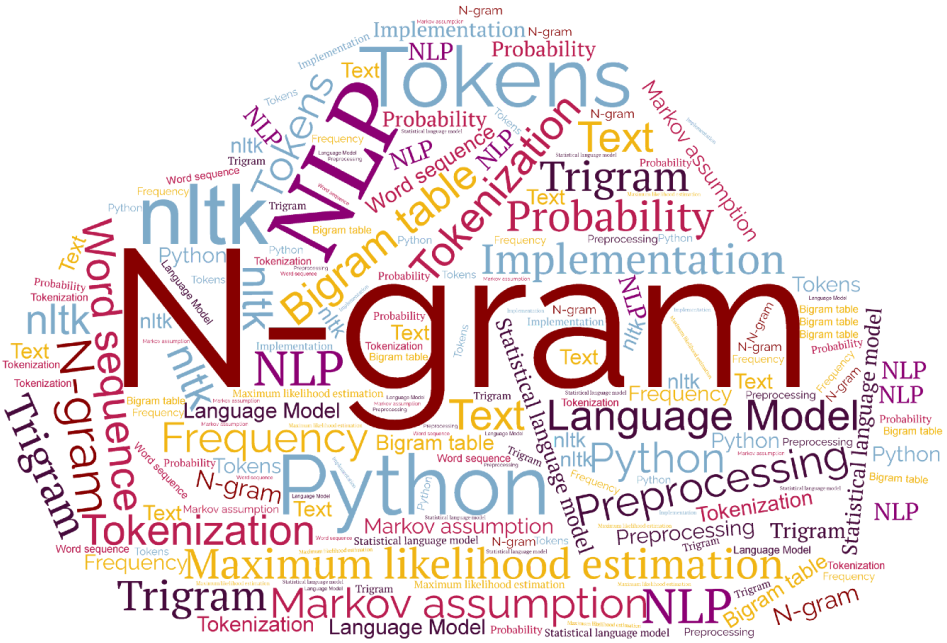
INSTITUTO TECNOLÓGICO BELTRÁN
Centro de Tecnología e Innovación

Informe Práctico

Trabajo Práctico N.º 5

Modelos Probabilísticos

N-Grama



Nombre: Coral Tolazzi
Tema: MODELOS Y CLASIFICACIÓN
Profesora: Yanina Ximena Scudero
Cuatrimestre y Año: 1 Cuatrimestre del 2025

Instituto tecnológico Beltrán
Procesamiento del Lenguaje Natural

Ejercicio:

Dado el corpus del archivo CorpusEducacion.txt:

- Obtener en gráfico de barras la comparación entre 2-gramas y 3-gramas
- 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.

Objetivo del Código

El objetivo principal es:

- Leer un corpus de texto.
- Limpiarlo eliminando palabras irrelevantes (stopwords, signos de puntuación, etc.).
- Lematizarlo para reducir las palabras a su forma base.
- Generar y contar **bi-gramas** y **tri-gramas** frecuentes.
- Visualizar gráficamente los n-gramas más comunes.

Descripción del Código

1. Importación de librerías

Se usan librerías de procesamiento de lenguaje natural (NLTK), visualización (matplotlib), conteo (Counter), y extracción de características (CountVectorizer).

2. Función run()

Función principal que coordina el proceso completo:

- Lee el corpus desde archivo.
- Limpia y lematiza las oraciones.
- Extrae los n-gramas (2-gramas y 3-gramas).
- Muestra una gráfica con los más frecuentes.

3. Lectura del corpus: obtener_corpus()

Lee el archivo corpus.txt y devuelve una lista de oraciones (líneas sin espacios en blanco).

4. Mostrar oraciones: mostrar_corpus()

Imprime en consola el contenido del corpus o corpus limpio.

5. Limpieza del corpus: limpiar_corpus()

Procesa cada oración:

- Tokeniza las palabras.
- Elimina **stopwords** en español, signos de puntuación y números.

- Lematiza las palabras.

6. Eliminación de palabras vacías: sacar_stopwords_esp()

Filtra palabras que:

- Están en la lista de *stopwords* en español (como “el”, “de”, “la”).
- Son signos de puntuación o caracteres especiales.
- Son números.

7. Lematización: lematizar() y get_wordnet_pos()

Reduce las palabras a su forma base (por ejemplo: “estudiantes” → “estudiante”), usando la clase WordNetLemmatizer.

get_wordnet_pos() determina el tipo de palabra (sustantivo, verbo, adjetivo o adverbio) para mejorar la lematización.

8. Extracción de n-gramas: n_grama()

- Usa CountVectorizer para obtener todos los **bi-gramas** (2 palabras) y **tri-gramas** (3 palabras) con al menos 2 repeticiones (min_df=2).
- Muestra los n-gramas generados y la cantidad.
- Devuelve un diccionario con la frecuencia de cada n-grama.

9. Gráfico de los n-gramas: graficar_comparacion_ngrams()

- Muestra un gráfico de barras horizontal con los **top 10 n-gramas** más frecuentes.
- Usa matplotlib para generar la visualización.

Requisitos:

Antes de ejecutar este código, es necesario tener instalados los siguientes recursos de NLTK:

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

Conclusiones:

Este script ofrece una solución práctica para el análisis de lenguaje natural, enfocándose en la detección de n-gramas frecuentes dentro de un corpus textual. A través de un proceso que incluye limpieza del texto, eliminación de palabras vacías, lematización y extracción de bi-gramas y tri-gramas, se facilita la identificación de patrones lingüísticos recurrentes.

Su aplicación resulta útil en tareas como el análisis de opiniones, estudios de contenido, procesamiento previo para modelos de aprendizaje automático, y exploración de tendencias discursivas en textos. En conjunto, la herramienta permite obtener una representación más estructurada y significativa del lenguaje presente en los datos analizados.