3A. Práctica: Predicción de Palabras

Nombre: **Luis Fernando Izquierdo Berdugo**

Materia: **Procesamiento de Información**

Fecha: **14 de Octubre de 2024**

Instrucciones:

Entregar un notebook con el siguiente código.

1. Calcular los modelos de probabilidad MLE y MLE con suavizado de Laplace, para un modelo de bigramas. El corpus es el contenido del archivo "europarl.es"

Para el preprocesamiento de datos:

- Convertir a minúsculas
- Borrar puntuación
- Agregar marcadores de inicio y fin de oración (<s> ... <e>)
- Para facilitar el ejercicio, considerar cada línea como una oración, sin necesidad de encontrar las oraciones por cada párrafo

Nota: no remover stopwords, ya que ayudan a formar las oraciones.

2. Calcular si las siguientes oraciones son posibles, es decir, calcular las probabilidades de las siguientes oraciones, usando el modelo de MLE y MLE con suavizado de Laplace. Comparar las probabilidades.

    A. "<s> el parlamento debe enviar un mensaje <e>"
    B. "<s> el parlamento debe enviar un consejo <e>"
    C. "<s> el abismo entre pobres y ricos <e>"
    D. "<s> el abismo entre ricos y pobres <e>"
    E. "<s> el abismo de la cantera entre pobres y ricos <e>"
    F. "<s> la comisión debe ser totalmente transparente <e>"
    G. "<s> la comisión debe ser transparente <e>"

Implementación

Lo primero que se ejecutó fue la lectura y preprocesamiento del archivo. En la función `preprocess` se ejecutan los siguientes pasos:

- Se convierte la línea a minúsculas
- Con la librería `re` se elimina la puntuación
- Se usa una `f-string` para cambiar a que la línea inicie con `<s>` y termine con `<e>`
- Se regresa la línea

Posterior a esto, se hace una función lambda para procesar todas las líneas del texto.

In [8]:
```python
import re


# Paso 1: Leer el archivo
with open("europarl.es", "r", encoding="utf-8") as file:
    lines = file.readlines()

# Paso 2: Preprocesar el texto
def preprocess(line):
    line = line.lower()
    line = re.sub(r'[^\w\s]', '', line)  # Borrar puntuación
    line = f"<s> {line.strip()} <e>"
    return line

preprocessed_lines = [preprocess(line) for line in lines]
```

El siguiente paso es construir el modelo de bigramas. Se utilizó un contador para cada palabra individual (unigramas) y un diccionario de contadores para los bigramas.

A continuación se itera sobre las líneas preprocesadas y se divide en palabras cada línea, lo cual actualiza el contador de unigramas para la línea actual. Con esto se itera sobre los índices de las palabras en la línea (a excepción del último índice).

Finalmente se actualiza el contador de bigramas y `words[i]` es la primera palabra del bigrama, mientras `words[i + 1]` es la segunda palabra.

Ejemplificado con la oración `<s> Hola soy Luis <e>`,

tendríamos en el contador de unigramas `{"<s>": 1, "Hola": 1, "soy": 1, "Luis": 1, "<e>": 1}`

y en el contador de bigramas se pasarían `{"<s>": {"Hola": 1}, "Hola": {"soy": 1}, "soy": {"Luis": 1}, "Luis": {"<e>": 1}}`

Esto se irá repitiendo en cada línea del texto, lo cual construye los modelos de unigrama y bigrama necesarios para calcular las probabilidades.

In [9]:
```python
from collections import defaultdict, Counter
# Paso 3: Construir el modelo de bigramas
unigram_counts = Counter()
bigram_counts = defaultdict(Counter)

for line in preprocessed_lines:
    words = line.split()
    unigram_counts.update(words)
    for i in range(len(words) - 1):
        bigram_counts[words[i]][words[i + 1]] += 1
```

A continuación se genera el código para definir una función que calcule la probabilidad de un bigrama con el método "Maximum Likelihood Estimation" (MLE). La probabilidad MLE de un bigrama se calcula como la frecuencia del bigrama dividido por la frecuencia del unigrama que lo precede.

En la función se revisa si la primera palabra `word1` está en los bigramas, así como revisa que la segunda palabra se encuentre dentro del bigrama de la primera palabra; en caso de que si lo encuentre se divide la frecuencia del bigrama `bigram_counts[word1][word2]` entre la frecuencia del unigrama de la palabra 1 `unigram_counts[word1]`, lo cual nos da la probabilidad MLE.

Si el bigrama no existe, la función devuelve una probabilidad de 0.

In [10]:
```
# Paso 4: Calcular las probabilidades MLE
def mle_probability(bigram_counts, unigram_counts, word1, word2):
    if word1 in bigram_counts and word2 in bigram_counts[word1]:
        return bigram_counts[word1][word2] / unigram_counts[word1]
    else:
        return 0
```

Lo siguiente fue crear la función para el suavizado de Laplace. Esta es bastante similar a la probabilidad MLE, con la diferencia que al contador del bigrama se le añade 1 y al contador del unigrama se le añade el tamaño del vocabulario (que más adelante tomará valor como el conteo de unigramas).

In [11]:
```
# Paso 5: Calcular las probabilidades MLE con suavizado de Laplace
def laplace_probability(bigram_counts, unigram_counts, word1, word2, vocab_size
    return (bigram_counts[word1][word2] + 1) / (unigram_counts[word1] + vocab_s
```

Se crean las funciones para calcular las probabilidades de una oración tanto con MLE como con MLE con suavizado de Laplace. En estas funciones se divide la oración en palabras, se inicia la variable de probabilidad y se itera sobre los índices de las palabras en la oración, con esto se usan las funciones respectivas (MLE o MLE con Laplace) para obtener la probabilidad y se multiplican por la acumulada del bigrama actual, lo cual finalmente nos devolverá la probabilidad total de la oración.

In [12]:
```
# Función para calcular la probabilidad de una oración usando MLE
def sentence_probability_mle(sentence, bigram_counts, unigram_counts):
    words = sentence.split()
    prob = 1.0
    for i in range(len(words) - 1):
        prob *= mle_probability(bigram_counts, unigram_counts, words[i], words[
    return prob

# Función para calcular la probabilidad de una oración usando MLE con suavizado
def sentence_probability_laplace(sentence, bigram_counts, unigram_counts, vocab
    words = sentence.split()
    prob = 1.0
    for i in range(len(words) - 1):
        prob *= laplace_probability(bigram_counts, unigram_counts, words[i], wo
    return prob
```

Finalmente definimos las oraciones a buscar, obtenemos el tamaño del vocabulario y lanzamos las funciones para calcular las probabilidades de oraciones tanto en MLE como en MLE con suavizado de Laplace.

In [14]:
```
# Definir las oraciones
sentences = [
```

```
    "<s> el parlamento debe enviar un mensaje <e>",
    "<s> el parlamento debe enviar un consejo <e>",
    "<s> el abismo entre pobres y ricos <e>",
    "<s> el abismo entre ricos y pobres <e>",
    "<s> el abismo de la cantera entre pobres y ricos <e>",
    "<s> la comisión debe ser totalmente transparente <e>",
    "<s> la comisión debe ser transparente <e>"
]

# Calcular y comparar las probabilidades
vocab_size = len(unigram_counts)
for sentence in sentences:
    mle_prob = sentence_probability_mle(sentence, bigram_counts, unigram_counts
    laplace_prob = sentence_probability_laplace(sentence, bigram_counts, unigra
    print(f"Oración: '{sentence}'")
    print(f"Probabilidad MLE: {mle_prob}")
    print(f"Probabilidad MLE con suavizado de Laplace: {laplace_prob}\n")
```

```
Oración: '<s> el parlamento debe enviar un mensaje <e>'
Probabilidad MLE: 4.452453175934305e-13
Probabilidad MLE con suavizado de Laplace: 6.012934083279632e-21

Oración: '<s> el parlamento debe enviar un consejo <e>'
Probabilidad MLE: 3.3686200213034554e-13
Probabilidad MLE con suavizado de Laplace: 9.396583560317146e-20

Oración: '<s> el abismo entre pobres y ricos <e>'
Probabilidad MLE: 3.820757157773409e-17
Probabilidad MLE con suavizado de Laplace: 1.6797412983867322e-26

Oración: '<s> el abismo entre ricos y pobres <e>'
Probabilidad MLE: 8.677949590529031e-15
Probabilidad MLE con suavizado de Laplace: 1.1658204487612676e-24

Oración: '<s> el abismo de la cantera entre pobres y ricos <e>'
Probabilidad MLE: 0.0
Probabilidad MLE con suavizado de Laplace: 2.2193756868953378e-37

Oración: '<s> la comisión debe ser totalmente transparente <e>'
Probabilidad MLE: 3.609720553302905e-11
Probabilidad MLE con suavizado de Laplace: 7.525729057212547e-19

Oración: '<s> la comisión debe ser transparente <e>'
Probabilidad MLE: 2.560495112476194e-09
Probabilidad MLE con suavizado de Laplace: 4.141288972676337e-15
```