Práctica 3

Realicen los siguientes ejercicios. Recuerden **revisaré todo en GitHub**.

1. Distancia de Hamming

En un script independiente, creen una función que calcule la distancia de Hamming entre dos cadenas.

2. Análisis de Frecuencias

En un segundo script, realizarán el análisis de frecuencias de las palabras del texto "practica3.txt", tal y como se pide en las siguientes secciones. El código se puede organizar como lo deseen, pero el resultado final de las tres secciones debe aparecer al llamar sólo ese script.

2.1. Tokenización

Primero, será necesario tokenizar el documento. En el Listing 1 se muestra un ejemplo de tokenización con NLTK.

```
import nltk
from nltk.tokenize.toktok import ToktokTokenizer

# Tokenizador TokTok (palabras)
toktok = ToktokTokenizer()

# Tokenizador de oraciones
es_tokenizador_oraciones = nltk.data.load('tokenizers/punkt/spanish.
    pickle')

# Obtener oraciones de un parrafo
parrafo = "Este␣es␣un␣texto␣de␣prueba.␣¿Crees␣que␣pueda␣tokenizar␣bien␣
    cada␣oración?␣¡Ya␣lo␣veremos!"
oraciones = es_tokenizador_oraciones.tokenize(parrafo)

# Obtener tokens de cada oración
for s in oraciones:
        print([t for t in toktok.tokenize(s)])
```

Listing 1: Tokenizando con NLTK

- Creen una función de tokenización que reciba la ruta de un archivo y regrese una lista de tokens.

2.2. Distribución de Frecuencias

Tokenicen el texto "practica3.txt" con su función de tokenización, y encuentren la distribución de frecuencias de todos los tipos en el vocabulario.

- Impriman las frecuencias en orden decreciente en pantalla (usen el método `sort` o la función `sorted`).

2.3. Reducción de dimensionalidad

NLTK tiene una lista de las palabras funcionales en Español. Ésta se puede obtener como se muestra en el Listing 2.

```
from nltk.corpus import stopwords

print(stopwords.words("spanish"))
```

Listing 2: Palabras funcionales con NLTK

- Usen NLTK para remover las palabras funcionales de la tokenización del archivo "practica3.txt".

2.3.1. Distribución de Frecuencias con Stemming

NLTK tiene una implementación de la versión 2 del algoritmo de Porter (*Snowball*). El Listing 3 muestra cómo cargar las reglas para Español.

```
from nltk.stem.snowball import SnowballStemmer

# Stemmer en Español
stemmer = SnowballStemmer("spanish")

tokens = [] # <- aquí va el texto tokenizado
for t in tokens:
        # Obtener la raiz
        print(stemmer.stem(t))
```

Listing 3: Stemming con NLTK

- Apliquen el algoritmo de *stemming* a todos los tokens del archivo "practica3.txt" sin palabras funcionales y encuentren la distribución de frecuencias.
- Impriman las frecuencias en orden decreciente en pantalla (usen el método `sort` o la función `sorted`).

2.4. Distribución de Frecuencias con Lematización ingenua

Finalmente:

- Creen una función de lematización ingenua usando el archivo "lemmatization-es.txt", el cual contiene un conjunto de palabras con sus respectivos lemas.
- Usen su función para lematizar todos los tokens del archivo "practica3.txt" sin palabras funcionales y encuentren la distribución de frecuencias.
- Impriman las frecuencias en orden decreciente en pantalla (usen el método `sort` o la función `sorted`).

3. Corrector ortográfico simple

Utilicen la distancia de Levenshtine (`from nltk.metrics.distance import edit_distance`) para hacer un corrector ortográfico simple que corrija la ortografía del archivo "corrigeme.txt".

- Tienen "lemmatization-es.txt" y "listado_general.txt" como recursos auxiliares.
- Escriban el resultado en un archivo nuevo, títulado "corregido.txt".
    - **Pongan una oración por cada línea de texto.**

El código tiene que estar en un script independiente de los dos anteriores.

4. Repositorio

Guarden su código en GitHub. En el README del repositorio, pongan:

1. El contenido del archivo "corregido.txt".
2. La distribución de frecuencias de "corregido.txt" (**no** de "practica3.txt"), en orden decreciente, y sin palabras funcionales:
    - Sin cambios
    - Con stemming
    - Con lematización