Procesamiento Natural del Lenguaje

Clase N° 3 - Limpieza, normalización y tokenización

Docente: James Tomalá Robles

Volviendo al flujo de preprocesamiento

Librería NLTK

librería de Python diseñada para trabajar con el procesamiento del lenguaje natural (NLP)

Algunos procesamientos:
Tokenización: Dividir texto en palabras, frases, párrafos, etc.
Stemming y lematización: Reducir palabras a sus raíces o formas base.
Etiquetado de partes del discurso (POS tagging): Asignar etiquetas gramaticales a palabras (sustantivo, verbo, adjetivo, etc.).
Chunking: Extraer frases cortas de una oración.
Parsing: Analizar la estructura gramatical de las oraciones.

Corpus y paquetes necesarios para el procesamiento lingüístico.

Limpieza y normalización:

Limpieza del Texto

La limpieza de texto generalmente implica eliminar o corregir elementos no deseados o inconsistentes en el texto. Ej. eliminar:

1. **caracteres especiales y puntuación innecesaria**.
2. **números**.
3. **espacios en blanco adicionales**.
4. **HTML o XML tags**.

Normalización del Texto

La normalización del texto incluye técnicas para transformar el texto en una forma estándar y uniforme. Esto puede incluir:

1. **Convertir todo el texto a minúsculas** para asegurar que las comparaciones de palabras no sean sensibles a mayúsculas/minúsculas.
2. **Eliminar stopwords** para reducir el ruido.

Consulta

Una palabra común en un idioma , ¿servirá para discriminar o diferenciar un texto de otro ?

Stopword

"stopword" (o palabra vacía) se refiere a palabras muy comunes y frecuentes en un idioma que generalmente se consideran irrelevantes para el análisis textual y que se eliminan durante el preprocesamiento de datos. Estas palabras no aportan mucho valor semántico ni informativo.

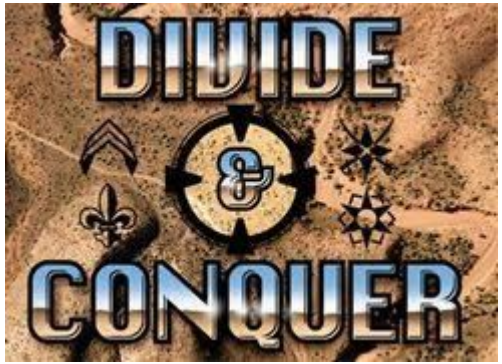
```
:
# verificar si tiene instalado el corpus del stopwords
import nltk
from nltk.corpus import stopwords

## trabajar con stopwords
nltk.download('stopwords')
stop_words = set( stopwords.words('spanish'))
# descargar corpus si no lo tiene
#nltk.download()
```

Stop Words

- a
- I
- the
- in
- of
- for
- at
- to
- on
- with
- from

Consulta

¿ Será más sencillo analizar un texto (por ejemplo encontrar patrones) considerándolo como un todo o es mejor analizar pedazos más pequeños ?



Tokenizar:

La tokenización divide el texto en partes más pequeñas para facilitar el análisis mecánico y ayudar a las máquinas a comprender el lenguaje humano.

Tipos:

Tokenización de palabras.

Tokenización por caracteres.

Tokenización por oraciones.

Algunas aplicaciones:

Traductores.

Motores de Búsqueda.

Reconocimiento de voz.

Algunos códigos para Tokenizar en python

```
texto.split(",")
```

```
## Tokenización
import nltk
nltk.download('punkt') #descarga de tokenizadores
from nltk.tokenize import word_tokenize #¿cómo se
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\56951\AppData\Roaming\nl
[nltk_data]   Package punkt is already up-to-date
```

```
## aplicar el tokenizador de palabras del nltk
words= word_tokenize(texto_sin_html)
```

```
from nltk.tokenize import TweetTokenizer
s = "@amankedia I'm going to buy a Rolexxxxxxxx watch!!! :-D #happ
tokenizer = TweetTokenizer()
tokenizer.tokenize(s)
```

```
['@amankedia',
 "I'm",
 'going',
 'to',
 'buy',
```

Tokenizando con expresiones regulares:

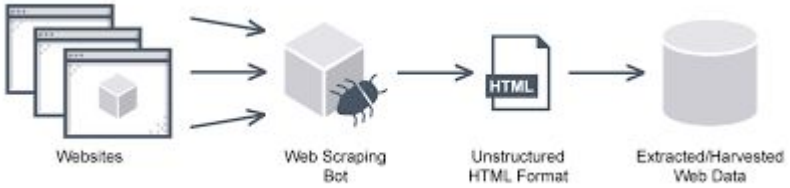
RegexpTokenizer

incluye la opción para reconocer caracteres Unicode, lo que permite trabajar con caracteres especiales del español, como las letras con acentos (á, é, í, ó, ú), la ñ y la ü

```
from nltk.tokenize import RegexpTokenizer

## Crear instancia de objeto
tokenizador = RegexpTokenizer("[^\W\d_]+")

## aplicar el método tokenize
tokens= tokenizador.tokenize(texto_ejemplo)
```

```
['DOCTYPE',
 'html',
 'html',
 'lang',
```

Caso Práctico



Ud. está trabajando en una empres inmobiliaria del cono Sur (Chile y Argentina); necesita realizar un análisis de las palabras más frecuentes de dos portales inmobiliarios.

Acceda a los datos usando web scraping

URL="https://www.portalinmobiliario.com/" # URL para Chile

URL="https://www.inmobusqueda.com.ar/" # URL para Argentina

Realice el preprocesamiento de texto aplicando, limpieza, normalización y tokenización.

Visualice los resultados usando distribución de frecuencia o nube de palabras.

Web Scraping : técnica utilizada para extraer datos de sitios web. Implica ; comúnmente utilizada para recopilar datos de una manera automatizada y estructurada que puede ser útil para análisis, investigación, minería de datos y muchas otras aplicaciones.

Otras librerías más específicas

Beautiful Soup

```
from bs4 import BeautifulSoup

#crea un objeto tipo soup
soup = BeautifulSoup(texto_ejemplo, "html.parser")

ELEMENTOS_SUPRIMIR = ('style', 'script', 'head', 'title')

def limpiar_parte_html(soup):
    # concatena todas las cadenas del html siempre y cuando n

    texto_limpio  = ' '.join([
        s for s in soup.strings
        if s.parent.name not in ELEMENTOS_SUPRIMIR
    ])
    #  retorna el texto limpio
    return texto_limpio
    #return re.sub('\s{3,}',' ', text )

texto= limpiar_parte_html(soup)

print(texto)
```

Haciendo nube de palabras

```
from wordcloud import WordCloud

### importante: el argumento debe estar en una sola cadena de texto

wordcloud = WordCloud().generate(texto_limpio)

plt.imshow(wordcloud, interpolation='bilinear' )
plt.axis("off")
plt.show()
```

Cierre

https://quizizz.com/embed/quiz/654d3f6353e96d62bbd84eee