Universidad Rey Juan Carlos

TEMA 2 – HOJA DE EJERCICIOS II

Se proponen diferentes ejercicios breves de procesamiento básico. **Todos los ejercicios propuestos en el listado se deben resolver utilizando la librería spaCy**.

spaCy (https://spacy.io/) es una librería de software de código abierto diseñada para facilitar tareas de procesamiento avanzado del lenguaje natural. Es fácil de usar y destaca por su velocidad a la hora de realizar el procesamiento del texto.

Las diferentes opciones de instalación de spaCy se pueden encontrar aquí.

Pipeline en spaCy. Cuando se llama a nlp sobre un string, spaCy primero genera tokens del texto y crea un objeto del tipo Doc (ver Figura 1).

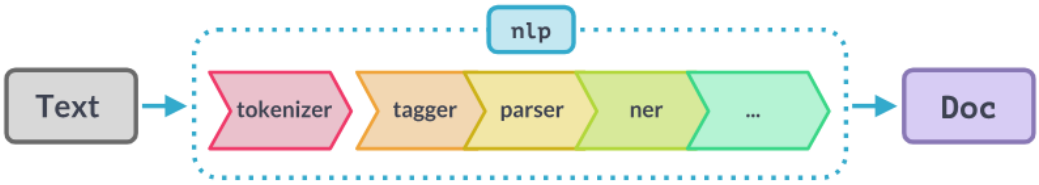

Figura 1. Pipeline al procesar un texto con spaCy.

Es necesario descargar los modelos de los lenguajes que se quieran utilizar. Los modelos disponibles se pueden consultar aquí.

Ejercicios:

1. Dado el siguiente texto en español:

   "*El 2025 viene cargado de innovaciones que repercutirán en nuestro modo de vida y de trabajo diario. Este año, las tendencias apuestan por los avances en inteligencia artificial (IA), automatización, conectividad y sostenibilidad. No sólo prometen una mayor eficiencia en el trabajo y una optimización de los procesos de decisión, sino también la superación de muchos de los retos actuales. Estas tendencias brindarán nuevas oportunidades tanto a empresas como a particulares. Echa un vistazo a las 5 tendencias tecnológicas más importantes, que marcarán el 2025.”*

   Hay que tokenizarlo y hacer lo siguiente:

   a. Mostrar cada token en una línea diferente.

   b. Añadir los diferentes tokens en una lista y luego mostrar la lista.

c. Mostrar para cada token y en una línea diferente, el texto del token, la categoría gramatical y el lema.

d. Contar el número de palabras vacías que se encuentran en el texto. Imprimir el número de palabras vacías y luego imprimir el número y las palabras vacías sin repeticiones.

e. Mostrar los adjetivos que aparecen en el texto.

f. Mostrar los nombres que aparecen en el texto.

g. Mostrar para cada token su categoría gramatical y la explicación asociada a dicha categoría. Para ello se puede utilizar *spacy.explain*, que da una descripción acerca del tag particular del token.

2. Dado el mismo texto en español del ejercicio anterior, se pide lo siguiente:

   a. Imprimir el número de oraciones que hay en el párrafo e imprimir cada una de ellas.

   b. Para cada oración, eliminar las palabras vacías y formar la oración de nuevo sin ellas.

3. Dado el mismo texto en español de los ejercicios anteriores se quieren calcular estadísticas sobre frecuencias de las palabras del texto. Para ello, primero eliminar las palabras vacías y los signos de puntuación y sobre el listado de palabras resultante se pide:

   a. Obtener la frecuencia de cada palabra, imprimiendo el par palabra-frecuencia.

   b. Mostrar sólo aquellas palabras cuya frecuencia sea mayor que 1

4. Implementar una función para preprocesar un texto con operaciones típicas para normalizarlo, es decir, dado un texto se debe convertir a minúscula, lematizar cada token, eliminar *stop words* y los signos de puntuación. (Lo que se pide es agrupar estas operaciones en una única función para que se pueda invocar como una "única acción" en un momento dado).

5. Dados tres textos cortos en español, dos de la misma temática y uno de temática diferente (openAustralia1.txt, openAustralia2.txt y crisisUcrania.txt), se pide:

    a. Comprobar cuál es su similitud (entre cada par de documentos), haciendo uso del atributo similarity de la clase Doc y sin hacer ningún tipo de preprocesamiento del texto.

    b. Comprobar cuál es su similitud una vez que se procesan los textos eliminando signos de puntuación, espacios, palabras vacías y aquellos tokens con longitud inferior a 4 caracteres. Una vez realizado este preprocesamiento, se obtendría de nuevo un objeto de tipo Doc para cada uno y se comprobaría la similitud.

    c. ¿Y si además del preprocesamiento anterior se lematizan los tokens? ¿Hay variaciones en la similitud?

6. Dado el texto contenido en el archivo "atp.txt" se pide extraer la siguiente información utilizando el matcher de spacy:

    a. Poner un patrón para obtener la cadena "(ATP)"

    b. Poner un patrón para obtener la cadena "Súper challengers"

    c. Poner un patrón para obtener las cadenas de texto con nombres y adjetivos.

    Los atributos válidos para especificar en los patrones, así como los operadores de cuantificación, entre otras cosas, se pueden encontrar aquí: https://spacy.io/usage/rule-based-matching.

7. Dada toda la colección de textos en español, se quiere extraer, para cada documento, la siguiente información utilizando el matcher de spacy:

    a. Las cadenas de texto con nombres y adjetivos

    b. Las cadenas de texto con un número seguido de un nombre

    c. Las cadenas de texto con un nombre propio seguido de un adjetivo

    d. Las cadenas de texto que contengan el lema "tener"

    e. Encontrar diferentes cadenas de texto en los documentos. Se deberá hacer uso del matcher de frases (PhraseMatcher). Las cadenas para buscar son las siguientes: "actividad física", "cuerpo saludable", "gobierno federal" y "Grand Slam"

8. Se puede combinar el uso de patrones con expresiones regulares.

    a. Buscar cadenas de texto que comiencen por la letra ‘a’ y vayan seguidas de un adjetivo. Pista: se pondría el diccionario del patrón como sigue {"TEXT": {"REGEX": "ExpRegular"}}