Introducción a la Ciencia de Datos

2024

Segunda tarea

Pedro Ignacio Lasa
Andrés Santos

Contenido

Introducción:

El procesamiento del lenguaje natural (NLP) ofrece una poderosa herramienta para analizar y comprender diferentes documentos de texto, permitiéndonos explorar patrones y relaciones en los textos de manera estructurada. Los análisis propuestos en esta tarea se aplican, al igual que en la tarea número uno, a la base de datos de la obra de Shakespeare; por lo tanto,

El objetivo de esta segunda entrega es la aplicación de técnicas de reducción de dimensionalidad (PCA) y de técnicas de aprendizaje automático (machine learning) supervisado. Basándonos en las obras de Shakespeare, dividimos la muestra en un set de datos de entrenamiento y otro de prueba o test; eventualmente útil para aplicar técnicas de predicción.

La tarea se divide en dos partes: en una primera instancia, aplicamos tareas de limpieza de texto (estudiadas en la primera entrega), transformamos el texto en instancias numéricas aplicando “bag of words” y “tf-itf”, e incluimos análisis de reducción de dimensionalidad con el uso de componentes principales observando la varianza acumulada en cada componente. En la segunda parte, nos centramos en modelos supervisados de clasificación, utilizando el algoritmo Naive Bayes Multinomial, y el modelo Logístico Multinomial. Se evaluó su rendimiento con el set de datos de prueba utilizando la matriz de confusión y las principales métricas con las cuales evaluamos la performance.

Parte 1: Dataset y representación numérica de texto

El trabajo parte de las mismas tablas utilizadas en la tarea 1 con información sobre la obra de Shakespeare, estos son: “df_works”, “‘df_paragraphs”, “df_chapters”, "df_work" y “df_charapters”.

Al igual que la tarea 1, se realizó la limpieza de texto de la de la columna “Plaintext” que forma parte del data set “df_paragraphs”, en el cual se incluyen todos los diálogos de las obras del autor. Para ello se hizo uso de la misma función “clean_text” desarrollada en la tarea 1, la cual elimina los símbolos de puntuación, transforma las mayúsculas en minúsculas, y sustituye las contracciones por las palabras completas. Es preciso mencionar que nuestra función no incluye la eliminación de las stop_words.

.

Con el texto ya normalizado y limpio, se utilizó la función ‘merge’ de pandas para combinar las diferentes tablas obteniendo nuestro data set, “df_dataset”, que será el punto de partida para todos los análisis que se presentan en el presente informe. El mismo está conformado por las siguientes columnas: CleanText (texto limpio), CharName (personaje), Title (nombre de la obra), y GenreType (género).

Posteriormente, reducimos nuestra base quedándonos sólo con los párrafos de tres personajes: "Antony", "Cleopatra" y "Queen Margaret".

Dividimos el conjunto de datos en dos subconjuntos: **entrenamiento** (train) y **prueba** (test) con una proporción de 70% y 30% respectivamente. La división se realiza de forma estratificada, lo que significa que se mantiene la proporcionalidad de las clases, en nuestro caso, personajes. Para ello, aplicamos el parámetro “stratify” de la función “train_test_split” de scikit-learn, fijando también el parámetro “random_state” para obtener resultados reproducibles. Con una cantidad de 438 y 188 instancias, los grupos de entrenamiento y prueba respectivamente.

En el siguiente gráfico de tortas se presenta la proporción de párrafos por personaje en cada subconjunto de datos (entrenamiento y prueba). Se verifica que se mantiene la misma proporcionalidad en ambos extractos:

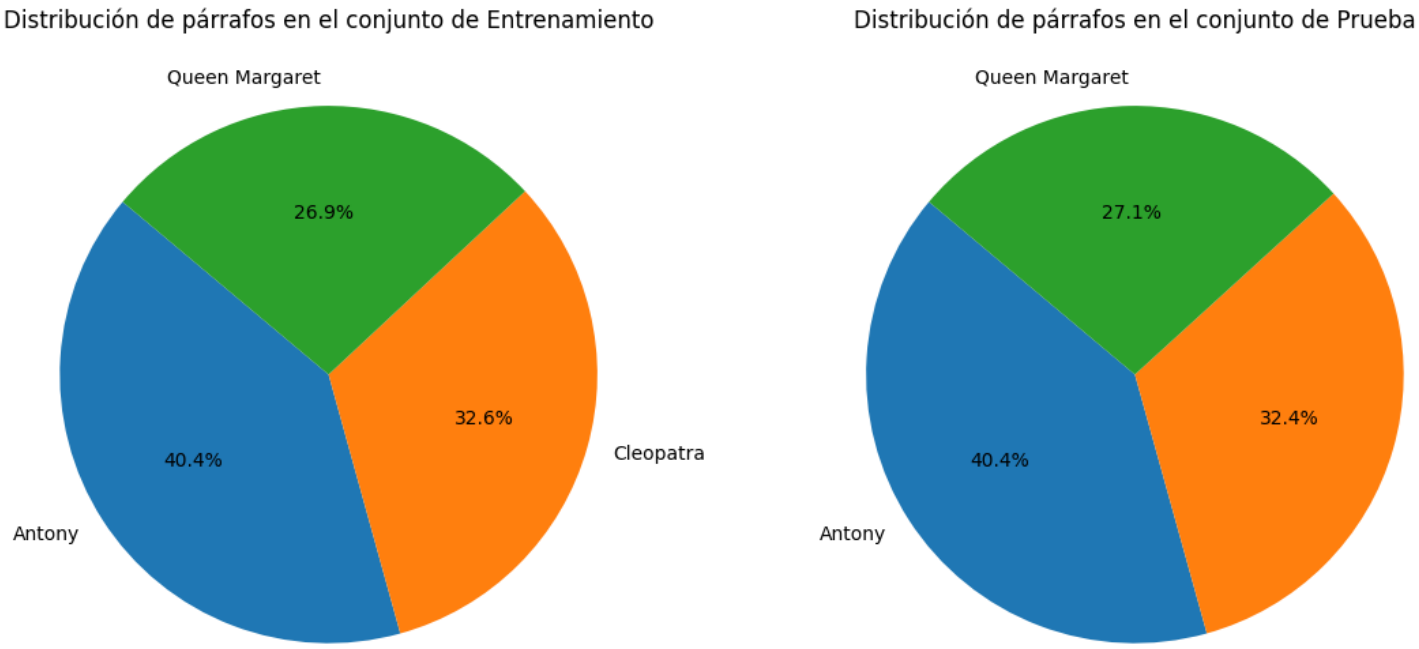

Gráfico 1: Proporción de párrafos de cada personaje en el set de entrenamiento y prueba.

A pesar que se mantiene una proporcionalidad entre ambos sets de datos, el personaje de Antony concentra la clase mayoritaria (este punto puede impactar a la hora de predecir y evaluar el modelo, detalle analizado en la segunda parte). Dependiendo del fin último de nuestro estudio, también sería pertinente verificar la proporcionalidad de la cantidad de palabras de cada personaje en cada subgrupo.

Bag of words

Transformamos el texto del conjunto de entrenamiento a la representación numérica (features) de conteo de palabras o “bag of words". Cada párrafo se representa como un vector de características, basado en la frecuencia de aparición de la palabra en dicho contexto. La técnica de "bag of words" convierte texto en una representación numérica; la función “CountVectorizer”

se encarga de convertir la colección de documentos de texto en una matriz de conteos de palabras, donde cada columna corresponde a la frecuencia de cada palabra en cada documento.

Por otro lado, el parámetro 'stop_words=False' indica que no se eliminarán palabras comunes (stop words) durante el procesamiento; el término 'ngram_range= (1,1)' indica que solo consideramos los unigramas. Con esta parametrización se pierde información sobre el orden de las palabras en el documento asociado, lo que significa que, si quisiéramos revertir la transformación y obtener el texto original, no se podría realizar. Hacemos un paréntesis aquí para explicar qué son los N-gramas.

Un n−grama es una secuencia contigua de n elementos de un texto. Estos elementos pueden ser palabras o caracteres. Los n−gramas se utilizan en procesamiento de lenguaje natural (NLP) para capturar contextos y patrones en el texto. Por ejemplo:

Secuencias de 1 elemento, 1-grama o unigrama: "Jugamos un juego" se convierte en ["Jugamos", "un", "juego"]. En el caso de 2-gramas o bigramas se convierte en ["Jugamos un", "un juego"].

El resultado final de nuestra transformación con la función “CountVectorizer”, es la matriz 'X_train_counts' dispersa (sparse matrix), de tamaño 438 x 2811. Las 2811 columnas corresponden a la cantidad de palabras únicas que aparecen en el cuerpo de texto proporcionado, en nuestro caso, en los 438 párrafos. La matriz dispersa almacena los conteos de términos en formato CSR (Compressed Sparse Row), lo que ahorra espacio al almacenar sólo los valores no nulos y sus índices.

Para explicar mejor el concepto de “bags of words” desarrollaremos un ejemplo:

texts = ["El gato está en la casa",
"El perro está en el patio",
"El gato y el perro está en la calle"]

El conjunto vocabulario es: {'el', 'gato', 'está', 'en', 'la', 'casa', 'perro', 'patio', 'calle'}

El conjunto vocabulario con conteo es: {'el': 2, 'gato': 5, 'está': 4, 'en': 3, 'la': 6, 'casa': 1, 'perro': 8, 'patio': 7, 'calle': 0}

La representación "bag of words" en matriz (matriz de conteos):

[[0 1 1 1 1 1 1 0 0]
[0 0 2 1 1 0 0 1 1]
[1 0 2 1 1 1 1 0 1]]

En el contexto de procesamiento de lenguaje natural, las matrices de conteos de términos suelen ser dispersas porque:

- *Gran Vocabulario: En un cuerpo de texto, el vocabulario puede ser muy grande, pero cada documento generalmente contiene solo una pequeña fracción de esos términos lo que resulta en una matriz donde la mayoría de las entradas son cero.*
- *Eficiencia de memoria: Las matrices dispersas almacenan sólo los elementos no nulos, lo que reduce significativamente el uso de memoria.*
- *Eficiencia Computacional: Muchas operaciones matemáticas y algoritmos están optimizados para trabajar con matrices dispersas, lo que puede acelerar los cálculos y reducir el tiempo de procesamiento.*

Tamaño de la Matriz Dispersa vs. Densa:

Se comparó la cantidad de bytes que contiene nuestra matriz de datos una vez transformada a un “bag of words”, en formato denso o disperso. Luego se hace el mismo ejercicio de comparación con la matriz parametrizada por TF-IDF, transofmación que explicaremos más adelante.

1) Matriz de Conteos o “bag of words”:

- La matriz dispersa ocupa 45.959 bytes.
- La matriz densa ocupa 9.849.744 bytes.

2) Matriz TF-IDF:

- La matriz dispersa ocupa 2.092 bytes.
- La matriz densa ocupa 9.849.744 bytes.

En el primer caso, la matriz densa es 214 veces más grande que la matriz dispersa, y en el segundo caso, la relación es del orden de los 4000 veces mayor.

Las matrices dispersas son mucho más eficientes en términos de uso de memoria, en comparación con las matrices densas. Para la matriz de conteos, la eficiencia se debe a que la mayoría de los elementos en la matriz son ceros, ya que cada documento sólo contiene una pequeña fracción del vocabulario total. Para la matriz TF-IDF, la eficiencia es aún mayor porque los valores están normalizados y muchos de ellos pueden ser cercanos a cero, lo que permite una mayor compresión en la representación dispersa.

TF-IDF (Term Frequency-Inverse Document Frequency)

A nuestra matriz dispersa “X_train_counts”, le aplicamos una transformación adicional llamada TF-IDF, (Term Frequency-Inverse Document Frequency). Esta es una técnica de vectorización que convierte un conjunto de documentos de texto en una matriz de características numéricas. A diferencia de la simple frecuencia de palabras (“bag of words”), TF-IDF pondera dicha frecuencia con el tamaño del documento (o párrafo en nuestro caso) (TF), y luego, hace una

segunda ponderación por el logaritmo del inverso de la frecuencia de la palabra en todos los documentos del corpus (IDF). En nuestro caso, el corpus son todos los párrafos. El resultado pretende ser una medida numérica que expresa cuán relevante es una palabra para un documento en una colección de documentos.

Fórmula TF:

$$TF(t,d) = \frac{Nº\ de\ veces\ que\ aparece\ el\ término\ t\ en\ el\ documento\ \ d}{Nº\ total\ de\ términos\ en\ el\ documento\ d}$$

Fórmula IDF:

$$IDF(t,D) = \ log\left(\frac{Nº\ \ de\ documentos\ en\ el\ cuerpo\ D}{1\ +\ Nº\ \ de\ documentos\ en\ D\ que\ \ contienen\ el\ término\ t}\right)$$

Luego, TF-IDF se calcula como:

$$TF - IDF\ (t,d,D)\ =\ TF(t,d)\ \times\ IDF(t,D)$$

Análisis de Componentes Principales (PCA)

El análisis de componentes principales (PCA) es una técnica estadística utilizada para reducir la dimensionalidad de un conjunto de datos, conservando la mayor cantidad de variabilidad presente en los datos originales o, dicho de otra manera, con la mínima pérdida de información. PCA reduce la dimensionalidad de los datos, lo que facilita la visualización de patrones y estructuras en los datos. Además, está técnica puede ayudar a visualizar grupos de datos que estén correlacionados.

Avanzamos en nuestro análisis aplicando PCA a nuestro grupo de entrenamiento, y contrastamos los resultados obtenidos aplicando dos parametrizaciones diferentes en el tratamiento previo de los datos:

Base de entrenamiento	stop_words	ngram_range	use_idf	Observaciones
X_train_tf	None	(1,1)	True	con stop words, unigramas, tf-itf
X_train_tf_english	english	(1,2)	True	sin stop words, unigramas y bigramas, tf-itf

Graficamos las dos primeras componentes principales en cada uno del set de datos:

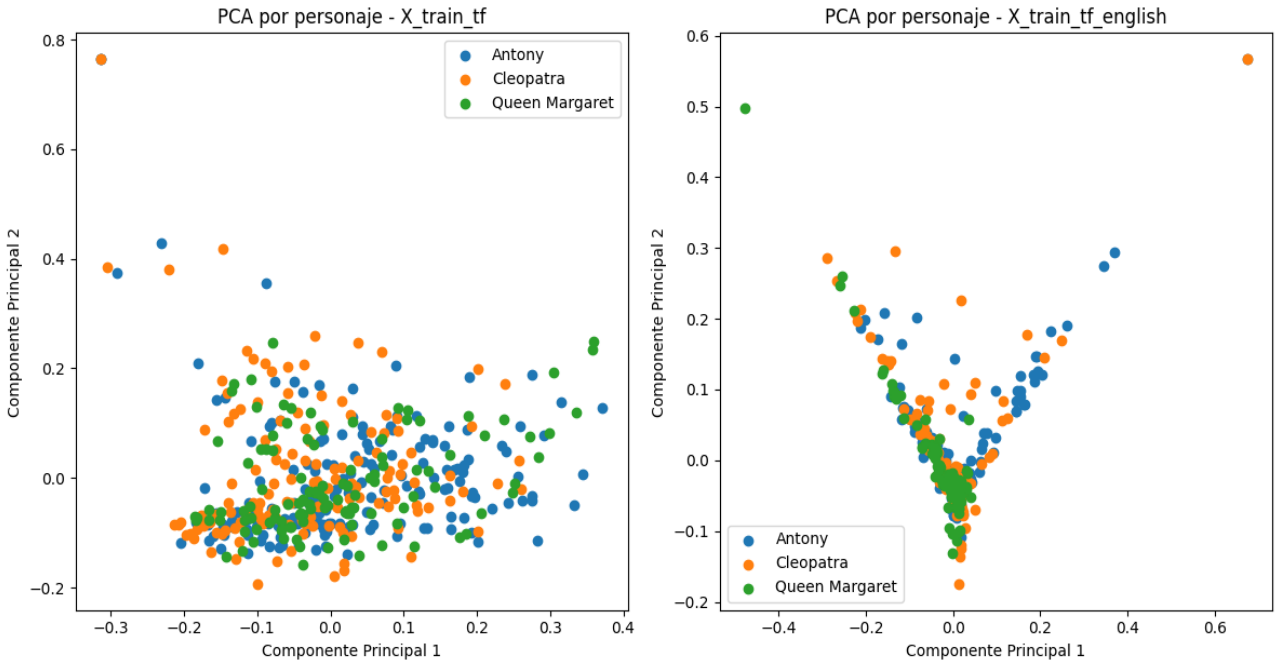

Gráfico 2: Representación en el plano de las dos primeras componentes principales.

En ninguno de los dos casos pareciera posible discriminar agrupaciones claras de las diferentes clases. Si estudiamos la varianza explicada en los primeras 10 componentes principales comprobamos que la información que nos pueden arrojar es muy escasa.

A continuación graficamos, para cada una de las bases de entrenamiento, la varianza explicada acumulada hasta las 10 primeras componentes principales:

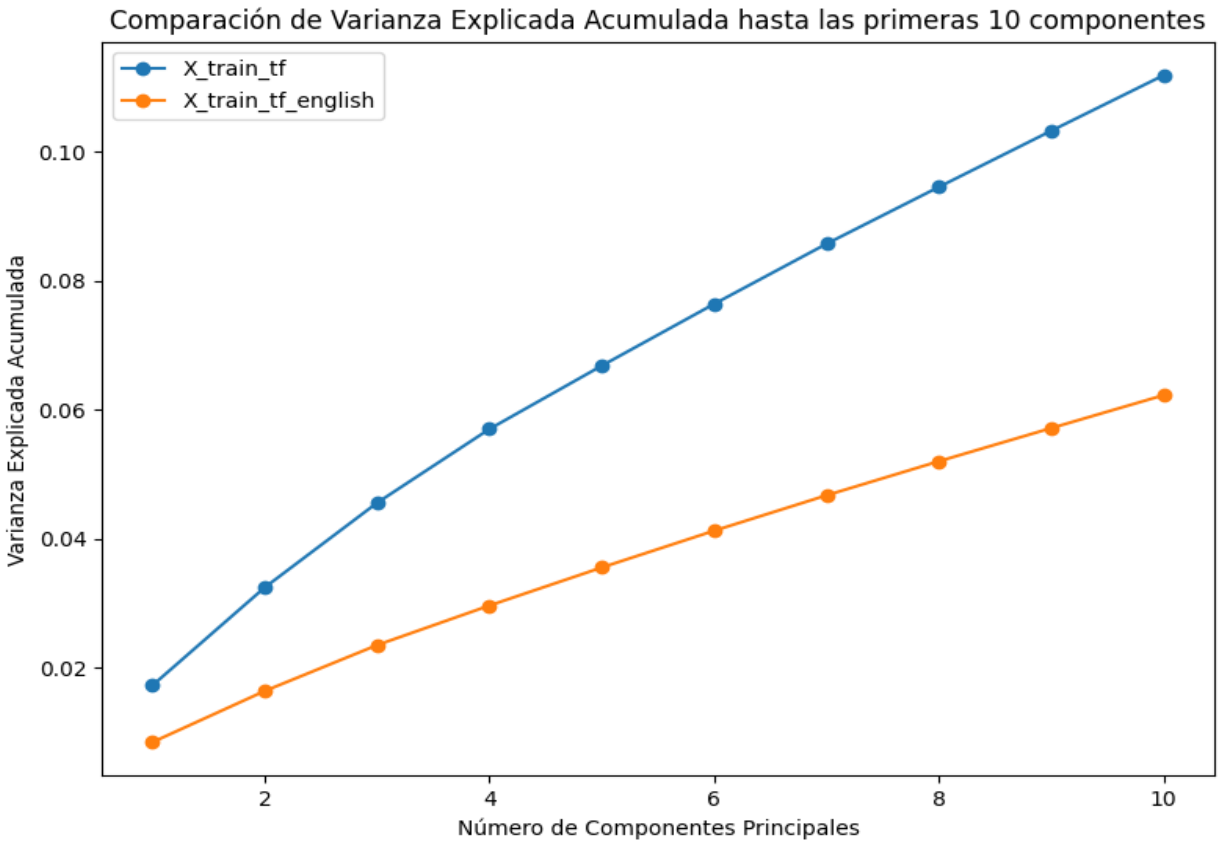

Gráfico 3: Varianza explicada de los primeros 10 componentes principales.

Como se aprecia en el gráfico 3, es muy poca la varianza explicada por las dos primeras componentes principales, tanto en **X_train_tf** como en **X_train_tf_english.** En ambos casos, la variabilidad explicada no llega a ser del 4% de los datos, es necesario aumentar los componentes principales y así, llegar a una variabilidad de los datos razonable.

La base parametrizada con stop-words y con unigramas, (curva azul), tiene una varianza explicada sensiblemente superior que la otra base, en la que se eliminaron las stop-words y que contempla tanto unigramas como bigramas, (curva naranja). De todas maneras, la varianza explicada en los primeros 10 componentes principales es insignificante para cualquiera de los dos casos, si lo que pretendemos es identificar el autor de cada diálogo por su similaridad y correlación respecto a los de su misma clase.

Veremos ahora cómo se comportan si graficamos la varianza explicada acumulada de todos los componentes principales:

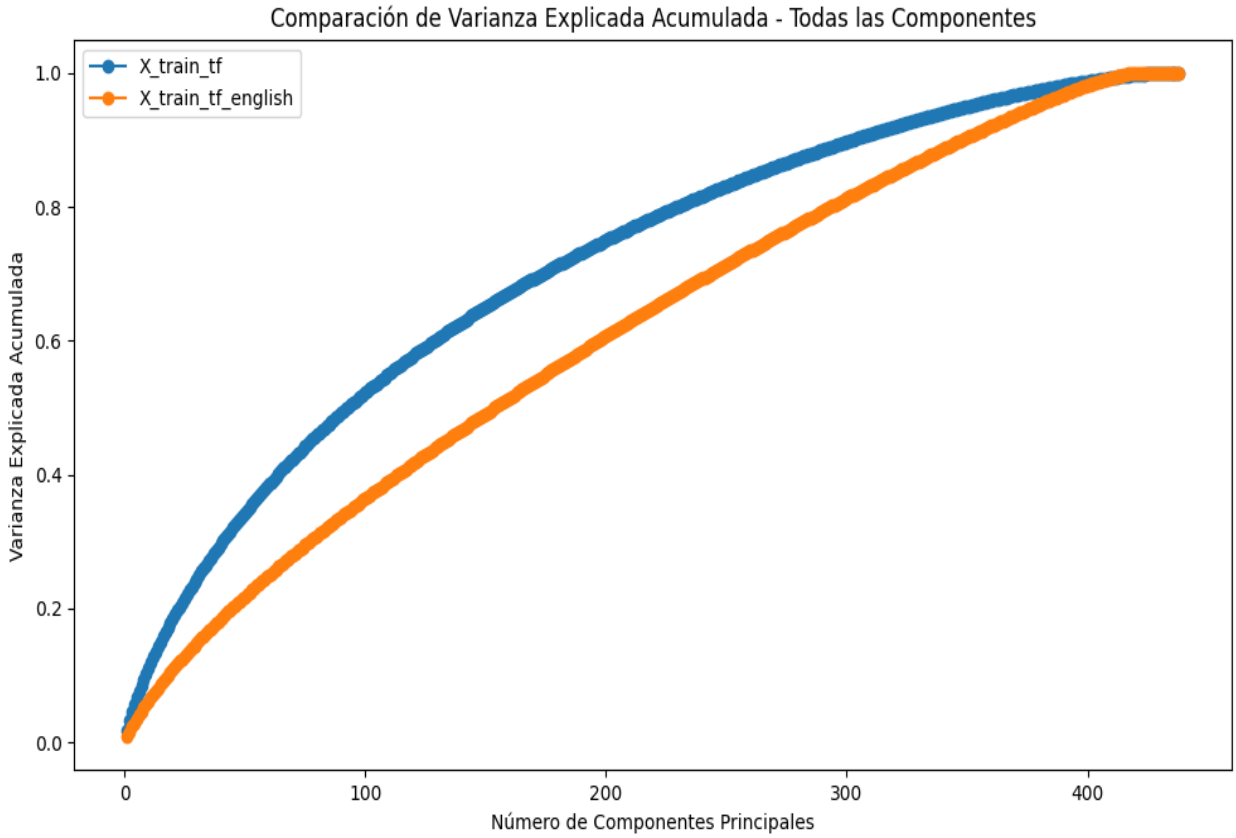

Gráfico 4: Varianza explicada de todos los componentes principales

Como comentario general, podemos apreciar que se precisan muchos componentes principales para lograr una pérdida de información poco significativa. Si se desea explicar entre el 80% y el 90% de los datos, se necesitan más de 300 componentes principales cuando partimos de 438 instancias o párrafos. La reducción de la dimensionalidad no parece ser una herramienta útil en este caso. Aun así, podemos decir que la base de entrenamiento tratada que incluye las stop words y solo toma unigramas funciona relativamente mejor que la parametrizada sin stop words, con unigramas y bigramas.

Parte 2: Entrenamiento y evaluación de modelos

Multinomial Naive Bayes

En el ámbito de la clasificación de textos, el modelo Multinomial Naive Bayes (MNB) se destaca como uno de los más utilizados. Basado en el teorema de Bayes, este algoritmo asume que las características predictoras, representadas por conteos o frecuencias de palabras, son independientes entre sí. A partir de los datos de entrenamiento, estima las probabilidades de cada palabra o característica en cada clase y, posteriormente, emplea el teorema de Bayes para calcular la probabilidad de que una nueva muestra pertenezca a cada una de ellas.

Por otro lado, una herramienta fundamental para evaluar el desempeño de modelos de clasificación se apoya en la matriz de confusión; la misma, presenta las predicciones del modelo junto a las etiquetas reales, permitiendo el cálculo de métricas como la precisión y el recall para cada clase. Un modelo ideal se caracteriza por una alta concentración de valores en la diagonal principal de la matriz, lo que indica un mayor número de aciertos.

Partiendo de los datos de entrenamiento **X_train_tf**, entrenamos un modelo MNB y evaluamos los resultados en el grupo de prueba **X_test_tfidf**. A continuación, presentamos como resultado del análisis la matriz de confusión:

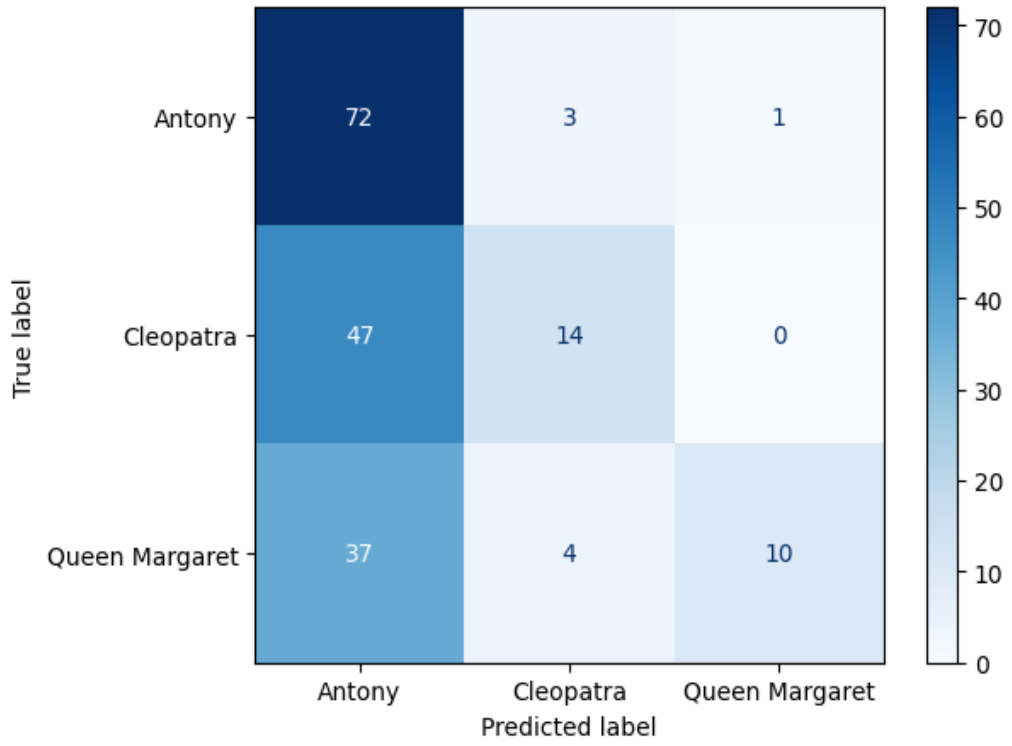

Gráfico 5: Matriz de confusión

Se observa una clara disparidad en la distribución de datos, beneficiando al conjunto "Antony" que posee la mayor cantidad de datos, en detrimento de los otros dos conjuntos. Esta situación sugiere una precisión total promedio, producto de un alto número de aciertos en la clase mayoritaria mientras que las demás clases presentan un bajo rendimiento. En efecto, el valor de la precisión total es del 51%, lo cual es bastante pobre (el modelo clasifica correctamente el 51% de las instancias).

La matriz de confusión revela el desequilibrio en los datos (algo mencionado en la parte uno), que puede inclinar al modelo hacia la clase mayoritaria, en detrimento de las minoritarias. Esto podría resultar en un alto valor de accuracy, pero este valor no reflejaría un buen desempeño del modelo, ya que simplemente estaría prediciendo la clase dominante en la mayoría de los casos sin una verdadera capacidad de discriminación.

Presentaremos a continuación las principales métricas, con las cuales evaluaremos la performance del modelo. Pero antes debemos definir algunos conceptos previos:

- Verdaderos positivos, (TP): Cantidad de instancias positivas que el modelo las califica positivas.
- Verdaderos negativos (TN): Cantidad de instancias falsas que el modelo las califica falsas.
- Falsos positivos (FP): Cantidad de instancias negativas que el modelo califica como positivas.
- Falsos negativos (FN): Cantidad de instancias positivas que el modelo califica como falsas.

Ahora, estamos en condiciones de definir las diferentes métricas mencionadas. Utilizaremos sus denominaciones en inglés, ya que es la forma más común de referirse a ellas:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \; ; \qquad Precisión = \frac{TP}{TP + FP} \; ;$$

$$Recall = \frac{TP}{TP + FN}; \qquad F1 - score = \frac{2 \times Precisión \times Recall}{Precisión + Recall} \; ;$$

Support: número de ocurrencias reales de cada clase en el conjunto de datos.

Dependiendo de los objetivos últimos de nuestro análisis y del modelo seleccionado, se podrá priorizar una métrica por sobre la otra.

Un ejemplo clásico para ilustrar esto es un modelo que diagnostica una cierta enfermedad. En este caso, la gran mayoría de la población de estudio será negativa (o sana). Si uno se enfocara en obtener un alto "accuracy", bastaría con clasificar todos los casos como negativos. Sin embargo, es más importante priorizar un alto "recall" sobre el "accuracy". En decir, es preferible sobrediagnosticar que sub-diagnosticar. Con un alto recall, nos aseguramos de minimizar los falsos negativos (FN) y garantizamos que quienes padezcan la enfermedad sean bien

diagnosticados. Como contraparte, probablemente obtendremos un mayor número de falsos positivos (FP), o sea que personas sanas serán diagnosticadas erróneamente con la enfermedad. Es una decisión de compromiso.

.

Otro ejemplo (ficticio) que ilustra esto, pero en dirección contraria al ejemplo anterior, podría ser el de un modelo utilizado para determinar la culpabilidad o la inocencia de los ciudadanos respecto a un determinado delito. En este caso, debemos asegurarnos que los clasificados como positivos así lo sean, y no condenar a personas inocentes. Por eso vamos a priorizar la "precisión" por sobre el "recall".

En el reporte a continuación presentamos el resultado de las diferentes métricas en nuestro modelo, para las diferentes clases.

	precision	recall	f1-score	support
Antony	0.46	0.95	0.62	76.00
Cleopatra	0.67	0.23	0.34	61.00
Queen Margaret	0.91	0.20	0.32	51.00
accuracy	0.51	0.51	0.51	0.51
macro avg	0.68	0.46	0.43	188.00
weighted avg	0.65	0.51	0.45	188.00

Un dato que destaca del cuadro anterior es que la clase "Antony" tiene un recall del 95%, mientras que las clases "Cleopatra" y "Queen Margaret" tienen solo un 23% y un 20%, respectivamente. Esto sugiere que nuestro modelo tiene un sesgo a favor de la clase "Antony". Esto es esperable, ya que es la clase más frecuente en nuestra base de datos. Si nuestra base de datos estuviera aún más desbalanceada, este sesgo sería probablemente aún más pronunciado. Por esta razón, es fundamental contar con una base de datos representativa y bien equilibrada del universo que estamos estudiando.

Hiper-parámetros con Cross Validation

Buscaremos ahora optimizar nuestro modelo con la búsqueda de hiper-parámetros y la aplicación de la validación cruzada o “cross validation”.

La validación cruzada se erige como una técnica fundamental para evaluar la capacidad de generalización de un modelo predictivo. A diferencia del enfoque tradicional de entrenar y evaluar el modelo una sola vez, la validación cruzada divide los datos en múltiples subconjuntos ("folds"), repitiendo el proceso de entrenamiento y evaluación en cada uno de ellos. Esta estrategia proporciona una estimación más robusta y confiable del rendimiento del modelo en escenarios reales.

Para evaluar el modelo de Naive Bayes multivariado y optimizar sus parámetros, se emplea la técnica de validación cruzada estratificada con cuatro particiones distintas. A continuación, se evaluará el rendimiento del modelo con diferentes combinaciones de parámetros, seleccionando aquella que maximice las métricas de evaluación.

Las diferentes parametrizaciones a testear son las siguientes:

```
param_sets = [
    {"stop_words": None, "ngram": (1,2), "idf": True},
    {"stop_words": None, "ngram": (1,1), "idf": False},
    {"stop_words": 'english', "ngram": (1,2), "idf": True},
    {"stop_words": 'english', "ngram": (1,3), "idf": True},
    {"stop_words": 'english', "ngram": (2,2), "idf": True},
    {"stop_words": None, "ngram": (1,1), "idf": True},
    {"stop_words": None, "ngram": (1,2), "idf": False},
]
```

Explicamos brevemente el significado de cada uno de los parámetros:

```
"stop_words": None      # indica que no se quitan las stop words
"stop_words": 'english' #indica que sí se quintan las stop words

"ngram": (1,1)           # se toman solo unigramas
"ngram": (2,2)           # se toman solo bigramas
"ngram": (1,2)           # se toman unigramas y bigramas
"ngram": (1,3)           # se toman unigramas, bigramas, y trigramas

"idf": True              # se aplica la ponderación completa tf-idf
"idf": False             # se aplica solo la ponderación tf
```

El resultado obtenido se presenta en los siguientes gráficos de violín de las métricas “Accuracy” y “Recall”:

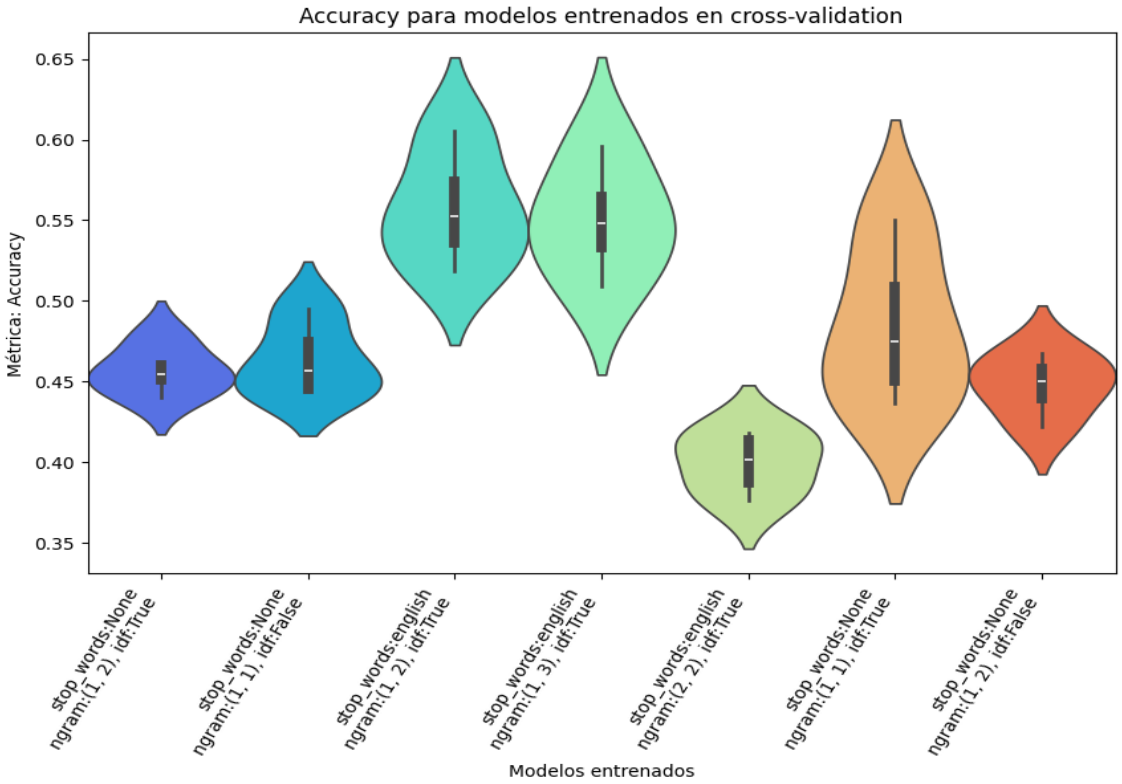

Gráfico 6: Gráfico de violín con la métrica de Accuracy.

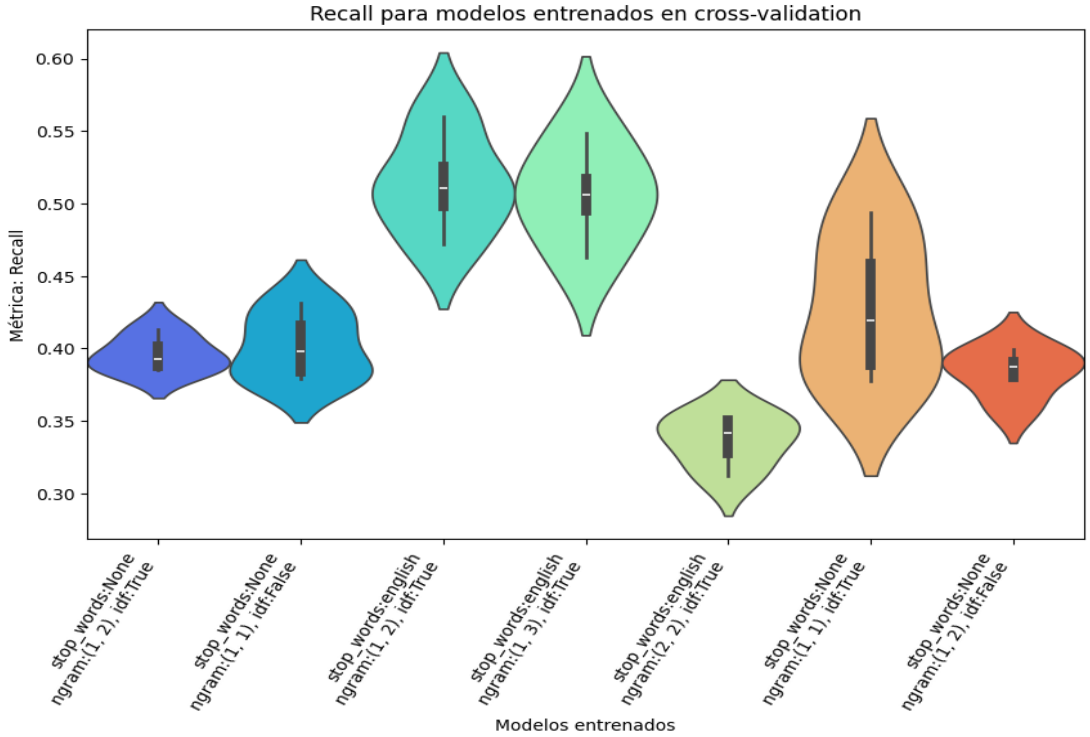

Gráfico 7: Gráfico de violín con la métrica de Recall.

Los gráficos anteriores muestran que tanto para las métricas “Accuracy” y “Recall” la parametrización con los valores promedios más altos es la de `"stop_words": 'english', "ngram": (1,2), "idf": True`. Este resultado resulta llamativo, cuando en el contexto de PCA previamente se había observado un rendimiento inferior de esta parametrización con respecto a la alternativa de: `"stop_words": None, "ngram": (1,1), "idf": False`.

En estos casos, el Acurrency presenta un máximo de 65%, mientras que el Recall posee un máximo del 60%.

Entrenamiento con los mejores parámetros encontrados

Se entrenó nuevamente el modelo multivariado de Naive Bayes, utilizando la base de datos de entrenamiento con la mejor parametrización encontrada en el punto anterior. Por último, se evalúa el modelo sobre el conjunto de prueba obteniendo los siguientes resultados para la matriz de confusión y métricas:

Gráfico 8: Matriz de confusión luego de aplicar hiper parámetros

	precision	recall	f1-score	support
Antony	0.50	0.91	0.65	76.00
Cleopatra	0.69	0.33	0.44	61.00
Queen Margaret	0.82	0.35	0.49	51.00
accuracy	0.57	0.57	0.57	0.57
macro avg	0.67	0.53	0.53	188.00
weighted avg	0.65	0.57	0.54	188.00

Los resultados obtenidos son apreciablemente mejores que los obtenidos con la otra parametrización, pero aún son bastante pobres. Por ejemplo, el valor de accuracy que mejoró unos 6 puntos (de 51% a 57%).

Los modelos basados en “bag-of-words” y TF-IDF tienen varias limitaciones cuando se trata del análisis de texto. En primer lugar, estos modelos no capturan totalmente el orden de las palabras en un documento, lo que significa que se pierde información crucial sobre la sintaxis y la semántica. Por ejemplo, las frases "el perro persigue al gato" y "el gato persigue al perro" serían tratadas de manera similar en un modelo bag-of-words o TF-IDF, a pesar de tener significados completamente diferentes. Además, estos enfoques no consideran las relaciones entre las palabras, lo que limita su capacidad para comprender el contexto y los matices del lenguaje. Hay que considerar aquí la excepción de cuando utilizamos bigramas o trigramas en “bag of words”, donde parte de estas relaciones sí se capta.

Otra limitación importante, es que los modelos bag-of-words y TF-IDF generan representaciones de alta dimensionalidad, especialmente cuando se trabaja con grandes vocabularios. Esto puede llevar a problemas de sobreajuste y a un aumento significativo en los recursos computacionales necesarios para el entrenamiento y la predicción. Además, estos modelos no son efectivos para capturar el significado de palabras que tienen múltiples sentidos o palabras que son similares, lo que puede afectar negativamente la precisión del análisis de texto. Por estas razones, los modelos más avanzados como los basados en embeddings de palabras o técnicas de procesamiento de lenguaje natural más complejas (ejemplo: transformers) suelen ser preferidos para tareas de análisis de texto más sofisticadas.

Regresión Logística Multinomial

Se entrenó y evaluó un modelo logit multinomial (también conocido como regresión logística multinomial) utilizando las mismas “features” o características de texto y se comparó con los resultados obtenidos del modelo multinomial de Naive Bayes. La Regresión Logística Multinomial es una extensión del modelo logit binomial, se utiliza cuando la variable de respuesta tiene más de dos categorías. Funciona estimando las probabilidades de las diferentes clases basadas en una función logística; es decir, predice la probabilidad de ocurrencia de más de dos posibles resultados categóricos.

Tal como se realizó con el modelo anterior, se aplica la validación cruzada con 4 splits diferentes, se evalúan varias parametrizaciones diferentes para encontrar la combinación de parámetros que obtenga las mejores métricas.

La mejor parametrización obtenida en este caso es:

```
{'stop_words': 'english', 'ngram_range': (1, 1), 'use_idf': False}
```

Usando esta parametrización, se entrena el modelo logístico con el mismo grupo de entrenamiento utilizado anteriormente y se evalúa sobre el grupo de prueba, obteniendo los siguientes resultados:

Reporte de clasificación:

	precision	recall	f1-score	support
Antony	0.56	0.82	0.67	76
Cleopatra	0.58	0.41	0.48	61
Queen Margaret	0.80	0.55	0.65	51
accuracy			0.61	188
macro avg	0.65	0.59	0.60	188
weighted avg	0.63	0.61	0.60	188

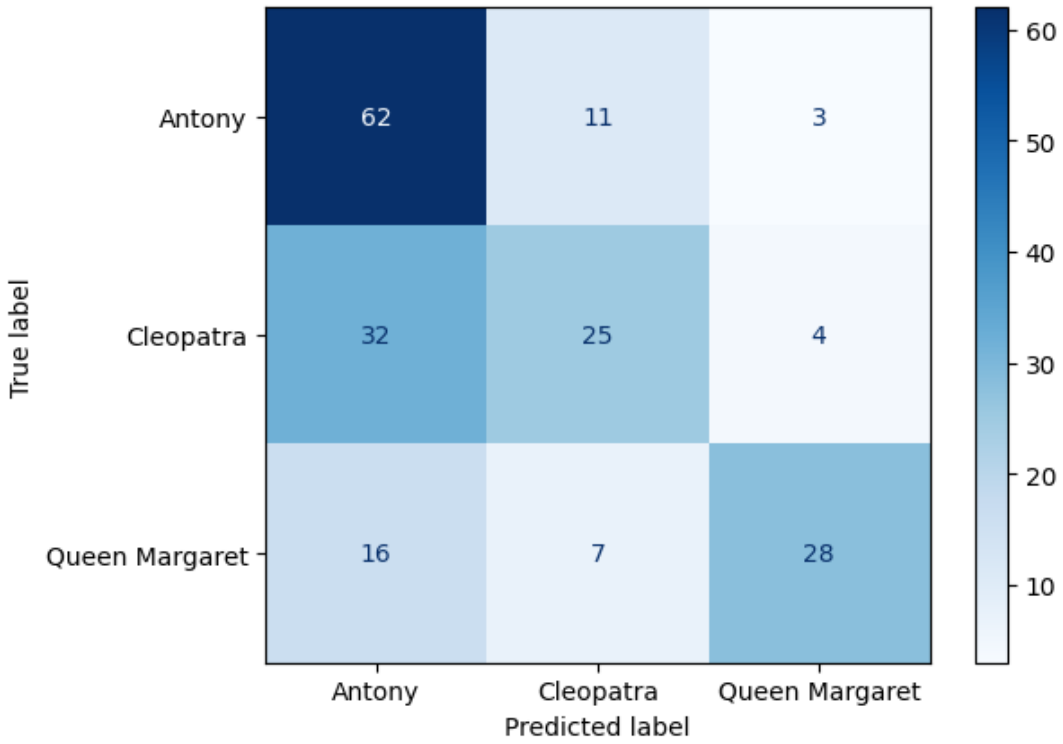

Gráfico 9: Matriz de confusión del modelo logístico.

Los resultados son levemente mejores que los obtenidos con el modelo multinomial Naive Bayes. El “accuracy” aumentó del 57% al 61%, y por su lado, el recall también aumenta su promedio siendo del 53% para NV, y de 59% para el modelo logístico.

Es curioso ver como diferentes parametrizaciones optimizan los diferentes modelos. Cada modelo tuvo una parametrización diferente que maximiza su performance.

Cambiamos la base de datos: “Henry V”

Un desafío adicional a considerar es el desequilibrio de datos que puede surgir debido a la variación en la cantidad de instancias por clase. Este desequilibrio, donde algunas clases poseen un número significativamente mayor de instancias que otras, puede conducir a un modelo con baja precisión para las clases minoritarias, ya que tiende a predecir las clases mayoritarias por defecto. Para abordar este problema, se pueden emplear técnicas de reequilibrio de datos como el sobremuestreo o el submuestreo. El sobremuestreo consiste en aumentar artificialmente el número de instancias en las clases minoritarias, mientras que el submuestreo reduce el número de instancias en las clases mayoritarias. De esta manera, se logra un conjunto de datos más balanceado y se mejora el rendimiento del modelo en todas las clases.

Partiremos ahora de una base diferente. Sustituiremos al personaje “*Antony*” por el de “*Henry V*” en nuestra base de datos. Estudiaremos qué ocurre con nuestros modelos cuando el desbalance de clases es más acentuado.

Los tamaños de los grupos de entrenamiento y test son los siguientes:

	Cleopatra	Henry V	Queen Margaret
Entrenamiento	143	264	118
Prueba	60	113	51

En el siguiente gráfico de torta se puede comprobar que se mantiene la proporción de clases entre ambos grupos:

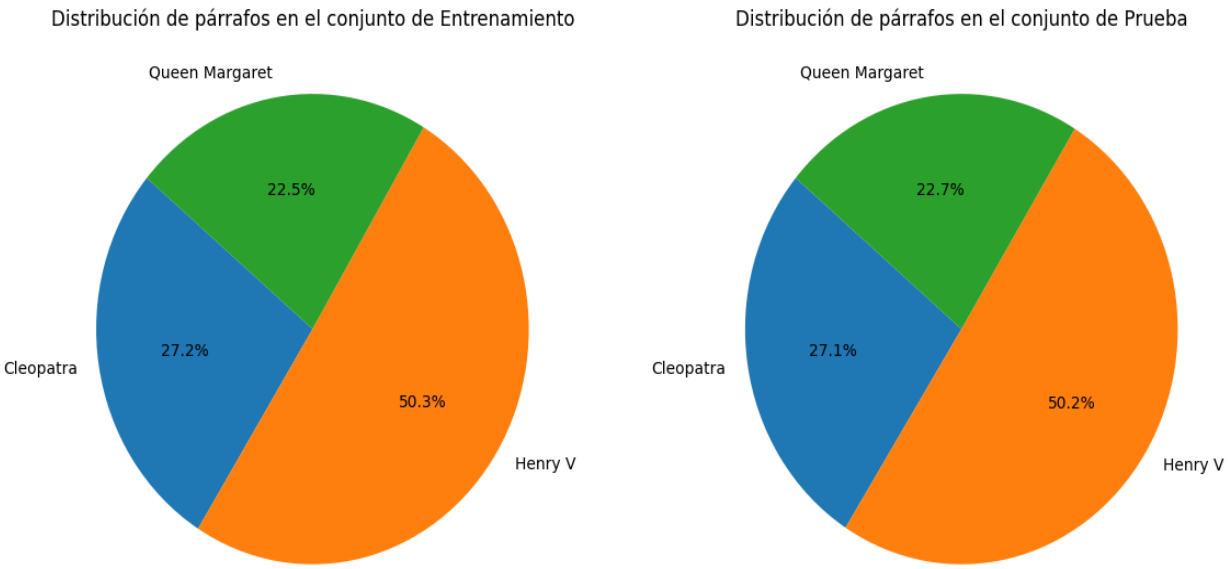

Gráfico 10: Proporción de párrafos de cada personaje en el set de entrenamiento y prueba.

Transformamos nuestro nuevo data frame a un “bag of words”, y luego a tf-itf utilizando la siguiente parametrización:

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

Aplicamos PCA al grupo de entrenamiento y obtenemos el siguiente gráfico de las dos primeras componentes principales:

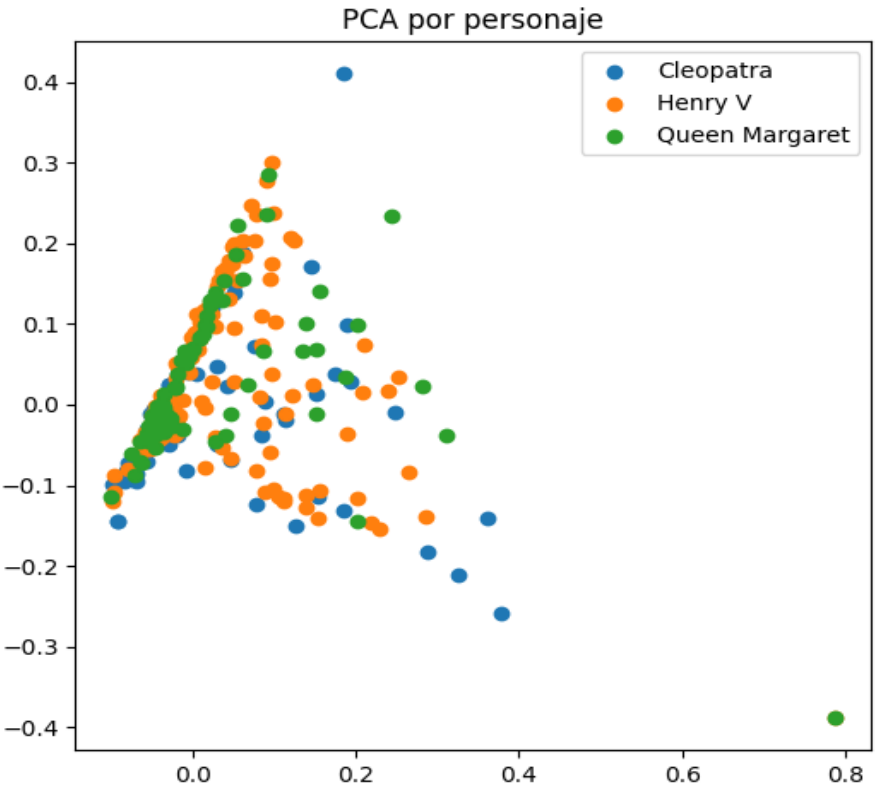

Gráfico 11: Representación en el plano de las dos primeras componentes principales con "Henry V".

En el gráfico 11 se muestran las dos primeras componentes principales sin la posibilidad de identificar grupos. Ahora aplicamos el modelo de multinomial de Naive Bayes y obtenemos la matriz de confusión:

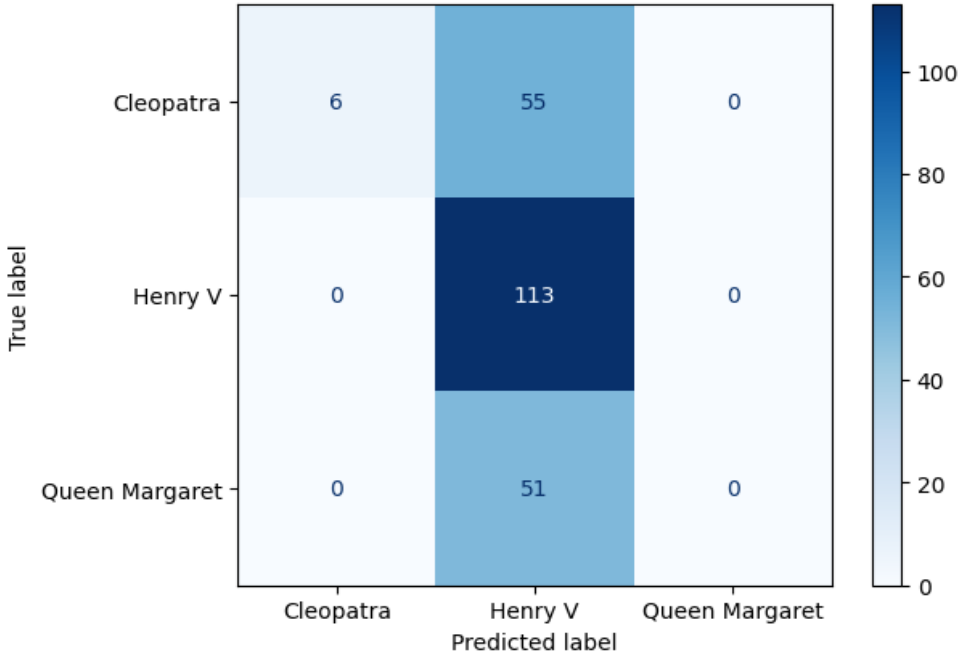

Gráfico 12: Matriz de confusión con "Henry V".

```
Accuracy: 0.5288888888888889
Classification Report: params={'stop_words':None, 'ngram': (1, 1), 'idf':
False}
                precision    recall  f1-score   support

     Cleopatra       1.00      0.10      0.18        61
       Henry V       0.52      1.00      0.68       113
Queen Margaret       0.00      0.00      0.00        51

      Accuracy                           0.53       225
     macro avg       0.51      0.37      0.29       225
  weighted avg       0.53      0.53      0.39       225
```

Los resultados obtenidos comprueban el desbalance analizado con anterioridad. Una base de datos muy desbalanceada generará modelos con sesgo muy marcados a etiquetar con la clase más frecuente en la base de datos, en este caso “Henry V”. Observemos que el recall de Cleopatra es 0,1, y el de Queen Margaret 0.

Consideramos los embeddings de palabras ("Word Embeddings") como técnicas avanzadas para extraer características ("features") de texto. Los embeddings de palabras son una técnica avanzada de procesamiento del lenguaje natural (NLP) que mapea palabras a vectores de alta dimensionalidad. A diferencia de enfoques tradicionales como "bag of words" y TF-IDF, que representan palabras como índices o frecuencias, los embeddings de palabras capturan contextos semánticos y sintácticos. Uno de los métodos más conocidos para generar embeddings de palabras es "Word2Vec", desarrollado por Google. "Word2Vec" utiliza una red neuronal para aprender estas representaciones.

Conclusiones

La tarea propuesta nos introdujo a una serie de conceptos clave en el procesamiento de lenguaje natural (NLP), como los n-gramas, las técnicas de bag-of-words, y TF-IDF. Además, implicó la aplicación de una serie de herramientas y métricas para la evaluación de modelos de predicción, como la matriz de confusión, accuracy, precision y recall. Este enfoque integral nos permitió no solo entender los fundamentos teóricos, sino también experimentar con las aplicaciones prácticas de estos conceptos.

Respecto a los resultados concretos del análisis, pudimos observar que el uso de PCA no parece ser una herramienta efectiva en el análisis de texto. Una prueba de esto es que las primeras 10 componentes principales alcanzaron apenas una varianza explicada acumulada del 12% en el mejor de los casos. Esto sugiere que la reducción de dimensionalidad mediante PCA no logra capturar la complejidad y la riqueza semántica inherente a los datos textuales.

Por otro lado, vimos cómo modelos de clasificación supervisada, como el Multinomial de Naive Bayes y la Regresión Logística Multinomial, tuvieron rendimientos similares y en general pobres en ambos casos. Ambos modelos mostraron ser muy susceptibles a caer en sesgos a favor de las clases mayoritarias cuando la base de datos de entrenamiento no está bien balanceada en sus clases. Este hallazgo subraya la importancia de utilizar técnicas de balanceo de clases, como el sobremuestreo o el submuestreo, para mejorar la equidad y precisión de los modelos.

Una observación interesante fue cómo diferentes modelos maximizan su rendimiento con diferentes parametrizaciones. Esta variabilidad destaca la necesidad de realizar una optimización de hiper-parámetros cuidadosa y específica para cada modelo y conjunto de datos. La capacidad de ajustar los parámetros adecuadamente puede significar la diferencia entre un modelo mediocre y uno de alto rendimiento.

En resumen, esta tarea no solo nos permitió consolidar nuestro conocimiento sobre las técnicas de NLP y las métricas de evaluación de modelos, sino que también nos enseñó lecciones valiosas sobre las limitaciones y desafíos en el análisis de texto. La experiencia adquirida refuerza la importancia de un enfoque crítico y adaptativo en el diseño y la evaluación de modelos de procesamiento de lenguaje natural.