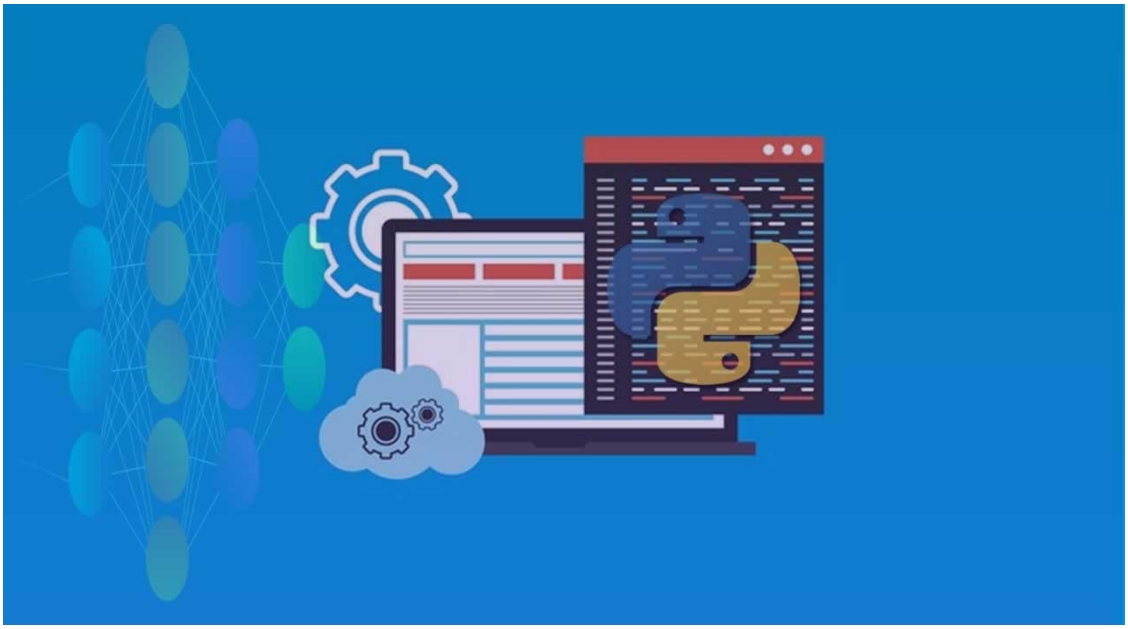
Modelos Probabilísticos en PLN

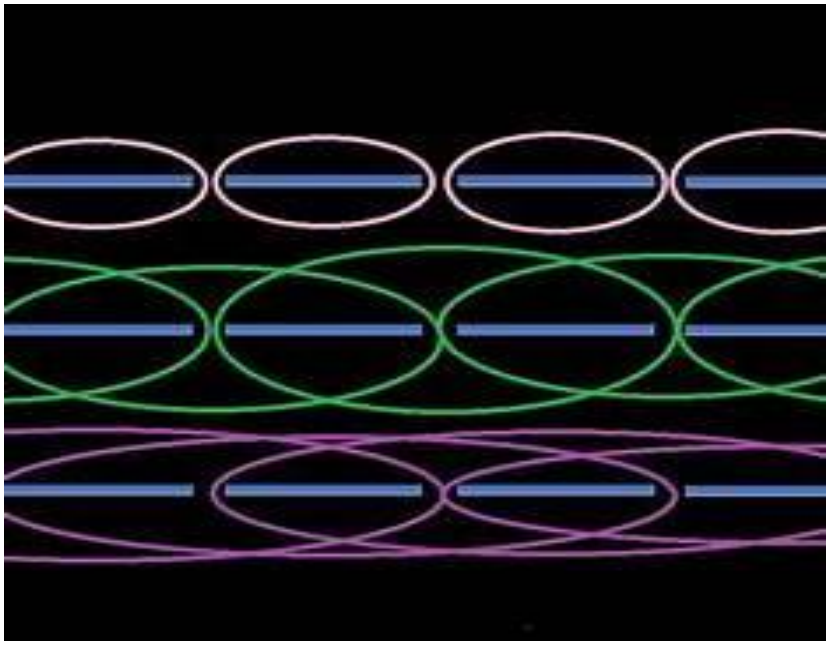
Análisis de N-Gramas en CorpusEducacion.txt



Alumno **= *Nicolas Mesquiatti***

Materia **= *Técnicas de procesamiento del habla***

Unigrama – bigrama -trigrama



Ejercicio

-Modelos de N-gramas – Práctica

Dado el corpus del archivo CorpusEducacion.txt, obtener en gráfico de barras la comparación entre 2-gramas

y 3-gramas luego de preparar el texto con tokens, lemas y stop_words.

Organizar el código con funciones.

Definir la cantidad de apariciones de las palabras en 2 (min_df = 2).

el archivo CorpusEducacion.txt es un resumen de las opiniones reales de alumnos de Colombia que expresan sus deseos con respecto a la educación superior en el año 2025

INTRODUCCION

Este informe presenta el análisis de **bi-gramas y tri-gramas** generados a partir del corpus `CorpusEducacion.txt`. Para identificar patrones de lenguaje y términos clave, se utilizó procesamiento de lenguaje natural (NLP) con técnicas de n-gramas. Además, modularizado con un archivo aparte donde se encuentran todas las funciones

Carga del corpus

```
#Cargo el corpus
with open("D:/Tecnicas De Procesamiento del habla/N-gramas/CorpusEducacion.txt", "r", encoding="latin-1") as archivo:
    corpus = archivo.read()
```

1)

- Procesar y limpiar el texto mediante **tokenización, lematización y eliminación de stopwords**.

En el que utilizamos las siguientes librerías:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from collections import Counter
```

En la siguiente imagen podemos ver como quedaron definidas las funciones para realizar cada tarea de tokenizar, lematizar y eliminar stopwords

Funciones.py

```python
#Primero Tokenizamos
def tokenizar(texto):
    # Tokenizamos el texto en palabras
    tokens = word_tokenize(texto)
    return tokens

# Segundo quitamos las stopwords
def quitarStopwords_eng(texto):
    ingles = stopwords.words("spanish")  # Cambié a español para el corpus de educación
    texto_limpio = [
        w.lower() for w in texto
        if w.lower() not in ingles
        and w not in string.punctuation
        and not any(c in w for c in ["|", "--", "''", "``", "()", "_", "-","...","2025"
    ]
    return texto_limpio

# Tercero lematizamos
def get_wordnet_pos(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)
def lematizar(texto):
    texto_lema = [lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in texto]
    return texto_lema
```

Main.py (***Ademas de realizar esas tareas tambien en el main pase todo a minúscula para poder limpiarlo completo)***

```python
tokens = tokenizar(corpus)
print("Tokens:", tokens)

#Convertir los tokens a minúsculas
tokens_lower = [t.lower() for t in tokens]
#print("2. Minúsculas:", tokens_lower)

#Elimino la puntuación como ".",",", "!", etc.
tokens_sin_punt = [t for t in tokens_lower if t not in string.punctuation]
#print("3. Sin puntuación:", tokens_sin_punt)

#Texto sin stopwords
texto_limpio = quitarStopwords_eng(tokens)
#print("Texto sin stopwords:", texto_limpio)

#Lematizar el texto
texto_lema = lematizar(texto_limpio)
#print("Texto lematizado:", texto_lema)
```

2)

- Extraer **bi-gramas (pares de palabras) y tri-gramas (tres palabras seguidas)** con `CountVectorizer`.

En el que utilizamos las siguientes librerías:

```
from sklearn.feature_extraction.text import CountVectorizer
```

En funciones.py realizamos una función para obtener n-gramas donde el rango son (2, 3) bigramas y trigramas

Funciones.py

```
def obtener_ngramas(oraciones_limpias):
    oraciones_limpias = oraciones_limpias.split(".")  # Divide en oraciones
    vectorizer = CountVectorizer(ngram_range=(2, 3), min_df=1, max_df=0.9)
    X = vectorizer.fit_transform(oraciones_limpias)
    vocab = vectorizer.get_feature_names_out()
    counts = X.toarray().sum(axis=0)
    freqs = Counter(dict(zip(vocab, counts)))
    #print(list(freqs.keys()))
    print("\nCantidad de N-Gramas (bi-gramas y tri-gramas con al menos 2 repeticiones):"
    return freqs
```

Main.py (Antes de llamar a la función, convertí el texto a cadena para que la función lo divida en oraciones)

```
texto_procesado = " ".join(texto_lema)
print("Texto procesado:", texto_procesado)
#Llamo a la función para obtener n-gramas
ngramas = obtener_ngramas(texto_procesado)
```

3)

- Visualizar los resultados mediante **gráficos de barras** para comparar la frecuencia de n-gramas.

En el que utilizamos las siguientes librerías:

```
import matplotlib.pyplot as plt
```

Realice dos gráficos: uno de barras verticales que hace un top 10 n-gramas, el otro de barras horizontales que hace un top 20 ordenado de mayor a menor más fácil de leer

Funciones.py

```python
def graficar_ngramas(freqs):
    # Obtener los 10 n-gramas más frecuentes
    top_10_ngrams = dict(freqs.most_common(10))

    # Crear el gráfico
    plt.figure(figsize=(12, 6))
    plt.bar(top_10_ngrams.keys(), top_10_ngrams.values(), color='skyblue')

    plt.xticks(rotation=45, ha="right", fontsize=10)
    plt.title("Top 10 N-Gramas más frecuentes")
    plt.xlabel("N-Gramas")
    plt.ylabel("Frecuencia")

    # Mostrar el gráfico
    plt.tight_layout()
    plt.show()


def graficar_comparacion_ngrams(freqs, top_n=20):
    top = freqs.most_common(top_n)
    ngrams, values2 = zip(*top) if top else ([], [])
    np.arange(len(ngrams))
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.barh(ngrams[::-1], values2[::-1], color='skyblue')
    plt.title("Top 2-gramas y 3-gramas")
    plt.xlabel("Frecuencia")
    plt.tight_layout()
    plt.show()
```

Main.py

```python
graficar_ngramas(ngramas)
graficar_comparacion_ngrams(ngramas)
```

Resultado

Gráfico 1

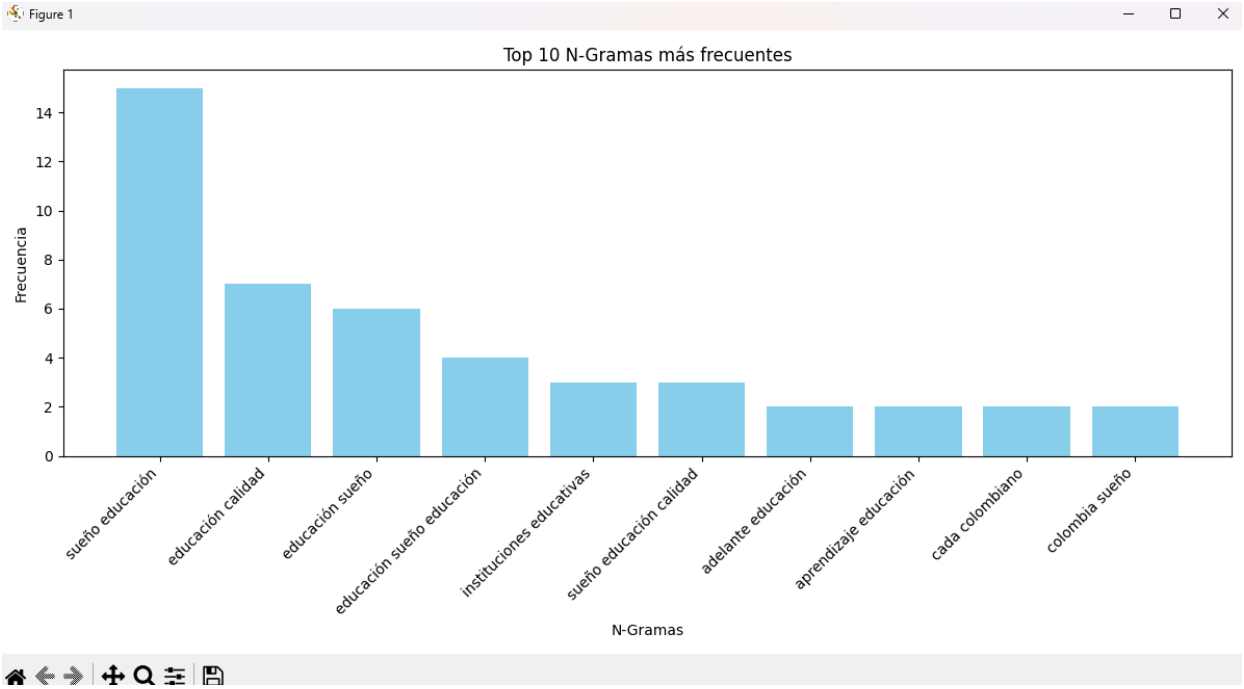
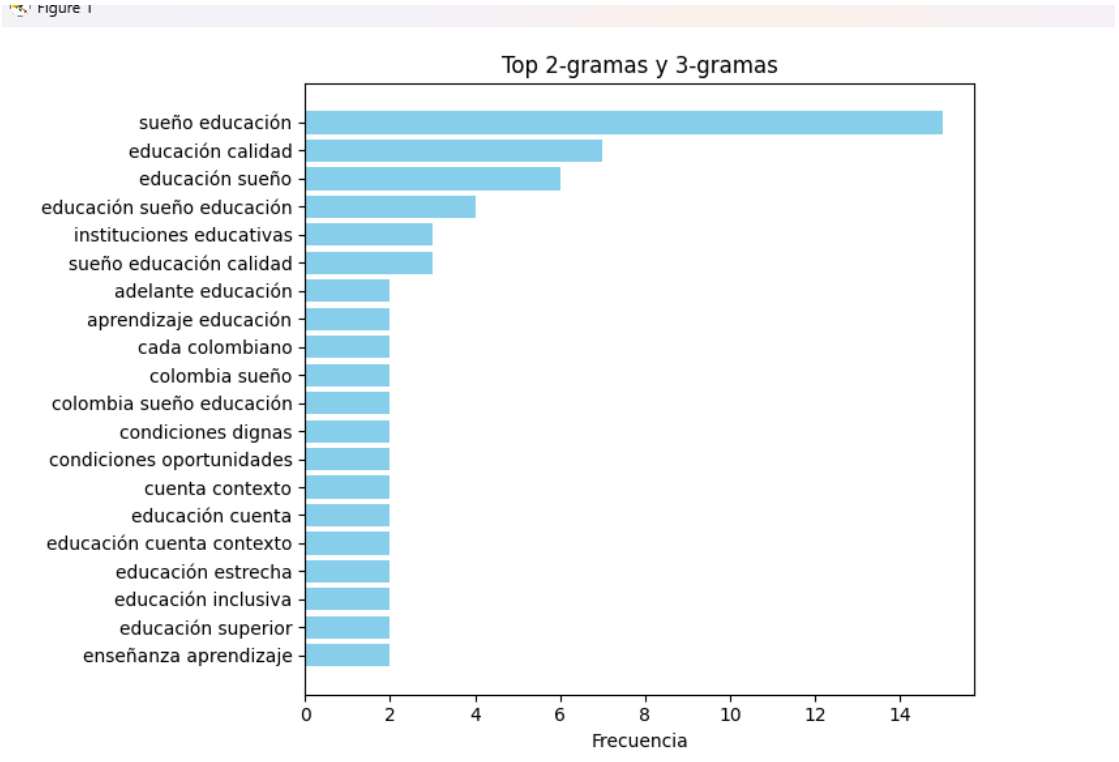


Gráfico 2



Aclaración: **Ademas de los dos gráficos, en el main printie el texto en tokens y procesado en cadena ya todo limpio**

Conclusiones

- **Los términos más recurrentes indican un fuerte interés en la mejora educativa**, reflejando preocupaciones sobre **calidad, oportunidades y acceso**.
- **Los bi-gramas más repetidos** incluyen frases como "sueño educación" y "educación calidad", lo que sugiere una visión aspiracional de la enseñanza.
- **Los tri-gramas destacan frases como "igualdad condiciones oportunidades"**, indicando un foco en la equidad y el acceso a la educación.
- **El uso de n-gramas nos ayuda a entender patrones en los datos textuales** y revelar cuáles son los temas más mencionados por los estudiantes.