Clase 8

Repaso
NLTK

nltk NLTK es un módulo de Python que contiene muchas funciones diseñadas para su uso en el análisis lingüístico de documentos y en el procesamiento de lenguaje natural. Para poder utilizar las funciones de este módulo primero debemos importarlo con `import`.

download() NLTK es un módulo muy grande, cuando lo descargan e instalan (si instalaron Anaconda, el paquete lo hizo por ustedes) no se descarga NLTK en su totalidad, también está modulado y las partes las pueden encontrar en línea. La función `download()` de `nltk` abrirá un pequeño navegador con el que pueden descargar módulos de NLTK.

punkt Uno de los módulos que vamos a utilizar (y que conseguimos con la función `download()` es `punkt`. Éste módulo contiene modelos para la tokenización de textos.

word_tokenize() Tras descargar el módulo `punkt` se le agrega a `nltk` la función `word_tokenize`. La función recibe como parámetro el texto que se quiere tokenizar, y como un segundo parámetro opcional el idioma (hay idiomas que tienen diferentes reglas de tokenización), y regresa una lista con todos los tokens del texto. Algo muy similar a lo que logramos con nuestro tokenizador de expresiones regulares.

Repaso

Conjuntos y conteo

set() Vimos también la función **`set()`** que convierte una lista en un conjunto (un *set*. Los conjuntos tienen propiedades diferentes a las listas, NO están ordenados, pero sus elementos tampoco se pueden repetir. Para los ejemplos que usamos en la clase, esa propiedad nos fue de utilidad para eliminar todos los tokens repetidos.

count() Ésta es una función de las listas. Nos permite saber cuántos elementos con un mismo valor contiene la lista, el valor se recibe como parámetro de la función.

Repaso

Funciones

def ... (): Para definir funciones en Python utilizamos la instrucción **`def`** seguido del nombre de nuestra función y sus paréntesis, dentro de los paréntesis podemos poner el nombre de los parámetros que esperamos recibir y que se podrán usar en el cuerpo de la función como variables, por último ponemos dos puntos para comenzar el bloque te código que será el contenido de nuestra función.

return Si queremos que nuestra función regrese un valor (al igual otras funciones que hemos visto como **`read()`**) es necesario que utilicemos la instrucción **`return`** seguido del valor que queremos regresar. Con esto podemos asignar el resultado de nuestra función a una variable.

Repaso

nltk.Text

Text() Con ésta función de **`nltk`** podemos convertir una lista de tokens en un **`Text`** de NLTK. Este es un tipo de variable (como los hay listas, conjuntos, números, etc.), y como tal tiene funciones propias diseñadas para el análisis de textos y el procesamiento de lenguaje natural.

- {} Las llaves, una nueva herramienta para las expresiones regulares. Son repetidores (como el + o el *) la diferencia está en que dentro de las llaves podemos especificar la cantidad de veces que queremos que algo se repita. Se establece con un número, pero también podemos especificar rangos (entre 10 y 30, por ejemplo) si separamos los números con una coma dentro de las llaves.

Repaso

nltk.Text

concordance() Esta es una función del **`Text`** de NLTK. Nos permite obtener las concordancias de una palabra, es decir, la palabra en su contexto. La palabra se usa como parámetro de la función. Además, tiene los parámetros opcionales **`width=`** y **`lines=`** para cambiar la cantidad de caracteres que se toman de cada lado de la palabra y la cantidad máxima de resultados que muestra. Esta función es similar a **`print()`** y NO devuelve ningún valor.

similar() Otra función del **`Text`** de NLTK. Nos permite obtener las palabras similares a una palabra que recibe como parámetro. Para encontrar dichas palabras similares, compara los contextos de las palabras y regresa la que tienen los contextos más parecidos.

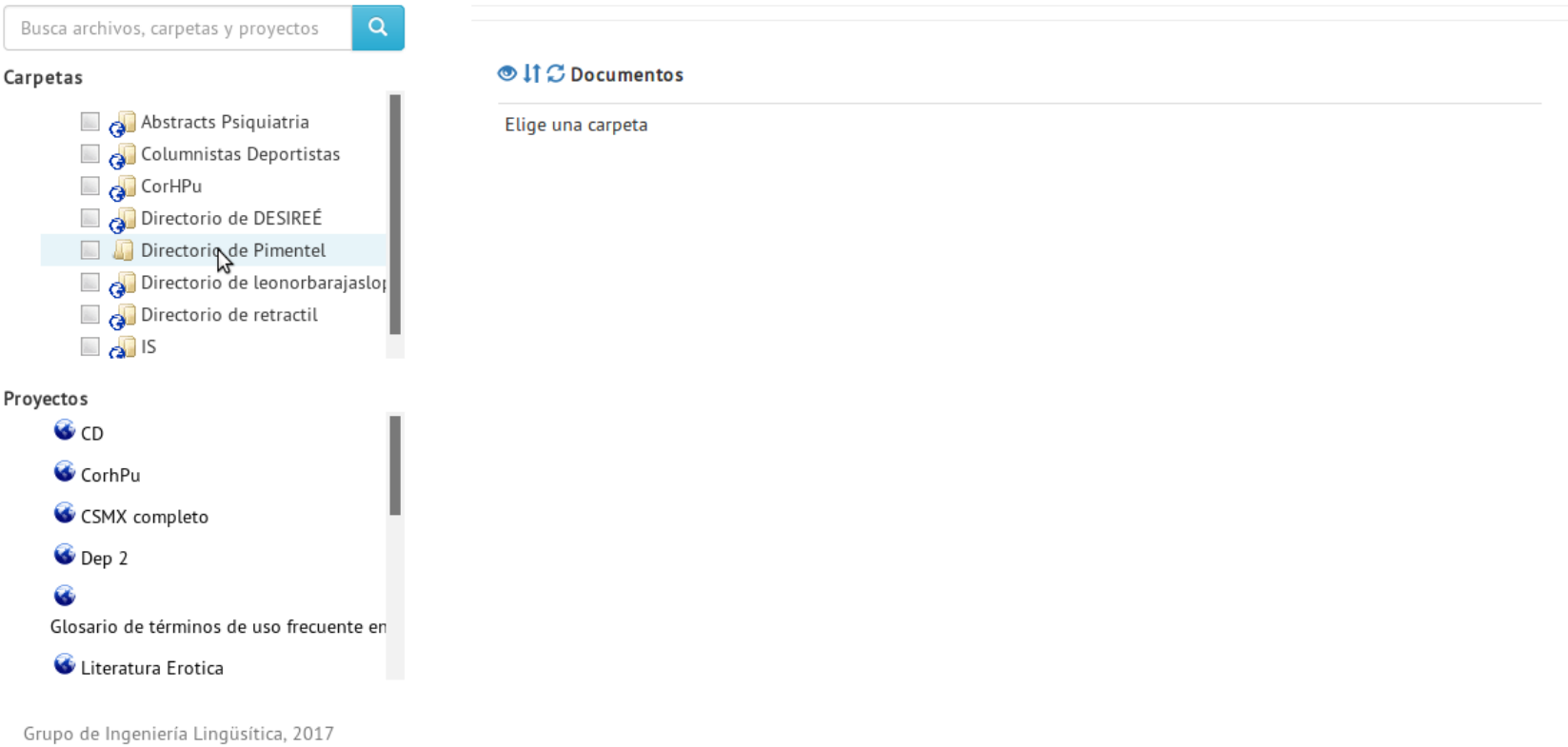
Herramientas

http://www.corpus.unam.mx/geco

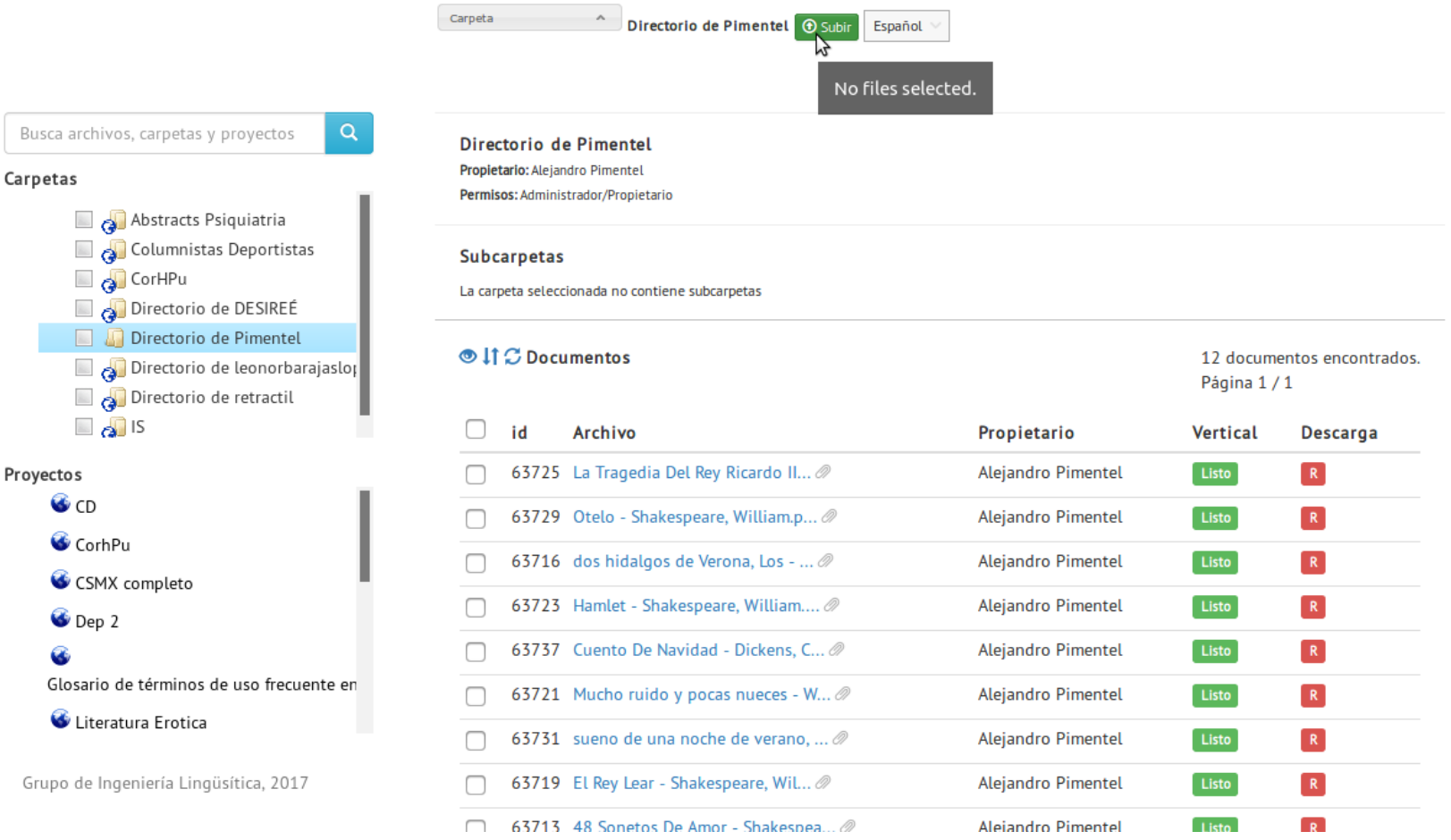
Herramientas

Herramientas

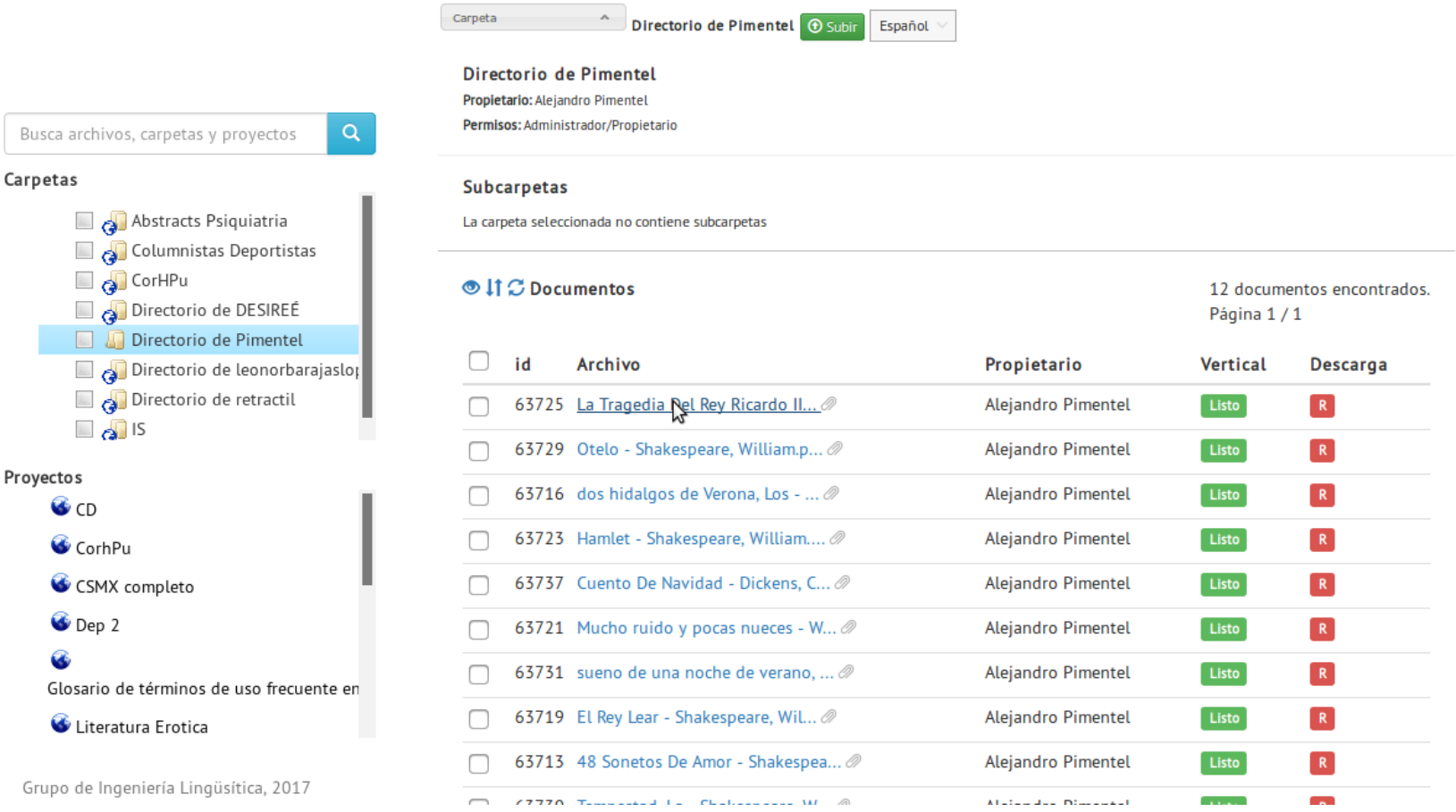
Herramientas



Herramientas



Herramientas



Ejercicio 17

Notarán que el texto, si bien es confiable en el contenido, su formato es horrible. Además, se va mucha basura, en particular los números de página.

- Limpien el texto de los números de página, y de todos los espacios extras que hay entre las palabras.
- También tokenizen el texto y obtengan un `Text` de NLTK para conseguir concordancias y palabras similares. Pueden elegir las palabras que quieran, sean creativos.

NLTK

Palabras similares

- Para este tipo de funciones, entre más texto se tenga para hacer el análisis es mejor.
- Hasta ahora, hemos usado un texto corto para los ejemplos, probemos ahora con el conjunto de todos los que tenemos.

```
import nltk

carpeta_nombre="Documentos\\"
archivo_nombre="DOF_P_IFT_291116_672_Acc.txt"


with open(carpeta_nombre+archivo_nombre,"r") as archivo:
        texto=archivo.read()

tokens=nltk.word_tokenize(texto,"spanish")

texto_nltk=nltk.Text(tokens)
texto_nltk.similar("artículo")
```

NLTK

Contextos comunes

- Por supuesto, podemos ver cuál es el contexto que comparten las palabras similares.

```
tokens=nltk.word_tokenize(texto,"spanish")

texto_nltk=nltk.Text(tokens)
texto_nltk.similar("artículo")
print()
texto_nltk.common_contexts(["artículo","instituto"])
```

- La función `print()` es para separar los resultados y que sea más claro el contenido de cada parte.

NLTK

Dispersión

- Otra función muy interesante es **`dispersion_plot()`**
- Esta función muestra una gráfica con la aparición de una lista de palabras buscadas a lo largo de todo el texto.

```
tokens=nltk.word_tokenize(texto,"spanish")
texto_nltk=nltk.Text(tokens)

lista_palabras=["Instituto","Ley","Elija","ley"]
texto_nltk.dispersion_plot(lista_palabras)
```

NLTK

Distribución de frecuencias

```
import nltk

carpeta_nombre="Documentos\\"
archivo_nombre="P_IFT_290216_73_Acc.txt"

with open(carpeta_nombre+archivo_nombre,"r") as archivo:
        texto=archivo.read()

tokens=nltk.word_tokenize(texto,"spanish")

texto_nltk=nltk.Text(tokens)

distribucion=nltk.FreqDist(texto_nltk)

lista_frecuencias=distribucion.most_common()
print(lista_frecuencias)
```

NLTK

Distribución de frecuencias

- De la distribución de frecuencias también podemos obtener la frecuencia de una palabra en particular.
- Como podrán ver, esto se logra de manera similar a los índices de una lista, pero en lugar del índice (el número que indica la posición dentro de la lista) se usa la palabra misma como texto.

```
# A esta altura ya tenemos la lista de tokens en "tokens".

texto_nltk=nltk.Text(tokens)

distribucion=nltk.FreqDist(texto_nltk)

print(distribucion["Instituto"])
```

Diccionarios

en Python

- Esas "listas" que en lugar de usar índices usan palabras, se llaman diccionarios. Es otra herramienta que tiene Python.
- OJO, el resultado de la función **`FreqDist()`** de NLTK en realidad no es un diccionario, ya que tiene funciones propias de NLTK, pero se comporta como un diccionario para obtener su contenido usando palabras.

```
info={"nombre":"Mi nombre","apellido":"Mi apellido"}
info["edad"]=100
info["curso"]="Python"

for dato in info:
        print(dato,":",info[dato])
```