TOP 20 NLTK

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

>>> texts()

>>> text3.concordance("string")
```
ej. god

Intentar hacerlo sobre una oración expecificada por nosotros, explicar objetos, Text
http://www.nltk.org/_modules/nltk/text.html#Text.concordance
http://stackoverflow.com/questions/29110950/python-concordance-command-in-nltk

```
>>> text3.similar("string")

>>> text3.common_context(["god", "said"])

>>> text4.dispersion_plot(["citizens","democracy", "freedom"])

# Each stripe represents the occurrence of a word, each row
represents the entire text.

>>> len(textN), textN[144]

>>> len(set(textN)) #¿Qué hace? Buscar la documentación de set().
```

Ejercicio: escribir una función que devuelva la diversidad léxica.

Some Statics...

```
>>> fdist = FreqDist(text1)

>>> fdist
```

¿Qué estructura de datos nos devuelve *fdist*? Buscar otros métodos de esa estructura y probarlos.

```
>>> vocabulary = fdist.keys()

>>> vocabulary[:50]

>>> fdist.items()

>>> fdist.tabulate(50)

>>> fdist.plot(50)

>>> fdist.plot(50, cumulative=True)
```

```
>>> fdist.freq(word)

>>> fdist.max()

>>> help(fdist.plot) #:q para salir
```

Descubriendo el telón.

```
>>> text4.collocations()
```

http://www.nltk.org/_modules/nltk/collocations.html

```
>>> nltk.chat.chatbots()
```

http://www.nltk.org/api/nltk.chat.html
http://www.nltk.org/_modules/nltk/chat/suntsu.html#suntsu_chat

ACCESS CORPUS

```
>>> from nltk.corpus import cess_esp as esp

>>> esp.fileids()

>>> esp.categories()

>>> esp.raw()

>>> esp.words()

>>> esp.tagged_words()

>>> esp.sents()

>>> esp.tagged_sents()

>>> esp.paras()

>>> esp.tagged_paras()

>>> esp.readme()
```

Example	Description
fileids()	The files of the corpus
fileids([categories])	The files of the corpus corresponding to these categories
categories()	The categories of the corpus
categories([fileids])	The categories of the corpus corresponding to these files
raw()	The raw content of the corpus
raw(fileids=[f1,f2,f3])	The raw content of the specified files
raw(categories=[c1,c2])	The raw content of the specified categories
words()	The words of the whole corpus
words(fileids=[f1,f2,f3])	The words of the specified fileids
words(categories=[c1,c2])	The words of the specified categories
sents()	The sentences of the specified categories
sents(fileids=[f1,f2,f3])	The sentences of the specified fileids
sents(categories=[c1,c2])	The sentences of the specified categories
abspath(fileid)	The location of the given file on disk
encoding(fileid)	The encoding of the file (if known)
open(fileid)	Open a stream for reading the given corpus file
root()	The path to the root of locally installed corpus
readme()	The contents of the README file of the corpus

```
>>> from nltk.corpus import names, stopwords, words
>>> names.fileids()
>>> stopwords.fileids()
```

LOADING YOUR OWN CORPUS:

```
>>> from nltk.corpus import PlainTextCorpusReader
>>> corpus = PlaintextCorpusReader(os.getcwd(), "[a-zA-Z0-9]*.txt")
>>> corpus.fileids()
>>> corpus.words("facundo.txt")
>>> from nltk.text import Text
>>> facundo = Text(corpus.words("facundo.txt"))
>>> facundo.concordance("salvaje")
>>> corpus.sents("facundo.txt")
```

To create a new corpus reader, you will first need to look up the signature for that corpus reader's constructor. Different corpus readers have different constructor signatures, but most of the constructor signatures have the basic form:

SomeCorpusReader(root, files, ...options...)

Where root is an absolute path to the directory containing the corpus data files; files is either a list of file names (relative to root) or a regexp specifying which files should be included; and options are additional reader-specific options. For example, we can create a customized corpus reader for the genesis corpus that uses a different sentence tokenizer as follows:

```
>>> # Find the directory where the corpus lives.
>>> genesis_dir = nltk.data.find('corpora/genesis')
>>> # Create our custom sentence tokenizer.
>>> my_sent_tokenizer = nltk.RegexpTokenizer('[^.!?]+')
>>> # Create the new corpus reader object.
>>> my_genesis = nltk.corpus.PlaintextCorpusReader(
...     genesis_dir, '.*\.txt', sent_tokenizer=my_sent_tokenizer)
>>> # Use the new corpus reader object.
>>> print(my_genesis.sents('english-kjv.txt')[0]) # doctest:
+NORMALIZE_WHITESPACE
['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven',
 'and', 'the', 'earth']
```

If you wish to read your own plaintext corpus, which is stored in the directory '/usr/share/some-corpus', then you can create a corpus reader for it with:

```
>>> my_corpus = nltk.corpus.PlaintextCorpusReader(
...     '/usr/share/some-corpus', '.*\.txt') # doctest: +SKIP
```

For a complete list of corpus reader subclasses, see the API documentation for *nltk.corpus.reader*.

Estilística y análisis del discurso ¿?

A conditional frequency distribution is a collection of frequency distributions, each one for a different “condition”. Whereas FreqDist() takes a simple list as input, ConditionalFreqDist() takes a list of pairs. Each pair has the form (condition, event).

1)

```
>>> from nltk.corpus import brown
>>> news_text = brown.words(categories='news')
>>> fdist = nltk.FreqDist([w.lower() for w in news_text])
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> for m in modals:
...    print m + ':', fdist[m],
```

2)

```
>>> cfd = nltk.ConditionalFreqDist(
...    (genre, word)
...    for genre in brown.categories()
...    for word in brown.words(categories=genre))
>>>  genres  =  ['news',  'religion',  'hobbies',  'science_fiction',
'romance', 'humor']
>>> modals = ['can', 'could', 'may', 'might', 'must', 'will']
>>> cfd.tabulate(conditions=genres, samples=modals)
```

3)

```
>>> cfd = nltk.ConditionalFreqDist(
...    (target, file[:4])
...    for fileid in inaugural.fileids()
...    for w in inaugural.words(fileid)
...    for target in ['america', 'citizen']
...    if w.lower().startswith(target))
>>> cfd.plot()
```

Example	Description
`cfdist = ConditionalFreqDist(pairs)`	Create a conditional frequency distribution from a list of pairs
`cfdist.conditions()`	Alphabetically sorted list of conditions
`cfdist[condition]`	The frequency distribution for this condition
`cfdist[condition][sample]`	Frequency for the given sample for this condition
`cfdist.tabulate()`	Tabulate the conditional frequency distribution
`cfdist.tabulate(samples, conditions)`	Tabulation limited to the specified samples and conditions
`cfdist.plot()`	Graphical plot of the conditional frequency distribution
`cfdist.plot(samples, conditions)`	Graphical plot limited to the specified samples and conditions
`cfdist1 < cfdist2`	Test if samples in `cfdist1` occur less frequently than in `cfdist2`

GENERATE RANDOM TEXT WITH BIGRAMS → Script

BLLIP RERANKING PARSER

```
>>> from bllipparser import RerankingParser
>>>from bllipparser.ModelFetcher import download_and_install_model
>>> model_dir = download_and_install_model('WSJ', '/tmp/models')
>>> rrp = RerankingParser.from_unified_model_dir(model_dir)
>>> rrp.simple_parse('This is simple.')
```

Interactive shell

```
$python -mbllipparser model
```