Introducción a la Ciencia de Datos

Maestría en Ciencia de Datos y Aprendizaje Automático

Facultad de Ingeniería, UdelaR

02/07/2024

Tarea 2

Grupo: Joana Auriello, Pablo Molina

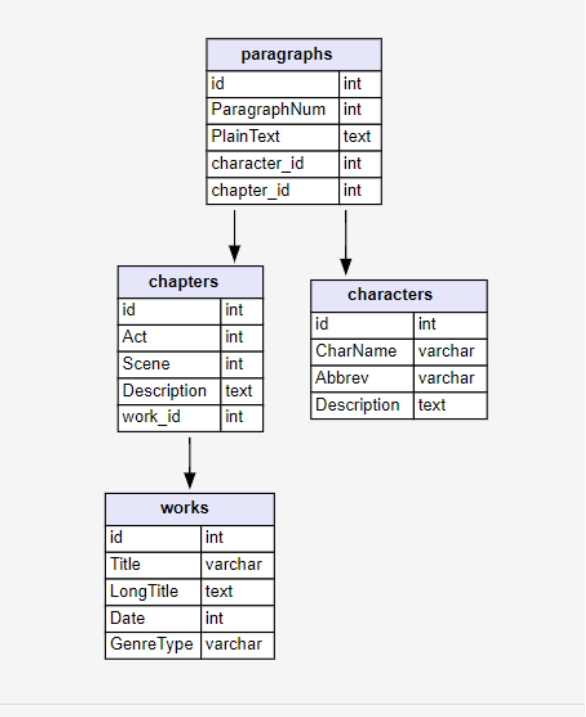
ÍNDICE

1. Introducción y EDA

El siguiente informe es la continuación del trabajo realizado anteriormente, en los siguientes capítulos brindaremos un resumen del trabajo realizado y los resultados obtenidos, para mayor detalle sobre el modelo entidad-relación, la calidad de los datos o las tendencias encontradas en la cantidad de palabras por obra, año o personaje, remitirse al siguiente link.

Modelo entidad-relación

Ambos trabajos fueron realizados usando la base de las obras de Shakespeare, esta está diseñada para almacenar información sobre las obras, capítulos, personajes y párrafos de los textos.



Todas las tablas cuentan con un indicador único de línea como clave primaría. Las relaciones entre tablas estarán determinadas por cómo estas claves primarias de las tablas forman parte de otras tablas como clave foránea.

2. Limpieza de Datos

En resumen, las tablas incluidas en la base de datos presentan una alta calidad de datos con buena completitud, unicidad y consistencia. Con el fin de analizar el contenido de las obras, se realizaron transformaciones a los párrafos con el fin de obtener las palabras que lo componen. A este nuevo campo de palabras se le realizaron las siguientes transformaciones con el fin de limpiar los datos: pasar todas las palabras a minúsculas, eliminar signos de puntuación y expandir las abreviaciones, todo esto se realizó con el fin de obtener la verdadera frecuencia de las palabras.

3. Análisis previo

El análisis previo se basó en identificar tendencias en la distribución de la cantidad de palabras y los géneros de las obras según diversos factores. De este análisis obtenemos resultados como la concentración de las obras trágicas en un breve periodo de años

Los principales resultados son:

- El autor tuvo 13 años de actividad, siendo 1603 el único año donde no publicó ninguna obra
- Las tragedias se concentran principalmente entre 1600 y 1608
- Una vez retirados de la consideración poet y stage director, los tres personajes que tienen mayor parlamento son: Henry V, Falstafft y Hamlet

4. Reducción del dataset

En esta sección, se describe el proceso de reducción del dataset para enfocarnos en tres personajes específicos de las obras de Shakespeare: Antony, Cleopatra y Queen Margaret.

Primero, se procedió a integrar varios conjuntos de datos que contenían información sobre los párrafos, capítulos, obras y personajes. Esto permitió tener un dataset consolidado que incluye el texto limpio de los párrafos, el nombre de los personajes, los títulos de las obras y los géneros literarios.

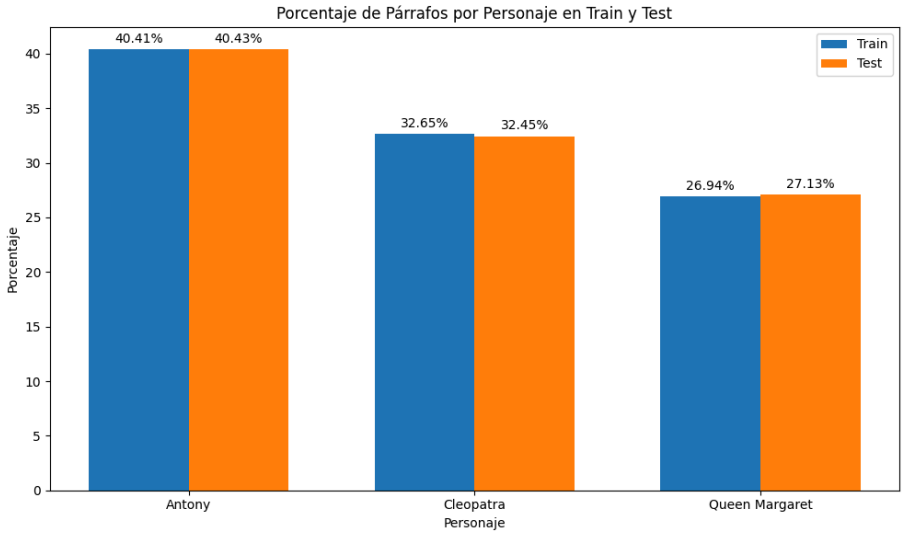
Para centrarnos en los personajes mencionados, se filtró el dataset original, excluyendo cualquier párrafo que no fuera hablado por Antony, Cleopatra o Queen Margaret.

Una vez reducido el dataset, se realizó un recuento de los párrafos correspondientes a cada uno de estos personajes. Los resultados mostraron que Antony tenía 253 párrafos, Cleopatra 204 párrafos y Queen Margaret 169 párrafos. Esta distribución evidencia una variación en la cantidad de texto disponible para cada personaje, lo cual se consideró en los análisis posteriores.

El siguiente paso fue dividir el dataset en conjuntos de entrenamiento y prueba. Para ello, se utilizó un enfoque de muestreo estratificado, asegurando que el 30% de los datos se destinara al conjunto de prueba y que las proporciones de los personajes se mantuvieran consistentes en ambos conjuntos. Este método también implicó fijar un estado aleatorio para garantizar la reproducibilidad de los resultados.

Al final de este proceso, los conjuntos de datos quedaron conformados de la siguiente manera: 438 instancias en el conjunto de entrenamiento y 188 instancias en el conjunto de prueba.

Se genera la siguiente visualización para observar que el balance de párrafos de cada personaje es similar en train y test:



Como se evidencia, hay un balance en train y test de la cantidad de párrafos para los tres personajes seleccionados.

5. Técnicas de Procesamiento de Lenguaje Natural

5.1 Bag of Words

¿Cómo funciona la técnica Bag of Words?

La técnica Bag of Words convierte el texto en una representación numérica. Se construye un vocabulario de palabras conocidas de todo el corpus (conjunto de documentos) y luego se cuenta cuántas veces aparece cada palabra en cada documento.

1. **Tokenización**: El primer paso consiste en dividir el texto en palabras individuales, conocidas como tokens. La tokenización separa el texto en unidades básicas de significado, que en este caso son palabras, eliminando cualquier puntuación y convirtiendo todas las palabras a minúsculas para asegurar uniformidad.

2. **Construcción del vocabulario**: Se crea un diccionario (o vocabulario) que contiene todas las palabras únicas presentes en el corpus (conjunto de documentos). Cada palabra del vocabulario se asocia con un índice único. Este paso es crucial, ya que define el conjunto de términos que se utilizarán para representar los documentos.

3. **Vectorización**: Cada documento del corpus se convierte en un vector. En este vector, cada dimensión corresponde a una palabra del vocabulario y el valor en cada posición es el conteo de cuántas veces aparece esa palabra en el documento. Este vector puede interpretarse como una lista de frecuencias de palabras.

Sparse Matrix

La matriz resultante de aplicar la técnica Bag of Words a un corpus es conocida como una matriz dispersa (sparse matrix). Esta matriz tiene una fila por cada documento y una columna por cada palabra del vocabulario. La mayoría de las entradas de esta matriz serán ceros porque en textos largos y vocabularios grandes, no todas las palabras aparecen en cada documento.

El uso de matrices dispersas es crucial para manejar eficientemente el espacio y el tiempo de computación. Las matrices dispersas permiten almacenar y operar grandes conjuntos de datos de manera eficiente, utilizando estructuras de datos especializadas que solo almacenan las posiciones y valores de los elementos distintos de cero. Sin estas matrices, necesitaríamos una cantidad significativa de memoria RAM para almacenar grandes matrices llenas de ceros, lo que no sería práctico ni eficiente.

Ejemplo de Bag of Words

Supongamos que tenemos los siguientes textos:

- Texto 1: "El gato se sienta en la ventana"
- Texto 2: "El perro ladra en la casa"
- Texto 3: "El gato y el perro son amigos"

Luego de aplicar CountVectorizer, el resultado esperado de la matriz debería mostrar el vocabulario extraído y una matriz de conteos representada como un DataFrame.

El vocabulario construido a partir de los textos es el siguiente:

Índice	Palabra
0	amigos
1	casa
2	el
3	en
4	gato
5	ladra
6	la
7	perro
8	se
9	sienta
10	son
11	ventana
12	y

La matriz resultante de conteos se representa a continuación. Cada fila del DataFrame corresponde a un texto, y cada columna a una palabra del vocabulario. Los valores indican el número de veces que aparece cada palabra en cada texto.

	amigos	casa	el	en	gato	ladra	la	perro	se	sienta	son	ventana	y
Texto 1	0	0	1	1	1	0	1	0	1	1	0	1	0
Texto 2	0	1	1	1	0	1	1	1	0	0	0	0	0
Texto 3	1	0	2	0	1	0	0	1	0	0	1	0	1

Explicación de la Matriz Resultante

Cada valor en la matriz indica el número de veces que una palabra específica del vocabulario aparece en un texto determinado. Por ejemplo, en el Texto 1, la palabra "gato" aparece una vez, mientras que la palabra "ventana" también aparece una vez. En el Texto 3, la palabra "el" aparece dos veces, lo que refleja su frecuencia más alta en ese texto en particular.

La matriz tiene dimensiones 3×13, lo que significa que hay 3 textos y el vocabulario contiene 13 palabras únicas. La mayoría de los valores en la matriz son ceros, ya que no todas las palabras aparecen en cada texto. Esto convierte a la matriz en una Sparse Matrix (matriz dispersa).

Este ejemplo ilustra cómo la técnica Bag of Words transforma un conjunto de textos en una representación numérica que puede ser utilizada para diversos análisis y modelos de aprendizaje automático, facilitando el procesamiento y análisis de grandes cantidades de datos textuales.

5.2 TF-IDF

¿Qué es un n-grama?

Un n-grama es una secuencia continua de n elementos (palabras, caracteres, etc.) en un texto. En el procesamiento de lenguaje natural, los n-gramas se utilizan para capturar la relación entre palabras y la estructura de un texto.

- **Unigrama (1-grama)**: Consiste en una sola palabra. Ejemplo: "El", "gato", "se".
- **Bigrama (2-grama)**: Secuencia de dos palabras. Ejemplo: "El gato", "gato se".
- **Trigrama (3-grama)**: Secuencia de tres palabras. Ejemplo: "El gato se", "gato se sienta".

Los n-gramas ayudan a capturar más contexto que los unigramas, ya que consideran combinaciones de palabras que pueden tener un significado especial cuando están juntas.

Representación Numérica con TF-IDF

La transformación TF-IDF (Term Frequency - Inverse Document Frequency) es una técnica que pondera la frecuencia de las palabras en un documento considerando también la frecuencia inversa en el corpus total. Esto ayuda a reducir la importancia de palabras muy comunes que no son muy informativas y resalta las palabras que son más importantes para un documento específico.

- **Term Frequency (TF)**: La frecuencia de una palabra en un documento.

- **Inverse Document Frequency (IDF)**: Una medida de cuánto de común o rara es una palabra en todos los documentos del corpus.

La fórmula del TF-IDF para una palabra t en un documento d es:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

donde:

- TF(t,d) es la frecuencia de término de ttt en el documento d.
- N es el número total de documentos.
- DF(t) es el número de documentos que contienen el término t.

Explicación de la Transformación TF-IDF

La transformación TF-IDF pondera las palabras y n-gramas en función de su importancia. Las palabras que son comunes en muchos documentos reciben una puntuación más baja, mientras que las palabras que son más distintivas para un documento reciben una puntuación más alta. Esto ayuda a resaltar las características más relevantes del texto para tareas como clasificación y recuperación de información.

Continuando con nuestro ejemplo de la sección anterior, luego de aplicar CountVectorizer y TfidfTransformer, obtenemos una matriz TF-IDF. A continuación se muestra el vocabulario con bigramas y la matriz TF-IDF resultante:

Vocabulario con Bigramas:

- amigos, casa, el, el gato, el perro, en, en la, gato, gato el, gato se, la, la casa, la ventana, ladra, ladra en, perro, perro ladra, perro son, se, se sienta, sienta, sienta en, son, son amigos, ventana

Matriz TF-IDF:

	amigos	casa	el	el gato	el perro	en	en la	gato	gato el	gato se	...	se	se sienta	sienta	sienta en	son	son amigos	ventana
Texto 1	0	0	0.18456	0.237655	0	0.237655	0.237655	0.237655	0	0.312487	...	0.312487	0.312487	0.312487	0.312487	0	0	0.312487
Texto 2	0	0.348349	0.205741	0	0.264928	0.264928	0.264928	0	0	0	...	0	0	0	0	0	0	0
Texto 3	0.338858	0	0.40027	0.25771	0.25771	0	0	0.25771	0.338858	0	...	0	0	0	0	0.338858	0.338858	0

El resultado muestra un vocabulario que incluye unigramas y bigramas, y la matriz TF-IDF que pondera cada n-grama en los documentos. La transformación TF-IDF pondera las palabras y n-gramas en función de su importancia. Las palabras que son comunes en muchos documentos reciben una puntuación más baja, mientras que las palabras que son más distintivas para un documento reciben una puntuación más alta. Esto ayuda a resaltar las características más relevantes del texto para tareas como clasificación y recuperación de información.

5.3 PCA

PCA sobre el conjunto de entrenamiento

Aplicación de PCA sobre los vectores TF-IDF

En esta sección, se utilizó la técnica de TF-IDF para transformar el conjunto de datos de entrenamiento en una representación numérica que puede ser utilizada en modelos de aprendizaje automático. El objetivo de esta transformación es ponderar la frecuencia de las palabras en cada documento considerando su frecuencia inversa en el corpus total, lo que permite destacar las palabras más relevantes y reducir la influencia de aquellas que son demasiado comunes y, por tanto, menos informativas.

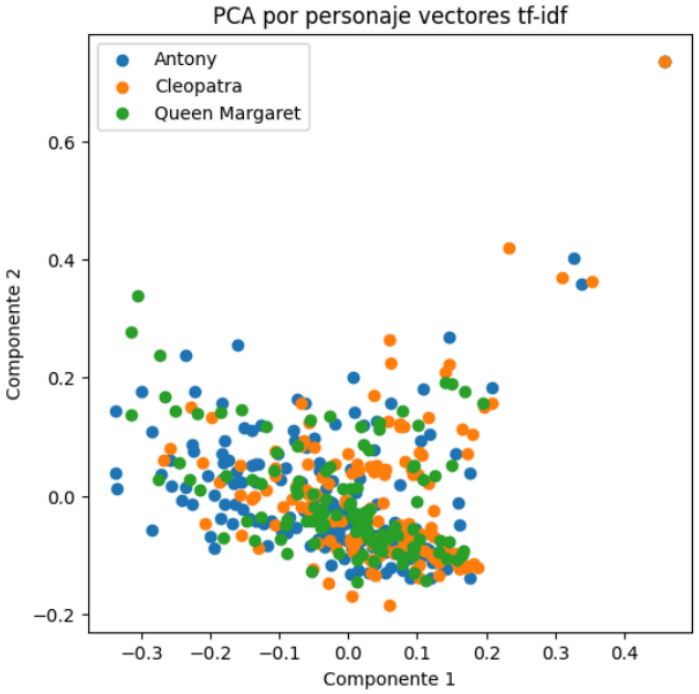
Primero, se aplicó el CountVectorizer para convertir el texto en una matriz de conteos de unigramas(n-grama(1,1)). Posteriormente, se utilizó el TfidfTransformer para transformar esta matriz de conteos en una matriz TF-IDF. Este proceso se llevó a cabo tanto con el parámetro use_idf=False, obteniendo una simple frecuencia de términos, como con use_idf=True, aplicando la ponderación inversa de frecuencia de documentos.

En el notebook se presenta una muestra de la matriz TF-IDF obtenida, que ilustra cómo cada documento se representa como un vector en el que cada dimensión corresponde a una palabra del vocabulario y el valor en cada posición es la puntuación TF-IDF de esa palabra en el documento.

La matriz resultante tiene dimensiones (438, 2807), lo que indica que hay 438 documentos en el conjunto de entrenamiento y un vocabulario de 2807 palabras únicas.

A continuación se aplicó PCA para reducir los vectores de TF-IDF a dos componentes principales. Esto permite visualizar los datos en un espacio bidimensional. La matriz de entrenamiento resultante después de la reducción con PCA tiene dimensiones de 438x2.

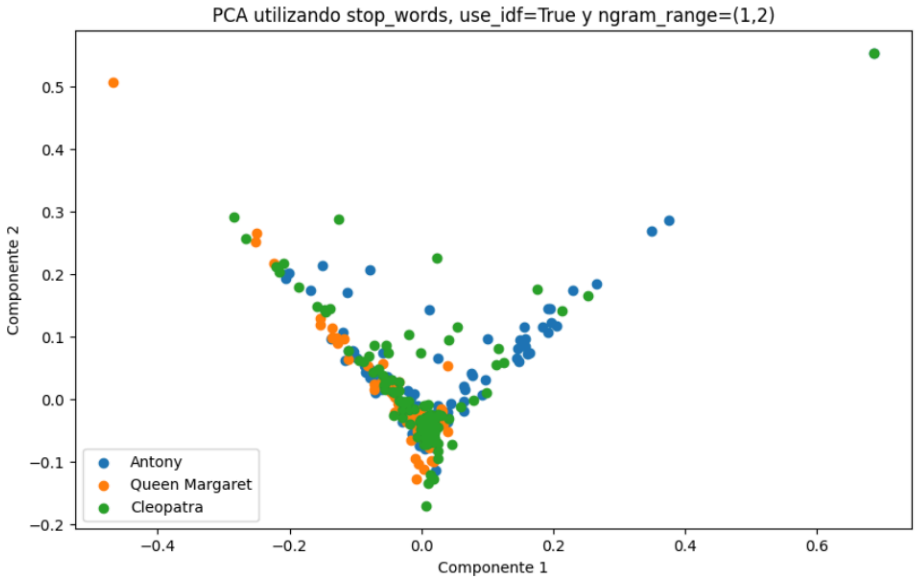
Se graficaron las dos primeras componentes principales, coloreando los puntos según el personaje al que corresponden. Esto permite observar cómo se distribuyen los diferentes personajes en el espacio de las componentes principales.



Al observar la gráfica, se puede ver que las dos primeras componentes principales capturan cierta variación en los datos, pero no separan claramente los textos de los diferentes personajes. Esto sugiere que, utilizando solo dos componentes principales, no es posible distinguir completamente entre los textos de Antony, Cleopatra y Queen Margaret.

PCA sobre el conjunto de entrenamiento considerando stop words en inglés y ngram=(1,2)

En esta sección, se aplicó Análisis de Componentes Principales (PCA) a los vectores TF-IDF del conjunto de entrenamiento utilizando un enfoque más refinado. Este enfoque incluye el filtrado de palabras comunes (stop words) en inglés, el uso del parámetro use_idf=True y la consideración de bigramas (ngram_range=(1,2)).



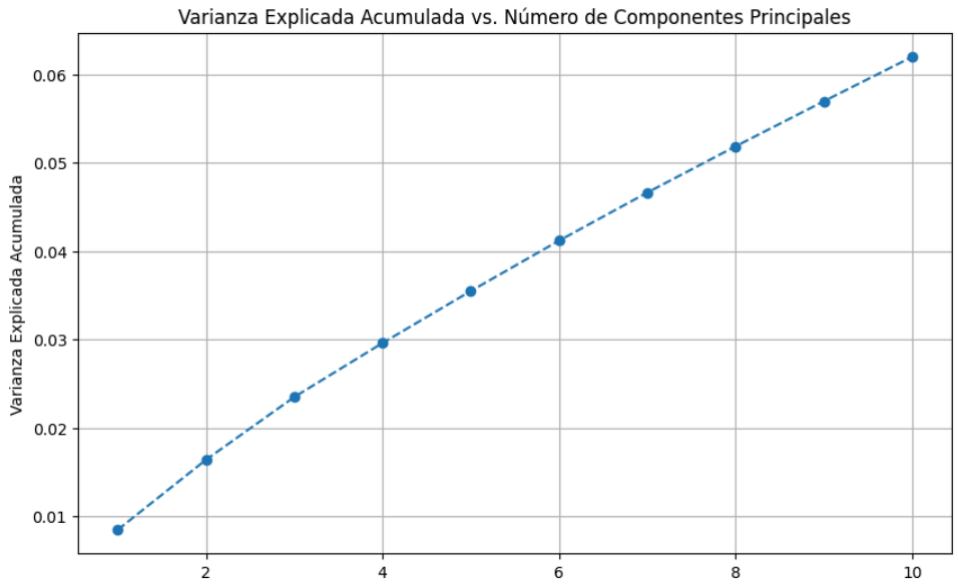
¿Se Pueden Separar los Personajes Utilizando Solo 2 Componentes Principales?

Comparando esta gráfica con la anterior, no se observa una mejora significativa en la separación de los personajes. Los textos de Cleopatra, Antony y Queen Margaret siguen estando mezclados y no hay una diferenciación clara entre ellos utilizando solo dos componentes principales.

Esto sugiere que, a pesar de utilizar un enfoque más sofisticado con el filtrado de stopwords, el uso de IDF y la inclusión de bigramas, las dos primeras componentes principales no capturan suficiente información para separar los personajes de manera efectiva.

Se procederá a analizar cómo varía la varianza explicada a medida que se agregan más componentes principales (hasta 10 componentes). Esto ayudará a entender cuántas componentes son necesarias para capturar una proporción significativa de la varianza total en los datos.

Variación de la Varianza Explicada



El gráfico muestra cómo la varianza explicada se incrementa a medida que se añaden más componentes principales.

Observamos que las primeras dos componentes principales explican aproximadamente el 2% de la varianza total en los datos. Incluso al considerar hasta 10 componentes principales, la varianza explicada acumulada no supera el 6%. Esto indica que los datos son altamente dispersos y que se necesitan muchos más componentes principales para capturar una proporción significativa de la varianza.

La baja varianza explicada por las primeras componentes principales sugiere que la información relevante para separar a los personajes está distribuida a lo largo de muchas dimensiones, lo que dificulta la separación clara con pocas componentes principales.

6. Entrenamiento y Evaluación de Modelos

6.1 Multinomial Naive Bayes

El modelo **Multinomial Naive Bayes** es una técnica de clasificación utilizada especialmente para la clasificación de documentos y problemas relacionados con el procesamiento del lenguaje natural. Este modelo asume que las características (palabras) se distribuyen de manera multinomial dado el tipo de clase. Es particularmente útil cuando se trabaja con datos de texto, donde la representación de las características se hace frecuentemente mediante conteos o frecuencias de palabras.

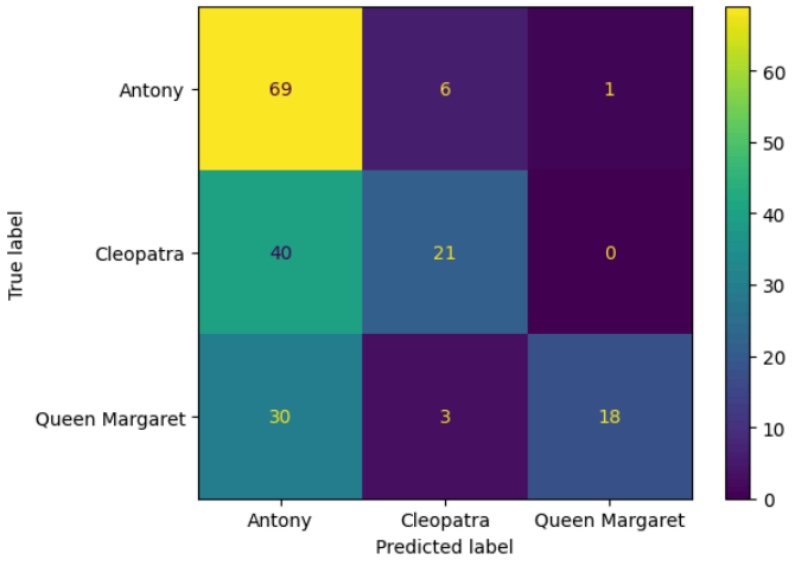
Predicción sobre test y análisis

Entrenamos el modelo utilizando el conjunto de datos de entrenamiento transformado y realizamos predicciones sobre los datos de prueba. A continuación, evaluamos el desempeño del modelo mediante la métrica de precisión (accuracy) y generamos la matriz de confusión.

Resultados

Accuracy, Matriz de confusión, Reporte de clasificación(precisión y recall).

Accuracy: 0.574468085106383



	precision	recall	f1-score	support
Antony	0.50	0.91	0.64	76
Cleopatra	0.70	0.34	0.46	61
Queen Margaret	0.95	0.35	0.51	51
accuracy			0.57	188
macro avg	0.71	0.54	0.54	188
weighted avg	0.68	0.57	0.55	188

Precisión y recall son métricas que nos permiten entender el rendimiento de nuestro modelo en cada clase específica. **Precisión** mide la proporción de verdaderos positivos sobre todos los positivos predichos, mientras que **recall** mide la proporción de verdaderos positivos sobre todos los positivos reales.

Los valores de precisión y recall se derivan de la matriz de confusión. Precisión para una clase se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos. Recall se calcula como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos.

Precisión = Verdaderos positivos / (Verdaderos positivos + Falsos positivos)

Recall = Verdaderos positivos / (Verdaderos positivos + Falsos negativos)

Adicionalmente, el **F1 Score** es una métrica de evaluación utilizada en problemas de clasificación que combina la precisión y el recall en una sola métrica. Es especialmente útil cuando se tiene un desequilibrio en las clases. El F1 Score se calcula como la media armónica de la precisión y el recall, proporcionando un equilibrio entre ambos:

$$F1\,Score = 2 \times \frac{(\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}$$

A continuación, se detallan los resultados para cada personaje:

- **Antony:**
    - **Precisión:** 0.50
    - **Recall:** 0.91

La precisión de 0.50 significa que, de todas las veces que el modelo predijo "Antony", el 50% de esas predicciones fueron correctas. En la matriz de confusión, de las 76 muestras reales de "Antony", 69 fueron correctamente identificadas, pero también se cometieron 7 errores (6 fueron predicciones de "Cleopatra" y 1 de "Queen Margaret").

El recall de 0.91 indica que el 91% de todas las muestras que realmente eran "Antony" fueron correctamente identificadas por el modelo. En la matriz de confusión, de las 76 muestras reales de "Antony", 69 fueron correctamente identificadas.

- **Cleopatra:**
    - **Precisión:** 0.70

    - **Recall:** 0.34

La precisión de 0.70 significa que, de todas las veces que el modelo predijo "Cleopatra", el 70% de esas predicciones fueron correctas. En la matriz de confusión, esto se refleja en 21 predicciones correctas para "Cleopatra" y 40 predicciones incorrectas (que fueron confundidas con "Antony" y "Queen Margaret").

El recall de 0.34 indica que solo el 34% de todas las muestras que realmente eran "Cleopatra" fueron correctamente identificadas por el modelo. En la matriz de confusión, de las 61 muestras reales de "Cleopatra", 21 fueron correctamente identificadas.

- **Queen Margaret:**
    - **Precisión:** 0.95
    - **Recall:** 0.35

La precisión de 0.95 significa que, de todas las veces que el modelo predijo "Queen Margaret", el 95% de esas predicciones fueron correctas. En la matriz de confusión, esto se refleja en 18 predicciones correctas para "Queen Margaret" y 3 predicciones incorrectas.

Sin embargo, el recall de 0.35 indica que solo el 35% de todas las muestras que realmente eran "Queen Margaret" fueron correctamente identificadas por el modelo. En la matriz de confusión, de las 51 muestras reales de "Queen Margaret", 18 fueron correctamente identificadas.

Resumen:

- **Antony** tiene una alta tasa de recall (0.91), lo que significa que la mayoría de las muestras reales de "Antony" fueron correctamente identificadas. Sin embargo, su precisión es moderada (0.50), lo que indica que hay un número significativo de falsos positivos (predicciones incorrectas de "Antony").
- **Cleopatra** tiene una precisión relativamente alta (0.70), pero su recall es bajo (0.34), lo que significa que, aunque la mayoría de las predicciones de "Cleopatra" son correctas, el modelo no logra identificar muchas de las instancias reales de "Cleopatra".
- **Queen Margaret** tiene una precisión muy alta (0.95), lo que significa que cuando el modelo predice "Queen Margaret", casi siempre es correcto. Sin embargo, su recall es bajo (0.35), indicando que el modelo falla en identificar muchas de las instancias reales de "Queen Margaret".

Estos resultados reflejan un modelo que tiene problemas para equilibrar la identificación correcta de todas las clases, especialmente en conjuntos de datos desbalanceados. La precisión y el recall son métricas complementarias que ayudan a entender mejor el rendimiento del modelo en cada clase.

Discusión sobre la Métrica de Accuracy

Mirar solo el valor de accuracy puede ser engañoso, especialmente en conjuntos de datos desbalanceados. En este caso, aunque tenemos una accuracy de 0.57, esto no refleja adecuadamente el rendimiento del modelo en cada clase. Esto se debe a que el modelo podría estar prediciendo principalmente la clase mayoritaria ("Antony" en este caso), ignorando las clases menos representadas.

Antony es la clase que más se encuentra presente en la base. Un alto valor de accuracy podría indicar que el modelo está funcionando bien, pero en realidad podría estar ignorando las clases minoritarias y simplemente prediciendo la clase mayoritaria. Esto no proporciona una visión precisa del rendimiento del modelo para todas las clases.

Si el desbalance de datos fuera aún mayor, el valor de accuracy se vería aún más distorsionado. El modelo podría lograr una alta precisión prediciendo siempre la clase más común, mientras que fallaría en identificar correctamente las clases menos representadas. Esto enfatiza la importancia de utilizar métricas adicionales como precisión, recall y F1-score para evaluar el rendimiento del modelo de manera más completa.

6.2 Cross-validation

La validación cruzada es una técnica utilizada para evaluar la capacidad de generalización de un modelo de aprendizaje automático. Consiste en dividir el conjunto de datos en varias partes o "pliegues". En cada iteración, una de las partes se utiliza como conjunto de validación y las restantes como conjunto de entrenamiento. Este proceso se repite hasta que cada parte haya sido utilizado una vez como conjunto de validación. Los resultados se promedian para proporcionar una estimación más robusta del rendimiento del modelo, ayudando a detectar problemas como el sobreajuste.

Existen varias variaciones de la validación cruzada, cada una con características específicas:

1. **K-Fold Cross-Validation:** Es la variación más común, donde los datos se dividen en 'k' pliegues de igual tamaño. El proceso se repite 'k' veces, y cada pliegue se usa exactamente una vez como conjunto de validación. Finalmente, se promedian los resultados para obtener una estimación global.

2. **Stratified K-Fold Cross-Validation:** Similar a la K-Fold, pero en este caso, la división de los datos se hace de manera que cada pliegue tenga la misma proporción de cada clase, lo cual es especialmente útil en conjuntos de datos desbalanceados.

3. **Leave-One-Out Cross-Validation (LOOCV):** Es un caso extremo de K-Fold donde 'k' es igual al número de muestras en el conjunto de datos. En cada iteración, sólo una muestra se utiliza como conjunto de validación y el resto como conjunto de entrenamiento. Aunque proporciona una evaluación muy precisa, puede ser computacionalmente costosa para conjuntos de datos grandes.

Estas variaciones de la validación cruzada permiten evaluar el rendimiento de un modelo de manera más precisa y confiable, adaptándose a diferentes características y restricciones de los conjuntos de datos.

Combinaciones adicionales para param_sets

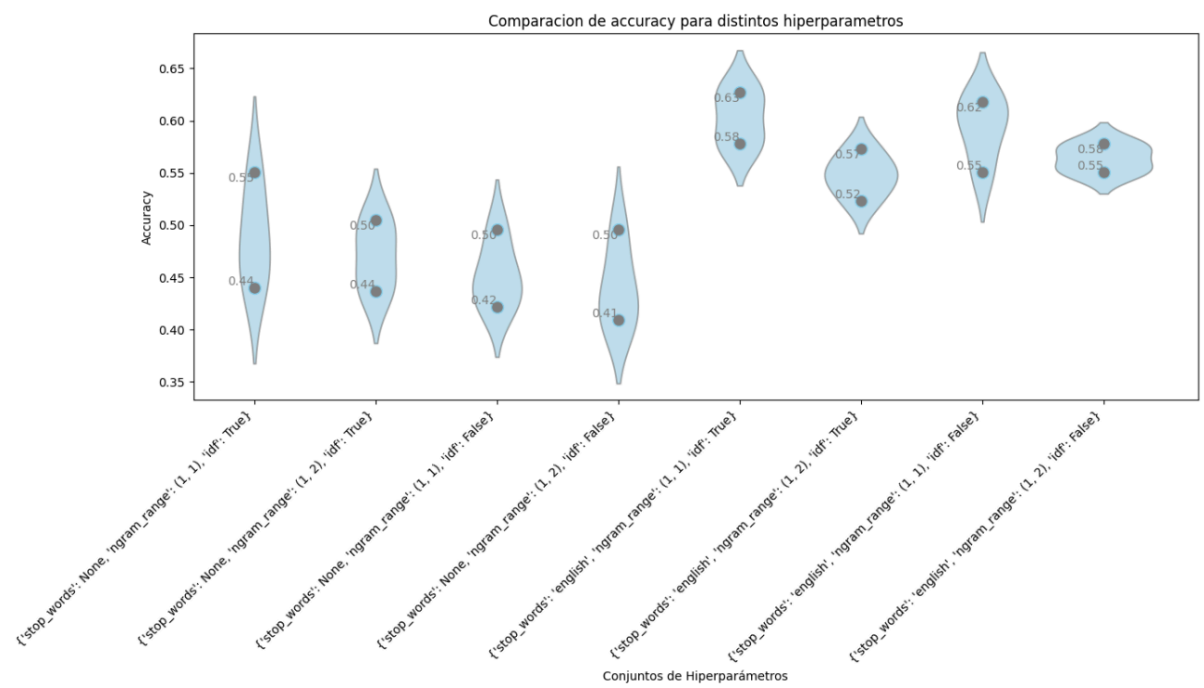
Las combinaciones adicionales en param_sets están diseñadas para explorar cómo diferentes configuraciones de hiperparámetros afectan el rendimiento del modelo. Aplicamos las siguientes:

1. Sin stop words, unigrama, con IDF
2. Sin stop words, bigrama, con IDF
3. Sin stop words, unigrama, sin IDF
4. Sin stop words, bigrama, sin IDF
5. Con stop words en inglés, unigrama, con IDF
6. Con stop words en inglés, bigrama, con IDF
7. Con stop words en inglés, unigrama, sin IDF
8. Con stop words en inglés, bigrama, sin IDF

Utilizamos StratifiedKFold que es una técnica de validación cruzada que garantiza que la distribución de las clases sea la misma en cada parte. Como comentamos previamente, esto es especialmente útil en problemas de clasificación donde las clases pueden estar desbalanceadas.

Se divide el conjunto de datos en 4 partes(folds). Cada parte se usa una vez como conjunto de validación y las otras tres veces como conjunto de entrenamiento. A su vez, utilizamos el shuffle, para mezclar los datos antes de dividirlos en partes. Esto ayuda a garantizar que cada parte sea representativa de la distribución de los datos.

```
Params: {'stop_words': None, 'ngram_range': (1, 1), 'idf': True} | Mean Accuracy: 0.4886 | Std: 0.0416
Params: {'stop_words': None, 'ngram_range': (1, 2), 'idf': True} | Mean Accuracy: 0.4704 | Std: 0.0282
Params: {'stop_words': None, 'ngram_range': (1, 1), 'idf': False} | Mean Accuracy: 0.4521 | Std: 0.0275
Params: {'stop_words': None, 'ngram_range': (1, 2), 'idf': False} | Mean Accuracy: 0.4407 | Std: 0.0346
Params: {'stop_words': 'english', 'ngram_range': (1, 1), 'idf': True} | Mean Accuracy: 0.6027 | Std: 0.0229
Params: {'stop_words': 'english', 'ngram_range': (1, 2), 'idf': True} | Mean Accuracy: 0.5479 | Std: 0.0177
Params: {'stop_words': 'english', 'ngram_range': (1, 1), 'idf': False} | Mean Accuracy: 0.5936 | Std: 0.0270
Params: {'stop_words': 'english', 'ngram_range': (1, 2), 'idf': False} | Mean Accuracy: 0.5639 | Std: 0.0117
```



La visualización muestra la comparación del accuracy obtenida en cada iteración de validación cruzada para distintos conjuntos de hiper parámetros. Cada violín representa la distribución de los valores de accuracy a lo largo de las diferentes partes (folds) de validación cruzada para un conjunto específico de hiper parámetros.

En el eje X, se listan los distintos conjuntos de hiper parámetros evaluados, que incluyen combinaciones de uso de stopwords (sin stop words y con stop words en inglés), rangos de n-gramas (unigramas y bigramas), y uso de IDF (Inversa Frecuencia de Documento). En el eje Y, se muestra la precisión (accuracy) alcanzada.

Los gráficos de violín permiten observar la variabilidad de la accuracy a través de las partes. Los puntos y las etiquetas dentro de cada violín indican los valores mínimos y máximos de accuracy, proporcionando una visión clara de los rangos de rendimiento de cada modelo.

Variabilidad de la accuracy:

- Algunos conjuntos de hiper parámetros muestran una mayor variabilidad en la accuracy (con bigramas y sin IDF), lo que sugiere que el rendimiento del modelo puede ser inconsistente dependiendo del pliegue de validación.

- Otros conjuntos de hiper parámetros tienen una variabilidad menor, indicando un rendimiento más consistente.

Rendimiento Global:

- Los modelos que utilizan stop words en inglés y bigramas tienden a tener accuracy más altas en general, con menores valores mínimos y máximos, indicando que estos parámetros podrían ser más efectivos.

- Por otro lado, los modelos sin stop words muestran una mayor dispersión y en algunos casos menor accuracy, lo que sugiere que la eliminación de stopwords podría mejorar el rendimiento.

Impacto del Uso de IDF:

- La inclusión del IDF parece mejorar la accuracy en varios conjuntos de hiper parámetros, como se observa en los violines con valores más altos y consistentes.
- Sin embargo, la combinación de bigramas sin IDF muestra una variabilidad significativa, lo que sugiere que IDF es particularmente beneficioso cuando se utilizan n-gramas más grandes.

6.3 Selección de modelo

Primero, identificamos el conjunto de hiper parámetros que produjo el mejor rendimiento (mayor accuracy promedio) en la validación cruzada. Luego, usaremos estos parámetros para entrenar el modelo sobre todo el conjunto de entrenamiento disponible.

Mejores parámetros seleccionados:

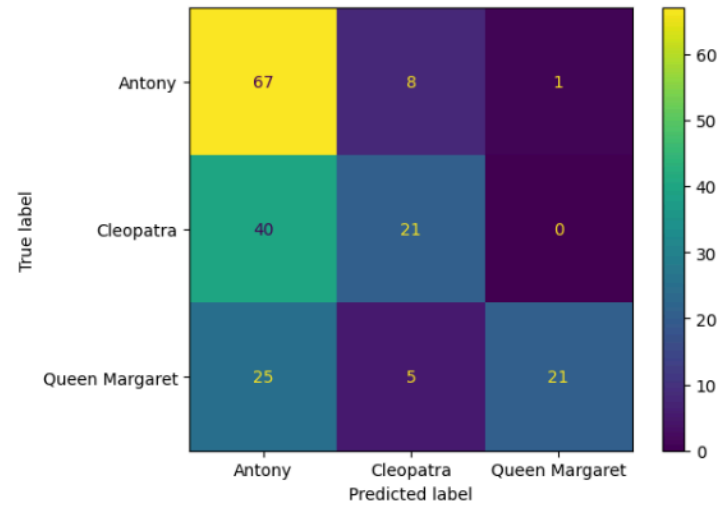
{'stop_words': 'english', 'ngram_range': (1, 1), 'idf': True}

Resultados

```
Final Accuracy: 0.5798
Final Classification Report:
                precision    recall  f1-score   support

       Antony       0.51      0.88      0.64        76
    Cleopatra       0.62      0.34      0.44        61
Queen Margaret      0.95      0.41      0.58        51

     accuracy                           0.58       188
    macro avg       0.69      0.55      0.55       188
 weighted avg       0.66      0.58      0.56       188
```



Los resultados obtenidos con los mejores hiper parámetros muestran una ligera mejora, pero no una accuracy significativamente mejor. Esto era esperable dado que el conjunto de

datos parece tener un desbalance entre las diferentes clases (personajes). Este desbalance puede llevar al modelo a favorecer la clase mayoritaria, resultando en un rendimiento pobre en las clases minoritarias.

Se podría modificar el proceso de entrenamiento para tener en cuenta el desbalance de clases usando pesos de clase. Esto puede ayudar al modelo a prestar más atención a las clases minoritarias.

Limitaciones de Utilizar un Modelo Basado en Bag-of-Words o TF-IDF

Los modelos basados en Bag-of-Words (BoW) y TF-IDF tienen varias limitaciones en cuanto al análisis de texto.

En primer lugar, ambos métodos ignoran el contexto y el orden de las palabras. Tratan el texto como una simple colección de palabras, sin considerar las relaciones sintácticas y semánticas entre ellas. Esto puede llevar a una comprensión limitada del significado del texto, ya que el contexto es crucial para interpretar el sentido de las palabras en una oración.

Otra limitación significativa es la alta dimensionalidad que estos métodos suelen generar. Al considerar un gran número de unigramas, bigramas o incluso n-gramas más largos, los vectores de características pueden volverse muy grandes, lo que no solo aumenta el costo computacional, sino que también puede causar problemas de sobreajuste, especialmente en conjuntos de datos pequeños.

Además, los modelos basados en BoW y TF-IDF no capturan la semántica de las palabras. Por ejemplo, palabras sinónimas como "bueno" y "excelente" se tratarán como características completamente diferentes, lo que puede reducir la efectividad del modelo. Estos métodos también dependen en gran medida del preprocesamiento del texto, como la eliminación de stopwords, lematización o stemming, y diferentes estrategias de preprocesamiento pueden llevar a resultados significativamente distintos.

Por último, estas técnicas no pueden capturar relaciones de largo alcance entre palabras en un documento, ignoran el contexto de las palabras y ello puede llevar a una interpretación errónea (en especial cuando una palabra puede tener distintos significados según el contexto). Para superar muchas de estas limitaciones, se pueden usar técnicas más avanzadas basadas en embeddings de palabras como Word2Vec, GloVe o modelos de lenguaje profundo como BERT, que proporcionan representaciones de texto más ricas y contextualmente informadas.

6.4 Modelo extra: Regresión Logística

La regresión logística es un modelo de clasificación que utiliza una función logística para modelar la probabilidad de que una muestra pertenezca a una clase particular. Es lineal en el espacio de los parámetros y puede manejar relaciones no lineales al combinar múltiples características. Este modelo es adecuado para problemas de clasificación binaria y multiclase, y puede ser regularizado para prevenir el sobreajuste.

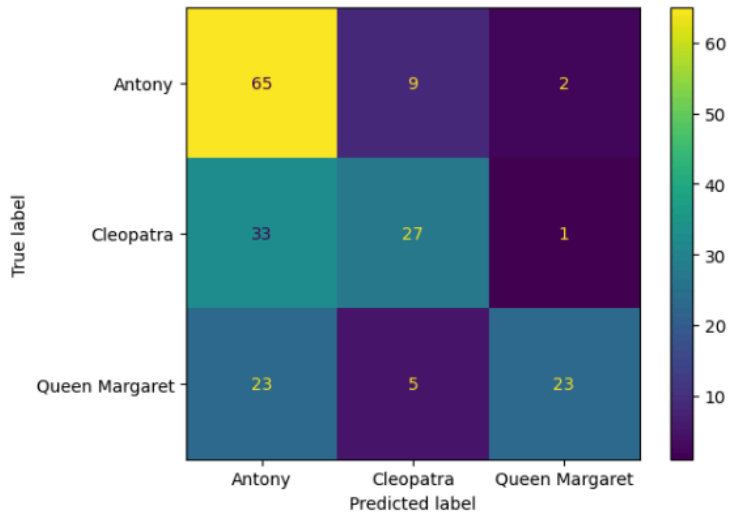
Seleccionamos la regresión logística para este ejemplo debido a su simplicidad, eficiencia y buen rendimiento en problemas de clasificación de texto. Es un modelo lineal fácil de entender e implementar, y funciona bien con características dispersas y de alta dimensionalidad, como las generadas por TF-IDF. Además sus coeficientes pueden ser interpretados para entender la importancia relativa de diferentes características, esto le da una ventaja ante modelos de “caja negra”.

Sin embargo, también hay otros modelos en scikit-learn que podrían ser evaluados para mejorar el rendimiento. Por ejemplo, Support Vector Machines (SVM) son potentes para clasificación de texto y pueden manejar alta dimensionalidad, aunque pueden ser computacionalmente costosos. Los Random Forests pueden capturar relaciones no lineales y son robustos a características irrelevantes, pero pueden ser menos interpretativos y más lentos de entrenar.

```
Final Classification Report with Logistic Regression:
                precision    recall  f1-score   support

       Antony       0.54      0.86      0.66        76
    Cleopatra       0.66      0.44      0.53        61
Queen Margaret      0.88      0.45      0.60        51

     accuracy                           0.61       188
    macro avg       0.69      0.58      0.60       188
 weighted avg       0.67      0.61      0.60       188
```



Accuracy con regresión logística: 0.6117

Métrica	Naive Bayes Multinomial	Regresión Logística
Accuracy	0.5798	0.6117
Macro Avg		
- Precision	0.69	0.69
- Recall	0.55	0.58
- F1-Score	0.55	0.6
Weighted Avg		
- Precision	0.66	0.67
- Recall	0.58	0.61
- F1-Score	0.56	0.6
Antony		
- Precision	0.51	0.54
- Recall	0.88	0.86
- F1-Score	0.64	0.66
Cleopatra		
- Precision	0.62	0.66
- Recall	0.34	0.44
- F1-Score	0.44	0.53
Queen Margaret		
- Precision	0.95	0.88
- Recall	0.41	0.45
- F1-Score	0.58	0.6

La Regresión Logística supera a Naive Bayes Multinomial en términos de accuracy global, con una mejora de 0.5798 a 0.6117. Para la clase "Antony", la Regresión Logística muestra una mejora en el f1-score de 0.64 a 0.66, manteniendo un recall alto aunque ligeramente menor al de Bayes.

Para "Cleopatra", la Regresión Logística mejora significativamente tanto en precisión como en recall, resultando en un aumento del f1-score de 0.44 a 0.53. Aunque la precisión para "Queen Margaret" disminuye ligeramente en la Regresión Logística, el recall y el f1-score mejoran, indicando un mejor equilibrio en la clasificación de esta clase.

En general, la Regresión Logística proporciona un mejor rendimiento en términos de precisión, recall y f1-score en comparación con Naive Bayes Multinomial, esto nos permite afirmar que maneja de mejor manera el desbalance de clases de este conjunto de datos.

6.5 Modificación de personaje

Implementación

Realizamos un conteo de párrafos por personajes para seleccionar personajes similares en número de párrafos:

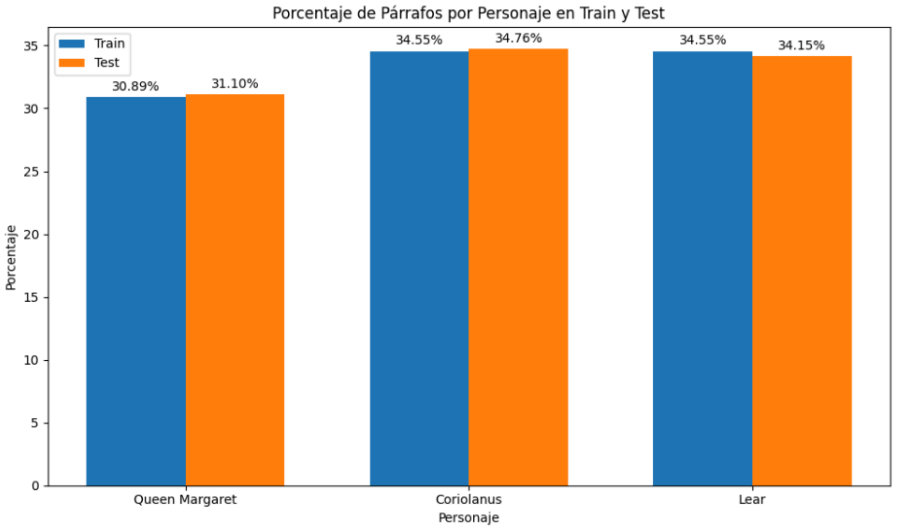
```
CharName
Coriolanus                                189
Lear                                      188
Henry VI                                  183
Earl of Warwick                           182
Richard Plantagenet (Duke of Gloucester)  172
Name: count, dtype: int64
```

En el análisis previo, se seleccionaron tres personajes para el modelo de clasificación: **Queen Margaret, Cleopatra y Antony**.

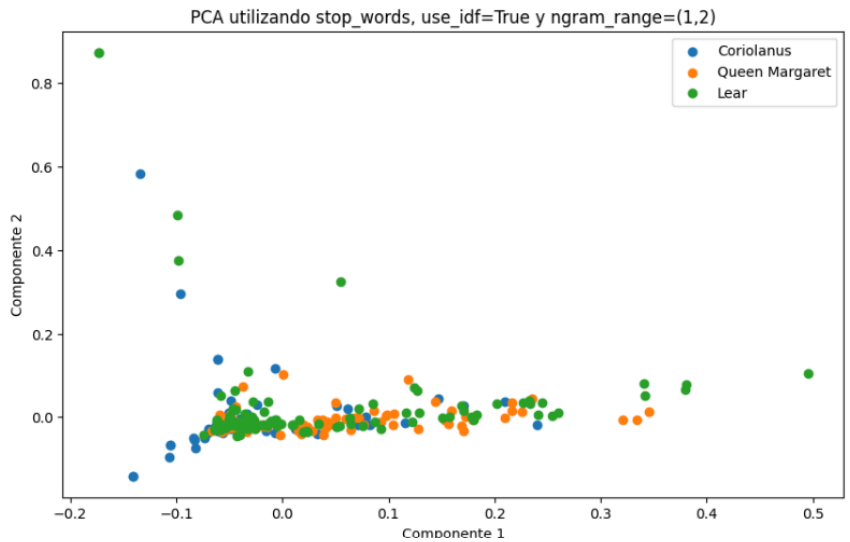
En esta evaluación, se cambiaron dos personajes: **Antony** fue reemplazado por **Coriolanus** y **Cleopatra** fue reemplazada por **Lear**. Los nuevos personajes, **Coriolanus** y **Lear**, fueron seleccionados en base a un análisis cuantitativo de la cantidad de párrafos que cada personaje tenía en el corpus de datos. El objetivo principal de esta selección fue asegurar un conjunto de datos equilibrado para el entrenamiento y evaluación del modelo.

Se realizó una visualización para constatar el balance en los conjuntos de entrenamiento y prueba, y se aplicó PCA para observar la distribución de los datos en un espacio de menor dimensionalidad.

La gráfica de porcentaje de párrafos por personaje muestra un balance razonable entre los conjuntos de entrenamiento y prueba para los personajes seleccionados.



Se aplica PCA con stop_words, idf y ngram (1,2).



A simple vista, la implementación no mejoró en términos de diferenciación de personajes. Los personajes siguen sin ser fácilmente separables en el espacio de componentes principales. Esto sugiere que el problema puede ser más complejo y que PCA por sí sola no es suficiente para mejorar la clasificación. Para abordar estos desafíos, podemos considerar técnicas de submuestreo y sobremuestreo para mejorar la representación de los datos.

Sobre-Muestreo

El sobre-muestreo implica aumentar el número de muestras de las clases minoritarias. Una técnica común es SMOTE (Synthetic Minority Over-sampling Technique), que genera nuevas muestras sintéticas en el espacio de características. Esto ayuda a equilibrar el conjunto de datos y mejorar el rendimiento del modelo en las clases minoritarias.

Submuestreo

El submuestreo, por otro lado, reduce el número de muestras de las clases mayoritarias. Esto puede ayudar a que el modelo no se sesgue hacia la clase más común. Puede ser útil cuando hay una gran disparidad entre las clases, pero puede resultar en la pérdida de información valiosa de las clases mayoritarias si se eliminan demasiadas muestras.

Dada la falta de diferenciación en las visualizaciones de PCA, aplicar submuestreo y sobremuestreo podría ser útil para:

1. **Balance de Datos:** Equilibrar las clases podría mejorar la capacidad del modelo para aprender características distintivas de cada clase.
2. **Modelo de Estimación:** Después de balancear los datos, el modelo de estimación (como Naive Bayes) podría tener un mejor desempeño, al no estar sesgado hacia la clase mayoritaria.

6.6 Técnicas alternativas para extraer features de texto

Una técnica alternativa para extraer features de texto es el uso de word embeddings. Los word embeddings son representaciones vectoriales densas de palabras en un espacio de

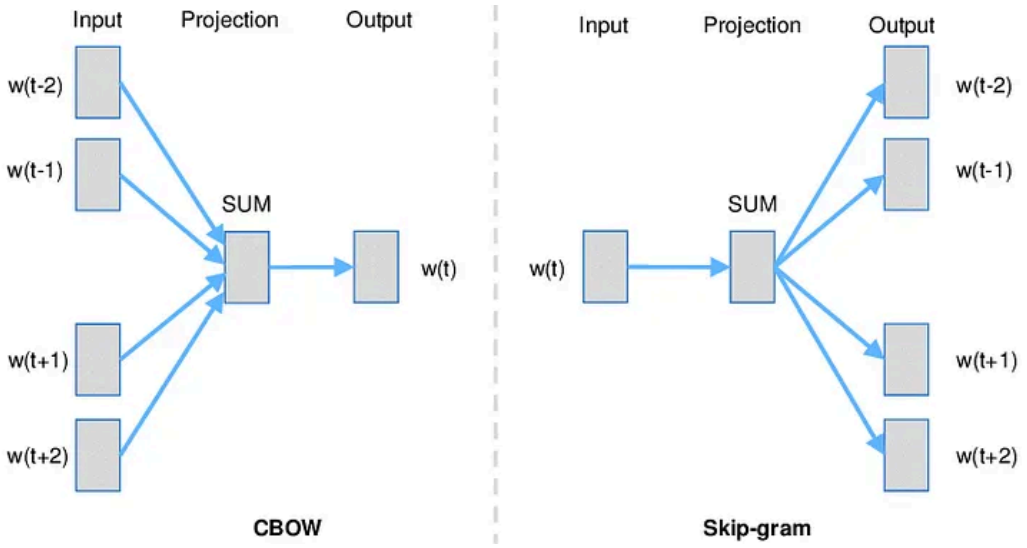
características continuo, donde palabras con significados similares tienen representaciones vectoriales similares

Word2Vec

Es un modelo de aprendizaje profundo que genera representaciones vectoriales de palabras (también conocidas como embeddings) a partir de grandes corpus de texto. Estas representaciones vectoriales permiten que las palabras con significados similares tengan representaciones similares en un espacio vectorial.

Este modelo representado en el espacio nos permite introducir analogías, es decir que existen relaciones lineales entre palabras con significados similares. Por ejemplo vector (rey) — vector (‘hombre’) + vector (‘mujer) nos posicionará cerca del vector (‘reina’).

Para entrenar este modelo se utilizan dos arquitecturas principales:



1. **Modelo CBOW (Continuous Bag of Words):**

Este modelo predice una palabra basada en su contexto (palabras vecinas). Toma una ventana de palabras alrededor de una palabra objetivo y utiliza estas palabras para predecir la palabra objetivo. El modelo aprende a representar cada palabra en un espacio de alta dimensionalidad.

2. **Modelo Skip-Gram:**

Este modelo hace lo contrario: toma una palabra objetivo y trata de predecir sus palabras vecinas. Es eficaz para aprender representaciones útiles incluso con pocas muestras.

Las principales limitaciones que se le atribuye al modelo son:

- Los balances de sus estimaciones son estáticos, es decir, que no toma en cuenta todo el contexto de la palabra al momento de estima, e incluso en su entrenamiento, la cantidad de palabras que puede tomar como contexto es limitada.
- Manejo de palabras por fuera del entrenamiento, el modelo no predice correctamente palabras a la que no estuvo expuesto o variaciones de palabras que si vio durante el entrenamiento.

- El modelo genera un solo vector por palabra, esto significa que una palabra no puede tener dos o más significados en diferentes contextos.

BERT (Bidirectional Encoder Representations from Transformers)

BERT es un modelo de lenguaje profundo basado en transformadores que genera representaciones contextuales bidireccionales de palabras. A diferencia de Word2Vec y GloVe, que generan embeddings estáticos, BERT considera el contexto completo de una palabra en una oración, proporcionando embeddings dinámicos que varían según el contexto.

Diferencias Esperadas en los Resultados

- **Captura de Semántica y Contexto:** Los word embeddings capturan mejor las relaciones semánticas y contextuales entre palabras. Mientras que TF-IDF representa cada palabra de forma independiente, los embeddings permiten que el modelo entienda similitudes y relaciones entre palabras, mejorando la capacidad para capturar el significado de las oraciones.
- **Reducción de Dimensionalidad:** Los embeddings generan vectores densos de menor dimensionalidad en comparación con las representaciones dispersas y de alta dimensionalidad de TF-IDF. Esto puede reducir el costo computacional y mejorar la eficiencia del modelo.
- **Robustez en Datos Desbalanceados:** Los embeddings pueden proporcionar características más robustas que mejoran la clasificación de clases minoritarias, ya que capturan mejor la información semántica que puede ser útil para distinguir entre clases.
- **Desempeño en Tareas Complejas:** Para tareas de procesamiento de lenguaje natural (NLP) más complejas, como análisis de sentimientos, traducción automática y preguntas y respuestas, los embeddings basados en modelos como BERT han demostrado un rendimiento superior debido a su capacidad para entender el contexto completo.

Implementación World 2 Vec

	precision	recall	f1-score	support
Antony	0.40	0.91	0.55	76
Cleopatra	0.20	0.05	0.08	61
Queen Margaret	0.00	0.00	0.00	51
accuracy			0.38	188
macro avg	0.20	0.32	0.21	188
weighted avg	0.23	0.38	0.25	188

Usamos biblioteca gensim.

El uso de Word2Vec en este caso específico no mejoró el rendimiento del modelo en comparación con Naive Bayes Multinomial y Regresión Logística. Las técnicas de embeddings como Word2Vec pueden requerir un ajuste más fino y un preprocesamiento adicional para ser efectivas. Además, el contexto y la calidad del corpus de entrenamiento pueden influir significativamente en los resultados. Las técnicas basadas en TF-IDF

combinadas con modelos como Regresión Logística parecen ser más efectivas para este conjunto de datos y tarea de clasificación específica.

Implementación BERT

```
Final Accuracy with BERT: 0.5532
Final Classification Report with BERT:
                precision    recall  f1-score   support

       Antony       0.54      0.64      0.59        76
     Cleopatra      0.44      0.36      0.40        61
Queen Margaret      0.70      0.65      0.67        51

      accuracy                           0.55       188
     macro avg      0.56      0.55      0.55       188
  weighted avg      0.55      0.55      0.55       188
```

Los resultados al implementar BERT son similares que Word2Vec, indicando nuevamente que puede requerir un ajuste más fino y preprocesamiento adicional para utilizar estas técnicas.

6.7 Modelo fasttext

El modelo fastText surge de los laboratorios de Facebook como una alternativa rápida, eficiente y escalable para clasificación de texto, con buenos resultados en términos de precisión y facilidad de uso.

La innovación de fastText fue plantear las palabras como una “bolsa” de n-gramas. Este proceso divide la palabra en pequeños grupos de caracteres, los agrupa cada tantos caracteres como se les haya indicado, luego se crea un único vector como la suma o promedio de todos los caracteres del n-grama. Este proceso de subdivisión de las palabras le da una ventaja con respecto a word2vec cuando se tiene que enfrentar con variaciones de palabras o palabras que están fuera del vocabulario con el que fue entrenado.

Ventajas

1. **Velocidad y Eficiencia**: fastText es conocido por su rapidez en el entrenamiento y la inferencia, lo que lo hace adecuado para aplicaciones en tiempo real y grandes volúmenes de datos.

2. **Simplicidad**: La implementación de fastText es sencilla y requiere menos ajustes en comparación con otros modelos más complejos.

3. **Escalabilidad** Por los dos factores mencionados anteriormente el modelo se considera fácilmente escalable.

4. **Soporte para N-gramas**: fastText puede manejar n-gramas de palabras, lo que le permite capturar dependencias locales y mejorar el rendimiento en tareas de clasificación de texto.

5. **Embeddings de Palabras**: Genera embeddings de palabras preentrenados que pueden ser utilizados para otras tareas de procesamiento de lenguaje natural.

6. **Manejo de palabras OOV(Out of vocabulary):** El modelo maneja de buena manera la clasificación de palabras que no haya visto anteriormente, también variaciones morfologicas.

Desventajas:

1. **Limitaciones Semánticas**: Aunque fastText maneja n-gramas, aún puede tener dificultades para capturar contextos más complejos y relaciones semánticas profundas en comparación con modelos como BERT.

2. **Representaciones Estáticas**: Los embeddings generados por fastText son estáticos, lo que significa que no varían con el contexto, a diferencia de los embeddings contextuales de BERT.

3. **Rendimiento en Tareas Complejas**: En tareas de procesamiento de lenguaje natural más complejas, fastText puede no alcanzar el mismo nivel de precisión que los modelos de lenguaje profundo basados en transformadores.

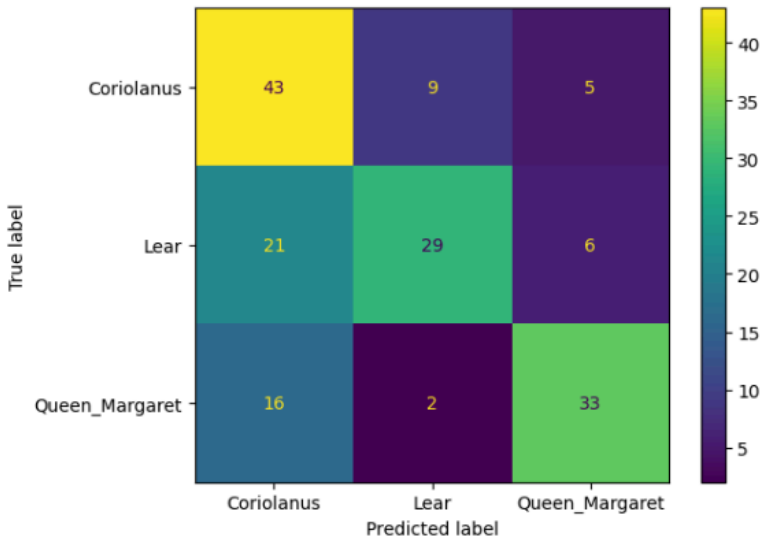
Sin embargo, en tareas que requieren una comprensión más profunda del contexto y las relaciones semánticas, los modelos de lenguaje profundo como BERT pueden proporcionar mejores resultados. Comparar las métricas de evaluación y la matriz de confusión te permitirá determinar si fastText es adecuado para tu caso de uso específico.

Implementación del modelo

```
Final Classification Report with fastText:
                precision    recall  f1-score   support

    Coriolanus       0.54      0.75      0.63        57
          Lear       0.72      0.52      0.60        56
Queen_Margaret       0.75      0.65      0.69        51

      accuracy                           0.64       164
     macro avg       0.67      0.64      0.64       164
  weighted avg       0.67      0.64      0.64       164
```



El informe de clasificación muestra que el modelo tiene un rendimiento variable según el personaje. Para Coriolanus, la precisión es de 0.54, el recall es de 0.75 y el F1-score es de 0.63, lo que indica que el modelo identifica correctamente a Coriolanus en el 54% de las veces que lo predice y cubre el 75% de todas las instancias reales de Coriolanus. Lear tiene una precisión de 0.72, un recall de 0.52 y un F1-score de 0.60, sugiriendo que aunque el modelo es bastante preciso cuando predice a Lear, no lo identifica tan a menudo como debería. Queen Margaret tiene los mejores resultados con una precisión de 0.75, un recall de 0.65 y un F1-score de 0.69.

Respecto de la matriz de confusión, se observa que Coriolanus es frecuentemente identificado correctamente (43 veces), pero hay errores significativos con 9 instancias clasificadas incorrectamente como Lear y 5 como Queen Margaret. Lear presenta una cantidad considerable de errores, siendo confundido como Coriolanus en 21 ocasiones y como Queen Margaret en 6. Por último, Queen Margaret también muestra errores, siendo mal clasificada como Coriolanus en 16 ocasiones y solo 2 veces como Lear.

En general, el modelo tiene una precisión global de 0.67, lo que refleja un rendimiento moderado. Estos resultados sugieren que aunque FastText puede capturar ciertas características distintivas del lenguaje de cada personaje, todavía hay espacio para mejorar la discriminación entre personajes, posiblemente ajustando hiperparámetros o integrando más características de los textos.

7. Conclusiones

Luego de correr varias técnicas de procesamiento de lenguaje, notamos que si bien se puede lograr mejores resultados con modelos más avanzados, esta mejora es marginal. Por ejemplo, la regresión logística tuvo una precisión de 0.61, mientras que fastText 0.67, en una aplicación productiva del modelo, la mejora del 0.06% puede no cubrir el retorno de inversión de la diferencia de costo de procesamiento y de la dificultad extra de implementación.

Existe un problema de raíz ocasionado por el desbalanceamiento de los datos que se vio plasmado en todos los modelos, todos estos estaban sesgados a predecir con mayor frecuencia el grupo mayoritario de la muestra, teniendo problemas para identificar correctamente las categorías minoritarias. Para solucionar este problema se pueden seguir dos estrategias a nivel de datos, primero la de submuestreo, esta requiere reducir la cantidad de ejemplos de la clase mayoritaria para equilibrar el conjunto de datos, esta toma como base que se cuenta con mucha información y por ende descartar una parte no nos afectara gravemente. La segunda y como contracara de esta es el sobre muestreo, esta puede ser llevaba a cabo al aumentar los registros de la clase minoritaria ya sea duplicando ejemplos existentes o generando nuevos ejemplos sintéticos, este último requeriría de un modelo capaz de generar datos sintéticos que se comporten como la clase minoritaria a la cual le queremos atribuir estos datos.

8. Referencias

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

https://nlp.stanford.edu/projects/glove/

https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51#:~:text=BERT%20

https://www.tensorflow.org/text/guide/word_embeddings

https://jalammar.github.io/illustrated-word2vec/

https://medium.com/aimonks/fasttext-revolutionizing-word-embeddings-and-text-classification-fd0d60f11ffa

https://fasttext.cc/

https://medium.com/@pooja93palod/understanding-word2vec-a-beginners-guide-to-word-embeddings-6ecb893dbf61