Preprocesamiento de corpus de N-gramas

Algunos pasos comunes de preprocesamiento para los modelos de lenguaje incluyen:

- Cambiar a minúsculas
- Eliminar caracteres especiales
- Dividir el texto en una lista de oraciones
- Dividir la oración en palabras de la lista

Datos del estudiante

1. Dany Salcca Lagar
2. 191849

In [1]:
```
import nltk
import re

nltk.download('punkt')    # sentence tokenizer
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[1]: True

Lowercase

Las palabras al comienzo de una oración y los nombres comienzan con una letra mayúscula. Sin embargo, al contar palabras, desea tratarlas de la misma manera que si aparecieran en medio de una oración.

In [2]:
```
# cambiar el corpus a minúsculas
corpus = "Learning% makes 'me' happy. I am happy be-cause I am learning! :)"
corpus = corpus.lower()

# note que la palabra "learning" será la misma independientemente de su posición en la ora
print(corpus)
```

```
learning% makes 'me' happy. i am happy be-cause i am learning! :)
```

Eliminar caracteres especiales

Es posible que sea necesario eliminar algunos de los caracteres del corpus antes de que comencemos a procesar el texto para encontrar n-gramas.

A menudo, los caracteres especiales, como las comillas dobles '"' o el guión '-', se eliminan y los puntos intermedios, como el punto '.' o el signo de interrogación '?' quedan en el corpus.

In [3]:
```
# remove special characters
corpus = "learning% makes 'me' happy. i am happy be-cause i am learning! :)"
corpus = re.sub(r"[^a-zA-Z0-9.?! ]+", "", corpus)
print(corpus)
```

```
learning makes me happy. i am happy because i am learning!
```

División de texto

Los siguientes ejemplos ilustran cómo utilizar este método. El código muestra:

- cómo dividir una cadena que contiene una fecha en una matriz de partes de fecha
- cómo dividir una cadena con el tiempo en una matriz que contiene horas, minutos y segundos

Además, tenga en cuenta lo que sucede si hay varios delimitadores consecutivos, como entre "mayo" y "9".

In [4]:
```
# dividir texto por un delimitador a array
input_date="Sat May  9 07:33:35 CEST 2020"

# obtener las partes de la fecha en un array
date_parts = input_date.split(" ")
print(f"date parts = {date_parts}")

#obtener las partes del tiempo en un array
time_parts = date_parts[4].split(":")
print(f"time parts = {time_parts}")
```

```
date parts = ['Sat', 'May', '', '9', '07:33:35', 'CEST', '2020']
time parts = ['07', '33', '35']
```

Tokenización de oraciones

Una vez que tenga una lista de oraciones, el siguiente paso es dividir cada oración en una lista de palabras.

Este proceso se puede realizar de varias formas, incluso usando el método str.split descrito anteriormente, pero usaremos la biblioteca NLTK nltk para ayudarnos con eso.

In [5]:
```
# tokenize la oración en una array de palabras

sentence = 'i am happy because i am learning.'
tokenized_sentence = nltk.word_tokenize(sentence)
print(f'{sentence} -> {tokenized_sentence}')
```

```
i am happy because i am learning. -> ['i', 'am', 'happy', 'because', 'i', 'am', 'learning',
'.']
```

Ahora que la oración está tokenizada, puede trabajar con cada palabra de la oración por separado. Esto será útil más adelante al crear y contar N-gramas. En el siguiente ejemplo de código, verá cómo encontrar la longitud de cada palabra.

In [6]:
```
# encontrar la longitud de cada palabra en la oración tokenizada
sentence = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']
word_lengths = [(word, len(word)) for word in sentence] # Create a list with the word leng
print(f' Lengths of the words: \n{word_lengths}')
```

```
 Lengths of the words:
[('i', 1), ('am', 2), ('happy', 5), ('because', 7), ('i', 1), ('am', 2), ('learning', 8),
('.', 1)]
```

N-grams

Oraciones a n-gram

El siguiente paso es construir n-gramas a partir de oraciones tokenizadas.

Una ventana deslizante de tamaño n-palabras puede generar los n-gramas. La ventana escanea la lista de palabras a partir del principio de la oración, avanzando paso a paso una palabra hasta llegar al final de la oración.

Aquí hay un método de ejemplo que imprime todos los trigramas en la oración dada.

In [7]:
```
def sentence_to_trigram(tokenized_sentence):
    for i in range(len(tokenized_sentence) - 3 + 1):
        trigram = tokenized_sentence[i : i + 3]
        print(trigram)

tokenized_sentence = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']

print(f'List all trigrams of sentence: {tokenized_sentence}\n')
sentence_to_trigram(tokenized_sentence)
```

```
List all trigrams of sentence: ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']

['i', 'am', 'happy']
['am', 'happy', 'because']
['happy', 'because', 'i']
['because', 'i', 'am']
['i', 'am', 'learning']
['am', 'learning', '.']
```

Actividad 1:

---

sentence_to_unigram()

In [8]:
```
def sentence_to_unigram(tokenized_sentence):
    for token in tokenized_sentence:
        print(token)
```

prueba

In [9]:
```
tokenized_sentence = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']

print(f'Listar todos los unigramas de la oración: {tokenized_sentence}\n')
sentence_to_unigram(tokenized_sentence)
```

```
Listar todos los unigramas de la oración: ['i', 'am', 'happy', 'because', 'i', 'am', 'learn
ing', '.']

i
am
happy
because
i
am
learning
.
```

sentence_to_bigram()

In [10]:
```
def sentence_to_bigram(tokenized_sentence):
    for i in range(len(tokenized_sentence) - 2 + 1):
        bigram = tokenized_sentence[i : i + 2]
        print(bigram)
```

prueba

In [11]:
```
tokenized_sentence = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']

print(f'Listar todos los bigramas de la oración: {tokenized_sentence}\n')
sentence_to_bigram(tokenized_sentence)
```

```
Listar todos los bigramas de la oración: ['i', 'am', 'happy', 'because', 'i', 'am', 'learni
ng', '.']

['i', 'am']
['am', 'happy']
['happy', 'because']
['because', 'i']
['i', 'am']
['am', 'learning']
['learning', '.']
```

Prefijo de un n-gram

Como viste, la probabilidad de n-gramas a menudo se calcula en función de los conteos de (n-1)-gramas. El prefijo es necesario en la fórmula para calcular la probabilidad de un n-grama.

$$P(w_n|w_1^{n-1}) = \frac{C(w_1^n)}{C(w_1^{n-1})}$$

El siguiente código muestra cómo obtener un prefijo de (n-1)-gram de n-grama en un ejemplo de obtención de trigrama de 4-gram.

In [12]:
```
# Obtiene el prefijo trigram  de un 4-gram
fourgram = ['i', 'am', 'happy','because']
trigram = fourgram[0:-1] # Obtiene los elementos desde 0, incluido, hasta el último elemen
print(trigram)
```

```
['i', 'am', 'happy']
```

Palabra de inicio y final de la oración $< s >$ y $< /s >$

Se debe agregar algunos caracteres especiales al principio y al final de cada oración:

- $< s >$ al principio
- $< /s >$ al final

Para n-grams, debemos anteponer n-1 de caracteres al comienzo de la oración.

Veamos cómo puede implementar esto en código.

In [13]:
```
# cuando trabaje con trigramas, debe anteponer 2 <s> y agregar uno </s>
n = 3
tokenized_sentence = ['i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.']
tokenized_sentence = ["<s>"] * (n - 1) + tokenized_sentence + ["</s>"]
print(tokenized_sentence)
```

```
['<s>', '<s>', 'i', 'am', 'happy', 'because', 'i', 'am', 'learning', '.', '</s>']
```

Actividad 2:

Implemente la probabilidad de n-gram (n:1, 2, 3..)

In [14]:
```
from collections import defaultdict


def calculate_ngram_probability(tokenized_sentence, n):
    ngram_counts = defaultdict(int)
    total_ngrams = 0


    # Cuenta las ocurrencias de cada n-grama
    for i in range(len(tokenized_sentence) - n + 1):
        ngram = tuple(tokenized_sentence[i : i + n])
        ngram_counts[ngram] += 1
        total_ngrams += 1


    # Calculando las probabilidades de cada n-grama
    ngram_probabilities = {}
    for ngram, count in ngram_counts.items():
        ngram_probabilities[ngram] = count / total_ngrams


    return ngram_probabilities
```

Probar con un corpus libre (ex: wikipedia)

Descargando el corpus de wikipedia

In [15]:
```
!wget https://dumps.wikimedia.org/eswiki/latest/eswiki-latest-abstract.xml.gz
```

```
--2023-06-12 05:03:34--  https://dumps.wikimedia.org/eswiki/latest/eswiki-latest-abstract.x
ml.gz
Resolving dumps.wikimedia.org (dumps.wikimedia.org)... 208.80.154.142, 2620:0:861:2:208:80:
154:142
Connecting to dumps.wikimedia.org (dumps.wikimedia.org)|208.80.154.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 244810130 (233M) [application/octet-stream]
Saving to: ‘eswiki-latest-abstract.xml.gz’

eswiki-latest-abstr 100%[===================>] 233.47M  3.63MB/s    in 67s

2023-06-12 05:04:41 (3.50 MB/s) - ‘eswiki-latest-abstract.xml.gz’ saved [244810130/24481013
0]
```

Librerias necesarias

In [16]:
```
import gzip
import xml.etree.ElementTree as ET
from collections import defaultdict
```

Procesando el corpus y almacenandolo en una lista

In [17]:
```
# Ruta del archivo descargado
file_path = "eswiki-latest-abstract.xml.gz"

# Descomprimiendo el archivo
xml_file_path = "eswiki-latest-abstract.xml"
with gzip.open(file_path, "rb") as f_in:
    with open(xml_file_path, "wb") as f_out:
        f_out.write(f_in.read())

# Procesando el archivo XML y extraer los textos
tokenized_sentences = []
tree = ET.parse(xml_file_path)
root = tree.getroot()
for child in root:
    abstract_element = child.find("abstract")
    if abstract_element is not None and abstract_element.text:
        text = abstract_element.text
        tokenized_sentence = nltk.word_tokenize(text)  # Tokenización básica (separando po
        tokenized_sentences.append(tokenized_sentence)
```

Realizando las pruebas con 5grama, 10grama y 15grama

In [18]:
```
#Corpus de wikipedia a utilizar para las pruebas
len(tokenized_sentences[1200])
```

Out[18]:
```
140
```

In [19]:
```
len(tokenized_sentences[300])
```

Out[19]:
```
98
```

In [20]:
```
len(tokenized_sentences[100])
```

Out[20]:
```
43
```

In [21]:
```
# Calcular las probabilidades de los ngramas
gram_probabilities5 = calculate_ngram_probability(tokenized_sentences[1200], 5)
gram_probabilities10 = calculate_ngram_probability(tokenized_sentences[300], 10)
gram_probabilities15 = calculate_ngram_probability(tokenized_sentences[100], 15)
```

In [22]:
```
print("n-gramas (n = 5):")
for ngram, probability in gram_probabilities5.items():
    print(f"{' '.join(ngram)}: \u001b[34m{probability}\u001b[0m")
```

n-gramas (n = 5):
Un préstamo lingüístico es una: 0.007352941176470588
préstamo lingüístico es una palabra: 0.007352941176470588
lingüístico es una palabra ,: 0.007352941176470588
es una palabra , morfema: 0.007352941176470588
una palabra , morfema o: 0.007352941176470588
palabra , morfema o expresión: 0.007352941176470588
, morfema o expresión de: 0.007352941176470588
morfema o expresión de un: 0.007352941176470588
o expresión de un idioma: 0.007352941176470588
expresión de un idioma que: 0.007352941176470588
de un idioma que es: 0.007352941176470588
un idioma que es adoptada: 0.007352941176470588
idioma que es adoptada por: 0.007352941176470588
que es adoptada por otro: 0.007352941176470588
es adoptada por otro idioma: 0.007352941176470588
adoptada por otro idioma .: 0.007352941176470588
por otro idioma . A: 0.007352941176470588
otro idioma . A menudo: 0.007352941176470588
idioma . A menudo resulta: 0.007352941176470588
. A menudo resulta de: 0.007352941176470588
A menudo resulta de la: 0.007352941176470588
menudo resulta de la influencia: 0.007352941176470588
resulta de la influencia cultural: 0.007352941176470588
de la influencia cultural de: 0.007352941176470588
la influencia cultural de los: 0.007352941176470588
influencia cultural de los hablantes: 0.007352941176470588
cultural de los hablantes del: 0.007352941176470588
de los hablantes del primer: 0.007352941176470588
los hablantes del primer idioma: 0.007352941176470588
hablantes del primer idioma sobre: 0.007352941176470588
del primer idioma sobre los: 0.007352941176470588
primer idioma sobre los del: 0.007352941176470588
idioma sobre los del segundo: 0.007352941176470588
sobre los del segundo .: 0.007352941176470588
los del segundo . Cuando: 0.007352941176470588
del segundo . Cuando el: 0.007352941176470588
segundo . Cuando el elemento: 0.007352941176470588
. Cuando el elemento prestado: 0.007352941176470588
Cuando el elemento prestado es: 0.007352941176470588
el elemento prestado es una: 0.007352941176470588
elemento prestado es una palabra: 0.007352941176470588
prestado es una palabra léxica: 0.007352941176470588
es una palabra léxica ,: 0.007352941176470588
una palabra léxica , normalmente: 0.007352941176470588
palabra léxica , normalmente un: 0.007352941176470588
léxica , normalmente un adjetivo: 0.007352941176470588
, normalmente un adjetivo ,: 0.007352941176470588
normalmente un adjetivo , un: 0.007352941176470588
un adjetivo , un nombre: 0.007352941176470588
adjetivo , un nombre o: 0.007352941176470588
, un nombre o un: 0.007352941176470588
un nombre o un verbo: 0.007352941176470588
nombre o un verbo ,: 0.007352941176470588
o un verbo , hablamos: 0.007352941176470588
un verbo , hablamos de: 0.007352941176470588
verbo , hablamos de préstamo: 0.007352941176470588
, hablamos de préstamo léxico: 0.007352941176470588
hablamos de préstamo léxico .: 0.007352941176470588

de préstamo léxico . El: 0.007352941176470588
préstamo léxico . El préstamo: 0.007352941176470588
léxico . El préstamo léxico: 0.007352941176470588
. El préstamo léxico es: 0.007352941176470588
El préstamo léxico es de: 0.007352941176470588
préstamo léxico es de lejos: 0.007352941176470588
léxico es de lejos el: 0.007352941176470588
es de lejos el más: 0.007352941176470588
de lejos el más frecuente: 0.007352941176470588
lejos el más frecuente de: 0.007352941176470588
el más frecuente de todos: 0.007352941176470588
más frecuente de todos ,: 0.007352941176470588
frecuente de todos , pero: 0.007352941176470588
de todos , pero también: 0.007352941176470588
todos , pero también existe: 0.007352941176470588
, pero también existe el: 0.007352941176470588
pero también existe el préstamo: 0.007352941176470588
también existe el préstamo gramatical: 0.007352941176470588
existe el préstamo gramatical ,: 0.007352941176470588
el préstamo gramatical , que: 0.007352941176470588
préstamo gramatical , que sucede: 0.007352941176470588
gramatical , que sucede especialmente: 0.007352941176470588
, que sucede especialmente cuando: 0.007352941176470588
que sucede especialmente cuando un: 0.007352941176470588
sucede especialmente cuando un número: 0.007352941176470588
especialmente cuando un número importante: 0.007352941176470588
cuando un número importante de: 0.007352941176470588
un número importante de hablantes: 0.007352941176470588
número importante de hablantes bilingües: 0.007352941176470588
importante de hablantes bilingües de: 0.007352941176470588
de hablantes bilingües de las: 0.007352941176470588
hablantes bilingües de las dos: 0.007352941176470588
bilingües de las dos lenguas: 0.007352941176470588
de las dos lenguas usan: 0.007352941176470588
las dos lenguas usan partículas: 0.007352941176470588
dos lenguas usan partículas ,: 0.007352941176470588
lenguas usan partículas , morfemas: 0.007352941176470588
usan partículas , morfemas y: 0.007352941176470588
partículas , morfemas y elementos: 0.007352941176470588
, morfemas y elementos no: 0.007352941176470588
morfemas y elementos no léxicos: 0.007352941176470588
y elementos no léxicos de: 0.007352941176470588
elementos no léxicos de una: 0.007352941176470588
no léxicos de una lengua: 0.007352941176470588
léxicos de una lengua cuando: 0.007352941176470588
de una lengua cuando se: 0.007352941176470588
una lengua cuando se habla: 0.007352941176470588
lengua cuando se habla la: 0.007352941176470588
cuando se habla la otra: 0.007352941176470588
se habla la otra .: 0.007352941176470588
habla la otra . Otro: 0.007352941176470588
la otra . Otro fenómeno: 0.007352941176470588
otra . Otro fenómeno similar: 0.007352941176470588
. Otro fenómeno similar es: 0.007352941176470588
Otro fenómeno similar es el: 0.007352941176470588
fenómeno similar es el calco: 0.007352941176470588
similar es el calco semántico: 0.007352941176470588
es el calco semántico ,: 0.007352941176470588
el calco semántico , en: 0.007352941176470588

```
calco semántico , en el: 0.007352941176470588
semántico , en el cual: 0.007352941176470588
, en el cual aunque: 0.007352941176470588
en el cual aunque no: 0.007352941176470588
el cual aunque no se: 0.007352941176470588
cual aunque no se toma: 0.007352941176470588
aunque no se toma una: 0.007352941176470588
no se toma una expresión: 0.007352941176470588
se toma una expresión literal: 0.007352941176470588
toma una expresión literal de: 0.007352941176470588
una expresión literal de otro: 0.007352941176470588
expresión literal de otro idioma: 0.007352941176470588
literal de otro idioma ,: 0.007352941176470588
de otro idioma , sí: 0.007352941176470588
otro idioma , sí se: 0.007352941176470588
idioma , sí se toma: 0.007352941176470588
, sí se toma su: 0.007352941176470588
sí se toma su significado: 0.007352941176470588
se toma su significado .: 0.007352941176470588
```

In [23]:

```
print("n-gramas (n = 10):")
for ngram, probability in gram_probabilities10.items():
    print(f"{' '.join(ngram)}: \u001b[34m{probability}\u001b[0m")
```

n-gramas (n = 10):
Las comelináceas (nombre científico Commelinaceae) forman una familia: 0.011235955056179775
comelináceas (nombre científico Commelinaceae) forman una familia de: 0.01123595505617975
(nombre científico Commelinaceae) forman una familia de plantas: 0.011235955056179775
nombre científico Commelinaceae) forman una familia de plantas monocotiledóneas: 0.011235955056179775
científico Commelinaceae) forman una familia de plantas monocotiledóneas representadas: 0.011235955056179775
Commelinaceae) forman una familia de plantas monocotiledóneas representadas por: 0.011235955056179775
) forman una familia de plantas monocotiledóneas representadas por hierbas: 0.011235955056179775
forman una familia de plantas monocotiledóneas representadas por hierbas carnosas: 0.011235955056179775
una familia de plantas monocotiledóneas representadas por hierbas carnosas ,: 0.011235955056179775
familia de plantas monocotiledóneas representadas por hierbas carnosas , a: 0.011235955056179775
de plantas monocotiledóneas representadas por hierbas carnosas , a veces: 0.011235955056179775
plantas monocotiledóneas representadas por hierbas carnosas , a veces suculentas: 0.011235955056179775
monocotiledóneas representadas por hierbas carnosas , a veces suculentas ,: 0.011235955056179775
representadas por hierbas carnosas , a veces suculentas , con: 0.011235955056179775
por hierbas carnosas , a veces suculentas , con hojas: 0.011235955056179775
hierbas carnosas , a veces suculentas , con hojas planas: 0.011235955056179775
carnosas , a veces suculentas , con hojas planas o: 0.011235955056179775
, a veces suculentas , con hojas planas o con: 0.011235955056179775
a veces suculentas , con hojas planas o con forma: 0.011235955056179775
veces suculentas , con hojas planas o con forma de: 0.011235955056179775
suculentas , con hojas planas o con forma de V: 0.011235955056179775
, con hojas planas o con forma de V en: 0.011235955056179775
con hojas planas o con forma de V en el: 0.011235955056179775
hojas planas o con forma de V en el corte: 0.011235955056179775
planas o con forma de V en el corte transversal: 0.011235955056179775
o con forma de V en el corte transversal ,: 0.011235955056179775
con forma de V en el corte transversal , en: 0.011235955056179775
forma de V en el corte transversal , en la: 0.011235955056179775
de V en el corte transversal , en la base: 0.011235955056179775
V en el corte transversal , en la base de: 0.011235955056179775
en el corte transversal , en la base de las: 0.011235955056179775
el corte transversal , en la base de las hojas: 0.011235955056179775
corte transversal , en la base de las hojas con: 0.011235955056179775
transversal , en la base de las hojas con una: 0.011235955056179775
, en la base de las hojas con una vaina: 0.011235955056179775
en la base de las hojas con una vaina cerrada: 0.011235955056179775
la base de las hojas con una vaina cerrada .: 0.011235955056179775
base de las hojas con una vaina cerrada . La: 0.011235955056179775
de las hojas con una vaina cerrada . La flor: 0.011235955056179775
las hojas con una vaina cerrada . La flor posee: 0.011235955056179775
hojas con una vaina cerrada . La flor posee un: 0.011235955056179775
con una vaina cerrada . La flor posee un perianto: 0.011235955056179775
una vaina cerrada . La flor posee un perianto dividido: 0.011235955056179775
vaina cerrada . La flor posee un perianto dividido en: 0.011235955056179775
cerrada . La flor posee un perianto dividido en 3: 0.011235955056179775
. La flor posee un perianto dividido en 3 sépalos: 0.011235955056179775

```
La flor posee un perianto dividido en 3 sépalos y: 0.011235955056179775
flor posee un perianto dividido en 3 sépalos y 3: 0.011235955056179775
posee un perianto dividido en 3 sépalos y 3 pétalos: 0.011235955056179775
un perianto dividido en 3 sépalos y 3 pétalos (: 0.011235955056179775
perianto dividido en 3 sépalos y 3 pétalos ( aunque: 0.011235955056179775
dividido en 3 sépalos y 3 pétalos ( aunque a: 0.011235955056179775
en 3 sépalos y 3 pétalos ( aunque a veces: 0.011235955056179775
3 sépalos y 3 pétalos ( aunque a veces el: 0.011235955056179775
sépalos y 3 pétalos ( aunque a veces el tercer: 0.011235955056179775
y 3 pétalos ( aunque a veces el tercer pétalo: 0.011235955056179775
3 pétalos ( aunque a veces el tercer pétalo ,: 0.011235955056179775
pétalos ( aunque a veces el tercer pétalo , de: 0.011235955056179775
( aunque a veces el tercer pétalo , de posición: 0.011235955056179775
aunque a veces el tercer pétalo , de posición abaxial: 0.011235955056179775
a veces el tercer pétalo , de posición abaxial ,: 0.011235955056179775
veces el tercer pétalo , de posición abaxial , es: 0.011235955056179775
el tercer pétalo , de posición abaxial , es de: 0.011235955056179775
tercer pétalo , de posición abaxial , es de otro: 0.011235955056179775
pétalo , de posición abaxial , es de otro color: 0.011235955056179775
, de posición abaxial , es de otro color y: 0.011235955056179775
de posición abaxial , es de otro color y pequeño: 0.011235955056179775
posición abaxial , es de otro color y pequeño e: 0.011235955056179775
abaxial , es de otro color y pequeño e inconspicuo: 0.011235955056179775
, es de otro color y pequeño e inconspicuo ,: 0.011235955056179775
es de otro color y pequeño e inconspicuo , pareciendo: 0.011235955056179775
de otro color y pequeño e inconspicuo , pareciendo que: 0.011235955056179775
otro color y pequeño e inconspicuo , pareciendo que hay: 0.011235955056179775
color y pequeño e inconspicuo , pareciendo que hay solo: 0.011235955056179775
y pequeño e inconspicuo , pareciendo que hay solo dos: 0.011235955056179775
pequeño e inconspicuo , pareciendo que hay solo dos pétalos: 0.011235955056179775
e inconspicuo , pareciendo que hay solo dos pétalos ): 0.011235955056179775
inconspicuo , pareciendo que hay solo dos pétalos ) ;: 0.011235955056179775
, pareciendo que hay solo dos pétalos ) ; sus: 0.011235955056179775
pareciendo que hay solo dos pétalos ) ; sus estambres: 0.011235955056179775
que hay solo dos pétalos ) ; sus estambres se: 0.011235955056179775
hay solo dos pétalos ) ; sus estambres se caracterizan: 0.011235955056179775
solo dos pétalos ) ; sus estambres se caracterizan por: 0.011235955056179775
dos pétalos ) ; sus estambres se caracterizan por poseer: 0.011235955056179775
pétalos ) ; sus estambres se caracterizan por poseer pelos: 0.011235955056179775
) ; sus estambres se caracterizan por poseer pelos en: 0.011235955056179775
; sus estambres se caracterizan por poseer pelos en sus: 0.011235955056179775
sus estambres se caracterizan por poseer pelos en sus filamentos: 0.011235955056179775
estambres se caracterizan por poseer pelos en sus filamentos .: 0.011235955056179775
```

In [24]:
```
print("n-gramas (n = 15):")
for ngram, probability in gram_probabilities15.items():
    print(f"{' '.join(ngram)}: \u001b[34m{probability}\u001b[0m")
```

n-gramas (n = 15):
El alfabeto o abecedario de una lengua o idioma es el conjunto ordenado de sus: 0.034482758620689655
alfabeto o abecedario de una lengua o idioma es el conjunto ordenado de sus letras: 0.034482758620689655
o abecedario de una lengua o idioma es el conjunto ordenado de sus letras .: 0.034482758620689655
abecedario de una lengua o idioma es el conjunto ordenado de sus letras . Es: 0.034482758620689655
de una lengua o idioma es el conjunto ordenado de sus letras . Es también: 0.034482758620689655
una lengua o idioma es el conjunto ordenado de sus letras . Es también la: 0.034482758620689655
lengua o idioma es el conjunto ordenado de sus letras . Es también la agrupación: 0.034482758620689655
o idioma es el conjunto ordenado de sus letras . Es también la agrupación que: 0.034482758620689655
idioma es el conjunto ordenado de sus letras . Es también la agrupación que se: 0.034482758620689655
es el conjunto ordenado de sus letras . Es también la agrupación que se lee: 0.034482758620689655
el conjunto ordenado de sus letras . Es también la agrupación que se lee con: 0.034482758620689655
conjunto ordenado de sus letras . Es también la agrupación que se lee con un: 0.034482758620689655
ordenado de sus letras . Es también la agrupación que se lee con un orden: 0.034482758620689655
de sus letras . Es también la agrupación que se lee con un orden determinado: 0.034482758620689655
sus letras . Es también la agrupación que se lee con un orden determinado de: 0.034482758620689655
letras . Es también la agrupación que se lee con un orden determinado de las: 0.034482758620689655
. Es también la agrupación que se lee con un orden determinado de las grafías: 0.034482758620689655
Es también la agrupación que se lee con un orden determinado de las grafías utilizadas: 0.034482758620689655
también la agrupación que se lee con un orden determinado de las grafías utilizadas para: 0.034482758620689655
la agrupación que se lee con un orden determinado de las grafías utilizadas para representar: 0.034482758620689655
agrupación que se lee con un orden determinado de las grafías utilizadas para representar el: 0.034482758620689655
que se lee con un orden determinado de las grafías utilizadas para representar el lenguaje: 0.034482758620689655
se lee con un orden determinado de las grafías utilizadas para representar el lenguaje que: 0.034482758620689655
lee con un orden determinado de las grafías utilizadas para representar el lenguaje que sirve: 0.034482758620689655
con un orden determinado de las grafías utilizadas para representar el lenguaje que sirve de: 0.034482758620689655
un orden determinado de las grafías utilizadas para representar el lenguaje que sirve de sistema: 0.034482758620689655
orden determinado de las grafías utilizadas para representar el lenguaje que sirve de sistema de: 0.034482758620689655
determinado de las grafías utilizadas para representar el lenguaje que sirve de sistema de comunicación: 0.034482758620689655
de las grafías utilizadas para representar el lenguaje que sirve de sistema de comunicación .: 0.034482758620689655

Conclusiones

Conclusión 1

Los n-gramas capturan relaciones locales entre palabras en un texto, pero pierden información contextual más amplia. Esto significa que los n-gramas pueden no ser suficientes para capturar la estructura y el significado global de un texto.

Conclusión 2

Los n-gramas son sensibles a errores ortográficos, ruido en los datos y variaciones gramaticales. Pequeños cambios en un n-grama pueden generar un nuevo n-grama que se trata como completamente distinto, lo que puede afectar la precisión y coherencia de los modelos basados en n-gramas.

Conclusión 3

Los n-gramas especialmente los trigramas y más altos, sufren la falta de datos. A medida que aumenta el valor de n, es más probable que los n-gramas sean menos frecuentes en el corpus, lo que dificulta estimar con precisión sus probabilidades. Esto puede llevar a una mayor incertidumbre en las predicciones basadas en n-gramas menos comunes.