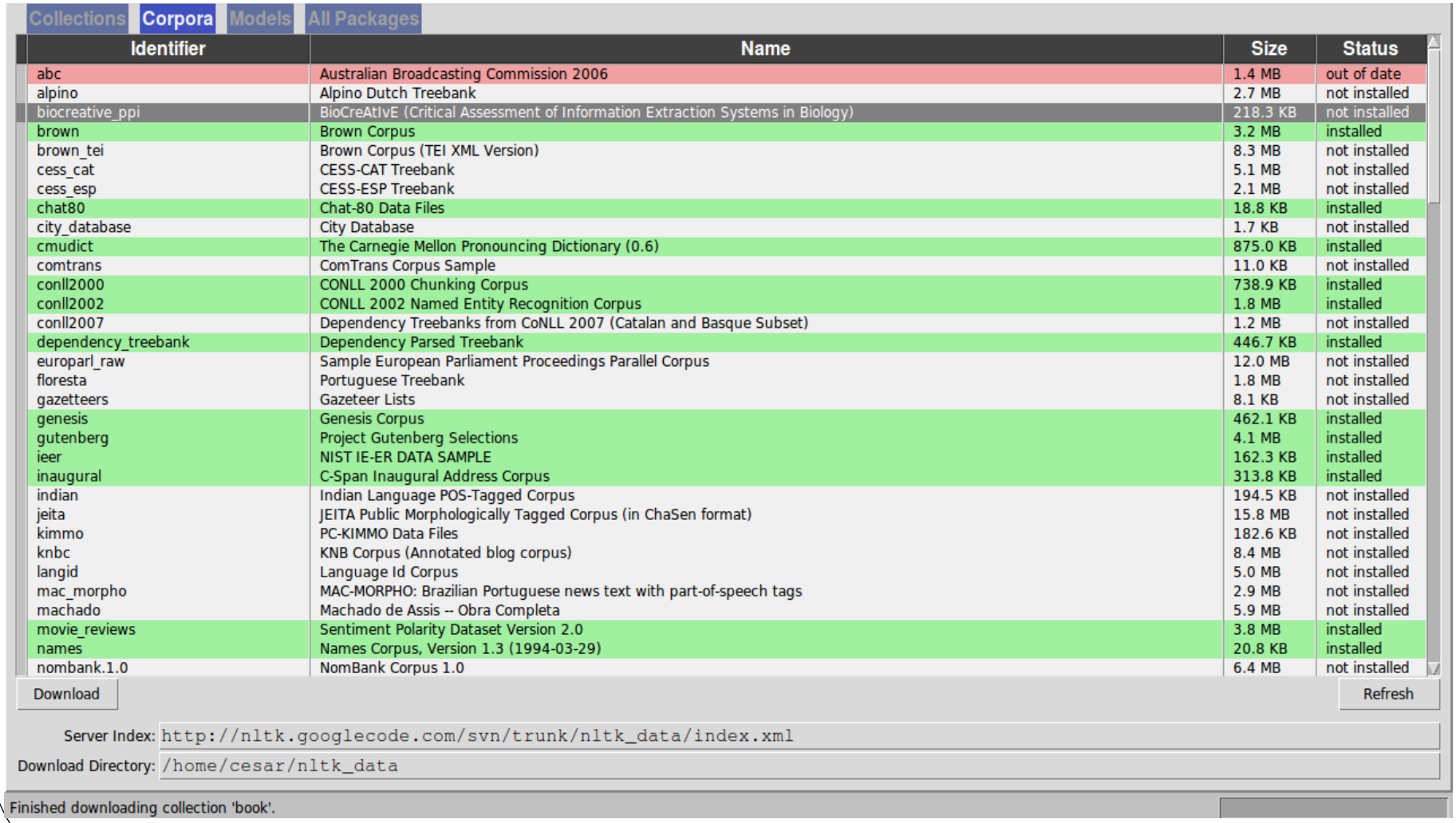
Análisis de corpus por computadora (3)

NLTK nos brinda la opción de trabajar con varios corpus etiquetados desarrollados en varias lenguas: p.e., hay un corpus en español con etiquetas morfo-sintácticas, el cual es útil para la representación de frases y oraciones a través de árboles sintácticos.

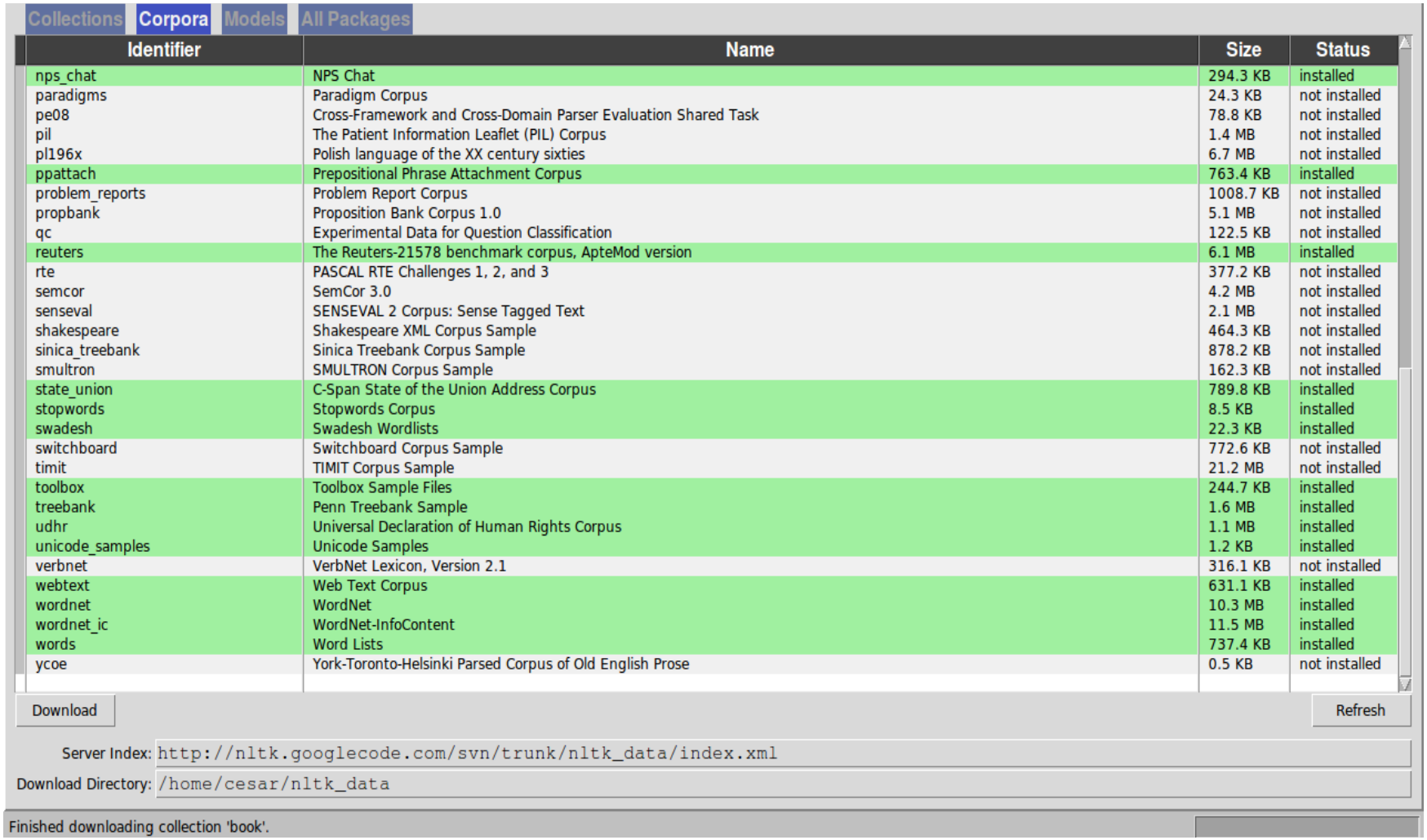
Del mismo modo, NLTK también puede trabajar con corpus sin etiquetas (esto es, *raw corpora*). Si nuestras consultas son simples (esto es, no requerimos de un análisis detallado), podemos aprovechar esta posibilidad.



Corpora anotados en NLTK (1)



Corpora anotados en NLTK (2)



Collections | Corpora | Models | All Packages

Identifier	Name	Size	Status
nps_chat	NPS Chat	294.3 KB	installed
paradigms	Paradigm Corpus	24.3 KB	not installed
pe08	Cross-Framework and Cross-Domain Parser Evaluation Shared Task	78.8 KB	not installed
pil	The Patient Information Leaflet (PIL) Corpus	1.4 MB	not installed
pl196x	Polish language of the XX century sixties	6.7 MB	not installed
ppattach	Prepositional Phrase Attachment Corpus	763.4 KB	installed
problem_reports	Problem Report Corpus	1008.7 KB	not installed
propbank	Proposition Bank Corpus 1.0	5.1 MB	not installed
qc	Experimental Data for Question Classification	122.5 KB	not installed
reuters	The Reuters-21578 benchmark corpus, ApteMod version	6.1 MB	installed
rte	PASCAL RTE Challenges 1, 2, and 3	377.2 KB	not installed
semcor	SemCor 3.0	4.2 MB	not installed
senseval	SENSEVAL 2 Corpus: Sense Tagged Text	2.1 MB	not installed
shakespeare	Shakespeare XML Corpus Sample	464.3 KB	not installed
sinica_treebank	Sinica Treebank Corpus Sample	878.2 KB	not installed
smultron	SMULTRON Corpus Sample	162.3 KB	not installed
state_union	C-Span State of the Union Address Corpus	789.8 KB	installed
stopwords	Stopwords Corpus	8.5 KB	installed
swadesh	Swadesh Wordlists	22.3 KB	installed
switchboard	Switchboard Corpus Sample	772.6 KB	not installed
timit	TIMIT Corpus Sample	21.2 MB	not installed
toolbox	Toolbox Sample Files	244.7 KB	installed
treebank	Penn Treebank Sample	1.6 MB	installed
udhr	Universal Declaration of Human Rights Corpus	1.1 MB	installed
unicode_samples	Unicode Samples	1.2 KB	installed
verbnet	VerbNet Lexicon, Version 2.1	316.1 KB	not installed
webtext	Web Text Corpus	631.1 KB	installed
wordnet	WordNet	10.3 MB	installed
wordnet_ic	WordNet-InfoContent	11.5 MB	installed
words	Word Lists	737.4 KB	installed
ycoe	York-Toronto-Helsinki Parsed Corpus of Old English Prose	0.5 KB	not installed

Download | Refresh

Server Index: http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml

Download Directory: /home/cesar/nltk_data

Finished downloading collection 'book'.

Primera prueba: búsqueda de palabras (1)

Utilicemos uno de los primeros *demos* que ofrece NLTK para hacer búsquedas de palabras:

```
from nltk.book import *
```

Tenemos algunos textos en los cuales se pueden hacer concordancias "de juguete", de forma automática:

```
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Primera prueba: búsqueda de palabras (2)

¿Qué podemos hacer?:

1. Buscar concordancias:
“nombre_del_Texto”.concordance("Palabra/Palabras_a_buscar")

2. Comparar concordancias entre textos:
“Nombre_del_texto”.similar("Palabra/Palabras_a_buscar")

3. Graficar cuál es la regularidad de uso (o dispersión) de una palabra dentro de un documento:
“Nombre_del_texto”.dispersion_plot(["Palabra/Palabras_a_buscar"])

Segunda prueba: tokenización (1)

¿Estoy limitado a consultar únicamente estos textos? Respuesta rápida:

¡NO!

Podemos hacer estas mismas consultas en otros textos, pero requerimos primero ajustarlos un poco para que funcione este proceso, en específico:

1. ¿Qué es nuestro documento para la máquina: una cadena de caracteres, o un listado de palabras?
2. Si es una lista de palabras, ¿están “tokenizadas” o no?
3. ¿Qué tengo que hacer para “tokenizar” mi texto?

Segunda prueba: tokenización (2)

La **tokenización** es un proceso por el cual “editamos” las palabras insertas en un texto, de modo que podamos ubicar tanto las formas canónicas (o **types**) de tales palabras, como sus variantes flexionadas o conjugadas (esto es, **tokens**). Veamos el siguiente ejemplo:

¿Cuántas palabras hay aquí?

Juan miró a la chava que miraba hacia la puerta.

Types: 8

Tokens: 10

Segunda prueba: tokenización (3)

Para tokenizar un texto en NLTK, ocupamos la siguiente instrucción:

```
>>> tokens = nltk.word_tokenize(Text_Ariel01)
```

¿Qué clase de objetos tenemos en nuestra lista? Según NLTK, ahora los identificamos como *tokens*:

```
>>> type(tokens)
<type 'list'>
```

¿Cuantos tenemos? :

```
>>> len(tokens)
26405
```

Segunda prueba: tokenización (4)

Nuestro resultado es el siguiente:

tokens[:100]

'THE', 'DEVELOPMENT', 'OF', 'PERSON', 'AGREEMENT', 'MARKERS', ':', 'FROM', 'PRONOUNS', 'TO', 'HIGHER', 'ACCESSIBILITY', 'MARKERS', '*', 'Mira', 'Ariel', '(', '2000', ')', 'Tel-Aviv', 'University', '1', '.', 'From', 'free', 'pronouns', 'to', 'verbal', 'agreement', ':', 'Introduction', 'Grammatical', 'items', 'often', 'begin', 'their', 'linguistic', 'life', 'as', 'regular', 'lexical', 'items', '(', 'Meillet', '1912', ')', '.', 'Grammaticization', 'is', 'said', 'to', 'have', 'occurred', 'when', 'the', 'position', 'of', 'such', 'lexemes', '(', 'or', 'even', 'phrases', ')', 'becomes', 'fixed', ',', 'their', 'meaning', 'is', 'generalized/bleached', ',', 'their', 'domain', 'of', 'applicability', 'enlarged', ',', 'significantly', 'raising', 'their', 'frequency', ',', 'their', 'form', 'modified', '(', 'usually', 'phonetically', 'reduced', ')', ',', 'and', 'their', 'occurrence', 'made', 'obligatory', ',', 'even', 'when'

Segunda prueba: tokenización (5)

Un atajo: podemos tokenizar directamente nuestro documento con las siguientes funciones:

```
>>> import ntlk
>>> Text_Ariel01 = nltk.Text(tokens)
>>> type(Text_Ariel01)
  class 'nltk.text.Text'
```

Tercera prueba: concordancias (1)

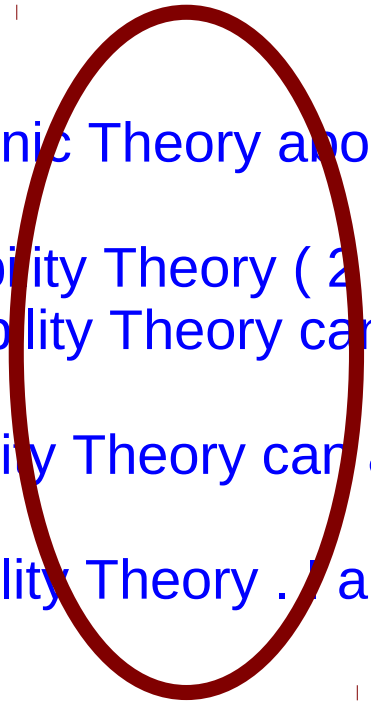
Se pueden obtener algunos candidatos a concordancias, p.e.:

```
>>> Text_Ariel01.collocations()
Building collocations list
third person; Accessibility Theory; typological markedness; verbal
forms; highly accessible; third persons; person referents;
first/second persons; University Press; John Benjamins;
second person;
high accessibility; first/second person; full NPs; frequency-driven
morphologization; agreement markers; verbal agreement; zero subjects;
future tense; discourse topic
```

Tercera prueba: concordancias (2)

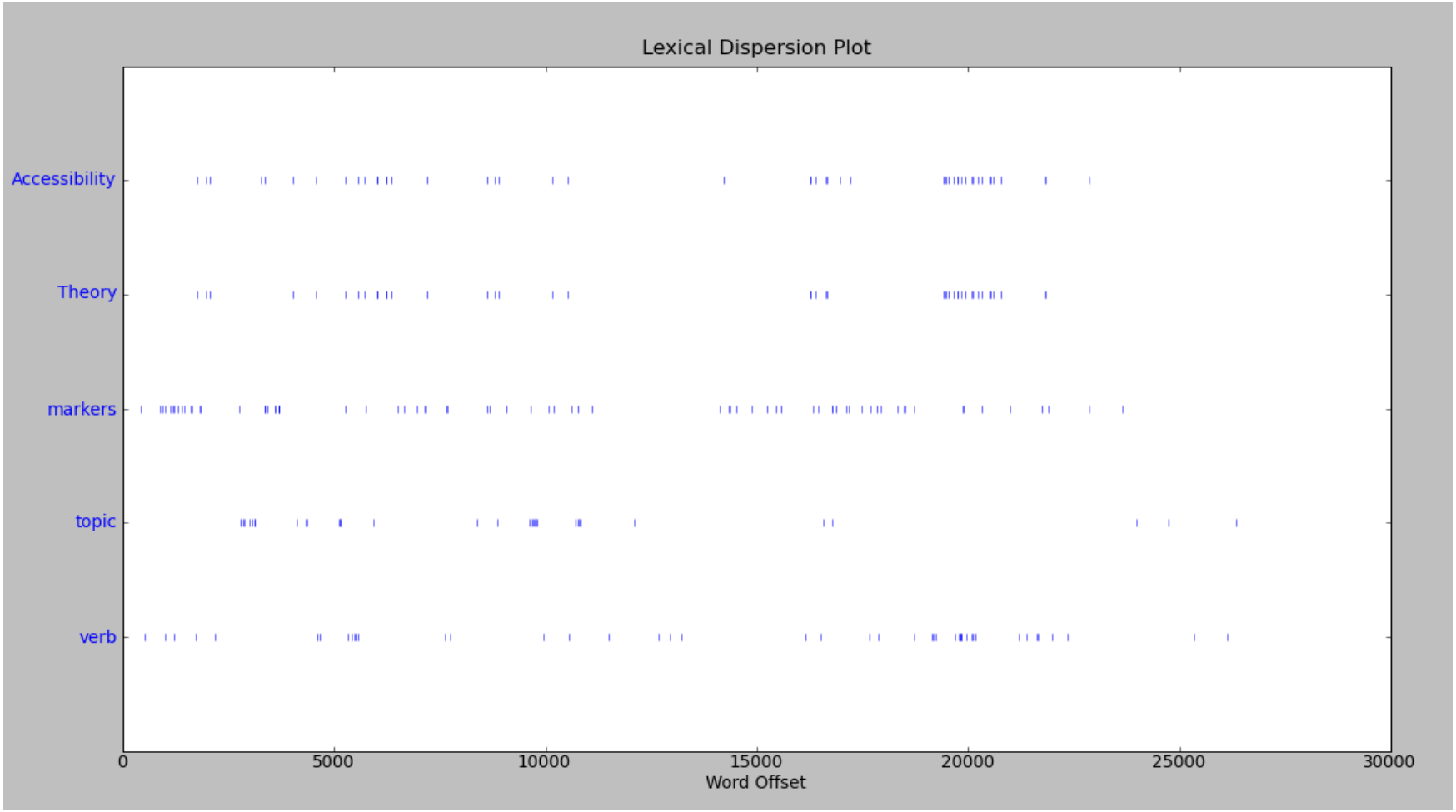
Podemos obtener concordancias con el proceso siguiente:

```
>>> Text_Ariel01.concordance("Theory")
Building index...
Displaying 25 of 64 matches:
es this insight into his synchronic Theory about pronoun uses , cites 18th and
achment ( 2 . 1 ) and accessibility Theory ( 2 . 2 ) . I will suggest that bot
argument will be that Accessibility Theory can resolve both questions in a uni
d . I argue that only Accessibility Theory can account for this new developmen
 account , but not to Accessibility Theory . I also specify what would count a
```



Tercera prueba: concordancias (3)

En esta gráfica, podemos ver la distribución de las palabras dentro del texto de Ariel. Algunas son muy recurrentes, y otras no tanto:



Cuarta prueba: identificando raíces y lemas (1)

Otro tipo de “edición” que podemos hacer dentro de nuestros textos es ajustar la flexión de nuestras palabras (sean nombres, verbos, artículos, adjetivos, etc.), a una forma canónica, incluso reduciéndola a su raíz básica. Esto se hace usando el **algoritmo de Porter**.

Algoritmo de Porter: Martin Porter desarrolló un algoritmo que, automáticamente, permite identificar asociar la raíz (o stemmer) de un conjunto de palabras similares. Vean el siguiente sitio WEB:

http://tartarus.org/~martin/PorterStemmer/def.txt

Cuarta prueba: identificando raíces y lemas (2)

Porter's algorithm

- Commonest algorithm for stemming English
  - Results suggest it's at least as good as other stemming options
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

Cuarta prueba: identificando raíces y lemas (3)

Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

- Weight of word sensitive rules
- *(m>1) EMENT*
  - *replacement* *replac*
  - *cement* *cement*

Cuarta prueba: identificando raíces y lemas (4)

Regulars and Irregulars

- Ok so it gets a little complicated by the fact that some words misbehave (refuse to follow the rules)
  - Mouse/mice, goose/geese, ox/oxen
  - Go/went, fly/flew
- The terms regular and irregular will be used to refer to words that follow the rules and those that don't.

Cuarta prueba: identificando raíces y lemas (5)

Morphological Parsing: Goal

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	'ducks'
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	'duck'
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	'I drink'
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	'I sing'
goose	goose +N +Sg	canto	canto +N +Masc +Sg	'song'
goose	goose +V	puse	poner +V +Perf +1P +Sg	'I was able'
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	'he/she came'
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	'wine'
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	'place'
caught	catch +V +Past			

Figure 3.2 Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

Cuarta prueba: identificando raíces y lemas (6)

Veamos qué podemos obtener aplicando el algoritmo de Porter al texto de Ariel:

```
>>> Text_Ariel01 = open 'Escritorio/ariel01.txt', 'rU' .read()
>>> import nltk
>>> import re
>>> Tokens_Ariel01 = nltk.word_tokenize(Text_Ariel01)
>>> porter = nltk.PorterStemmer()
>>> [porter.stem(t) for t in Tokens_Ariel01]
```

Cuarta prueba: identificando raíces y lemas (7)

El resultado es:

['THE', 'DEVELOP', 'OF', 'PERSON', 'AGREEMENT', 'MARKER', ':', 'FROM', 'PRONOUN', 'TO', 'HIGHER', 'ACCESS', 'MARKER', '*', 'Mira', 'Ariel', '(', '2000', ')', 'Tel-Aviv', 'Univers', '1', '.', 'From', 'free', 'pronoun', 'to', 'verbal', 'agreement', ':', 'Introduct', 'Grammat', 'item', 'often', 'begin', 'their', 'linguist', 'life', 'as', 'regular', 'lexic', 'item', '(', 'Meillet', '1912', ')', '.', 'Grammatic', 'is', 'said', 'to', 'have', 'occur', 'when', 'the', 'posit', 'of', 'such', 'lexem', '(', 'or', 'even', 'phrase', ')', 'becom', 'fix', ',', 'their', 'mean', 'is', 'generalized/bleach', ',', 'their', 'domain', 'of', 'applic', 'enlarg', ',', 'significantli', 'rais', 'their', 'frequenc', ',', 'their', 'form', 'modifi', '(', 'usual', 'phonet', 'reduc', ')', ',', 'and', 'their', 'occurr', 'made', 'obligatori', ',', 'even', 'when', 'inform', 'redund', '(', 'see', 'Bybe', 'et', 'al', '.', '1994', ')', '.', ...]

Quinta prueba: lematización basada en WordNet (1)

Finalmente, también se puede lematizar un documento. Para ello, NLTK emplea como diccionario una red léxica (otros lo llamarían *ontología*) con un listado de palabras asociadas a uno o varios ítems léxicos. Esta red léxica es *WordNet.*

http://wordnet.princeton.edu/perl/webwn

La instrucción es:

```
>>> Text_Ariel01 = open('Escritorio/ariel01.txt', 'rU').read()
>>> import nltk
>>> import re
>>> wnl_Ariel01 = nltk.WordNetLemmatizer()
>>> Tokens_Ariel01 = nltk.word_tokenize(Text_Ariel01)
>>>  wnl_Ariel01.lemmatize(t) for t in Tokens_Ariel01]
```

Quinta prueba: lematización basada en WordNet (2)

El resultado es:

['THE', 'DEVELOPMENT', 'OF', 'PERSON', 'AGREEMENT', 'MARKERS', ':', 'FROM', 'PRONOUNS', 'TO', 'HIGHER', 'ACCESSIBILITY', 'MARKERS', '*', 'Mira', 'Ariel', '(', '2000', ')', 'Tel-Aviv', 'University', '1', '.', 'From', 'free', 'pronoun', 'to', 'verbal', 'agreement', ':', 'Introduction', 'Grammatical', 'item', 'often', 'begin', 'their', 'linguistic', 'life', 'as', 'regular', 'lexical', 'item', '(', 'Meillet', '1912', ')', '.', 'Grammaticization', 'is', 'said', 'to', 'have', 'occurred', 'when', 'the', 'position', 'of', 'such', 'lexeme', '(', 'or', 'even', 'phrase', ')', 'becomes', 'fixed', ',', 'their', 'meaning', 'is', 'generalized/bleached', ',', 'their', 'domain', 'of', 'applicability', 'enlarged', ',', 'significantly', 'raising', 'their', 'frequency', ',', 'their', 'form', 'modified', '(', 'usually', 'phonetically', 'reduced', ')', ',', 'and', 'their', 'occurrence', 'made', 'obligatory', ',', 'even', 'when', 'informationally', 'redundant', '(', 'see', 'Bybee', 'et', 'al', '.', '1994', ')', '.' ...]

Gracias por su atención

Blog del curso: http://discurso-uaq.weebly.com/