

---

# ConTexto

*Versión 0.1*

**UCD - DNP**

**15 de septiembre de 2020**

---

## Contents

---

La librería de procesamiento y análisis de texto, ConTexto, tiene como objetivo principal proporcionar herramientas que simplifiquen las tareas y proyectos que involucren análisis de texto. La librería fue desarrollada en el lenguaje de programación de *Python* y contiene un conjunto de funciones que permiten realizar transformaciones y análisis de textos de forma simple, utilizando diferentes técnicas, para lectura y escritura de archivos de texto, incluyendo reconocimiento óptico de caracteres (OCR), limpieza de textos y remoción de palabras no deseadas para el análisis (*stop words*), traducción y corrección de textos, generación de nubes de palabras, cálculo de similitudes entre textos, entre otras, reconocidas por su buen desempeño.

La librería surge como solución a tres principales aspectos, primero, la necesidad de integrar todos los esfuerzos y desarrollos que ha hecho la Unidad de Científicos de Datos (UCD) del DNP, en proyectos relacionados con la analítica de texto, segundo, evitar reprocesos en la construcción de scripts para estas tareas, y finalmente, contribuir en la reducción de la escasez de librerías enfocadas en el análisis de texto en español que existe actualmente.

Esta página contiene toda la información relacionada con la librería, en el panel de navegación se tiene acceso a las diferentes secciones, las cuales cubren la instalación de la librería, la documentación de los módulos y funciones, ejemplos y demás información de interés.

### 2.1 Instalación ConTexto

Para la instalación de la librería se debe utilizar el gestor de paquetes `pip`, por buenas prácticas se sugiere antes de la instalación crear un entorno virtual que permita aislar las librerías y evitar conflictos de versiones con el entorno de desarrollo base del computador.

```
pip install contexto
```

**Para Python 3.8x usar:**

```
pip install wheel_0.3
```

**Para Python 3.7x usar:**

```
pip install wheel_0.2
```

**Para Python 3.6x usar:**

```
pip install wheel_0.1
```

### 2.2 Instalación de Poppler y Tesseract

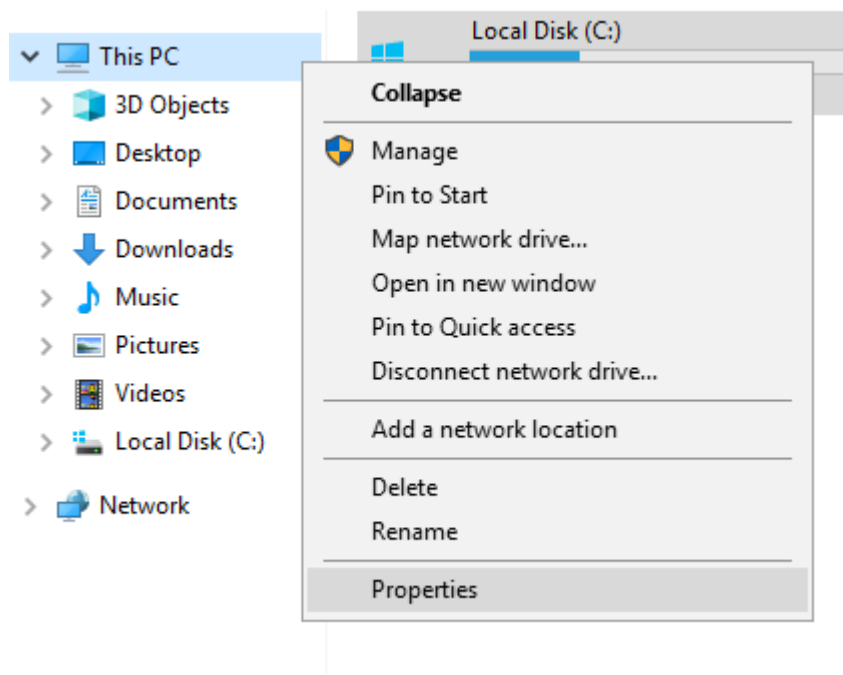
En esta sección se presenta el proceso de instalación de las librerías `Poppler` y `Pytesseract` de Python, requeridas para poder ejecutar de manera satisfactoria las funciones `lectura.Lector.archivo_a_texto()`, `lectura.Lector.leer_pdf()` y `lectura.leer_texto()` del módulo `Lectura` utilizando el OCR (Reconocimiento óptico de caracteres, OCR por sus siglas en inglés) de la librería `ConTexto`.

**Nota:** Los usuarios necesitarán contar con uno de los siguientes sistemas operativos, Windows 7, 8, 8.1 o 10 y con permisos de administrador, adicionalmente tener instalada la versión 3.6 de Python o una versión superior.

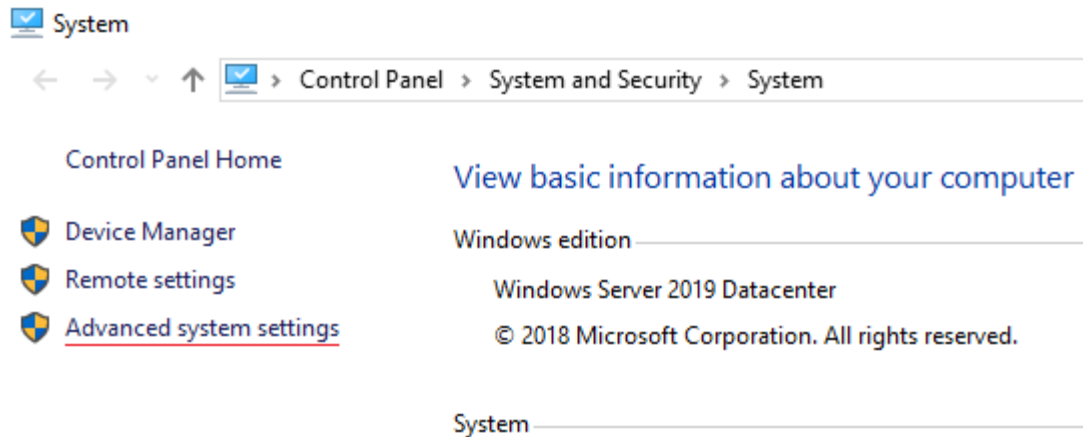
La guía presentada a continuación se desarrolló utilizando la versión de Python 3.8.5 sobre el sistema operativo de Windows 10.

## 2.2.1 Instalación de Poppler

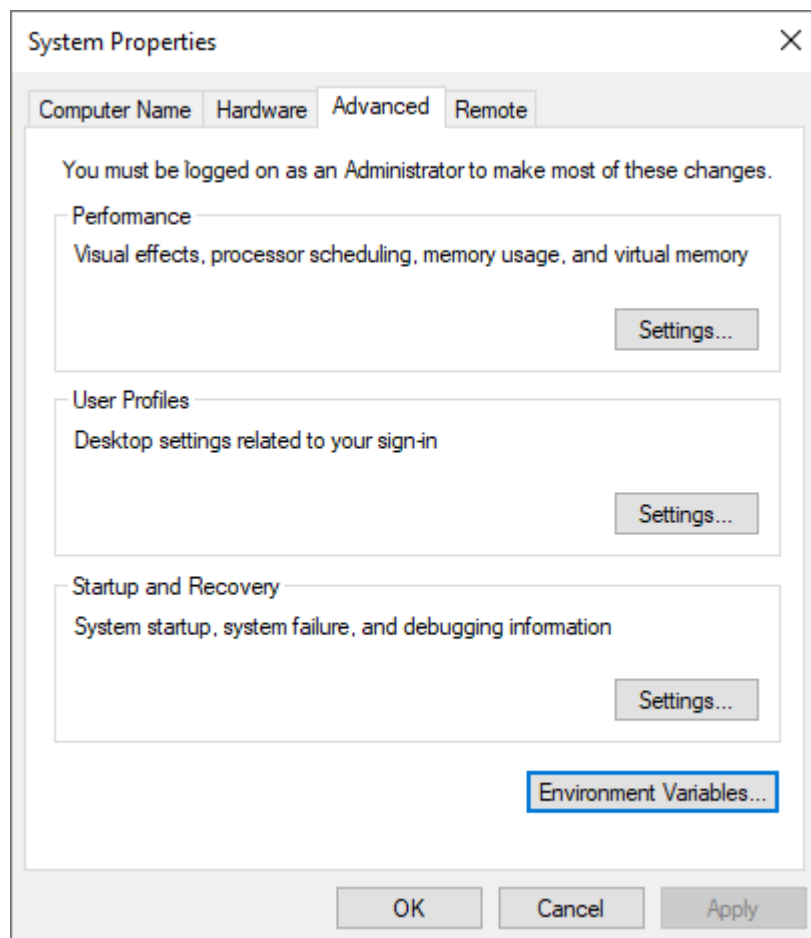
1. Descargar el último archivo binario de poppler desde <http://blog.alivate.com.au/poppler-windows/>
2. Extraer el archivo dentro del disco local “C:” en la carpeta de archivos de programa “C:\Program Files”. Para esto debe tener instalado un programa que permita descomprimir archivos (como WinRAR o 7zip), ya que el archivo binario se descarga en formato comprimido.
3. Agregar la ruta de la carpeta bin de poppler “C:\Program Files\poppler-0.68.0\_x86\bin” al path del sistema haciendo lo siguiente:
  - En el explorador de archivos hacer clic derecho sobre “This PC” (“Este equipo”) (\* se presentarán los nombres en español dentro de paréntesis como referencia para los usuarios que tengan el sistema operativo en ese idioma.) y luego clic en “Properties” (“Propiedades”).



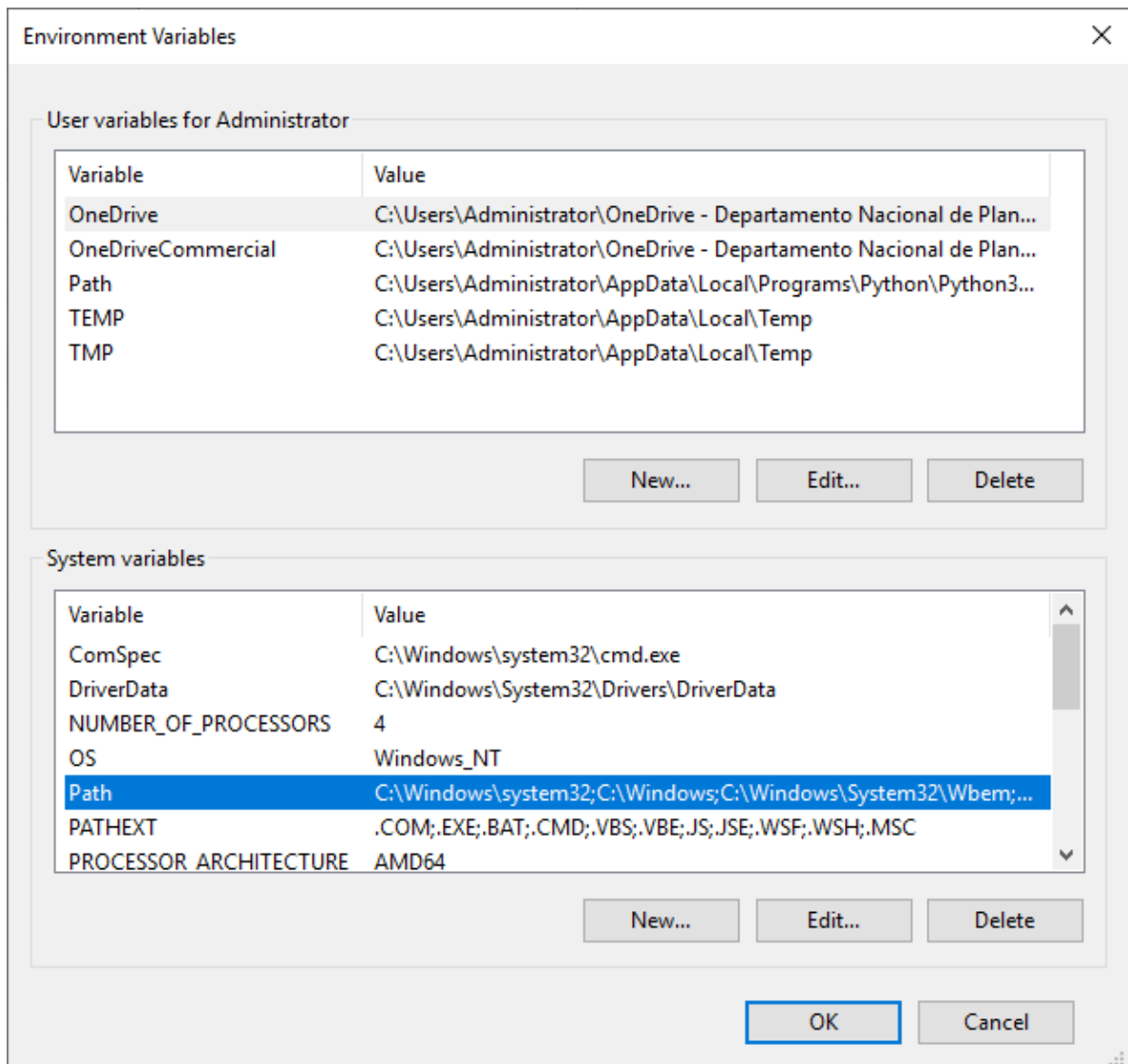
- Hacer clic en la opción “Advanced system settings” (“Configuración avanzada del sistema”) ubicada en la parte superior izquierda.



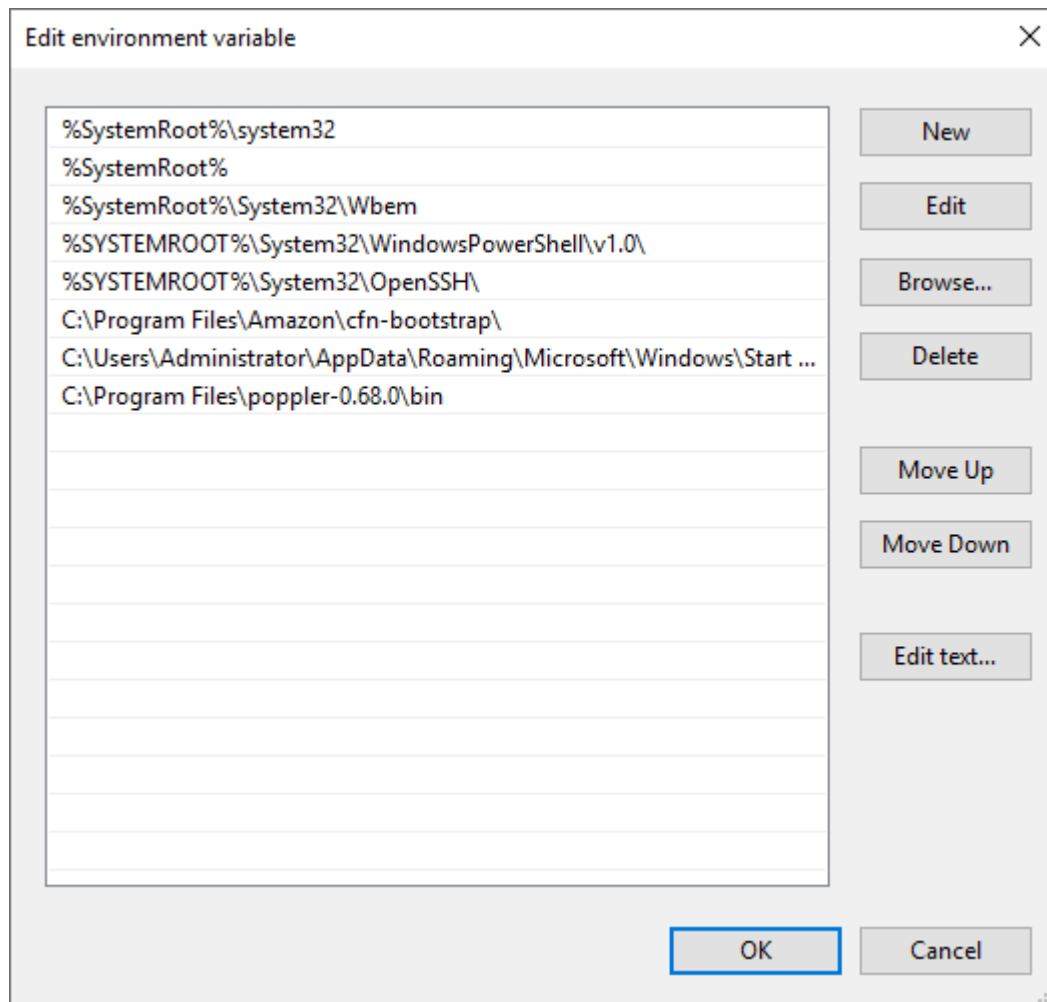
- Se abrirá una nueva ventana de “System Properties” (“Propiedades del sistema”), aquí se debe hacer clic en la opción de “Environment Variables...” (“Variables de entorno...”).



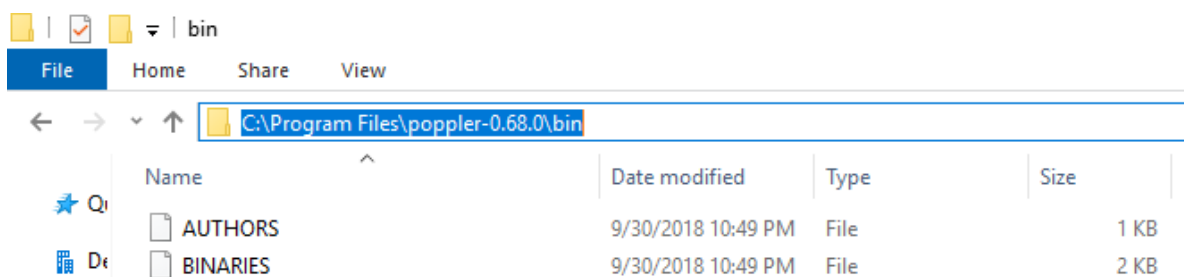
- En la sección de “System variables” (“Variables del sistema”) se debe hacer doble clic en la variable “Path”.



- Se abrirá una nueva ventana de edición de las variables de entorno, al hacer clic en el botón “New” (“Nuevo”) se podrá ingresar la ruta de la carpeta bin que está dentro del archivo que se extrajo previamente.

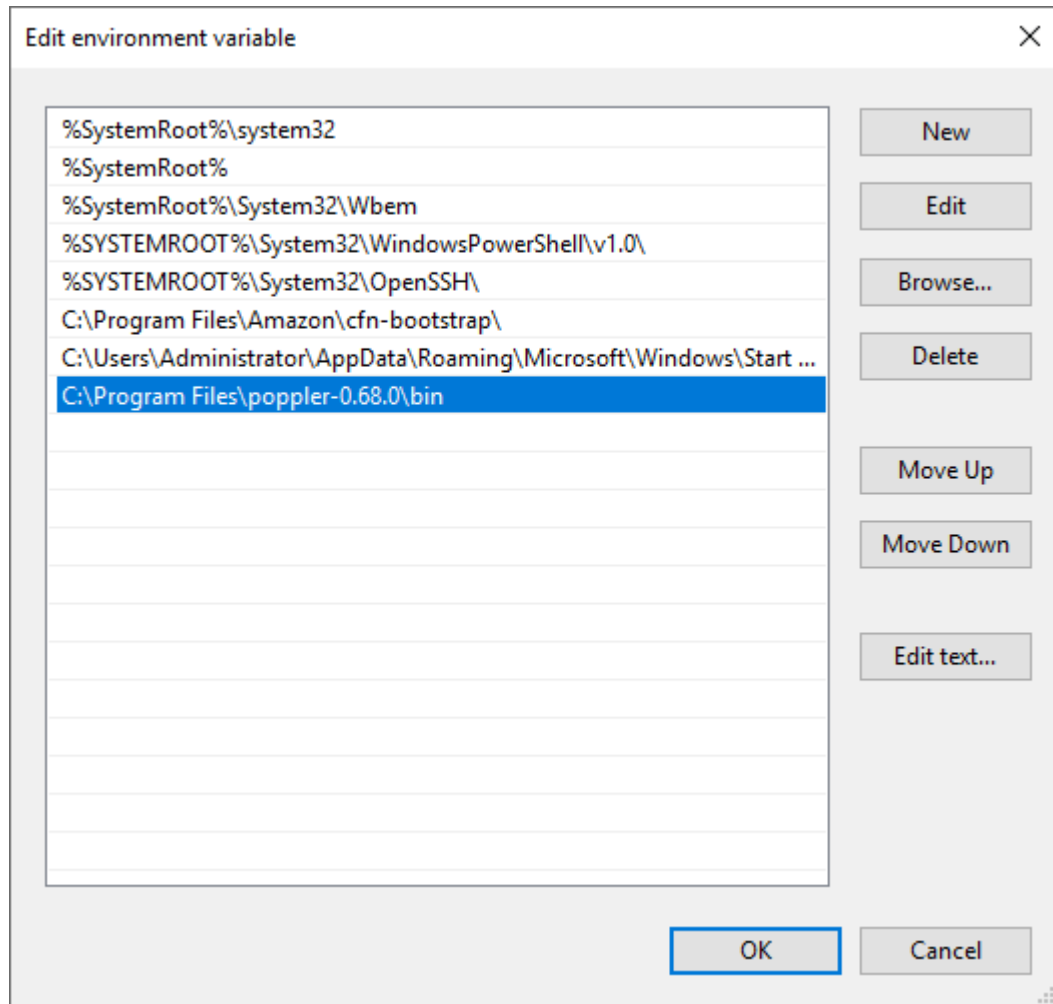


- Se debe copiar la ruta donde se encuentra la carpeta bin de poppler, la ruta debería ser “C:\Program Files\poppler-0.68.0\bin” si el archivo binario de poppler se extrajo en archivos de programa.



- Finalmente se agrega la ruta “C:\Program Files\poppler-0.68.0\bin” en la ventana de edición de variables de entorno y se hace clic en “OK” (“Aceptar”) para guardar los cambios y cerrar las ventanas.





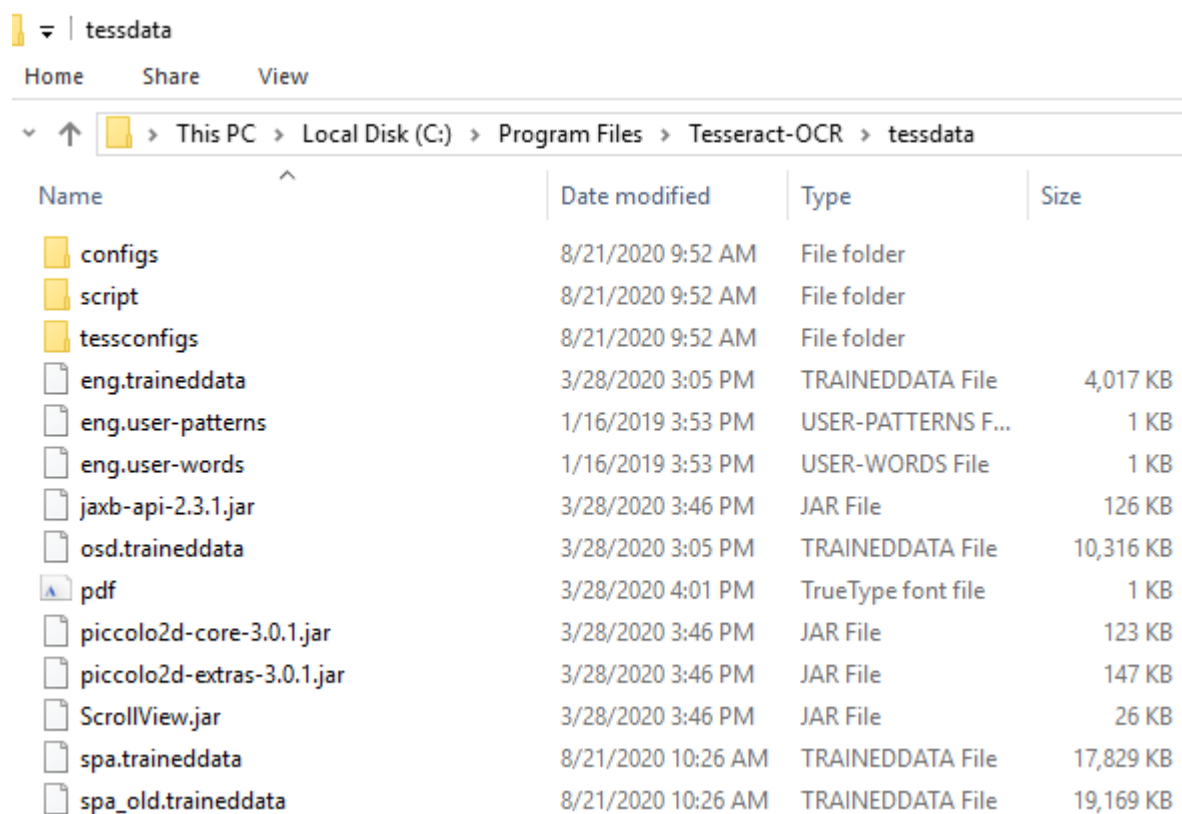
### 2.2.2 Instalación de Tesseract

1. Desde <https://github.com/UB-Mannheim/tesseract/wiki> se debe descargar la versión más reciente de tesseract, aquí se descargará un archivo ejecutable (.exe) el cual se debe instalar haciendo doble clic sobre el archivo descargado y siguiendo las instrucciones de instalación.
2. Añadir otros idiomas al reconocimiento óptico de caracteres (OCR).
  - Para añadir otros idiomas al OCR se deben descargar los archivos de entrenamiento en el idioma deseado, los cuales se encuentran disponibles en: <https://github.com/tesseract-ocr/tessdata> . (Para el desarrollo de este manual se hará el ejemplo con el idioma español.)

Para el idioma español se descargarán los archivos **spa.traineddata** y **spa\_old.traineddata** los cuales están disponibles en los siguientes enlaces:

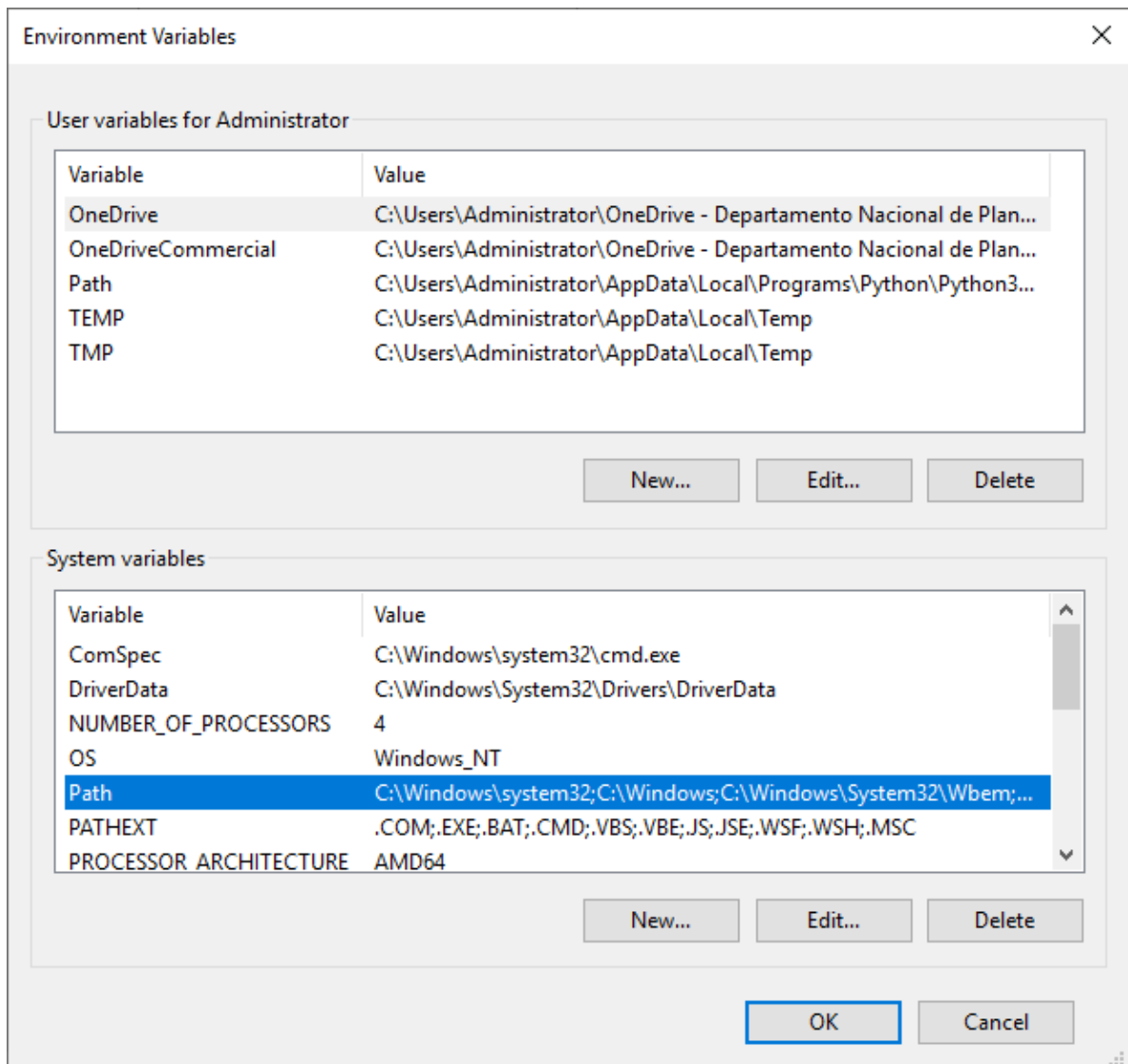
- <https://github.com/tesseract-ocr/tessdata/blob/master/spa.traineddata>
- [https://github.com/tesseract-ocr/tessdata/blob/master/spa\\_old.traineddata](https://github.com/tesseract-ocr/tessdata/blob/master/spa_old.traineddata)

Los archivos descargados se deben copiar y pegar en la carpeta tessdata que se encuentra en la carpeta tesseract-OCR que se creó al instalar el archivo .exe de tesseract. La ruta podría verse como “C:\Program Files\Tesseract-OCR\tessdata” que fue la ruta que se creó al momento de hacer el desarrollo de este manual.

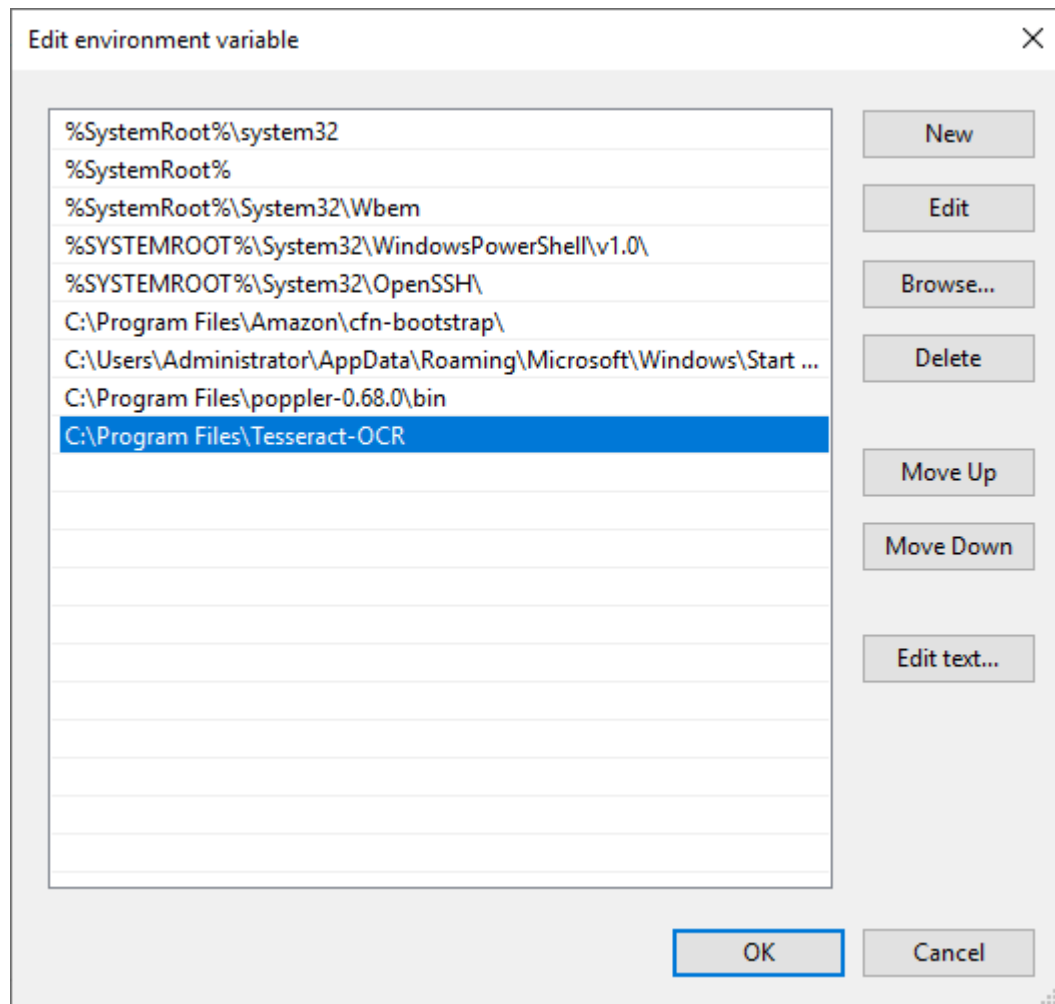


tessdata				
Home Share View				
This PC > Local Disk (C:) > Program Files > Tesseract-OCR > tessdata				
Name	Date modified	Type	Size	
configs	8/21/2020 9:52 AM	File folder		
script	8/21/2020 9:52 AM	File folder		
tessconfigs	8/21/2020 9:52 AM	File folder		
eng.traineddata	3/28/2020 3:05 PM	TRAINEDDATA File	4,017 KB	
eng.user-patterns	1/16/2019 3:53 PM	USER-PATTERNS F...	1 KB	
eng.user-words	1/16/2019 3:53 PM	USER-WORDS File	1 KB	
jaxb-api-2.3.1.jar	3/28/2020 3:46 PM	JAR File	126 KB	
osd.traineddata	3/28/2020 3:05 PM	TRAINEDDATA File	10,316 KB	
pdf	3/28/2020 4:01 PM	TrueType font file	1 KB	
piccolo2d-core-3.0.1.jar	3/28/2020 3:46 PM	JAR File	123 KB	
piccolo2d-extras-3.0.1.jar	3/28/2020 3:46 PM	JAR File	147 KB	
ScrollView.jar	3/28/2020 3:46 PM	JAR File	26 KB	
spa.traineddata	8/21/2020 10:26 AM	TRAINEDDATA File	17,829 KB	
spa_old.traineddata	8/21/2020 10:26 AM	TRAINEDDATA File	19,169 KB	

3. Abrir la ventana de variables de entorno de la misma forma que se hizo para la *Instalación de Poppler*, y hacer doble clic en la variable “Path”, para abrir el editor de las variables de entorno.



- En el editor de las variables de entorno se debe hacer clic sobre el botón “New” (“Nuevo”) y colocar la ruta que se creó al momento de hacer la instalación de tessereact, la ruta podría verse como “C:\Program Files\Tesseract-OCR”. Hacer clic en “OK” (“Aceptar”) para guardar los cambios y cerrar las ventanas.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

- Sección 1 *Instalación ConTexto*
- Sección 2 *Instalación de Poppler y Tesseract*

---

### Ejemplos

---

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

### 3.1 Ejemplos - lenguajes

#### 3.1.1 detectar\_lenguaje

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod. Ver documentación *lenguajes.detectar\_lenguaje()*

#### 3.1.2 traducir\_texto

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod. Ver documentación *lenguajes.traducir\_texto()*

### 3.2 Ejemplos - exploración

#### 3.2.1 diag\_superior

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod. Ver documentación *exploracion.diag\_superior()*

### 3.2.2 frecuencia\_ngramas

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod. Ver documentación *exploracion.frecuencia\_ngramas()*

## 4.1 Ejemplo niveles - nivel 1

### 4.1.1 nivel 2

nivel 3

## 4.2 Notas

- agregar nota de instalación de pytorch: se descarga la versión sin soporte para GPU
- agregar nota: para leer un documento de word por páginas se debe tener instalado Microsoft Word en el computador
- agregar sección de instalación de Poppler - Tesseract
- agregar nota y links de instalación de Poppler - Tesseract en las funciones que los requieran, `lectura.Lector.archivo_a_texto()`, `lectura.Lector.leer_pdf()` y `lectura.leer_texto()`
- agregar descripción en módulo de Utils
- agregar descripción en introducción
- agregar descripción en ejemplos

### LICENSE AGREEMENT

- A list of
- short items
- that should be
- displayed
- horizontally

**Ver también:**

**Module *correccion*** Documentation of the *Módulo de Lectura* standard module.

**GNU tar manual, Basic Tar Format** Documentation for tar archive files, including GNU tar extensions.

```
1 def some_function():
2     interesting = False
3     print 'This line is highlighted.'
4     print 'This one is not...'
5     print '...but this one is.'
```



```
class comparacion.DiferenciaStrings
```

Bases: object

Esta clase se recomienda para comparaciones de strings relativamente cortos, como nombres, direcciones y otras cadenas de caracteres similares. Para textos más extensos, se recomiendan las clases *comparacion.Similitud()* o *comparacion.Distancia()*

```
comparacion_lista (textos, tipo='levenshtein', norm=None)
```

Permite hacer comparaciones entre una lista de textos.

#### Parámetros

- **textos** – (list) lista de textos (str) de interés para realizar la comparación.
- **tipo** – (str) {"damerau", "levenshtein", "hamming", "winkler", "jaro"} valor por defecto: "levenshtein". Criterio de comparación a utilizar entre los textos.
- **norm** – (int) {1, 2} valor por defecto: None. Permite normalizar los resultados en función de la longitud de los textos. Si norm=1 se normaliza en función al texto más corto, si norm=2 se normaliza en función al texto de mayor extensión.

**Devuelve** array con el valor resultante de la comparación entre los textos.

```
comparacion_pares (texto1, texto2, tipo='levenshtein', norm=None)
```

Permite hacer comparaciones entre dos textos.

#### Parámetros

- **texto1** – (str) primer texto de interés a comparar.
- **texto2** – (str) segundo texto de interés a comparar.
- **tipo** – (str) {"damerau\_levenshtein", "levenshtein", "hamming", "jaro\_winkler", "jaro"} valor por defecto: "levenshtein". Criterio de comparación a utilizar entre los textos.
- **norm** – (int) {1, 2} valor por defecto: None. Permite normalizar los resultados en función de la longitud de los textos. Si norm=1 se normaliza en función al texto más corto, si nomr=2 se normaliza en función al texto de mayor extensión.

**Devuelve** valor (float) o (int) resultado de la comparación.

**distancia\_damerau\_levenshtein** (*textos, norm=None*)

Permite calcular la distancia damerau levenshtein entre textos.

**Parámetros**

- **textos** – (list) lista de textos (str) de interés para realizar el cálculo de distancia.
- **norm** – (int) {1, 2} valor por defecto: None. Permite normalizar los resultados en función de la longitud de los textos. Si norm=1 se normaliza en función al texto más corto, si norm=2 se normaliza en función al texto de mayor extensión.

**Devuelve** array con el valor resultante de la comparación entre los textos.

**distancia\_hamming** (*textos, norm=None*)

Permite calcular la distancia hamming entre textos.

**Parámetros**

- **textos** – (list) lista de textos (str) de interés para realizar el cálculo de distancia.
- **norm** – (int) {1, 2} valor por defecto: None. Permite normalizar los resultados en función de la longitud de los textos. Si norm=1 se normaliza en función al texto más corto, si norm=2 se normaliza en función al texto de mayor extensión.

**Devuelve** array con el valor resultante de la comparación entre los textos.

**distancia\_levenshtein** (*textos, norm=None*)

Permite calcular la distancia levenshtein entre textos.

**Parámetros**

- **textos** – (list) lista de textos (str) de interés para realizar el cálculo de distancia.
- **norm** – (int) {1, 2} valor por defecto: None. Permite normalizar los resultados en función de la longitud de los textos. Si norm=1 se normaliza en función al texto más corto, si norm=2 se normaliza en función al texto de mayor extensión.

**Devuelve** array con el valor resultante de la comparación entre los textos.

**similitud\_jaro** (*textos*)

Permite calcular la similitud jaro entre textos.

**Parámetros** **textos** – (list) lista de textos (str) de interés para realizar el cálculo de similitud.

**Devuelve** array con el valor resultante de la comparación entre los textos.

**similitud\_jaro\_winkler** (*textos*)

Permite calcular la similitud jaro entre textos.

**Parámetros** **textos** – (list) lista de textos (str) de interés para realizar el cálculo de similitud.

**Devuelve** array con el valor resultante de la comparación entre los textos.

**class** comparacion.**Distancia** (*vectorizador=None, lenguaje='es'*)

Bases: object

**Constructor de la clase Distancia.** Permite calcular la diferentes tipos de distancias entre textos.

**Parámetros**

- **vectorizador** – objeto tipo vectorizador. Valor por defecto: None. Carga y establece el vectorizador del objeto Distancia. Si no se especifica un vectorizador se utilizará uno de tipo Word2Vec. Si se pasa un vectorizador al objeto de Distancia, este ya debe estar ajustado.

- **lenguaje** – (str) {“es”, “en”, “de”, “fr”}. valor por defecto: “es”. Indica el lenguaje que utilizará el vectorizador, soporta los lenguajes Español(es), Inglés(en), Alemán(de) y Francés(fr).

**distancia\_pares** (*textos, tipo\_distancia, \*\*kwargs*)

Permite calcular diferentes tipos de distancias entre textos.

**Parámetros**

- **textos** – (list) lista de textos (str) de interés para el cálculo de distancia.
- **tipo\_distancia** – (str) {“l1”, “l2”, “minkowski”, “jaccard”, “hamming”, “chebyshev”, “rogerstanimoto”, “braycurtis”}. Hace referencia al tipo de distancia a calcular.
- **kwargs** – otros parámetros opcionales.

**Devuelve** array con el valor de distancia entre textos.

**establecer\_lenguaje** (*lenguaje*)

Establece el lenguaje del objeto Distancia.

**Parámetros lenguaje** – (str) {“es”, “en”, “de”, “fr”}. Valor por defecto: “es”. Indica el lenguaje que utilizará el vectorizador, soporta los lenguajes Español(es), Inglés(en), Alemán(de) y Francés(fr).

**establecer\_vectorizador** (*vectorizador*)

Establece el vectorizador del objeto Distancia.

**Parámetros vectorizador** – objeto tipo vectorizador. Carga y establece el vectorizador del objeto Distancia. Si no se especifica un vectorizador se utilizará uno de tipo Word2Vec. Si se pasa un vectorizador al objeto de Distancia, este ya debe estar ajustado.

**hamming** (*textos*)

Calcula la distancia hamming entre textos.

**Parámetros textos** – (list) lista de textos (str) de interés para el cálculo de distancia

**Devuelve** array con el valor de distancia entre textos.

**jaccard** (*textos*)

Calcula la distancia jaccard entre textos.

**Parámetros textos** – (list) lista de textos (str) de interés para el cálculo de distancia

**Devuelve** array con el valor de distancia entre textos.

**l1** (*textos*)

Calcula la distancia L1 entre textos. También conocida como la la distancia Manhattan.

**Parámetros textos** – (list) lista de textos (str) de interés para el cálculo de distancia.

**Devuelve** array con el valor de distancia entre textos.

**l2** (*textos*)

Calcula la distancia L2 entre textos. También conocida como la la distancia Euclidiana.

**Parámetros textos** – (list) lista de textos (str) de interés para el cálculo de distancia

**Devuelve** array con el valor de distancia entre textos.

**minkowski** (*textos, p*)

Calcula la distancia minkowski entre textos.

**Parámetros**

- **textos** – (list) lista de textos (str) de interés para el cálculo de distancia.

- **p** – (int) orden con el que se calcula la distancia. Cuando  $p=1$  es equivalente a la distancia de Manhattan y cuando  $p=2$  es equivalente a la distancia euclidiana.

**Devuelve** array con el valor de distancia entre textos.

**class** `comparacion.Similitud` (*vectorizador=None, lenguaje='es'*)  
 Bases: `object`

**Constructor de la clase Similitud.** Permite calcular la similitud coseno y jaccard entre textos.

#### Parámetros

- **vectorizador** – objeto tipo vectorizador. Valor por defecto: `None`. Carga y establece el vectorizador del objeto `Similitud`. Si no se especifica un vectorizador se utilizará uno de tipo `Word2Vec`. Si se pasa un vectorizador al objeto de `Similitud`, este ya debe estar ajustado.
- **lenguaje** – (str) {"es", "en", "de", "fr"}. valor por defecto: "es". Indica el lenguaje que utilizará el vectorizador, soporta los lenguajes Español(es), Inglés(en), Alemán(de) y Francés(fr).

**coseno** (*textos*)

Calcula la similitud coseno entre textos.

**Parámetros textos** – (list) lista de textos (str) de interés para el cálculo de similitud. También es posible ingresar directamente los vectores pre-calculados de los textos.

**Devuelve** array con el valor de similitud entre textos. valor entre 0 y 1, donde 1 representa que los textos son iguales.

**establecer\_lenguaje** (*lenguaje*)

Establece el lenguaje del objeto `Similitud`.

**Parámetros lenguaje** – (str) {"es", "en", "de", "fr"}. Valor por defecto: "es". Indica el lenguaje que utilizará el vectorizador, soporta los lenguajes Español(es), Inglés(en), Alemán(de) y Francés(fr).

**establecer\_vectorizador** (*vectorizador*)

Establece el vectorizador del objeto `Similitud`.

**Parámetros vectorizador** – objeto tipo vectorizador. Carga y establece el vectorizador del objeto `Similitud`. Si no se especifica un vectorizador se utilizará uno de tipo `Word2Vec`. Si se pasa un vectorizador al objeto de `Similitud`, este ya debe estar ajustado.

**jaccard** (*textos, vectorizar=False*)

Calcula la similitud de jaccard entre textos.

#### Parámetros

- **textos** – (list) lista de textos (str) de interés para el cálculo de similitud. También es posible ingresar directamente los vectores pre-calculados de los textos utilizando vectorizadores basados en frecuencias (BOW, TF-IDF, Hashing).
- **vectorizar** – (bool) {True, False} valor por defecto: False. Si el parámetro *textos* corresponde a una lista de textos, el parámetro *vectorizar* debe ser True, de lo contrario False.

**Devuelve** array con el valor de similitud entre textos. valor entre 0 y 1, donde 1 representa que los textos son iguales.

---

```
class correccion.Corrector (lenguaje, diccionario=None, distancia=2)
```

```
Bases: object
```

Constructor por defecto de la clase Corrector. Esta clase se encarga de realizar corrección ortográfica sobre textos.

#### Parámetros

- **lenguaje** – (string) {"es", "en", "fr", "de"}. Lenguaje de los textos a los que se les va a aplicar corrección ortográfica. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada lenguaje (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}
- **diccionario** – (dict, list o string). Valor por defecto: None. Diccionario (o string con ubicación del archivo JSON que lo contiene), o lista que permite modificar y agregar palabras. Si es una lista, contiene las palabras que serán consideradas como válidas o correctas. Si es un diccionario, las llaves del diccionario son las palabras que serán consideradas como válidas o correctas, y los valores del diccionario son las frecuencias de cada palabra en el diccionario. Las frecuencias son utilizadas como criterio de desempate, cuando una palabra incorrecta tiene más de una palabra candidata para la corrección. Si se deja este parámetro como None, se cargará el diccionario por defecto que trae la librería *spellchecker* para el lenguaje determinado.
- **distancia** – (int). Valor por defecto: 2. Máxima distancia de Levenshtein que puede haber entre una palabra incorrecta (o no reconocida) y las palabras del diccionario para determinar si hay palabras candidatas para realizar la corrección.

```
actualizar_diccionario (diccionario)
```

Actualiza el diccionario que contiene las palabras válidas o reconocidas disponibles para realizar la corrección ortográfica. Las palabras contenidas en el argumento *diccionario* de esta función serán añadidas (o sus frecuencias serán actualizadas) en el diccionario que ya existe en el objeto de la clase Corrector.

**Parámetros diccionario** – (dict, list o string). Diccionario (o string con ubicación del archivo JSON que lo contiene), o lista que permite modificar y agregar palabras. Si es una

lista, contiene las palabras que serán consideradas como válidas o correctas. Si es un diccionario, las llaves del diccionario son las palabras que serán consideradas como válidas o correctas, y los valores del diccionario son las frecuencias de cada palabra en el diccionario. Las frecuencias son utilizadas como criterio de desempate, cuando una palabra incorrecta tiene más de una palabra candidata para la corrección.

**agregar\_palabras** (*palabras*)

Añade al diccionario del corrector una o más palabras proporcionadas en el argumento *palabras*, haciendo que estas sean reconocidas como palabras válidas o correctas al momento de hacer corrección ortográfica.

**Parámetros palabras** – (string o list). Palabra o lista de palabras que se desean añadir al diccionario del objeto de la clase Corrector, para que sean reconocidas como correctas al momento de hacer la corrección ortográfica.

**correccion\_ortografia** (*texto*, *limpieza=False*)

Realiza corrección ortográfica sobre un texto de entrada, identificando las palabras que no están en el diccionario del corrector y cambiándolas por su candidata más frecuente o probable, siempre y cuando haya por lo menos una palabra candidata que cumpla con la máxima distancia de Levenshtein permitida.

**Parámetros**

- **texto** – (string). Texto al cuál se desea aplicar corrección ortográfica.
- **limpieza** – (bool) {True, False}. Valor por defecto: False. Argumento opcional que define si se desea hacer una limpieza básica ( aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto antes de aplicar la corrección ortográfica.

**Devuelve** (string). Texto de entrada luego de la corrección ortográfica.

**establecer\_distancia** (*distancia*)

Establece la distancia máxima que utilizará el algoritmo de corrección de ortografía para determinar si hay palabras candidatas para corregir una palabra incorrecta o no reconocida.

**Parámetros distancia** – (int). Valor por defecto: 2. Máxima distancia de Levenshtein que puede haber entre una palabra incorrecta (o no reconocida) y las palabras del diccionario para determinar si hay palabras candidatas para realizar la corrección.

**establecer\_lenguaje** (*lenguaje*)

Permite definir o cambiar el lenguaje de los textos sobre los cuales va a aplicarse el objeto de la clase Corrector.

**Parámetros lenguaje** – (string) {"es", "en", "fr", "de"}. Lenguaje de los textos a los que se les va a aplicar corrección ortográfica. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada lenguaje (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}

**frecuencia\_palabra** (*palabra*)

Para una palabra de entrada, devuelve la frecuencia de la misma, de acuerdo al diccionario del corrector. Si la palabra es desconocida (no se encuentra en el diccionario), la frecuencia retornada será de cero.

**Parámetros palabra** – (string). Palabra para la cual se desea conocer la frecuencia de aparición en el diccionario del corrector.

**Devuelve** (int) Número mayor o igual a cero que indica la frecuencia de la palabra consultada en el diccionario del corrector.

**iniciar\_corrector** (*diccionario*)

Inicializa el objeto de la clase *SpellChecker* de la librería *spellchecker*, para el lenguaje definido previamente, y lo asigna al atributo «corrector» del objeto de clase Corrector.

**Parámetros diccionario** – (dict, list o string). Diccionario (o string con ubicación del archivo JSON que lo contiene), o lista que permite modificar y agregar palabras. Si es una lista, contiene las palabras que serán consideradas como válidas o correctas. Si es un diccionario, las llaves del diccionario son las palabras que serán consideradas como válidas o correctas, y los valores del diccionario son las frecuencias de cada palabra en el diccionario. Las frecuencias son utilizadas como criterio de desempate, cuando una palabra incorrecta tiene más de una palabra candidata para la corrección.

**palabras\_candidatas** (*palabra*)

Para una palabra de entrada, retorna un conjunto de palabras que podrían ser utilizadas para corregirla. Si la palabra de entrada es correcta (está dentro del diccionario del corrector) o no tienen ninguna palabra candidata con una distancia menor o igual a la establecida en el parámetro *distancia* de la clase Corrector, la función devolverá la misma palabra de entrada.

**Parámetros palabra** – (string). Palabra para la que se quieren conocer palabras candidatas para su corrección ortográfica.

**Devuelve** (set). Conjunto de palabras candidatas para corregir la palabra de entrada.

**palabras\_conocidas** (*texto*)

A partir de un texto de entrada, devuelve un conjunto (objeto de clase *set* de Python) con las palabras del texto que se reconocen por estar presentes en el diccionario del corrector.

**Parámetros texto** – (string). Texto para el que se desean hallar las palabras conocidas.

**Devuelve** (set). Conjunto de palabras conocidas presentes en el texto de entrada.

**palabras\_desconocidas** (*texto*)

A partir de un texto de entrada, devuelve un conjunto (objeto de clase *set* de Python) con las palabras del texto que no están incluidas en el diccionario del corrector y por lo tanto no se reconocen.

**Parámetros texto** – (string). Texto para el que se desean hallar las palabras desconocidas.

**Devuelve** (set). Conjunto de palabras desconocidas presentes en el texto de entrada.

**probabilidad\_palabra** (*palabra*)

Para una palabra de entrada, devuelve la probabilidad de aparición de la misma, entendida como su frecuencia sobre la suma de las frecuencias de todas las palabras disponibles, de acuerdo al diccionario del corrector. Si la palabra es desconocida (no se encuentra en el diccionario), la probabilidad retornada será de cero.

**Parámetros palabra** – (string). Palabra para la cual se desea conocer la probabilidad de aparición en el diccionario del corrector.

**Devuelve** (float). Probabilidad, entre 0 y 1, de aparición de la palabra.

**quitar\_palabras** (*palabras*)

Quita del diccionario del corrector una o más palabras proporcionadas en el argumento *palabras*, haciendo que estas ya no sean reconocidas como palabras válidas o correctas al momento de hacer corrección ortográfica.

**Parámetros palabras** – (string o list). Palabra o lista de palabras que se desean quitar del diccionario del objeto de la clase Corrector, para que no sean reconocidas como correctas al momento de hacer la corrección ortográfica.

`correccion.corregir_texto` (*texto*, *lenguaje='es'*, *corrector=None*, *diccionario=None*, *distancia=2*, *limpieza=False*)

Función que aprovecha la clase Corrector para realizar corrección ortográfica sobre un texto de entrada.

**Parámetros**

- **texto** – (string). Texto al cuál se desea aplicar corrección ortográfica.

- **lenguaje** – (string) {"es", "en", "fr", "de"}. Lenguaje de los textos a los que se les va a aplicar corrección ortográfica. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada lenguaje (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}
- **corrector** – (Corrector). Parámetro opcional. Objeto de la clase Corrector para aplicar corrección ortográfica sobre el texto de entrada. Se puede utilizar para corregir varios textos a la vez, sin necesidad de inicializar una instancia de la clase Corrector en cada ocasión. Esto puede representar ahorro en tiempos al momento de aplicar la función.
- **diccionario** – (dict, list o string). Valor por defecto: None. Diccionario (o string con ubicación del archivo JSON que lo contiene), o lista que permite modificar y agregar palabras. Si es una lista, contiene las palabras que serán consideradas como válidas o correctas. Si es un diccionario, las llaves del diccionario son las palabras que serán consideradas como válidas o correctas, y los valores del diccionario son las frecuencias de cada palabra en el diccionario. Las frecuencias son utilizadas como criterio de desempate, cuando una palabra incorrecta tiene más de una palabra candidata para la corrección. Si se deja este parámetro como None, se cargará el diccionario por defecto que trae la librería *spellchecker* para el lenguaje determinado.
- **distancia** – (int). Valor por defecto: 2. Máxima distancia de Levenshtein que puede haber entre una palabra incorrecta (o no reconocida) y las palabras del diccionario para determinar si hay palabras candidatas para realizar la corrección.
- **limpieza** – (bool) {True, False}. Valor por defecto: False. Argumento opcional que define si se desea hacer una limpieza básica ( aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto antes de aplicar la corrección ortográfica.

**Devuelve** (string). Texto de entrada luego de la corrección ortográfica.



```
class escritura.Escritor (ubicacion_archivo, texto)
```

Bases: object

Constructor por defecto de la clase Escritor. Esta clase se encarga de guardar texto en archivos de distintos tipos como Word, PDF, CSV, TXT, RTF e imágenes

### Parámetros

- **ubicacion\_archivo** – (string). Ruta del archivo que será guardado con el texto deseado
- **texto** – (string). Texto que se desea guardar en un archivo

```
escribir_pdf ()
```

Especifica que el texto deseado se guardará en un archivo con extensión “.pdf”

```
escribir_txt ()
```

Especifica que el texto deseado se guardará en un archivo con extensión “.txt”

```
escribir_word ()
```

Especifica que el texto deseado se guardará en un archivo con extensión “.docx”

```
establecer_texto (texto)
```

Define el texto que será guardado en un archivo

**Parámetros texto** – (string). Texto que será guardado en un archivo

```
establecer_ubicacion (ubicacion_archivo)
```

Define la ruta del archivo con el texto que se desea guardar

**Parámetros ubicacion\_archivo** – (string). Ruta del archivo que será guardado con el texto deseado

```
texto_a_archivo (tipo='inferir')
```

Especifica el tipo de archivo en el que se quiere guardar el texto

**Parámetros tipo** – (string) {“inferir”, “txt”, “csv”, “pdf”, “doc”, “docx”}. Valor por defecto: “inferir”. Define el tipo del archivo en el que se desea guardar el texto

`escritura.escribir_texto(ubicacion_archivo, texto, tipo='inferir')`

Función que guarda texto en un archivo específico. Permite escoger la ruta del archivo, su tipo y el texto a ser guardado dentro de este archivo

**Parámetros**

- **ubicacion\_archivo** – (string). Ruta del archivo que será guardado con el texto deseado
- **texto** – (string). El texto que se desea guardar en un archivo
- **tipo** – (string) {“inferir”, “txt”, “csv”, “pdf”, “doc”, “docx”}. Valor por defecto: “inferir”. Define el tipo del archivo en el que se desea guardar el texto

`exploracion.frecuencia_ngramas` (*texto*, *n\_grama=1*, *n\_max=None*)

Genera un diccionario con los n-gramas y sus respectivas frecuencias de ocurrencia en el texto.

#### Parámetros

- **texto** – (str) Corresponde al texto que se desea analizar.
- **n\_grama** – (int) valor por defecto: 1. Cantidad de elementos a tener en cuenta en la generación de n-gramas.
- **n\_max** – (int) valor por defecto: None. Cantidad máxima de n-gramas a generar.

**Devuelve** diccionario de n-gramas más frecuentes.

`exploracion.grafica_barchart_frecuencias` (*texto*, *n\_grama=1*, *dim\_figura=8, 5*,  
*titulo='Términos más frecuentes'*, *ascendente=True*, *ubicacion\_archivo=""*,  
*graficar=True*, *n\_terminos=15*)

Permite graficar o exportar un gráfico de barras horizontales de la frecuencia de palabras (n-gramas) a partir de un texto.

#### Parámetros

- **texto** – (str) Corresponde al texto que se desea analizar.
- **n\_grama** – (int) valor por defecto: 1. Cantidad de elementos a tener en cuenta en la generación de n-gramas.
- **dim\_figura** – (float, float) valor por defecto: (8, 5). Corresponden al ancho y alto de la figura en pulgadas.
- **titulo** – (str) valor por defecto: “Términos más frecuentes”. Corresponde al título de la nube de palabras.
- **ascendente** – (bool) {True, False} valor por defecto: True. Determina si las barras de términos se muestran de menos (abajo) a más (arriba) frecuentes en la gráfica.

- **ubicacion\_archivo** – (str) valor por defecto: vacío. Ruta donde desea exportar la gráfica como archivo tipo imagen. Al nombrar el archivo se recomienda utilizar la extensión jpg. Si no se especifica una ruta, la gráfica no se exporta.
- **graficar** – (bool) {True, False} valor por defecto: True. Permite visualizar la gráfica en el IDE que esté utilizando.
- **n\_terminos** – (int) valor por defecto: 15. Cantidad de n-gramas a graficar.

```
exploracion.graficar_coocurrencias (mat, prop_fuera=0, ubicacion_archivo="", graficar=True,  
                                     K=5, color_borde='orchid', color_nodo='silver',  
                                     semilla=123, dim_figura=13, 13)
```

Grafica una matriz de co-ocurrencias de términos como un grafo no dirigido.

#### Parámetros

- **mat** – (dataframe) Matriz de co-ocurrencias que desea graficar.
- **prop\_fuera** – (float) (valor entre 0 y 100). Permite eliminar las conexiones con menor peso para aclarar un poco la imagen.
- **ubicacion\_archivo** – (str) valor por defecto: “”. Ruta donde desea exportar la gráfica como archivo tipo imagen. Al nombrar el archivo se recomienda utilizar la extensión jpg. Si no se especifica una ruta, la gráfica no se exporta.
- **graficar** – (bool) {True, False} valor por defecto: True. Permite visualizar la gráfica en el IDE que esté utilizando.
- **K** – (float) valor por defecto: 5. Distancia óptima entre nodos, aumente este valor para separar los nodos.
- **color\_borde** – (str) valor por defecto: “orchid”. Corresponde al color de los bordes de la red, se puede asignar el nombre de un color predefinido o el código hexadecimal de un color.
- **color\_nodo** – (str) valor por defecto: “silver”. Corresponde al color de los nodos, se puede asignar el nombre de un color predefinido o el código hexadecimal de un color.
- **semilla** – (int) valor por defecto: 123. Estado inicial del generador aleatorio para establecer la posición de los nodos.
- **dim\_figura** – (float, float) valor por defecto: (13, 13). Corresponden al ancho y alto de la figura en pulgadas.

```
exploracion.matriz_coocurrencias (texto, min_frec=1, max_num=200, modo='documento', ven-  
                                   tana=3, tri_sup=True, limpiar=False)
```

Calcula la matriz de co-ocurrencias de un texto.

#### Parámetros

- **texto** – (str o list) Corresponde al texto (o lista de textos/documentos) que se desea analizar.
- **min\_frec** – (int) valor por defecto: 1. Frecuencia mínima de aparición de palabras, si la frecuencia de una palabra es menor a min\_frec, dicha palabra es excluida de la matriz.
- **max\_num** – (int) valor por defecto: 200. Número máximo de palabras a dejar en la matriz (se eligen las más frecuentes).
- **modo** – (str) {“documento”, “ventana”} valor por defecto: “documento”. Corresponde al modo de análisis, con “documento” se calcula la co-ocurrencia de términos sin importar la distancia entre estos, con “ventana” se calcula la co-ocurrencia de términos teniendo en cuenta una distancia máxima entre estos.

- **ventana** – (int) valor por defecto: 3. Tamaño de la ventana (solo se usa cuando modo=”ventana”). Número de palabras anteriores o posteriores a tener en cuenta con respecto al término de análisis, equivalente a calcular la co-ocurrencia con n-gramas, siendo  $n=ventana+1$ .
- **tri\_sup** – (bool) {True, False} valor por defecto: True. Si el valor es True devuelve la versión diagonal superior de la matriz de co-ocurrencias, si es False devuelve la matriz completa.
- **limpiar** – (bool) {True, False}. Valor por defecto: False. Define si se desea hacer una limpieza básica (aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto de entrada, antes de calcular las co-ocurrencias.

**Devuelve** dataframe de pandas con las co-ocurrencias de los textos de entrada.

```
exploracion.nube_palabras(texto, n_grama=1, n_terminos=100, graficar=True, dim_figura=10,
                           10, hor=0.6, titulo='Términos más frecuentes', ubicacion_archivo="",
                           mask=None, semilla=1234, devolver_nube=False)
```

Permite graficar o exportar una nube de palabras (n-gramas) a partir de un texto.

#### Parámetros

- **texto** – (str) Corresponde al texto que se desea analizar.
- **n\_grama** – (int) valor por defecto: 1. Cantidad de elementos a tener en cuenta en la generación de n-gramas.
- **n\_terminos** – (int) valor por defecto: 100. Cantidad de n-gramas a graficar.
- **graficar** – (bool) {True, False} valor por defecto: True. Permite visualizar la gráfica en el IDE que esté utilizando.
- **dim\_figura** – (float, float) valor por defecto: (10, 10). Corresponden al ancho y alto de la figura en pulgadas.
- **hor** – (float) (valor de 0 a 1). Corresponde a la orientación de las palabras en el gráfico, siendo 0 una distribución vertical, 1 una distribución horizontal y una distribución mixta a cualquier valor entre 0 y 1.
- **titulo** – (str) valor por defecto: “Términos más frecuentes”. Corresponde al título de la nube de palabras.
- **ubicacion\_archivo** – (str) valor por defecto: vacío. Ruta donde desea exportar la gráfica como archivo tipo imagen. Al nombrar el archivo se recomienda utilizar la extensión jpg. Si no se especifica una ruta, la gráfica no se exporta.
- **mask** – (array) o None, valor por defecto: None. Correspondiente a la máscara base donde se dibujan las palabras, por defecto se utiliza una máscara circular.
- **semilla** – (int) valor por defecto: 1234. Corresponde al estado inicial del generador, este incide en la posición y color de las palabras. En caso de querer replicar la nube de palabras, se recomienda utilizar un mismo valor de semilla.
- **devolver\_nube** – (bool) {True, False} valor por defecto: False. Indica si desea obtener la nube de palabras como un objeto tipo WordCloud.

**Devuelve** objeto tipo WordCloud, solo si la variable devolver\_nube=True.

```
exploracion.obtener_ngramas(texto, n=1, devolver_lista=True, limpiar=False)
```

Permite generar n-gramas a partir de un texto.

#### Parámetros

- **texto** – (str) Corresponde al texto que se desea analizar.

- **n** – (int) Cantidad de elementos a tener en cuenta en la generación de n-gramas. Por ejemplo, si  $n=1$  se retornarán palabras, y si  $n=2$  se retornarán bigramas.
- **devolver\_lista** – (bool) {True, False} valor por defecto: True. Si el valor es True se retorna un objeto tipo lista; si el valor es False se retorna un objeto tipo generador.
- **limpiar** – (bool) {True, False}. Valor por defecto: False. Define si se desea hacer una limpieza básica (aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto de entrada, antes de encontrar los n-gramas.

**Devuelve** n-gramas generados con las características especificadas.

`exploracion.par_nubes (texto, n1=1, n2=2, dim_figura=20, 11, ubicacion_archivo="", graficar=True)`

Permite graficar o exportar un par de nubes de palabras (una junto a otra) a partir de un texto.

#### Parámetros

- **texto** – (str) Corresponde al texto que se desea analizar.
- **n1** – (int) valor por defecto: 1. Cantidad de elementos a tener en cuenta en la generación de n-gramas de la nube de palabras izquierda.
- **n2** – (int) valor por defecto: 2. Cantidad de elementos a tener en cuenta en la generación de n-gramas de la nube de palabras derecha.
- **dim\_figura** – (float, float) valor por defecto: (20, 10). Corresponden al ancho y alto de la figura en pulgadas.
- **ubicacion\_archivo** – (str) valor por defecto: "". Ruta donde desea exportar la gráfica como archivo tipo imagen. Al nombrar el archivo se recomienda utilizar la extensión jpg. Si no se especifica una ruta, la gráfica no se exporta.
- **graficar** – (bool) {True, False} valor por defecto: True. Permite visualizar la gráfica en el [IDE](#) que esté utilizando.

## Lectura

Código, funciones y clases relacionadas a la carga y lectura de diferentes tipos de archivo (word, txt, rtf, pdf inicialmente).

**class** lectura.**Lector** (*ubicacion\_archivo*)

Bases: object

Constructor por defecto de la clase Lector. Esta clase se encarga de extraer el texto de archivos de distintos tipos como Word, PDF, CSV, TXT, RTF e imágenes

**Parámetros ubicacion\_archivo** – (string). Ruta del archivo que se desea leer

**archivo\_a\_texto** (*tipo='inferir', extraer\_medios=False, dir\_medios='temp/img\_dir',  
por\_paginas=False, encoding='utf-8', ocr=False, preprocesamiento=3,  
lenguaje='spa', oem=2, psm=3, password=None*)

Se lleva a cabo la lectura del texto de un archivo y permite escoger el tipo, si es por páginas, la codificación, si se utiliza OCR, el tipo de preprocesamiento, entre otros.

#### Parámetros

- **tipo** – (string) {"inferir", "txt", "csv", "pdf", "rtf", "doc", "docx", "npg", "jpg", "jpeg"}  
Valor por defecto: "inferir". Se define el tipo (o extensión) del archivo que se desea leer
- **extraer\_medios** – (bool) {True, False}. Valor por defecto: False. Especifica si se desean extraer las imágenes dentro del archivo de Word para ser guardadas aparte como archivos de imágenes. Funciona únicamente para archivos ".docx" (no ".doc") y si el parámetro "por\_paginas" es False.
- **dir\_medios** – (string). Valor por defecto: "temp/img\_dir/" Ruta de la carpeta donde se guardan las imágenes del archivo Word cuyas imágenes se extrajeron (se especificó extraer\_medios = True)
- **por\_paginas** – (bool) {True, False}. Valor por defecto: False. Se define si se desea extraer el texto por páginas.
- **encoding** – (string). Valor por defecto: "utf-8". Especifica la codificación del texto que se desea leer

- **ocr** – (bool) {True, False}. Valor por defecto: False Especifica si se desea utilizar reconocimiento óptico de caracteres sobre el archivo cuyo texto se quiere extraer. Se utiliza usualmente cuando el archivo es una imagen o documento escaneado
- **preprocesamiento** – (int) {1,2,3,4,5}. Valor por defecto: 3 Especifica el nivel de preprocesamiento que se lleva a cabo antes de extraer el texto del archivo. Aplica cuando se utiliza reconocimiento óptico de caracteres (parámetro ocr es True). Las opciones son las siguientes: 1: se convierte la imagen a escala de grises. 2: se convierte la imagen a escala de grises y se aplica blurring. 3: se convierte la imagen a escala de grises y se aplica el umbral de imagen con el método de OTSU. 4: se endereza el texto, se convierte la imagen a escala de grises y se aplica umbral adaptativo. 5: se endereza el texto, se convierte la imagen a escala de grises, se aplica umbral de imagen con el método de OTSU, blurring y umbral adaptativo.
- **lenguaje** – (string). {"es", "en"}. Valor por defecto: "spa" Se define el lenguaje del texto que se desea extraer. Tiene las opciones de español ("es") e inglés ("en"). Aplica cuando se extrae el texto de imágenes o archivos escaneados
- **oem** – (int) {0, 1, 2, 3}. Valor por defecto: 2. OEM hace referencia al modo del motor OCR (OCR engine mode en inglés). Tesseract tiene 2 motores, Legacy Tesseract y LSTM, y los parámetros de "oem" permiten escoger cada uno de estos motores por separado, ambos al tiempo o automáticamente:  
  
0: utilizar únicamente el motor Legacy. 1: utilizar únicamente el motor de redes neuronales LSTM. 2: utilizar los motores Legacy y LSTM. 3: escoger el motor según lo que hay disponible.
- **psm** – (int) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}. Valor por defecto: 3 PSM hace referencia a los modos de segmentación de las páginas (page segmentation modes, en inglés) de la librería Pytesseract. Cada número hace referencia a un modo de segmentación: 0: orientation y detección de script (OSD) únicamente. 1: segmentación automática de páginas con OSD. 2: segmentación automática de páginas sin OSD ni OCR. 3: segmentación completamente automática de páginas sin OSD. 4: supone una única columna de texto de tamaños variables. 5: supone un único bloque uniforme de texto alineado de forma vertical. 6: asume un único bloque uniforme de texto. 7: trata la imagen como una única línea de texto. 8: trata la imagen como una única palabra. 9: trata la imagen como una única palabra dentro de un círculo. 10: trata la imagen como un único carácter. 11: Buscador de texto disperso. Encontrar la mayor cantidad de texto posible sin un orden en particular. 12: Buscador de texto disperso con OSD. 13: trata el texto como una única línea, sin utilizar métodos específicos de Tesseract.
- **password** – (string). Valor por defecto: None. Contraseña del archivo cuyo texto se desea extraer, en caso de requerirlo

**Devuelve** (string). Texto extraído del archivo con la clase Lector

**establecer\_ubicacion** (*ubicacion\_archivo*)

Define la ubicación del archivo que se desea leer

**Parámetros ubicacion\_archivo** – (string). Ruta del archivo que se desea leer

**leer\_imagen** (*preprocesamiento, lenguaje, oem, psm*)

Se lleva a cabo la lectura del texto de archivos de tipo imagen, con extensión "png", "jpg" o "jpeg"

**Parámetros**

- **preprocesamiento** – (int) {1,2,3,4,5}. Especifica el nivel de preprocesamiento que se lleva a cabo antes de extraer el texto del archivo. Aplica cuando se utiliza reconocimiento óptico de caracteres (parámetro ocr es True). Las opciones son las siguientes: 1: se convierte la imagen a escala de grises. 2: se convierte la imagen a escala de grises y se aplica



blurring. 3: se convierte la imagen a escala de grises y se aplica el umbral de imagen con el método de OTSU. 4: se endereza el texto, se convierte la imagen a escala de grises y se aplica umbral adaptativo. 5: se endereza el texto, se convierte la imagen a escala de grises, se aplica umbral de imagen con el método de OTSU, blurring y umbral adaptativo.

- **lenguaje** – (string). {"es", "en"} Se define el lenguaje del texto que se desea extraer. Tiene las opciones de español ("es") e inglés ("en")
- **oem** – (int) {0, 1, 2, 3}. OEM hace referencia al modo del motor OCR (OCR engine mode en inglés). Tesseract tiene 2 motores, Legacy Tesseract y LSTM, y los parámetros de "oem" permiten escoger cada uno de estos motores por separado, ambos al tiempo o automáticamente: 0: utilizar únicamente el motor Legacy. 1: utilizar únicamente el motor de redes neuronales LSTM. 2: utilizar los motores Legacy y LSTM. 3: escoger el motor según lo que hay disponible.
- **psm** – (int) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}. PSM hace referencia a los modos de segmentación de las páginas (page segmentation modes, en inglés) de la librería Pytesseract. Cada número hace referencia a un modo de segmentación: 0: orientation y detección de script (OSD) únicamente. 1: segmentación automática de páginas con OSD. 2: segmentación automática de páginas sin OSD ni OCR. 3: segmentación completamente automática de páginas sin OSD. 4: supone una única columna de texto de tamaños variables. 5: supone un único bloque uniforme de texto alineado de forma vertical. 6: asume un único bloque uniforme de texto. 7: trata la imagen como una única línea de texto. 8: trata la imagen como una única palabra. 9: trata la imagen como una única palabra dentro de un círculo. 10: trata la imagen como un único carácter. 11: Buscador de texto disperso. Encontrar la mayor cantidad de texto posible sin un orden en particular. 12: Buscador de texto disperso con OSD. 13: trata el texto como una única línea, sin utilizar métodos específicos de Tesseract.

**Devuelve** (string). Texto del archivo tipo imagen leído con la clase Lector

**leer\_pdf** (*por\_paginas, ocr, preprocesamiento, lenguaje, oem, psm, password=None*)

Se lleva a cabo la lectura del texto de archivos con extensión ".pdf"

#### Parámetros

- **por\_paginas** – (bool) {True, False}. Especifica si se desea extraer el texto del archivo Word con separador de páginas. Este separador se encuentra como "" dentro del texto extraído.
- **ocr** – (bool) {True, False}. Especifica si se desea utilizar reconocimiento óptico de caracteres sobre el archivo cuyo texto se quiere extraer. Se utiliza usualmente cuando el archivo es una imagen o documento escaneado
- **preprocesamiento** – (int) {1,2,3,4,5}. Especifica el nivel de preprocesamiento que se lleva a cabo antes de extraer el texto del archivo. Aplica cuando se utiliza reconocimiento óptico de caracteres (parámetro ocr es True). Las opciones son las siguientes: 1: se convierte la imagen a escala de grises. 2: se convierte la imagen a escala de grises y se aplica blurring. 3: se convierte la imagen a escala de grises y se aplica el umbral de imagen con el método de OTSU. 4: se endereza el texto, se convierte la imagen a escala de grises y se aplica umbral adaptativo. 5: se endereza el texto, se convierte la imagen a escala de grises, se aplica umbral de imagen con el método de OTSU, blurring y umbral adaptativo.
- **lenguaje** – (string). {"es", "en"} Se define el lenguaje del texto que se desea extraer. Aplica cuando se utiliza reconocimiento óptico de caracteres (el parámetro ocr es True). Tiene las opciones de español ("es") e inglés ("en")
- **oem** – (int) {0, 1, 2, 3}. OEM hace referencia al modo del motor OCR (OCR engine mode en inglés). Tesseract tiene 2 motores, Legacy Tesseract y LSTM, y los parámetros

de “oem” permiten escoger cada uno de estos motores por separado, ambos al tiempo o automáticamente:

0: utilizar únicamente el motor Legacy. 1: utilizar únicamente el motor de redes neuronales LSTM. 2: utilizar los motores Legacy y LSTM. 3: escoger el motor según lo que hay disponible.

- **psm** – (int) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}. PSM hace referencia a los modos de segmentación de las páginas (page segmentation modes, en inglés) de la librería Pytesseract. Cada número hace referencia a un modo de segmentación: 0: orientation y detección de script (OSD) únicamente. 1: segmentación automática de páginas con OSD. 2: segmentación automática de páginas sin OSD ni OCR. 3: segmentación completamente automática de páginas sin OSD. 4: supone una única columna de texto de tamaños variables. 5: supone un único bloque uniforme de texto alineado de forma vertical. 6: asume un único bloque uniforme de texto. 7: trata la imagen como una única línea de texto. 8: trata la imagen como una única palabra. 9: trata la imagen como una única palabra dentro de un círculo. 10: trata la imagen como un único carácter. 11: Buscador de texto disperso. Encontrar la mayor cantidad de texto posible sin un orden en particular. 12: Buscador de texto disperso con OSD. 13: trata el texto como una única línea, sin utilizar métodos específicos de Tesseract.

**Devuelve** (string). Texto del archivo “.pdf” leído con la clase Lector

**leer\_rtf()**

Se lleva a cabo la lectura del texto de archivos con extensión “.rtf”

**Devuelve** (string). Texto del archivo “.rtf” leído con la clase Lector

**leer\_txt** (*encoding='utf-8'*)

Se lleva a cabo la lectura del texto de archivos con extensión “.txt”

**Parámetros encoding** – (string). Valor por defecto: “utf-8”. Especifica la codificación del texto que se desea leer

**Devuelve** (string). Texto del archivo “.txt” leído con la clase Lector

**leer\_word** (*por\_paginas, extraer\_medios, dir\_medios*)

Se lleva a cabo la lectura del texto de archivos con extensión “.docx” o “.doc”

**Parámetros**

- **por\_paginas** – (bool) {True, False}. Especifica si se desea extraer el texto del archivo Word con separador de páginas. Este separador se encuentra como “” dentro del texto extraído.
- **extraer\_medios** – (bool) {True, False}. Especifica si se desean extraer las imágenes dentro del archivo de Word para ser guardadas aparte como archivos de imágenes. Funciona únicamente para archivos “.docx” (no “.doc”) y si el parámetro “por\_paginas” es False.
- **dir\_medios** – (string)- Ruta de la carpeta donde se guardan las imágenes del archivo Word cuyas imágenes se extrajeron (se especificó extraer\_medios = True)

**Devuelve** (string). Texto del archivo “.docx” o “.doc” leído con la clase Lector

`lectura.leer_texto(ubicacion_archivo, tipo='inferir', extraer_medios=False, dir_medios='temp/img_dir/', por_paginas=False, encoding='utf-8', ocr=False, preprocesamiento=3, lenguaje='spa', oem=2, psm=3, password=None)`

Función que se encarga de extraer el texto de un archivo. Permite especificar la ruta del archivo, escoger el tipo, si es por páginas, la codificación, si se utiliza OCR, el tipo de preprocesamiento, entre otros.

**Parámetros**

- **ubicacion\_archivo** – (string). Ruta del archivo que se desea leer
- **tipo** – (string) {“inferir”, “txt”, “csv”, “pdf”, “rtf”, “doc”, “docx”, “npg”, “jpg”, “jpeg”}. Valor por defecto: “inferir”. Se define el tipo (o extensión) del archivo que se desea leer
- **extraer\_medios** – (bool) {True, False}. Valor por defecto: False. Especifica si se desean extraer las imágenes dentro del archivo de Word para ser guardadas aparte como archivos de imágenes. Funciona únicamente para archivos “.docx” (no “.doc”) y si el parámetro “por\_paginas” es False.
- **dir\_medios** – (string). Valor por defecto: “temp/img\_dir/” Ruta de la carpeta donde se guardan las imágenes del archivo Word cuyas imágenes se extrajeron (se especificó extraer\_medios = True)
- **por\_paginas** – (bool) {True, False}. Valor por defecto: False. Se define si se desea extraer el texto por páginas.
- **encoding** – (string). Valor por defecto: “utf-8”. Especifica la codificación del texto que se desea leer
- **ocr** – (bool) {True, False}. Valor por defecto: False Especifica si se desea utilizar reconocimiento óptico de caracteres sobre el archivo cuyo texto se quiere extraer. Se utiliza usualmente cuando el archivo es una imagen o documento escaneado
- **preprocesamiento** – (int) {1,2,3,4,5}. Valor por defecto: 3 Especifica el nivel de pre-procesamiento que se lleva a cabo antes de extraer el texto del archivo. Aplica cuando se utiliza reconocimiento óptico de caracteres (parámetro ocr es True). Las opciones son las siguientes: 1: se convierte la imagen a escala de grises. 2: se convierte la imagen a escala de grises y se aplica blurring. 3: se convierte la imagen a escala de grises y se aplica el umbral de imagen con el método de OTSU. 4: se endereza el texto, se convierte la imagen a escala de grises y se aplica umbral adaptativo. 5: se endereza el texto, se convierte la imagen a escala de grises, se aplica umbral de imagen con el método de OTSU, blurring y umbral adaptativo.
- **lenguaje** – (string). {“es”, “en”}. Valor por defecto: “spa” Se define el lenguaje del texto que se desea extraer. Tiene las opciones de español (“es”) e inglés (“en”). Aplica cuando se extrae el texto de imágenes o archivos escaneados
- **oem** – (int) {0, 1, 2, 3}. Valor por defecto: 2. OEM hace referencia al modo del motor OCR (OCR engine mode en inglés). Tesseract tiene 2 motores, Legacy Tesseract y LSTM, y los parámetros de “oem” permiten escoger cada uno de estos motores por separado, ambos al tiempo o automáticamente:  
0: utilizar únicamente el motor Legacy. 1: utilizar únicamente el motor de redes neuronales LSTM. 2: utilizar los motores Legacy y LSTM. 3: escoger el motor según lo que hay disponible.
- **psm** – (int) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}. Valor por defecto: 3 PSM hace referencia a los modos de segmentación de las páginas (page segmentation modes, en inglés) de la librería Pytesseract. Cada número hace referencia a un modo de segmentación: 0: orientation y detección de script (OSD) únicamente. 1: segmentación automática de páginas con OSD. 2: segmentación automática de páginas sin OSD ni OCR. 3: segmentación completamente automática de páginas sin OSD. 4: supone una única columna de texto de tamaños variables. 5: supone un único bloque uniforme de texto alineado de forma vertical. 6: asume un único bloque uniforme de texto. 7: trata la imagen como una única línea de texto. 8: trata la imagen como una única palabra. 9: trata la imagen como una única palabra dentro de un círculo. 10: trata la imagen como un único carácter. 11: Buscador de texto disperso. Encontrar la mayor cantidad de texto posible sin un orden en particular. 12:

Buscador de texto disperso con OSD. 13: trata el texto como una única línea, sin utilizar métodos específicos de Tesseract.

- **password** – (string). Valor por defecto: None. Contraseña del archivo cuyo texto se desea extraer, en caso de requerirlo

**Devuelve** (string). Texto extraído del archivo especificado con la función “leer\_texto”

---

## Lematización

---

```
class lematizacion.LematizadorSpacy (lenguaje, dict_lemmas=None, dim_modelo='md')
```

Bases: object

Constructor por defecto de la clase LematizadorSpacy. Esta clase se encarga de manejar todas las funciones asociadas a la lematización del texto con la librería Spacy

### Parámetros

- **lenguaje** – (string) {"es", "en", "fr", "de"} Se define el lenguaje del texto a ser tratado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada idioma (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}
- **diccionario** – (dict o string). Diccionario (o *string* con ubicación del archivo JSON que lo contiene) que permite modificar y agregar lemas. Las llaves del diccionario son las palabras completas y los valores del diccionario son los lemas escogidos para cada palabra.
- **dim\_modelo** – (string) {"lg", "md", "sm"}. Se define el tamaño del modelo. "lg" es grande (large), "md" es mediano (medium) y "sm" es pequeño (small). Los modelos más grandes obtienen mejores predicciones pero requieren mayor tiempo de carga

**Devuelve** objeto del tipo de la clase LematizadorSpacy

**establecer\_lenguaje** (*lenguaje*)

Define el lenguaje del lematizador

**Parámetros lenguaje** – (string) {"es", "en", "fr", "de"}. Se define el lenguaje del texto a ser tratado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada idioma (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}

**iniciar\_lematizador** (*dim\_modelo*)

Inicia el lematizador con el tamaño del modelo de lematización basado en los tamaños de la librería Spacy. Los modelos más grandes obtienen mejores predicciones pero requieren mayor tiempo de carga

**Parámetros** **dim\_modelo** – (string) {"lg", "md", "sm"}. Valor por defecto: «md». Se define el tamaño del modelo. "lg" es grande (large), "md" es mediano (medium) y "sm" es pequeño (small).

**lematizar** (*texto*, *limpiar=True*)

Se lleva a cabo el proceso de lematización del texto, el cual puede ser limpiado antes de la lematización.

**Parámetros**

- **texto** – (string). El texto que se desea lematizar
- **limpiar** – (bool) {True, False}. Valor por defecto: True. Especifica si se desea hacer una limpieza básica del texto antes de la lematización

**Devuelve** (string). Retorna el texto lematizado.

**modificar\_lemmas** (*dict\_lemmas*)

Define lemas asociados a palabras escogidas por el usuario. Estos nuevos lemas serán adicionados al diccionario de lemas del lematizador

**Parámetros** **dict\_lemmas** – (dict). Diccionario que permite modificar y agregar lemas. Las llaves del diccionario son las palabras completas y los valores del diccionario son los lemas escogidos para cada palabra

**class** lematizacion.**LematizadorStanza** (*lenguaje*, *modelo\_lemas=""*, *dict\_lemmas=None*, *archivo\_salida=""*)

Bases: object

Constructor por defecto de la clase LematizadorStanza. Esta clase se encarga de manejar todas las funciones asociadas a la lematización del texto con la librería Stanza

**Parámetros**

- **lenguaje** – (string). {"es", "en", "fr", "de"}. Se define el lenguaje del texto a ser tratado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada idioma (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}
- **modelo\_lemas** – (string). Valor por defecto: None. Especifica la ruta de un modelo de lemas personalizado. En caso de ser vacío, se utiliza el modelo genérico de lemas de Stanza
- **diccionario** – (dict o string). Diccionario (o *string* con ubicación del archivo JSON que lo contiene) que permite modificar y agregar lemas. Las llaves del diccionario son las palabras completas y los valores del diccionario son los lemas escogidos para cada palabra.
- **archivo\_salida** – (string). Valor por defecto: None. Especifica la ruta del archivo de salida del modelo de lemas modificado. En caso de ser vacío, el resultado de la lematización se guarda en un archivo temporal que eventualmente será borrado

**establecer\_lenguaje** (*lenguaje*)

Define el lenguaje del lematizador

**Parámetros** **lenguaje** – (string). {"es", "en", "fr", "de"} Se define el lenguaje del texto a ser tratado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr") y alemán ("de"). Se aceptan las siguientes variaciones para cada idioma (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng"

."english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}

**iniciar\_lematizador** (*modelo\_lemas*)

Inicia el lematizador de Stanza y permite ingresar la ruta de un modelo de lematización personalizado. En caso de no especificar la ruta se utilizaría un modelo genérico

**Parámetros modelo\_lemas** – (string). Valor por defecto: None. Especifica la ruta de un modelo de lemas personalizado. En caso de ser vacío, se utiliza el modelo genérico de lemas de Stanza

**lematizar** (*texto*, *limpiar=True*)

Se lleva a cabo el proceso de lematización del texto, el cual puede ser limpiado antes de la lematización.

**Parámetros**

- **texto** – (string). El texto que se desea lematizar
- **limpiar** – (bool) {True, False}. Valor por defecto: True. Especifica si se desea hacer una limpieza básica del texto antes de la lematización

**Devuelve** (string). Retorna el texto lematizado.

**modificar\_lemmas** (*dict\_lemmas*, *archivo\_entrada=""*, *archivo\_salida=""*, *gpu=False*)

Permite modificar o añadir nuevos lemas al modelo original de lematización. Adicionalmente, permite ingresar un modelo personalizado desde una ruta específica.

**Parámetros**

- **dict\_lemmas** – (dict). Diccionario que permite modificar y agregar lemas. Las llaves del diccionario son las palabras completas y el valor el lema escogido para cada palabra
- **archivo\_entrada** – (string). Valor por defecto: None. En caso de ser vacío, se escoge un modelo de lematización genérico. De lo contrario, se especifica la ruta del modelo de lematización personalizado a ser utilizado
- **archivo\_salida** – (string). Valor por defecto: None. Especifica la ruta del archivo de salida del modelo de lemas modificado. En caso de ser vacío, el resultado de la lematización se guarda en un archivo temporal que eventualmente será borrado
- **gpu** – (bool) {True, False}. Valor por defecto: False. Se escoge si correr el proceso de lematización con GPU (True) o CPU (False)

`lematizacion.lematizar_texto(texto, lenguaje='es', libreria='spacy', limpiar=True, lematizador=None, dict_lemmas=None, dim_modelo='md', modelo_lemas="", archivo_salida="")`

Función que retorna un texto lematizado por la librería Spacy o Stanza. Permite escoger el idioma de lematización, si hacer limpieza del texto, modificar los modelos de lematizaciones originales y guardar los modelos modificados

**Parámetros**

- **texto** – (string). Texto de entrada a ser lematizado
- **lenguaje** – (string). {"es", "en", "fr", "de", "auto"} Se define el lenguaje del texto a ser tratado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr"), alemán ("de") y automático ("auto", se adivina el lenguaje). Se aceptan las siguientes variaciones para cada idioma (sin importar acentos ni mayúsculas): español: {"es", "español", "esp", "spanish", "sp", "spa"} inglés: {"en", "eng", "english", "inglés", "ing", "in"} francés: {"fr", "fra", "fre", "french", "francés"} alemán: {"ge", "de", "german", "al", "alemán", "ale"}

- **libreria** – (string). {"spacy", "stanza"}. Valor por defecto: "spacy" Se define la librería de Python de lematización para ser utilizada en " el texto. Las opciones son las librerías Spacy y Stanza
- **limpiar** – (bool) {True, False}. Valor por defecto: True. Especifica si se desea hacer una limpieza básica del texto antes de la lematización
- **lematizador** – objeto de lematización de la librería Spacy o Stanza (LematizadorSpacy o LematizadorStanza). Permite utilizar un solo lematizador para varios textos
- **dict\_lemmas** – (dict). Diccionario que permite modificar y agregar lemas. Las llaves del diccionario son las palabras completas y los valores del diccionario son los lemas escogidos para cada palabra
- **dim\_modelo** – (string) {"lg", "md", "sm"}. Valor por defecto: "md" Se define el tamaño del modelo. "lg" es grande (large), "md" es mediano (medium) y "sm" es pequeño (small). Aplica únicamente para el lematizador Spacy
- **modelo\_lemas** – (string). Valor por defecto: None. Especifica la ruta de un modelo de lemas personalizado. En caso de ser vacío, se utiliza el modelo genérico de lemas de Stanza. Aplica únicamente para la lematización con Stanza
- **archivo\_salida** – (string). Valor por defecto: None. Especifica la ruta del archivo de salida del modelo de lemas modificado. En caso de ser vacío, el resultado de la lematización se guarda en un archivo temporal que eventualmente será borrado. Aplica únicamente para la lematización con Stanza

**Devuelve** (string). Texto lematizado



`lenguajes.detectar_lenguaje` (*texto*, *devolver\_proba=False*)

**Identifica el lenguaje en el que está escrito el texto de entrada.** *Ver ejemplo*

**Parámetros**

- **texto** – (str) Corresponde al texto que se desea analizar.
- **devolver\_proba** – (bool) {True, False}, valor por defecto: False. Indica si se retorna el porcentaje de confiabilidad del lenguaje identificado.

**Devuelve** string del lenguaje identificado siguiendo el estandar [ISO 639-1](#). Si `devolver_proba` es True retorna una tupla.

`lenguajes.traducir_texto` (*texto*, *lenguaje\_destino*)

Permite hacer traducciones a un texto de interés.

**Parámetros**

- **texto** – (str) Corresponde al texto que se desea traducir.
- **lenguaje\_destino** – (str) {"es", "en", "de", "fr"}. Indica el lenguaje al que desea traducir el texto, soporta Español(es), Inglés(en), Alemán(de) y Francés(fr).

**Devuelve** string del texto traducido.

`limpieza.cargar_stopwords(ubicacion_archivo, encoding='utf8')`

Función para cargar las listas de palabras y expresiones que se desean eliminar de un texto a partir de un archivo plano

### Parámetros

- **ubicacion\_archivo** – (str) Ubicación del archivo plano que contiene la lista de palabras y/o lista de palabras separadas por espacios, comas o saltos de línea.
- **encoding** – (str), optional. Codificación del archivo de texto. Por defecto “utf-8”.

### Devuelve

(tuple). tupla que contiene:

lista\_palabras (list): Lista que contiene las palabras que se desean quitar en un texto.

lista\_expresiones (list): Lista que contiene las expresiones que se desean quitar de un texto.

`limpieza.limpieza_basica(texto, quitar_numeros=True)`

Limpieza básica del texto. Esta función realiza una limpieza básica del texto de entrada, transforma todo el texto a letras minúsculas, quita signos de puntuación y caracteres especiales, remueve espacios múltiples dejando solo espacio sencillo y caracteres de salto de línea o tabulaciones.

### Parámetros

- **texto** – (str) Texto de entrada al que se le aplicará la limpieza básica.
- **quitar\_numeros** – (bool), optional. Parámetro para quitar los números dentro del texto. Por defecto *True*, se quitan los números dentro del texto.

**Devuelve** (str) Texto después de la limpieza básica.

`limpieza.limpieza_texto(texto, lista_palabras=[], lista_expresiones=[], ubicacion_archivo=None, n_min=0, quitar_numeros=True, quitar_acentos=False)`

Limpieza completa de texto. Esta función hace una limpieza exhaustiva del texto de entrada. Es capaz de quitar palabras y expresiones contenidas en *lista\_palabras* y *lista\_expresiones*, quita acentos de las palabras, números y palabras de longitud menor a *n\_min*.

### Parámetros

- **texto** – (str) Texto de entrada al que se le aplicará el proceso de limpieza.
- **lista\_palabras** – (list), optional. Lista de palabras que se quieren quitar del texto. Por ejemplo, la lista [*“hola”, “de”, “a”*] eliminará esas palabras.
- **lista\_expresiones** – (list), optional. Lista de expresiones que se quieren quitar al texto. A diferencia de *lista\_palabras*, esta puede contener palabras compuestas. Por ejemplo, [*“San Juan de Dios”, “Distrito Capital”, “fuente de agua”*]; esta lista quitará esas palabras compuestas del texto de entrada.
- **ubicacion\_archivo** – None, optional. Ubicación del archivo plano que contiene la lista de palabras y/o lista de palabras separadas por espacios, comas o saltos de línea. Por defecto es *None*, en caso contrario no es necesario especificar los parámetros *lista\_palabras* y *lista\_expresiones*.
- **n\_min** – (int), optional. Longitud mínima de las palabras aceptadas en el texto de entrada.
- **quitar\_numeros** – (bool), optional. Por defecto *True*. Si *False*, no se quitan los números dentro del texto de entrada.
- **quitar\_acentos** – (bool), optional. Por defecto *False*. Opción para determinar si se quitan acentos (tildes, diéresis, virgulilla) del texto.

**Devuelve** (str) Texto después de la limpieza completa.

`limpieza.lista_apellidos()`

Genera lista de apellidos más comunes del español.

**Devuelve** (list) Lista de apellidos más comunes del español.

`limpieza.lista_geo_colombia(tipo='todos')`

Genera lista de nombres de municipios y departamentos de Colombia.

**Parámetros tipo** – {*“todos”, “municipios”, “departamentos”*}, optional. Permite la selección de los nombres en la lista. Por defecto *tipo=“todos”*, genera nombres de municipios y departamentos, *tipo=“municipios”* genera nombres solo de municipios y *tipo=“departamentos”* genera nombres solo de departamentos.

**Devuelve** (list) Lista de nombres de municipios, departamentos o ambos.

`limpieza.lista_nombres(tipo='todos')`

Genera lista de nombres más comunes del español. Retorna lista con los nombres más comunes, tanto para hombre y mujer del idioma español. La función permite generar lista de nombres solo de mujeres o solo de hombres con el parámetro *tipo*.

**Parámetros tipo** – {*“todos”, “mujeres”, “hombres”*}, optional. Permite generar lista de nombres de: solo nombres de mujeres (*tipo=“mujeres”*), solo nombres de hombres. (*tipo=“hombres”*) o ambos (*tipo=“todos”*). Por defecto *tipo=“todos”*.

**Devuelve** (list) Lista de nombres en español.

`limpieza.lista_stopwords(lenguaje='es')`

Genera una lista de stopwords (palabras que se quieren quitar de un texto). Función que genera una lista de stopwords de un idioma predeterminado. Por defecto, genera stopwords en español.

**Parámetros lenguaje** – (str), optional. Lenguaje de las stopwords. Por defecto *es*, que genera las stopwords más comunes del español.

**Devuelve** (list) Lista de palabras stopwords del idioma seleccionado.

`limpieza.quitar_repetidos(texto, sep=' ', remover_espacios=True)`

Función para quitar frases o palabras repetidas que están separadas por un carácter en específico.

**Parámetros**

- **texto** – (str) Texto de entrada.
- **sep** – (str), optional. Separador determinado para encontrar palabras repetidas. Por defecto es “|”.
- **remover\_espacios** – (bool), optional. Si *True* quita los espacios presentes al inicio y al final de una palabra.

**Devuelve** (str) Texto sin palabras o expresiones repetidas.

`limpieza.remover_acentos(texto)`

Quita los acentos (tildes, diéresis, virgulilla) de un texto de entrada. Esta reemplaza cada carácter con acento en el texto por su equivalente sin acento.

**Parámetros** **texto** – (str) Texto al que se le quieren quitar los acentos.

**Devuelve** (str) Texto sin acentos después de la limpieza.

`limpieza.remover_palabras_cortas(texto, n_min)`

Quita las palabras en el texto con longitud estrictamente menor a *n\_min*.

**Parámetros**

- **texto** – (str) Texto de entrada al que se quitarán las palabras menores a *n\_min*.
- **n\_min** – (int) Longitud mínima de las palabras aceptadas en el texto de entrada.

**Devuelve** (str) Texto sin las palabras de longitud menor a *n\_min*.

`limpieza.remover_stopwords(texto, lista_palabras=[], lista_expresiones=[], ubicacion_archivo=None)`

Quita las palabras y expresiones determinadas de un texto. Esta función quita del texto de entrada, palabras específicas contenidas en *lista\_palabras*, o expresiones de palabras contenidas en *lista\_expresiones*.

**Parámetros**

- **texto** – (str) Texto al cual se le quitarán palabras y expresiones contenidas en *lista\_palabras* y *lista\_expresiones*.
- **lista\_palabras** – list, optional. Lista de palabras que se quieren quitar del texto. Por ejemplo, la lista [*“hola”, “de”, “a”*] eliminará esas palabras.
- **lista\_expresiones** – list, optional. Lista de expresiones que se quieren quitar al texto. A diferencia de *lista\_palabras*, esta puede contener palabras compuestas. Por ejemplo, [*“San Juan de Dios”, “Distrito Capital”, “fuente de agua”*]; esta lista quitará esas palabras compuestas del texto de entrada.
- **ubicacion\_archivo** – None, optional. Ubicación del archivo plano que contiene la lista de palabras y/o lista de palabras separadas por espacios, comas o saltos de línea. Por defecto es *None*, en caso contrario no es necesario especificar los parámetros *lista\_palabras* y *lista\_expresiones*.

**Devuelve** (str) Texto sin las palabras y expresiones incluidas en la limpieza.

**class** stemming.Stemmer(*lenguaje*)

Bases: object

Constructor por defecto de la clase Stemmer. Esta clase se encarga de hacer la operación de *stemming*, o reducción de palabras a su raíz, en textos.

**Parámetros lenguaje** – (string) {"es", "en", "fr", "de"}. Lenguaje de los textos a los que se va a aplicar *stemming*.

**Devuelve** (Stemmer). Objeto del tipo de la clase Stemmer

**establecer\_lenguaje**(*lenguaje*)

Permite definir o cambiar el lenguaje de los textos sobre los cuales va a aplicarse el objeto de la clase Stemmer.

**Parámetros lenguaje** – (string) {"es", "en", "fr", "de"}. Lenguaje de los textos a los que se va a aplicar *stemming*.

**iniciar\_stemmer**()

Inicializa el objeto de la clase *SnowballStemmer* de la librería NLTK, para el lenguaje definido previamente, y lo asigna al atributo «stemmer» del objeto de clase Stemmer.

**stemming**(*texto*, *limpiar=False*)

Aplica *stemming* sobre un texto de entrada, y devuelve el texto resultante.

**Parámetros**

- **texto** – (string). Texto al que se desea aplicar el *stemming*.
- **limpiar** – (bool) {True, False}. Valor por defecto: False. Argumento opcional que define si se desea hacer una limpieza básica ( aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto antes de aplicar el *stemming*.

**Devuelve** (string). Texto luego de la aplicación del *stemming*.

**stemming.stem\_texto**(*texto*, *lenguaje='es'*, *limpiar=False*, *stemmer=None*)

Función que aprovecha la clase Stemmer para realizar *stemming*, o reducción de palabras a su raíz, en un texto de entrada.

### Parámetros

- **texto** – (string). Texto al que se desea aplicar el *stemming*.
- **lenguaje** – (string) {“es”, “en”, “fr”, “de”}. Lenguaje del texto al que se va a aplicar *stemming*.
- **limpiar** – (bool) {True, False}. Valor por defecto: False. Define si se desea hacer una limpieza básica (aplicando la función *limpieza\_basica* del módulo *limpieza*) al texto de entrada, antes de aplicar el *stemming*.
- **stemmer** – (Stemmer). Parámetro opcional. Objeto de la clase Stemmer para aplicar *stemming* sobre el texto de entrada. Se puede utilizar para aplicar *stemming* a varios textos a la vez, sin necesidad de inicializar una instancia de la clase Stemmer en cada ocasión. Esto puede representar ahorro en tiempos al momento de aplicar la función.

**Devuelve** (string). Texto luego de la aplicación del *stemming*.

```
class vectorizacion.VectorizadorDoc2Vec(n_elementos=100, minima_cuenta=5, epocas=20,  
                                         semilla=1, archivo_modelo="")
```

Bases: object

**Constructor de la clase VectorizadorDoc2Vec.** Permite hacer vectorizaciones usando Doc2Vec.

### Parámetros

- **n\_elementos** – (int) valor por defecto: 100. Número de términos a ser tenidos en cuenta en la vectorización.
- **minima\_cuenta** – (int) valor por defecto: 5. Frecuencia mínima que debe tener cada término para ser tenido en cuenta en el modelo.
- **epocas** – (int) valor por defecto: 20. Número de iteraciones que realiza la red neuronal para entrenar el modelo.
- **semilla** – (int) valor por defecto: 1. El modelo tiene un componente aleatorio. Se establece una semilla para poder replicar los resultados.
- **archivo\_modelo** – (str) valor por defecto: vacío. Ruta de archivo de un vectorizador en formato pickle (pk). Permite cargar un vectorizador generado previamente, los demás parámetros de inicialización no serán tenidos en cuenta, pues se tomarán del vectorizador cargado.

```
entrenar_modelo(corpus_entrenamiento, actualizar=False, archivo_salida="")
```

Permite entrenar un modelo a partir de un corpus de entrenamiento.

### Parámetros

- **corpus\_entrenamiento** – (list) lista de textos de interes para entrenar el modelo.
- **actualizar** – (bool) {True, False} valor por defecto: False. Si True, las nuevas palabras en los documentos se agregarán al vocabulario del modelo.
- **archivo\_salida** – (str) valor por defecto: vacío. Ruta donde desea exportar el vectorizador ajustado. Usar formato pickle (pk)

**Devuelve****vectorizar** (*textos*, *alpha*=0.025, *num\_pasos*=50, *semilla*=13)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documentos-términos. **Llama la función `vectorizar_texto`.**

**Parámetros**

- **textos** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **alpha** – (float) valor por defecto: 0.025. Tasa de aprendizaje del modelo.
- **num\_pasos** – (int) valor por defecto: 50. Número de iteraciones usadas para entrenar el nuevo documento. Los valores más grandes toman más tiempo, pero pueden mejorar la calidad y la estabilidad de ejecución a ejecución de los vectores inferidos. Si no se especifica, se reutilizará el valor de épocas de la inicialización del modelo.
- **semilla** – (int) valor por defecto: 13. El vectorizador tiene un componente aleatorio. Se establece una semilla para poder replicar los resultados.

**Devuelve** vectores (numpy.ndarray) documentos-términos de la vectorización de los textos.

**vectorizar\_texto** (*texto*, *alpha*=0.025, *num\_pasos*=50, *semilla*=13)

Vectoriza el texto utilizando el vectorizador. Transformando el texto en un vector documento-términos.

**Parámetros**

- **texto** – string de interés para ser vectorizado.
- **alpha** – (float) valor por defecto: 0.025. Tasa de aprendizaje del modelo.
- **num\_pasos** – (int) valor por defecto: 50. Número de iteraciones usadas para entrenar el nuevo documento. Los valores más grandes toman más tiempo, pero pueden mejorar la calidad y la estabilidad de ejecución a ejecución de los vectores inferidos. Si no se especifica, se reutilizará el valor de épocas de la inicialización del modelo.
- **semilla** – (int) valor por defecto: 13. El vectorizador tiene un componente aleatorio. Se establece una semilla para poder replicar los resultados.

**Devuelve** vectores (numpy.ndarray) documentos-términos de la vectorización de los textos.

```
class vectorizacion.VectorizadorFrecuencias (tipo='bow', rango_ngramas=1, l,  
                                           max_elementos=None, idf=True,  
                                           archivo_modelo="")
```

Bases: object

**Constructor de la clase VectorizadorFrecuencias.** Permite hacer vectorizaciones usando Bag of Words (BOW) o TF-IDF.

**Parámetros**

- **tipo** – (str) {"bow", "tfidf"}, valor por defecto: "bow". Determina el tipo de vectorizador a inicializar.
- **rango\_ngramas** – (tupla) (int, int) valor por defecto: (1, 1). Límite inferior y superior del rango de valores n para los diferentes n-gramas que se van a extraer. Por ejemplo, un rango\_ngramas de (1, 1) significa solo unigramas, (1, 2) significa unigramas y bigramas, y (2, 2) significa solo bigramas.
- **max\_elementos** – (int) valor por defecto: None. Número máximo de términos a ser tenidos en cuenta, se ecogen los max\_elementos términos más frecuentes.
- **idf** – (bool) {True, False} valor por defecto: True. Habilita la reponderación de pesos de los términos usando IDF.



- **archivo\_modelo** – (str) valor por defecto: vacío. Ruta de archivo de un vectorizador en formato pickle (pk). Permite cargar un vectorizador generado previamente, los demás parámetros de inicialización no serán tenidos en cuenta, pues se tomarán del vectorizador cargado.

**ajustar** (*x*, *archivo\_salida=""*)

Permite al vectorizador aprender el vocabulario de acuerdo a los textos de entrada.

#### Parámetros

- **x** – objeto iterable con textos (str), unicode u archivos de texto de interés para ser procesados por el vectorizador.
- **archivo\_salida** – (str) valor por defecto: vacío. Ruta donde desea exportar el vectorizador ajustado. Usar formato pickle (pk)

**fit** (*x*, *archivo\_salida=""*)

Permite al vectorizador aprender el vocabulario de acuerdo a los textos de entrada. **Llama la función ajustar.**

#### Parámetros

- **x** – objeto iterable con textos (str), unicode u archivos de texto de interés para ser procesados por el vectorizador.
- **archivo\_salida** – (str) valor por defecto: vacío. Ruta donde desea exportar el vectorizador ajustado. Usar formato pickle (pk)

**inversa** (*x*)

A partir de un vector o grupo de vectores, devuelve los términos con frecuencia mayor a 0 en el documento.

**Parámetros** **x** – vector o grupo de vectores correspondientes a la vectorización de textos generada por un vectorizador.

**Devuelve** (list) lista de arrays con los términos más frecuentes correspondientes a los vectores de cada texto/documento.

**transform** (*x*, *disperso=False*)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documento-términos. **Llama la función vectorizar.**

#### Parámetros

- **x** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **disperso** – (bool) {True, False} valor por defecto: False. Si es True retorna los resultados como una matriz dispersa (csr\_matrix). Si es False retorna los resultados como un numpy array

**Devuelve** vectores documentos-términos de la vectorización de los textos.

**vectorizar** (*textos*, *disperso=False*)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documentos-términos

#### Parámetros

- **textos** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **disperso** – (bool) {True, False} valor por defecto: False. Si es True retorna los resultados como una matriz dispersa (csr\_matrix). Si es False retorna los resultados como un numpy array

**Devuelve** vectores documentos-términos de la vectorización de los textos.

**vocabulario()**

Retorna el vocabulario del vectorizador entrenado.

**Devuelve** dataframe con el vocabulario del vectorizador.

**class** `vectorizacion.VectorizadorHash` (*n\_elementos=100, rango\_ngramas=1, 1*)

Bases: `object`

**Constructor de la clase VectorizadorHash.** Permite hacer vectorizaciones usando hashing.

#### Parámetros

- **n\_elementos** – (int) Hace referencia al número de elementos o características (columnas) en las matrices de salida.
- **rango\_ngramas** – (tupla) (int, int) valor por defecto: (1, 1). Límite inferior y superior del rango de valores n para los diferentes n-gramas que se van a extraer. Por ejemplo, un rango\_ngramas de (1, 1) significa solo unigramas, (1, 2) significa unigramas y bigramas, y (2, 2) significa solo bigramas.

**transform** (*x, disperso=False*)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documento-términos. **Llama la función vectorizar.**

#### Parámetros

- **x** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **disperso** – (bool) {True, False} valor por defecto: False. Si es True retorna los resultados como una matriz dispersa (csr\_matrix). Si es False retorna los resultados como un numpy array.

**Devuelve** vectores documentos-términos de la vectorización de los textos.

**vectorizar** (*textos, disperso=False*)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documentos-términos

#### Parámetros

- **textos** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **disperso** – (bool) {True, False} valor por defecto: False. Si es True retorna los resultados como una matriz dispersa (csr\_matrix). Si es False retorna los resultados como un numpy array

**Devuelve** vectores documentos-términos de la vectorización de los textos.

**class** `vectorizacion.VectorizadorWord2Vec` (*lenguaje='es', dim\_modelo='md'*)

Bases: `object`

**Constructor de la clase VectorizadorWord2Vec.** Permite hacer vectorizaciones utilizando Word2Vec. Utiliza un modelo pre entrenado.

#### Parámetros

- **lenguaje** – (string) {"es", "en", "fr", "it"} Define el lenguaje del corpus utilizado al pre entrenar el vectorizador, el lenguaje debe corresponder al idioma del texto a ser analizado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr"), italiano ("it") y otros.

- **dim\_modelo** – (str) {"sm", "md", "lg"}, valor por defecto: "md". Define el tamaño del corpus utilizado al pre entrenar el vectorizador. Siendo sm - pequeño, md - mediano, lg - grande.

#### **establecer\_lenguaje** (*lenguaje*)

Permite establecer el lenguaje a ser utilizado por el vectorizador.

**Parámetros lenguaje** – (string) {"es", "en", "fr", "it"} Define el lenguaje del corpus utilizado al pre entrenar el vectorizador, el lenguaje debe corresponder al idioma del texto a ser analizado. Los lenguajes posibles son español ("es"), inglés ("en"), francés ("fr"), italiano ("it") y otros.

#### **iniciar\_vectorizador** (*dim\_modelo*)

Permite cargar el modelo de vectorizador a utilizar.

**Parámetros dim\_modelo** – (str) {"sm", "md", "lg"}, valor por defecto: "md". Define el tamaño del corpus utilizado al pre entrenar el vectorizador. Siendo sm - pequeño, md - mediano, lg - grande.

#### **similitud\_textos** (*t1, t2*)

Calcula la similitud coseno entre 2 palabras o textos.

##### **Parámetros**

- **t1** – (str) primer texto de interés para el cálculo de similitud.
- **t2** – (str) segundo texto de interés para el cálculo de similitud.

**Devuelve** (float) valor entre 0 y 1, donde 1 representa que los textos son iguales.

#### **vectores\_palabras** (*texto, tipo='diccionario'*)

Retorna las palabras y vectores que pertenecen a un texto.

##### **Parámetros**

- **texto** – (str) texto de interés a ser procesado.
- **tipo** – (str) {"diccionario", "dataframe"}, valor por defecto: "diccionario". Si es "diccionario" retorna los resultados como un objeto tipo diccionario, si es "dataframe", retorna los resultados como un objeto tipo dataframe.

**Devuelve** palabras y vectores del texto.

#### **vectorizar** (*textos, quitar\_desconocidas: bool = False*)

Vectoriza los textos utilizando el vectorizador. Transformando los textos en una matriz documentos-términos. **Llama la función vectorizar\_texto.**

##### **Parámetros**

- **textos** – string (str) o lista (list) con textos de interés para ser vectorizados.
- **quitar\_desconocidas** – (bool) {True, False} valor por defecto: False. Cuando este argumento es False, para cada palabra desconocida se incluirá un vector de solo ceros, lo que afectará el vector promedio resultante. Si es True hará que la función sea ligeramente más demorada, pero quizás más precisa, al no tener en cuenta palabras que no están incluidas en el modelo.

**Devuelve** vectores (numpy.ndarray) documentos-términos de la vectorización de los textos.

#### **vectorizar\_texto** (*texto, quitar\_desconocidas: bool = False*)

Vectoriza el texto utilizando el vectorizador. Transformando el texto en un vector documento-términos.

##### **Parámetros**

- **texto** – string de interés para ser vectorizado.
- **quitar\_desconocidas** – (bool) {True, False} valor por defecto: False. Cuando este argumento es False, para cada palabra desconocida se incluirá un vector de solo ceros, lo que afectará el vector promedio resultante. Si es True hará que la función sea ligeramente más demorada, pero quizás más precisa, al no tener en cuenta palabras que no están incluidas en el modelo.

**Devuelve** vector (numpy.ndarray) documento-términos de la vectorización del texto.

## 15.1 Auxiliares

`utils.auxiliares.cargar_objeto` (*nombre\_archivo*)

Carga un objeto en Python, desde un archivo Pickle cuya ubicación es determinada por el usuario.

**Parámetros** `nombre_archivo` – (str). Ubicación del archivo que contiene el objeto que se desea cargar.

**Devuelve** (objeto Python). Objeto en Python contenido en el archivo.

`utils.auxiliares.guardar_objeto` (*objeto, nombre\_archivo*)

Guarda, en un archivo Pickle, un objeto de Python determinado por el usuario.

**Parámetros**

- **objeto** – (objeto Python). Objeto que se desea guardar.
- **nombre\_archivo** – (str). Ubicación y nombre del archivo en donde se desea guardar el objeto.

`utils.auxiliares.verificar_crear_dir` (*ubicacion\_directorio*)

Verifica si existe un directorio en la ubicación determinada por el usuario. Si el directorio no existe, la función lo crea.

**Parámetros** `ubicacion_directorio` – (str). Ubicación del directorio que se desea verificar o crear.

## 15.2 Tokenización

**class** `utils.tokenizacion.TokenizadorNLTK` (*tokenizador=None, destokenizador=None*)

Bases: `object`

Constructor por defecto de la clase `TokenizadorNLTK`. Esta clase se apoya en la librería `NLTK` para definir acciones de tokenización y detokenización (operación inversa en la que se pasa de tokens a texto) de textos.

### Parámetros

- **tokenizador** – (objeto de tokenización de `NLTK`) Valor por defecto: `None`. Objeto encargado de la tokenización de textos. Si el valor es “`None`”, se cargará por defecto una instancia de la clase `ToktokTokenizer`, de la librería `NLTK`.
- **destokenizador** – (objeto de detokenización de `NLTK`) Valor por defecto: `None`. Objeto encargado de la detokenización de textos. Si el valor es “`None`”, se cargará por defecto una instancia de la clase `TreebankWordDetokenizer`, de la librería `NLTK`.

**destokenizar** (*lista\_tokens*)

Realiza la función de detokenización (unir una lista de tokens, produciendo un texto) sobre una o varias listas de tokens de entrada.

**Parámetros lista\_tokens** – (list). Lista de tokens, si es para un solo texto. Si es para varios textos, se introduce una lista en la que cada elemento (uno para cada texto) es una lista de tokens.

**Devuelve** (str o lista de strings). Devuelve un solo string si se introdujo solo una lista de tokens. Si se introdujeron varias listas de tokens, devuelve una lista de strings.

**establecer\_destokenizador** (*destokenizador*)

Permite definir o cambiar el detokenizador a utilizar.

**Parámetros destokenizador** – (objeto de detokenización de `NLTK`). Objeto encargado de la detokenización de textos. Si el valor es “`None`”, se cargará por defecto una instancia de la clase `TreebankWordDetokenizer`, de la librería `NLTK`.

**establecer\_tokenizador** (*tokenizador*)

Permite definir o cambiar el tokenizador a utilizar.

**Parámetros tokenizador** – (objeto de tokenización de `NLTK`). Objeto encargado de la tokenización de textos. Si el valor es “`None`”, se cargará por defecto una instancia de la clase `ToktokTokenizer`, de la librería `NLTK`.

**post\_destokenizacion** (*texto*)

Hace algunos ajustes al texto, una vez ha pasado por la detokenización, para que cumpla con las reglas de puntuación de los idiomas español e inglés. Es posible que para otros idiomas sea necesario incluir ajustes adicionales. Si el texto de entrada no contiene signos de puntuación (por ejemplo, si fue pasado por alguna función de limpieza previamente), la salida será igual a la entrada.

**Parámetros texto** – (str). Texto que resulta de detokenizar una lista de tokens.

**Devuelve** (str). Texto con los signos de puntuación ajustados.

**tokenizar** (*texto*)

Realiza la función de tokenización (separar un texto en componentes sueltos, o tokens) sobre uno o varios textos de entrada.

**Parámetros texto** – (str o lista de strings). Texto o lista de textos sobre los cuales se desea aplicar la tokenización.

**Devuelve** (list). Si se ingresó un solo texto, devuelve la lista de tokens del texto. Si se ingresó una lista de textos, se devuelve una lista en la que cada elemento es una lista de tokens, con un elemento para cada texto de entrada.

`utils.tokenizacion.destokenizar(tokens, tokenizador=None)`

Función que aprovecha la clase `TokenizadorNLTK` para realizar la función de detokenización (unir una lista de tokens, produciendo un texto) sobre una o varias listas de tokens de entrada.

#### Parámetros

- **tokens** – (list). Lista de tokens, si es para un solo texto. Si es para varios textos, se introduce una lista en la que cada elemento (uno para cada texto) es una lista de tokens.
- **tokenizador** – Valor por defecto: `None`. Objeto encargado de la tokenización y detokenización de textos. Si el valor es “None”, se cargará por defecto una instancia de la clase `TokenizadorNLTK`.

**Devuelve** (str o lista de strings). Devuelve un solo string si se introdujo solo una lista de tokens. Si se introdujeron varias listas de tokens, devuelve una lista de strings.

`utils.tokenizacion.tokenizar(texto, tokenizador=None)`

Función que aprovecha la clase `TokenizadorNLTK` para realizar la función de tokenización (separar un texto en componentes sueltos, o tokens) sobre uno o varios textos de entrada.

#### Parámetros

- **texto** – (str o lista de strings). Texto o lista de textos sobre los cuales se desea aplicar la tokenización.
- **tokenizador** – Valor por defecto: `None`. Objeto encargado de la tokenización y detokenización de textos. Si el valor es “None”, se cargará por defecto una instancia de la clase `TokenizadorNLTK`.

**Devuelve** (list). Si se ingresó un solo texto, devuelve la lista de tokens del texto. Si se ingresó una lista de textos, se devuelve una lista en la que cada elemento es una lista de tokens, con un elemento para cada texto de entrada.



La librería de procesamiento y análisis de texto, ConTexto, tiene como objetivo principal proporcionar herramientas que simplifiquen las tareas y proyectos que involucren análisis de texto. La librería fue desarrollada en el lenguaje de programación de *Python* y contiene un conjunto de funciones que permiten realizar transformaciones y análisis de textos de forma simple, utilizando diferentes técnicas, para lectura y escritura de archivos de texto, incluyendo reconocimiento óptico de caracteres (OCR), limpieza de textos y remoción de palabras no deseadas para el análisis (*stop words*), traducción y corrección de textos, generación de nubes de palabras, cálculo de similitudes entre textos, entre otras, reconocidas por su buen desempeño.

La librería surge como solución a tres principales aspectos, primero, la necesidad de integrar todos los esfuerzos y desarrollos que ha hecho la Unidad de Científicos de Datos (UCD) del DNP, en proyectos relacionados con la analítica de texto, segundo, evitar reprocesos en la construcción de scripts para estas tareas, y finalmente, contribuir en la reducción de la escasez de librerías enfocadas en el análisis de texto en español que existe actualmente.

Esta página contiene toda la información relacionada con la librería, en el panel de navegación se tiene acceso a las diferentes secciones, las cuales cubren la instalación de la librería, la documentación de los módulos y funciones, ejemplos y demás información de interés.



### C

comparacion, ??  
correccion, ??

### e

escritura, ??  
exploracion, ??

### I

lectura, ??  
lematizacion, ??  
lenguajes, ??  
limpieza, ??

### S

stemming, ??

### U

utils.auxiliares, ??  
utils.tokenizacion, ??

### V

vectorizacion, ??