Tarea 2

Introducción a la Ciencia de Datos

Materia: Introducción a la Ciencia de Datos

Sofía Zeballos

Estudiante de Maestría en Bioinformática

Pablo Schiavone

Estudiante de Maestría en Ciencia de Datos y Aprendizaje Automátic

Montevideo 06 de julio de 2023

Resumen

El objetivo de la tarea es realizar un análisis exploratorio de la base de datos que representa las obras realizadas por Shakespeare. El análisis está compuesto por dos partes: el estudio y preprocesamiento de los datos, y la implementación de algoritmos de aprendizaje automático para la clasificación de algunos personajes. El código implementado e información adicional se encuentra disponible en https://github.com/pschiavone1/ICD2023-GRUPO18.

Introducción

Los datos utilizados en este trabajo provienen de una base de datos publicada en el siguiente link https://relational.fit.cvut.cz/dataset/Shakespeare la cual cuenta con documentación que explica sus relaciones.

Antes de comenzar con el análisis de datos se estudia el esquema relacional (ER) para entender las relaciones y verificar si por la carga semántica de sus tablas y atributos se puede entender a qué objeto de la realidad hace referencia. A continuación se muestra el ER (Figura 1).

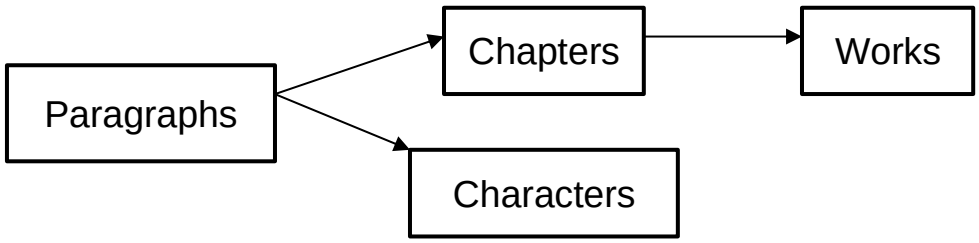

Figura 1: Descripción del esquema relacional de la base.

De acuerdo con el estudio del ER, entendemos que se puede interpretar el objetivo de cada tabla y atributo. A continuación, se muestran las cuatro tablas y lo que se espera de cada atributo.

Chapters		**Chapters**	
id	Identificador de párrafo	**id**	Identificador de capítulo
ParagraphNum	Número de párrafo	**Act**	Acto
PlainText	Líneas del párrafo	**Scene**	Escena
character_id	Identificador del personaje (Relación con Characters)	**Description**	Descripción de l escena
chapter_id	Identificador de capítulo (Relación con Chapter)	**work_id**	Identificador de obra (relación con Works)
Characters		**Works**	
id	Identificador de personaje	**id**	Identificador de obra
CharName	Nombre del personaje	**Title**	Título de la obra
Abbrev	Abreviatura del personaje	**LongTitle**	Título largo de la obra
Description	Descripción del personaje	**Date**	Fecha de publicación
		GenreType	Género literario

Tabla 1: Entidades del ER y atributos correspondientes.

Estudio y pre-procesamiento de los datos.

Una vez entendido el ER se comienza a revisar que la información cargada en cada tabla cumpla con lo esperado.

Durante la primera exploración de datos se observa que en la tabla *Paragraphs* existen datos en el atributo PlainText que no coinciden con el dominio esperado (líneas de un párrafo). También se observa que en los casos que no coincide PlainText con el dominio se repite el charapter_id. En particular el character_id 1261(Figura S 1).

Por otro lado, se detecta que la tabla characters no solo contiene personajes, sino que se utiliza para identificar otro tipo de entidades como directrices. A modo de ejemplo se muestra en la Figura S 2 lo ocurrido con el id 1261 de la tabla characters.

Debido a que no sabemos si el registro con id 1261 es el único que no pertenece a un personaje, se consulta todos los personajes con descripción nula o vacía observándose nuevos registros que no acompañan la interpretación de ER (Figura S 3).

Durante el análisis exploratorio de datos se intentó encontrar un patrón que nos indique qué registros alojados en characters no son un personaje sin encontrar alguno categórico. La mejor aproximación encontrada es con description vacía pudiendo detectar grupos de personajes y acciones como por ejemplo First Apparition, All, Messenger, etc. Por otro lado se detectan nombres de roles, grupos y personajes repetidos (Figura S 4).

Respecto a la tabla de capítulos (df_chapters) se corroboró que tuviera la misma cantidad de obras que la tabla de obras (df_works) y que no hubiera valores nulos en ninguna columna (datos no mostrados, ver notebook adjunto).

Finalmente, la tabla de obras no posee datos nulos ni nombres de obras duplicadas. Adicionalmente se corroboró que la columna ‘GenreType’ no tiene ningún error de escritura (datos no mostrados, ver notebook adjunto).

Siguiendo en la exploración de los datos, buscamos el personaje con mayor cantidad de párrafos. Tomando en cuenta el análisis previo se establecen los siguientes criterios:

- Personajes con el mismo nombre y distinto ID se identifican como personajes distintos.
- Los nombres que identifican a grupos o roles se toman como personajes individuales.
- No se adicionan los párrafos si un personaje participa de forma grupal.

Debido a que la tabla characters es utilizada para múltiples propósitos (además de proporcionar el nombre del personaje), no es posible asegurar que el join de paragraphs y characerts agrupados por charName y character_id nos de la cantidad de párrafos por personaje. Como se observa en la Figura S 5, el “personaje” con la mayor cantidad de párrafos claramente no corresponde a un personaje.

El siguiente “personaje” con mayor cantidad de párrafos es Poet. Para verificar que efectivamente sea un personaje se filtra la tabla characters por el charactr_id obtenidos en el top 10 de cantidad de párrafos. Dada la descripción del Poet podemos afirmar que tampoco es un personaje (Figura S 6).

Dada la agrupación por párrafo más la descripción de charaters podemos decir que el personaje con más párrafos es Sir John Falstaff.

Continuamos evaluando la publicación de obras del autor durante todo el período de publicaciones. Se decidió hacer una gráfica de barras en donde muestra la cantidad de obras realizadas en períodos de dos años (Figura 2). Las 43 obras del autor fueron escritas en un período de 23 años, con un promedio de aproximadamente dos obras por año. Escribió el 60% de su obra en los primeros 12 años de su carrera, con un pico en su producción entre 1593 y 1594.

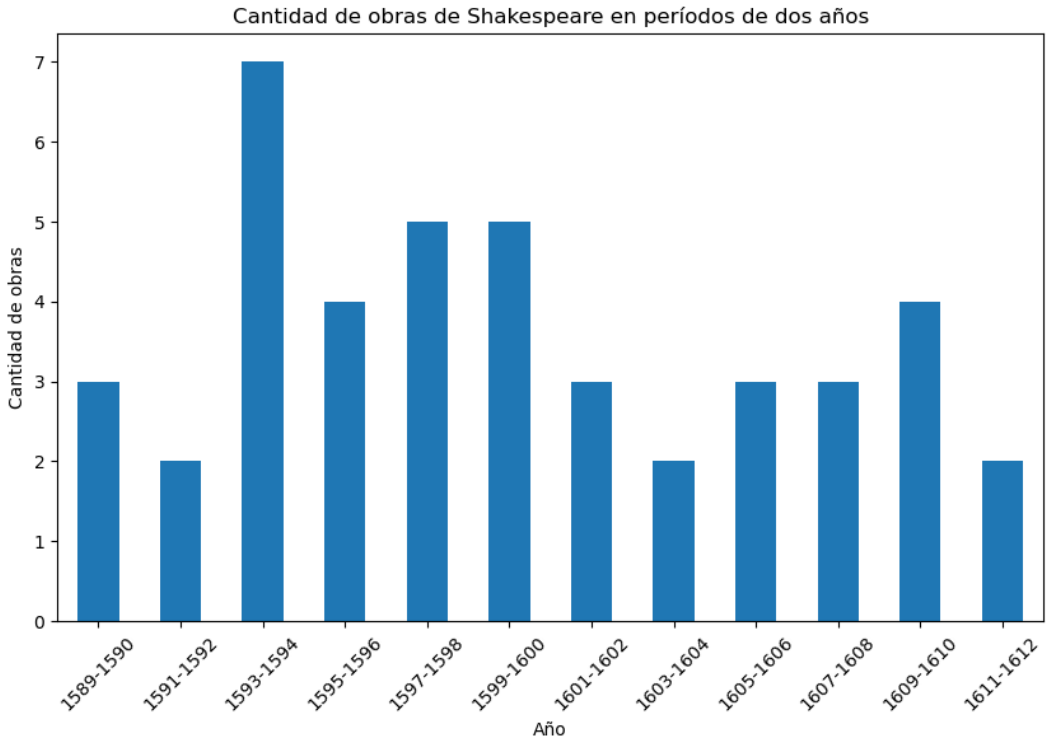

Figura 2: Cantidad de obras de Shakespeare cada 2 años.

Si se observa la naturaleza de sus obras (Figura 3) hay una clara tendencia a producir comedias, tragedias y obras históricas, con una menor proporción de poemas y sonetos.

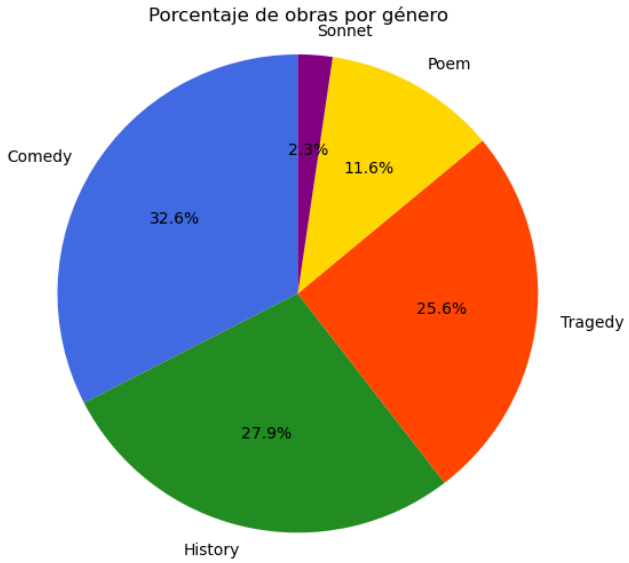

Figura 3: Porcentaje de obras según el género.

Al observar el género de su producción, resulta interesante identificar tendencias en sus obras en el correr de los años. Esto se puede observar en la Figura 4, en donde resulta claro que la producción de comedias fue constante a lo largo de su carrera, mientras que las obras históricas se concentran en los primeros y últimos años. En los años que disminuye su publicación de estas obras se ve una concentración en la producción de tragedias (entre 1599 y 1608). Respecto a los poemas, no parece haber una tendencia clara, mientras que sólo escribió un soneto en toda su carrera.

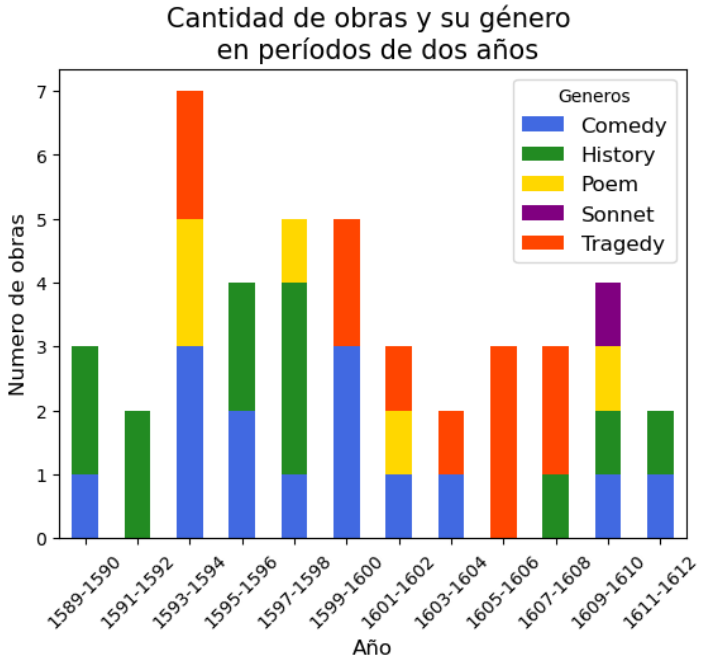

Figura 4: Obras agrupadas cada 2 años y género.

Respecto al léxico de su obra, se buscaron las 10 palabras más comunes de toda su producción. No se observó ninguna que pueda aportar información adicional o relevante, por lo que se decide quitar de la búsqueda las palabras denominadas stop Word (Figura 5).

La aparición de títulos nobiliarios en las palabras más usadas nos hace suponer que se está utilizando un lenguaje formal y un contexto histórico, pero si no tuviéramos ningún tipo de conocimiento previo al análisis podríamos suponer que es una ambientación histórica, una narrativa de fantasía, un lenguaje formal, un lenguaje arcaico o simplemente un estilo más elaborado y cortés.

Figura 5: Top 10 palabras sin stop words de toda la obra de Shakespeare.

Un siguiente estudio de palabras podría ser qué combinación de 2 palabras aparecen más seguido. Viéndolas aisladas nos hace suponer que good puede ser seguida por lord o sir.

Para obtener una idea del lenguaje utilizado por género de las obras, se podría realizar una visualización de barras horizontales como la de la figura 11, pero con una barra por cada género en cada palabra. De la misma forma, se podría mostrar la palabra más utilizada en las obras escritas en cada año, de forma de poder evidenciar un cambio en el estilo del lenguaje.

Respecto a la visualización de las palabras más usadas por personaje, entendemos que una gráfica de CloudWord por personaje nos podría llegar a dar una idea del rol que ocupan los 10 personajes con mayor cantidad de palabras.

Por otro lado, si vemos la cantidad de palabras por personajes (Figura 6), se vuelve a dar el error derivado de la utilización polifuncional de la tabla characters. Se observa en el puesto 1 y 2 Poet y Stage Directions que no son personajes.

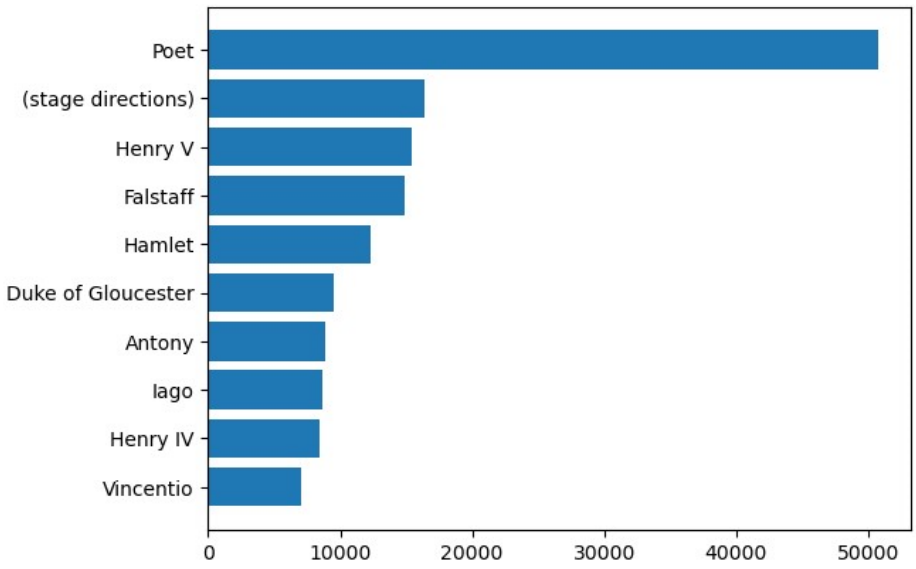

Figura 6: Cantidad de palabras por personaje

Una posible solución es filtrar todos los párrafos donde character_id correspondan al identificar de Poet o Stage direction. Esta solución eliminaría el texto de los poemas. Otra solución sería buscar una gramática en el texto de los poemas que nos permita obtener los personajes (si es que existen), agregarlos a la tabla characters y asignar el párrafo al personaje correspondiente.

A raíz de este análisis preliminar, algunas preguntas que se podrían llegar a responder son:

1) Personajes con mayor cantidad de interacciones entre sí:
   Tendríamos que buscar una gramática que nos permita identificar con qué personaje está interactuando, mapear el personaje de la tabla characters, recorrer la tabla paragraph y sumar 1 en el contador correspondiente al par de personajes.

2) Personajes que aparecen en más de una obra:
   En este caso excluimos a los poemas dado que no existe una relación directa entre párrafo y personaje. Se haría un join entre paragraph con chapter agrupados por work_id y character_id

3) Obras con mayor intervención del director:
   Se podría observar con una gráfica de barra del resultado de la agrupación entre párrafos y capítulos donde character_id=1261.

4) Tendencias en la directrices del director:
   Se haría un análisis exploratorio entre palabras más usadas, frases y/o conjuntos de palabras utilizadas por el personaje con id=1261.
   Para profundizar en este tema se podría agrupar por año o género o ambos.
   La visualización depende va a depender de lo complejo en identificar tendencias o de la tendencia detectada.

Clasificación de personajes.

Dada la naturaleza de los datos y la información disponible de cada personaje, en esta sección buscaremos entrenar algoritmos de aprendizaje automático para la clasificación de algunos de ellos. Para ello partimos de la tabla *Paragraphs* después de haber hecho una limpieza de caracteres. Por motivos de cómputo, se decidió tomar los párrafos de tres personajes en particular: Queen Margaret (*Richard III, Henry IV*), Antony (*Antony y Cleopatra, Julio Cesar*) y Cleopatra (*Antony y Cleopatra*). Luego de filtrada la tabla, la cantidad de párrafos por personaje es la siguiente: Antony: 253, Cleopatra: 204 y Margaret: 169.

El primer paso fue realizar la partición entre la muestra de entrenamiento y la muestra de evaluación. Para esto se tomó como variable predictiva los párrafos de cada personaje y el nombre del personaje como la variable de respuesta. Para realizar la partición se utilizó la función `train_test_split()` del paquete scikit-learn de python, indicando que la partición fuera del 30% de los datos para la muestra de evaluación y que mantuviera las proporciones de entrenamiento y evaluación de cada personaje (con el argumento `stratify=y`). En la Figura 7 se puede observar el porcentaje de párrafos de la muestra de entrenamiento y evaluación manteniéndose con respecto a la cantidad original.

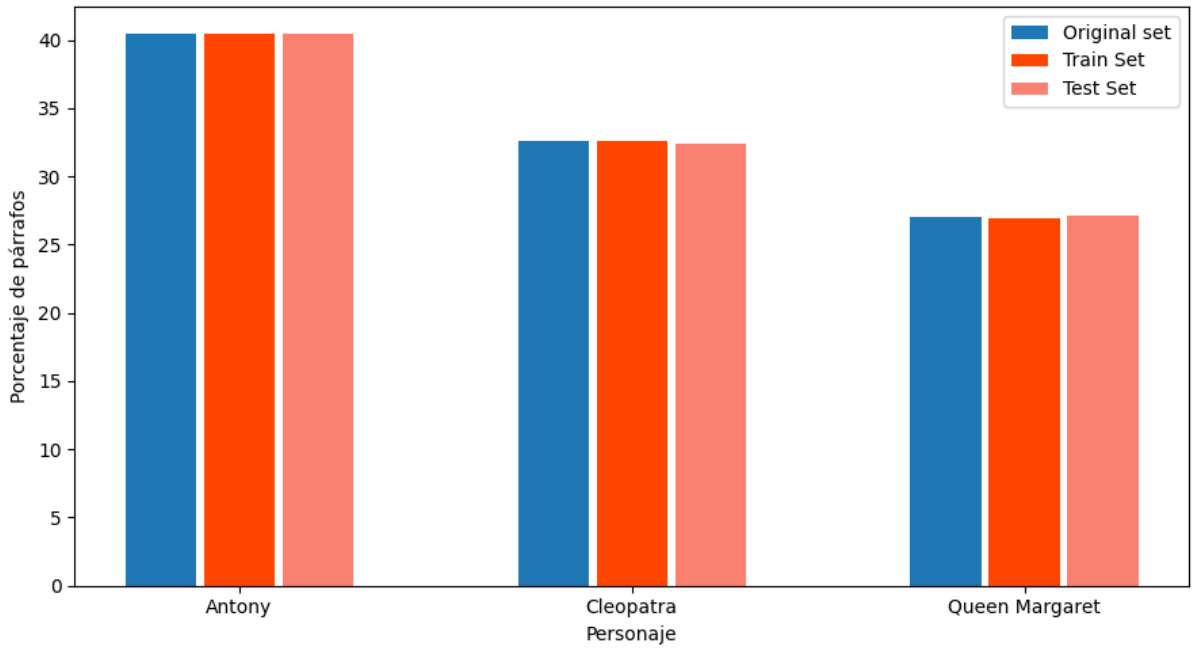

Figura 7: Porcentaje de párrafos en las muestras de entrenamiento (train set) y evaluación (test set) con y sin estratificar.

Dado que los párrafos están compuestos por palabras, es necesario convertir el texto en una representación numérica. Para esto utilizamos la técnica *bag of words (BOW)* la cual utiliza tokenización y conteo de palabras contemplando únicamente la cantidad de ocurrencias sin importar la posición relativa de ellas. El proceso de tokenización es la acción de asignarle un valor numérico a cada palabra, y el conteo es el número de apariciones de cada palabra por documento.

Ejemplificando, tomamos el primer párrafo de nuestro dataset: “`let it alone let s to billiards come charmian`”. Una posible tokenización de este párrafo podría ser “`11 22`

`33 11 44 55 66  77  88”` (obsérvese que el número 11 se repite porque corresponde a *let* que se repite en el primer documento). Luego se realiza el conteo de palabras. Para cada párrafo se genera un arreglo de *n* dimensiones, siendo *n* la cantidad palabras (tokens) diferentes en todo nuestro dataset. Para cada posición del arreglo, se coloca la cantidad de veces que aparece ese token en el párrafo. En el dataset de entrenamiento tenemos 438 documentos (párrafos) que se componen por 2821 tokens. Esto nos da 438 arreglos de 2821 posiciones resultando en una matriz de 438 x 2821.

La gráfica resultante del arreglo del ejemplo se puede ver en la Figura 8. El eje de las *x* representa los tokens distintos (tokens del conjunto de documentos de entrenamiento) y el eje de las *y* la cantidad de veces que aparece en nuestro documento de ejemplo.

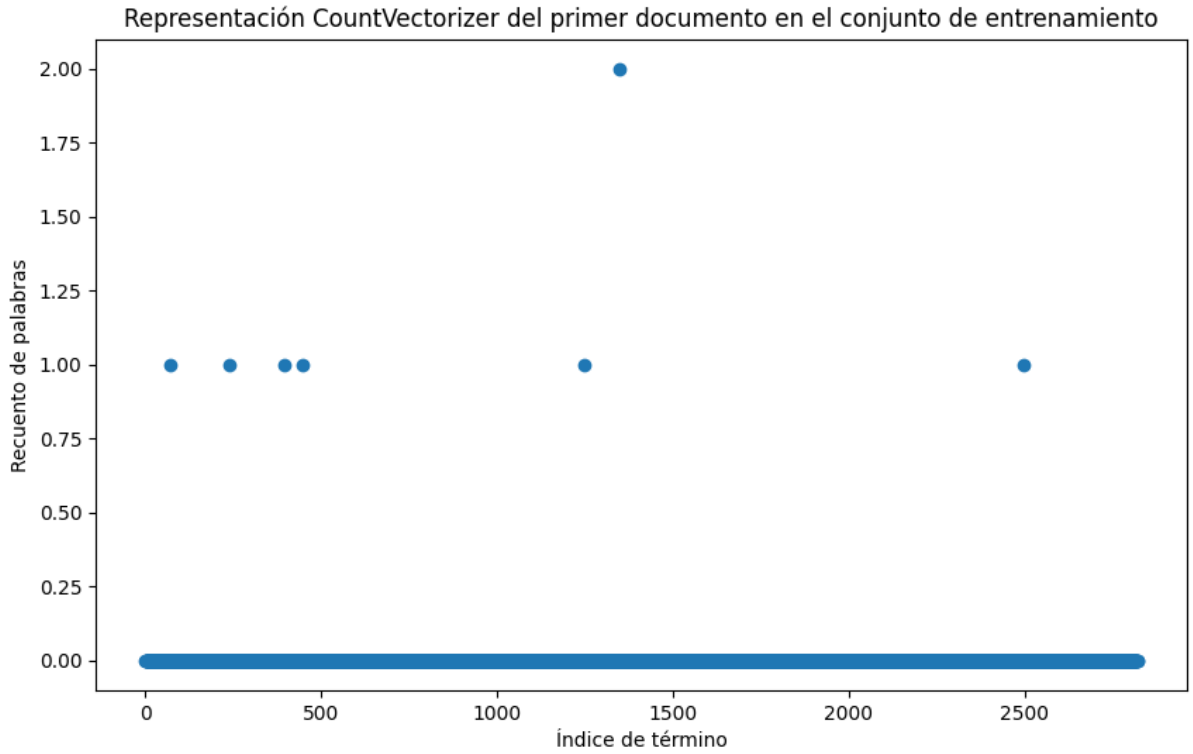

Figura 8: Gráfica de primer vector.

Como se ve en la gráfica, la mayor cantidad de tokens tienen asignado el valor 0. Si hacemos una visualización para cada arreglo se repite el comportamiento. Por lo cual podemos afirmar que es una matriz de donde la mayor parte de sus elementos son 0s, lo cual coincide con la definición de matriz dispersa.

La tokenización de palabras de un documento es un caso particular de tomar n-gramas con n=1. Los n-grama son secuencias de n elementos consecutivos tomados de una secuencia más larga (como una oración o párrafo). Si tomamos como ejemplo “`let it alone let s to billiards come charmian”` un bi-grama podría ser “`let it”`. Cabe mencionar que en los n-gramas el orden de los elementos importa. Dicho con un ejemplo, “`let it”` es distinto que `“it let”`.

La matriz resultante de BOW es n*m, donde n es la cantidad de documentos y m es la cantidad de elementos distintos de todos los documentos. Esto muestra que es una técnica

no escalable dado que si pensamos en grandes volúmenes de datos la matriz resultante puede llegar a tamaño que son imposible de cargar en memoria.

Para que la importancia relativa de un término (palabra o ngrama) no sea únicamente la cantidad de veces que aparece en el documento se utiliza la técnica Term Frequency - Inverse Document Frequency (TF-IDF). TF-IDF mide la importancia relativa de un término en un documento dentro de una colección de documentos. Para calcularlo utiliza la siguiente fórmula:

$$TFIDF = TF * IDF \qquad ec.1$$

Donde TF es la frecuencia de término (ec. 2) y mide la importancia relativa de un término en un documento; e IDF es la frecuencia inversa de un documento (ec. 3) y mide la rareza de un término en el conjunto de documentos.

$$TF = \frac{Cant.\ de\ ngramas}{total\ de\ ngrams} \qquad ec.2$$

$$IDF = log(\frac{Nro\ de\ documentos}{Nro\ de\ documentos\ que\ tienen\ el\ término}) \qquad ec.3$$

Por consiguiente, cuanto mayor sea el número dado por TF-IDF, mayor es la importancia relativa para el documento. Para aplicar esto a nuestro subset de datos, utilizamos las funciones `CountVectorizer(stop_words=None, ngram_range=(1,1))` y `TfidfTransformer(use_idf=False)` de scikit-learn. Los argumentos dentro de las funciones permiten no utilizar stop words, ngramas de una palabra y la transformación la hace sin la ponderación de idf.

Luego de obtener esta representación numérica de la muestra de entrenamiento, realizamos un análisis de componentes principales (PCA) para evaluar si era posible separar a estos tres personajes graficando únicamente las dos primeras componentes principales (ver Figura 9). Para esto se utilizó la función `PCA()` de scikit-learn.

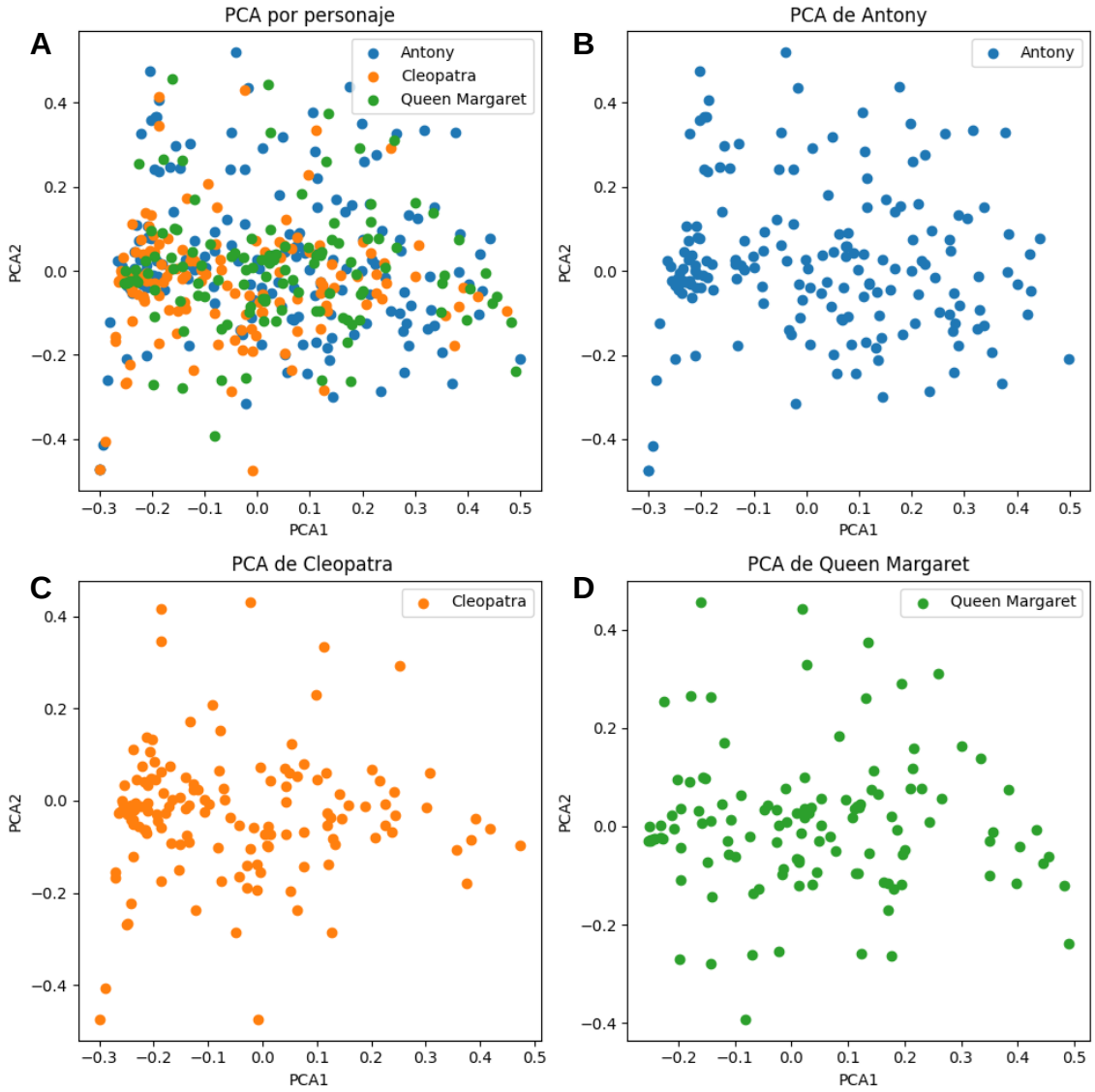

Figura 9: Análisis de componentes principales (PCA) utilizando los párrafos vectorizados de Antony(B), Cleopatra(C) y Queen Margaret (D).

Observando las dos primeras componentes principales, no se ve un patrón característico de alguno de los personajes (Figura 9A). Se observa que los puntos se acumulan entre -0.2 y 0.2 en el eje de las ordenadas con una mínima disminución de la densidad a medida que aumenta el valor de la abscisa y una distribución pareja entre Cleopatra y Queen Margaret, observándose una mayor dispersión en Anthony. Esto indica que no se obtiene una distinción clara de los personajes a partir de sus párrafos.

Ante estos resultados, se volvió a aplicar una PCA, pero esta vez alterando los argumentos de la etapa de vectorización, permitiendo que se quiten las stop words en inglés, la posibilidad de utilizar bigramas e idf. El resultado se muestra en la Figura 10.

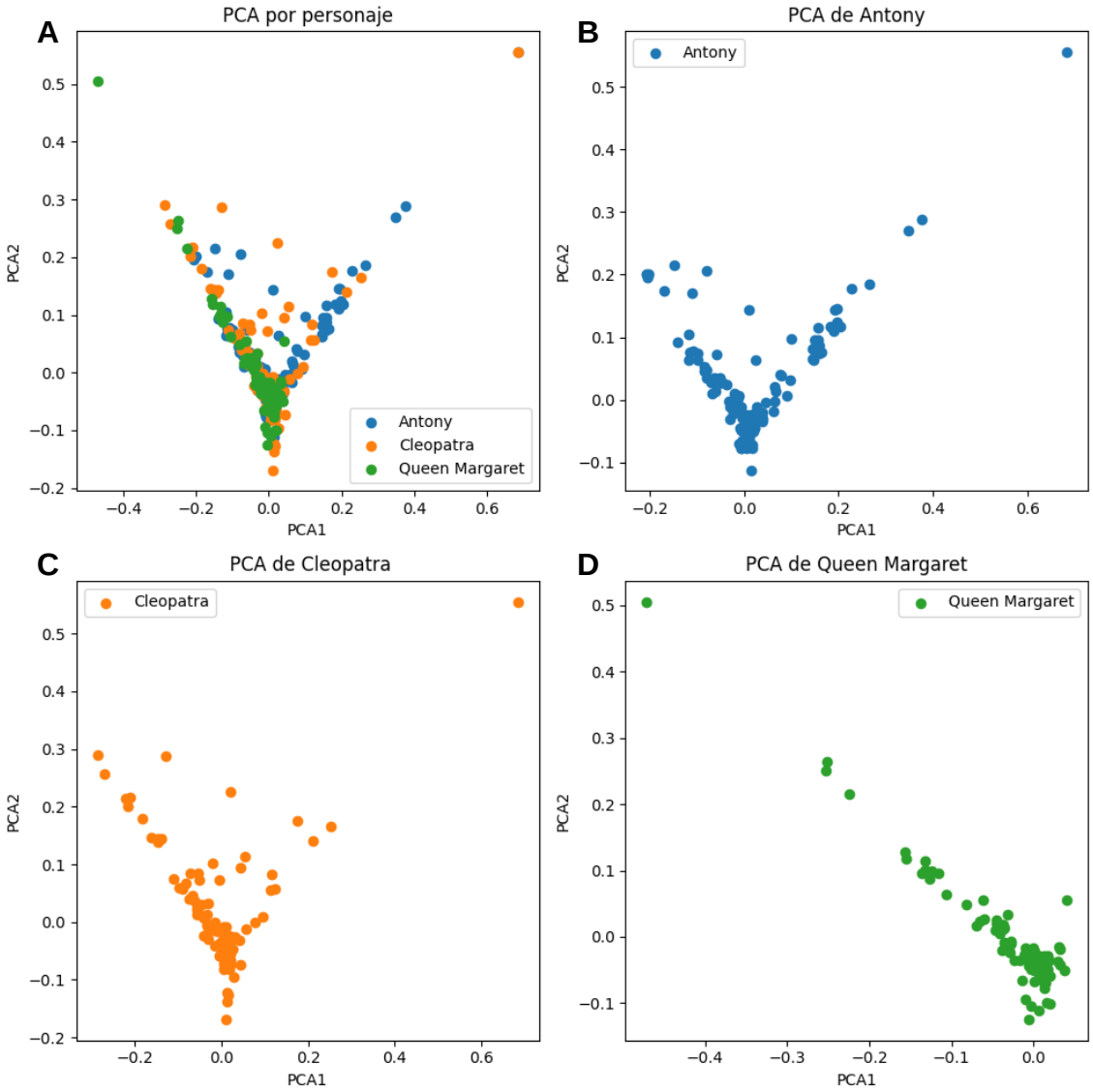

Figura 10: Análisis de componentes principales (PCA) utilizando los párrafos vectorizados de Antony(B), Cleopatra(C) y Queen Margaret (D). Para vectorizar los párrafos se quitaron las stop words, se permitió la ponderación de idf y bigramas.

La Figura 10 refuerza la idea de que los tres personajes comparten cierta identidad en los diálogos que poseen. Esta vez, sin embargo, se observa una zona en donde únicamente se encuentran puntos correspondientes a Antony y Cleopatra, lo cual puede deberse a que provienen de la misma obra. En cambio, no existe una región del espacio en donde únicamente se encuentre Queen Margaret, lo cual llama la atención debido a que el personaje aparece en 4 obras.

La muestra de entrenamiento está en la dimensión 2591, en donde con las PCAs de las figuras 15 y 16 estamos reduciendo el dataset a dos dimensiones. Es posible que en esta reducción se pierda información importante para la separación de clases. Una forma de verificar esto es graficar la varianza explicada de la PCA. La varianza explicada se refiere a la proporción de la varianza total de los datos que es explicada por cada una de las componentes principales. Se calcula como el cociente entre la varianza de esa componente y la varianza total de los datos originales. Indica qué tan bien esa componente representa la

variabilidad presente en los datos. En la Figura 11 se muestra la varianza explicada a partir de la PCA de la figura 16.

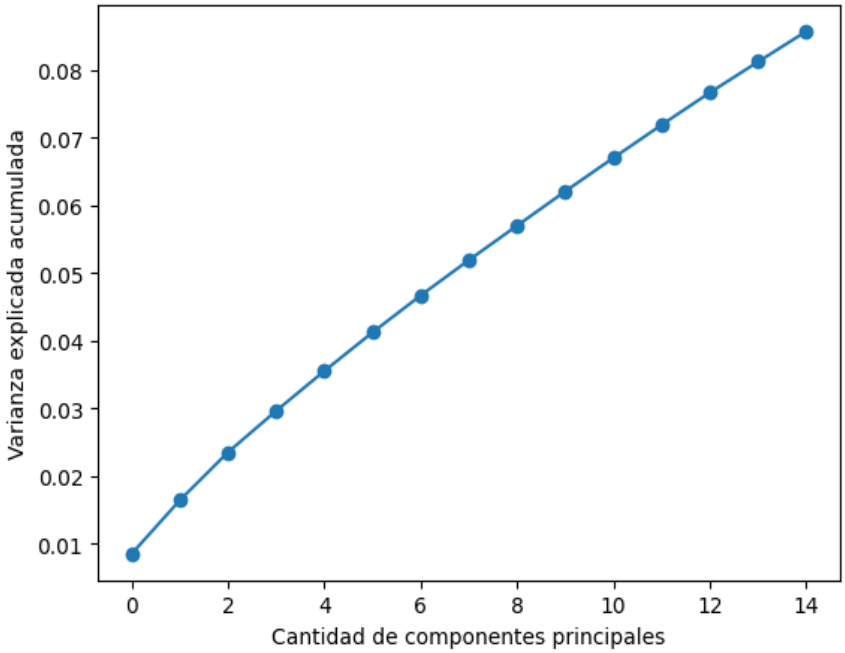

Figura 11: Varianza explicada acumulada. Se utilizó la muestra de entrenamiento vectorizada de la Figura 10.

Como se puede observar en la Figura 11, las dos primeras componentes principales explican menos del 30% de la varianza. Esto puede explicar la dificultad de identificar personajes utilizando únicamente las dos primeras componentes principales.

Dado que una reducción de la dimensionalidad por PCA no es suficiente para distinguir estos tres personajes, se aplicaron algoritmos de aprendizaje automático supervisado para clasificarlos. El primer modelo utilizado fue Multinomial Naive Bayes (MNB), algoritmo de clasificación basado en el teorema de Bayes y diseñado específicamente para trabajar con características discretas o categóricas. Para esto se utilizó la función `MultinomialNB()` con sus valores predeterminados. La matriz de confusión resultante se muestra en la Figura 12. Se observa que el modelo es muy bueno para predecir la etiqueta de los datos de entrenamiento. Sin embargo, en la Figura 12B se observa que al predecir con datos nunca mostrados al modelo, cataloga a la mayoría de los párrafos como pertenecientes a Antony. Esto indica que el modelo no es muy bueno a la hora de distinguir entre las tres clases.

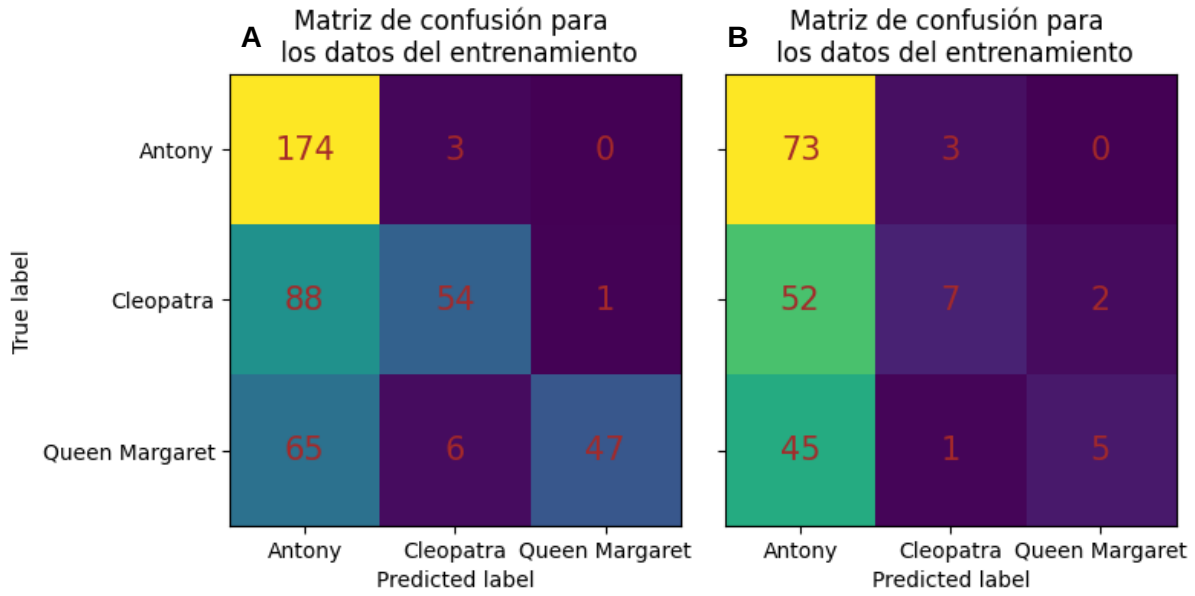

Figura 12: Matriz de confusión para un algoritmo MNB entrenado con valores default y testeado con la muestra de entrenamiento (A) y la muestra de evaluación (B).

Una forma de evaluar el rendimiento de un modelo de predicción es utilizar las métricas de precision, recall y el score F1. La precisión evalúa la proporción de casos positivos que fueron correctamente identificados por el modelo en relación con el total de casos clasificados como positivos, se enfoca en la calidad de las predicciones positivas. El recall es la proporción de casos positivos que fueron correctamente identificados por el modelo en relación con el total de casos reales positivos, se enfoca en la capacidad del modelo para encontrar todos los casos positivos. Finalmente, el score F1 es una métrica que combina el precision y recall en un solo valor, utilizando la media armónica de ambos. La media armónica da más peso a los valores más bajos, por lo que el F1-score penaliza los casos en los que el precision y recall difieren considerablemente. El F1-score proporciona una medida equilibrada entre precision y recall.

Cuando se trata de clasificación multiclase, se pueden calcular las métricas de precision, recall y F1-score para cada clase individualmente y se toman ciertas consideraciones para evaluar el rendimiento global del modelo. En nuestro caso, que tenemos un desequilibrio de clases, se suele utilizar el promedio ponderado (weighted-average), el cual pondera cada clase según su proporción en los datos de evaluación, lo que significa que las clases más grandes tienen un impacto mayor en el promedio ponderado. Los resultados de estas métricas para el modelo MNB se muestran en la Tabla 2.

Tabla 2: valores de precision, recall y f1-score para el modelo MNB.

Etiqueta	Precision	recall	F1-score
Antony	0.43	0.96	0.59
Cleopatra	0.64	0.11	0.19
Queen Margaret	0.71	0.1	0.17
Promedio ponderado	0.57	0.45	**0.35**
Accuracy	0.45		

Como se puede observar en la Tabla 2, el promedio ponderado del score F1 es de 0.35. Para mejorar esta métrica se optimizaron los hiper-parámetros del algoritmo `MultinomialNB()` a través de técnicas de validación cruzada (cross-validation). La idea principal detrás de esta técnica es utilizar diferentes particiones o divisiones de los datos disponibles para entrenar y evaluar el modelo de manera repetida. En lugar de realizar una única división de los datos en conjuntos de entrenamiento y prueba, la validación cruzada realiza múltiples divisiones, lo que permite obtener una estimación más confiable del rendimiento del modelo. En nuestro caso, los hiper-parámetros utilizados fueron: el uso de stop-words, la posibilidad de utilizar idf y el uso de diferentes tamaños de n-gramas. Los resultados se muestran en la Figura 13.

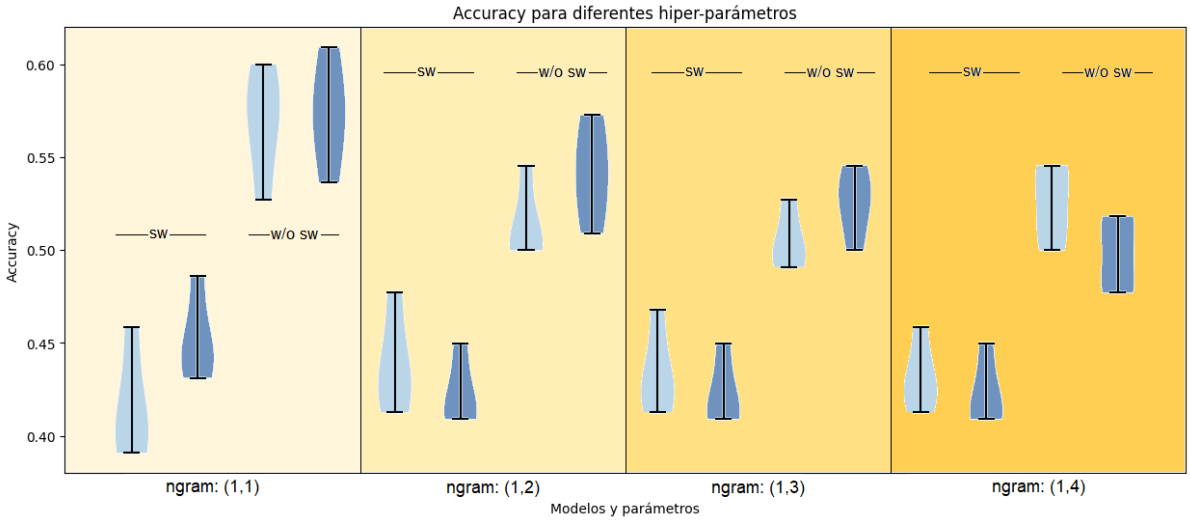

Figura 13: variabilidad del accuracy en los diferentes parámetros evaluados usando un cross validation de 4-fold. En azul claro se muestran los resultados sin el uso de idf, mientras que en azul oscuro se hizo uso de esta técnica.

Se observa en la Figura 13 que al quitar las stop words mejora notablemente el accuracy de los modelos, a diferencia de lo que ocurre al aumentar la cantidad de n-gramas en donde el desempeño del modelo disminuye. Respecto al uso de idf, no parece existir una tendencia clara. Vistos estos resultados, decidimos utilizar los parámetros que obtenían un mejor accuracy: "stop_words": "english", "ngram": (1,1), "idf": False.

Para comprobar que existe una mejora, se realiza nuevamente el entrenamiento del algoritmo con la combinación de parámetros que dan lugar al mejor accuracy obteniendo la siguiente matriz de confusión (Figura 14) y métricas (Tabla 3). En ella se observa que aún existe un bajo porcentaje de aciertos. Esto se debe a las limitaciones de bag of word y de tf-idf como ignorar el contenido semántico, sobrevalorar palabras comunes que no aportan características del personaje, la no existencia de palabras en el conjunto de entrenamiento e ignorar el orden de las palabras.

Tabla 3: Valores de precision, recall y f1-score para el modelo MNB optimizado.

Etiqueta	Precision	recall	F1-score
Antony	0.54	0.93	0.69
Cleopatra	0.74	0.46	0.54
Queen Margaret	0.91	0.39	0.55
Promedio ponderado	0.71	0.62	**0.60**
Accuracy	0.62		

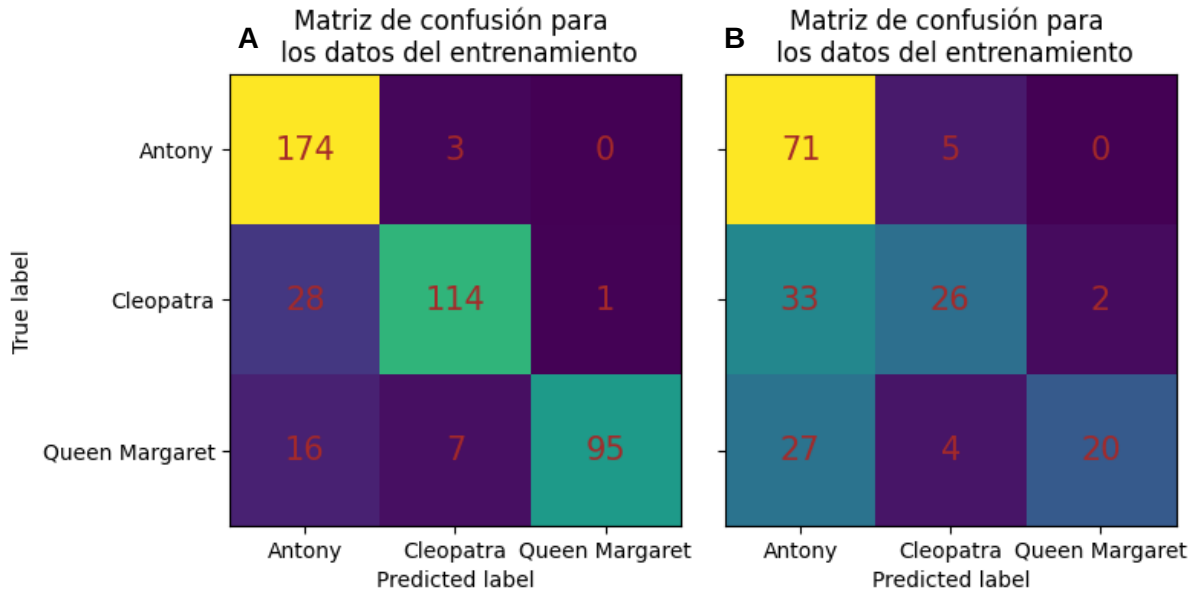

Figura 14: Matriz de confusión para un algoritmo MNB entrenado con los argumentos: "stop_words": "english", "ngram": (1,1), "idf": False. Fue testeado con la muestra de entrenamiento (A) y la muestra de evaluación (B).

Considerando que los resultados obtenidos tras el ajuste y entrenamiento del modelo no fueron satisfactorios, se decidió explorar y evaluar otras opciones de modelos. Entre los posibles modelos que ofrece scikit-learn y el estudio de componentes realizado, se estableció que el modelo a elegir debe poder trabajar con alta dimensionalidad. Por tales motivos se decidió evaluar Support Vector Machine (SVM) y Random Forest (RF).

Support Vector Classifier (SVC) es un modelo de aprendizaje supervisado que se encuentra en la biblioteca SVM y se usa para tareas de clasificación (en nuestro caso estamos clasificando párrafos por personaje). SVC busca encontrar uno o varios hiperplanos óptimos para separar las diferentes clases de datos en un espacio dimensional superior. Para lograrlo usa los vectores definidos por los puntos de las clases que estén más cerca (vector de soporte). En el caso ideal donde las clases sean linealmente separables, el hiperplano en 2 dimensiones sería una recta, en 3 dimensiones un hiperplano y en n dimensiones n hiperplanos.

Para los casos que no es posible separar las clases linealmente, SVC ofrece diferentes parámetros que permiten ajustar el modelo para que encuentre nuevas figuras o sacrifique datos para mejorar las predicciones. Entre los parámetros que ofrece, creemos que es importante destacar el kernel dado que determina la forma en que se mapean los datos a un espacio dimensional superior. Las posibles formas que ofrece el kernel son lineal, polinómico, radial y/o sigmoidal y cada una de ellas tienen sus propios parámetros de ajustes.

Random forest es un modelo predictivo que puede ser usado para regresión lineal o clasificación. En nuestro caso vamos a usar el modelo `RandomForestClassifier()` (caso de clasificación) de la librería sklearn.ensemble. La técnica que utiliza es dividir el conjunto de datos en muestras aleatorias para crear un árbol de decisión por cada división. Luego de tener todos los árboles creados (bosque), utiliza el promedio en caso de regresión o por votación de mayorías en caso de clasificación para decidir la predicción. En nuestro caso se generó una matriz de decisión de 438x3 donde cada fila representa a cada párrafo y las columnas a cada personaje (Antony, Cleopatra y Queen Margaret). Supongamos que cada

columna corresponde a Antony, Cleopatra y Queen Margaret respectivamente. Si el algoritmos tuviera que predecir la clasificación del párrafo 5 diría que es Queen Margaret dado que es el que tiene mayor promedio (un 0.69 % de los árboles) que decidieron por Queen Margaret.

Al igual que SVM, Random forest también permite establecer diferentes parámetros como el criterio, profundidad de los árboles de evaluación, cantidad de árboles en el bosque, etc. Para no realizar manualmente la búsqueda de los parámetros óptimos se utilizó la función GridSearchCV de la librería sklearn.model_selection en ambos modelos.

Entre los modelos entrenados no se observa ninguno que se destaque frente al otro obteniendo tasas de error similares en cada uno. En la Figura 15 muestra la matriz de confusión de cada modelo.

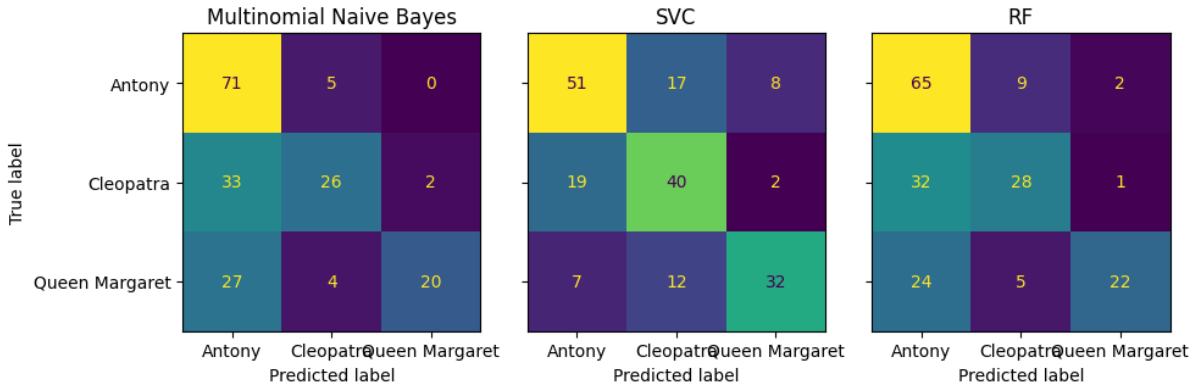

Figura 15: Matrices de confusión para tres modelos: Multinomial Naive Bayes (A), Support Vector Classifier (SVC) (B) y Random Forest (RF) (C).

Una vez analizado el comportamiento de los modelos, se cambia un personaje para que los datos sean desbalanceados. En nuestro caso cambiamos a Anthony por Falstaff logrando que el 55,8% del dataset pertenezca a párrafos de Falstaff, el 24,2 % a Cleopatra y el 20% Queen Margaret. Para que el análisis sea lo más fiel posible se realizó el mismo preprocesamiento de datos realizado en la etapa previa. Al realizar el gráfico PCA con 2 componentes se obtiene las siguientes gráficas (ver Figura 16).

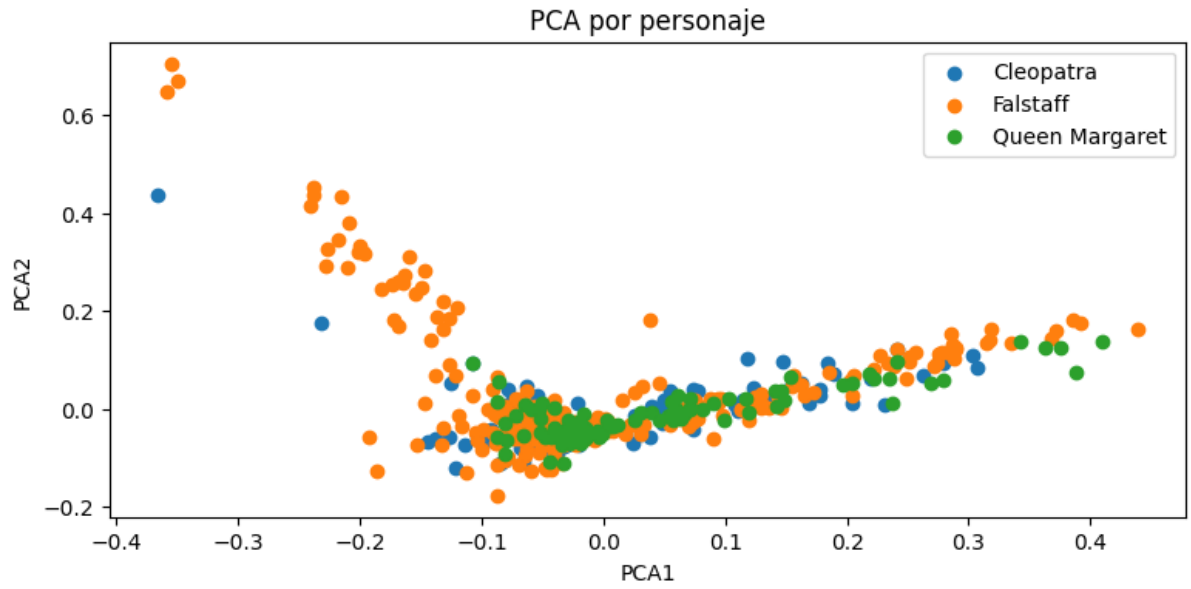

Figura 16: Análisis de componentes principales (PCA) utilizando los párrafos vectorizados de Falstaff, Cleopatra y Queen Margaret.

Al igual que en el análisis anterior, no se observa un patrón con el que se pueda identificar a algún personaje. Si observamos la distribución de los datos por personaje vemos que forman la misma figura (solapamiento de clases) con mayor o menor densidad. Esto hace imposible obtener una separación de las clases con 2 componentes.

Para verificar la eficacia de los modelos se entraron los modelos con el nuevo conjunto de datos arrojando los siguiente resultados (ver Figura 17)

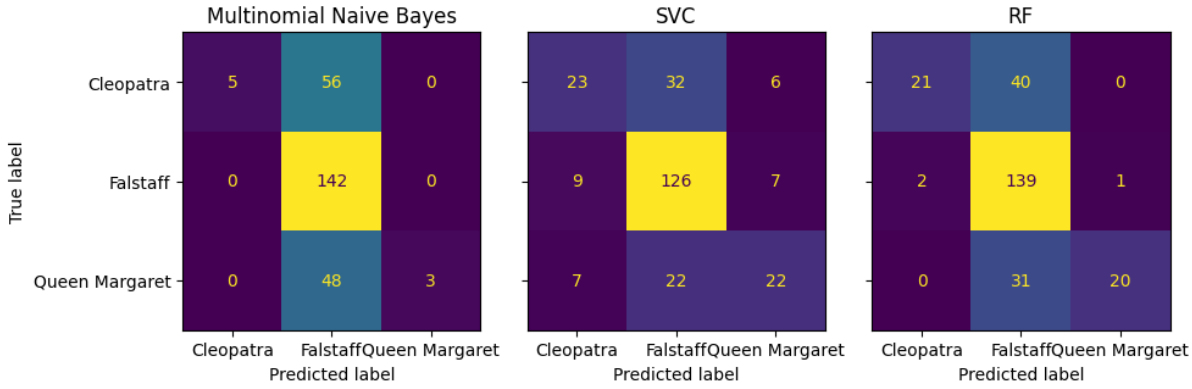

Figura 17: Matrices de confusión para tres modelos: Multinomial Naive Bayes (A), Support Vector Classifier (SVC) (B) y Random Forest (RF) (C).

Si comparamos las matrices, observamos que el 97 % de las predicciones de Naive Bayes corresponden a Falstaff (Tabla 4), marcando una clara tendencia a inclinarse por la clase de mayor cardinalidad siendo este el peor modelo de los evaluados cuando existe un desbalance en los datos. Si comparamos SVC con RF no podemos afirmar que uno tenga mejor comportamiento que otro dado que en términos generales ambas predicciones son similares (métricas en anexo). Dicho esto, todos los modelos marcan una clara tendencia a inclinarse por la clase (párrafos por personaje) de mayor cardinalidad (Falstaff).

Tabla 4: Porcentaje de predicciones que los tres modelos deciden por Falstaff.

Modelo	% de predicciones por Falstaff
NB	97 %
SVC	72 %
RF	87 %

Para evitar el sesgo causado por el desbalance se utilizan dos enfoques. Uno es el sobremuestreo y el otro es el submuestreo. El objetivo es balancear las cargas de las clases.

El submuestreo consiste en seleccionar un subconjunto de la clase mayoritaria para que sea de un tamaño preestablecido. Esta técnica lleva a una pérdida de datos pudiendo haber eliminado datos relevantes, dejando datos homogéneos con pocas características en común. No obstante, existen varios algoritmos para reducir este riesgo como separar en grupos los datos y seleccionar una muestra representativa de cada grupo (cluster), utilizar la menor distancia promedio de los k-vecinos más cercanos (NearMiss) para seleccionar los datos, etc.

El sobremuestreo consiste en rellenar las clases de menor tamaño para que sean de un tamaño preestablecido. Algunos de los algoritmos para realizarlo son duplicar los datos de forma aleatoria para que las clases más pequeñas sean de un tamaño preestablecido, relleno por interpolación (SMOTE), etc.

Luego de haber entrenado varios modelos a partir de bag of word y no haber encontrado resultados satisfactorios, podemos llegar a pensar que necesitamos alguna técnica de representación de palabras que nos permita entrenar modelos capaces de interpretar la semántica y/o el contexto de las palabras. Una técnica que nos lo permite es Word Embeddings. Consiste en representar las palabras o frases en vectores.

Una implementación de embedding puede ser Word2Vec. Se basa en que cada palabra tiene un significado según su contexto. Para lograrlo crea una representación donde las palabras con contextos similares están cerca, y las palabras relacionadas tienen una relación de distancia y sentido en el espacio vectorial. Por ejemplo, el vector formado por el punto que representa la palabra gato y la palabra gatos debe tener igual sentido y longitud que el vector formado por los puntos que representan las palabras perro y perros.

Para crearlo utiliza un modelo predictivo no supervisado de redes neuronales que puede ser entrenado para predecir una palabra según su contexto o predecir un contexto según una palabra. A estas técnicas se les llama Continuous Bag of Words (CBOW) y Skip-gram respectivamente.

El modelo es entrenado por un algoritmo que utiliza una ventana deslizante de n elementos que se mueve a lo largo del corpus (todos los documentos) para entrenar una red neuronal. Si lo que estamos buscando es predecir una palabra según su contexto (CBOW), el algoritmo va a iterar el corpus hasta lograr una tasa de predicción aceptable para la palabra que ocupa la posición central de cada ventana deslizante.

Si lo que buscamos es Skip-gram la red neuronal va a ser entrenada de la misma manera pero va a intentar predecir la siguiente palabra o la anterior (contexto) de la que se encuentra en la posición central de la ventana deslizante.

Esperaría que con el embedding de Word2Vec se pueda identificar mejor los grupos de palabras que lo que se identifican con BOW. Si esto es así, al convertir el embedding en un formato reconocible por los modelos probados (NB, SVM y RF) y volver a entrenarlos, la tasa de acierto debería ser mejor que con BOW.

No solo Word2Vec utiliza word embedding sino que existen otros que también lo utilizan como BERT, GloVe, Fasttext, etc. Cada uno de ellos tiene su propia implementación de embedding. Fastext en particular puede ser utilizado para clasificación. En este caso vamos a comparar Fastext contra los modelos entrenados con BOW.

Al comparar FasText con Naive Bayes (Figura 18) vemos una mejora significativa en la precisión de predicciones que corresponde a Cleopatra y a Queen Margaret y disminución en la precisión respecto a Antony. Sin embargo, la tasa general es ligeramente superior.

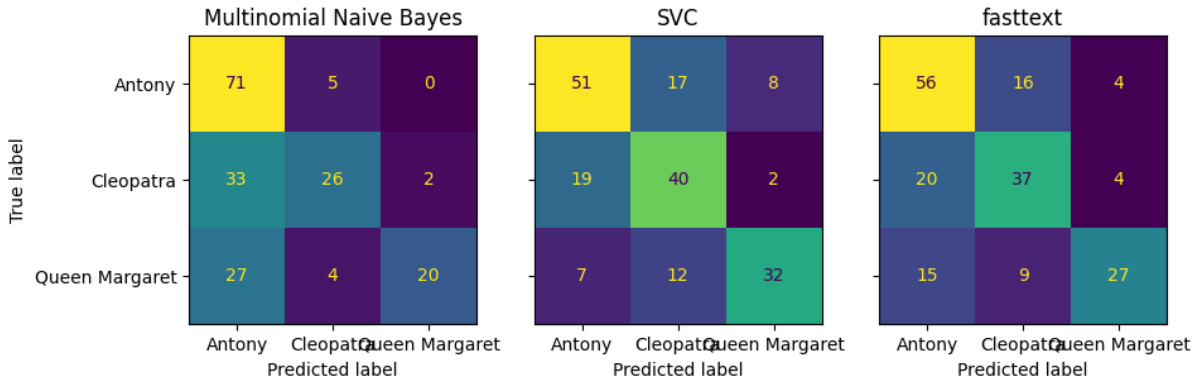

Figura 18: Comparación del modelo FastTree con Multinomial Naive Bayes y Support Vector Classifier para predecir los personajes Antony, Cleopatra y Queen Margaret.

Al repetir las pruebas con el modelo desbalanceado (Figura 19) vemos una clara mejoría respecto a Naive Bayes y un rendimiento muy parejo con SVC.

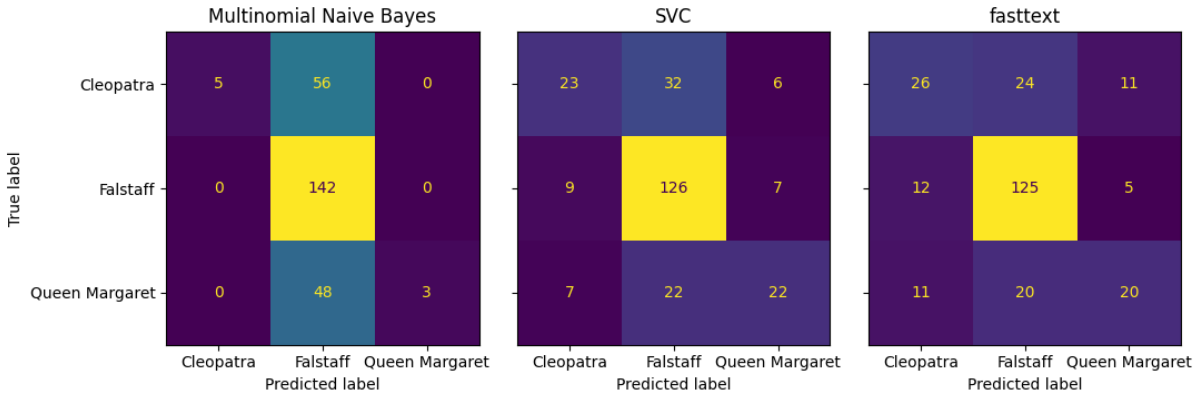

Figura 19: Comparación del modelo FastTree con Multinomial Naive Bayes y Support Vector Classifier para predecir los personajes Falstaff, Cleopatra y Queen Margaret.

Según las comparaciones, se observa que dentro de los modelos probados el que mejor tasa de predicción es FastText sin lograr alcanzar resultados muy destacables frente a SVC. Como

ventaja vemos que la tasa de efectividad de FastText con datos desbalanceados es similar a la que tiene con datos balanceados. En cambio, es necesario preparar los datos (ej quitar stop word, símbolos de puntuación, etc) antes de entrenar el modelo.

Anexo

```
In [11]: #Visualización paragraphs
         df_paragraphs
```

Out[11]:

	id	ParagraphNum	PlainText	character_id	chapter_id
0	630863	3	[Enter DUKE ORSINO, CURIO, and other Lords; Mu...	1261	18704
1	630864	4	If music be the food of love, play on;\nGive m...	840	18704
2	630865	19	Will you go hunt, my lord?	297	18704
3	630866	20	What, Curio?	840	18704
4	630867	21	The hart.	297	18704
...	...	...	...	...	...
35460	666323	3460	That she is living,\nWere it but told you, sho...	866	19648
35461	666324	3467	You gods, look down\nAnd from your sacred vial...	584	19648
35462	666325	3475	There's time enough for that;\nLest they desir...	866	19648
35463	666326	3483	O, peace, Paulina!\nThou shouldst a husband ta...	667	19648
35464	666327	3504	[Exeunt]	1261	19648

Figura S 1: Exploración de tabla Paragraphs sin filtros

Revisión de df_characters con id 1261

```
df_characters.loc[df_characters['id'] == 1261]
```

	id	CharName	Abbrev	Description
1260	1261	(stage directions)	xxx	NaN

Figura S 2: Datos del personaje con id 1261

Revisión de df_characters con Description null o vacía

```
df_characters.loc[df_characters['Description'].isnull() | (df_characters['Description'] == "")]
```

	id	CharName	Abbrev	Description
0	1	First Apparition	First Apparition	
1	2	First Citizen	First Citizen	
2	3	First Conspirator	First Conspirator	
3	4	First Gentleman	First Gentleman	
4	5	First Goth	First Goth	
...	...	...	...	...
1252	1253	Simpcox's Wife	Wife	
1257	1258	Cardinal Wolsey	CARDINAL WOLSEY	
1259	1260	Earl of Worcester	EARL OF WORCESTER	
1260	1261	(stage directions)	xxx	
1262	1263	Young Clifford	YOUNG CLIFFORD	

Figura S 3: characters con descripción nula o vacía

ChaName repetidos en df_characters

```
count_char= df_characters['CharName'].value_counts()
count_char.loc[count_char > 1].reset_index(name='Count')
```

	index	Count
0	All	23
1	Messenger	23
2	Servant	21
3	Lord	9
4	Page	8
...	...	...
120	All Lords	2
121	Lucilius	2
122	Lieutenant	2
123	Varrius	2
124	First Musician	2

125 rows × 2 columns

Figura S 4: CharName repetidos

Exploración de Datos

```
merged_df = pd.merge(df_paragraphs, df_characters, left_on='character_id', right_on='id')
merged_df['CharName'] = merged_df['CharName'].str.upper()
#cantidad de parrafos por personaje
merged_df.groupby(["character_id","CharName"])["ParagraphNum"].count().sort_values(ascending=False).reset_index(name
```

	character_id	CharName	Count
0	1261	(STAGE DIRECTIONS)	3751
1	894	POET	733
2	393	FALSTAFF	471
3	573	HENRY V	377
4	559	HAMLET	358
5	531	DUKE OF GLOUCESTER	285
6	844	OTHELLO	274
7	600	IAGO	272
8	120	ANTONY	253
9	945	RICHARD III	246

Figura S 5: Cantidad de párrafo agrupados por nombre de personaje y ID

```
merged_df = pd.merge(df_paragraphs, df_characters, left_on='character_id', right_on='id')
merged_df['CharName'] = merged_df['CharName'].str.upper()
#cantidad de parrafos por personaje
character_id=merged_df.groupby(["character_id","CharName"])["ParagraphNum"].count().sort_values(ascending=False).res
character_id=character_id['character_id']
df1_filtrado = df_characters[df_characters['id'].isin(character_id)]
df1_filtrado
```

	id	CharName	Abbrev	Description
119	120	Antony	ANTONY	(Marcus Antonius)
392	393	Falstaff	FALSTAFF	Sir John Falstaff
530	531	Duke of Gloucester	GLOUCESTER	brother to the King
558	559	Hamlet	Ham	son of the former king and nephew to the prese...
572	573	Henry V	HENRY5	Prince, King of England
599	600	Iago	IAGO	Othello's ancient (?)
843	844	Othello	OTHELLO	A noble Moor in the service of the Ventian state
893	894	Poet	Poet	the voice of Shakespeare's poetry
944	945	Richard III	RICHARD3	son of Richard Plantagenet, duke of York; was ...
1260	1261	(stage directions)	xxx	NaN

Figura S 6: personajes que aparecen en el top de agrupación por párrafo

Tabla de comparativa de modelos

	personaje	Precision	Recall	F1-Score	Support
NB	Cleopatra	1.00	0.07	0.12	61
	Falstaff	0.57	1.00	0.73	142
	Queen Margaret	1.00	0.04	0.08	51
	Accuracy			0.58	254
	Macro Avg	0.86	0.37	0.31	254
	Weighted Avg	0.76	0.58	0.45	254
SVC	Cleopatra	0.64	0.38	0.47	61
	Falstaff	0.70	0.91	0.79	142
	Queen Margaret	0.67	0.43	0.52	51

	Accuracy			0.69	254
	Macro Avg	0.67	0.57	0.60	254
	Weighted Avg	0.68	0.69	0.66	254
RF	Cleopatra	1.00	0.28	0.44	61
	Falstaff	0.65	1.00	0.78	142
	Queen Margaret	1.00	0.33	0.50	51
	Accuracy			0.69	254
	Macro Avg	0.88	0.54	0.57	254
	Weighted Avg	0.80	0.69	0.64	254