Procesamiento de Lenguaje natural sobre discursos de candidatos a las elecciones de Estados Unidos 2020

Introducción a la Ciencia de Datos 2025

Análisis temporal, conteo de palabras y menciones mutuas

Introducción

En las elecciones presidenciales de EE. UU. 2020, los discursos públicos de los candidatos reflejaron las prioridades y estrategias comunicativas en un entorno marcado por la pandemia y la polarización política. Este informe se centra en los cinco aspirantes con mayor número de intervenciones.

El análisis cuantitativo y cualitativo de esos discursos permite identificar patrones discursivos, variaciones temporales en el volumen de intervenciones y diferencias en el vocabulario utilizado por cada aspirante. Además, al centrarse en los cinco candidatos con mayor número de discursos, se obtiene una muestra representativa de las narrativas predominantes en los principales partidos políticos y en los eventos más relevantes de la campaña electoral.

Alguna preguntas clave que intenta responder éste análisis son:

¿Cómo varía el volumen de discursos a lo largo de la campaña?
¿Qué vocabulario caracteriza a cada candidato?
¿Con qué frecuencia se mencionan unos a otros?

Datos y Metodología

- Datos: se utiliza una base de datos abierta con discursos políticos en el marco de las elecciones de Estados Unidos en 2020.
- Carga y limpieza de datos: comprobación de valores faltantes (‘df_speeches.info()’) conteo de discursos por candidato y filtrado de los cinco con más registros.
- Análisis temporal: series de tiempo del número de discursos, señalando picos en debates y convenciones.
- Normalización de texto: la función ‘clean_text()’ extiende el preprocesamiento (minúsculas, eliminación de puntuación, dígitos y tokens atípicos).
- Conteo de palabras y frecuencias: uso de Counter para extraer los top-30 términos por candidato y cálculo del total de palabras.
- Análisis morfológico: filtrado de sustantivos y adjetivos mediante ‘pOs(spaCy)’ y ‘WordNet’, con visualizaciones de word clouds para cada categoría.

- Matriz de menciones mutuas: construcción de una matriz 5×5 que cuantifica cuántas veces cada candidato menciona a otro, visualizada también como grafo.
- Diversidad léxica: aunque no se calculó un ratio tipo-token explícito, el desglose de sustantivos y adjetivos ofrece una aproximación a la variedad léxica empleada.

Este conjunto de análisis ofrece la base necesaria para un informe centrado en los hallazgos clave, las tendencias temporales y las dinámicas de vocabulario entre los candidatos.

Limpieza de datos

Realizamos la carga de los datos utilizando la librería pandas por medio del comando 'pd.read_csv' y comprobamos las características de la tabla con el comando 'df_speeches.info()'. Cuenta con 269 filas y seis columnas:

- *speaker*: exponente del discurso
- *title*: título del discurso
- *text*: texto del discurso
- *date*: fecha
- *location*: lugar
- *type*: tipo del dato

Todas las filas tienen valores en las columnas de *title*, *text* y *date*, pero hay 3 que tienen el *speaker* nulo, 18 filas con *location* nula y 21 con el *type* nulo. Con respecto a los valores nulos de estas dos últimas columnas, decidimos rellenarlos con un valor desconocido ('unknown'), para que no nos genere ningún problema posterior. Por otro lado, todas las columnas son de tipo "object" y como necesitamos operar luego con la fecha de los discursos, transformamos la columna *date* a tipo fecha con el comando 'pd.to_datetime'.

Luego analizamos la cantidad de filas por *speaker*, con el fin de contar los discursos que tiene cada exponente. Encontramos que existen exponentes múltiples, por ejemplo, está el debate entre Joe Biden y Donald Trump en conjunto en una misma fila. Nos interesan también este y otros casos, es decir que queremos contabilizarlos como nuevos discursos y conocer qué se dice en ellos, por lo que no los desestimamos. Decidimos tratarlos como discursos nuevos y distintos entre sí, de manera que a este debate de Joe Biden y Donald Trump lo dividimos en dos discursos, dos nuevas filas, uno con lo que dice cada uno. Repetimos este procedimiento para los discursos múltiples que nos interesan puesto que tienen exponentes con posibilidades de ser candidatos con más discursos. Luego de este tratamiento de los datos, la tabla queda con 278 filas.

Los cinco exponentes con más discursos resultantes de esta base de datos trabajada son: los candidatos a presidente Joe Biden y Donald Trump, los candidatos a vicepresidente Mike Pence y Kamala Harris, y Bernie Sanders. Entendemos que los primeros cuatro califican como candidatos, ya sea como candidatos a presidente o como candidatos a vicepresidente. Bernie Sanders, según entendemos, por más que no fue candidato a presidente durante todo el período, sí lo fue en los primeros meses hasta que suspendió su campaña en abril de 2020 y es de nuestro interés analizar también los discursos de un

candidato con estas características, considerando que tiene gran cantidad de discursos, incluso más que Kamala Harris en todo el período considerado.

Resultados y Discusiones

1. Discursos de los candidatos a lo largo del tiempo

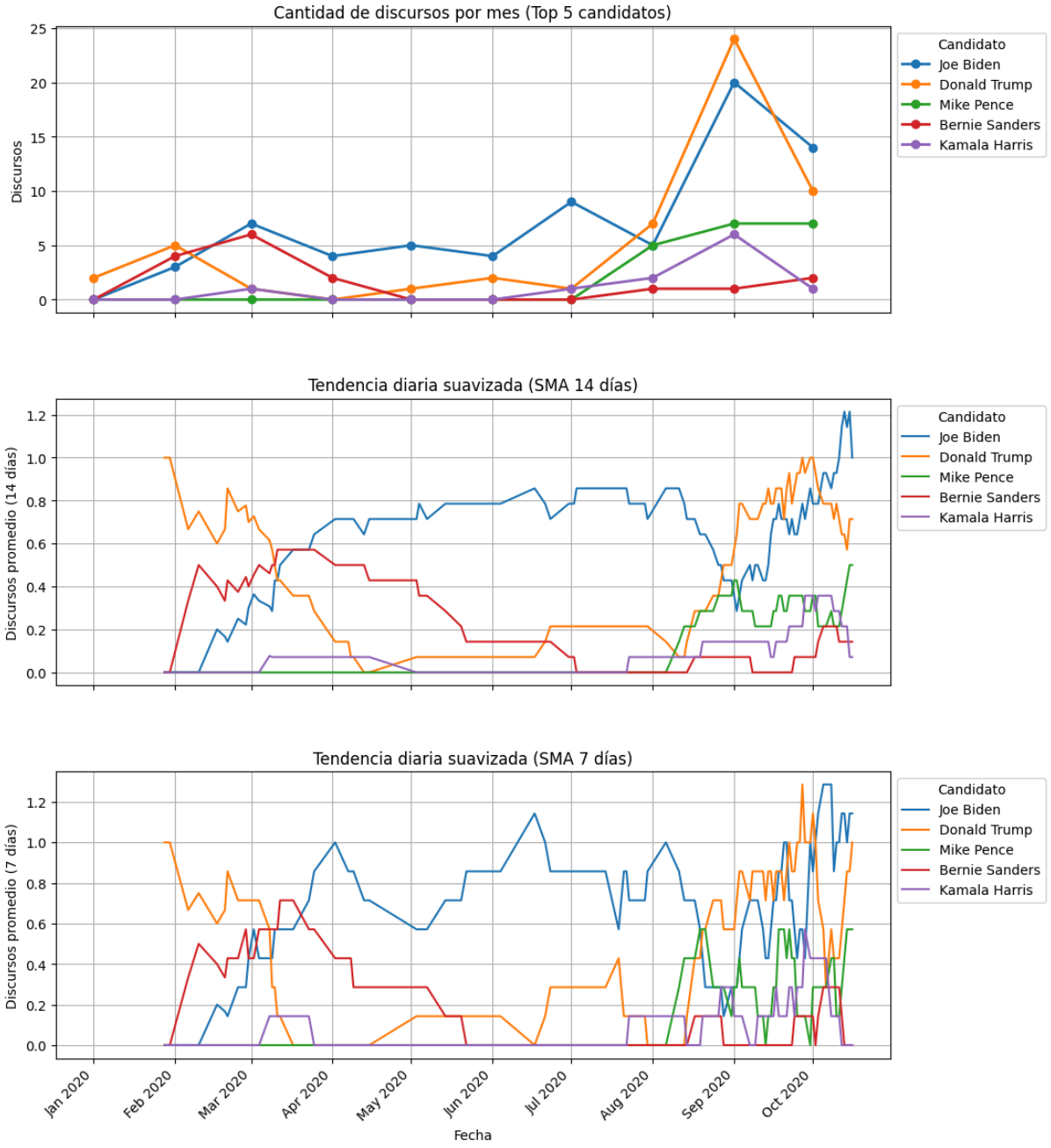
Contexto teórico: Media Móvil Simple(SMA)

Una Media Móvil Simple (Simple Moving Average SMA inglés) es un promedio aritmético que suaviza una serie temporal para revelar su tendencia subyacente. Por ejemplo, en vez de quedarnos con el valor de un solo día, calculamos el promedio de ese día y de los N días anteriores para cada punto en el tiempo. Así, cada nuevo valor de la serie resulta de condensar lo que ocurrió en ese pequeño tramo de días, atenuando los picos y valles aislados y mostrando con más claridad la tendencia general. Si en el arranque no hay suficientes días, basta con promediar los que sí estén disponibles para no dejar espacios vacíos. Finalmente, si quisiéramos centrar ese promedio, en lugar de mirar solo hacia atrás, podríamos repartir la ventana a ambos lados del día en cuestión, obteniendo una curva aún más “alineada” con los máximos y mínimos reales.

¿Para qué sirve?

- Reducir el ruido de corto plazo y destacar la dirección general.
- Comparar horizontes: SMA corta (7 días) muestra reacciones inmediatas; SMA más larga (14 días) revela la tendencia a mediano plazo.
- Alineación de picos: usando ‘center=True’ alinear visualmente picos y valles, pero con desfase, mientras que con ‘center=False’ mantienen fidelidad a la secuencia temporal real.

Tendencias: Discursos



Análisis mensual

En la primera gráfica de esta figura se presenta la cantidad de discursos en cada mes. Vemos que Joe Biden comienza casi inactivo en enero–febrero, sube con fuerza en marzo (7 discursos) y mantiene un flujo moderado (4–5/mes) hasta julio, cuando da un salto a 9. El pico máximo es en septiembre (20) antes de bajar ligeramente en octubre (14).
Donald Trump tiene 2 discursos en enero, tiene un pequeño repunte en febrero (5), luego cae a 1–2/mes hasta agosto, y en septiembre alcanza su máximo (24) —el mes más “movido” globalmente— antes de descender a 10 en octubre.

Mike Pence prácticamente estuvo ausente hasta julio, luego se incorporó con 5 discursos en agosto, 7 en septiembre y 7 en octubre, mostrando su actividad creciente en el último tercio del período.
Bernie Sanders concentró casi toda su actividad en marzo (6) y abril (2), para casi desaparecer de mayo en adelante (1–2/mes o cero). Refleja que su campaña principal fue pre-primarias.
Kamala Harris tuvo una presencia prácticamente nula hasta agosto (2 discursos), luego un pico en septiembre (6) y se reduce en octubre (1). Muestra una entrada tardía, ligada a su anuncio de candidatura y debates de Vicepresidentes.
Lo más relevante es el mes de setiembre, recta final antes de las elecciones, donde Biden y Trump concentran sus máximos de cantidad de discursos.
Los candidatos con menor perfil (Sanders, Harris, Pence) tienen ratos muy focalizados: Sanders en marzo, Harris y Pence a partir de agosto.

SMA 14 días

Con una media 14 días eliminamos casi todo el ruido semanal, revelando tendencias a mediano plazo. Esto se presenta en la segunda gráfica de la figura.
Vemos que Biden ya estabiliza su promedio en ~0.7–0.8 discursos/día desde abril hasta poco antes de la campaña final, cuando sube hacia ~1.0.
Trump muestra claramente tres fases: alta actividad inicial (enero-febrero), declive prolongado y recuperación lenta desde julio hasta ~0.9 en septiembre.
Sanders queda casi “aplanado” después de marzo, y Harris y Pence sólo emergen con una rampa suave en el último tercio.
A grandes rasgos se elimina casi todo el ruido semanal, y se aprecian las tendencias de mediano plazo.

SMA 7 días

En la tercera gráfica de la figura se aprecia una rápida subida de Sanders en marzo, que alcanza ~0.7 discursos/día (equivalente a 5–6 en una semana) y luego cae.
Biden muestra un ascenso constante desde marzo hasta un pico de ~1.2 discursos/día en octubre.
Trump cae fuerte en abril–junio (~0.1–0.2) y luego sube de mayo en adelante, recuperando ~1.3 en septiembre.
Pence y Harris comienzan a marcar tendencia apenas entran en campaña: Harris ronda 0.3–0.5 en septiembre y Pence 0.5–0.6.
O sea se detectan subidas súbitas, ya que la ventana de 7 días conserva más variabilidad.
A grandes rasgos se revela cómo reaccionan los candidatos a eventos concretos (debates, convenciones) con subidas y caídas que no se aprecian en el agregado mensual.

Análisis integrado

En septiembre, cuando el recuento mensual de discursos alcanza su punto más alto para Biden y Trump, vemos en la curva de 14 días que ese repunte no es un simple pico aislado sino una tendencia sostenida: la línea se inclina claramente hacia arriba desde julio,

confirmando que ambos construyeron un gran impulso en esa fase de campaña. Al mismo tiempo, la versión de 7 días resalta las respuestas más inmediatas a eventos puntuales.

Por ejemplo, pequeños saltos en la actividad de Biden tras debates clave o fluctuaciones más abruptas de Trump alrededor de anuncios específicos que el promedio de dos semanas suaviza. Mientras Sanders dispara su actividad en marzo y luego cae bruscamente (algo que la SMA-14 atenúa pero la SMA-7 evidencia con claridad en sus subidas repentinas).

Harris y Pence apenas emergen en agosto, con una pendiente suave en la media de 14 días y picos semanales que delatan su entrada en escena. Así, al superponer recuento mensual y medias móviles de 14 y 7 días obtenemos un relato continuo: los hitos generales, la confirmación de su persistencia y la intensidad de las reacciones inmediatas.

2. Conteo de palabras

Con el objetivo de realizar el análisis cuantitativo y cualitativo de los discursos, necesitamos hacer el conteo de palabras, es decir contar las veces que aparecen las distintas palabras en los discursos de cada candidato según distintos criterios. Para esto normalizamos el texto, pasando todas las palabras a minúsculas, y eliminamos los signos de puntuación (los más comunes incluido el punto y también el signo de exclamación, el de interrogación, los paréntesis, comillas y porcentaje). De esta manera nos aseguramos de que contamos la misma palabra las veces que sea así aparezca de diferentes formas. Un detalle a tener en cuenta es que a las palabras que son compuestas (con apóstrofe) las contamos como tales, por más que puedan escribirse como dos palabras distintas, ya que en el idioma inglés bajo consenso general de la lingüística pueden ser consideradas como palabras.

2.1 Cantidad de palabras por candidato

Con el fin de encontrar los candidatos con mayor cantidad de palabras, contamos la cantidad de palabras de cada uno, lo que se visualiza en la siguiente tabla:

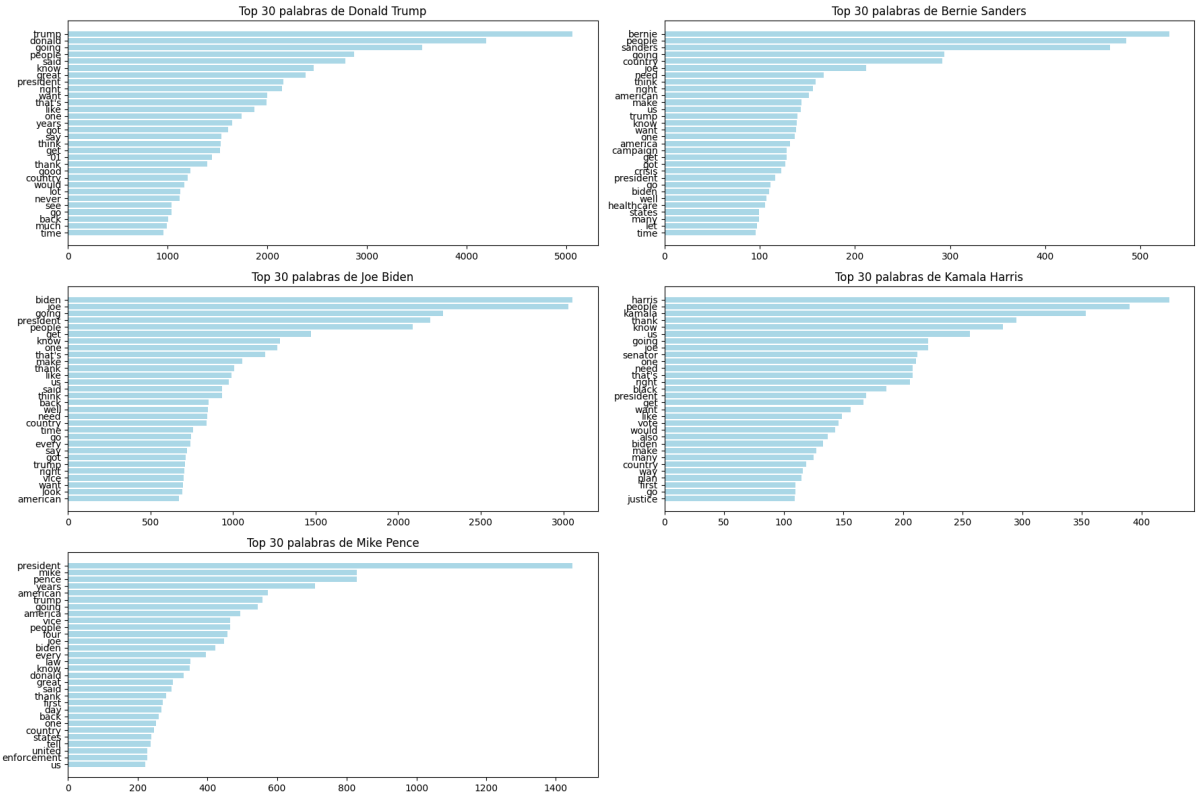
Candidato	Cantidad de palabras total
Donald Trump	533.016
Joe Biden	414.607
Mike Pence	108.661
Kamala Harris	67.966
Bernie Sanders	55.539

Vemos que el candidato con más palabras es Donald Trump, seguido por Joe Biden. Esto indica la cantidad de palabras totales en todos los discursos. Claramente era más probable que los candidatos con más discursos tuvieran más palabras pronunciadas, aunque en este caso particular se invierte el orden entre estos dos candidatos, indicando que Trump pronuncia más palabras por discurso. Si se quisiera saber esto, deberíamos realizar un promedio. Por otro lado, contamos las palabras cada vez que aparecen así estén repetidas, si se quisiera indagar en la calidad del vocabulario usado por cada candidato deberíamos contar la cantidad de palabras únicas.

2.2 Palabras más frecuentes por candidato

Barcharts

Se hizo en primer lugar un conteo inicial de las 30 palabras más usadas por los candidatos. Al visualizar el Top-30 completo, mediante *barcharts*, vimos que las palabras más frecuentes usadas por Biden y Trump eran “the” (~17500 menciones) y “to” (~12000). Para los casos de Sanders, Pence y Harris, estas mismas palabras, apenas superan las 3 000–5 000. Por ello se procedió a la extracción de palabras con un contenido relevante. De asta forma aplicamos la lista de *stopwords* de NLTK, conjunto estándar de palabras vacías en inglés, y filtramos los artículos (“the”, “a”), preposiciones (“of”, “in”) y conjunciones (“and”, “but”) antes de contar.
El efecto fue inmediato como puede verse en la siguiente figura, en los *barcharts* filtrados, Biden pasa a tener en cabeza “biden” (~3100), “joe” (~3000) y “people” (~2200); Trump presenta “trump” (~5200), “donald” (~4000) y “said” (~2200).
En los demás candidatos se aprecian con más peso terminos como “campaign”, “president” o “senator” con recuentos de varios cientos.



Esta visualización podría modificarse con el fin de encontrar diferencias entre partidos políticos por ejemplo indicando con distintos colores las barras de los gráficos que corresponden a candidatos del partido demócrata (Joe Biden, Kamala Harris y Bernie Sanders) y las barras de los gráficos que corresponden a candidatos del partido republicano (Donald Trump y Mike Pence). También se podrían multiplicar estas gráficas por diferentes períodos, viendo por ejemplo las palabras más pronunciadas por cada candidato en el primer mes considerado y por otro lado en el último mes considerado. Por último se podría hacer un análisis discriminando los lugares donde se pronuncian los discursos, por ejemplo visualizar las palabras más frecuentes de cada candidato en discursos virtuales.

Elección de Biblioteca: identificación y conteo de sustantivo, verbos y adjetivos

En un primer acercamiento empleamos WordNet por su extenso inventario léxico, pero pronto comprobamos que ese filtro puramente estático solo verifica la existencia de una palabra sin analizar su uso real: WordNet te dirá que “record” puede ser sustantivo o verbo, pero no sabe en qué contexto aparece en tu discurso. En cambio, spaCy POS-tagging tokeniza cada texto, o sea, separa palabras y puntuación y asigna a cada token una categoría gramatical (sustantivo, verbo, adjetivo) basándose en el contexto, y por tanto, generando una desambiguación de los casos, como por ejemplo: “record the speech” vs. “set a record”.

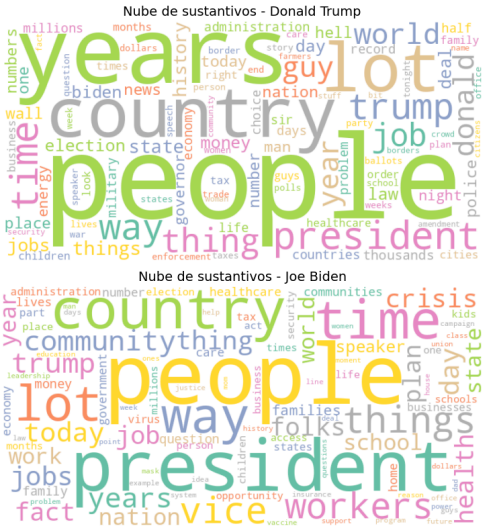
Además, spaCy aplica lematización agrupando palabras relacionadas a su significado fundamental, por ejemplo agrupa “running” con “run”, “children” con “child”), y gracias a su implementación en Cython, que compila el código crítico a C, y a su arquitectura modular

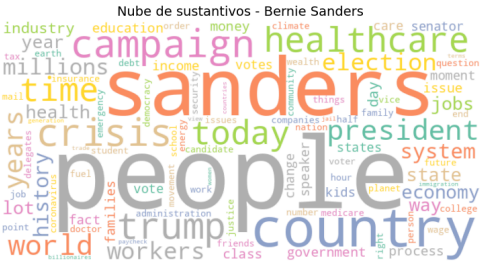
hacen que sea rápido y eficiente. Por estas razones, spaCy se convirtió en la base de nuestro análisis gramatical.

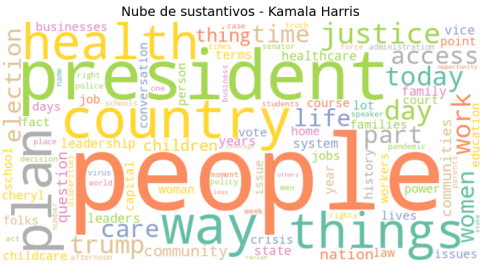
Extraer simultáneamente **sustantivos**, **verbos** y **adjetivos** nos brinda una visión muy completa de los discursos. Los sustantivos señalan entidades y conceptos clave; los verbos revelan acciones y compromisos (“prometer”, “construir”) y los adjetivos aportan matices valorativos (“importante”, “nuevo”).

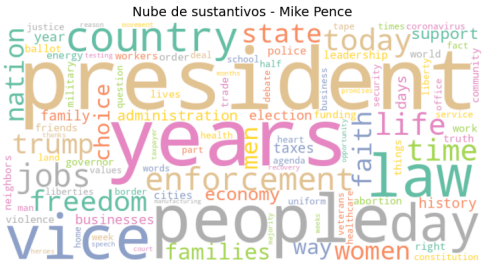
Wordclouds

Para representar estas tres categorías, generamos word clouds que capturan tanto de qué hablan, así de cómo presentan sus principales ideas. Este enfoque filtra el ruido y destaca los términos de verdadero peso. De esta manera vemos en la siguiente figura que “country” y “people” están dentro de los sustantivos más utilizados por los cinco candidatos y además que “president” está entre los más utilizados de todos excepto en Sanders, lo que se puede explicar por la suspensión temprana de su candidatura. En cambio, menciona mucho las palabras “campaign”, “workers” y “healthcare”. En la nube de palabras de Trump, podemos ver las palabras “job”, “wall”, “hell”, etcétera, y en la nube de palabras de Biden vemos “crisis”, “vice”, “way”, etcétera.









2.3 Menciones mutuas

Normalización de los nombres

En los textos originales las referencias a un candidato aparecen de formas muy variadas ("Joe", "Biden", "President Biden", etc.), lo que dificulta un conteo confiable. Para unificar todas esas variantes en una sola cadena consistente ("Joe Biden", "Donald Trump", etc.) aplicamos un proceso de tres pasos:
Eliminar encabezados(columna text): quitamos todo lo anterior al primer salto de línea para descartar timestamps o metadatos.
Unificar en apellidos: reemplazamos cada "Nombre Apellido" por su apellido ("Joe Biden" → "Biden"), asegurando que cualquier mención completa quede registrada como una única unidad.
Reconstruir nombres completos: transformamos cada apellido de vuelta a "Nombre Apellido" ("Biden" → "Joe Biden"), de modo que todas las menciones finales sean idénticas y contables sin ambigüedades.

Matriz de menciones

Con los nombres normalizados, construimos una **matriz de menciones** 5×5 —donde la fila i y la columna j indican cuántas veces el candidato i mencionó al candidato j, sumando para cada discurso, las ocurrencias exactas de "Nombre Apellido".

A continuación se presenta dicha matriz, la cual revela quiénes se citan a sí mismos más veces y cuánto se citan entre pares de candidatos.

menciona a:	**Joe Biden**	**Donald Trump**	**Mike Pence**	**Bernie Sanders**	**Kamala Harris**
Joe Biden	3298	724	29	59	95
Donald Trump	1391	5551	256	244	107
Mike Pence	462	588	840	11	82
Bernie Sanders	241	144	0	530	1
Kamala Harris	281	122	8	5	438

A partir de esta matriz encontramos en primer lugar que existe una tendencia a autopromocionarse. En la diagonal se destacan los números de auto-mención de Trump y Biden, con valores mucho más grandes que los de sus compañeros de fórmula o rivales secundarios. Esto es en parte porque también son quienes tienen ayor cantidad de discursos, pero comparando con las menciones al resto de los candidatos vemos que ambos concentraron gran parte de su discurso en sí mismos. Mientras tanto, Pence, Harris y Sanders o bien se centraron más en hablar de otros o lo hicieron con menor volumen personal.

También en cuanto a los enfrentamientos principales, los valores más altos se encuentran entre Trump y Biden. Trump menciona a Biden una mayor cantidad de veces que a la inversa. Esto refleja que la campaña giró, sobre todo, alrededor de su rivalidad directa. En cambio, las referencias de los vicepresidentes a los candidatos del otro bando son mucho más moderadas.

Posibles preguntas complementarias al análisis

¿Quiénes son los centros de la conversación tanto por la atención que concentran como por la que generan en el debate entre candidatos?

Para responder a esta pregunta, imaginamos a cada candidato como un punto en un plano donde el eje X refleja cuántas veces él mismo alude a otros (usando el método G.out_degree(weight='weight') de NetworkX) y el eje Y muestra cuántas veces los demás lo citan (con G.in_degree(weight='weight')). Aquellos situados en la esquina superior derecha son los auténticos “centros de gravedad” del discurso, pues no solo lanzan el debate hacia sus rivales con frecuencia, sino que también concentran la atención del resto.

¿Cómo se distribuyen geográficamente los discursos de campaña, y en qué estados se concentraron más intervenciones de los candidatos?

Para responder a esta cuestión, se podría partir de la columna “location” de df_speeches que contiene cadenas del tipo “estado, ciudad” en dónde se brindaron los discursos, y posiblemente, eliminando las versiones en cadenas de television. A partir de allí:
Se podrían extraer los valores, limpiando espacios y unificando mayúsculas, de modo que cada registro cuente con dos campos separados. A continuación, emplearíamos una librería como GeoPandas junto con Geopy para traducir esos nombres de lugares a coordenadas espaciales, de modo que cada discurso se convierta en un punto cartográfico. Con esa información, integraríamos los recuentos de intervenciones en un mapa base de los estados de EE. UU., usando herramientas de visualización como Folium o Plotly para colorear cada polígono estatal según la cantidad de eventos allí celebrados. El resultado es un mapa intuitivo que destaca de un vistazo los “puntos calientes” de la campaña.

¿Qué cadenas televisivas cubrieron con mayor frecuencia los encuentros de cada candidato y, en conjunto, cuál fue la red que más los entrevistó?

Antes de nada, revisaríamos todos los valores de la columna location para muestrear cada cadena mencionada y compararlos con una lista de referencia de emisoras de EE. UU. (ABC, NBC, CNN, CBS, Fox, etc.). De esta forma, podríamos detectar variantes (“ABC News”, “ABC-US”, “American Broadcasting Co.”) y asegurarnos de que ninguna se nos escape.

A continuación, unificaríamos cada una de esas variantes bajo el nombre oficial de la red (por ejemplo, todo “ABC News” o “ABC-US” pasaría a “ABC”), de modo que al agrupar realmente sumemos todas las entrevistas para cada estación. Luego, con esos valores ya limpios, agruparíamos por candidato y por cadena para contar cuántas veces cada aspirante participó en ABC, NBC, CNN, etc., y al mismo tiempo totalizaríamos las apariciones de cada red para ver cuál dominó la cobertura de debates y town halls. Finalmente, lo presentaríamos en un gráfico claro —barras apiladas o heatmap— que muestre de un vistazo qué cadena entrevistó más a cada candidato y cuál fue la más activa en toda la campaña.

Entrenamiento y evaluación de modelos

Introducción

Los discursos de los candidatos en unas elecciones son importantes para determinar la comunicación de sus ideas, los temas sobre los que ponen foco e influir en los votantes. Nuestra motivación es analizar el contenido de los discursos de los tres candidatos a las elecciones en Estados Unidos en el 2020 que más discursos expusieron para explorar hasta qué punto se pueden distinguir entre sí a partir del uso del lenguaje.

En este informe se abordó esta cuestión mediante técnicas de procesamiento de lenguaje natural y aprendizaje automático. El objetivo principal es construir un modelo capaz de predecir a qué candidato pertenece un discurso, basándose en su contenido textual. Para esto se implementan transformaciones de los textos mediante representaciones numéricas y se implementa y evalúa un clasificador basado en el modelo *Multinomial Naive Bayes*, explorando distintos enfoques de vectorización, estrategias de reducción de dimensión y búsqueda de hiperparámetros.

Pre-procesamiento del dataset

El análisis parte de un conjunto de datos que contiene discursos políticos correspondientes a la campaña electoral de Estados Unidos en 2020. Cada fila del archivo original representa un discurso completo, acompañado de información contextual como el nombre del orador, la fecha, el título del evento, la ubicación geográfica y el tipo de discurso.

Con el objetivo de facilitar el análisis textual posterior, se realizó un proceso de preprocesamiento orientado a organizar y limpiar los datos. El primer paso consistió en dividir cada discurso en fragmentos individuales asociados a quién estaba hablando en ese momento. Para ello, se aprovechó el formato regular de las transcripciones, que indican explícitamente cuándo comienza a hablar cada orador mediante expresiones del tipo “Nombre: (mm:ss)”. A partir de este patrón, se extrajeron los párrafos correspondientes a cada intervención, generando un nuevo conjunto de datos con una fila por párrafo y orador.

A cada discurso original se le asignó un identificador único (‘speech_id’) que permitió mantener el vínculo entre los párrafos segmentados y el discurso del que provenían. Esta información se utilizó más adelante para reconstruir los discursos completos por orador.

Dado que un mismo orador puede aparecer con múltiples variantes de nombre (por ejemplo, “President Donald Trump” y “Donald Trump”), se realizó un proceso de unificación. Se definieron reglas específicas para los cinco candidatos más relevantes del conjunto, y se agruparon todas sus variantes bajo un mismo nombre estandarizado. Este paso fue fundamental para asegurar la correcta contabilización y análisis posterior.

Con los nombres ya normalizados, se agruparon nuevamente los párrafos correspondientes a cada orador dentro de cada evento, concatenando sus textos para reconstruir discursos

completos por orador. El resultado fue un nuevo conjunto de datos, más limpio y estructurado, en el que cada fila representa una única participación de un orador en un evento determinado.

Sobre estos textos reconstruidos se aplicó una limpieza adicional. Se eliminaron signos de puntuación, se convirtieron todos los caracteres a minúsculas y se normalizaron otros aspectos ortográficos básicos. Esta versión depurada del texto se almacenó en una nueva columna ‘CleanText’, que será utilizada en la representación numérica de los discursos.

Finalmente, se identificaron los oradores con mayor cantidad de discursos. El objetivo era quedarnos con los tres que tuvieran más discursos, pero resultó que luego de la limpieza y reorganización de los datos, se incrementó el número de discurso de todos los oradores importantes con respecto a la contabilización original y resultó un empate en cantidad entre Mike Pence y Bernie Sanders. Se tomó el criterio de elegir de los dos el orador que tenía mayor cantidad de discursos originalmente, Mike Pence, en el entendido de que si su nombre aparecía como speaker más veces, era más veces el orador principal a lo largo de los discursos. Posteriormente, se filtró el dataset para conservar únicamente las intervenciones de los tres oradores elegidos. Este conjunto reducido, pero representativo, es el que se utilizará en el resto del análisis.

División del conjunto de datos: entrenamiento y evaluación

Como se indicó, el objetivo del análisis es construir un modelo capaz de predecir a qué candidato corresponde un discurso, basándose únicamente en su contenido textual. Para entrenar este modelo de forma controlada y evaluar su capacidad de generalización, se dividió el conjunto de datos en dos subconjuntos: uno destinado al entrenamiento y otro reservado para la prueba posterior.

La partición se realizó mediante la función ‘train_test_split’, provista por la biblioteca scikit-learn. Esta función permite dividir el dataset de manera aleatoria, especificando el tamaño del conjunto de evaluación y la estrategia de muestreo. En este caso, se destinó un 30% de los datos al conjunto de testeo (‘test_size’=0.3) y se utilizó muestreo estratificado a través del parámetro ‘stratify', tomando como criterio la columna ‘paragraph_speaker’. Esta decisión garantiza que la proporción de discursos de cada candidato se mantenga constante en ambos subconjuntos. Además, se fijó el parámetro ‘random_state’ en 42 para asegurar la reproducibilidad de los resultados.

A partir de la partición, se definieron los vectores de datos ‘X_dev’ e ‘X_test’, que contienen los textos limpios (‘CleanText’) correspondientes a los conjuntos de entrenamiento y test, respectivamente. Sus etiquetas asociadas (es decir, los nombres de los oradores) se almacenaron en las variables ‘y_dev’ e ‘y_test’.

Para validar que el muestreo estratificado se aplicó correctamente, se calculó la frecuencia relativa de discursos por orador en ambos subconjuntos. En la Tabla 1 se muestran las cantidades y también las proporciones de dichos oradores dentro de cada subconjunto. Se puede ver que las proporciones observadas fueron prácticamente idénticas, siendo Joe Biden el orador de cerca del 51% de los discursos en ambos conjuntos, Donald Trump

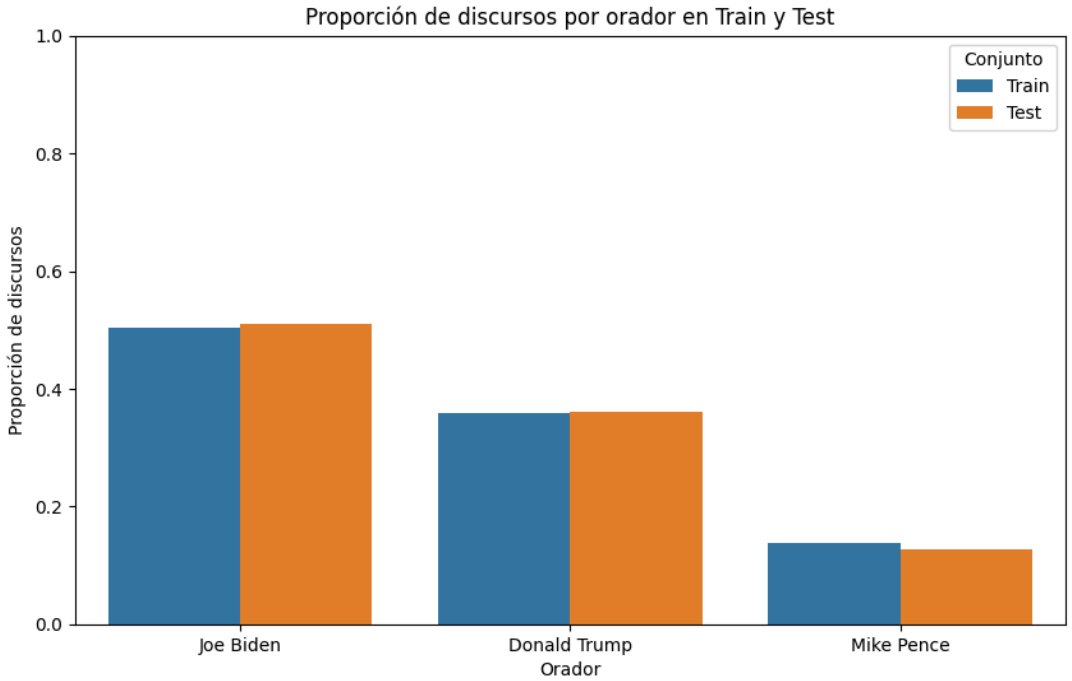
orador de cerca del 35%, y Mike Pence de cerca del 13%, lo que confirma que se preservó el balance original entre clases. El conjunto de entrenamiento resultó conformado por 109 discursos (70% del total), y el de evaluación por 47 (30% del total).

Tabla 1

Orador	**Train (abs)**	**Train (%)**	**Test (abs)**	**Test (%)**	**Total**
Joe Biden	55	50.5%	24	51.1%	79
Donald Trump	39	35.8%	17	36.2%	56
Mike Pence	15	13.8%	6	12.8%	21
Total	**109**	**100%**	**47**	**100%**	**156**

Para complementar la validación numérica del muestreo estratificado, se construyó una visualización gráfica, la Figura 1, que muestra la proporción de discursos de cada candidato en los subconjuntos de entrenamiento y prueba. No solo se ve que en ambos subconjuntos (entrenamiento en azul y prueba en anaranjado) las proporciones de discursos por orador se conservan muy similares, sino que la proporción de discursos de Joe Biden es la mayor con poco más del 50% y la de Mike Pence la más pequeña con menos del 15%. Por un lado, esto nos da una validación visual adicional del muestreo estratificado, ayudando a confirmar que no habría sesgos de clase introducidos por la partición y por otro lado nos puede advertir de que la menor cantidad de discursos de Mike Pence nos puede generar dificultades en el momento de la clasificación o de la predicción.

Figura 1



Transformación del texto a representación numérica

Los algoritmos de aprendizaje automático no permiten procesar texto “crudo” (string), necesitan vectores numéricos como entrada, por lo que, como se está trabajando con discursos, es necesario primero convertir los documentos de texto en un formato numérico que los modelos de *machine learning* puedan usar para entrenar.

La forma más intuitiva de hacerlo[1] es usando la representación “*Bag of Words*” (bolsas de palabras):

1 - Se asigna un número entero fijo j a cada palabra que aparezca en cualquier documento del conjunto de entrenamiento, construyendo así un diccionario que asocia palabras con índices enteros

2 - Se cuenta para cada documento i la cantidad de veces que aparece la palabra representada con el número j

3 - Se almacena el valor en la matriz X en la posición [i,j]

Almacenar X como una matriz ‘NumPy’ numérica decimal ‘float32’, es decir guardando 32 bits (4 bytes) por número, no sería manejable. Las palabras distintas en el diccionario suelen ser más de 100.000 y si se trabajaran con 10.000 documentos, el tamaño de la matriz X sería de 10.000 x 100.000, lo que multiplicado por los 4 bytes por número daría un total de 4GB de RAM.

Afortunadamente, la mayoría de los valores en X son ceros por lo general, ya que en cada documento hay un bajo porcentaje de las palabras del diccionario utilizadas. Un documento típico usa menos de unos miles de palabras distintas. Por esta razón, decimos que las representaciones *Bag of Words* son conjuntos de datos de alta dimensión y dispersos (*sparse datasets*). Se puede ahorrar mucha memoria almacenando solo las partes no nulas de los vectores de características y las posiciones donde aparecen. Esto, junto con otras cosas, es lo que hace ‘CountVectorizer()’ de la librería *scikit-learn*.

Ejemplo:

```
docs= [ "me gusta el mate y el café",
        "el café me gusta mucho"]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(docs)
```

Tabla 2

	el	gusta	mate	me	mucho	y	café
doc0	2	1	1	1	0	1	1
doc1	1	1	0	1	1	0	1

[1] https://scikit-learn.org/1.4/tutorial/text_analytics/working_with_text_data.html

Los ceros no son guardados, tan solo las ocurrencias y sus posiciones. Por el hecho de que no guarda las palabras en un orden necesario es que se llama “*Bag of Words*”.

La clase ‘CountVectorizer()’ por defecto elimina las palabras vacías (*stop words*) y signos de puntuación y también cuenta palabras sueltas, pero soporta combinaciones de palabras, es decir, n-gramas. Estas son secuencias contiguas de n elementos extraídos de un texto (pueden ser palabras, caracteres o incluso sílabas, según el nivel de granularidad que se desee). En este informe se trabajó con unigramas (*1-grams*) y bigramas (*2-grams*). Los bigramas permiten capturar el contexto local añadiendo información estructural, lo que mejora la desambiguación y precisión. Por ejemplo, un unigrama no distingue “New” de “New York”, pero un bigrama sí identifica esa combinación como unidad. Un problema que surge al aumentar n es que el tamaño del vocabulario crece exponencialmente, lo que dispara la dimensionalidad y hace a la matriz de características más dispersa. Por este motivo, hay mayor uso de memoria y un mayor coste computacional, además de aumentar el riesgo de sobreajuste si no hay suficientes datos. Se debe tener esto en cuenta a la otra de elegir el modelo de ajuste.

Por otro lado, contar las ocurrencias puede conllevar un problema: los documentos más largos tendrán valores promedio más altos que los documentos más cortos, aunque hablen de lo mismo. Es decir, si simplemente se cuentan cuántas veces aparece cada palabra, los textos largos tendrán más palabras, entonces los modelos podrían confundir "muchas palabras" con "más importancia", por lo que se necesitan transformaciones adicionales:

- *Term Frequency* (TF): Divide la cantidad de ocurrencias de cada palabra por el total de palabra del documento para evitar las discrepancias en cuanto al largo de los documentos. Esto es la frecuencia relativa: cuánto representa una palabra dada dentro del texto, lo cual es indispensable medir.

- *Inverse Document Frequency* (IDF): Otro refinamiento, que como no es indispensable debe probarse su uso en distintos modelos, consiste en reducir el peso de las palabras que aparecen a través de varios documentos y, por lo tanto, son menos informativas y no ayudan a distinguirlos.

Entonces esta última transformación '*Term Frequency times Inverse Document Frequency*' se hace para incorporar la frecuencia de una palabra en un documento (TF) y penalizar las palabras comunes dándole más peso a las más raras (IDF). Luego de aplicar este procedimiento a los discursos de los tres candidatos del subconjunto de entrenamiento, las dimensiones de la matriz con TF-IDF fue de 109 discursos por 11.795 palabras distintas.

Análisis de Componentes Principales

El Análisis de Componentes Principales o PCA es una técnica de reducción de dimensiones usada en estadística, machine learning y análisis de datos. Consiste en transformar un

conjunto de variables posiblemente correlacionadas en un nuevo conjunto más pequeño de variables no correlacionadas llamadas componentes principales, que retienen la mayor parte de la variabilidad (información) del conjunto original. Como sirve para reducir la estructura de los datos, sirve para la exploración de estos, detectando patrones, grupos, anomalías y las combinaciones de variables que explican más la variación. Si los resultados arrojan con este análisis que pocos componentes explican mucha de la variabilidad de los datos, se podría concluir que las variables están muy correlacionadas por lo que se puede posteriormente eliminar la colinealidad, ayudando a combatir el overfitting al reducir el ruido. Esto es necesario como análisis previo para poder realizar una modelización acorde a la estructura de los datos y, por ende, con mayor capacidad de predicción.

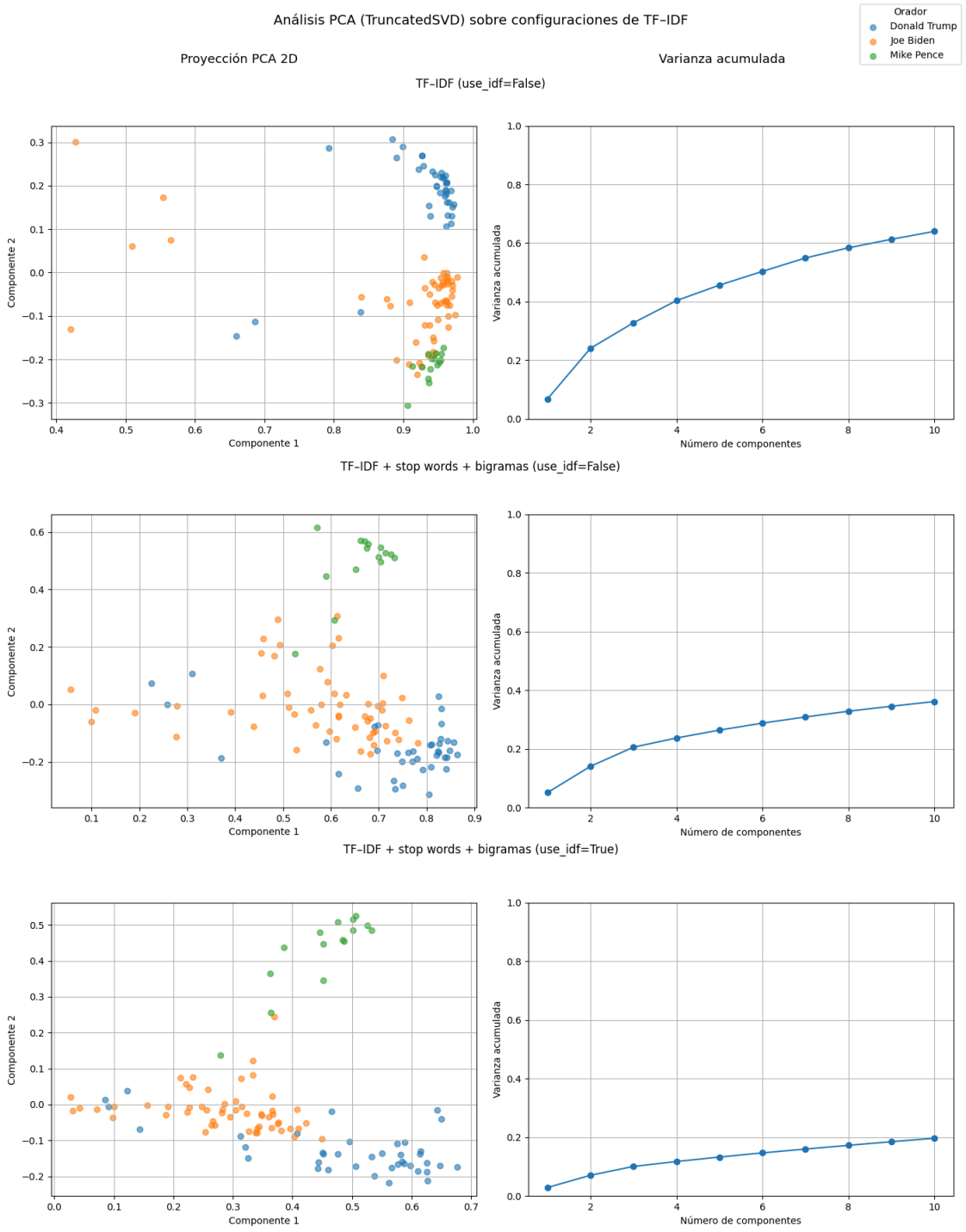
Con la transformación del texto de los discursos a la representación numérica hecha, se realizó este análisis PCA, con el fin de explorar la forma que tenían los datos por medio de una representación gráfica. Sin embargo, se encontró que la PCA clásica puede tener algunos problemas, por lo que se decidió optar por el análisis de componentes principales 'TruncatedSVD', que es una técnica de reducción de dimensionalidad, parecida a la PCA clásica, pero diseñada especialmente para datos dispersos (sparse), como los que se obtienen con Bag of Words o TF-IDF. También es una clase incluida en la librería de scitik-learn. Mientras que PCA usa la matriz de covarianza, 'TruncatedSVD' opera directamente sobre la matriz original, truncando los valores singulares más importantes, lo que lo hace más eficiente con matrices dispersas. La PCA centra los datos (resta la media) y convierte cualquier sparse en una matriz densa, disparando el uso de memoria. Esta técnica alternativa, en cambio, trabaja sin necesidad de centrar ni densificar. Es mucho más eficiente en tiempo y memoria, escala mejor para grandes cantidades de documentos y vocabularios. Además, es la técnica recomendada por scikit-learn para reducción de dimensión en NLP y, utilizando '.explained_variance_ratio_', se obtiene una buena aproximación de la varianza explicada por los componentes.

Considerando el amplio set de herramientas provistos por scikit-learn, se aplicó el 'TruncatedSVD' a diferentes escenarios de prueba con diferentes transformaciones de los vectores TF-IDF:

1. 'use_idf=False': sin aplicar el factor IDF, es decir, sin ponderar más las palabras que son raras en todos los documentos.
2. 'use_idf=False' - stop words + bigramas: sin aplicar IDF pero sí eliminando las palabras vacías (stop words) en inglés e incluyendo combinaciones de dos palabras contiguas (n-gramas con n=2).
3. 'use_idf=True' - stop words + bigramas: aplicando IDF, eliminando stop words e incluyendo bigramas.

En la Figura 2 se muestran los resultados, tal que las gráficas de la izquierda muestran en cada caso la representación en forma de puntos de los documentos en los dos componentes principales y las gráficas de la derecha muestran la varianza explicada acumulada en función de los componentes principales considerados.

Figura 2



En las gráficas de más arriba (escenario 1.) se ve el escenario en que no se aplica el IDF. Hay casos que están acumulados para mayores valores del componente 1 y algunos pocos casos atípicos para valores bajos. Esto puede ser explicado por la falta de penalización de palabras comunes. Documentos que contienen palabras muy frecuentes se acumulan en una región del espacio, mientras que documentos que usan palabras más inusuales se despegan hacia otra región distinta. Esto es lo que puede estar generando concentraciones artificiales en el espacio vectorial. Por otro lado, el componente 2 pareciera estar ordenando los discursos por orador, aunque no podrían distinguirse los discursos de Biden y Pence como grupos claramente separados. En la gráfica de varianza se ve que la varianza

explicada acumulada no llega a 70% en el componente 10. Este valor es bastante bajo e incluso puede estar inflado por el hecho de que las palabras que son frecuentes en todos los discursos dominan los vectores TF, generando una estructura muy direccional en el espacio de los documentos.

En las gráficas del medio (escenario 2.) se difumina la concentración de documentos a un lado del componente 1. Los documentos representados en puntos se dispersan en el espacio y quizá los que más pueden distinguirse ahora son los discursos de Pence. Se eliminaron los stop words, por lo que ahora los vectores son menos parecidos entre sí, ya que se redujo la correlación artificial entre documentos y se empiezan a captar diferencias más sustantivas. Por otro lado, los bigramas son más específicos y reflejan temas concretos. Sin embargo, la varianza explicada es menor para cada cantidad de componentes. Esto es debido a que ahora el espacio vectorial tiene más dimensiones informativas y menos redundancia, lo que reparte la varianza entre más componentes. Es decir, la varianza explicada acumulada por los primeros componentes es baja aunque el análisis represente mejor las diferencias entre los documentos. En este caso, la mayor dispersión debida a la diferenciación semántica real necesita más dimensiones para explicar la varianza.

Este fenómeno se profundiza en el tercer escenario, en el que además de eliminarse las stop words y utilizarse bigramas, se aplica el IDF. En las gráficas de más abajo de la Figura 2 se representa esta situación. La de la izquierda muestra que el mapa se dispersa aún más debido a que se le quita peso a las palabras que son frecuentes en todos los documentos, dejando menos representaciones de puntos aislados. También se ve que los discursos de Biden y Pence se diferencian un poco más que en el escenario anterior. En el último gráfico se ve que la varianza explicada acumulada por componente utilizado es aún menos que en el escenario anterior, llegando apenas al 20% con el décimo componente. Esto, como se venía viendo en el segundo escenario, no es necesariamente un problema, sino más quizá un reflejo de que se están representando los textos con más fidelidad semántica, sin dominio de palabras comunes y redundantes y con una estructura de datos menos concentrada.

En conclusión, a medida de que se realizan transformaciones que tienen sentido en los textos, la varianza acumulada explicada por los primeros componentes disminuye. Y esto tiene sentido, ya que es el contenido real semántico el que está originalmente disperso y estos parámetros lo que hacen es captarlo, por lo que los componentes principales van a captar una menor proporción de esa información. Con la eliminación de las stop words se reduce la cantidad de palabras comunes, lo que reduce la repetición estructural. Con el uso de bigramas se aumenta la cantidad de features, lo que vuelve al espacio más disperso. Esto puede generar una sobredimensionalidad, por lo que hay que probar algunos modelos que lo usen y otros que no, para concluir si es mejor incorporarlo o evitar un sobreajuste innecesario o perjudicial. Por último, es probable que en los primeros escenarios las palabras frecuentes dominaran la varianza, y con la aplicación del IDF se atenúa su contribución.

Aún en el primer escenario, que es el peor especificado y con el que se obtiene mayor varianza explicada acumulada por cantidad de componentes principales, con dos componentes se está explicando muy poca varianza, poco más del 20%. Esto quiere decir

que no se pueden separar los candidatos utilizando solo dos componentes principales y en ningún caso los mapas realmente pueden mostrar clases claramente diferenciadas.

Por último, también se analizó el caso en el que se mantienen los signos de puntuación (‘token_pattern=r"(?u)\b\w[\w.,!?;:'\"-]*\b"’), quedando los vectores TF-IDF con un largo de 13.374 “palabras” distintas (diferenciando las que mantenían algún signo de puntuación adyacente y las que no). Este caso no hizo variar mucho la estructura de datos con respecto al escenario 3 y la varianza acumulada aumentó débilmente, lo que probablemente se debe a que la diferenciación hace a la matriz TF-IDF más artificialmente dispersa, con una varianza que los componentes pueden captar, pero que no ayuda al análisis de la diferenciación semántica real entre discursos.

Clasificación y evaluación usando el modelo Multinomial Naive Bayes

Como se explicó con anterioridad, al transformar los discursos en vectores numéricos mediante técnicas como Bag of Words o TF-IDF, es posible aplicar modelos de aprendizaje automático para clasificarlos según su contenido. Una de las suposiciones razonables en este contexto es que los documentos, representados como conteos de palabras, pueden modelarse a través de una distribución multinomial. Esta distribución describe la probabilidad de observar distintas combinaciones de frecuencias para un conjunto de categorías, y en este caso se asume que cada clase (por ejemplo, un candidato específico) tiene una distribución característica sobre el vocabulario. Así, un documento se interpreta como un conjunto de conteos de palabras, donde algunas tienen mayor o menor probabilidad de aparecer según la clase a la que pertenece el texto.

El modelo Multinomial Naive Bayes parte de esta idea. Estima, para cada clase, las probabilidades de aparición de las palabras en los documentos asociados, y luego utiliza el teorema de Bayes para calcular la probabilidad de que un nuevo documento corresponda a una clase determinada. Aunque el modelo asume independencia entre las palabras, simplificación que rara vez se cumple en lenguaje natural, suele funcionar bien en clasificación de texto gracias a su simplicidad, velocidad y capacidad para captar patrones de uso del lenguaje entre clases.

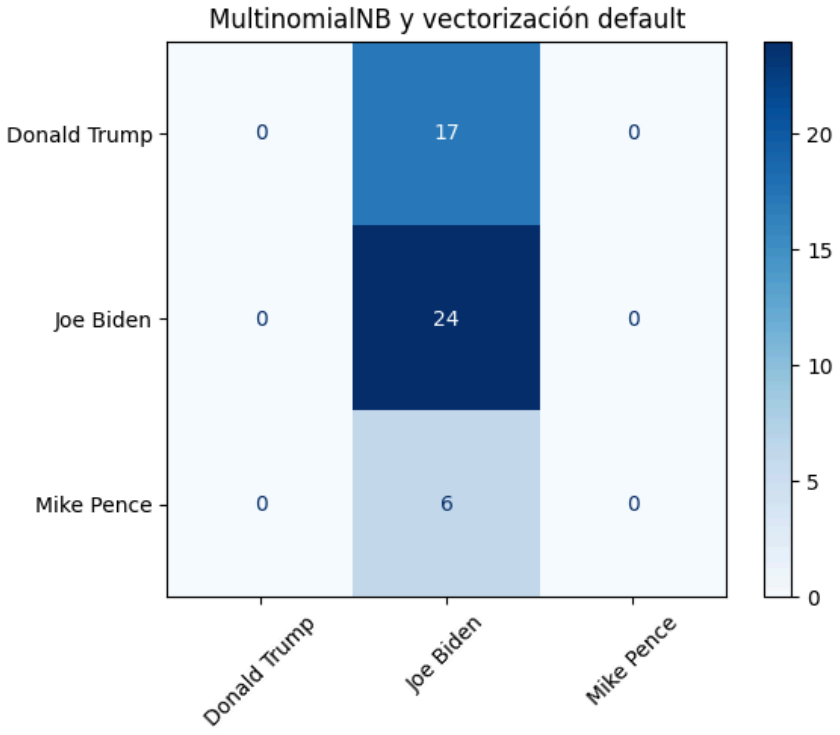
Se entrenó el modelo ‘MultinomialNB’ de la biblioteca scikit-learn, dejando los valores por defecto en sus hiperparámetros, y también una vectorización por defecto, con stop words y unigramas. Esta configuración inicial sirvió como punto de partida para evaluar la capacidad del modelo para distinguir entre los tres candidatos más frecuentes del conjunto de datos. A partir de este entrenamiento, se realizaron predicciones sobre el conjunto de testeo para medir su rendimiento.

Para interpretar correctamente el desempeño del modelo, se utilizaron métricas habituales en clasificación como accuracy, precision, recall y F1-score, cada una con un enfoque complementario. El accuracy indica el porcentaje de predicciones correctas sobre el total de ejemplos evaluados, pero puede resultar engañoso en contextos con clases desbalanceadas. Por eso se recurre a métricas más específicas: la precision mide cuántas de las predicciones hechas para una clase fueron correctas; el recall evalúa cuántos de los ejemplos reales de esa clase fueron efectivamente detectados por el modelo; y el F1-score combina ambos aspectos en un solo valor, representando un equilibrio entre precisión y

cobertura. Estas métricas, calculadas individualmente para cada clase, permiten analizar con mayor detalle los errores del modelo.

El modelo entrenado fue evaluado sobre el conjunto de test, obteniendo un valor de accuracy de 0.51. A primera vista, este valor podría parecer aceptable, pero en realidad resulta engañoso. El modelo predice casi todos los discursos como si fueran de Joe Biden, quien representa poco más del 50% de los ejemplos. Por lo tanto, acierta únicamente cuando se trata efectivamente de discursos de ese orador, y falla sistemáticamente en los casos correspondientes a Donald Trump y Mike Pence.

Figura 3



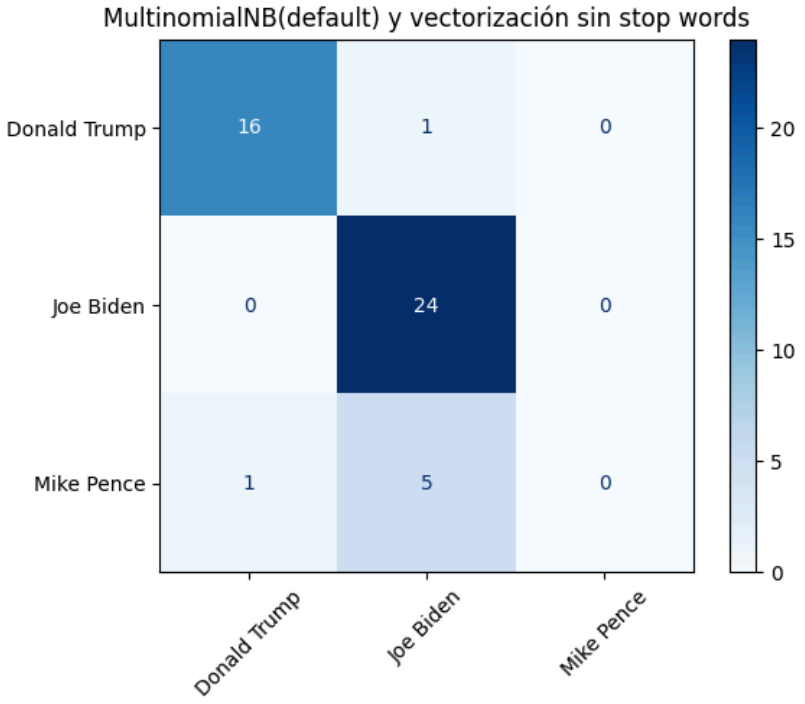
Esta limitación se hace evidente al analizar la matriz de confusión obtenida (Figura 3). En ella, cada fila representa la clase real (el orador correcto del discurso) y cada columna la clase predicha por el modelo. Idealmente, los valores deberían concentrarse en la diagonal principal, lo que indicaría una buena capacidad de predicción. En cambio, se observa que el modelo tiende a predecir sistemáticamente la clase “Joe Biden”, incluso cuando el discurso pertenece a otro orador. Como resultado, todos los valores se acumulan en una sola columna, mientras que las restantes permanecen vacías.

Joe Biden es el único orador con valores diferentes de cero en precision y recall, mientras que Donald Trump y Mike Pence no reciben ninguna predicción correcta y sus valores son cero. Los promedios globales (‘macro’ y ‘weighted’) también se ven arrastrados por esta situación, reflejando un rendimiento limitado del modelo para capturar la diversidad del conjunto de datos. Estos valores evidencian limitaciones importantes en la configuración actual del modelo, lo que motivó la exploración de ajustes en el preprocesamiento del texto para mejorar su desempeño

Ante las limitaciones observadas en la configuración inicial del modelo, se probó una nueva variante del preprocesamiento que consistió en eliminar las *stop words* ('stop_words='english'') y mantener únicamente unigramas ('ngram_range=(1,1)'). Con estos ajustes, se volvió a entrenar el modelo 'MultinomialNB' y se evaluó su desempeño sobre el conjunto de test.

Los resultados, tanto en la matriz como en métricas, mostraron una mejora significativa. El accuracy aumentó de 0.51 a 0.85 y el F1-macro pasó de aproximadamente 0.23 a 0.61. La nueva matriz de confusión (Figura 4) refleja una distribución mucho más balanceada: Joe Biden fue correctamente clasificado en los 24 discursos disponibles, y Donald Trump en 16 de 17. Sin embargo, el modelo continuó sin predecir correctamente los discursos de Mike Pence, cuyos valores de precision, recall y F1-score permanecen en cero.

Figura 4



Estos cambios mostraron cómo la eliminación de las stop words facilita al modelo identificar patrones de lenguaje más representativos de cada clase. En particular, se evidenció una mejora clara en la clasificación de los discursos de Donald Trump y Joe Biden, quienes ahora son correctamente reconocidos en la mayoría de los casos. Esto sugiere que, al eliminar palabras de alto peso pero bajo contenido informativo, el modelo logra captar diferencias sustanciales en el vocabulario utilizado por cada candidato. Aun así, la clase correspondiente a Mike Pence continuó sin ser correctamente identificada, lo cual se refleja en sus valores nulos de precision, recall y F1-score. Esto es algo que ya se podía prever cuando se analizó la frecuencia de discursos entre los tres candidatos y arrojó un desbalance en detrimento de Mike Pence.

Este tipo de resultado refuerza la importancia de analizar la matriz de confusión, que permite visualizar de forma directa qué clases están siendo aprendidas correctamente y cuáles no. Si bien el accuracy se incrementó de 0.51 a 0.85, ésta métrica agregada puede

seguir resultando engañosa si no se acompaña de un análisis más detallado. Incluso con un *accuracy* aparentemente alto, es posible que el modelo esté ignorando por completo algunas clases, que con la matriz de confusión y métricas como el *F1-macro* se puede detectar esa falla con claridad.

Optimización del modelo mediante validación cruzada y búsqueda de hiperparámetros

En el caso del modelo *Multinomial Naive Bayes*, se ajustaron dos hiperparámetros clave. El primero, 'alpha', regula el nivel de suavizado aplicado al estimar las probabilidades de cada palabra dentro de una clase. Este suavizado evita que una palabra nunca observada en una clase tenga probabilidad cero, lo que anularía toda la predicción. Un valor bajo de 'alpha' implica poco suavizado, haciendo que el modelo se ajuste más a los datos observados; en cambio, valores más altos estabilizan las probabilidades, lo cual puede ser útil ante vocabularios escasos o desbalanceados. El segundo hiperparámetro, 'fit_prior', determina si se tienen en cuenta las proporciones de cada clase en los datos de entrenamiento: si está activado, el modelo da más peso a las clases mayoritarias; si no, trata a todas por igual. En contextos con clases desbalanceadas, como en este caso, su activación puede reforzar el sesgo hacia las clases más frecuentes, mientras que desactivarlo permite explorar un escenario más neutral de clasificación.

En la sección anterior se evidenció que el uso de *accuracy* como métrica de evaluación podía resultar engañoso, especialmente en contextos con clases desbalanceadas. Por este motivo, se seleccionó 'f1_macro' como métrica principal para la búsqueda de hiperparámetros. Esta métrica combina dos aspectos fundamentales del desempeño de un clasificador: la *precision* (qué proporción de las predicciones fueron correctas) y el *recall* (qué proporción de los casos reales fueron detectados). El *F1-score* es el promedio armónico entre ambos, lo que permite penalizar los modelos que favorecen uno a costa del otro.

En problemas de clasificación *multiclase* como este, 'f1_macro' calcula el *F1-score* para cada clase por separado y luego realiza un promedio simple entre ellos. Esto implica que todas las clases aportan el mismo peso al resultado final, sin importar cuántos ejemplos tengan. Esta característica es especialmente relevante cuando se busca evaluar si el modelo logra tratar de forma justa también a las clases menos representadas, como ocurre con Mike Pence en este conjunto de datos.

Si bien existen otras variantes como 'f1_weighted', que pondera por la cantidad de ejemplos de cada clase, en este caso se priorizó una métrica que refleje de forma más equilibrada el desempeño del modelo sobre todas las clases por igual.

Para mejorar el rendimiento del modelo, se exploraron distintas combinaciones de hiperparámetros, se utilizó la herramienta 'GridSearchCV' de la biblioteca *scikit-learn*. Esta técnica realiza una búsqueda exhaustiva sobre una grilla predefinida de valores, evaluando cada combinación mediante validación cruzada. En este caso, se utilizó una validación cruzada de 5 *folds* (cv=5), lo que implica dividir el conjunto de entrenamiento en cinco

particiones: en cada iteración, el modelo se entrena con cuatro de ellas y se valida con la restante, rotando hasta cubrir todas. El rendimiento de cada combinación se estima como el promedio de los resultados obtenidos en los cinco *folds*.

La grilla de hiperparámetros utilizada (‘param_grid’) incluyó tanto parámetros propios del clasificador *Multinomial Naive Bayes* como opciones asociadas a la etapa de vectorización del texto. En particular, se evaluaron distintas combinaciones de ‘fit_prior’ (‘True’ y ‘False’), el uso de ‘use_idf’ (‘True’ o ‘False’), y el rango de n-gramas considerados ((1,1) y (1,2)), manteniendo en todos los casos la opción ‘stop_words='english''.

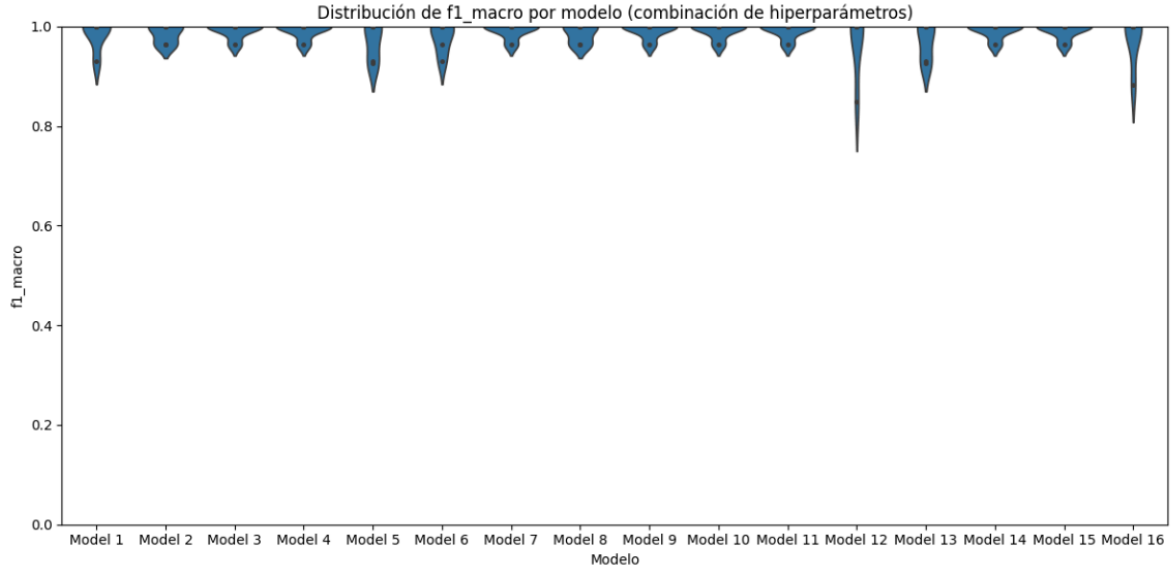
En cuanto al parámetro alpha, que controla el nivel de suavizado del clasificador, se decidió limitar la búsqueda a valores bajos (por ejemplo, 0.01) luego de una exploración preliminar más amplia. En esa etapa inicial se analizaron también valores más elevados de alpha (como 1, 0.5 etc), y se observó que su efecto sobre el rendimiento del modelo era notoriamente negativo. Las combinaciones con alpha alto tendían a producir valores significativamente menores de la métrica ‘f1_macro’, lo que motivó la exclusión de dichos valores en la grilla principal.

La evaluación de los modelos se realizó utilizando como métrica principal el promedio del ‘f1_macro’ a lo largo de los cinco *folds* de validación. Este enfoque permitió identificar la combinación de hiperparámetros con mejor rendimiento general, minimizando la influencia de particiones particulares. El mejor modelo alcanzó un valor promedio de ‘f1_macro’ de 0.9929 y corresponde a la siguiente configuración: *'clf__alpha': 0.01, 'clf__fit_prior': True, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 1), 'vect__stop_words': 'english'*.

De forma complementaria, se repitió el proceso de validación utilizando accuracy como métrica de evaluación. Si bien previamente se demostró que esta métrica puede ocultar errores importantes en contextos con clases desbalanceadas, se incluyó aquí como referencia general. El modelo seleccionado por *accuracy* resultó ser el mismo que el óptimo según ‘f1_macro’, con un valor promedio de 0.991, lo cual refuerza la solidez del modelo bajo diferentes criterios de evaluación.

Además del análisis numérico, se generó una visualización (Figura 5) que muestra la distribución del valor de *f1_macro* obtenido por cada combinación de hiperparámetros a lo largo de los cinco folds de validación cruzada. Para ello se utilizó un gráfico de violín, que permite observar tanto el valor promedio alcanzado por cada modelo como la variabilidad de sus resultados entre particiones. En este tipo de gráfico, cada “violín” representa la distribución de los cinco valores de *f1_macro* obtenidos por una combinación específica de parámetros. La forma y el ancho del violín reflejan cuán dispersos están esos valores: un violín más delgado indica mayor consistencia entre folds, mientras que uno más ancho señala mayor variabilidad. Los puntos internos indican los valores individuales alcanzados en cada partición de validación.

Figura 5



En la Figura 5, los modelos numerados del 1 al 8 corresponden a combinaciones con alfa = 0.01, mientras que los modelos del 9 al 16 utilizan alfa = 0.5. Esta separación permite visualizar con claridad el impacto del hiperparámetro sobre el rendimiento del modelo. Mientras que los modelos con alpha = 0.01 presentan valores de 'f1_macro' cercanos al máximo y distribuciones concentradas, las combinaciones con alpha = 0.5 muestran un descenso significativo en la métrica, con algunos modelos por debajo de 0.60. Este contraste evidencia que, dentro del conjunto de parámetros evaluados, éste hiperparametro es el factor más determinante en el rendimiento del modelo, mientras que los cambios en los parámetros de vectorización (como use_idf o ngram_range) generan variaciones menores.

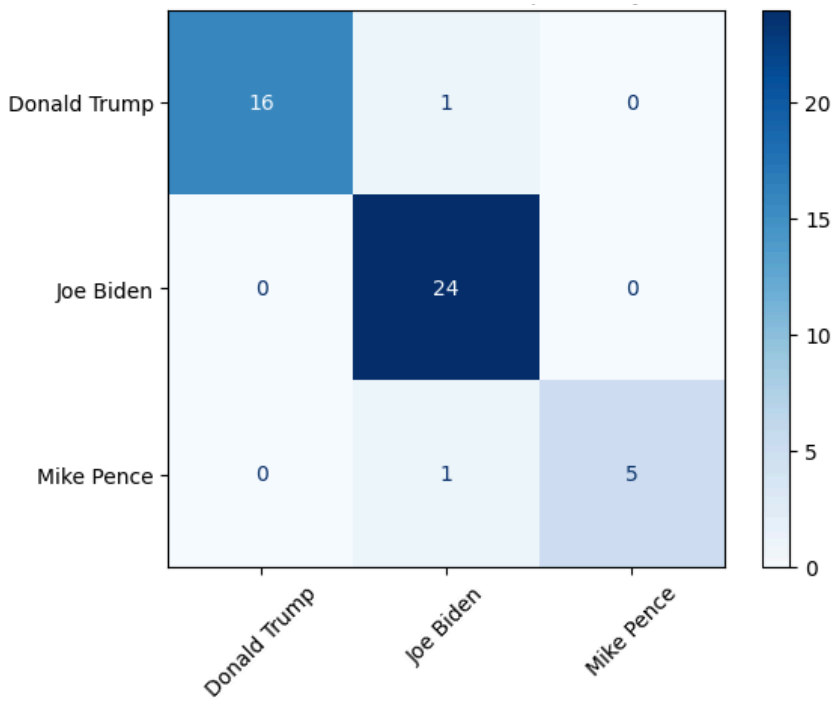
En conclusión, la optimización del modelo mostró que los cambios en los parámetros de vectorización, como el uso de use_idf o la inclusión de bigramas, no generaron diferencias significativas en el rendimiento. En cambio, el hiperparámetro alfa, responsable del nivel de suavizado en el clasificador, tuvo un impacto claro: al aumentar su valor de 0.01 a 0.5, el desempeño del modelo disminuyó notablemente. Esto confirma que, dentro de los parámetros evaluados, alfa fue el factor más determinante en la variación de resultados.

Evaluación del mejor modelo y reflexión sobre los límites del enfoque BoW/TF-IDF

Una vez identificada la mejor combinación de hiperparámetros, se volvió a entrenar el modelo *Multinomial Naive Bayes* utilizando la totalidad del conjunto de entrenamiento. El modelo final, con 'alpha=0.01', 'fit_prior=True', 'stop_words='english'' y 'ngram_range=(1,1)', fue evaluado sobre el conjunto de test.

Los resultados obtenidos fueron altamente satisfactorios: el modelo alcanzó un valor de *accuracy* de 0.96 y un 'f1_macro' de 0.95, mostrando un rendimiento elevado y equilibrado en las tres clases. Las métricas por clase confirman esta estabilidad, con valores de *precision*, *recall* y *f1-score* superiores al 0.90 para cada uno de los candidatos. La matriz de confusión (Figura 6) refleja esta misma tendencia, con muy pocos errores de clasificación.

Figura 6



A pesar de estos buenos resultados, el enfoque basado en *Bag of Words* y TF-IDF presenta limitaciones importantes. En primer lugar, no considera el orden ni la relación gramatical entre las palabras. Esto significa que expresiones como "no es bueno" y "es bueno" pueden tener representaciones muy similares, lo cual puede ser problemático si el significado del texto depende del contexto o de la forma en que se combinan las palabras.

Además, estas técnicas tratan cada término de forma aislada, sin capturar relaciones semánticas como sinónimos, negaciones o matices discursivos. Tanto BoW como TF-IDF generan representaciones estáticas: una misma palabra tendrá siempre la misma representación, independientemente del sentido que adquiera en diferentes frases o

situaciones. Esto limita su capacidad de adaptación a contextos específicos de tarea o corpus.

En este caso particular, el modelo logró un muy buen rendimiento, lo que sugiere que existían diferencias suficientes entre los discursos para que una representación superficial bastara. Sin embargo, en tareas que requieren una comprensión más profunda del lenguaje, o cuando las clases se diferencian por sutilezas discursivas o semánticas, estos enfoques podrían no ser suficientes. Modelos más avanzados, como los basados en embeddings contextuales (por ejemplo, BERT), ofrecen una mayor capacidad de representar el lenguaje con riqueza estructural y semántica, y serían más adecuados en esos escenarios.

Modelo de regresión logística

Se utilizó también el modelo de regresión logística para evaluar con validación cruzada para clasificar los discursos entre los tres candidatos elegidos con el objetivo de identificar cuál de los dos modelos de *scikit-learn* funciona mejor para predecir con el conjunto de datos que se está trabajando.

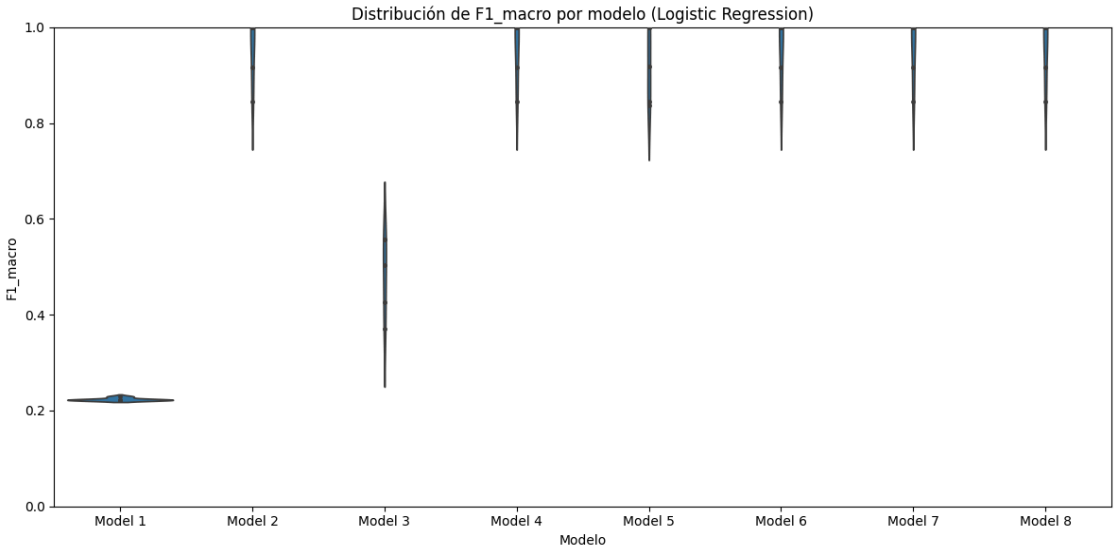
La regresión logística es otro modelo estadístico que se usa para clasificación. A pesar de su nombre, se implementa como un modelo lineal para clasificación y no como regresión. Toma inputs, en este caso los vectores TF-IDF, y calcula la probabilidad de que pertenezca a una clase.

Se asume que el objetivo (la clase a predecir) toma valores en el conjunto {0, 1} para cada dato. Como problema de optimización, se minimiza la función de costo logística más un término de regularización. Una vez ajustado el modelo, el método de predicción devuelve la probabilidad de pertenecer a la clase positiva. Este resultado numérico del modelo (una probabilidad) puede transformarse en una clasificación aplicando un umbral (por defecto 0.5).

Como algoritmos de regularización, scikit-learn incluye los siguientes ‘solvers’ para el caso multinomial: "liblinear", “lbfgs" (por defecto), "newton-cg", "newton-cholesky", "sag", "saga". Se eligió utilizar “lbfgs”, ya que es la opción segura y robusta para el caso multinomial. Por otro lado, el parámetro C controla la fuerza de la regularización del modelo. Es el parámetro que determina la relación de intercambio entre ajustar bien el entrenamiento y generalizar bien a la prueba. Con un C grande hay poca regularización y el modelo ajusta más a los datos y viceversa. Se probó con diferentes valores de C [0.01, 0.1, 1.0, 10.0]. Por último, se probaron configuraciones de ‘class_weight’: una sin ponderar cada clase en el cálculo de la función de pérdida durante el entrenamiento y otra balanceando cada clase inversamente proporcional a su frecuencia en el dataset.

Para entrenar el modelo se utilizaron las mismas features de texto utilizadas en el mejor modelo Multinomial Naive Bayes encontrado en la sección anterior, esto es, eliminando stop words en inglés y aplicando unigramas e IDF.

Figura 7



Aplicando ‘GridSearchCV’, se encontró que la mejor configuración para Regresión Logística incluía ‘C=1.0’, ‘class_weight='balanced'’ y ‘solver='liblinear'’. En la Figura 7 se ve la distribución de la métrica ‘f1_macro’ por modelo. Este mejor modelo resultante es el que aparece en sexto lugar. Evaluado sobre el conjunto de test, este modelo alcanzó un accuracy de 0.9362 y un ‘f1_macro’ de 0.91, con buen rendimiento en todas las clases. Sin embargo, al compararlo con el modelo Naive Bayes, que alcanzó un ‘f1_macro’ superior (≈ 0.993) y accuracy ≈ 0.99, se observa que, dicho modelo, resultó más efectivo para este conjunto de datos.

Esta diferencia puede atribuirse a que los discursos de los candidatos parecen tener patrones léxicos diferenciables, lo que favorece a Naive Bayes, que trabaja muy bien con features independientes y frecuencias de palabras. Regresión Logística, si bien más flexible y robusta en otros contextos, es más sensible al ruido en datos dispersos como los generados por TF-IDF. En este caso, la simplicidad estadística de Naive Bayes resultó ser una ventaja, permitiéndole generalizar mejor en esta tarea de clasificación de texto.

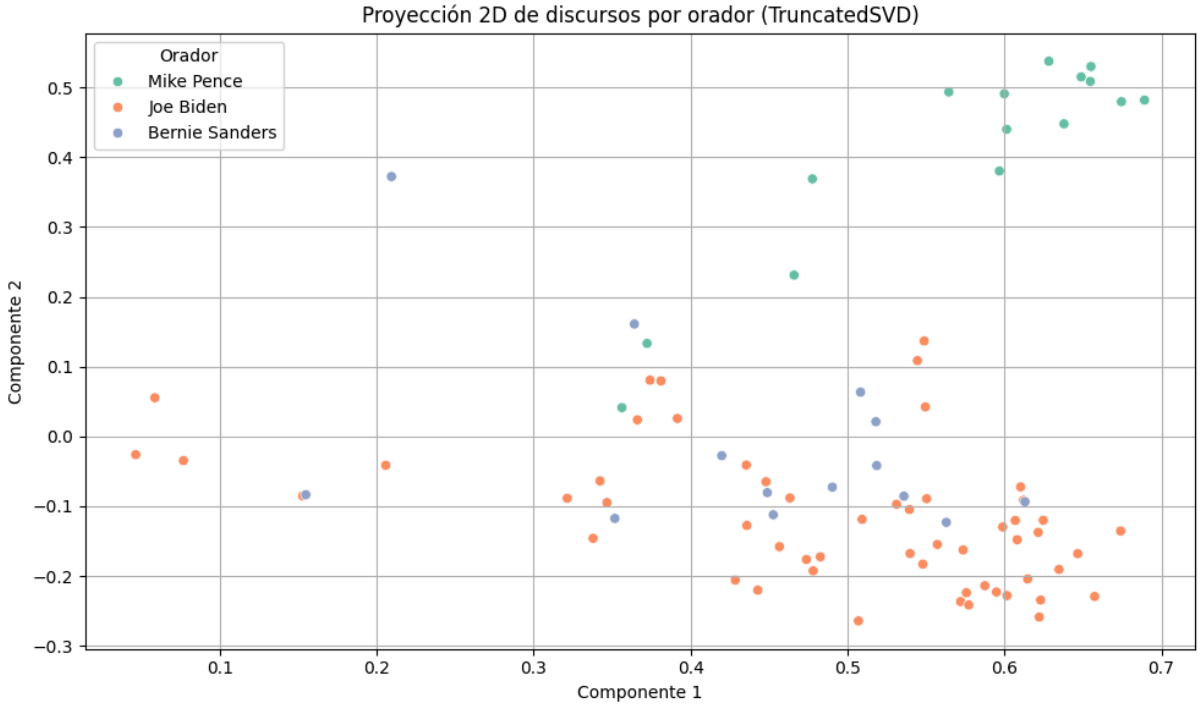
Caso de desbalance de los datos: cambio de candidato

Para analizar el impacto del desbalance de clases, se redefinió el conjunto de oradores seleccionados, utilizando a Joe Biden, Mike Pence y Bernie Sanders. Esta elección introdujo un escenario con una clase dominante (Joe Biden, con cerca del 65% de los discursos) y dos clases minoritarias (Pence y Sanders, ambas con cerca del 17-18%). Esta diferencia en la cantidad de ejemplos puede sesgar el modelo durante el aprendizaje, favoreciendo a la clase dominante.

Para explorar cómo se distribuyen las clases en el espacio de representación, se aplicó reducción de dimensión con 'TruncatedSVD' (PCA) sobre los vectores TF-IDF del conjunto de entrenamiento. La Figura 8 muestra el gráfico de dispersión en los dos componentes

principales. Se puede ver que Mike Pence aparece relativamente separado en un rincón del espacio vectorial, lo que sugiere que el modelo podría distinguirlo con facilidad si tiene suficiente representación. En cambio, los discursos de Bernie Sanders se encuentran parcialmente solapados con los de Joe Biden, lo que podría reflejar la dificultad en su correcta clasificación.

Figura 8



Al reemplazar a Donald Trump por Bernie Sanders, se alteró la naturaleza del contenido discursivo, y por ende también la distribución de clases en el espacio de representación y su balance en el conjunto de datos. El solapamiento entre Biden y Sanders puede explicarse por la afinidad ideológica o la similitud temática entre ambos candidatos, lo cual impacta directamente en el rendimiento del modelo para esas clases.

Por otro lado, al introducir a Sanders, se observa también un desbalance en la cantidad de documentos por clase. Este desequilibrio puede provocar que el modelo tienda a favorecer las clases mayoritarias (Biden) y cometa errores frecuentes al clasificar documentos de Pence por tener menos representación. Frente a este escenario, una posible solución sería aplicar técnicas de balanceo de clases sobre el conjunto de entrenamiento:

- Sobremuestreo (*oversampling*): consiste en aumentar artificialmente el número de ejemplos de la clase minoritaria, por ejemplo, replicando o generando nuevos discursos de Pence (duplicación aleatoria o técnicas sintéticas como SMOTE).

- Submuestreo (*undersampling*): reduce la cantidad de ejemplos de las clases mayoritarias para equilibrar las proporciones. Puede ser útil si hay muchas muestras de Biden, pero también implica perder información.

Cambiar uno de los candidatos en el set de clasificación modifica tanto el balance como la estructura semántica de las clases. El análisis con PCA permitió visualizar la separabilidad de los discursos, revelando que Pence es más fácilmente identificable, mientras que Sanders y Biden son más difíciles de distinguir. Esto, combinado con el desbalance de clases, justifica la necesidad de ponderación o técnicas de remuestreo para asegurar un modelo más justo y robusto. Estas pueden ser la de sobre muestreo o la de submuestreo. Ambas técnicas buscan mejorar el equilibrio del conjunto de entrenamiento y ayudar a que el modelo generalice mejor a clases minoritarias, pero deben aplicarse con cuidado para no introducir sesgos o pérdida de diversidad.

Técnica alternativa para extraer features de texto: Word Embeddings

Existen técnicas alternativas para extraer features de texto aparte de las Bag of Words o TF-IDF. Una muy utilizada en el procesamiento de texto es el uso de 'word embeddings', tales como Word2Vec, GloVe o FastText. A diferencia de Bag of Words o TF-IDF, que representan las palabras como vectores dispersos (sparse) y de alta dimensionalidad donde cada dimensión representa una palabra del vocabulario, los embeddings representan cada palabra mediante un vector denso de baja dimensión, que es aprendido a partir de un gran conjunto de documentos de texto.

Word2Vec[2] es una técnica basada en redes neuronales que aprende a representar palabras de forma tal que las que aparecen en contextos similares tengan representaciones similares. Existen dos variantes principales: CBOW (Continuous Bag of Words), que predice una palabra a partir de su contexto y skip-gram, que predice el contexto a partir de una palabra.

Por ejemplo, las palabras "rey" y "reina" terminan teniendo vectores similares, y además se pueden capturar relaciones semánticas como:

vector("rey") - vector("hombre") + vector("mujer") ≈ vector("reina")

Existen algunas diferencias que pueden esperarse en los resultados. En primer lugar, una ventaja semántica. Mientras que Bag of Words y TF-IDF solo capturan la frecuencia de aparición de palabras (sin contexto), los embeddings permiten capturar relaciones semánticas y de contexto. En segundo lugar, se espera una reducción de dimensionalidad. Los vectores de embeddings suelen tener entre 50 y 300 dimensiones, mucho menos que las miles de dimensiones de un modelo de bolsa de palabras. En tercer lugar, los modelos con embeddings suelen generalizar mejor cuando se enfrentan a palabras similares no vistas durante el entrenamiento. Por último, hay mayor reducción de ruido: palabras poco frecuentes, errores tipográficos o expresiones raras afectan menos porque no se depende tanto de la frecuencia exacta.

La gran limitación de estas técnicas es que requieren un entrenamiento más costoso o bien la utilización de modelos pre entrenados. Además, pueden perder interpretabilidad respecto a modelos basados en frecuencia o generar sobreajuste si el dataset es chico.

---

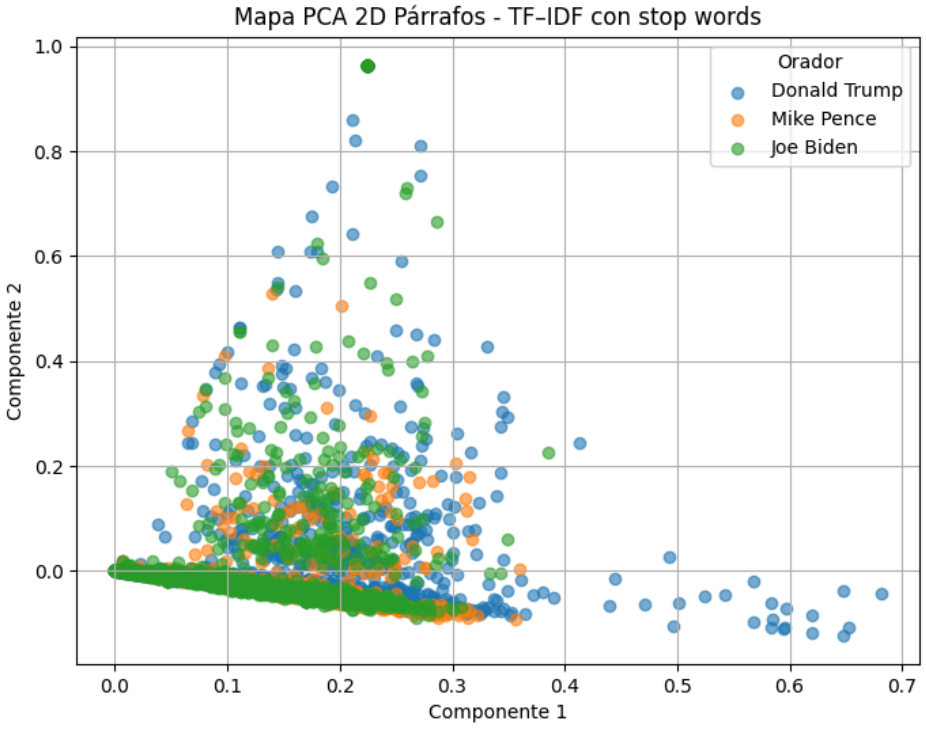
[2] https://www.tensorflow.org/text/tutorials/word2vec

En definitiva, estas técnicas utilizan herramientas que en teoría predicen mejor la mayor cantidad de las veces, especialmente cuando el vocabulario es rico, los textos son largos o variados, y hay ambigüedad semántica. Pero cuando de por sí los modelos basados en frecuencia ajustan bien, las limitaciones de estas técnicas alternativas podrían ser suficientes como para no tener que elegirlas.

Clasificación a nivel de párrafos

Con el objetivo de evaluar cómo se comportan estos modelos cuando la unidad de análisis no es el discurso si no el párrafo, se realizó la clasificación nuevamente partiendo del dataset dividido por párrafos. En este dataset con 5745 documentos (párrafos), la distribución de la clase cambia, de manera que Trump tiene el 47% de los párrafos, Biden el 42% y Pence el 11%. Con estas proporciones también se hace la partición estratificada entre entrenamiento y prueba y se aplica la representación numérica de los textos. Eliminando las stop words, solo con los unigramas y aplicando IDF, se implementa el análisis de componentes principales, cuyo mapa de representación en los dos componentes principales de muestra en la Figura 9.

Figura 9

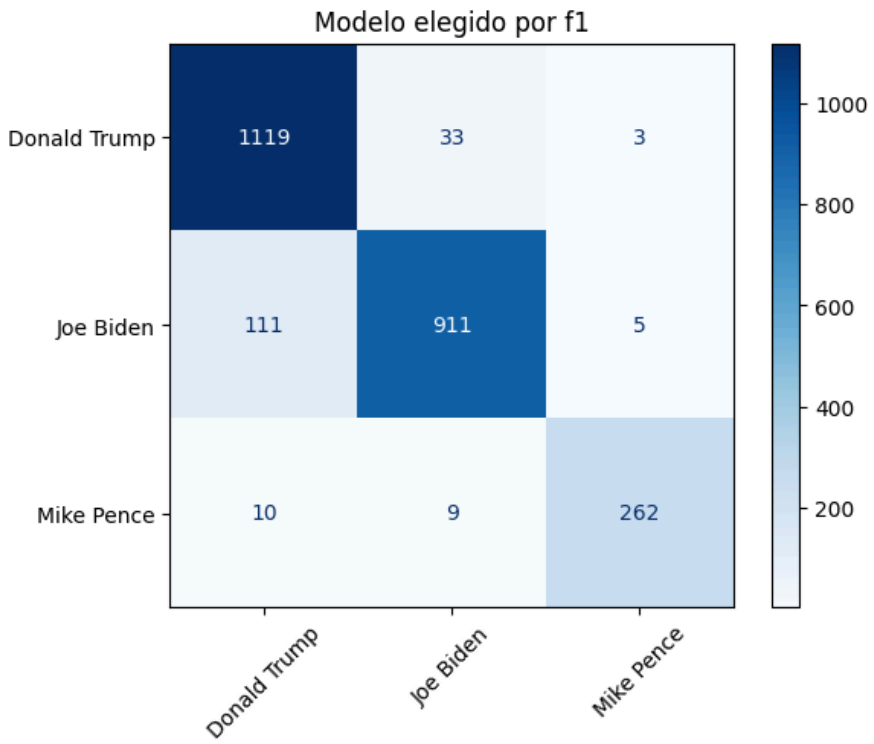


No se logran distinguir ninguna de las tres clases y, no se muestra, pero la varianza explicada acumulada por los primeros diez componentes resulta muy baja, por debajo del 10%. Esto se debe a la alta dimensionalidad, la matriz TF-IDF tiene muchas columnas (palabras únicas) y cada párrafo es un documento muy corto, con pocas palabras y mucho ruido. Esto genera una matriz muy dispersa.

Se probó de qué manera se comporta el modelo ‘MultinomialNB’, utilizando ‘GridSearchCV’ con diferentes valores de los parámetros, de la misma manera en que se hizo con la clasificación a nivel de discurso. Se comprobó que los modelos que surgen a partir de los hiperparámetros elegidos tienen mayor capacidad predictiva que el modelo elegido para el caso de los discursos aplicado a estos datos por párrafos. Con estos datos, el mejor modelo bajo el criterio de ‘accuracy’ es el que tiene los siguientes parámetros: {'clf__alpha': 0.01, 'clf__fit_prior': False, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 2)}; mientras que el mejor modelo bajo el criterio de ‘f1_score’ es el siguiente: {'clf__alpha': 0.01, 'clf__fit_prior': True, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 2)}. Varía el ‘prior’ de la distribución ‘MultinomialNB’, pero lo que importa es que en ambos casos es mejor un modelo con bigramas. Esto tiene sentido, ya que los bigramas capturan secuencias de palabras que pueden ser más características del estilo discursivo de cada orador, como frases hechas, expresiones frecuentes o construcciones sintácticas propias, lo que puede resultar clave cuando se trabaja con párrafos, donde el contexto es limitado.

En este escenario, donde los textos son breves y menos informativos por sí solos, incorporar bigramas mejora la capacidad del modelo para diferenciar entre oradores que comparten parte del vocabulario pero no necesariamente la forma de expresarse. Esto sugiere que, al trabajar con unidades pequeñas de texto como párrafos, es fundamental utilizar una representación más rica del lenguaje, como los n-gramas, para compensar la pérdida de contexto global.

Figura 10



Haciendo la prueba en el subconjunto de test con el mejor modelo, elegido por el criterio de mejor ‘f1_score’, se encuentra (Figura 10) que predice correctamente una gran proporción de los párrafos, aunque también predice erróneamente en mayor proporción que en el caso de los discursos, con un accuracy de 0.94 y f1_score de 0.94. Este modelo, si bien se

comporta bien, predice peor que el que toma por unidad a los discursos debido a que el párrafo contiene menos palabras y por ende menos información y aunque los bigramas ayudan a capturar patrones locales, no compensan completamente la pérdida de contexto que sí se tiene en un discurso entero.