082057 – Procesamiento del Lenguaje Natural

Lab 1

NLTK y Contar Palabras Corpora

1 IMPORTANTE

El ejercicio está diseñado para ayudarlo a seguir su progreso desde el principio del módulo. No se preocupe si no termina todas las preguntas en este ejercicio en el laboratorio.

Revise las partes restantes después del laboratorio y asegúrese de comprender los conceptos.

Dígale a su instructor cuánto progreso ha realizado y las dificultades que haya tenido. No se asignarán calificaciones ni calificaciones formales para este trabajo de laboratorio. Sin embargo, es obligatorio asegurarse de asistir a este laboratorio.

Ayude a sus compañeros de clase que pueden ser nuevos en Python. Puede discutir con otros compañeros cómo abordar los problemas, pero asegúrese de tener una buena comprensión de los métodos. Más recursos sobre Python serán compartidos en el campus virtual.

2 De qué se trata este laboratorio?

Este laboratorio explora algunos de los conceptos vistos en clase. Utilizará el Kit de herramientas de lenguaje natural (NLTK) para este laboratorio. Este primer laboratorio le presenta NLTK y lo lleva a través de los métodos de la biblioteca para acceder a los Corpus y trabajar con texto. En particular, este laboratorio se centra en calcular el recuento de palabras de un corpus.

3 Usando NLTK

NLTK es un set de librerías de Python para hacer NLP, muy utilizado entre investigadores y algo en industria (veremos que hay otros más adelante, más aplicables a industria pero menos "customizables").

Para empezar, abra una terminal en su computadora y tipee:

```
$ python
```

Este commando inicia el "Python interpreter". Si está trabajando en Windows, puede usar el Python 3.x IDLE GUI (desde el menú de INICIO).

Ahora puede empezar a usar comandos Python en el promt de Python >>>. Por ejemplo, para sumar dos número enteros, escriba:

```
>>> 6 + 4

10
```

Hay que instalar la librería de NLTK antes de empezar a usarla, para esto crearemos un entorno de conda y en ese entorno activo, instalaremos las librerías que necesitemos:

https://docs.conda.io/en/latest/miniconda.html

Cree un entorno de conda para Python 3.x (llamelo 'nlp-utn')

Active el entorno

Ahora debería ver algo así en su prompt:

(nlp-utn) Hernan@hermacair:

Lo que se encuentra entre paréntesis es el entorno activo de conda con el nombre que deseo llamarlo.

Ahora si, instalamos la librería de **nltk**, bien desde **conda** o bien desde el instalador de paquetes Python **pip**

Nota: La página official de NLTK es: http://nltk.org/

El próximo paso es importar las librerías de NLKT al entorno activo, y también bajar algunos corpora que vamos a usar en este lab:

```
>>> import nltk

>>> nltk.download()
```

En el dialog box que aparece, eleja que descargar de todos los disponibles en NLTK book.

Click en "download" o "descargar" y cuando todo esté listo, cierre el cuadro de diálogo.

Ahora si, desde el prompt, escribimos:

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

La lista muestra 9 textos o corpora, algunos de ellos son novelas inglesas que quizás reconozcas. El corpues de artículos de Wall Street Journal, tiene un total de 1 millon de palabras de los diarios del Wall Streen Journal (principalmente noticias financieras).

Para listar las palabras de un texto en particular, puede usar los métodos de tokens. Por ejemplo, los primeros 5 tokens en el corpus de "Moby Dick" (text1) pueden ser obtenidos escribiendo:

```
>>> text1.tokens[0:5]
[u'[', u'Moby', u'Dick', u'by', u'Herman']
```

Para hacer lo mismo en el Wall Street Journal corpus, tipeamos:

```
>>> text7.tokens[0:5]
[u'Pierre', u'Vinken', u',', u'61', u'years']
```

Note that you can see the list of corpora again by typing:

Notese que puede ver la lista de corpora otra vez, tipeando:

```
>>> texts()
```

También puede contar el número de ocurrencias de una palabra en particular, usando el método “count” y un parámetro con la palabra:

```
>>> text1.count("Moby")

84
```

4 Strings y expresiones regulares

Antes de trabajar con un corpora grande, vamos a tomar una oración simple y mirar a los problemas que deberían tenerse en cuenta cuando procesamos strings.

Trate de ejecutar lo siguiente:

```
>>> mysent = ‘‘The cat sat on the mat.’’

>>> from nltk import word tokenize


>>> mysent_tokens = word tokenize(mysent)

>>> mysent_tokens

[‘The’, ‘cat’, ‘sat’, ‘on’, ‘the’, ‘mat’, ‘.’]

>>> len(mysent tokens)
7
```

word_tokenize es un método built-in que “tokeniza” una cadena de **lenguaje natural**. **Notese** que la tokenización no está basada solamente en separar por espacios. len(mysent_tokens) retorna el tamaño de la lista de tokens dado como argumento

A veces no queremos considerar las puntuaciones (punctuations) de la misma manera que los otros tokens (por ejemplo cuando queremos saber el total de palabras que hay en un texto). Para

Para eliminar signos de puntuación, puede usar expresiones regulares. Una expresión regular le permite hacer coincidir una cadena con un patrón. Por ejemplo, puede verificar si la cadena completa coincide con algunos caracteres iniciales y finales. Es decir, la cadena comienza con 'a' o 'd' y termina con una 'k'?

O puede encontrar si una parte de la cadena coincide con algún patrón. Un ejemplo de este segundo caso sería encontrar si una cadena ‘**x’** es una subcadena de la cadena ‘**y**’. Hay algunos patrones de cadena que no pueden especificarse mediante expresiones regulares. No entraremos en detalles aquí, pero puede leer más al respecto.

Una forma común de usar expresiones regulares en Python es a través del método **re.search**. El siguiente fragmento muestra cómo buscar si "***leng***" es una subcadena del texto "**procesamiento del lenguaje natural**":

```
>>> from nltk import re
>>> m = re.search("lang", "procesamiento del lenguaje natural")
```

```
>>> m
< sre.SRE Match object at 0x10beaa920>
>>> m.start()
8
>>> m.end()
12
```

El método re.search devuelve un objeto de coincidencia si hay una coincidencia exitosa. Al llamar al inicio, los métodos de finalización en el objeto de coincidencia devuelven el índice de inicio y finalización de la cadena de consulta en la cadena dada.

Mira lo que pasa en este caso:

```
>>> n = re.search("procesan", "procesamiento del lenguaje natural")
```

```
>>> n
```

Mientras que en los casos anteriores, el patrón era un valor de cadena específico, generalmente las expresiones regulares involucran rangos y caracteres comodín, pero también hay formas abreviadas incorporadas para las expresiones de uso frecuente.

Ahora supongamos que consideramos como palabras, solo aquellos tokens que son alfanuméricos, es decir. solo contienen alfabetos, números o el hiper símbolos. La expresión regular que puede usar para este patrón es "\ w".

```
>>> mysent_tokens_nopunct = [word for word in word tokenize(mysent) if re.search("\w", word)]
>>> mysent_tokens_nopunct
['The', 'cat', 'sat', 'on', 'the', 'mat']
```

```
>>> len(mysent_tokens_nopunct)
```

```
6
```

El método set crea un set. Un set por definición, solo tendrá objetos únicos (sin repetición). Por lo tanto, cada palabra aparecerá una vez. Intente:

```
>>> set(mysent_tokens_nopunct)
```

El resultado se ve como lo que esperaba? (Pista: qué tamaño esperaba que tenga el set?)

Busque la lista de métodos para strings de Python y encuentre un método para solucionar este problema (https://docs.python.org/2/library/stdtypes.html#string-methods). (Pista: es más fácil aplicar el método antes de crear los tokens de mysent).

Una vez creados los nuevos mysent_tokens, use el set otra vez para asegurarse de que está conforme con el resultado. Llamando al método ***len*** en este set, encontrará el número de palabras únicas (también conocido como "types").

```
>>> len(set(mysent tokens nopunct))
```

```
5
```

Ahora si, estamos listos para aplicar estas técnicas a un corpora más grande, de NLKT.

```
>>> moby_dick_tokens = text1.tokens
```

```
>>> moby_dick tokens_nopunct = [word.lower() for word in moby dick tokens if re.search("\w", word)]
>>> moby dick tokens nopunct[0:5]
[u'moby', u'dick', u'by', u'herman', u'melville']
```

5 Probando sus conocimientos

Basado en lo que aprendió en las secciones previas, responda las siguientes preguntas:

** Remueva las puntuaciones y coviernta a minúsculas (lower case) antes de hacer las cuentas **

1. Cuál es el número de tokens en Moby Dick?

2. Cuál es el número de types en Moby Dick?

El type-token ratio es una medida de cuán diverso son los ítems léxicos en un texto dado. Está definido por el número de types dividido por el número de tokens. Cuánto más alto es este ratio, consideramos al texto como más diverso en palabras, también conocido como "**diversidad léxica**". En otras palabras, textos con mayor "lexical diversity" usan una gran variedad de palabras, de manera opuesta a los que usan las mismas palabras repetidamente. Para tener una intuición, suponga dos textos que tienen el mismo número de tokens, el que que tiene más types es el más diverso léxicamente.

Combinando estos dos métodos para contart types y tokens, puede encontrar el type-token ratio de Moby Dick. Haga lo mismo para el corpus del WSJ (Wall Street Journal).

3. Moby Dick type-token ratio =

4. WSJ type-token ratio is =

5. Cuál de los dos tiene más diversidad léxica?

6. Puede pensar una razón por por la cual ese corpus es más diverso que el otro?

También hablamos enc lase sobre estimar probabilidades de un corpus.

7. Cual es el "Maximum Likelikhood Estimate (MLE)" de la palabra "whale" (ballena) en Moby Dick?

$P_{\text{moby dick}}(\text{“whale”}) =$

8. Cuál es el MLE de "whale" en el corpus de WSJ?

$P_{WSJ}(\text{“whale”}) =$