Práctica 1

Integrantes:

Aguilar Torres Karla Daniela
Lara Sala Kevin Arturo
Martínez Martínez Vanessa

OBJETIVO

Preprocesar un corpus a partir de métodos basados en lenguajes formales y tokenizarlo en subpalabras.

DESARROLLO

1. **Escoger un corpus de cualquier idioma y de un tamaño mayor a 10 000 tokens (se puede tomar este corpus de la paquetería nltk.corpus). Este corpus se usará a lo largo del curso.**

   Para este caso se siguió la recomendación del profesor, donde se hace uso del conjunto de bibliotecas nltk. Elegimos el corpus de noticias en español cess_esp que incluye anotación morfo-sintáctica

```
1  import string
2  import nltk
3  from nltk.corpus import cess_esp #Corpus español
```

Figura 1: Título de la imagen.

2. **Limpiar el corpus: eliminar signos de puntuación, de interrogación, admiración y elementos no léxicos, y en general aquellos elementos ruidosos**

   Importamos la herramienta stopwords de la biblioteca nltk para ayudarnos a limpiar nuestro corpus de palabras en español que no aportan nada.

   Convertimos todas nuestras palabras a minúsculas con la función lower(), la función isalpha() nos ayudará a verificar que todo el alfabeto de nuestro corpus pertenezca de la a-z, esto limpiará nuestro Corpus de todos los signos y otros caracteres que pudiera haber. Por último creamos dos variables donde mostrará el número de tokens y tipos que contiene nuestro Corpus. Por último, separamos las palabras en caracteres separados [Línea 21].

```
4   from nltk.corpus import stopwords
5
6   import BPE
7
8   corpus = [i.lower() for i in cess_esp.words()]
9
10  tokens = len(corpus)
11  tipos = len(set(corpus))
12  print('Número de tókens: ', tokens)
13  print('Número de tipos: ', tipos)
14
15  clean_corpus = [i for i in corpus if i.isalpha()]
16
17  stopwords = set(stopwords.words('spanish'))
18  print()
19
20  clean_corpus = [i for i in clean_corpus if i not in stopwords]
21  clean_corpus = [' '.join(list(x)) for x in clean_corpus]
22
23  BPE.find_vocab(clean_corpus[:100])
```

Figura 2: Importación de bibliotecas y uso de corpus.

3. **Aplicar el algoritmo de BPE al corpus para obtener subpalabras:**

   a) **Formar un vocabulario inicial: cada palabra se asocia a la cadena de caracteres.**

   b) **Seleccionar el número de iteraciones que mejor se adapte al corpus elegido.**

   c) **Obtener el vocabulario final: cada palabra se asocia a la cadena de subpalabras.**

   d) **Sustituir en el corpus las palabras por la tokenización en subpalabras obtenidas.**

```python
import collections

from click import pass_obj

pair_list = collections.defaultdict(int)

def BPE(corpus, k):
    global pair_list

    if(k==0):
        return

    vocab = collections.Counter(corpus)
    for i, freq in vocab.items():
        char = i.split()
        for j in range( len(char)-1 ):
            pair_list[char[j], char[j+1]] += freq

    max_ = 0
    letter = ()
    for pair, freq in pair_list.items():
        if(freq > max_):
            max_ = freq
            letter = pair
    print(letter, max_)

    for i in range(100):
        for j in range(0, len(corpus[i])-2, 2):
            if(letter[0]==corpus[i][j] and letter[1]==corpus[i][j+2]):
                corpus[i] = corpus[i][0:j+1:] + corpus[i][j+2::]

    BPE(corpus, k-1)
    for i in range(100):
        print(corpus[i])
```

Figura 3: Código generado para algoritmo BPE.

Explicación:

- Previo a aplicar nuestro algoritmo, contamos el número de veces que aparece cada palabra en nuestro Corpus.
- Declaramos una variable llamada vocab e importamos las bibliotecas necesarias [Línea 13] .
- Con la función split() metemos las letras individuales en una lista y después con un ciclo contamos el número de apariciones de los pares de letra [Línea 17].
- Posteriormente obtenemos el par que más se repite con un ciclo para poder identificar este par [Líneas 21-25].
- Realizamos hasta 100 iteraciones, para obtener así una mejor visualización del efecto del algoritmo en el corpus.
- Encontramos los pares más frecuentes y procedemos a sustituirlos en nuestras palabras formando un nuevo símbolo, para ello en el código llevamos a cabo la evaluación de una condición: si comparamos la letra frecuente con la letra de nuestra palabra en el corpus es igual, va a ponerlas en el mismo lugar. Es decir reemplazamos el par, eliminando el espacio entre ambas letras y repetimos el proceso [Línea 32].

RESULTADOS

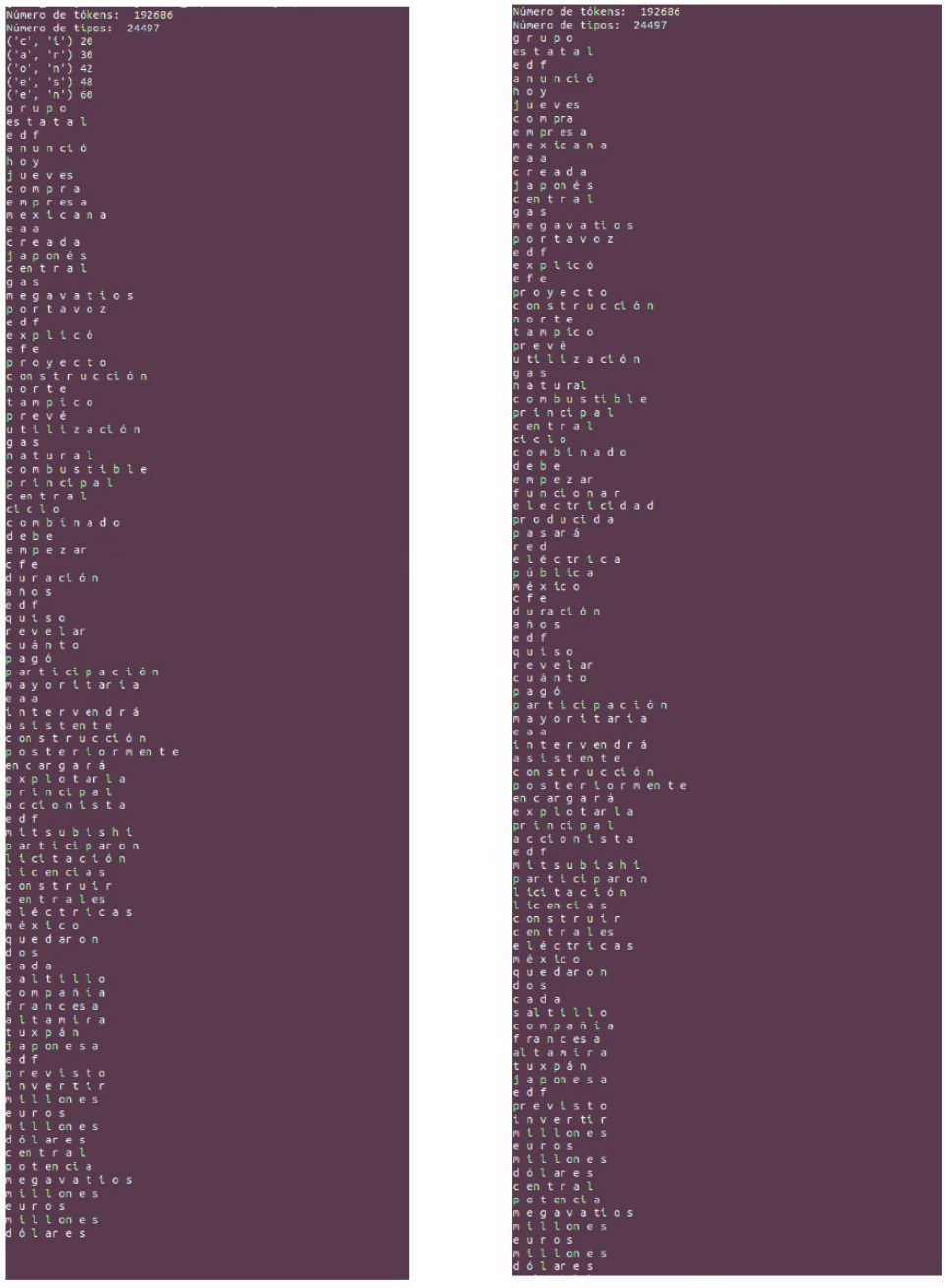

Figura 4: Resultados de aplicar el algoritmo BPE (a) con 5 iteraciones, (b) con 20 iteraciones.