INTRODUCCIÓN A LA CIENCIA DE DATOS

Tarea - 2

AÑO 2024

GRUPO 10:

Francisco Galletto

FECHA: 5 de julio de 2024

Tabla de contenidos

1. Dataset y representación numérica

1.1. Dataset reducido

Se crea un Dataset reducido con los datos de los personajes Antony, Cleopatra y Queen Margaret. Los párrafos de el Dataset reducido fueron filtados por medio de la función `clean_text()` utilizada en la Tarea-1.
Finalmente, utilizando la función `train_test_split` se obtienen los conjuntos de train y test mediante un muestreo estratificado que asegura obtener una muestra representativa del total. El muestreo correspondiente al test es del 30 % de los datos, utilizando el parámetro `random_state` en 42 (arbitrario) para garantizar reproducibilidad.

1.2. Balance en train y test

En la Figura 1 se puede visualizar la distribución de cantidad de párrafos por personaje para los datos de train y test. Luego, se verifica que se conserven los porcentajes de cada personaje respecto al total, esto se puede ver en la Tabla 1.

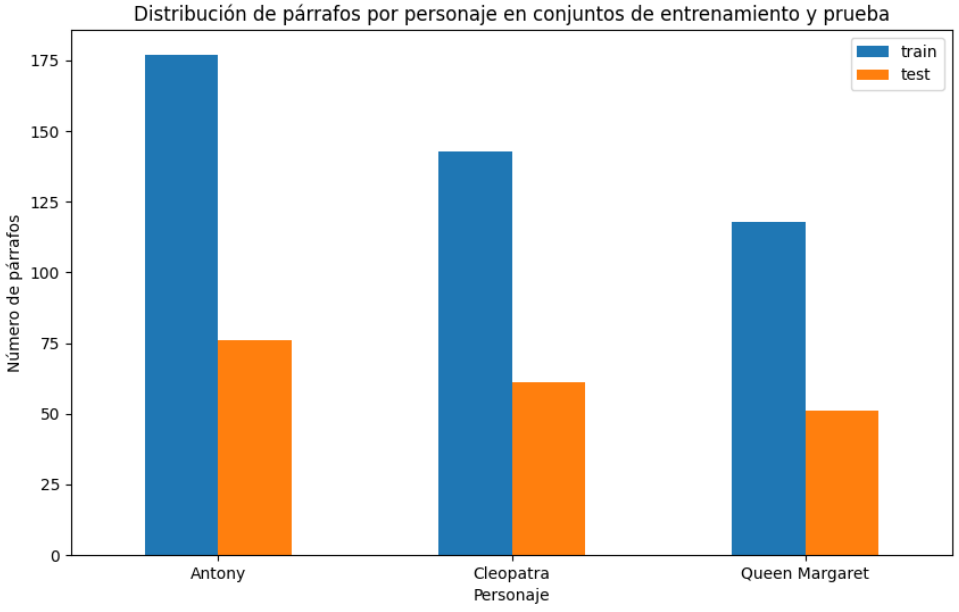

Figura 1: Cantidad de párrafos por personajes en los datos de train y test.

Tabla 1: Porcentaje de párrafos por personaje en los datos de train y test.

Personaje	Train (%)	Test (%)
Antony	40.4	40.4
Cleopatra	32.6	32.5
Queen Margaret	27	27.1

1.3. Bag of Words

La técnica Bag of Words (BoW) se utiliza en el procesamiento del lenguaje natural (NLP) para representar texto de forma numérica. Lo que hace es dividir el texto en palabras, crear el vocabulario de todas las palabras únicas presentes en el set de datos, para finalmente contar la cantidad de veces que aparece cada palabra en el vocabulario. Con esto se obtiene una matriz donde cada fila representa un texto, cada columna respresenta una palabra del vocabulario creado y cada valor de la matriz indica cuántas veces aparece la palabra en el texto correspondiente.
Para introducir un ejemplo, se toma una base de datos compuesta por:

```
text = ['El lugar es mágico', 'El lugar es raro', 'Un día lo soñé']
```

Se fijan los parámetros `ngram_range=(1,1)` y `stop_words=None`, donde `ngram_range` define la cantidad de **n-gramas** y `stop_words` especifica si se eliminan o no las palabras vacías. Con esto se obtiene el siguiente vocabulario y columna asociada a cada palabra:

```
{'el': 1, 'lugar': 4, 'es': 2,'mágico': 5, 'raro': 6, 'un': 8, 'día': 0,
'lo': 3, 'soñé': 7}
```

Lo que resulta en la siguiente matriz:

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Se puede ver que la cantidad de filas de la matriz es igual a la cantidad de frases, y la cantidad de columnas es igual a la cantidad de palabras únicas presentes en la base de datos.
Algo importante de este tipo de representación es que se obtienen matrices dispersas (**sparse matrix**), ya que la mayoría de sus elementos son cero. Esto resulta eficiente en términos de memoria dado que solo se almacenan los términos distintos de cero.

El método mostrado anteriormente se aplica al texto del set de datos de entrenamiento, obteniendose la matriz `X_train_counts` que tiene un tamaño de $(438, 2807)$.

1.4. Representación numérica TF-IDF

1.4.1. n-grama

Un n-grama es una secuencia contigua de n elementos presentes en un texto. Los mismos son ampliamente utilizados en el procesamiento del lenguaje natural para representar texto en forma numérica. Por ejemplo, se puede construir los n-gramas con elementos de distintos tipos como fonemas, sílabas, letras y palabras. El tipo de elementos utilizados va a depender del ámbito y objetivo dispuesto para cada aplicación particular.
Algunos tipos de n-gramas comúnmente utilizados son los Unigramas, Bigramas y Trigramas. En el primer caso se puede capturar información de la cantidad de

ocurrencia de cada palabra, a costa de perder información sobre el contexto. En el segundo y tercer caso se captura más el contexto, pero al aumentar la dimensión también aumenta el requerimiento de recursos computacionales.

1.4.2. TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) es una técnica utilizada para representar numéricamente la importancia de un elemento dentro de un documento perteneciente a una colección. Esto lo hace mediante la ponderación por un valor que dice qué tan común o raro es encontrar el elemento en la colección, siendo útil para reducir el impacto de palabras que ocurren con mucha frecuencia.
Para lograr esto se utilizan dos términos, por un lado TF y por otro IDF descriptos a continuación:

- **TF:** Cuantifica la frecuencia de un elemento en el documento. Se obtiene del cociente entre el número de veces que aparece el elemento (puede ser un n-gama) en el documento, y el número total de elementos en el documento.
- **IDF:** Mide qué tan raro es encontrar un elemento en la colección. Se obtiene de hacer el logaritmo del cociente entre el número total de documentos en la colección y el número de documentos que contienen al elemento.

Finalmente, se obtiene TF-IDF para un elemento dado de cierto documento perteneciente a la colección realizando el producto entre TF e IDF.

En este caso se obtiene una representación numérica TF-IDF mediante la función `TfidfTransformer` de `scikit-learn` para transformar la matriz de conteo de palabras (BoW) obtenida en la sección anterior en una matriz de frecuencias de elementos (TF) sin aplicar la parte de Inverse Document Frequency (IDF).

1.5. PCA

Se utiliza la técnica de componentes principales para reducir la dimensionalidad del conjunto de datos conservando la mayor parte de la varianza de los datos originales. Para lograr esto se encuentran las componentes principales que son las direcciones en las cuales se tiene la mayor variación de los datos. Esta técnica, además de ser utilizada en aprendizaje automático, es ampliamente utilizada en eólica para correlacionar estaciones de medida cercanas, encontrando los patrones meteorológicos principales en cada estación.

En este caso se obtienen las dos primeras componentes principales de los vectores TF-IDF, y luego se analizan los resultados al variar el filtado de stopwords (en inglés), utilización de Unigramas-Bigramas y utilización del factor IDF.

El primer caso se realiza sin filtrar por stopwords, utilizando unicamente Unigramas y sin utilizar el factor IDF. En la Figura 2 se muestra cómo es la distribución de los textos de cada personaje en el espacio de las dos primeras componentes principales. En este caso se puede ver que los datos están muy dispersos. Para ver la variabilidad

capturada según los componentes principales se muestra la Figura 3, donde se puede ver el cambio de la varianza respecto a las PCA.

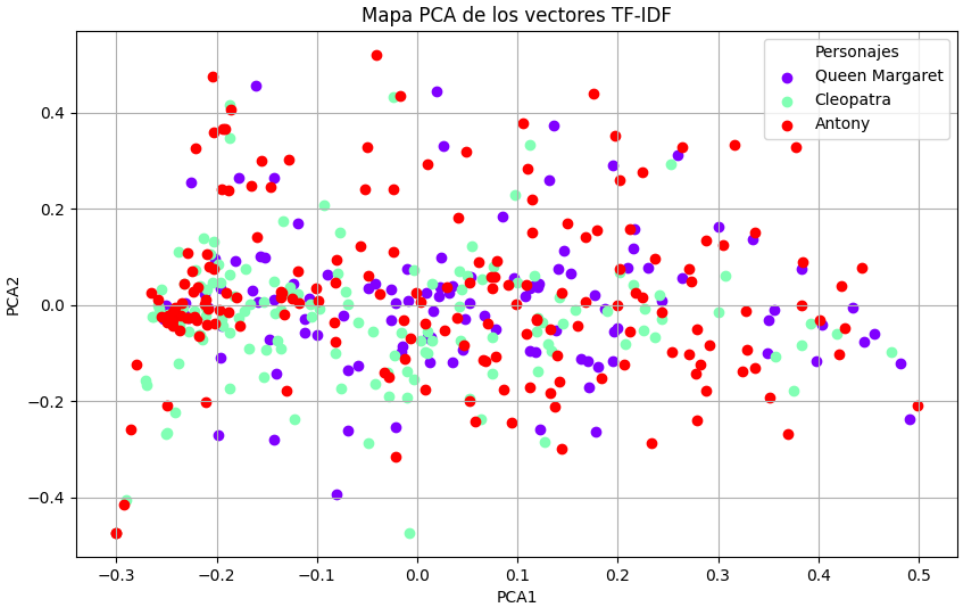

Figura 2: Mapa de los dos primeros componentes principales con `stopwords=None`, `ngram_range=(1, 1)` y `use_idf=True`.

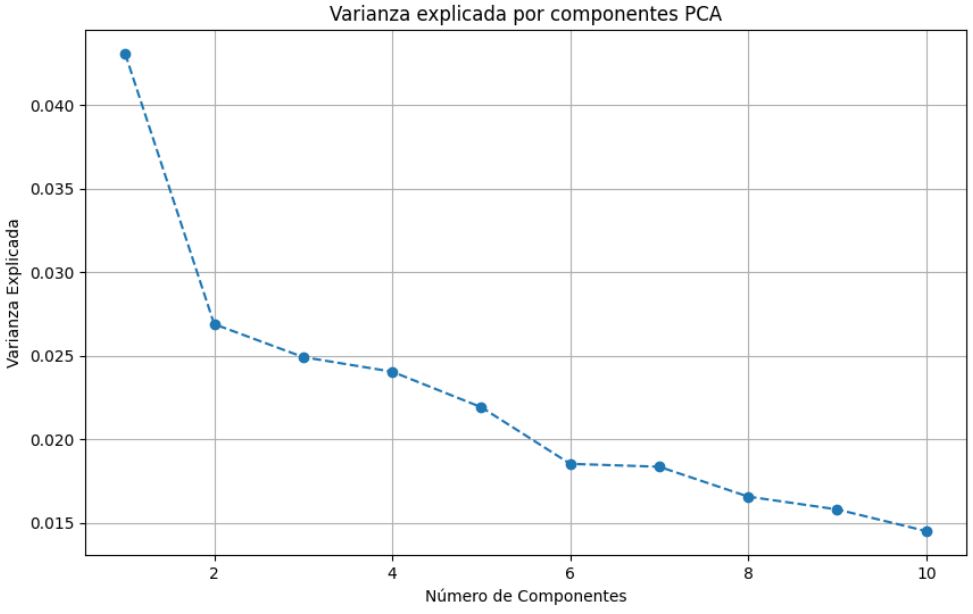

Figura 3: Varianza en función de los componentes proncipales con `stopwords=None`, `ngram_range=(1, 1)` y `use_idf=True`.

Luego se filtra por stopwords del idioma inglés, obteniendo el mapa PCA mostrado en la Figura 4. Se puede ver una reducción de la dispersión con respecto a los datos sin filtrar por stepwords. Para ver la variabilidad capturada según las componentes principales se muestra la Figura 5, donde se puede ver la reducción de la variabilidad capturada por las PCA respecto al caso anterior.

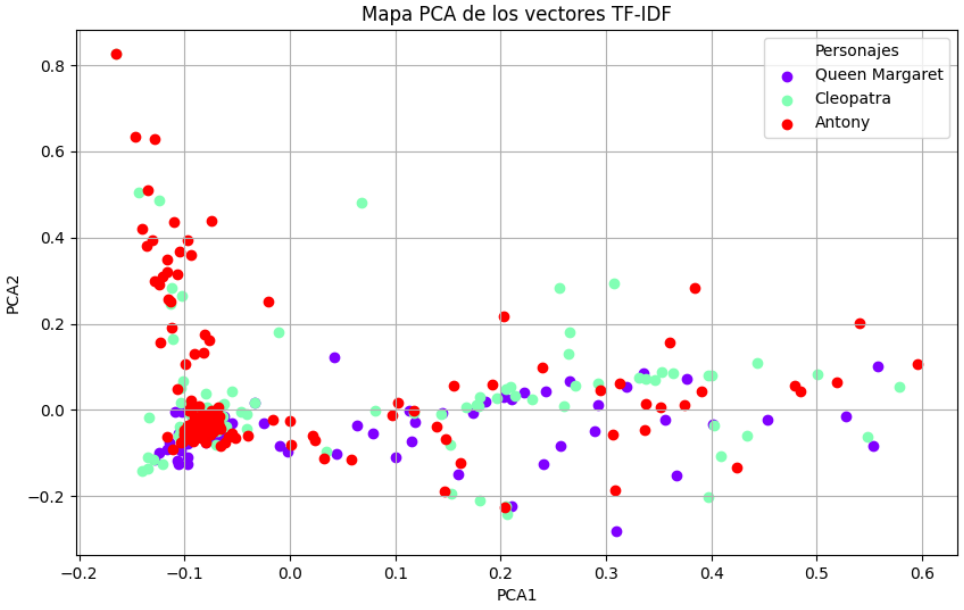

Figura 4: Mapa de los dos primeros componentes principales con `stopwords='english'`, `ngram_range=(1, 1)` y `use_idf=False`.

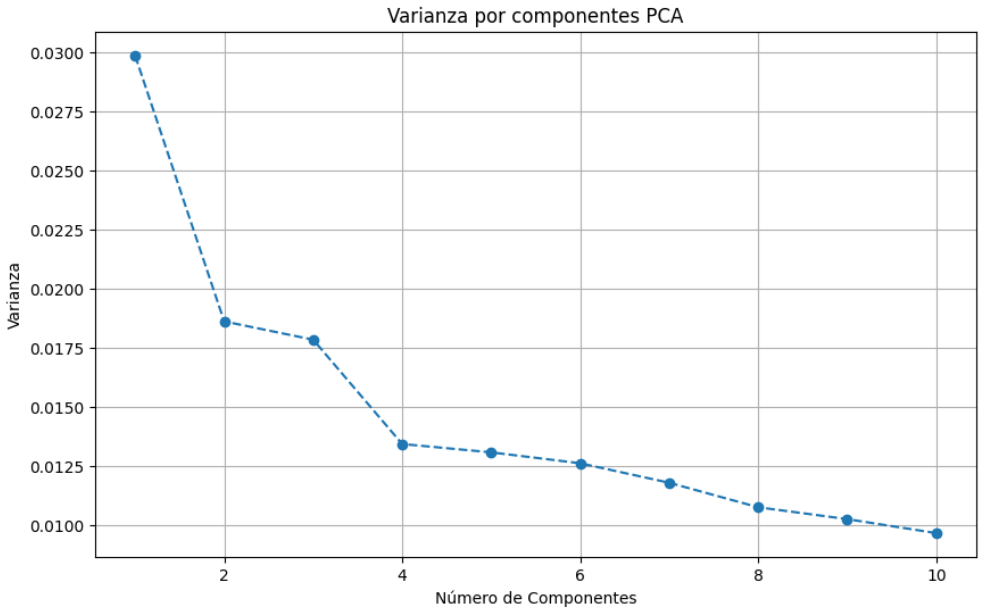

Figura 5: Varianza en función de los componentes principales con `stopwords='english'`, `ngram_range=(1, 1)` y `use_idf=False`.

Finalmente se filtra por stopwords, utilizando Unigramas y Bigramas para agrupar los elementos, y ahora sí se utiliza el factor IDF. Con esto se obtiene el mapa PCA de la Figura 6 donde se puede ver que los resultados son muy similares para los 3 personajes. Luego en la Figura 7 se puede ver la variabilidad de los resultados según los PCA, obteniendo una variabiliad significativamente inferior a los casos anteriores y muy similar para las dos primeros componentes principales.

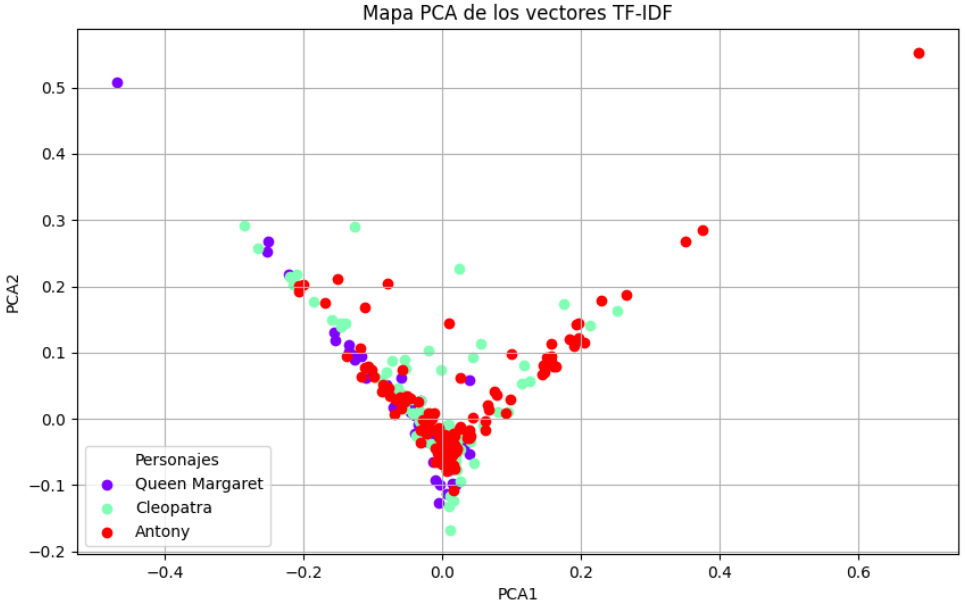

Figura 6: Mapa de los dos primeros componentes principales con `stopwords='english'`, `ngram_range=(1, 2)` y `use_idf=True`.

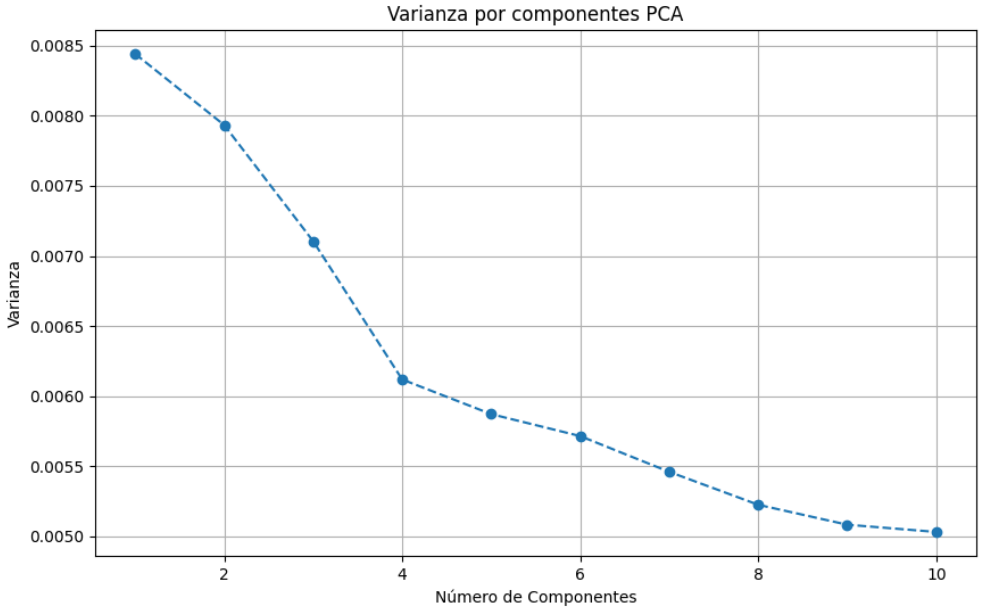

Figura 7: Varianza en función de los componentes principales con `stopwords='english'`, `ngram_range=(1, 2)` y `use_idf=True`.

2. Entrenamiento y evaluación de modelos

2.1. Multinomial Naive Bayes

Este modelo de aprendizaje estadístico se basa en el teorema de Bayes, y es utilizado normalmente para problemas de clasificación de texto donde la frecuencia de palabras en los documentos es importante. Este modelo asume la independencia entre las variables que representan la frecuencia y supone que la distribución de los datos es multinomial.

En este caso se entrena el modelo con el conjunto de entrenamiento filtado por stopwords, agrupando tanto por Unigramas como Bigramas y utilizando el factor IDF. Luego de entrenar el modelo se predice sobre el conjunto de test obteniendo un valor de accuracy de 0.5745, siendo el valor mencionado la proporción de predicciones acertadas sobre el total. Luego, para tener una visión más detallada del modelo, se obtiene la Matriz de Confusión (MC) mostrada en la Figura 8 donde se pueden ver el número de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos para cada clase. Los mejores resultados para el personaje Antony, siendo este último el personaje más frecuente en los datos.

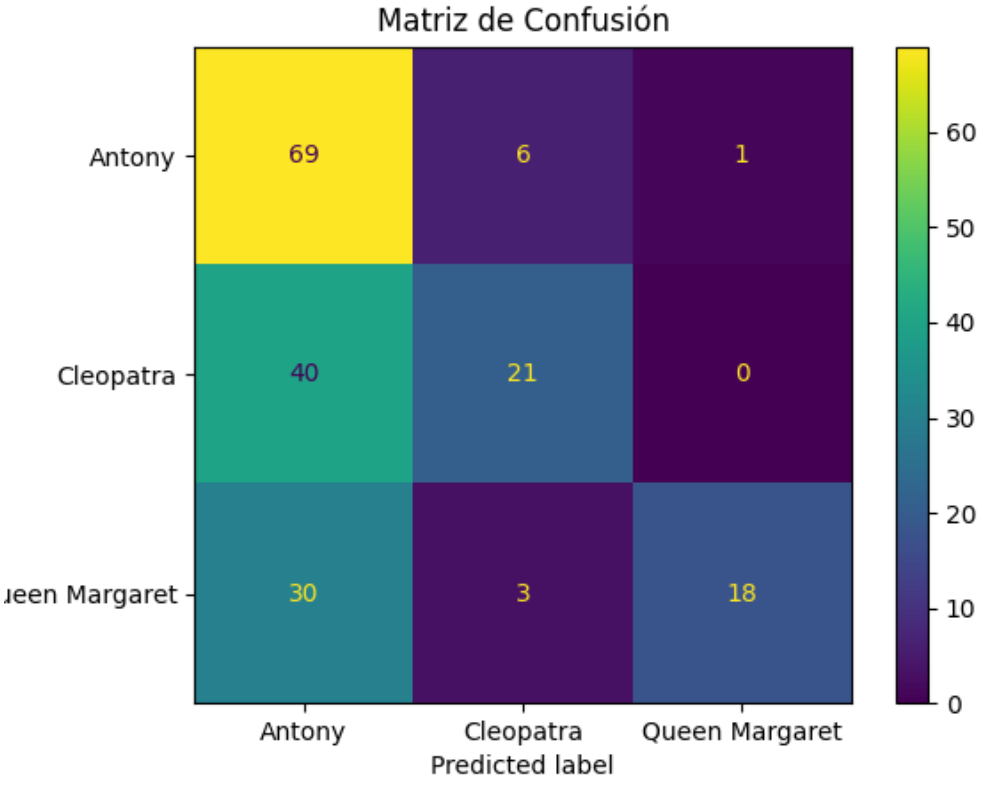

Figura 8: Matriz de Confusión para el modelo obtenido.

Además del valor de accuracy, se obtienen los valores de Precisión y Recall para

cada personaje, estos se reportan en la Tabla 2.
El valor de precisión es el cociente entre las predicciones correctas (cantidad de verdadero positivo en MC) y total de predicciones para una clase dada (columna correspondiente al personaje en MC). Por otro lado, el valor de Recall es el cociente entre las predicciones correctas y todas las predicciones verdaderas para una clase dada (fila correspondiente al personaje en MC).

Tabla 2: Valores de precisión y Recall obtenidos por personaje.

Personaje	Precisión	Recall
Antony	0.4964	0.9079
Cleopatra	0.7000	0.3443
Queen Margaret	0.9474	0.3529

En caso de solo basarse en el parámetro accuracy para medir que tan bueno es el modelo prediciendo, se estaría perdiendo información de la predicción por clase. Podría suceder que se tenga un desbalance mucho mayor en los datos y el modelo sea muy bueno prediciendo un personaje, pero muy malo prediciendo los demás y aún así el accuracy de similar a un caso con un desbalance menor.

2.2. Cross-Validation

La técnica de validación cruzada es utilizada para medir que tan buena es la capacidad de predicción de un modelo de apredizaje estadístico y seleccionar los mejores hiperparámetros (ej: stopwords, idf y n-gamas). Esta técnica es útil para mitigar el problema de sobreajuste. El proceso consiste en:

- Dividir los datos en k subconjuntos (folds) de tamaño similar.
- Entrenar el modelo k veces, utilizando en $k-1$ subconjuntos para el entrenamiento y un subconjunto para la validación.
- Los resultados de los k entrenamientos se promedian para obtener una estimación del rendimiento del modelo.

Se realiza la validación cruzada con 4 folds (`foldsn_splits=4`) manteniendo el porcentaje de muestreo en cada clase (muestreo estratificado), y `random_state=42` para tener reproducibilidad. En este caso los hiperparámetros modificados son:

- `foldsn_splits=None-'english'`
- `ngram=(1,1)`
- `idf=True-False`

Se obtiene el mejor promedio de accuracy (0.6027) con `foldsn_splits='english'`, `ngram=(1,1)` y `idf=True`.

2.3. Modelo con mejores parámetros

Utilizando los mejores hiperparámetros obtenidos en la Subsección 2.2, se entrena el modelo para evaluar sus métricas y Matriz de confusión. La Matriz de Confusión obtenida en este caso se muestra en la Figura 9. Luego, el accuracy obtenido en este caso es 0.5798 y el resto de las métricas se muestran en la Tabla 3.

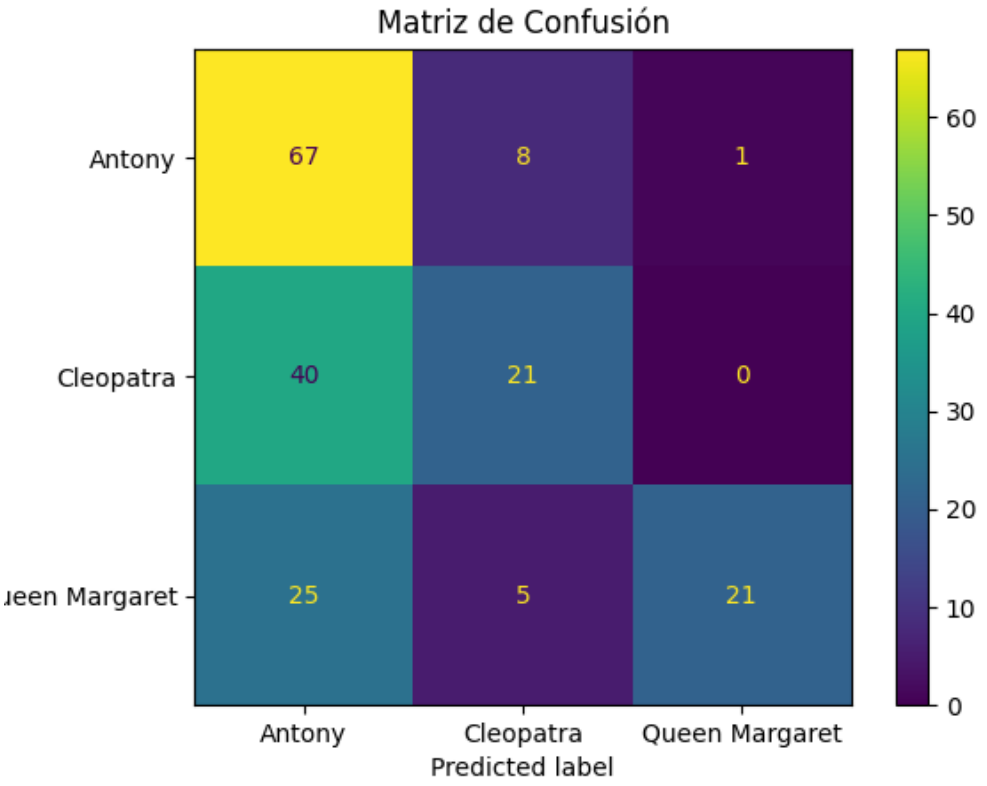

Figura 9: Matriz de Confusión para el modelo obtenido con los mejores parámetros.

Tabla 3: Valores de precisión y Recall obtenidos por personaje para el modelo con los mejores parámetros.

Personaje	Precisión	Recall
Antony	0.5076	0.8816
Cleopatra	0.6176	0.3443
Queen Margaret	0.9545	0.4118

En cuanto a las limitaciones de Bag-of-Words y TF-IDF, ambas técnicas tienen en común el no tener en cuenta el orden y relación entre las palabras, llevando a no poder contextualizar cuando realmente es necesario hacerlo. Estas herramientas son bastante básicas y pueden no capturar particularidades del lenguaje natural como la ambigüedad e interpretaciones múltiples de una misma palabra. Luego, en cuanto al modelo Multinomial Naive Bayes, el mismo es muy sensible a la calidad de los

datos de entrenamiento, si estos son incompletos o están muy desbalanceados se pueden producir resultados inexactos para ciertas clases pertenecientes al conjunto. Por último, el modelo se basa en la independencia entre variables para una clase dada, esto significa que la aparición o no de una variable no afecta a la otra, no representando lo sucedido en muchos casos donde el contexto es importante.

2.4. Support Vector Machine (SVM)

Es un algoritmo de aprendizaje supervisad muy utilizado para clasificación. El objetivo principal de SVM es encontrar un hiperplano que distinga claramente las clases de datos. Dado que el hiperplano separa el espacio de características en dos mitades (clases) se introduce el concepto de Margen, este último es la distancia entre el hiperplano y los puntos más cercanos a él de cualquiera de las clases, la idea es tener el mayor margen posible para tener una separación clara entre clases. El problema surge cuando los datos no son linealmente separables en su espacio original, la solución a esto son las funciones kernel que logran separar linealmente las clases proyectando los datos en un espacio de mayor dimensión, no siendo viable cuando se tienen grandes conjuntos de datos dada la complejidad del kernel. La elección del tipo de kernel es compleja y se necesita utilizar la validación cruzada. Dos tipos de kernel muy utilizados son el polinomial y gaussiano.

En el caso de clasificación de texto se suelen utilizar las estrategias de üno contra todos"(OvA) o üno contra uno"(OvO). En el OvA se entrena un clasificador SVM para cada clase, lo que se hace es distinguir entre una clase y todas las demás. En el caso de OVO se entrena un clasificador SVM para cada par de clases.

En este caso se aplica el modelo SVM utilizando los mismos hiperparámetros que en la Subsección 2.3 para poder comparar los resultados. La Matriz de Confusión obtenida en este caso se muestra en la Figura 9. Luego, el accuracy obtenido en este caso es 0.6543 y el resto de las métricas se muestran en la Tabla 3. Se puede ver que el accuracy obtenido es mayor que con Bayes, y el resto de las métricas obtenidas se aproximaron entre si, con un aumento de las que eran menores. Esto puede indicar que el desbalance en los datos tiene un efecto menor en los resultados.

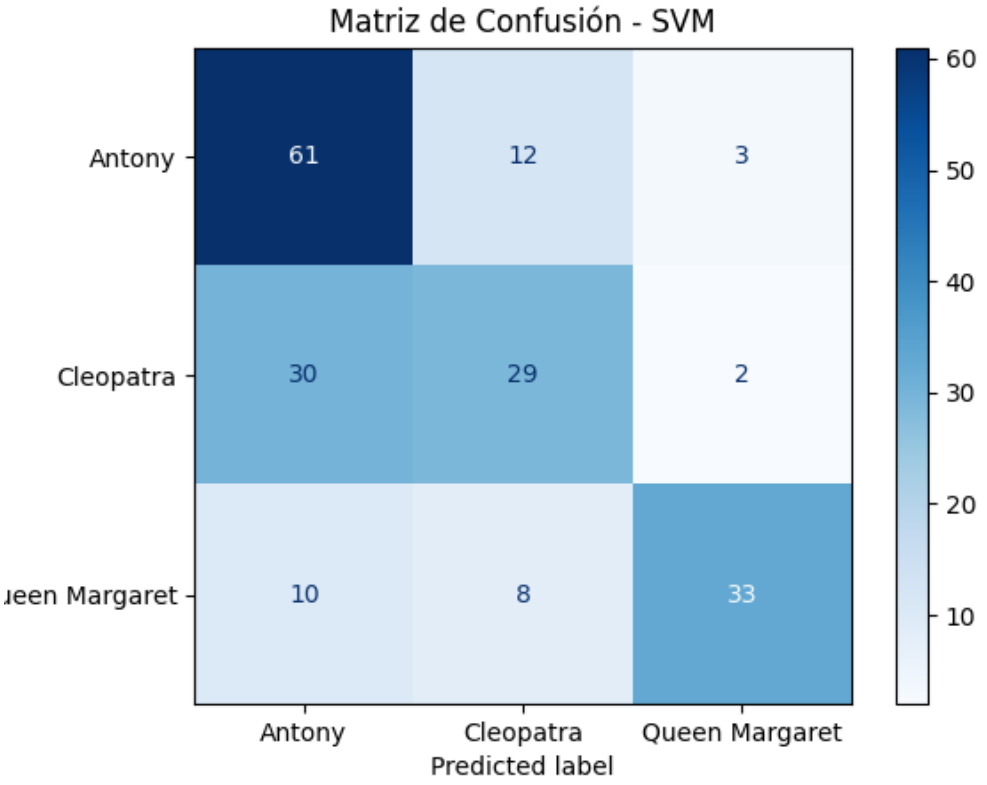

Figura 10: Matriz de Confusión obtenida con el modelo SVM.

Tabla 4: Valores de precisión y Recall obtenidos por personaje para el modelo SVM.

Personaje	Precisión	Recall
Antony	0.6040	0.8026
Cleopatra	0.5918	0.4754
Queen Margaret	0.8684	0.6471

2.5. Cambio de personaje

Con el fin de ver los efectos de un desbalance mayor en los datos, se modifica el Dataset reducido extrayendo los datos del personaje Antony e introduciendo los datos del personaje Poet. Se introduce el personaje Poet dado que de la Tarea-1 se sabe que es el personaje con mayor cantidad de párrafos, lo que se traduce en el mayor desbalance posible cambiando un solo personaje.

En la tabla Tabla 5 se muestra el porcentaje de párrafos de cada personaje respecto al total, utilizando los datos de train y test luego de haber obtenido el muestreo estratificado. Se puede ver el gran desbalance que existe dado que Poet representa casi el 70 % de los datos.

Tabla 5: Porcentaje de párrafos por personaje en los datos de train y test utilizando los datos de Poet en lugar de Antony.

Personaje	Train (%)	Test (%)
Poet	67.3	67.3
Cleopatra	17.9	17.8
Queen Margaret	14.8	14.9

Utilizando los parámetros `stopwords='english'`, `ngram_range=(1, 2)` y `use_idf=True`, se entrena el modelo Multinomial Naive Bayes con los nuevos datos. Del modelo se obtiene la Matriz de Confusión mostrada en la Figura 11, donde se puede ver que el modelo solo predice al personaje Poet. De esto se deduce que los valores de precisión para Poet y accuracy (0.6725) coinciden, el recall de Poet va a ser 1 y el resto de las métricas para los personajes restantes van a tener valor 0. Estos resultados evidencian el claro problema que tiene el modelo utilizado cuando los datos tienen un debalance muy grande, además de mostrar claramente que utilizar únicamente el valor de accuracy para medir la capacidad de predicción, puede inducir en un error en el análisis.

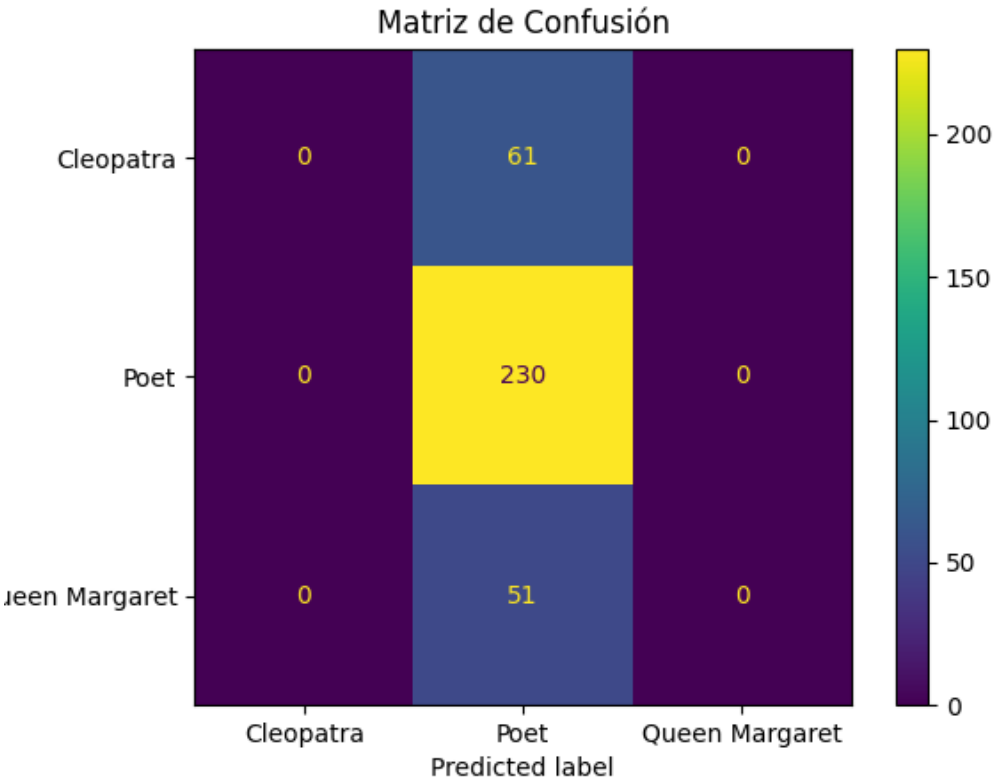

Figura 11: Matriz de Confusión obtenida con el modelo MNB utilizando Poet en el Dataset reducido.

2.6. Word2Vec

Una técnica alternativa para extraer features de texto es Word2Vec utilizada para el procesamiento del lenguaje natural. Esta técnica fue desarrollada por Google y utiliza redes neuronales para aprender representaciones vectoriales continuas de palabras. A continuación se presentan los dos modelos principales para entrenar Word2Vec:

- Skip-gram: Este modelo predice el contexto de una palabra dada. Dado un par de palabras en una oración, el objetivo es maximizar la probabilidad de predecir las palabras de contexto dentro de una ventana de tamaño definido alrededor de la palabra objetivo.
- CBOW (Continuous Bag of Words): Este modelo predice una palabra basada en su contexto. Dado un conjunto de palabras de contexto (las palabras que rodean a la palabra objetivo), el objetivo es predecir la palabra central.

Ventaja de Word2Vec respecto a Bag of Words y TF-IDF:

- Reducción de dimensionalidad: Mientras que BoW y TF-IDF producen vectores de alta dimensión (una palabra por dimensión tomando Unigramas), Word2Vec produce vectores de baja dimensión.
- Captura semántica: Word2Vec captura relaciones semánticas y sintácticas entre palabras. Esto es algo que BoW y TF-IDF no pueden hacer porque tratan cada palabra de manera independiente.
- Contextualización: Word2Vec tiene en cuenta el contexto en el que aparecen las palabras. BoW y TF-IDF ignoran el orden de las palabras y su contexto.

Una desventaja de este método respecto a BoW y TF-IDF es que produce vectores densos donde cada elemento tiene un valor, en cambio BoW y TF-IDF producen matrices dispersas (sparse) donde la mayoría de los elementos son ceros.