Universidad de la República

Facultad de Ingeniería

Maestría en Ciencia de Datos y Aprendizaje Automático

Introducción a la Ciencia de Datos 2023

Análisis de datos y aprendizaje automático en base a la obra de Shakespeare.

Ana Cortazzo
Luciana Olazábal

6 de julio de 2023

Contenido

1. Introducción

Este informe presenta métodos y resultados utilizados para el análisis y predicción de algunos aspectos relacionados a la obra de William Shakespeare[1]. Se utilizó una base de datos relacional abierta con la obra completa de William Shakespeare, disponible en este link. El objetivo del trabajo es proponer y entrenar modelos que a partir de un párrafo de la obra, sean capaces de predecir el personaje que lo dice.

El trabajo se realizó en tres etapas, iniciando con el análisis exploratorio que se presenta en la Sección 2, dónde se cargan los datos, se realiza una exploración y limpieza y se realizan visualizaciones de la obra de Shakespeare a lo largo de los años y un conteo de palabras y párrafos (considerando la obra completa y luego discriminando por trabajo). En la segunda etapa (Sección 3) se realiza un tratamiento de los datos de texto con el objetivo de extraer las características (*features*) a ser utilizadas en los algoritmos de aprendizaje automático. Finalmente, en la Sección 4 se presentan diferentes modelos utilizados para la predicción de resultados y una comparación entre ellos. La Sección 5 presenta las una discusión de los resultados y las conclusiones del trabajo.

El entorno utilizado para el desarrollo de este proyecto fue Jupyter Notebook, una aplicación de código abierto que proporciona un entorno de programación interactivo basado en lenguaje Python, que facilita la exploración de datos, y la biblioteca `scikit-learn` para los algoritmos de aprendizaje [Pedregosa et al., 2011]. Se puede acceder al código completo en el repositorio de GitHub.

2. Análisis exploratorio

2.1. Cargado, calidad y limpieza

La conexión con la base de datos se realizó utilizando el motor `SQLAlchemy` y la cadena de conexión adecuada, las tablas se importaron desde la base de datos y se almacenaron en el formato DataFrame de `Pandas` y se guardaron en archivos `.csv` en un directorio específico. Para facilitar la tarea se define la función `load_table` con todos los parámetros necesarios. Una vez cargadas las tablas se analiza la cantidad de registros y las propiedades que contiene cada una de ellas:

- `df_works`: información detallada de las obras de Shakespeare. Tiene 43 filas y 5 columnas:
  - `id` identificador de la obra
  - `Title` título de la obra (versión corta)
  - `LongTitle` titulo completo de la obra
  - `Date` año de publicación de la obra
  - `GenreType` género literario de la obra
- `df_chapters`: identifica los diferentes actos y escenas dentro de cada obra. Tiene 945 filas y 5 columnas:
  - `id` identificador del capítulo
  - `Act`número del Acto
  - `Scene` número de la escena
  - `Description` breve descrićión o título de la escena
  - `work_id` relaciona el `id` de la obra (relación con tabla `df_works`)
- `df_characters`: identifica los diferentes personajes que intervienen en las obras. Tiene 1266 filas y 4 columnas:
  - `id` identificador del personaje
  - `CharName` Nombre del o los personajes
  - `Abbrev` abreviación del nombre
  - `Description` información que describe al personaje
- `df_paragraphs`: Contiene el texto principal de las obras. Tiene 35465 y 5 columnas:
  - `id` identificador del párrafo
  - `ParagraphNum` número de párrafo dentro de la obra
  - `PlainText` texto real del párrafo
  - `character_id`indica qué personaje está hablando en el párrafo (si corresponde) (relación con tabla `df_character`)
  - `chapter_id` indica en qué capítulo (acto y escena) se encuentra el párrafo (relación con tabla `df_chapters`)

En primer lugar se hizo un relevamiento de datos faltantes (null, NA o NaN), para esto se utilizó la función `isna()` de `Pandas`. En segundo lugar se definió la función `puntuacion` que dada una columna de un DataFrame, realiza un búsqueda de los caracteres de puntuación (utilizando la cadena predefinida `string.punctuaction`) y devuelve una lista con todos los signos encontrados. Para la limpieza de datos se definió la función `clean_text`,

[1] Dramaturgo inglés (1564 - 1616). https://es.wikipedia.org/wiki/William_Shakespeare

Tabla 2.1: Ejemplo de texto antes y después de la limpieza de datos

PlainText	CleanText
[Enter DUKE ORSINO, CURIO, and other Lords; Mu...	enter duke orsino curio and other lords mu...
If music be the food of love, play on;\nGive m...	if music be the food of love play on give me...
Will you go hunt, my lord?	will you go hunt my lord
What, Curio?	what curio

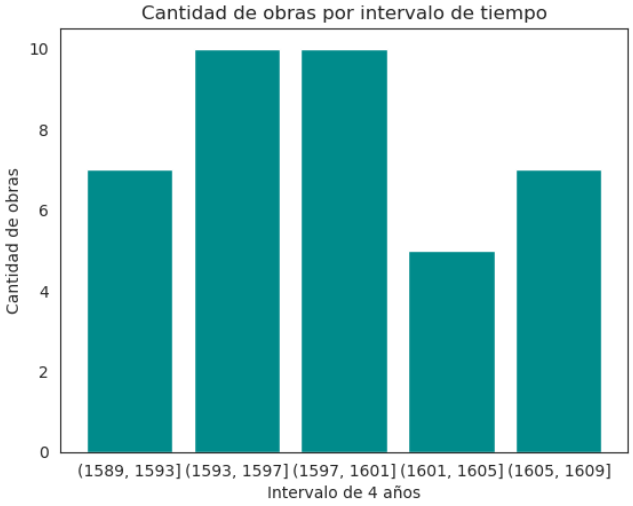


Figura 2.1: Obras publicadas en periodos de 4 años

la cuál remplaza todos los signos de puntuación por espacios, y convierte las mayúsculas a minúsculas, de esta forma nos quedamos con un texto limpio que facilita las tareas de análisis y conteo. En la tabla 2.1 se presenta, a modo de ejemplo, el antes y el después de la columna `PlainText` luego del proceso de limpieza.

2.2. Análisis y visualizaciones

¿Es posible identificar aspectos creativos de la obra de Shakespeare a partir del análisis de los datos? ¿Cuáles son las palabras que más aparecen en la obra? ¿Son representativas y características de su trabajo? Estas son algunas preguntas disparadoras para la discusión.

Si se empieza por analizar el histograma de la Figura 2.1, donde se muestra la cantidad de obras de Shakespeare a lo largo del tiempo en intervalos de 4 años, se pueden ver algunas tendencias. Por ejemplo, que su productividad en términos de producción individual de obras tiene un máximo entre los años 1593 y 1601, donde el autor llegó a escribir 20 obras en un intervalo de solamente 8 años. Esto es seguido por el período de 4 años de menor concentración de obras (5 obras en 4 años).

Al analizar el histograma, es importante tener en cuenta que las obras de Shakespeare abarcan una amplia gama de géneros, que van desde tragedias y comedias hasta novelas históricas y sonetos. Cada género puede tener una representación desigual en el histograma, lo que puede influir en la distribución de las obras a lo largo del tiempo. Para realizar un análisis más detallado en la Figura 2.2 se dividen las obras por género. Analizando la gráfica se puede observar que el autor era muy versátil, ya que logró abarcar a lo largo de su vida una variedad de géneros sumamente distintos entre sí. En sus primeros años como escritor predominan las comedias, y a medida que su carrera avanza, vemos un aumento en las tragedias, que exploran temas más profundos y oscuros, identificando un periodo en el que sólo escribió obras de este tipo (1605-1607)[2]. Tanto al comienzo como al final de su carrera literaria se ve una gran cantidad de obras históricas, probablemente influenciadas por el contexto histórico y cultural en el que vivió el autor.

[2]La etapa 1604-1608 es conocida como la de las grandes tragedias, en las que Shakespeare bucea en los sentimientos más profundos del ser humano Fernández and Tamaro [2004]

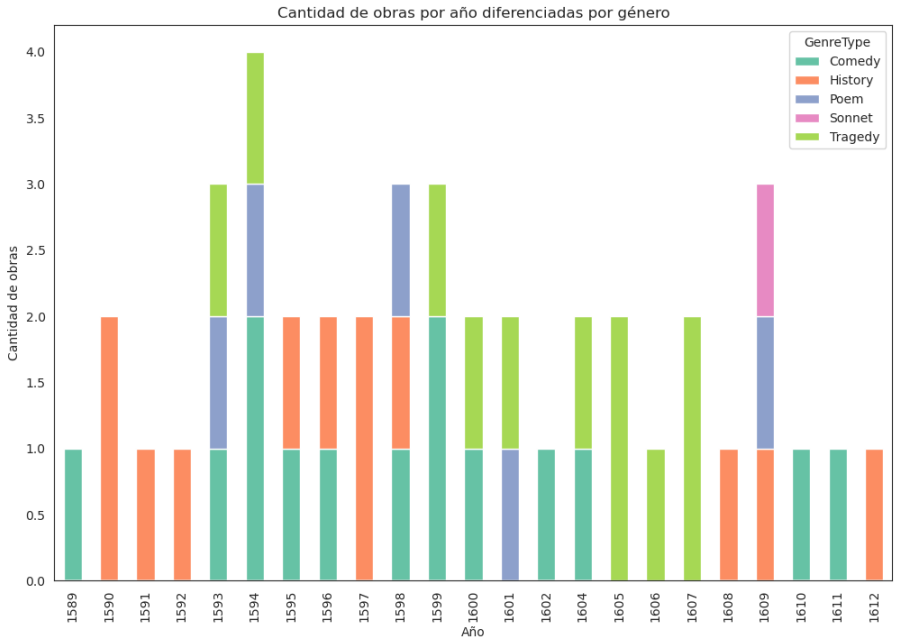


Figura 2.2: Obras publicadas por año diferenciadas por género

Observando únicamente la cantidad de obras en el tiempo, como se hizo en la Figura 2.1, no se tiene información de la carga de trabajo que significaron para el autor, pues algunas obras puede ser menos extensas que otras. Por ello, un dato interesante para determinar la evolución de la carrera literaria de este autor es determinar el número de palabras escrita por año. Esto nos puede dar una idea del volumen de trabajo que manejó el escritor a lo largo de su vida. En la Figura 5.1 (Anexo 1) se puede ver la cantidad de palabras escritas por año, 1594 fue el año de mayor volumen de trabajo en términos de palabras escritas, superando ampliamente a los demás. En este año Shakespeare escribió varias obras teatrales que se consideran importantes en su carrera como "Romeo y Julieta", "El mercader de Venecia" y "Sueño de una noche de verano".

Dentro de la exploración de datos se realizó un conteo directo de la cantidad de párrafos que dice cada personaje sin considerar en que obra aparece (Tabla 2.2) y agrupando los datos por personaje y obra (Tabla 2.3). La dirección de escenario (*stage directions*) posee un gran número de párrafos, pues se están contando todas las líneas de la dirección de escenario de todas las obras. Discriminando por obra se puede decir que el personaje con mayor cantidad de párrafos por obra es Hamlet.

Tabla 2.2: 10 personajes con mayor número de párrafos sin discriminar por obra

N.Párr.	Personaje
3751	(stage directions)
733	Poet
471	Falstaff
377	Henry V
358	Hamlet
285	Duke of Gloucester
274	Othello
272	Iago
253	Antony
246	Richard III

Tabla 2.3: 10 personajes con mayor número de párrafos por obra

N.Párr.	Personaje	Obra
358	Hamlet	Hamlet
274	Othello	Othello
272	Iago	Othello
269	Poet	Rape of Lucrece
210	Timon	Timon of athens
204	Cleopatra	Antony and Cleo.
202	Antony	Antony and Cleo.
201	Rosalind	As you like it
201	Poet	Venus and Adonis
194	Brutus	Julius Caesar

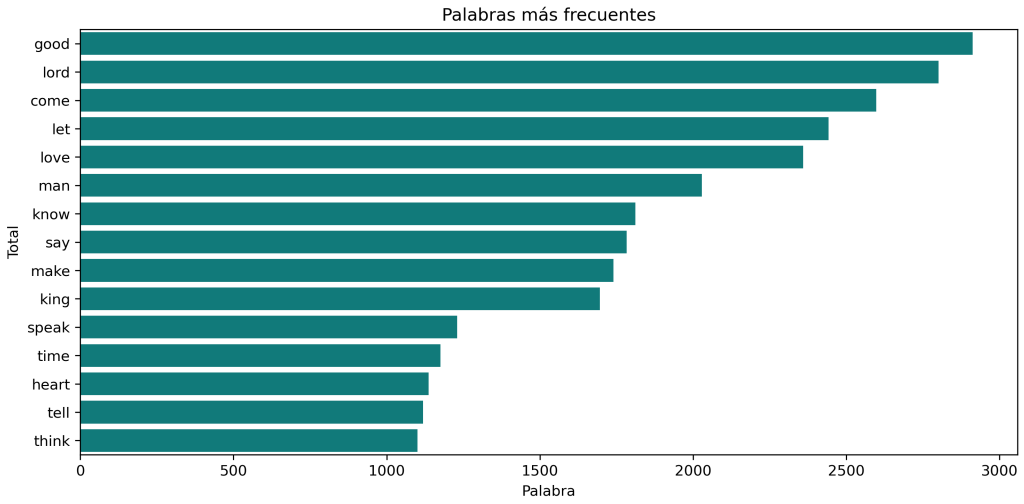


Figura 2.3: Palabras que más se repiten en la obra completa

Tabla 2.4: Las 5 palabras más frecuentes que siguen a la palabra "good"

Palabra	Conteo
lord	241
morrow	113
night	110
master	53
man	40

El análisis referido a la cantidad de palabras por personaje y las palabras más comunes se realizó en primer lugar considerando el corpus de texto sin remover las *Stop words*[3], y luego removiendo las *Stop words* utilizando una lista modificada de *Early Modern English* y *Old English*, ya que la obra se Shakespeare se ubica en esa época[4]. La Figura 2.3 muestra las 15 palabras más frecuentes en la obra de Shakespeare luego de eliminar las palabras vacías.

Las palabras "lord" y "man" pueden estar relacionadas con el contexto social y religioso en el que vivió el autor, ya que "Lord" podría hacer referencia tanto a la nobleza como a figuras divinas, y "man" podría representar la humanidad en general, no solamente referir a un hombre en particular, de la misma forma la palabra "king" se asocia al momento histórico y al contenido de algunas de sus obras.

Las palabras "say", "speak", "tell", "know" y "think" están relacionadas con el lenguaje y la comunicación. Tiene sentido que se repitan muchas veces ya que la palabra y el diálogo son elementos claves de las obras teatrales del autor.

Las palabras "heart", "tell", "think", "speak" y "make" podrían estar relacionadas con aspectos emocionales de los personajes, ya que el autor exploraba en muchas de sus obras las emociones humanas. El concepto del tiempo o "time" como aparece en el gráfico puede estar relacionado con reflexiones sobre el paso del tiempo y la experiencia humana, que también eran temas recurrentes en la obra del autor.

Si miramos ahora la palabra más frecuente "good" y analizamos el contexto en el que aparece (en este caso, indagando la palabra que le sigue) podemos deducir algunas expresiones comunes en la obra de Shakespeare. La Tabla 2.4 muestra las 5 palabras que más se repiten luego de "good". Vemos que la expresión "good lord" es la más utilizada, seguida de "good morrow"[5] y "good night".

[3] Los resultados se pueden ver en el Anexo 1.
[4] La lista utilizada fue adaptada de Clark [2018]
[5] "good morrow" = "good' morning". https://www.dictionary.com/browse/good-morrow

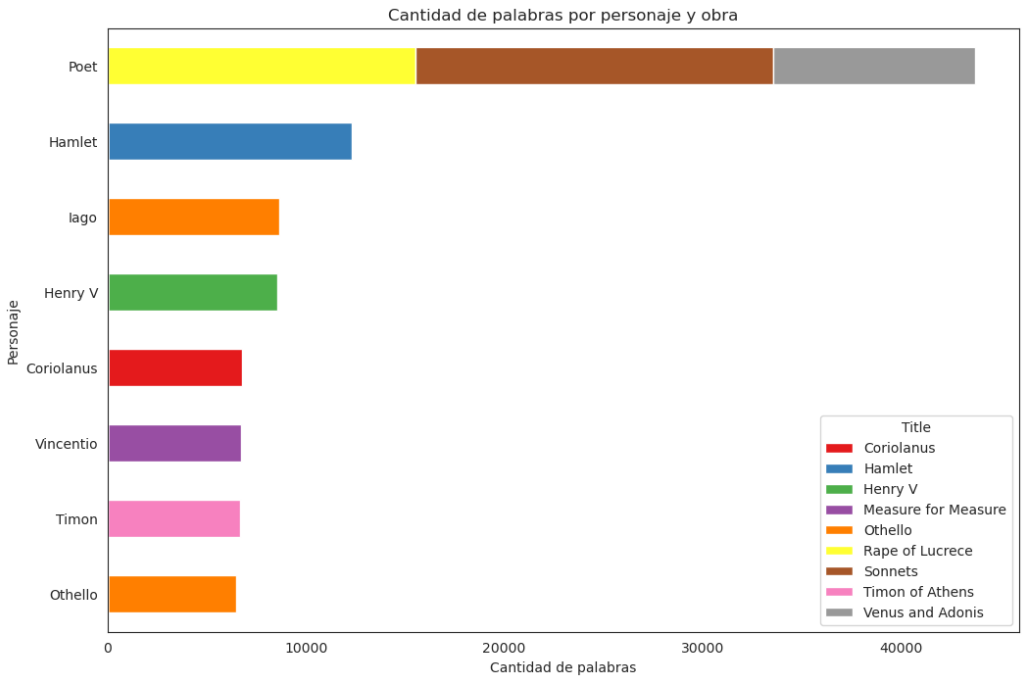


Figura 2.4: Personajes con mayor cantidad de palabras discriminados por obra

En el análisis de palabras por personaje discriminando por obra se muestra en la Figura 2.4. Hamlet es el personaje con mayor número de palabras (sin considerar el personaje *Poet* que está asociado a los poemas escritos por el autor y no necesariamente un personaje teatral). En la Figura 5.3 (Anexo 1) se presnetan los resultados cuando no se eliminan las Stop Words.

A modo de conclusión podemos enfatizar que las visualizaciones obtenidas respecto a la obra se Shakespare a lo largo de los años permitieron identificar algunos períodos claves en su trabajo, etapas dónde el género predominante es la tragedia, y otras dónde son las comedias o historias. Estos resultados están en concordancia con algunos estudios realizados dentro del campo literario. Realizar el conteo de párrafos por personaje, discriminado por obra, ayuda a tener una idea sobre la relevancia de determinados personajes o no en las obras, por ejemplo Hamlet es el personaje más relevante en la obra homónima, así como Othelo. Además, son dos de los personajes con mayor número de párrafos en la misma obra. Sin duda, los resultados con mayor potencial para el análisis de la obra son aquellos que realizan conteo de palabras. Al eliminar las *Stop Words* es posible observar elementos representativos de la escrita de Shakespeare, que se relacionan con la época histórica en la que el escritor vivió.

3. Transformaciones del texto

3.1. Representación numérica de texto

Para realizar la representación numérica del texto se parte del texto sin signos de puntuación ni contracciones obtenido luego de aplicar la función `clean_text`. Se seleccionaron 3 personajes para realizar el entrenamiento de modelos, en la Tabla 3.1 se muestra la cantidad de párrafos que corresponden a cada personaje en los datos iniciales, y qué cantidad se asignó en los conjuntos de entrenamiento (`train`) y de prueba (`test`). Se asignó el 30 % al conjunto de `test` estratificado, es decir, manteniendo la proporción del muestreo entre los personajes. En la Figura 3.1 se muestra el porcentaje relativo asignado a cada personaje en los conjuntos de `train` y `test` dónde se puede observar que la relación de párrafos por personajes en entrenamiento y prueba es casi idéntica para los tres personajes.

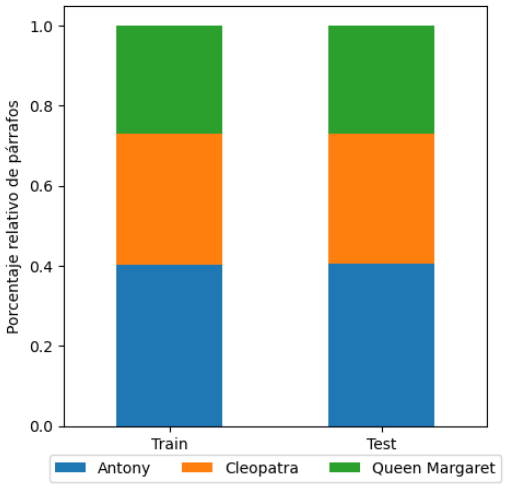


Figura 3.1: Porcentaje relativo de párrafos por personajes en conjuntos de entrenamiento y prueba

Tabla 3.1: Cantidad de párrafos por personaje en los conjuntos de entrenamiento y testeo

Personaje	**Cantidad de párrafos**		
	Dataset	**Train**	**Test**
Antony	253	177	76
Cleopatra	204	143	61
Queen Margaret	169	118	51

La representación numérica de conteo de palabras (Bag of Words) convierte texto en una matriz que representa la frecuencia de las palabras que se tiene en el conjunto de datos. Para esto, primero se divide al texto en unidades más pequeñas (palabras), para luego crear un vocabulario único a partir de todas las palabras presentes en el conjunto de datos y por último contar la frecuencia de cada palabra para crear una matriz donde las filas representan los personajes y las columnas representan las palabras únicas utilizadas en las obras. Cada entrada de la matriz indica el número de veces que una palabra es dicha por uno de los tres personajes.

La matriz resultante es una matriz dispersa (sparse matrix) porque la mayoría de las entradas son ceros. Esto se debe a que los personajes utilizan solo una fracción de todas las palabras de las obras, haciendo que la matriz resultante sea altamente dispersa y utilice estructuras de datos especializadas que solo almacenan las ubicaciones y valores no cero. Si no se trabajara con un conjunto de datos tan reducido, la matriz de conteo de palabras podría volverse muy grande y ocupar mucha memoria RAM.

Si, por ejemplo, se considerase que la única frase dicha por Anthony es "*grates me the sum*", por Cleopatra es "*if it be love indeed tell me how much*" y por Queen Margareth es "*bear with me i am hungry for revenge*", la matriz resultante es la traspuesta de la mostrada en la Tabla 3.2, sin aplicar supresión de *stop words*.

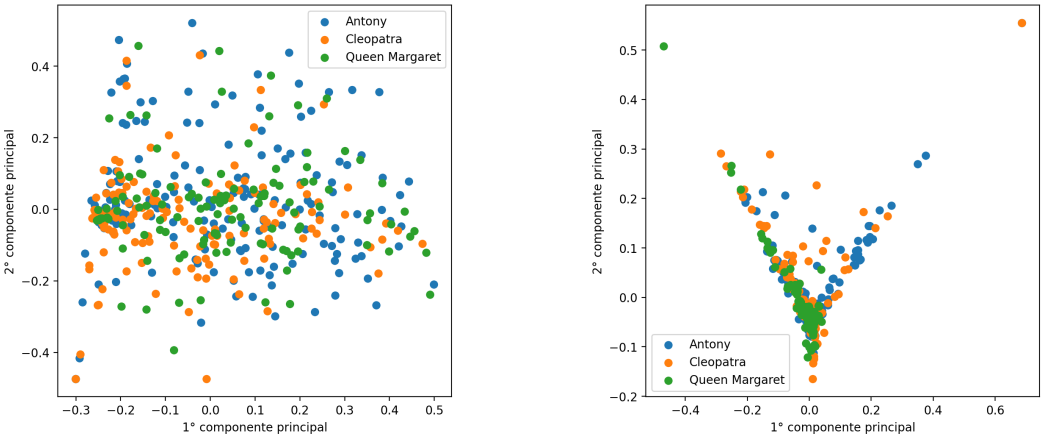
Un n-grama es una secuencia de n elementos consecutivos en un texto, donde los elementos pueden ser caracteres, palabras o hasta frases. Se utilizan en el procesamiento de lenguaje natural para capturar patrones y estructuras de texto más allá de palabras individuales.

La representación numérica Term Frequency - Inverse Document Frequency (TF-IDF) es una técnica que se utiliza para evaluar la importancia de una palabra en un texto o conjunto de textos. Por un lado, el concepto Term Frequency (TF) está asociado a la frecuencia de una palabra en un texto específico. Se calcula como el cociente entre el número de veces que aparece una palabra en un texto y el número total de palabras en ese texto. Por otro, Inverse Document Frequency (IDF) es un aforma de medir la rareza de una palabra en un conjunto de textos, calculado como el logaritmo del número total de textos dividido por el número de textos que contienen la palabra. El IDF penaliza las palabras que son comunes en todos los textos y da mayor importancia a las palabras más específicas.

En este caso, la transformación TF-IDF combina el valor de TF y el valor de IDF para obtener un peso numérico para cada palabra en cada personaje. Los términos que tienen una alta frecuencia en un personaje y una baja frecuencia en el conjunto de personajes tendrán un peso TF-IDF más alto. Esta transformación ayuda a destacar las palabras más relevantes y distintivas en el análisis de texto, ya que las palabras frecuentes en un personaje y poco comunes en otros suelen ser más informativas y descriptivas.

Tabla 3.2: Ejemplo de uso del método de representación mediante Bag of Words

	Anthony	Cleopatra	Queen Margareth
grates	1	0	0
me	1	1	1
the	1	0	0
sum	1	0	0
if	0	1	0
it	0	1	0
be	0	1	0
love	0	1	0
indeed	0	1	0
tell	0	1	0
how	0	1	0
much	0	1	0
bear	0	0	1
with	0	0	1
i	0	0	1
am	0	0	1
hungry	0	0	1
for	0	0	1
revenge	0	0	1



(a) Parámetros: `stop_words=None, ngram_range=(1,1), use_idf=False`

(b) Parámetros: `stop_words='english', ngram_range=(1,2), use_idf=True`

Figura 3.2: PCA por personaje.

3.2. Análisis de componentes principales

En el caso específico del análisis de texto, PCA aplicado a las representaciones numéricas de los textos (como la representación TF-IDF) puede proporcionar una forma de resumir la información textual en un espacio de menor dimensión, lo que facilita su visualización y análisis.

En la Figura 3.2(a) se ve el resultado del gráfico de dispersión donde cada punto representa un párrafo del conjunto de entrenamiento sin considerar las *Stop Words*. Los puntos correspondientes a los diferentes personajes se muestran con diferentes colores y los ejes son las componentes principales del PCA. La clara dispersión de los datos indica que las características o atributos utilizados para la representación numérica no son suficientes para capturar las diferencias significativas entre los personajes, o que la variabilidad en los datos es alta y no existe una estructura clara.

Al tener en cuenta dentro de los parámetros las *Stop Words* del idioma inglés, se obtiene una menor dispersión, como puede observarse en la Figura 3.2(b).

En términos de clasificación o separación de personajes, un gráfico disperso sugiere que los personajes pueden

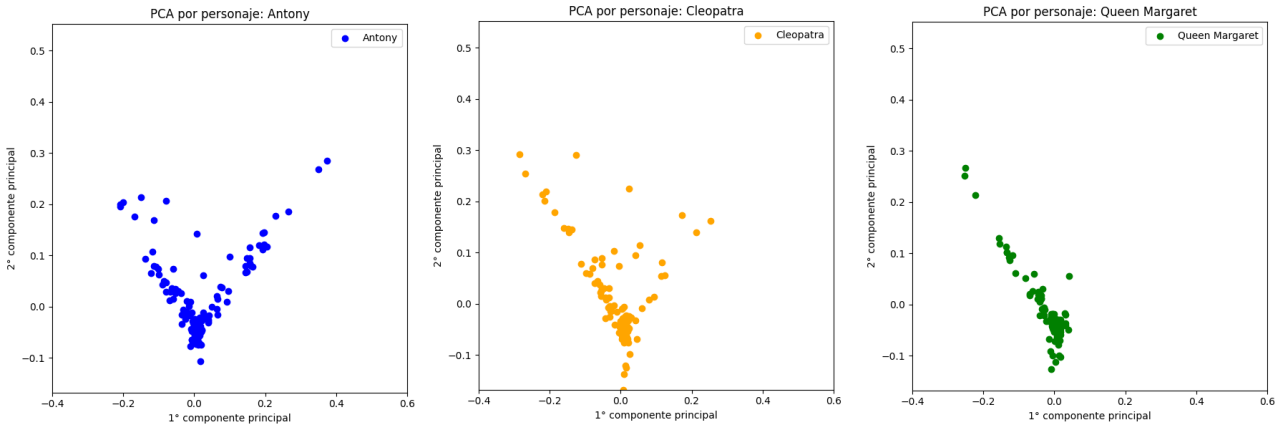
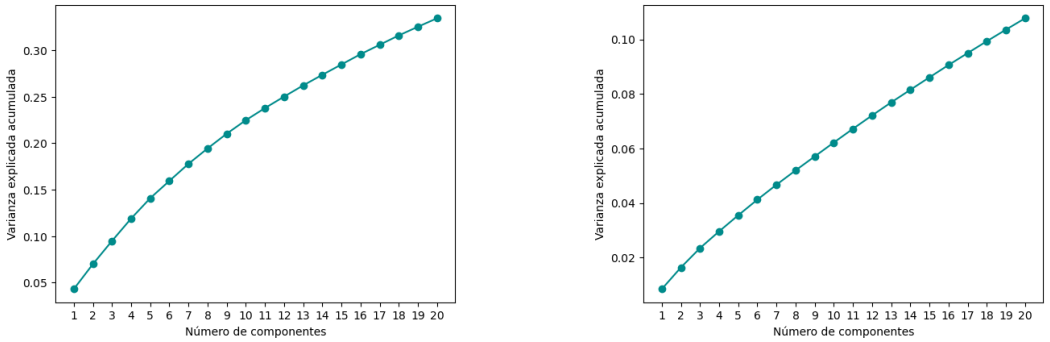


Figura 3.3: Parámetros: `stop_words='english', ngram_range=(1,2), use_idf=True`. Separado por personaje



(a) Parámetros: `stop_words=None, ngram_range=(1,1), use_idf=False`

(b) Parámetros: `stop_words='english', ngram_range=(1,2), use_idf=True`

Figura 3.4: Varianza explicada acumulada por número de componentes.

ser difíciles de distinguir o separar utilizando únicamente las dos primeras componentes principales. Es posible que se requieran más componentes o características para obtener una mejor separación entre los personajes en el espacio reducido. En la Figura 3.3 se presenta el PCA discriminado por personaje.

No solo la dispersión disminuye al aplicar las *Stop Words*, sino que también aparece una mayor concentración de puntos ubicados en una recta de pendiente negativa que pasa por el cero de la primera componente y valores negativos cercanos al cero de la segunda componente principal. Al discriminar por personaje (Figura 3.3) se hace más clara la diferencia entre los tres. Si bien todos los personajes muestran una acumulación de puntos sobre la recta negativa mencionada, Anthony mantiene una acumulación sobre otra recta de pendiente positiva, terminando por dibujar una V. Cleopatra, por su parte, mantiene cierta dispersión fuera de la recta común, pero no llega a verse como una acumulación. Queen Margareth es quien mantiene la acumulación de puntos más clara sobre la recta común.

Para determinar que sucede al agregar un mayor número de componentes, en la Figura 3.4 se muestran las varianzas explicadas acumuladas por número de componentes para el caso donde se consideran las *Stop Words* y el caso en que no. Al aumentar el número de componentes la varianza aumenta con una pendiente que va disminuyendo, esto es porque a medida que aumentan el número de componentes la varianza explicada llega a su límite de saturación, y agregar componentes no aporta a la explicación. En los dos casos analizados, considerando solo dos componentes, se obtiene una varianza explicada acumulada baja, lo que refuerza la idea de que no es posible usar solo dos componentes.

4. Entrenamiento de modelos

Para los modelos pertenecientes a la librería `scikit-learn` (*Multinomial Nive Bayes*, *Random Forest* y *K-Nearest Neighbors*) se utilizó la representación numérica establecida en la Sección 3, utilizando la estrategia de *bag of words* y *tf-idf*. Para cada modelo se seleccionaron los mejores parámetros utilizando `StratifiedKFold` o `GridSearchCV`. Para el modelo fasttext se utilizó la representación del texto requerida por el modelo, que es un archivo de texto donde cada línea contiene una etiqueta seguida del texto correspondiente.

Para evaluar la calidad de los modelos de clasificación entrenados se utilizaron los siguientes parámetros:

Accuracy (acierto) Es la proporción de instancias clasificadas correctamente sobre el total de instancias. Sea V_p y V_n los verdaderos positivos y negativos y F_p F_n los falsos positivos y negativos respectivamente, se puede calcular el accuracy como:

$$acc = \frac{V_p + V_n}{V_p + V_n + F_p + F_n}$$

Precision (Precisión) Es la proporción de instancias clasificadas correctamente como positivas (verdaderos positivos) sobre el total de instancias clasificadas como positivas (verdaderos positivos más falsos positivos). Cuanto mayor sea la precisión, mejor será el modelo en la identificación de instancias positivas de manera precisa.

$$precision = \frac{V_p}{V_p + F_p}$$

Recall Es la proporción de instancias positivas clasificadas correctamente (verdaderos positivos) sobre el total de instancias positivas (verdaderos positivos más falsos negativos).

$$recall = \frac{V_p}{V_p + F_n}$$

F1-score (medida-F) Es la media armónica entre precisión y recall, e intenta combinar ambas en un sólo número. Se determina mediante:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Matriz de confusión Una matriz cuadrada que muestra el número de instancias clasificadas correctamente e incorrectamente para cada clase. Las filas contienen información sobre el valor verdadero y las columnas el valor predicho por el modelo. La diagonal muestra las predicciones correctas.

4.1. *Multinomial Nive Bayes*

El modelo Multinomial Naive Bayes (MNB) es un algoritmo de clasificación que se basa en el teorema de Bayes. La suposición principal es que las características son independientes entre sí, lo cual puede ser una suposición simplificada pero útil en muchos casos. Durante el entrenamiento, el modelo aprenderá las probabilidades de ocurrencia de cada característica en cada clase.

En primer lugar se entrenó el modelo MNB con en conjunto de entrenamiento transformado con los parámetros stop_words:None, ngram:(1,1), idf:False, y se evaluó con el conjunto de test transformado con los mismos parámetros. La Figura 4.1 muestra la matriz de confusión obtenida y la Tabla 4.1 muestra los parámetros de rendimiento del modelo. Se puede observar una tendencia a predecir los párrafos al personaje Antony, Queen Margaret tiene un valor de recall muy bajo, esto nos dice que el modelo no tiene un buen rendimiento en la predicción de instancias asociadas a este personaje.

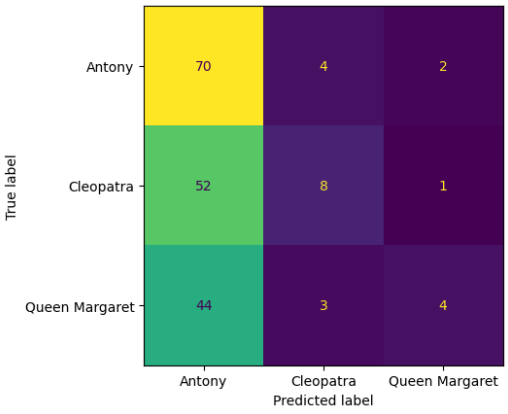


Figura 4.1: Matriz de confusión del modelo MNB entrenado con los parámetros stop_words:None, ngram:(1,1), idf:False.

Tabla 4.1: Reporte de rendimiento del modelo MNB

Personaje	Precision	Recall	f1-score
Antony	0.42	0.92	0.58
Cleopatra	0.53	0.13	0.21
Queen Margaret	0.57	0.08	0.14
		Accuracy	0.44

A partir de la diagonal de la matriz de confusión obtenemos el accuracy del modelo, ya que en la diagonal se muestran las instancias verdaderas, es decir, las que fueron clasificadas correctamente. La suma de la diagonal divido el total de instancias da el valor del accuracy, para la matriz de la Figura 4.1 acc= (70+8+4)/188 =0.44. La precisión para cada personaje también pude obtenerse a partir de la matriz, en este caso observando las columnas que muestran el total de instancias clasificadas como positivas, en este caso, para Antony tenemos precision = 70/(70+52+44) = 0.42. Las filas de la matriz muestran el total de instancias positivas y nos dan la información de recall, en este caso, para Antony recall = 70/(70+4+2) = 0.92.

Para determinar los mejores parámetros a ser usados en el modelo MNB se realizó validación cruzada. La técnica de validación cruzada (*cross validation*) consiste en dividir el conjunto de datos de entrenamiento en k partes, luego se utilizan $k-1$ partes para entrenar, y la restante para evaluar el modelo, sin necesidad de usar los datos de test en esta etapa. Se puede seleccionar el valor de k adecuado para subdividir los datos. En este caso se fijó $k = 4$ y un conjunto e 6 parámetros que varían las características de `stop words`, `n-gramas` y `tf-idf` del conjunto de datos.

La Figura 4.2 muestra la media del valor de accuracy en cada uno de los modelos entrenados (del 1 al 6) y la variabilidad de las mismas en todos los splits realizados (en este caso 4). Se puede observar que los modelos 3 y 4 presentan la menor variabilidad entre splits, mientras que los modelos 5 y 6 tienen los valores medios de accuracy más altos. Para entrenar el modelo nuevamente con los parámetros óptimos se seleccionaron las siguientes características: stop_words:'english', ngram:(1,1), idf:True (modelo 5).

En la Figura 4.3 se presenta la matriz de confusión para el MNB entrenado con los parámetros que optimizan el valor de accuracy y la Tabla 4.2 presenta el reporte de rendimiento de este modelo. Si comparamos los resultados obtenidos con los resultados de la Figura 4.1 es posible observar que no solo mejoró el valor de accuracy, sino que también la precisión y recall para cada clase (personajes) mejoraron.

En casos de desbalance de datos, donde una clase tiene muchas más instancias que otras, en este caso el personaje Antony tiene más párrafos que Cleopatra y Queen Margaret, el modelo puede tener un sesgo hacia la clase mayoritaria, es decir que tiende a predecir con mayor frecuencia la clase mayoritaria. Si solo observamos el valor de accuracy, este podría dar una percepción poco acertada sobre el comportamiento del modelo, ya que puede mostrar una alta tasa de acierto general debido al alto número de predicciones correctas para la clase mayoritaria, pero ocultar un rendimiento bajo en la clasificación de las clases minoritarias, que se pueden observar mirando la precisión y recall.

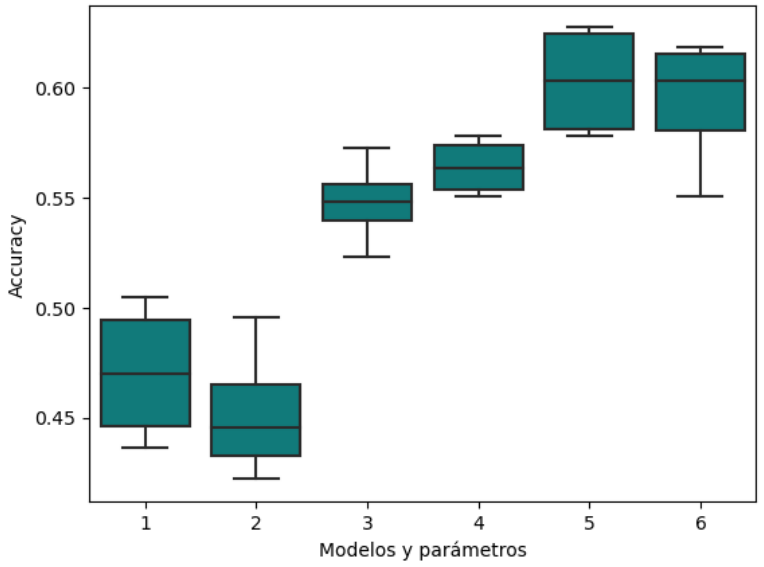


Figura 4.2: Comparación de métricas de accuracy para el modelo **Multinomial Naive Bayes** obtenidas mediante validación cruzada (`cross-validation`). Parámetros: `1:stop_words:None, ngram:(1,2), idf:True, 2:stop_words:None, ngram:(1,1), idf:False, 3:stop_words:'english', ngram:(1,2), idf:True, 4:stop_words:'english', ngram:(1,2), idf:False, 5:stop_words:'english', ngram:(1,1), idf:True, 6:stop_words:'english', ngram:(1,1), idf:False`

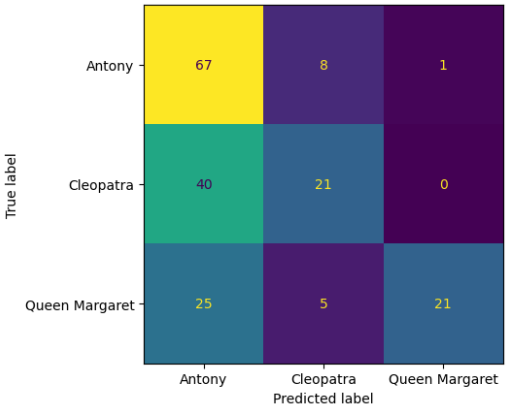


Figura 4.3: Matriz de confusión del modelo MNB entrenado con los parámetros óptimos.

Tabla 4.2: Reporte de rendimiento del modelo MNB

Personaje	Precision	Recall	f1-score
Antony	0.51	0.88	0.64
Cleopatra	0.62	0.34	0.44
Queen Margaret	0.95	0.41	0.58
		Accuracy	0.58

4.2. *Random Forest*

El método de clasificación *Random Forest* (RF) se basa en el concepto de ensamble, que implica combinar las predicciones de varios modelos individuales para obtener una predicción más precisa y robusta. Los modelos individuales utilizados en este caso son *árboles de decisión*[6]. Se crea un conjunto de árboles de decisión inde-

[6]De forma resumida es posible explicar el funcionamiento de un árbol de decisión de la siguiente manera: en cada nodo del árbol, se selecciona una característica y se define un umbral, luego busca la característica y el umbral que mejor separan los datos en términos de las etiquetas (en este caso de los personajes); con la característica y el umbral seleccionados, el árbol de decisión divide los datos en dos subconjuntos en cada nodo, uno para los valores que cumplen la condición y otro para los que no la cumplen, este proceso de división se repite recursivamente hasta que se cumpla un criterio de parada, como alcanzar una profundidad máxima o un número mínimo de instancias en un nodo; finalmente cuando se alcanza una hoja del árbol (un nodo terminal), se asigna una

pendientes para formar el Random Forest. Cada árbol se entrena con una muestra aleatoria del conjunto de entrenamiento y se selecciona un subconjunto aleatorio de características en cada división del árbol. Durante la construcción de cada árbol, se busca la mejor división en cada nodo utilizando criterios como la entropía o la impureza de Gini. Una vez que se han construido todos los árboles, se realiza la clasificación de una instancia de texto desconocida. Cada árbol emite una predicción de clasificación para la instancia de texto, y se utiliza un enfoque de "voto mayoritario" para determinar la clase final de la instancia. Por ejemplo, si la mayoría de los árboles clasifican la instancia como "Cleopatra", se considera que la instancia pertenece al personaje "Cleopatra".

Se utilizó el `RandomForestClassifier` de `sklearn.ensemble` con los datos de entrenamiento con los mismo parámetros que el modelo MNB óptimo, es decir: stop_words:'english', ngram:(1,1), idf:True. A través de `sklearn.model_selection` se utilizó `GridSearchCV` para determinar los mejores valores de número de árboles en el bosque, profundidad máxima de cada árbol, muestras requeridas para dividir un nodo y muestras requeridas en cada hoja.

```
[ ]: param_grid = {
         'n_estimators': [50, 100, 200],
         'max_depth': [5, 10, 15],
         'min_samples_split': [2, 5, 10],
         'min_samples_leaf': [1, 2, 4],  }
```

Finalmente se entrenó el modelo con los siguientes parámetros:

```
[ ]: clasificador = RandomForestClassifier(n_estimators=100, random_state=42,␣
     ↪criterion='entropy', max_depth=15, min_samples_leaf=2, min_samples_split=10)
     ranForest_clf = clasificador.fit(X_train_tf, y_train)
```

La Figura 4.4 muestra la matriz de confusión obtenida cuando se predice sobre el conjunto de test y la Tabla 4.3 presenta el reporte de rendimiento sobre estas predicciones.

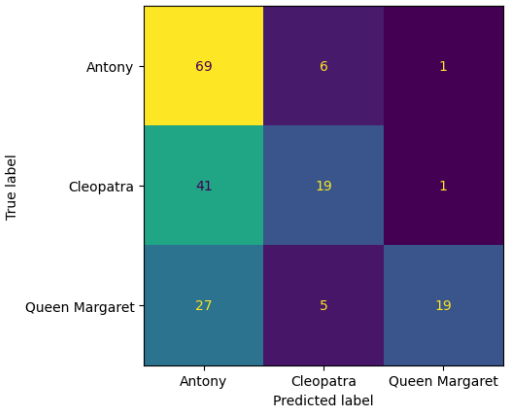


Figura 4.4: Matriz de confusión del modelo Random Forest

Tabla 4.3: Reporte de rendimiento del modelo RF

Personaje	Precision	Recall	f1-score
Antony	0.50	0.93	0.65
Cleopatra	0.69	0.30	0.41
Queen Margaret	0.95	0.35	0.51
		Accuracy	0.57

4.3. *K-Nearest Neighbors*

Dado un número entero positivo k y una observación de prueba x_0 , el clasificador *K-Nearest Neighbors* identifica primero los k puntos de los datos de entrenamiento más cercanos a x_0 (llamados k-vecinos). Una vez identificados los k-vecinos más próximos, el algoritmo cuenta cuántos de los k-vecinos más próximos pertenecen a determinada clase y divide ese recuento por k. Esta fracción representa la probabilidad estimada de la clase dados los puntos

etiqueta de clasificación (en este caso un personaje).

vecinos. Se supone que los puntos cercanos entre sí en el espacio de características tienden a pertenecer a la misma clase [James et al., 2021].

Se utilizó el `KNeighborsClassifier` de `sklearn.neighbors` con los datos de entrenamiento con los mismo parámetros que los modelos anteriores, es decir: stop_words:'english', ngram:(1,1), idf:True. A través de `sklearn.model_selection` se utilizó `GridSearchCV` con los parámetros k (número de vecinos), `weights` que controla cómo se ponderan los vecinos más cercanos en la predicción y `p` que determina la métrica utilizada para calcular la proximidad entre los puntos.

```
[ ]: param_grid = {
         'n_neighbors': range(1, 11),  # Rango de n_neighbors de 1 a 10
         'weights': ['uniform', 'distance'],
         'p': [1, 2], }
```

Finalmente se entrenó el modelo con los siguientes parámetros:

```
[ ]: clasificador2 = KNeighborsClassifier(n_neighbors=2, p=2, weights='uniform')
     KNeig_clf = clasificador2.fit(X_train_tf, y_train)
```

La Figura 4.5 muestra la matriz de confusión obtenida cuando se predice sobre el conjunto de test y la Tabla 4.4 presenta el reporte de rendimiento sobre estas predicciones.

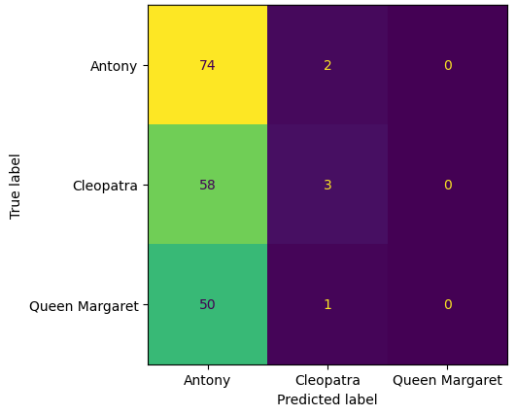


Figura 4.5: Matriz de confusión del modelo k-Neighbors

Tabla 4.4: Reporte de rendimiento del modelo KN

Personaje	Precision	Recall	f1-score
Antony	0.41	0.97	0.57
Cleopatra	0.50	0.05	0.09
Queen Margaret	0	0	0
		Accuracy	0.41

4.4. Modelo extra: `fasttext`

FastText es un modelo de aprendizaje automático que tiene la capacidad de generar representaciones de palabras considerando también las subpalabras (n-gramas de caracteres o *word-n-grams*) que las componen. Se basa en la idea de que una palabra puede ser representada por la suma de las representaciones de sus word-n-grams. Cada word-n-grams de caracteres tiene su propia representación vectorial, y la representación vectorial de la palabra se obtiene sumando las representaciones de las subpalabras. El algoritmo de entrenamiento de FastText también se basa en el uso de *bag of words*, que almacena las ocurrencias de las palabras y sus n-gramas de caracteres en el corpus de entrenamiento.

Los datos utilizados para entrenamiento y test utilizan la representación del texto requerida por el modelo, que es un archivo de texto donde cada línea contiene una etiqueta seguida del texto correspondiente. La división entrenamiento/test se realizó con la misma proporción que los modelos anteriores y también estratificada. Los parámetros usados para entrenar el modelo fueron epoch=100[7] y wordNgrams=2. La Figura 4.6 muestra la

[7] Se refiere al número de veces que se recorre todo el conjunto de datos durante el entrenamiento del modelo

matriz de confusión obtenida cuando se predice sobre el conjunto de entrenamiento y la Tabla 4.5 muestra las métricas consideradas para la evaluación del modelo.

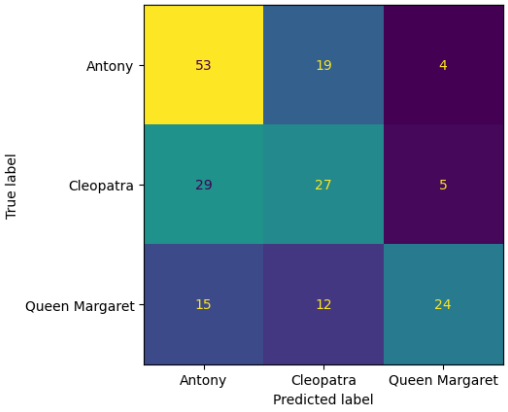


Figura 4.6: Matriz de confusión del modelo fasttext

Tabla 4.5: Reporte de rendimiento del modelo fasttext

Personaje	Precision	Recall	f1-score
Antony	0.54	0.67	0.60
Cleopatra	0.45	0.46	0.46
Queen Margaret	0.72	0.45	0.55
		Accuracy	0.54

4.5. Comparación de modelos

La Figura 4.7 presenta una comparación de los valores de accuracy, precisión promedio y recall promedio para los 4 modelos entrenados. El modelo RF presenta el mayor valor de precisión promedio, es el mejor modelo en la identificación de instancias positivas de manera precisa, pero no presenta tan buenos resultados de accuracy y recall promedio, siendo el modelo MNB quién presenta los mejores valores en promedio para estas métricas. Se puede observar que el modelo k-neighbors presenta las métricas más bajas, un resultado esperado si consideramos el análisis de PCA que nos muestra que las clases (es decir, los personajes) no presentan una separación considerable entre sí, lo que lleva a pensar que un análisis de clustering no era adecuado con los datos disponibles.

Respecto al modelo fasttext es posible observar que, si bien no presenta las mejores métricas, la complejidad del algoritmo es menor en comparación a los otros modelos, lo que puede implicar una ventaja. Además, el análisis de fasttext se realizó sin eliminar las stop words, lo que sin duda puede afectar el rendimiento del modelo.

Si bien los modelos basados en *bag of words* y *tf-idf* son ampliamente utilizados en el análisis de texto, también tienen algunas limitaciones importantes a tener en cuenta:

- tratan cada palabra de manera independiente y no consideran el orden ni la estructura gramatical de las palabras. Esto puede llevar a la pérdida de información contextual y a una comprensión limitada del significado real del texto. La información sobre la estructura y la coherencia de las frases se pierde en la representación *bag of words* y *tf-idf*.
- la presencia de stop words puede tener un impacto negativo en el rendimiento del modelo, aumentando la cantidad de elementos de la *bag of words*.
- requieren construir un vocabulario a partir de todas las palabras presentes en los datos de entrenamiento. Si el corpus de texto es grande, el tamaño del vocabulario también puede ser grande, lo que puede requerir más recursos de memoria. A medida que la cantidad de texto aumenta, la matriz dispersa que contiene la información de las características aumenta en dimensiones, siendo más costoso en términos de tiempo y memoria.

Una técnica alternativa para extraer características de texto es el uso *word embeddings* (incrustación de palabras). Estos modelos buscan representar las palabras en un espacio continuo de baja dimensión para capturar la información semántica y sintáctica, permiten una representación más rica y flexible de las palabras en comparación con enfoques tradicionales, lo que puede conducir a mejoras en tareas como la clasificación de texto,

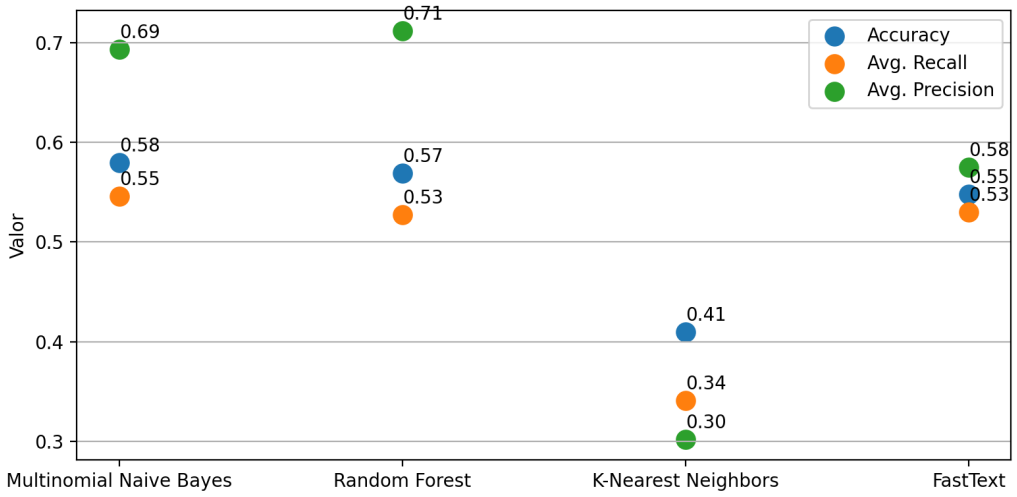


Figura 4.7: Comparación de métricas de accuracy, precisión promedio y recall promedio para los 4 modelos entrenados.

el análisis de sentimientos y la recuperación de información [Li and Yang, 2018].

La idea principal de esta técnica es capturar la semántica y las relaciones entre las palabras en función de su contexto de aparición en el corpus de texto. Los modelos aprenden a asignar vectores numéricos a las palabras de manera que las palabras con contextos similares tengan representaciones similares en el espacio vectorial.

La principal diferencia entre *word embeddings* y los enfoques basados en *bag of words* y *tf-idf* se da en la representación de las palabras, ya que los modelos basados en *bag of words* representan cada palabra de manera independiente y los modelos basados en *word embeddings* capturan las relaciones y la semántica de las palabras en función de su contexto.

4.6. Entrenamiento de modelos con otros personajes

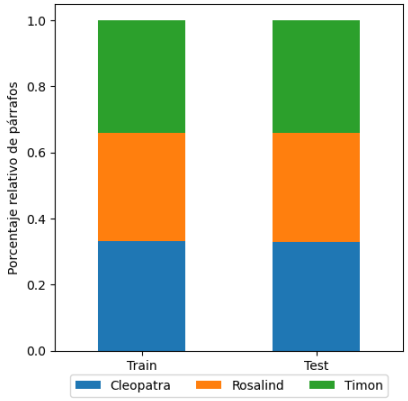
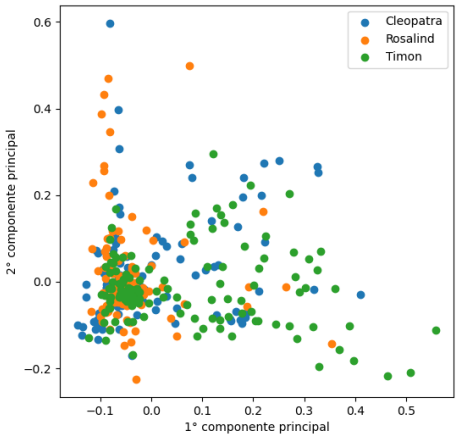


Figura 4.8: Porcentaje relativo de párrafos por personajes en conjuntos de entrenamiento y prueba

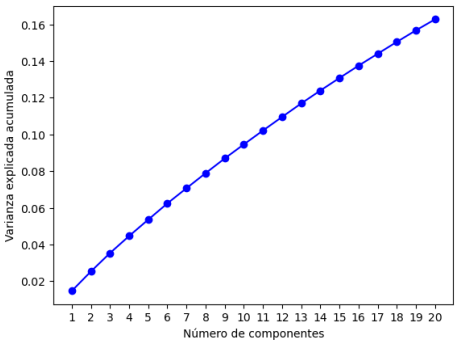
Tabla 4.6: Cantidad de párrafos por personaje en los conjuntos de entrenamiento y testeo

Personaje	Cantidad de párrafos		
	Dataset	Train	Test
Timon	210	147	63
Cleopatra	204	143	61
Rosalind	201	140	61

Con el objetivo de identificar de que forma el balance o desbalance de datos asociados a cada personaje puede afectar en el rendimiento de los modelos entrenados se seleccionaron tres personajes que presentan un número de párrafo similar: Cleopatra, Rosalind y Timon. En la Tabla 4.6 se muestra la cantidad de párrafos que



(a) PCA por personaje



(b) Varianza explicada acumulada por número de componentes

Figura 4.9: PCA y varianza explicada acumulada

corresponden a cada personaje en los datos iniciales, y qué cantidad se asignó en los conjuntos de entrenamiento (`train`) y de prueba (`test`). Se asignó el 30 % al conjunto de `test` estratificado, es decir, manteniendo la proporción del muestreo entre los personajes. En la Figura 4.8 se muestra el porcentaje relativo asignado a cada personaje en los conjuntos de `train` y `test` dónde se puede observar que la relación de párrafos por personajes en entrenamiento y prueba es casi idéntica para los tres personajes.

Luego de la división de conjunto en datos para entrenamiento y testeo se realizó la extracción de características cómo explicada en el Sección 3. Se realizó un PCA con los parámetros: stop_words='english', ngram_range=(1,1), use_idf=True, la Figura 4.9 presenta el PCA por personaje y la varianza explicada acumulada a medida que aumenta el número de componentes. Al igual que en el conjunto de datos anterior, se observa que no es posible representar el conjunto de datos usando dolo dos componentes y que los datos no presentan una separación adecuada en el espacio.

Luego de la transformación del texto y el análisis de PCA se seleccionaron los tres modelos con mejores resultados en la parte anterior para entrenar[8]: *Multinomial Nive Bayes*, *Random Forest* y `fasttext`. La búsqueda de los mejores parámetros para entrenar los modelos se realizó de la misma forma que antes (explicado en la Sección 4). En la Figura 4.10 se muestra la matriz de confusión obtenida para cada uno de los modelos entrenados. Es posible observar que los resultados obtenidos con este conjunto de datos es sensiblemente mejor en relación al conjunto de datos anterior. El valor de accuracy de los tres modelos fue superior, y se aumentó el valor de recall promedio también. En cambio, para RF y MNB el valor de precisión promedio disminuyó, es decir, se pierde precisión en la identificación de instancias de manera precisa. Estas métricas se muestran el la Figura 4.11.

Comparando estos resultados con los obtenidos para el conjunto anterior (con los personajes Antony, Cleopatra y Queen Margaret) podemos observar que una proporción desigual de ejemplos entre las diferentes clases en un conjunto de datos puede traer varios problemas en el análisis de texto. Cuando hay un desbalance significativo en las clases (por ejemplo Antony respecto a los otros dos personajes), los modelos tienden a favorecer la clase dominante (sesgo). Esto se debe a que el modelo puede lograr un valor alto de accuracy simplemente prediciendo la clase dominante en la mayoría de los casos y la precisión en las clases minoritarias puede ser baja.

Una técnica común para abordar el desbalance de datos es el muestreo estratégico, que implica sobre-muestreo de las clases minoritarias y/o sub-muestreo de las clases mayoritarias. Estas técnicas buscan equilibrar la proporción de ejemplos de entrenamiento entre las clases para mejorar el rendimiento del modelo en la clasificación de las clases minoritarias.

[8]El modelo k-neighbors no se consideró en esta parte.

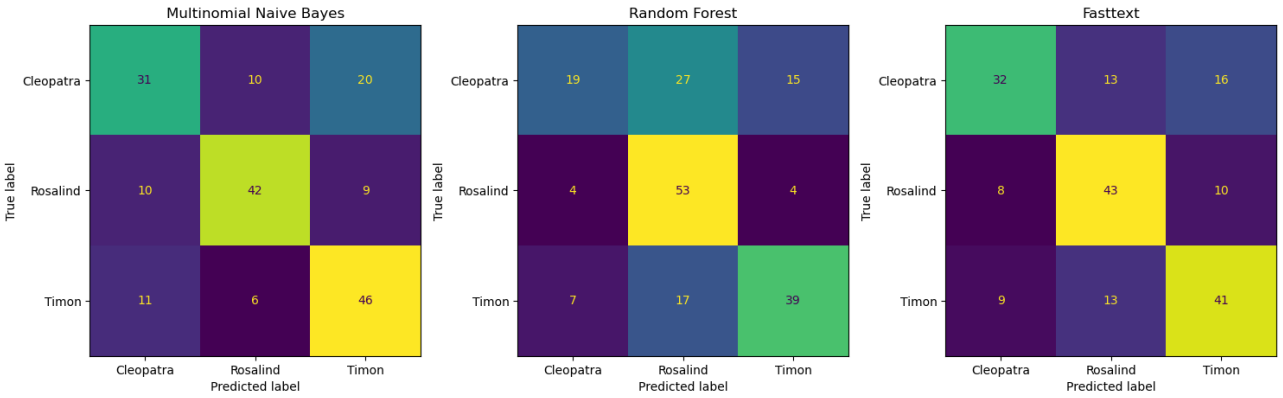


Figura 4.10: Matriz de confusión obtenida con los datos de test para los 3 modelos entrenados.

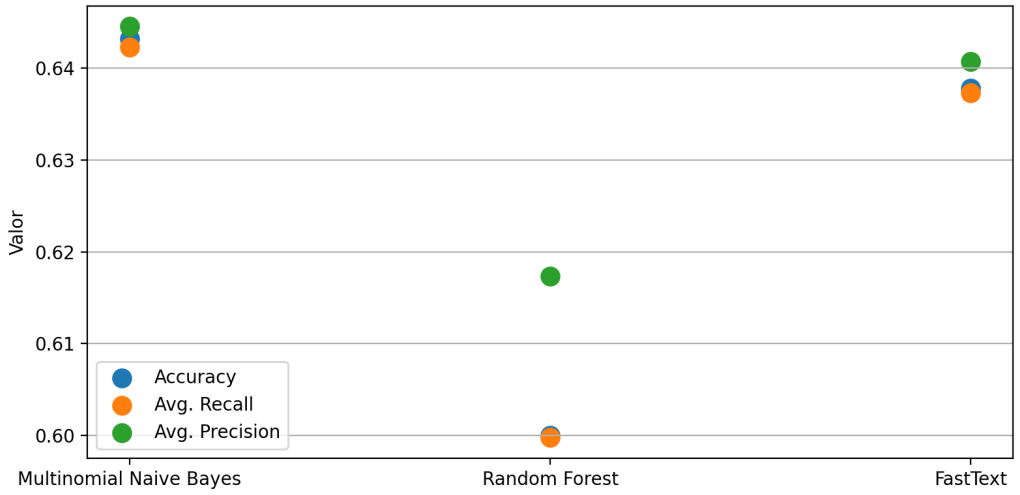


Figura 4.11: Comparación de métricas de accuracy, precisión promedio y recall promedio para los 3 modelos entrenados.

5. Discusión y conclusiones

La representación numérica del texto con la técnica de *Bag of words* y *tf-idf*, si bien mostró resultados aceptables en este ejemplo, tiene limitaciones y puede llevar a una pérdida de información contextual, así como un aumento considerable en la memoria requerida si aumenta el corpus del texto. Fue posible observar que si se eliminan las *Stop words* los resultados mejoran considerablemente. Respecto al PCA, es importante tener en cuenta que la interpretación de las componentes principales en el contexto del análisis de texto puede ser más desafiante que en otros dominios de datos más estructurados. En este caso quedó en evidencia que no es posible usar dos componentes para representar el texto y que las clases no presentan una separación adecuada.

Los diferentes modelos entrenados mostraron ventajas y desventajas, en particular, el MNB mostró ser el más adecuado para predecir el personaje a partir del párrafo. La comparación de modelos debe realizarse considerando varias métricas, y la matriz de confusión aporta información relevante para la evaluación y comparación.

Considerar solo el valor de accuracy al evaluar modelos de clasificación de texto puede tener limitaciones, en particular cuando el conjunto de datos presenta un desbalance significativo entre las clases. Si una clase está sobrerrepresentada en comparación con otras, un modelo puede obtener un alto accuracy simplemente al predecir la clase mayoritaria en la mayoría de los casos, sin tener un rendimiento satisfactorio en las clases minoritarias. Esto puede ser problemático en aplicaciones donde el objetivo es identificar correctamente todas las clases, no

solo la clase dominante.

Referencias

Michael Clark. *An Introduction to Text Processing and Analysis with R*, 2018. URL https://m-clark.github.io/text-analysis-with-R/. Accedido el 14 de mayo de 2023.

Tomás Fernández and Elena Tamaro. *Biografia de William Shakespeare. En Biografías y Vidas. La enciclopedia biográfica en línea [Internet]*, 2004. URL https://www.biografiasyvidas.com/biografia/s/shakespeare.htm. Accedido el 15 de mayo de 2023.

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning with Applications in R*, volume Second Edition. Springer, 2021.

Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. *Guide to big data applications*, pages 83–104, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Anexo 1 - Análisis exploratorio

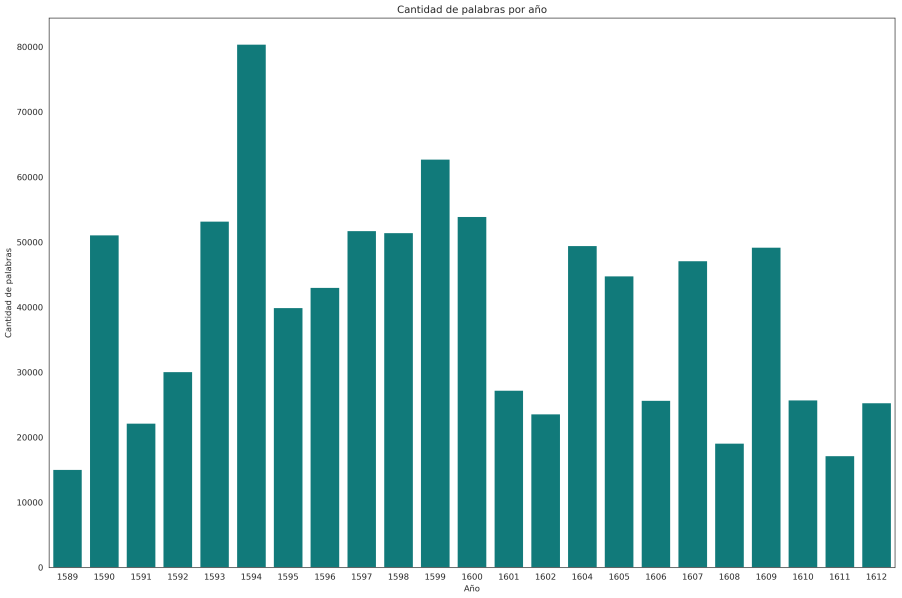


Figura 5.1: Cantidad de palabras escrita por año. 1594 fue el año de mayor volumen de trabajo en términos de palabras escritas, superando ampliamente a los demás. En este año Shakespeare escribió varias obras teatrales que se consideran importantes en su carrera como "Romeo y Julieta", "El mercader de Venecia" y "Sueño de una noche de verano". En el año 1603 no se tiene registro de escrituras del autor, y los períodos de menor concentración de palabras escritas se dan sobre el inicio y el final de su carrera como escritor. Sus años más productivos parecen ser los abarcados desde 1593 hasta 1600, cuando los géneros predominantes en su escritura eran la comedia y la historia.

Tabla 5.1: 10 palabras más frecuentes en la obra (sin remover *Stop Words*)

Palabra	**Conteo**
the	28933
and	27312
i	23006
to	20820
of	17179
a	15084
you	14227
my	12951
that	11910
in	11656

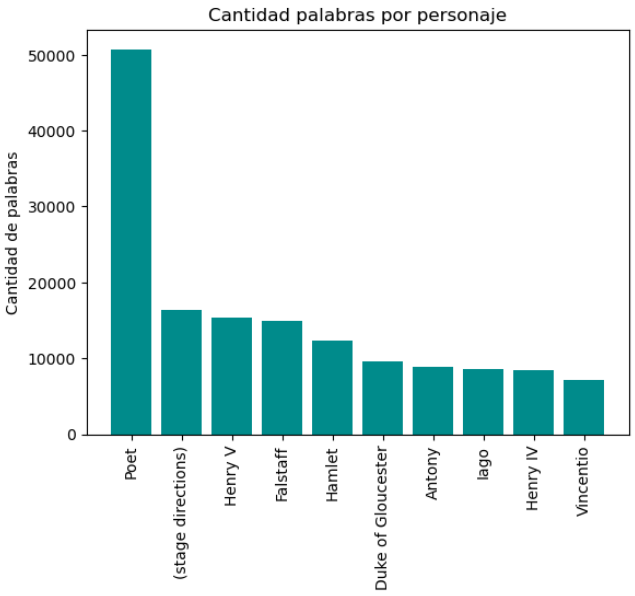


Figura 5.2: Visualización de los 10 personajes con mayor cantidad de palabras considerando la obra completa. El poeta, *Falstaff* y la dirección de escenario tienen un gran número de palabras cuando no se discrimina por obra

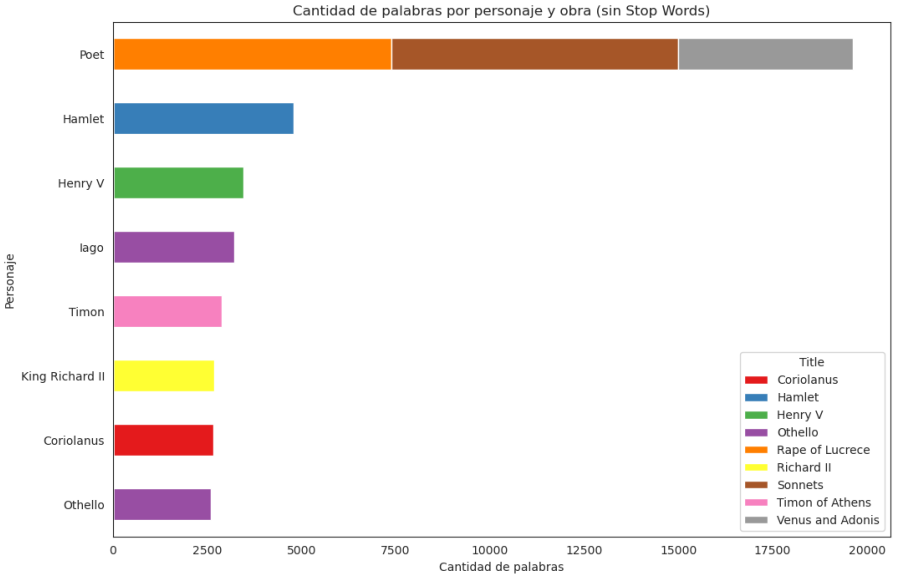


Figura 5.3: Personajes con mayor cantidad de palabras discriminados por obra (sin eliminar *Stop Words*).El poeta se mantiene como el personaje con mayor cantidad de palabras (en números absolutos y por obra). Hamlet al tercer lugar, dejando atrás a Henry V, personaje que aparece en más de una obra, por lo que el conteo total es mayor que el conteo por obra. Por otro parte, tanto Falstaff como la dirección de escena salen del ranking de los 10 personajes con mayor cantidad de palabras. Es interesante notar cómo la cantidad de palabras del personaje Poeta se encuentra dispersa en varias obras. El conteo inicial muestra al poeta con 50762 palabras, sin embargo, en una obra completa, el número máximo de palabras dicha por el personaje es de 18036 (en la obra Sonnets), y el segundo lugar es de 15530 (en la obra Rape of Lucrece).

Anexo 2 - Entrenamiento de modelos

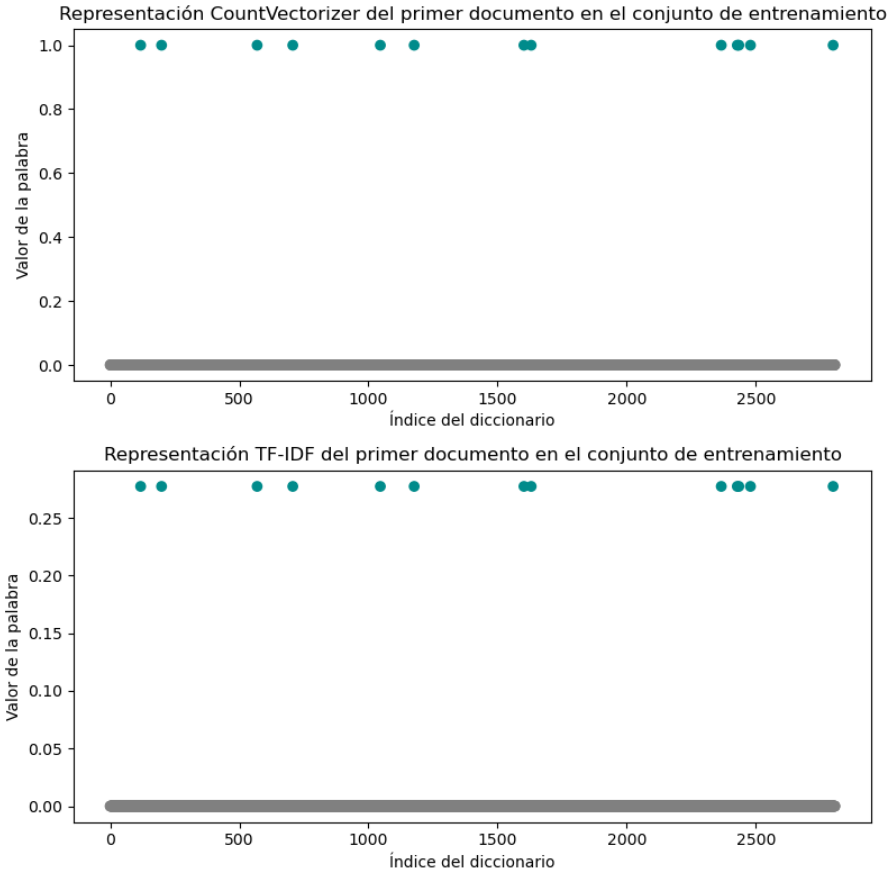


Figura 5.4: Representación numérica del párrafo *then you belike suspect these noblemen as guilty of duke humphrey s timeless death*

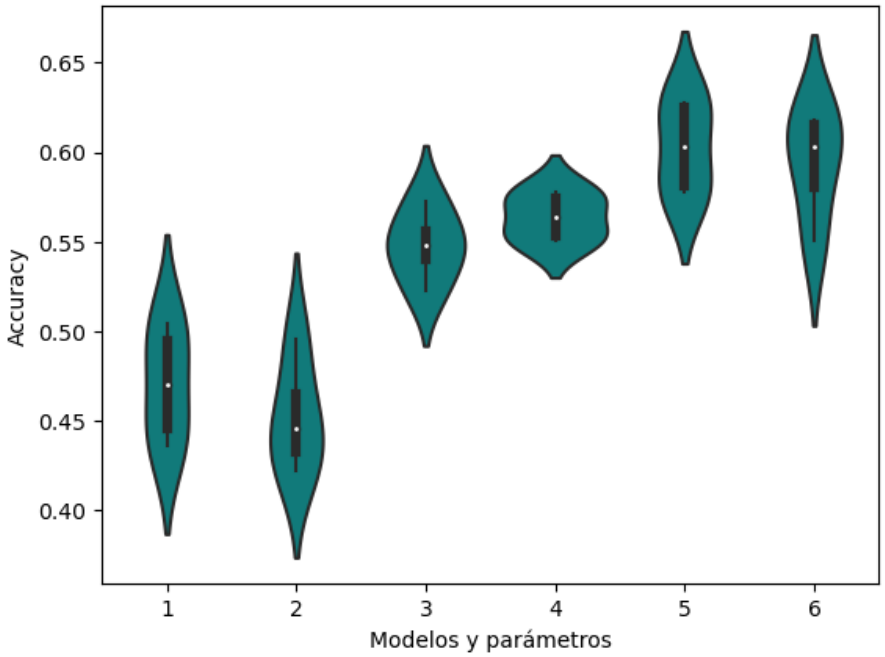


Figura 5.5: Comparación de métricas de accuracy para el modelo **Multinomial Naive Bayes** obtenidas mediante validación cruzada (`cross-validation`). Parámetros: `1:stop_words:None, ngram:(1,2), idf:True, 2:stop_words:None, ngram:(1,1), idf:False, 3:stop_words:'english', ngram:(1,2), idf:True, 4:stop_words:'english', ngram:(1,2), idf:False, 5:stop_words:'english', ngram:(1,1), idf:True, 6:stop_words:'english', ngram:(1,1), idf:False`