Introducción a la Ciencia de Datos

Curso 2023

Tarea 2

Grupo 8

Juan Manuel Varela - CI: 4.802.705-4

Nicolás Farías - CI: 4.143.102-6

Contenido

Introducción

Este documento es un informe sobre el trabajo realizado para la tarea del curso Introducción a la Ciencia de Datos. Se utilizó un conjunto de datos sobre la obra completa de William Shakespeare, extraídos de una base de datos relacional abierta disponible en:

- https://relational.fit.cvut.cz/dataset/Shakespeare

En una primera etapa, el trabajo consistió en extraer los datos de las tablas, procesarlos, y finalmente explorarlos para intentar contestar algunas preguntas sobre la obra de Shakespeare. Luego, se trabajó en entrenar algunos modelos de aprendizaje automático con el fin de predecir a qué personaje corresponde un párrafo determinado.

Carga y Exploración Inicial

La base de datos está compuesta por las siguientes tablas:

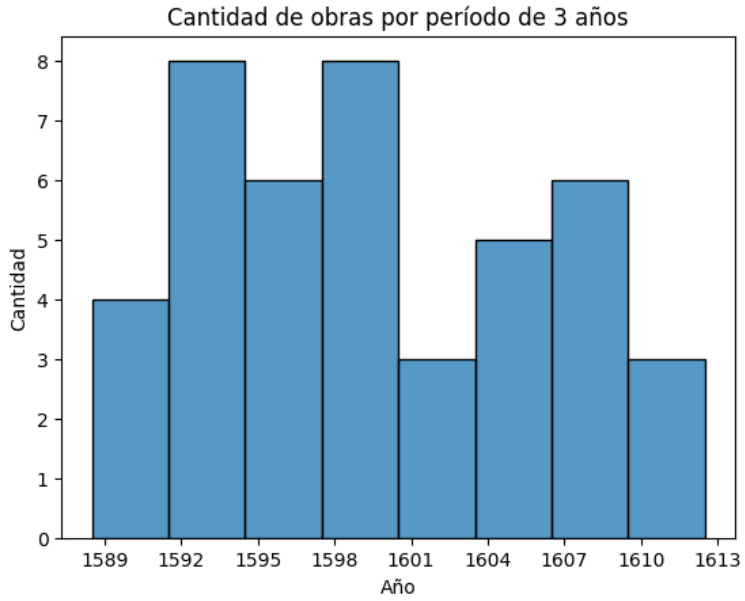
- ***works***: contiene todas las obras de Shakespeare con su año y género.
- ***chapters***: contiene los capítulos o escenas que componen las obras. Cada elemento se relaciona con una obra de la tabla ***works***.
- ***paragraphs***: contiene los párrafos (textos y diálogos) de las obras. Cada elemento se relaciona con un capítulo de la tabla ***chapters*** y con un personaje de la tabla ***characters***.
- ***characters***: contiene los personajes de las obras.

Se realizó una exploración visual y se utilizaron las funciones *info* y *nunique* de la biblioteca Pandas para hacer un diagnóstico inicial sobre la calidad de los datos. A continuación se detallan algunas consideraciones al respecto:

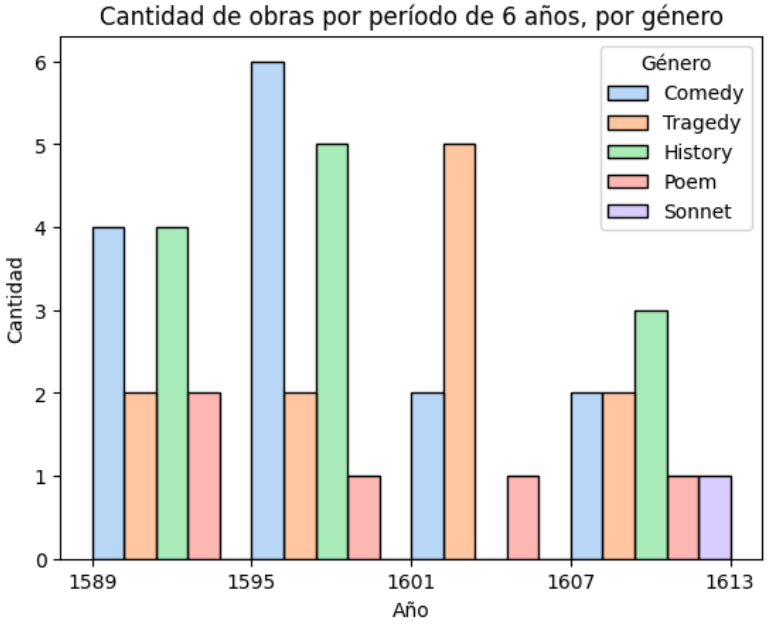
- Los tipos de datos son consistentes con la semántica de cada campo.
- Las columnas de nombre *id* efectivamente tienen valores únicos dentro de cada tabla.
- Muchos de los personajes (tabla *characters*) tienen el campo descripción vacío.
- Hay personajes con distinto *id* que tienen el mismo nombre. Esto puede deberse a que pertenecen a distintas obras.
- Muchos capítulos tienen distinto *id* pero la misma descripción, incluso dentro de una misma obra.
- Hay varios capítulos que no tienen descripción en las obras 28 (“Passionate Pilgrim”) y 35 (“Sonnets”).
- En los diálogos hay muchas contracciones con significados diferentes, algunas que ya no se utilizan en el inglés moderno.
- Hay 1220 personajes distintos en la tabla *paragraphs*, y 1266 personajes en total en la tabla *characters*, por lo tanto hay personajes que no tienen diálogos asignados.
- El personaje que tiene más párrafos asociados, con una distancia importante, tiene como nombre “(stage directions)”. Explorando algunos de estos párrafos se comprobó que efectivamente no son diálogos de un personaje real, sino que son indicaciones para la obra.

Obra de Shakespeare a los largo de los años

Se presenta a continuación un histograma que muestra la cantidad de obras escritas por Shakespeare agrupadas en períodos de tres años, comenzando en 1589, el año en que escribió su primera obra.



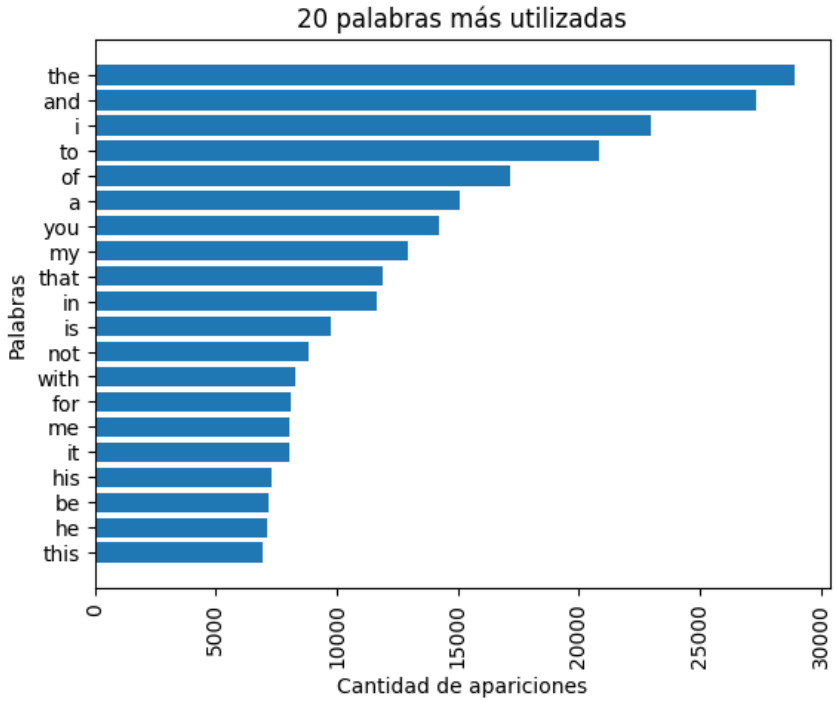
Podemos observar que el período de mayor productividad está comprendido entre los años 1592 y 1601. El resto de sus años de actividad escribió aproximadamente 4 obras cada 3 años.
Se presenta a continuación un histograma categorizado por género de las obras. En este caso, para facilitar la identificación de tendencias se agrupó las obras en periodos de 6 años.



Se puede observar como las obras de comedia e historia predominan en los inicios, disminuyendo hacia el final de su carrera, mientras que la tragedia toma un papel preponderante en la mitad de la misma. Los sonetos aparecen recién sobre el período de los últimos 6 años.

Análisis de palabras

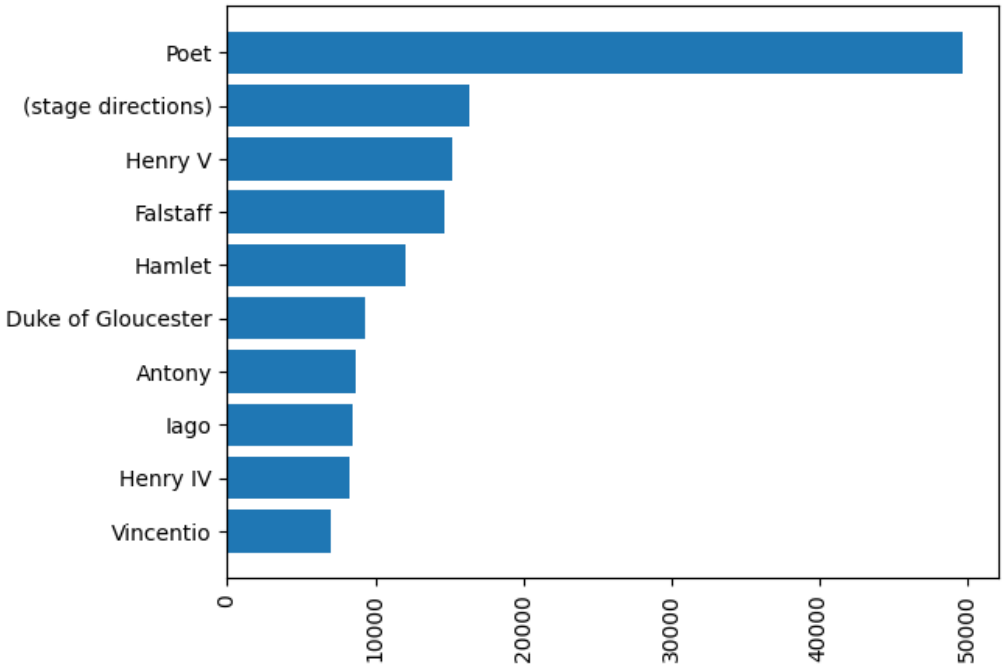
Para poder hacer un análisis de las palabras utilizadas en las distintas obras, es necesario realizar un trabajo de limpieza y normalización sobre el texto. Lo primero que se hizo fue pasar todo el texto a minúsculas. Luego de eso se eliminaron signos de puntuación y contracciones, con el objetivo de quedarnos únicamente con las palabras. Específicamente los caracteres y fragmentos eliminados fueron: "[", "\n", ",", ";", "?", ".", "!", "]", ":", "'d", "'s", "'ll", "'", "-". Si bien se podría intentar hacer un procesamiento más avanzado con respecto a las contracciones, consideramos que la cantidad no era significativa en el total de palabras, por lo que optamos simplemente por eliminarlas. Finalmente se generó un dataframe nuevo "df_words", separando las palabras de cada diálogo y colocándolas en filas diferentes. A partir de este *DataFrame* se cuenta la cantidad de apariciones de cada palabra y se ordena los resultados de mayor a menor.
Se presenta a continuación una gráfica mostrando las veinte palabras más usadas a lo largo de toda la obra, junto a la cantidad de veces que aparecen.



Se puede observar que la mayoría de las palabras corresponden a artículos, pronombres y preposiciones. Podría ser interesante utilizar un diccionario para quitar estas palabras de la comparación.
Por otro lado, dado que en el *DataFrame* de las palabras tenemos el identificador del capítulo, podemos hacer un *join* con las obras para tener disponible el género asociado. Esto nos permitiría realizar un filtrado previo y graficar las palabras más frecuentes para cada género, para comparar el vocabulario utilizado. También podríamos buscar los personajes con más palabras (como haremos más adelante) y graficar para cada uno de ellos las palabras más utilizadas, con el fin de identificar diferencias entre los distintos personajes.

Personajes con más palabras

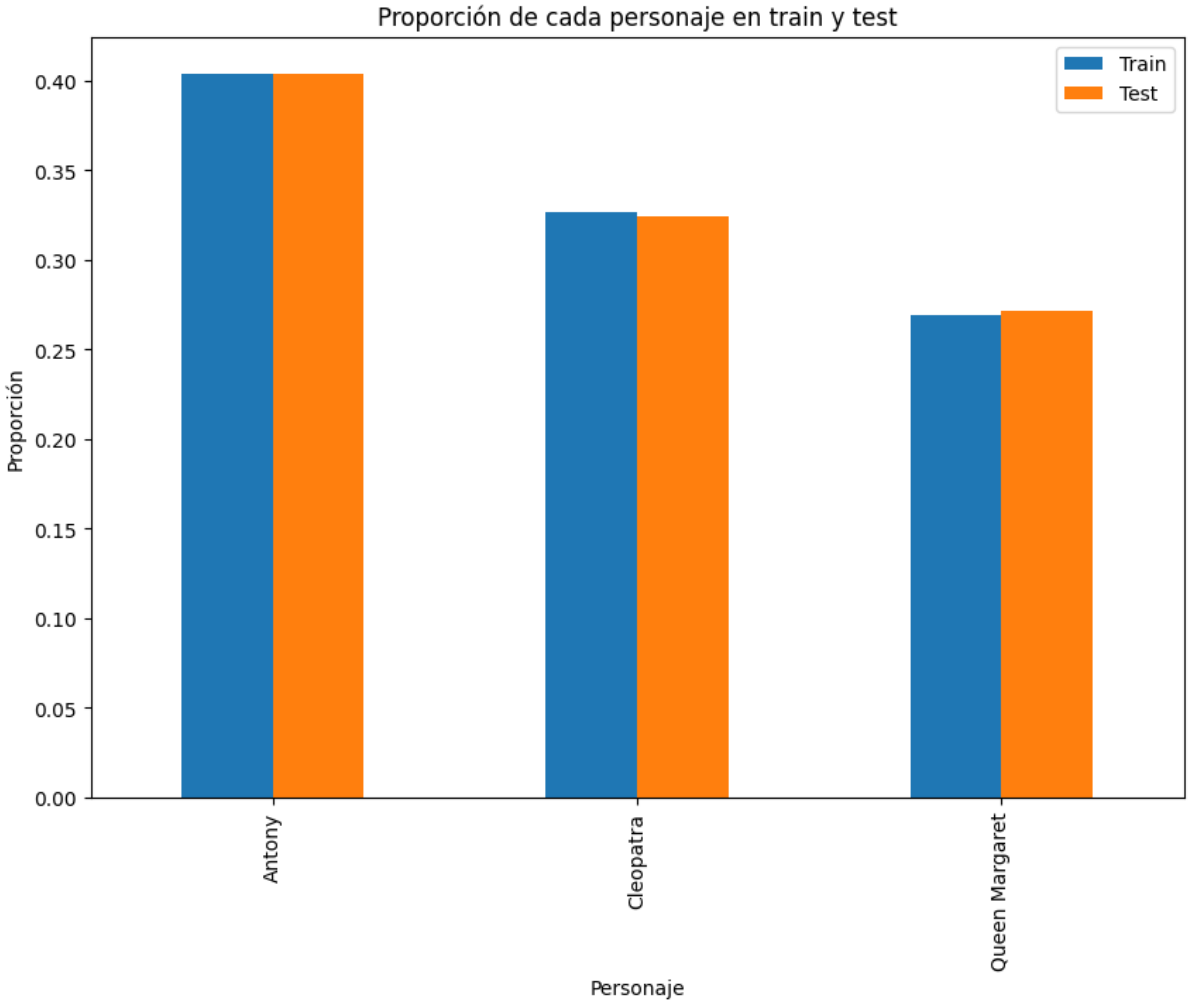
Se presenta a continuación una gráfica con los personajes que tienen más palabras asociadas a lo largo de toda la obra.



El personaje con más palabras es "Poet", con una amplia diferencia. Esto se debe a que este personaje tiene asignados todos los textos de las obras de poesía. El segundo personaje con más palabras es "(stage directions)", que como se vio anteriormente, tampoco es un personaje real, sino acotaciones. Se podrían descartar previamente todas las filas correspondientes a estos dos personajes, con el objetivo de visualizar la cantidad de palabras utilizadas por los que consideramos relevantes o diferenciar entre “Poet” o “(stage directions)” de distintas obras, para que no tengan tanto peso en el total de palabras.

División del Dataset

Para continuar trabajando sobre los datos, se crea un dataset reducido con solo tres personajes (Antony, Cleopatra y Queen Margaret) extraídos del dataset original. El texto de los párrafos de cada personaje fue procesado utilizando el mismo método de limpieza que se aplicó previamente al dataset completo. Posteriormente, se dividen los datos en un conjunto de entrenamiento y un conjunto de prueba, asignando un 30% del total para el conjunto de prueba, y utilizando un método de muestreo estratificado. Este método garantiza que la proporción de cada personaje se mantenga aproximadamente igual en ambos conjuntos, permitiendo un análisis equilibrado y representativo, ayudando a que luego los modelos de aprendizaje entrenados tengan un mejor rendimiento al generalizarlos. Se presenta a continuación un gráfico que muestra la proporción de cada personaje en el conjunto de entrenamiento y prueba.



Se puede ver que la proporción de cada personaje es aproximadamente la misma para ambos conjuntos.

Representación numérica de texto

En el siguiente paso se transformó el texto del conjunto de entrenamiento a una representación numérica utilizando la técnica *bag of words*. Esta es una representación numérica de conteo de palabras que convierte una colección de documentos de texto (en este caso párrafos) a una matriz de conteo de n-gramas. Un n-grama es un conjunto de n palabras agrupadas. En este caso tomamos n-gramas de longitud uno, por lo que estamos considerando palabras individualmente. Esta transformación produce una matriz de tamaño (NxM), donde N es el número total de párrafos y M es el número total de palabras distintas. Esta matriz tiene la característica de ser una *sparse matrix,* porque la mayoría de las entradas estarán rellenas de ceros, dado que al seleccionar cualquier fila individual (que representa un párrafo), la gran mayoría de las palabras del vocabulario no estarán presentes en el mismo.

Veamos a continuación un ejemplo para ilustrar el funcionamiento de la técnica:
Se tiene un conjunto que consta de tres documentos:

- "la mesa está rota"
- "la silla está sana"
- "la cama es grande"

En este caso el vocabulario está compuesto por 9 palabras diferentes: "la", "mesa", "está", "rota", "silla", "sana", "cama", "es", "grande".
Por lo tanto el resultado será la siguiente matriz de 3x9

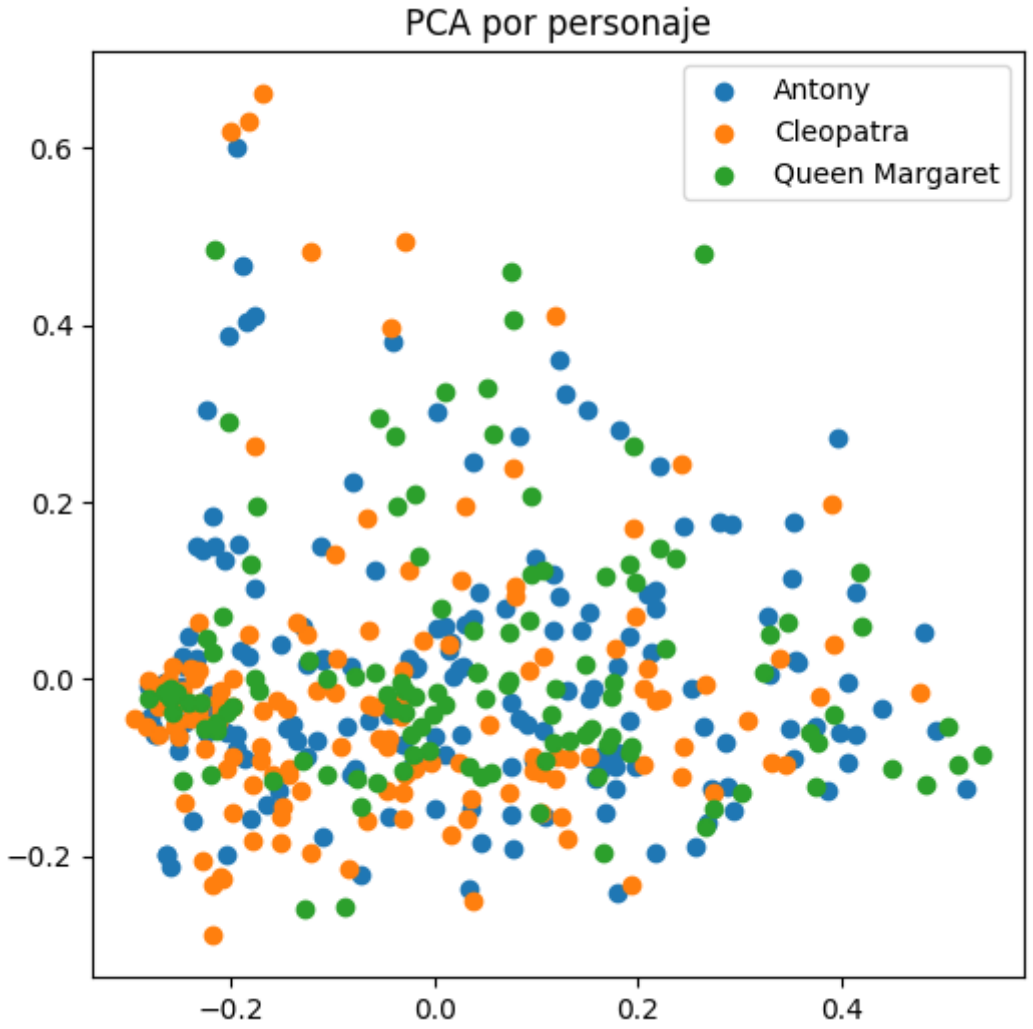
1 1 1 1 0 0 0 0 0
1 0 1 0 1 1 0 0 0
1 0 0 0 0 0 1 1 1

Un inconveniente que tiene esta representación es que el tamaño de la matriz aumenta muy rápidamente con el número de párrafos, ya que al agregar uno con palabras nuevas no sólo implicaría agregar una fila a la matriz, sino que también columnas correspondientes a las nuevas palabras. Imaginemos un escenario donde tenemos 10.000 párrafos y 100.000 palabras. Si suponemos que para almacenar cada número de la matriz necesitamos 4 bytes, toda la matriz requeriría 10.000 x 100.000 x 4 bytes de memoria, lo que nos da un total de 4GB. Para evitar este problema, lo que se hace habitualmente es usar representaciones pensadas específicamente para matrices de tipo *sparse matrix*, que almacenan únicamente los valores distintos de cero.

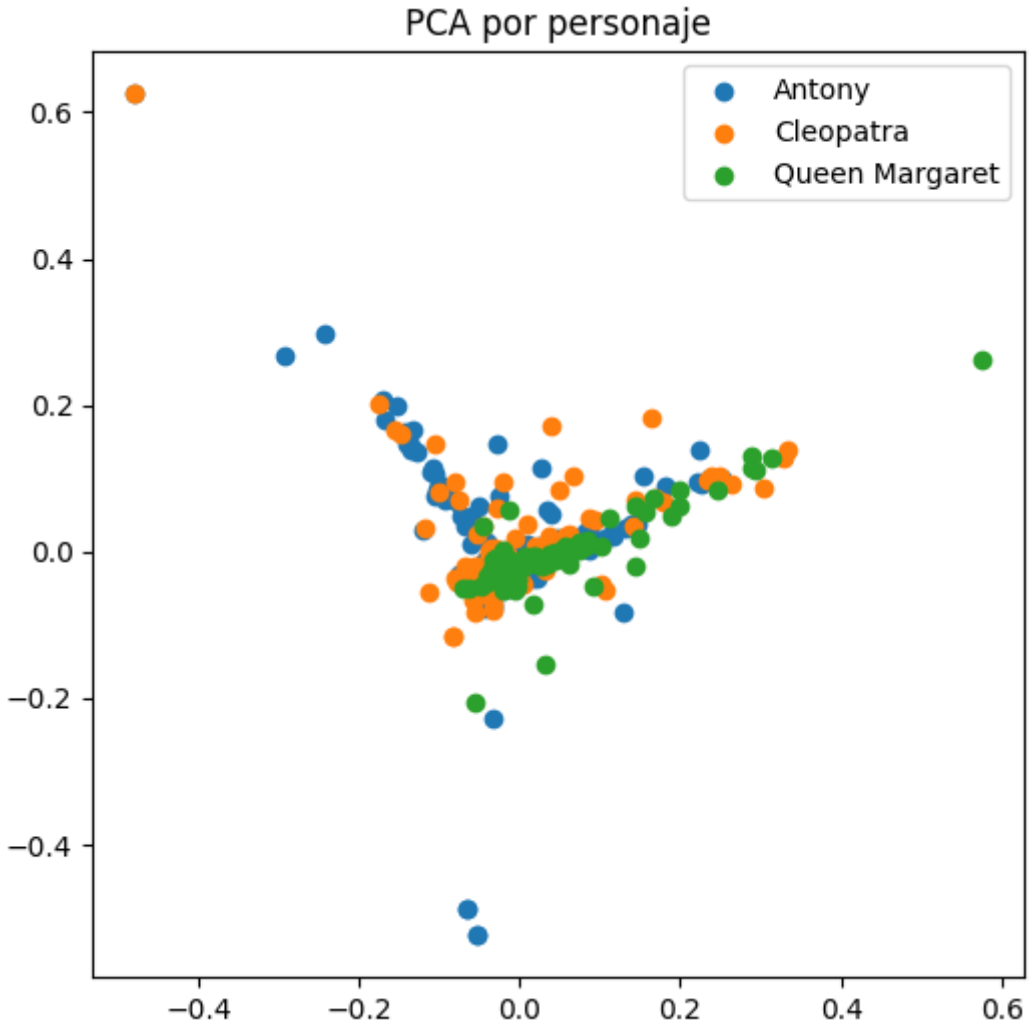
Una característica de la representación *bag of words* es que los párrafos más largos van a tener más entradas no nulas en la matriz, debido a la mayor cantidad de palabras. Para normalizar este aspecto lo que hacemos es transformar esta matriz a una representación TF (term-frequency), con la cual se logra que la sumatoria de cada fila sea igual a 1. La representación TF consiste en dividir cada fila de la matriz por la cantidad total de palabras de ese párrafo. Es decir, se divide cada entrada de la fila por la suma de todas las entradas de esa fila.

Principal Component Analysis

A partir de la representación TF realizamos la técnica PCA y graficamos los datos utilizando las primeras dos componentes.
El resultado se muestra en la siguiente imágen.

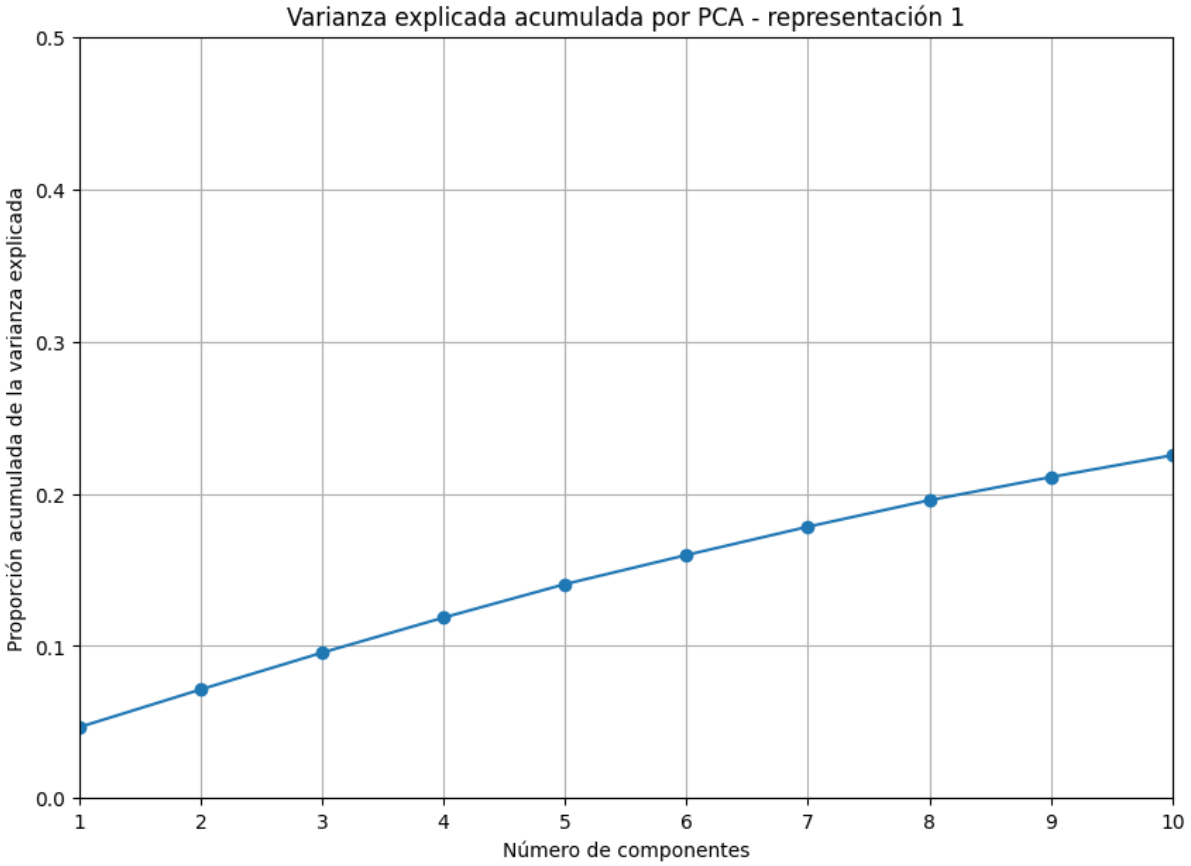


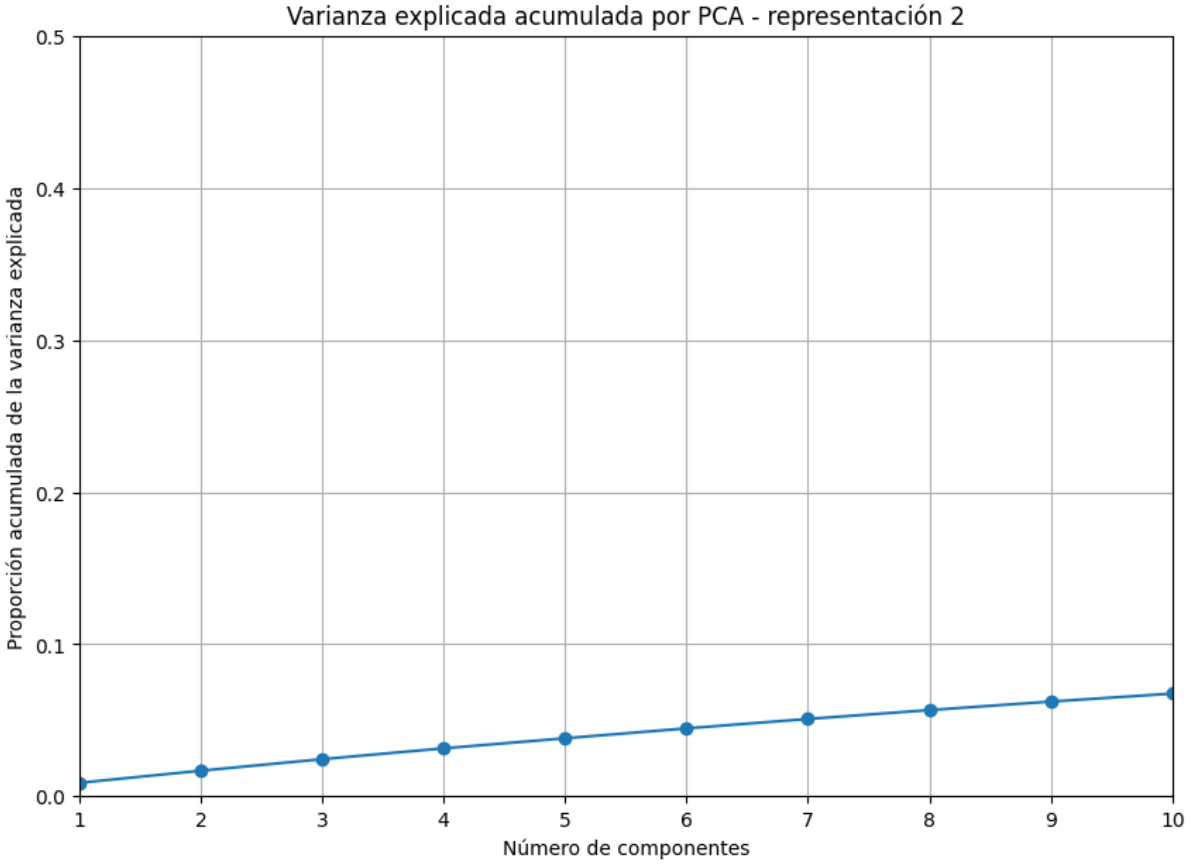
A su vez, se realizó otra prueba generando la matriz de bag of words a partir de n-gramas de longitud 1 y 2, filtrando stop words, y luego transformándola a la representación TF-IDF (term-frequency times inverse document-frequency).
La representación TF-IDF tiene en cuenta la "rareza" de cada palabra en el conjunto. Esto se hace multiplicando cada columna de la matriz TF descrita anteriormente por $log(\frac{\#\ de\ documentos\ totales}{\#de\ documentos\ que\ contienen\ la\ palabra})$. Esto hace que se le quite peso a las palabras más comunes.
Se realizó una PCA sobre esta segunda representación, graficando nuevamente los datos utilizando las primeras dos componentes.



Ambas gráficas nos dan la idea de que no es posible separar los personajes a partir de los dos componentes principales, ya que los puntos quedan muy mezclados. No obstante, en la segunda gráfica los puntos aparecen bastante más apretados que en la primera. Si bien en la segunda representación utilizamos el filtrado de stop words que nos reduce la cantidad de palabras, al tomar los n-gramas de tamaño uno y dos, el número total de términos (*features)* pasa a ser más de tres veces mayor que en la primera representación. Esto explicaría la diferencia de las gráficas, particularmente el hecho de que en la segunda representación los puntos casi no logran separarse con solo dos componentes. Al comparar la varianza explicada de ambas representaciones, se puede comprobar que efectivamente en el caso de la segunda, el valor acumulado con dos componentes es notoriamente inferior, lo cual es consistente con la diferencia entre las visualizaciones del mapa de puntos.

A continuación se muestra como evoluciona la varianza explicada a medida que se agregan componentes para ambas representaciones.

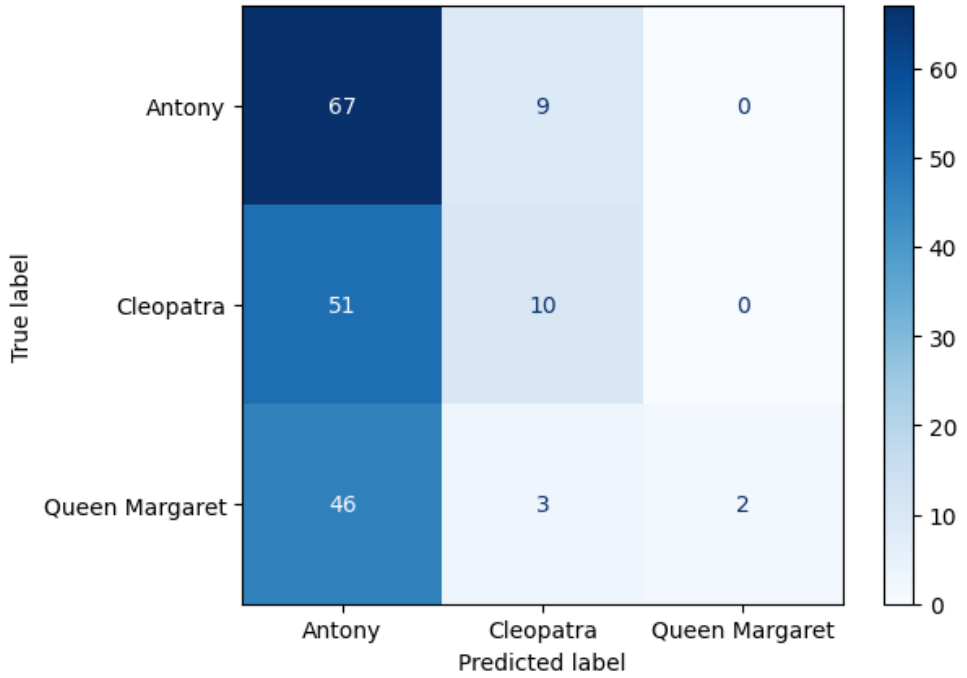

Varianza explicada acumulada por PCA - representación 1
Proporción acumulada de la varianza explicada
Número de componentes
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10


Varianza explicada acumulada por PCA - representación 2
Proporción acumulada de la varianza explicada
Número de componentes
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10

Entrenamiento y validación de modelos

Modelo Multinomial Naive Bayes

Como primer tipo de modelo se utilizó el Multinomial Naive Bayes. Se trabajó con la representación numérica del texto donde no se filtraron stop words, se consideraron n-gramas de longitud 1 y con representación TF.
Luego del entrenamiento, pasamos a evaluar el rendimiento del modelo sobre el conjunto de test, obteniendo un valor de *accuracy* de **0.42**. A continuación se presenta la matriz de confusión.



En la siguiente tabla presentamos los valores de *precision y recall* para cada personaje:

	Precision	*Recall*
Antony	0.41	0.88
Cleopatra	0.45	0.16
Queen Margaret	1.00	0.04

Para calcular el valor de *precision,* lo que se hace es analizar la columna de la matriz de confusión asociada al personaje en cuestión. Se utilizará como ejemplo Antony. Primero se toma el valor de las predicciones correctas, que es el que se encuentra en la diagonal (en este caso 67). Luego se divide ese valor entre la suma de toda la columna, que corresponde al total de veces que el modelo predijo a ese personaje (en este caso 67 + 51 + 46 = 164). Por lo tanto se tiene 67/164 = 0.41.
Para calcular el *recall* se hace lo mismo pero analizando por fila. Nuevamente tomando el caso de Antony, se divide el valor de la diagonal (67) por la suma de toda la fila, que corresponde al

total de veces en que efectivamente correspondía ese personaje (67 + 9 + 0 = 76). Por lo tanto se tiene 67/76 = 0.88.

Lo que se puede observar en este caso es que el modelo tiene mucho sesgo hacia el personaje Antony. Particularmente esto se ve en el valor alto de recall para este personaje, y extremadamente bajo para el resto. Esto puede deberse a que Antony es el personaje con el mayor porcentaje de representación en los datos. Si el desbalance de datos fuera aún más marcado hacia Antony, este modelo podría lograr una valor de *accuracy* bastante alto, dado que en la mayoría de los casos predice ese personaje. Incluso un modelo que únicamente predice el personaje Antony podría lograr un valor alto de *accuracy* en esos casos. Es por esto que mirar únicamente el valor de *accuracy* puede no ser suficiente para evaluar el rendimiento del modelo.

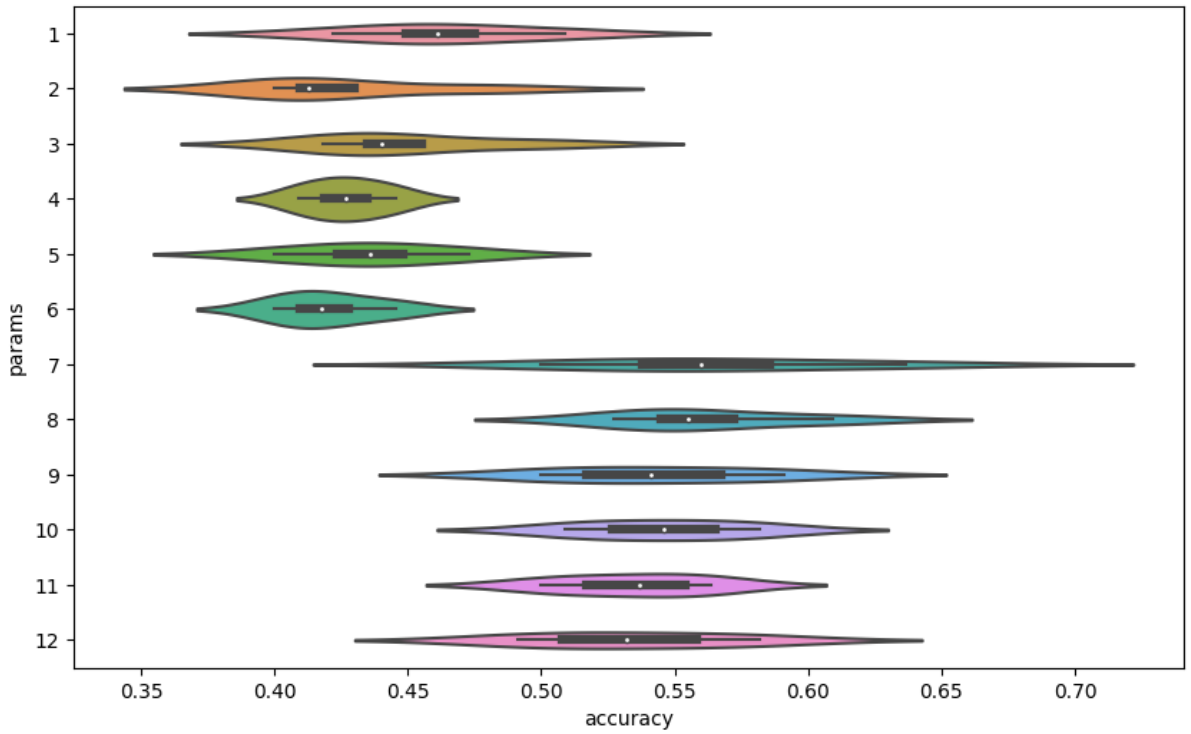
Búsqueda de hiper-parámetros

El próximo paso consiste en utilizar la técnica de validación cruzada y *grid search* para evaluar el modelo con distintas combinaciones de hiper-parámetros.
La validación cruzada es una técnica de evaluación y validación de modelos en la que el conjunto de datos se divide en k subconjuntos (o 'folds'). Luego, el modelo se entrena en k-1 subconjuntos y se prueba en el subconjunto restante. Este proceso se repite k veces, con cada subconjunto utilizado exactamente una vez como conjunto de prueba. El resultado final es el promedio de los resultados de las k iteraciones (en nuestro caso se utilizó k = 4).
Grid search consiste en probar sistemáticamente una serie de combinaciones de hiper-parámetros y elegir la combinación que produce el mejor rendimiento en la validación cruzada.

Las opciones de hiper parámetros que usamos son las siguientes:

- Filtrar stop words: si o no
- Largo de n-gramas a considerar: 1, 1 a 2, o 1 a 3.
- Representación IDF: si o no.

A continuación se muestra un gráfico de violín indicando el valor de accuracy para cada combinación de hiper-parámetros.



Se decide que los mejores hiper-parámetros son los del conjunto 8, ya que aunque el modelo tiene una accuracy algo menor que el del conjunto 7, la varianza es menor.

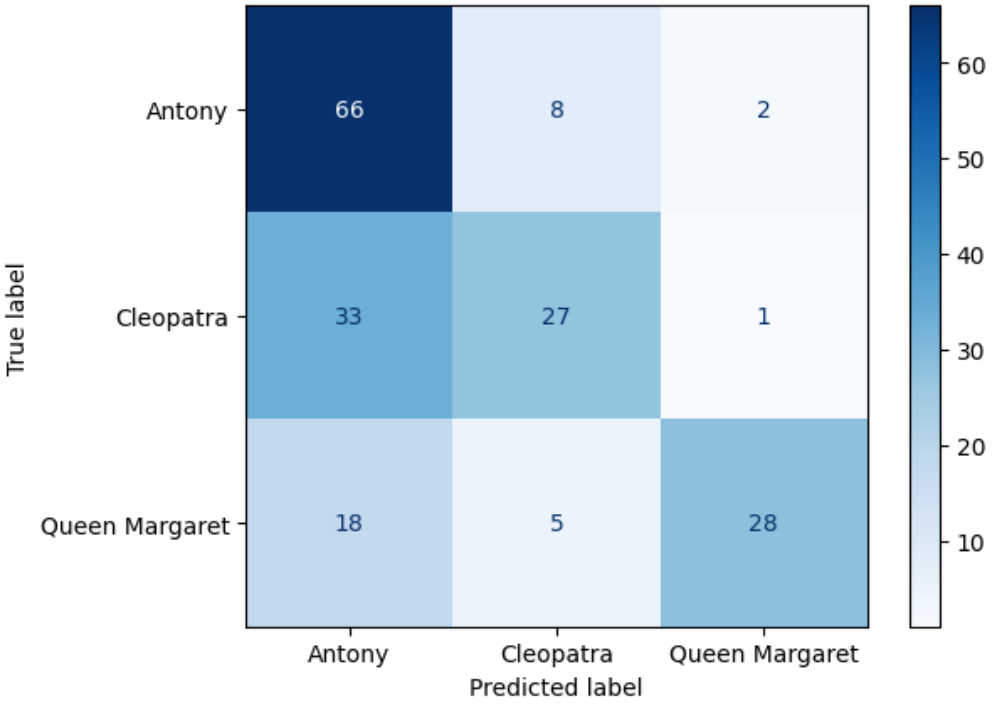
Este conjunto corresponde a los siguientes hiper-parámetros:

- Filtrar stop words: **si**
- Largo de n-gramas a considerar: **1**
- Representación IDF: **no**

Con este nuevo modelo se repitió el proceso de entrenamiento y evaluación, obteniendo las siguientes métricas.

accuracy: **0.64**

Matriz de confusión:



En la siguiente tabla se presentan los valores de *precision y recall* para cada personaje:

	Precision	*Recall*
Antony	0.56	0.87
Cleopatra	0.68	0.44
Queen Margaret	0.90	0.55

Como se puede observar con las métricas, claramente se logró una mejora en el rendimiento del modelo al utilizar esta combinación de hiper-parámetros, aunque sigue habiendo un sesgo hacia Antony.
Hay ciertas limitaciones inherentes al modelo de bag of words o tf-idf que dificultan seguir mejorando aún más las métricas. Una limitación es que estos modelos tratan a todas las palabras de manera independiente, por lo que se pierden las estructuras y el orden entre ellas. Esa independencia también hace que no se tenga en cuenta la relación (o cercanía) semántica entre palabras.

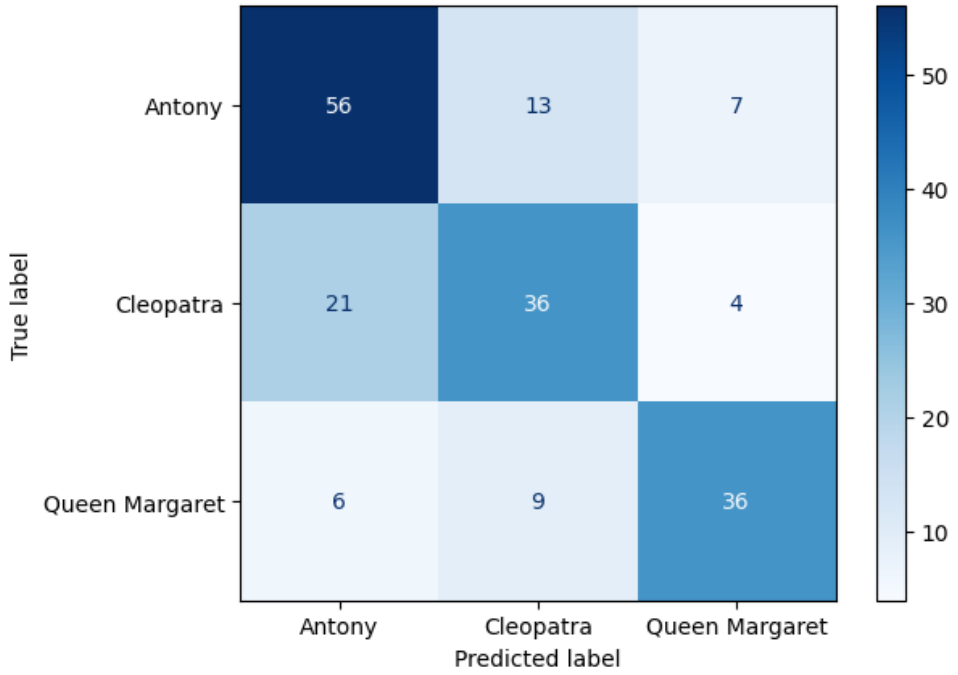
Modelo SVM

Se prueba ahora reemplazar el modelo Multinomial Naive Bayes por un modelo lineal SVM (Support Vector Machines). Para eso se utiliza la clase *sklearn.linear_model.SGDClassifier*, con los parámetros correspondientes a un modelo SVM lineal. En el modelo SVM lineal básicamente se intenta encontrar un hiper-plano que separe los puntos de cada clase de manera óptima. Específicamente se busca maximizar la distancia entre el hiper-plano y los puntos más cercanos de cada clase.

Lo que se hace entonces es entrenar este nuevo modelo utilizando la matriz de *features* que se generó con los hiper-parámetros del paso anterior. Luego de evaluarlo contra el conjunto de test, se obtuvieron las siguientes métricas.

accuracy: **0.68**

Matriz de confusión:



En la siguiente tabla se presentan los valores de *precision y recall* para cada personaje:

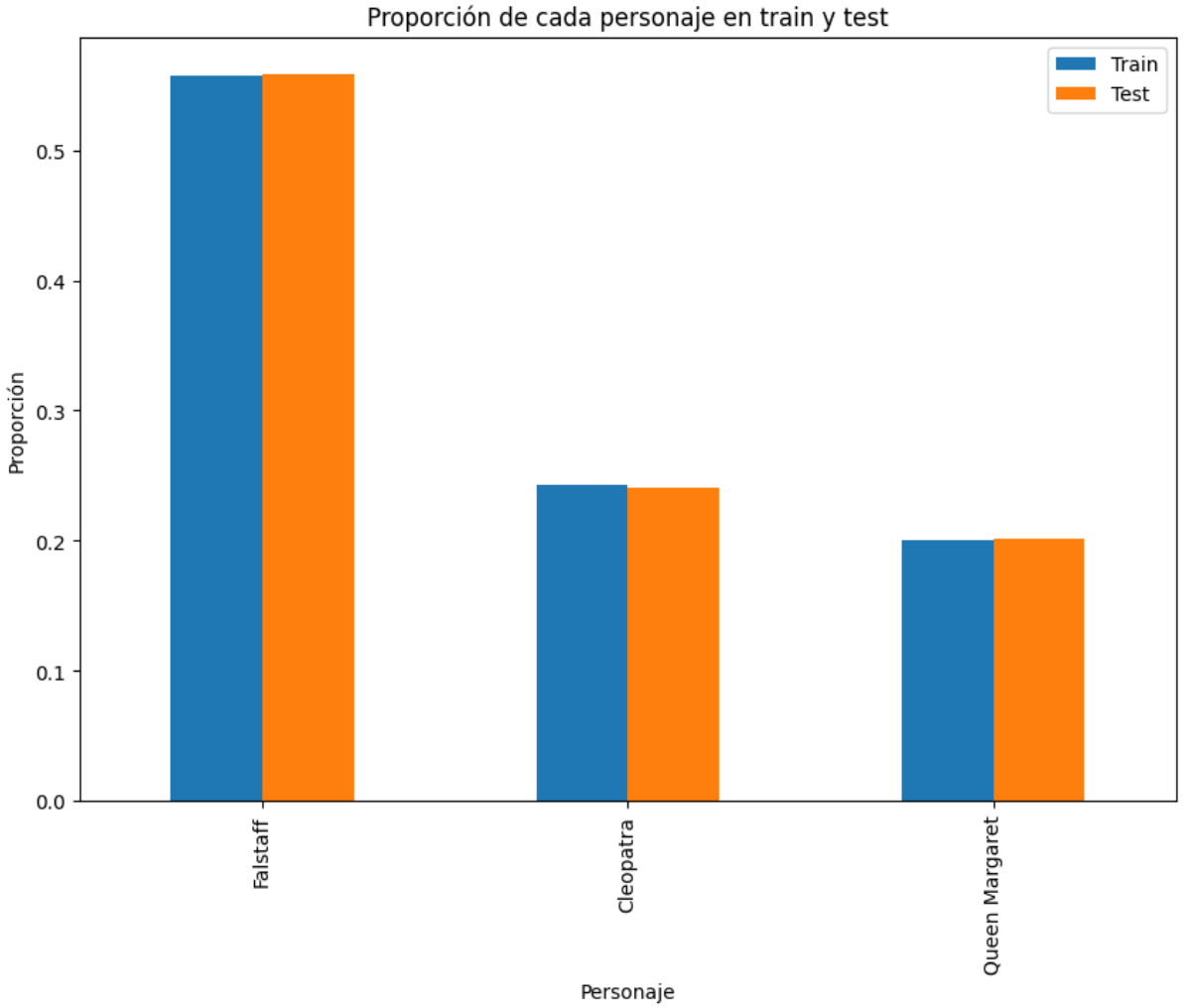
	Precision	*Recall*
Antony	0.67	0.74
Cleopatra	0.62	0.59
Queen Margaret	0.77	0.71

Se puede observar que se logra un pequeño aumento en la métrica de *accuracy*. Si bien también se logra un aumento en varias de las otras medidas, podemos encontrar que alguna disminuye, como ser la *precision* para el personaje Queen Margaret o el *recall* del personaje Antony.

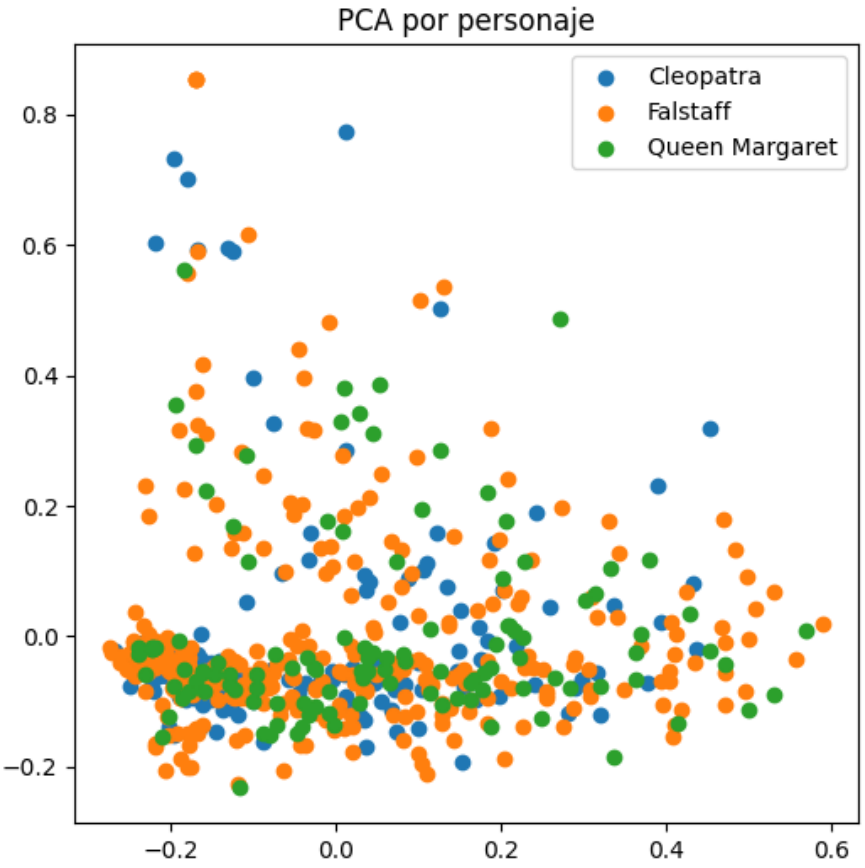
Cambio de dataset

Para evaluar el problema con otra combinación de personajes, lo que se hizo fue quitar a Antony y sustituirlo por Falstaff. Por ser Falstaff el personaje con mayor cantidad de párrafos, lo que ocurre en este nuevo escenario es que se acentúa aún más el desbalance de clases.

Se presenta a continuación un gráfico que muestra la proporción de cada personaje en el conjunto de entrenamiento y prueba, luego de repetir el proceso de división del *DataSet* con muestreo estratificado.

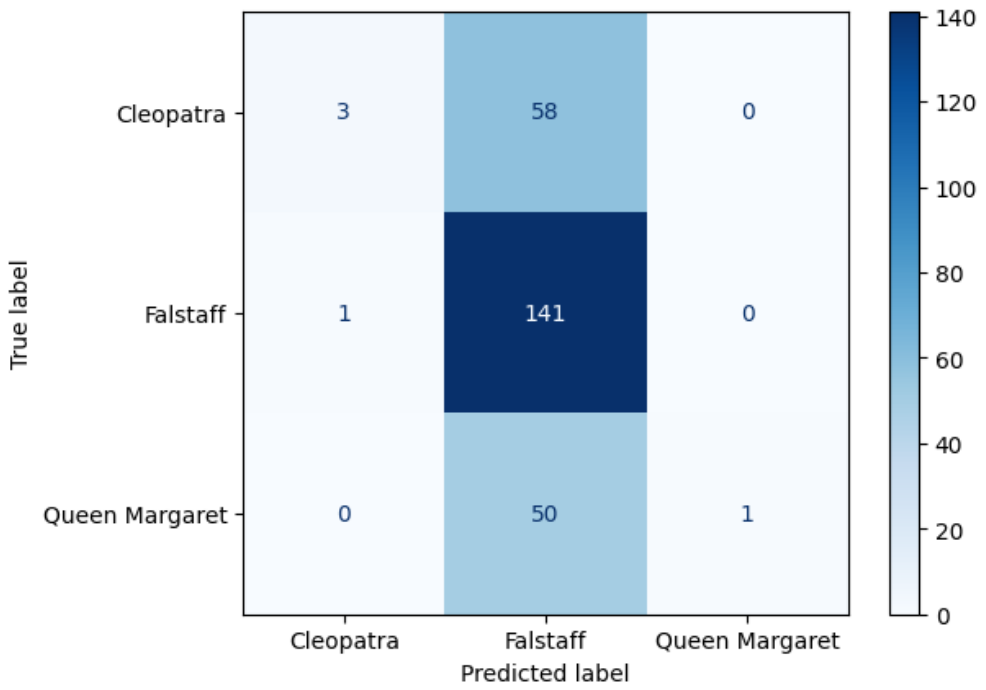


Luego de esto se repite el proceso de transformación de texto a *features* con la configuración inicial de los *encoders* (sin filtrado de stop words, n-gramas de longitud 1, representación TF). En la siguiente imagen se presenta la representación gráfica de los datos utilizando las dos primeras componentes de la PCA.



Debido al mayor desbalance de clases de este nuevo escenario, se puede apreciar como predominan los puntos de color anaranjado, correspondientes al personaje Falstaff.

Al entrenar y evaluar el modelo Multinomial Naive Bayes con la configuración inicial (sin buscar los mejores hiper-parámetros), se obtiene un valor de *accuracy* de **0.57**. Recordemos que en el escenario análogo con los personajes iniciales, el valor obtenido era 0.42. En una primera lectura se podría pensar que se está ante una mejora importante en el rendimiento del modelo. Sin embargo, al analizar el resto de las métricas, se puede ver que ese incremento del *accuracy* se debe simplemente al mayor desbalance de clases, y que no se trata efectivamente de una mejora. Este ejemplo permite ilustrar el problema que se mencionó anteriormente respecto a mirar únicamente la medida de *accuracy*.
A continuación se presenta la matriz de confusión y el resto de las métricas para el modelo con estos personajes:



	Precision	*Recall*
Cleopatra	0.75	0.05
Falstaff	0.57	0.99
Queen Margaret	1.00	0.02

En estos escenarios donde se tiene un marcado desbalance entre las clases, puede ser útil aplicar técnicas de sobre-muestreo o submuestreo para lograr un mayor equilibrio.
El sobre-muestreo básicamente consiste en aumentar la cantidad de instancias de las clases que están menos representadas en el *DataSet*. Una forma de hacerlo es simplemente duplicar algunas de las instancias existentes, ya sea eligiendolas de manera arbitraria o aleatoria.
Por otra parte, el submuestreo se trata de reducir la cantidad de instancias de las clases mayoritarias. Para esto podemos simplemente eliminar algunas instancias elegidas por algún criterio específico.

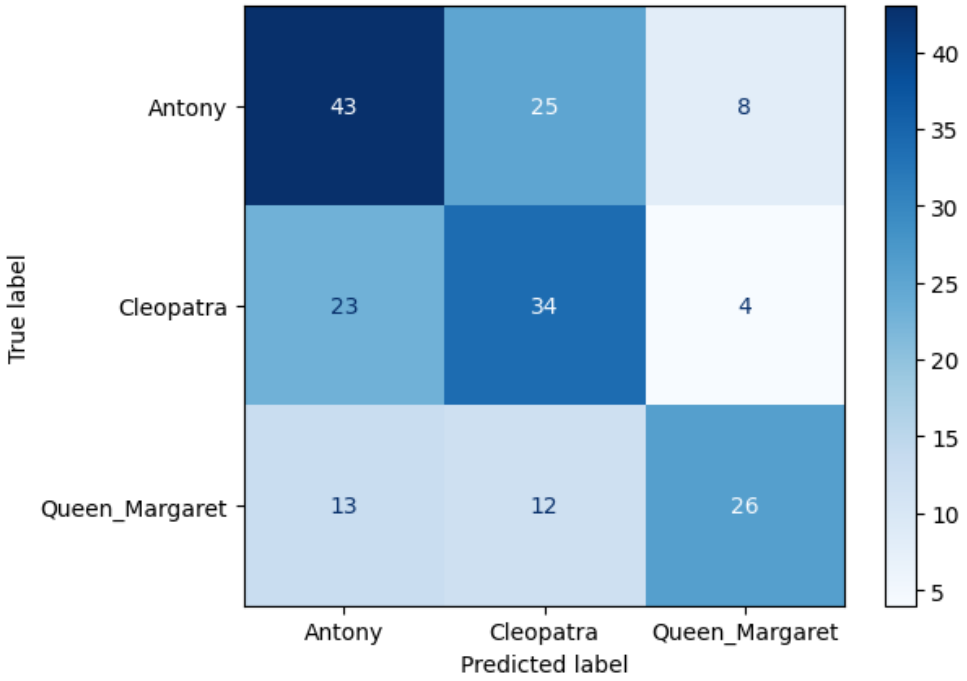
Word embeddings

Una técnica alternativa a bag of words para extraer features de texto es Word2Vec. Esta técnica toma un texto como entrada y produce un espacio vectorial de varias dimensiones, donde cada palabra única del texto se corresponde con un vector en el espacio. Los vectores se generan de manera que las palabras que comparten contextos similares en el texto, mantienen cierta cercanía en el espacio. Luego, para representar un párrafo, se puede combinar mediante alguna operación los vectores correspondientes a cada palabra presente en el mismo.
Una de las principales diferencias de los vectores resultantes de aplicar esta técnica con respecto a los resultantes de aplicar bag of words es que en general la cantidad de dimensiones será menor, ya que esta no depende de la cantidad de palabras, si no que se elige de antemano.

Otra característica es que el resultado no será una *sparse matrix*, ya que cada columna ahora no representa una palabra sino una dimensión del espacio.

fastText

Como último paso se entrenó y evaluó el modelo fastText. En este caso se utilizó el *DataSet* con la configuración original de personajes. No fue necesario utilizar los *encoders* previos, dado que el modelo fastText utiliza su propia representación de texto.
El valor de *accuracy* obtenido al evaluar con el conjunto de test fue **0.55**.
A continuación se presenta la matriz de confusión y el resto de las métricas habituales:



	Precision	*Recall*
Antony	0.54	0.57
Cleopatra	0.48	0.56
Queen Margaret	0.68	0.51

Si bien el valor obtenido de accuracy es más bajo que en los modelos Multinomial Naive Bayes y SVM lineal, se puede apreciar que los valores de recall y precision están más equilibrados que en el primer modelo, lo cual se puede interpretar como que tiene menor sesgo hacia un personaje específico.
De todas formas, se considera que el mejor resultado se obtuvo aplicando el modelo SVM lineal, donde se tiene mayor accuracy y menor sesgo.

FastText es un modelo rápido y eficiente, especialmente útil para grandes conjuntos de datos y soporta múltiples idiomas gracias a su enfoque basado en n-gramas. Sin embargo, no considera el contexto de las palabras como lo hacen otros modelos, puede no rendir tan bien con conjuntos de datos reducidos y requiere cierto preprocesamiento. Además, su interpretación puede ser difícil, similar a otros modelos de aprendizaje profundo.