Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Identificación de comunidades en intervalos de tiempo a través del lenguaje

Tesis de Licenciatura en Ciencias de la Computación

Martín Igal Browarnik

Directores: Dr. Esteban Feuerstein, Lic. Juan Manuel Ortiz de Zarate

Buenos Aires, 2021

IDENTIFICACIÓN DE COMUNIDADES EN INTERVALOS DE TIEMPO A TRAVÉS DEL LENGUAJE

La identificación de comunidades es una tarea ampliamente estudiada desde diversas disciplinas hace ya largos años; con la aparición de la digitalización de las relaciones humanas se hizo mucho más accesible analizar e identificar comunidades a escalas mucho más grandes. Las redes sociales, por ejemplo, juegan un rol sumamente preponderante en nuestras interacciones, influenciándolas directamente debido a sus algoritmos de segregación. Algunas de estas proveen APIs que permiten acceder a esta información e identificar facilmente las relaciones de los usuarios. En este trabajo desarrollamos distintas metodologías para entrenar modelos PLN (Procesamiento de lenguajes naturales) que logren identificar comunidades en redes sociales exclusivamente a través de su jerga, es decir por el lenguaje que utilizan, a lo largo del tiempo. Para esto, realizamos diversos experimentos que nos permiten determinar cuáles metodologías son mejores que otras dependiendo el contexto y cómo debemos mantener estos modelos para que sigan siendo útiles a lo largo del tiempo.

Palabras claves: Modelos de PLN, Detección de comunidades, Redes sociales.

Índice general

1. INTRODUCCIÓN

La identificación de comunidades es una tarea ampliamente estudiada desde diversas disciplinas hace ya largos años. En el famoso estudio de los 70' "An information flow model for conflict and fission in small groups" Zachary [1] analiza las comunidades que se encontraban latentes dentro de un grupo de Karate basándose únicamente en la estructura del grafo generado por sus relaciones.

Actualmente la interacción entre las personas se desarrolla en mayor parte a través de las redes sociales. Estas le permiten a los usuarios interactuar virtualmente con otras personas, compartir distintos tipos de contenidos, opiniones, etc. Si bien la utilización de las redes sociales no es un fenómeno nuevo, con la aparición del SARS-CoV-2 y el aislamiento preventivo obligatorio, casi cualquier comunicación entre personas pasó a ser a través de internet, incrementando el uso de las redes sociales y aumentando su audiencia. Hoy son un eje central en la vida de una gran parte de la sociedad y es por esta razón que también son motivo de estudio y análisis para distintas entidades, por ejemplo se utilizan para medir el rating de televisión, para publicitar productos o, como en nuestro caso, para conocer la inclinación política de las personas.

Las redes sociales también juegan un rol preponderante en nuestras interacciones influenciándolas directamente y produciendo, debido a sus algoritmos de segregación, "Filter Bubbles" [2] que agudizan la homofilia [4] (la tendencia de las personas a la atracción por sus semejantes, es decir, individuos que son similares en una o más características). En otras palabras, los algoritmos de segregación analizan la información personal del usuario (ubicación, historial de búsqueda, clicks, etc) y le presentan a este información que le pueda llegar a interesar, basándose en el historial de otros usuarios con configuraciones similares. Del mismo modo, estos algoritmos "esconden" toda información que no sea congruente con los intereses o puntos de vista de dicho usuario, generando comunidades que segregan a los distintos grupos de personas. Para cada usuario, se genera entonces un mundo de confort en el cual todo parece coincidir con sus puntos de vista, opiniones y gustos. Como consecuencia de esto, las personas de una comunidad tienden a estar convencidas de que lo que piensan es la única realidad posible, lo que las vuelve más intolerantes a opiniones diferentes y hace que discutan más fuertemente con las personas de otra comunidad [3]. Todo esto genera grietas en las relaciones que en algunos casos trascienden el ámbito de la virtualidad.

Nuestra hipótesis es que en comunidades entre las que se observan grandes polarizaciones, como en el ámbito de la política, las jergas se vuelven sumamente específicas y distinguibles dentro de cada comunidad, las cuales utilizan distintos términos o frases que son propias de ellas y no suelen ser utilizadas por otras. En estos casos, mediante algoritmos de PLN, se podrían generar modelos que permitan identificar, con una buena probabilidad, la comunidad de pertenencia de un usuario basándonos exclusivamente en la forma de escribir.

Para desarrollar estos modelos necesitamos analizar la jerga que utilizan los usuarios a lo largo del tiempo y abstraernos del vocabulario puntual utilizado durante la duración de un tópico (eventos específicos que se discuten durante un cierto tiempo). Independientemente de la mayor o menor duración de un tópico, estos son siempre temporales y, eventualmente, se diluirán dando lugar a un nuevo tópico de conversación. La jerga, por

el contrario, se mantiene más estable a lo largo del tiempo, trascendiendo la aparición de varios tópicos. A pesar de su naturaleza temporal, en algunos casos el tópico puede modificar la jerga y los usuarios de las comunidades podrían adoptar algunos de los términos introducidos por los tópicos que pasarían a formar parte de la jerga de esa comunidad.

En este trabajo, nuestro objetivo es encontrar la mejor estrategia para desarrollar modelos que permitan predecir la pertenencia de un usuario a una comunidad a lo largo del tiempo a través del texto de sus posteos, analizando la jerga propia de las comunidades de manera independiente de los tópicos.

Consideramos que en el área de la política y, más específicamente, en el contexto de las elecciones presidenciales, los cambios de tópico son algo frecuente, lo cual puede deteriorar la capacidad de predicción de los modelos generados. Por ese motivo, desarrollamos distintas estrategias de predicción buscando que la jerga sea lo más robusta posible para que los modelos desarrollados funcionen adecuadamente a lo largo del tiempo.

Para validar nuestra hipótesis, tomamos como casos de estudio las discusiones en Twitter en torno a las elecciones presidenciales del 2019 en Argentina, las cuales se desarrollaron en un contexto de una muy alta polarización entre los partidos políticos de Cambiemos y el Frente de Todos. Podemos ver en estos tweets ejemplos de la jerga utilizada por las 2 principales comunidades en donde el partido del *Frente de Todos* utiliza palabras como "Macrisis" 1.1 y "MacriGato" 1.2 mientras que el partido de *Cambiemos* utiliza palabras como "Kretina" 1.3 y "Alberso" 1.4.

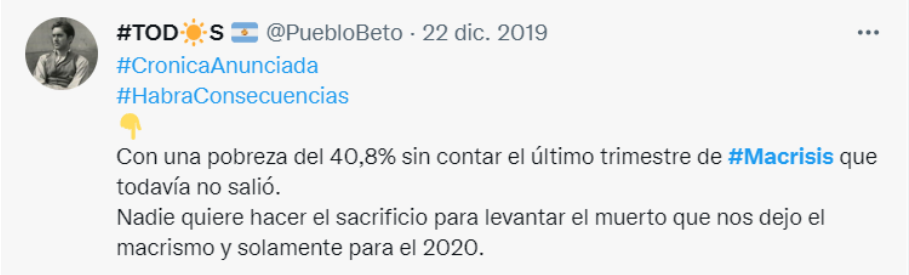

Fig. 1.1: La comunidad del frente de todos utiliza palabras como "Macrisis"

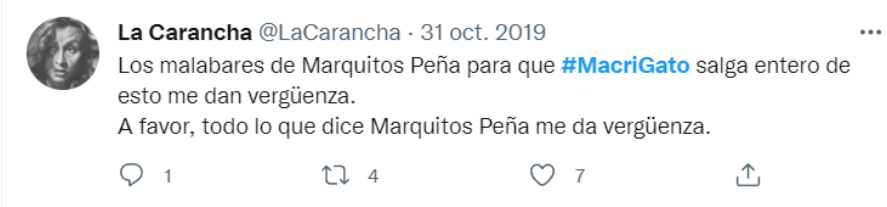

Fig. 1.2: La comunidad del frente de todos utiliza palabras como "MacriGato"

Como mencionamos anteriormente, a lo largo del tiempo ocurren cambios de tópico que pueden alterar el lenguaje por tiempos prolongados o de manera permanente. Entrenamos diversos modelos de PLN mediante Fasttext [10] capaces de predecir la comunidad de pertenencia de los usuarios, utilizando únicamente los tweets publicados entre los meses de agosto y diciembre del 2019, es decir, desde las PASO[1] hasta la asunción de Alberto

[1] Elecciones Primarias, Abiertas, Simultaneas y Obligatorias en Argentina

Fig. 1.3: La comunidad de Cambiemos utiliza palabras como “Kretina”

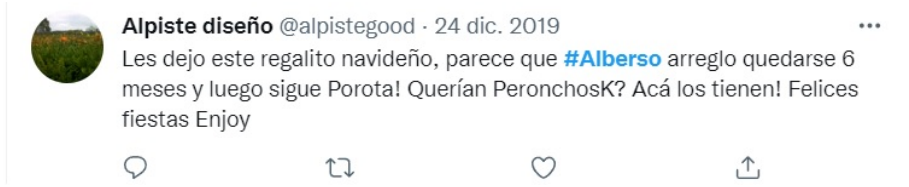

Fig. 1.4: La comunidad de Cambiemos utiliza palabras como “Alberso”

Fernandez como Presidente en diciembre. Para expandir lo más posible el análisis, buscamos que el método sea agnóstico del lenguaje y analizamos también la posición de la población brasilera en torno al Presidente Jair Bolsonaro en el contexto de la Pandemia del SARS-CoV-2 durante los meses de junio-julio del 2020. Utilizamos Walktrap [11] y Louvain [12] como ground-truth a la hora de medir el accuracy de los modelos desarrollados tanto para el caso de Argentina como para el caso de Brasil. Si bien es cierto que ninguno de estos métodos puede ser considerado un ground-truth real, para obtenerlo necesitaríamos, para cada dataset de entrenamiento, preguntarle a cada usuario con qué comunidad se siente identificado, lo cual nos resulta imposible para este trabajo. Por este motivo, para aproximarnos a la verdad, consideramos el resultado de estos métodos como ground truth.

Este trabajo lo dividimos en 2 etapas. En la primera, nuestro objetivo es encontrar la mejor manera para generar buenos modelos predictivos en un intervalo de tiempo determinado, y descubrimos que la mejor forma de entrenar estos modelos es utilizar los datos de 2 días no consecutivos de la semana (por ejemplo, martes y jueves). En la segunda etapa buscamos la mejor estrategia para conseguir que esos modelos se mantengan efectivos a lo largo del tiempo, para lo cual definimos una métrica que nos permite detectar cuándo es necesario reentrenar los modelos. Con esta métrica pudimos obtener un accuracy superior a 0.77.

En el capitulo 2 desarrollaremos las metodologías utilizadas para realizar los experimentos y detallaremos las dos etapas que mencionamos en el párrafo anterior. Ahondaremos también en cada una de esas etapas y en los procesos que se siguieron en cada una de ellas. En el capítulo 3 mostraremos los resultados obtenidos tanto para el entrenamiento como para el reentrenamiento de los modelos y detallando lo que observamos en cada una de las estrategias que se probaron. Finalmente, el capítulo 4 expone los resultados y las conclusiones de todo lo realizado. Observamos que, si bien los resultados obtenidos son positivos, aún no hay un método cuya precisión de predicción sea definitiva, con lo cual este trabajo deja abierta la posibilidad de seguir perfeccionando la métrica definida para el reentrenamiento y así seguir trabajando en una mejor estrategia para la predicción de comunidades.

1.1. Trabajos previos

Las comunidades en las redes sociales han sido objeto de estudio de diversos trabajos y se las ha analizado desde diferentes perspectivas. Los estudios que mencionamos a continuación tratan la interacción entre los usuarios en distintas redes sociales y los efectos que estas interacciones pueden tener. Esto permite analizar temas como la polarización o la jerga, entre otros, todos ellos de interés para nuestro estudio y para el análisis que desarrollaremos más adelante. Dividimos esta sección en 3 categorías centrales que nos marcan los principios para realizar este trabajo:

- La homofilia en las redes sociales, es decir, la tendencia de las personas a la atracción por sus semejantes, la cual nos permite establecer comunidades en relación a determinado eje (Sec. 1.1.1).
- La polarización en las redes sociales, lo que nos enseña que las redes sociales agravan la relación entre las distintas comunidades al generar filtros burbuja y al aumentar la diferencia de opiniones entre una comunidad y otra permitiendo distinguirlas más fácilmente (Sec. 1.1.2).
- El análisis de la jerga para identificar a las comunidades en las redes sociales, eje central de este trabajo en el que observamos que las personas dentro de cada comunidad utilizan una jerga similar que permite distinguirlas de otras y fomenta nuestra hipótesis de que esta jerga puede utilizarse para identificar a las comunidades a lo largo del tiempo (Sec. 1.1.3).

1.1.1. Homofilia en las redes sociales

Aruguete et al. [16] estudian que los usuarios de Twitter comparten exclusivamente los eventos políticos que son congruentes con sus puntos de vista. En este estudio, muestran que las redes sociales crean burbujas con los usuarios que se interesan por los mismos temas y, de esta manera, los exponen a información relacionada con sus mismos puntos de vista. Al compartir eventos, aumenta la frecuencia con la que usuarios de una misma burbuja reciben posteos del mismo tema. Como consecuencia, los tópicos de los posteos que esos usuarios realizan se vuelven "tendencia", alcanzando nuevas audiencias dentro de la misma comunidad.

Bessi et al. [17] analizan el consumo de la información de los usuarios de Facebook en Italia, haciendo foco sobre todo en las teorías conspirativas y en cómo los usuarios que creen en una teoría conspirativa son más propensos a creer en otras y a confiar mucho más en la información brindada dentro de una comunidad de una teoría conspirativa de la que participan. Para esto, crean un grafo de interacciones de los usuarios utilizando los posts de los mismos, siendo un "me gusta" un feedback positivo, el "compartir", la intención de ampliar la visibilidad del post y un comentario, la intención de crear un debate, aunque este último puede tener un refuerzo positivo o negativo respecto al posteo.

Estos dos últimos trabajos refuerzan el concepto de homofilia en las redes sociales, cómo los usuarios se suelen mover dentro de su mismo círculo generando comunidades más cerradas, interactuando mayormente entre usuarios que comparten los mismos intereses. En el trabajo de Bessi, utilizan la red social Facebook para poder crear los gráfos de interacciones, mientras que en nuestro trabajo utilizamos Twitter, red social con la cual podemos hacer una correlación de las interacciones, siendo los "me gusta", "compartir" y

"comentar" de Faceebok los "me gusta", "retweet" y "responder" de Twitter. Utilizando esta idea, asumimos que los retweets entre los usuarios son un apoyo a sus puntos de vista, permitiendo relacionarlos dentro de la misma comunidad.

1.1.2. Polarización en las redes sociales

Juan Manuel Ortiz de Zarate et al. [6] analizan las discusiones en las redes sociales con el objetivo de encontrar una buena forma de cuantificar el nivel de controversia de estas a partir del vocabulario. Explican que detectar estas discusiones puede ayudar a aplicar estrategias que puedan brindarle a los usuarios lecturas más saludables, evitando información falsa y la generación de más discusiones.

Para esto analizan la controversia en seis idiomas distintos utilizando la red social Twitter como fuente de datos. Utilizan el algortimo de Louvain [12] para la detección de las principales comunidades que forman parte de las discusiones y, finalmente, entrenan los modelos con FastText [10] y BERT [21] para comparar la performance. Entre otras cosas, muestran que las discusiones en Argentina y en Brasil están altamente polarizadas. Esto lo utilizamos como base de nuestro trabajo para analizar la jerga utilizada en estos dos países.

Federico Albanese et al. [15] hablan sobre cómo las comunidades van cambiando a lo largo del tiempo, ya que las personas cambian su opinión y pueden pasar de una comunidad a otra. Para eso, estudian los cambios de las relaciones de los usuarios en Twitter utilizando técnicas de PLN. Presentan un framework de machine learning para clasificar usuarios de redes sociales como "shifting users", es decir, usuarios que pueden llegar a cambiar de opinión a lo largo del tiempo basados en las propiedades topológicas de un grafo y el texto de las discusiones de Twitter. Si bien este trabajo está más enfocado en los usuarios que cambian de opinión a lo largo del tiempo que en la identificación de las comunidades en sí, los resultados demuestran que estos son muy pocos, lo que lleva a la conclusión de que las comunidades son estables y tiene sentido analizar su jerga para poder identificarlas a lo largo del tiempo

1.1.3. Análisis de la jerga para identificar a las comunidades en las redes sociales

Giorgia Ramponi et al. [8] en su trabajo analizan las interacciones de los usuarios en las redes sociales y cómo estas suelen ocurrir dentro de las mismas comunidades. Muestran que el vocabulario utilizado entre los usuarios de cada una de ellas es una característica muy distintiva de la comunidad y demuestran que analizando la sintaxis y la semántica del vocabulario utilizado en los tweets se pueden obtener buenos clasificadores y predictores dentro de una comunidad especifica. Esto resulta muy efectivo para identificar a las comunidades. También explica que uno de los campos más interesantes para aplicar este tipo de métodos es el de la política, ya que esta es de las más influenciadas en las redes sociales debido a que muchísimos políticos transmiten sus comentarios a través de Twitter.

Trang Tran et al. [7] cuentan que las comunidades están compuestas por personas que comparten intereses similares. Esto se refiere tanto a las comunidades online, es decir, aquellas que se crean a partir de una plataforma digital, como a las offline que consisten en grupos de personas que comparten los mismos intereses y se juntan físicamente, como puede ser el caso de un club de lectura. A diferencia de lo que ocurre con estas últimas

comunidades, en las versiones online existen datos concretos que reflejan la interacción entre sus miembros.

Los autores analizan la red social Reddit, la cual cuenta con miles de comunidades (subreddits), cada una con un tópico específico en el que los comentarios pueden ser votados como positivos o negativos. Para su análisis, utilizan modelos híbridos de n-grams y modelos de Asignación Latente de Dirichlet [20] para representar las jergas y los tópicos, y así demostrar que existe una jerga dentro de una comunidad que va más allá de los tópicos existentes dentro de las redes y que esta jerga es incluso más efectiva para identificar a las comunidades. Esto refuerza nuestra intención de evitar los tópicos a la hora de realizar el entrenamiento para poder obtener un predictor que funcione mejor a lo largo del tiempo.

Finalmente, Juan Manuel Ortiz de Zarate et al. [5], demuestran que mediante modelos de PLN se pueden detectar distintas comunidades en redes sociales exclusivamente a través de su jerga con una efectividad del 80 %. Este método fue aplicado a un momento específico y no a lo largo del tiempo. Como dijimos anteriormente, en este trabajo pretendemos extender lo desarrollado allí para que funcione no solo en “fotos” de un momento dado.

2. METODOLOGÍA

Basándonos en el método desarrollado en [5], dividimos el proceso en 2 etapas. En la primera etapa (Fig 2.1), que se desarrolla en la Sección 2.1 nuestro objetivo es encontrar la mejor manera para generar buenos modelos predictivos en un intervalo de tiempo determinado y en la segunda (Fig 2.2), buscamos la mejor estrategia para mantener a esos modelos efectivos a lo largo del tiempo.
A su vez, la primera etapa se divide en las siguientes sub etapas: Primero descargamos todos los tweets que utilizaron determinados keywords en las fechas que estudiamos segmentándolos por día. Al separar los datos de esta manera, obtenemos una unidad de tiempo que es lo suficientemente pequeña para identificar la aparición de un nuevo tópico y lo suficientemente grande como para analizar la jerga de nuestro predictor. Una vez que tenemos los datasets, generamos los grafos de interacción entre los usuarios para cada uno de ellos y utilizamos el algoritmo de Walktrap [11] para identificar las principales comunidades. Luego utilizamos los grafos para generar los modelos que vamos a utilizar para predecir la pertenencia de los usuarios a las 2 principales comunidades. Realizamos las predicciones con los modelos y analizamos los resultados.

La segunda etapa 2.2 también se divide en distintas sub etapas: En la primera seleccionamos un criterio para decidir en que momento se debe realizar un reentrenamiento. En base al criterio elegido, se vuelven a generar grafos de interacciones incorporando nuevos datos para el reentrenamiento. Luego utilizamos los grafos para generar los nuevos modelos y realizamos las predicciones sobre datos nuevos utilizando estos modelos. Repetimos este proceso cada vez que los resultados entren en el criterio seleccionado inicialmente.

2.1. Etapa 1

2.1.1. Obtención de datos

A fin de obtener los datos que vamos a utilizar para generar nuestros modelos, utilizamos la red social Twitter como fuente, dado que en ella, los usuarios expresan libremente sus opiniones y pueden interactuar con otros a través de retweets. Además, Twitter provee distintas APIs[1] para descargar libremente esos tweets. Elegimos como criterio de descarga de los tweets los *keywords* "*Macri*" y "*Bolsonaro*", ya que son importantes figuras políticas y ejes de discusión.

Los *keywords* son un conjunto de palabras relacionadas con un tema que utilizan los usuarios de manera inconsciente al escribir. Los *hashtags* por su parte, los escriben los usuarios con la intención de destacar determinados tópicos, por ejemplo #NoVuelvenMas o #Macrisis. Entonces, si hubiésemos elegido los *hashtags* como criterio de búsqueda no hubiéramos obtenido los tweets que no fueron etiquetados manualmente, perdiéndonos gran parte de la discusiones en la red. Además, dado que los *keywords* no están relacionados a un tópico especifico, es posible identificar mejor a las comunidades por sobre el contexto.

De esta manera, descargamos los tweets en formato de dataframes. Los dataframes son estructuras que permiten almacenar tablas de datos, en las que cada columna puede

[1] https://developer.twitter.com/en/docs/twitter-api

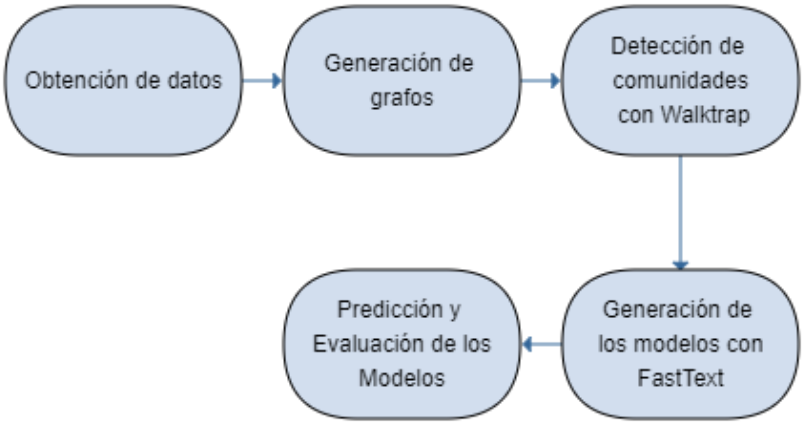

Fig. 2.1: Flujo de pasos para el entrenamiento de los modelos durante la etapa 1 para encontrar la mejor estrategia de entrenar modelos.

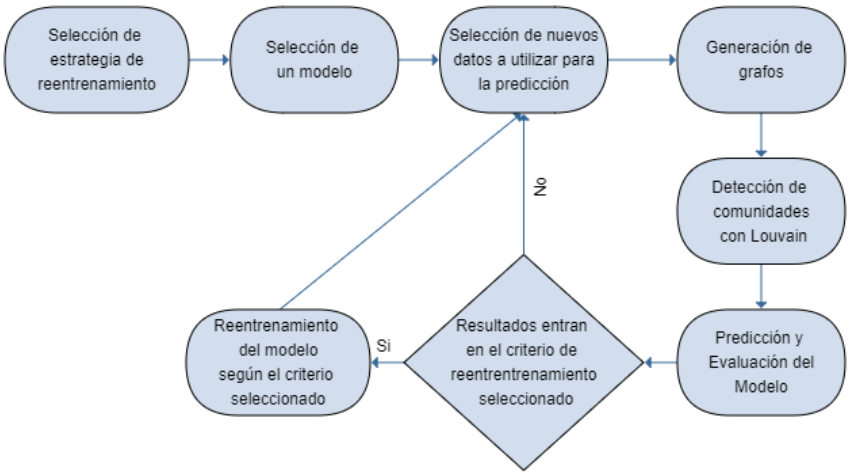

Fig. 2.2: Flujo de pasos para el reentrenamiento diario de los modelos, se repite el proceso hasta que no se encuentren datos nuevos para predecir

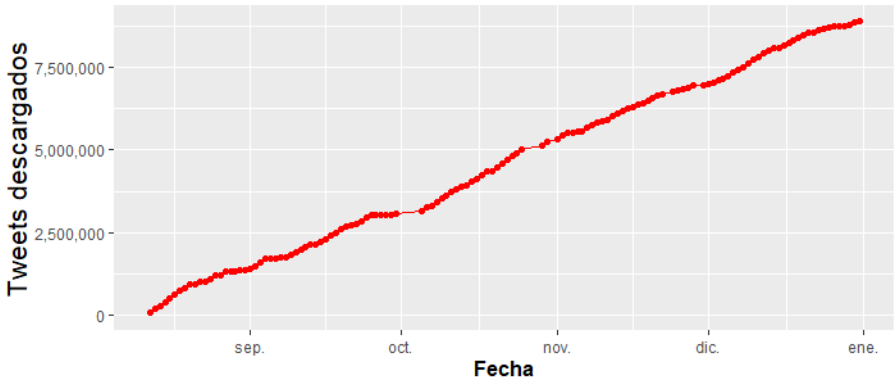

Fig. 2.3: Cantidad de tweets descargados a lo largo del tiempo para el dataset de las PASO

contener un tipo de dato distinto. En la tabla 2.1 podemos ver algunas de las columnas que contienen los datasets obtenidos.

Parámetro	**Definición**
screen_name	Usuario que realizó el tweet
text	Texto del tweet
reply_to_screen_name	Usuario al que le contestó
is_retweet	Si es un retweet o no
retweet_count	Cantidad de retweets que tiene
hashtags	Los hashtags asociados
mentions_screen_name	Menciones a otros usuarios
retweet_text	Texto del retweet

Tab. 2.1: Parámetros obtenidos de la librería de twitter

Una vez descargados los tweets, utilizamos la misma metodología que se aplicó en el trabajo [5] para normalizar los datos, pasando su codificación a ASCII, convirtiendo los emojis a texto con la librería text_clean de R[2], eliminando links, caracteres especiales y textos duplicados.

Para este trabajo descargamos entonces, 2 datasets distintos. El primero, del 12 de agosto del 2019, día posterior a las PASO, al 31 de diciembre del mismo año utilizando como criterio el uso de la palabra *Macri*, dado que como Presidente en este período, es uno de los ejes más importantes de la discusión electoral. Aproximadamente se descargan entre 50.000 y 80.000 tweets por día, teniendo un total aproximado de 9.5 millones de tweets como se ve en la figura 2.3.

En el segundo dataset, se descargaron los tweets de la población brasilera desde el 1 de junio hasta el 30 de julio del 2020 para poder analizar la polarización de la población en torno a *Bolsonaro* en el contexto de la pandemia del SARS-CoV-2. En este caso, se descargaron aproximadamente unos 100.000 tweets por día obteniendo un total aproximado de 6 millones de tweets, como se ve en la figura 2.4.

[2] https://www.rdocumentation.org/packages/textclean/versions/0.9.3/topics/

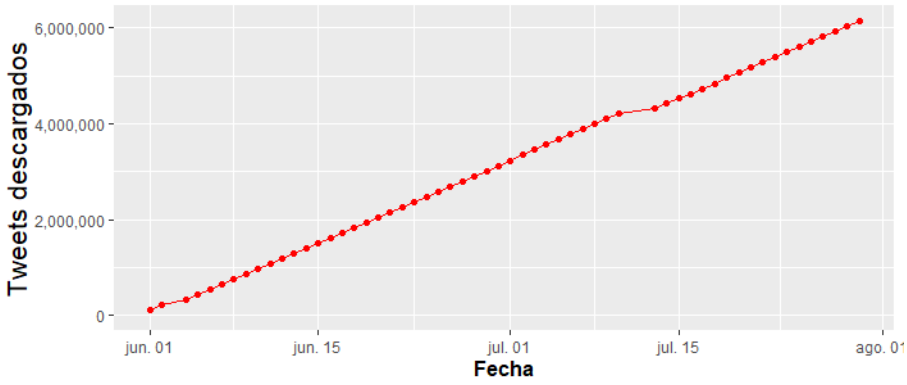

Fig. 2.4: Cantidad de tweets descargados de junio a agosto en brasil 2020 para el dataset de Bolsonaro

2.1.2. Generación del grafo

Para generar el grafo de interacciones entre los usuarios y poder posteriormente identificar las principales comunidades, consideramos a los usuarios como nodos y a los retweets como las aristas dirigidas que apuntan de un usuario a otro. Los retweets representan un apoyo al punto de vista del usuario que escribió el tweet original.

En este trabajo generamos un grafo de interacciones por día para realizar los experimentos, utilizando *Gephi*[3] con la configuración de ForceAtlas2 [18] para visualizar la interacción de los usuarios e ilustrar la polarización entre ambas comunidades. Esta configuración es particularmente buena para identificar las interacciones de las comunidades [19]. En ForceAtlas2, los usuarios que interactúan se atraen entre si formando una especie de burbuja que representa una comunidad. De la misma manera, aquellos que no lo hacen, se repelen distanciándose de los usuarios que no se relacionan dentro de esa comunidad. A modo de ejemplo, en el grafo de la figura 2.5 se puede observar la polarización entre las 2 principales comunidades en los dias del 15 y 17 de julio del 2019, cada nodo representa a un usuario, el tamaño de los nodos representa el grado de entrada del nodo, es decir, crece en función de los retweets. Utilizando a Walktrap para identificar a las comunidades, coloreamos con el color celeste a la comunidad del *Frente de Todos* y con el amarillo al partido de *Cambiemos*. Se puede ver cómo Anibal Fernandez (@FernandezAnibal) y Mariano Recalde (@marianorecalde) pertenecen a la comunidad del *Frente de Todos* mientras que Fernando Iglesias (@FerIglesias), Mario Raúl Negri (@marioraulnegri) e Ignacio Montes de Oca (@nachomdeo) pertenecen a *Cambiemos* y cómo las interacciones de los usuarios se dan, en su mayoría, entre miembros de la misma comunidad.

De la misma manera, ser generó un grafo, como se puede ver en la figura 2.6, para analizar la polarización en Brasil, en este caso, el color verde hace referencia al partido oficialista, mientras que el color rojo hace referencia al *Partido de los trabajadores*, se puede ver a Lula Da silva (@LulaOficial) en la comunidad del PT y en oposición, a Eduardo Bolsonaro (@BolsonaroSP) hijo de Jair Bolsonaro y diputado oficialista.

`replace_emoji`

[3] https://gephi.org/

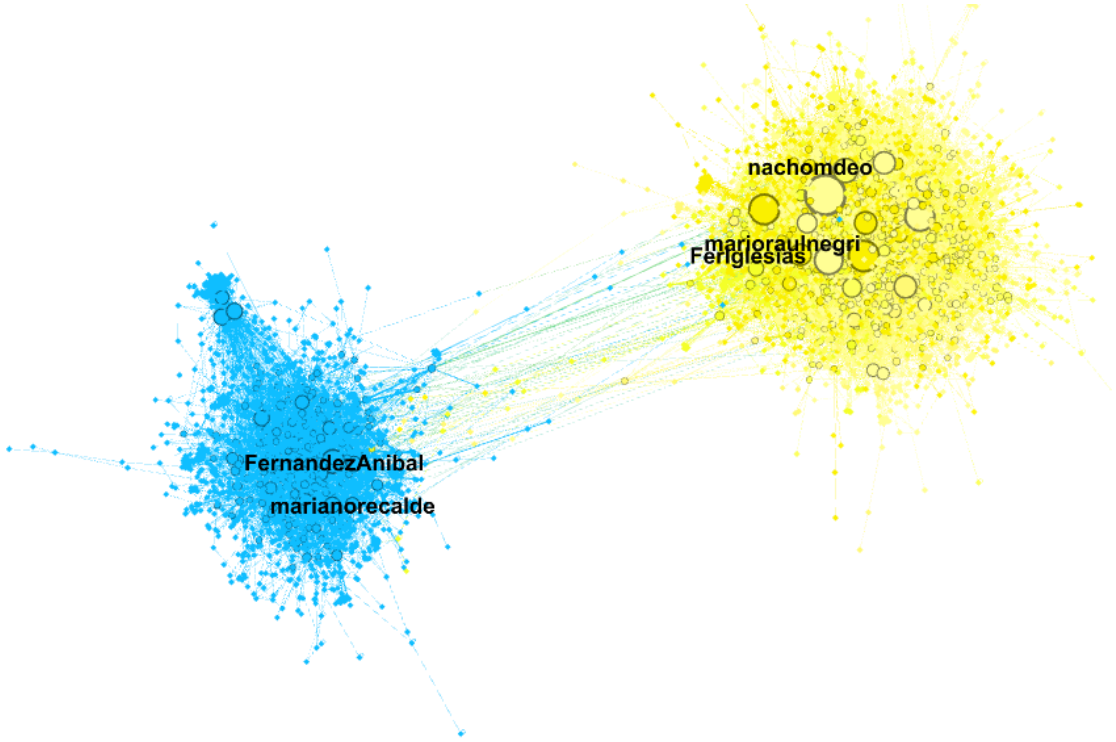

Fig. 2.5: Grafo de retweets del día 13 de octubre del 2019 visualizado a través de ForceAtlas2 y coloreado mediante Walktrap

2.1.3. Detección de comunidades con Walktrap

La segunda etapa consiste en utilizar el algoritmo de detección de comunidades de Walktrap [11] en los grafos para identificar a las principales comunidades y luego etiquetarlas según su pertenencia dentro de cada comunidad.
El algoritmo de Walktrap trabaja en grafos dirigidos, realiza caminos aleatorios (random walks) para calcular la distancia entre los nodos, los cuales son asignados a distintas comunidades según la distancia entre ellos. Cada nodo pertenece a una única comunidad. Este enfoque se basa en la intuición de que los caminos aleatorios tienden a quedar atrapados en subgrupos densos correspondientes a las comunidades.

Para poder clasificar los resultados obtenidos por estos algoritmos, es decir, asignarle una semántica a cada cluster identificado, previamente seleccionamos manualmente un conjunto de usuarios de cada comunidad identificando los usuarios que más fueron retweeteados y los usuarios con mayor cantidad de seguidores. Utilizamos este criterio de selección de usuarios dado que estos son muy influyentes. En algunos casos puede tratarse de gente famosa o de renombre y referentes dentro de cada comunidad. Como dijimos anteriormente, un retweet lo consideramos un apoyo a la opinión del usuario.

A modo de ilustración, analizamos, entre muchos otros, los grafos de polarización, como los de la figura 2.5. En este grafo podemos ver que en la comunidad celeste se encuentran nombres de usuario como los de *FernandezAnibal*, Secretario General de la presidencia, Jefe de Gabinete durante la presidencia de Cristina Fernandez de Kirchner en el 2015 y actual

Fig. 2.6: Grafo de retweets de los días 16 y 17 de junio del 2020 en Brasil visualizado a través de ForceAtlas2 y coloreado mediante Walktrap

Ministro de Seguridad de la Argentina [4], y *marianorecalde*, candidato a Jefe de Gobierno de la Ciudad de Buenos Aires en el 2015 por el Frente para la Victoria, Legislador de la Ciudad de Buenos Aires en representación de Unidad Ciudadana entre el 2017 y 2019 y actual Senador [5]. Por lo tanto, podemos inferir que la comunidad celeste apoya al partido del *Frente de Todos*. Por otra parte, en la comunidad amarilla figuran *marioraulnegri*, actual Diputado Nacional de la provincia de Córdoba, Presidente del interbloque parlamentario de Juntos por el Cambio [6], *FerIglesias*, Diputado de la Ciudad Autónoma de Buenos Aires por el partido de Cambiemos [7], y *nachomdeo*, periodista y escritor, antiperonista y opositor del gobierno de Cristina Fernandez de Kirchner [8]. De esta manera, podemos clasificar a la comunidad amarilla como la que apoya al partido de *Cambiemos*. Este análisis nos permite clasificar las comunidades obtenidas con Walktrap.

2.1.4. Generación de los modelos con FastText

Para generar los modelos, seleccionamos los días a partir de los cuales queremos entrenarlos y con el texto de los tweets de esos días, ejecutamos FastText [10], en modo supervisado utilizando los hiperparametros definidos en la tabla 2.2

Hiperparámetro	**Configuración usada**	**Definición**
dim	300	Tamaño de los vectores de las palabras
wordNGrams	2	Longitud máxima de la palabra en n-grams
ws	5	Tamaño de ventana de contexto, es decir, cantidad de palabras para adelante y para atras que mira posicionandose en una palabra.
epoch	20	Cantidad de veces que se ve cada muestra

Tab. 2.2: Hiperparámetros utilizados para ejecutar FastText

Elegimos Fasttext, una herramienta liberada por el equipo de Facebook Research, tanto para la generación de modelos como para la predicción de datos, la cual fue implementada de manera muy eficiente, lo que hace que funcione muy rápido. Esta herramienta es una mejora de Word2Vec [9] la cual, como lo indica su nombre, transforma las palabras en vectores con el objetivo de buscar que las palabras similares se encuentren en el mismo sub-espacio vectorial. Para lograr eso, entrena su red observando una cierta cantidad de palabras adelante y atrás de cada una (dependiendo del tamaño de ventana ingresado)

[4] https://www.argentina.gob.ar/seguridad
[5] https://www.senado.gob.ar/senadores/senador/498
[6] https://www.diputados.gov.ar/diputados/mnegri
[7] https://www.diputados.gov.ar/diputados/faiglesias
[8] https://www.penguinlibros.com/es/1692-ignacio-montes-de-oca

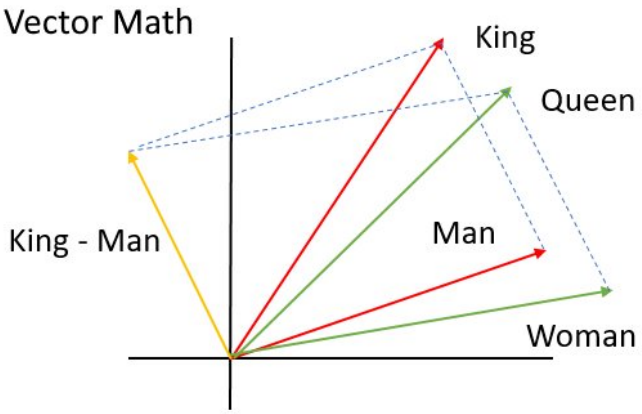

Fig. 2.7: Ejemplo de operaciones entre vectores de palabras

y en base a eso, calcula la probabilidad de que una palabra esté cerca de la otra en el futuro. Un ejemplo muy popular sobre las operaciones en vectores de palabras es el caso de realizar la operación *Rey - Hombre + Mujer = Reina*, representado en la figura 2.7, la cual describe una relación de género entre los vectores.

El problema que tiene esta herramienta es que no puede resolver sobre palabras que no se encontraban previamente en su base de entrenamiento. Para solucionar este problema, Fasttext introduce los n-grams por caracteres, mejora que permite dividir la palabra en n pequeños vectores. De esta manera, existen muchas nuevas combinaciones de las palabras permitiendo reconocer palabras que originalmente no existían y resolver también palabras parecidas o mal escritas, obteniendo una mejor interpretación del texto.

Para mejorar los resultados, utilizamos también *pretrained vectors* en español para los datasets de las elecciones en Argentina y en portugués para los datasets de Brasil, provistos por FastText[9], los cuales son modelos ya entrenados con el vocabulario de cada idioma que contienen información de la lengua utilizada en los tweets para mejorar la eficacia de nuestro modelo, creemos que esto resulta muy útil para modelos entrenados con pocos datos porque nos permite tener una mejor interpretación del lenguaje.

2.1.5. Predicción y evaluación con los modelos

Utilizamos los modelos entrenados para predecir la pertenencia de los usuarios a las principales comunidades en un nuevo conjunto de datos, luego los clasificamos con Walktrap, como ground-truth, y calculamos la métrica de ROC [13] para verificar la efectividad de estos modelos. Decidimos utilizar el área bajo la curva AUC de ROC para medir el accuracy/precisión de los modelos y poder seleccionar los buenos modelos y descartar los no tan buenos. En este trabajo consideramos una buena predicción cuando el AUC es superior a 0.7. Elegimos este valor luego de realizar distintos experimentos para los datasets de las elecciones del 2019 en Argentina. Con estos experimentos detectamos que, si utilizábamos un umbral más alto, necesitábamos realizar un entrenamiento cada 2 días o, en algunos casos, todos los días. Esto resultaba perjudicial para el modelo ya que, de ese modo, podíamos estar entrenando sobre un tópico y no tanto sobre la jerga.

La curva de ROC presenta la sensibilidad o verdaderos positivos (vp) en función de la

[9] https://fasttext.cc/docs/en/crawl-vectors.html

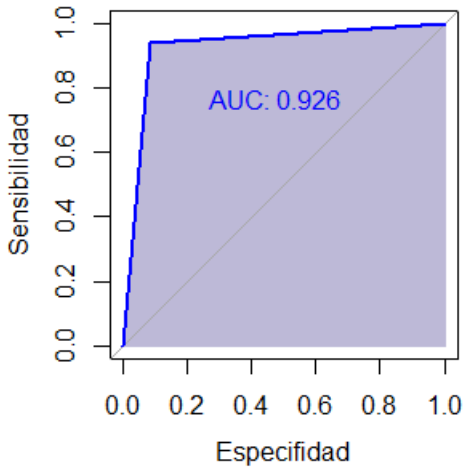

Fig. 2.8: Ejemplo de una curva de ROC y su AUC

especificidad o falsos positivos (*fp*) para distintos puntos de corte. Se entiende por *vp* al resultado positivo de un experimento que efectivamente era positivo. Mientras que el *fp* hace referencia al resultado positivo que se obtiene en un experimento pero que debería haber resultado negativo. Por ejemplo, si estamos midiendo la temperatura del agua en una olla y queremos ver si es mayor a cierto valor, no vamos a obtener el mismo resultado si medimos la temperatura cerca de la llama que si la medimos sobre la superficie, obteniendo probablemente un falso positivo en la medición. Utilizamos nuestro ground truth para identificar nuestros *vp* y nuestros *fp*, nos fijamos qué tasa de *vp* y *fp* obtenemos y la marcamos como un punto de la curva ROC. Si la predicción fuera perfecta, es decir, sin solapamiento, la curva pasaría por el punto (0,1) dejando por debajo de ella el cuadrante de ejes positivos y generando así un área igual a 1 bajo ella. Si la predicción fuera inútil (ambas tasas coincidieran y la sensibilidad fuese igual a la proporción de falsos positivos), la curva sería la diagonal de (0,0) a (1,1). En este ejemplo, que se puede ver en la figura 2.8, realizamos el cálculo del AUC entrenando un modelo con los días 6 y 7 de octubre del 2019 y realizando una predicción sobre el día 8 de octubre del mismo año obteniendo un AUC de 0.926.

2.2. Etapa 2

2.2.1. Selección de una estrategia de reentrenamiento de los modelos

Al pasar el tiempo, la jerga se va modificando producto de los tópicos que surgen. Es por eso que, para que los modelos nos sigan resultando útiles para clasificar a los usuarios dentro de ciertas comunidades, es necesario un reentrenamiento de los mismos. Para esto definimos distintas estrategias en base al paso del tiempo y a la experiencia obtenida en los experimentos de la etapa anterior, para reentrenar los modelos. Una vez seleccionada la estrategia que vamos a utilizar para reentrenar a nuestro modelo, seleccionamos los nuevos datos de entrada y realizamos los experimentos.

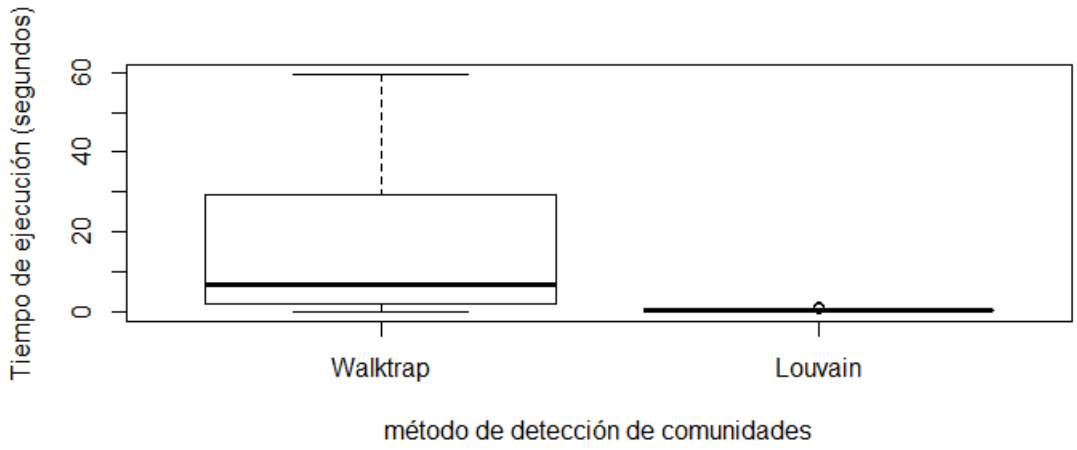

Fig. 2.9: Comparación de tiempos de ejecución de Walktrap vs Louvain en todo el período entre agosto y diciembre del 2019

2.2.2. Generación de los grafos y modelos

Según la estrategia seleccionada en el paso anterior, definiremos cómo vamos a reentrenar nuestro modelo, ya sea agregando días nuevos, eliminando días anteriores, etc. En cualquier caso, necesitamos volver a generar un nuevo grafo de interacciones para poder generar los modelos predictivos.

Para esta etapa decidimos utilizar a Louvain [12] como ground truth ya que al agregar cada vez más datos el método de Walktrap se vuelve muy costoso de ejecutar y Louvain logra identificar a las comunidades con un menor costo computacional y los resultados son bastante similares. El algoritmo de Louvain es un método heurístico goloso que esta basado en la optimización de la modularidad de los datos, funciona muy rápido y se puede correr en redes grandes sin riesgo a quedarse sin memoria. A diferencia de Walktrap, trabaja en grafos no dirigidos. Inicialmente cada usuario se asocia a una comunidad, luego se compara la densidad de aristas entre los nodos y se los asocia formando comunidades más grandes. Este proceso se repite hasta alcanzar la convergencia. Al ser heurístico, funciona mucho más rápido pero tiende a dar resultados menos óptimos. En el boxplot de la figura 2.9 se observa la comparación entre los tiempos de ejecución de ambos algoritmos en cada uno de los días desde el 12 de agosto del 2019 al 31 de diciembre del 2019. Este diagrama representa una serie de datos a través de sus cuartiles. La caja abarca el rango intercuartil de la distribución, es decir, el tramo que va desde el primer cuartil hasta el tercero, lo que incluye el 50 % de las observaciones centrales. La linea negra representa la mediana de los tiempos de ejecución, las lineas al principio y el final son los valores mínimos y máximos obtenidos. Se puede ver claramente cómo el algoritmo de Walktrap necesita de mucho más tiempo para ejecutarse.

2.2.3. Predicción y evaluación de los modelos

Una vez generado el modelo, se lo utiliza para predecir en el día siguiente al último día del entrenamiento. Para esto ejecutamos FastText para obtener la probabilidad de

pertenencia de un usuario a cada comunidad y utilizamos Louvain como ground-truth para poder identificar las predicciones correctas. Luego calculamos la métrica de ROC y analizamos el AUC para determinar la eficacia de la predicción. Ya con este resultado, se utiliza la estrategia seleccionada previamente y se determina si el resultado obtenido dictamina que es necesario un reentrenamiento, y de ser así, se vuelve a la etapa anterior para reentrenar el modelo; de lo contrario, se seleccionan los datos del siguiente día y se vuelve a predecir, repitiendo este proceso hasta que se haya analizado toda la información disponible. Como mencionamos anteriormente, en este trabajo separamos a los datasets en Tweets realizados por día para utilizar tanto en el entrenamiento como en la predicción, obteniendo una unidad de tiempo que es lo suficientemente pequeña para identificar la aparición de un nuevo tópico y lo suficientemente grande como para analizar la jerga de nuestro predictor.

3. EXPERIMENTOS Y RESULTADOS

3.1. Generación de los modelos

Nuestro objetivo en esta etapa es encontrar la mejor estrategia para crear modelos predictivos que permitan clasificar a las personas en las principales comunidades en base a la jerga utilizada. Para eso analizamos distintos experimentos hasta obtener el método de entrenamiento que consideramos que funciona mejor.

En una primera etapa, generamos modelos con pocos días de entrenamiento para identificar más fácilmente los posibles motivos de un buen o mal resultado en una predicción.

Analizamos modelos en intervalos de 7 días. Para eso generamos modelos de 1 día utilizando el 90 % de los tweets de ese día para predecir los 3 días siguientes, los 3 anteriores y el 10 % de los tweets no utilizados del día seleccionado. Nuestra hipótesis es que, como la jerga se mantiene a lo largo del tiempo, es indistinto predecir en días anteriores o posteriores a la generación del modelo. Este experimento no nos dio buenos resultados, pero nos hizo entender que entrenar en un día solo es una mala idea. Si bien los días contiguos al día de entrenamiento dieron un buen resultado, al alejarse 1 o 2 días más la predicción empieza a disminuir drásticamente, como se puede ver en la figura 3.1. Además, dependiendo del día seleccionado, se podría estar hablando de un tópico en particular y existe la posibilidad de que la predicción estuviera captando este tópico y no la jerga, como era nuestra intención. En consecuencia, decidimos ampliar este experimento incrementando la cantidad de días a utilizar para la generación del modelo. Utilizamos entonces n días contiguos para generar el modelo, pero los resultados obtenidos fueron muy similares. Analizando las conversaciones de los tweets, reconfirmamos nuestra teoría de que en el ámbito de la política y, en particular, en el contexto de las elecciones presidenciales, en una misma semana pueden aparecer diversos cambios de tópico producto de las campañas, discursos, etc. Así, entendimos que teníamos que buscar una estrategia que nos permitiera capturar la jerga abstrayéndonos de los tópicos conversados durante unos pocos días. Es en este contexto que decidimos modificar la forma de entrenar los modelos, bajo la hipótesis de que, al seleccionar una cierta cantidad de días no contiguos de cada semana, podríamos sacar el foco de lo que se trató unos días en particular y concentrarnos en la jerga utilizada por las personas, obteniendo mejores resultados en las predicciones.

3.1.1. Intervalos cerrados

La intención de este experimento fue la de evaluar la capacidad de predicción dentro de un intervalo cerrado utilizando días de este como entrenamiento. Si bien la jerga puede mutar, es algo que perdura a lo largo del tiempo. Es por esto que, si generamos un modelo con información de algunos días dentro de un intervalo cerrado, este modelo debería poder predecir con una buena probabilidad dentro de este intervalo, dado que este capturaría la jerga utilizada en este período. Para tener una buena estimación de la performance, el ROC AUC de los días de entrenamiento se estimó mediante cross-validation [14] utilizando el 90 % de los datos para entrenar y el 10 % restante para testear.

En este experimento predijimos dentro del mes de diciembre, utilizando solamente 8 días para el entrenamiento. Entrenamos un modelo A, con los días 1 al 4 y 28 al 31 de diciembre, es decir con los primeros y últimos días del intervalo y otro B utilizando 2 días

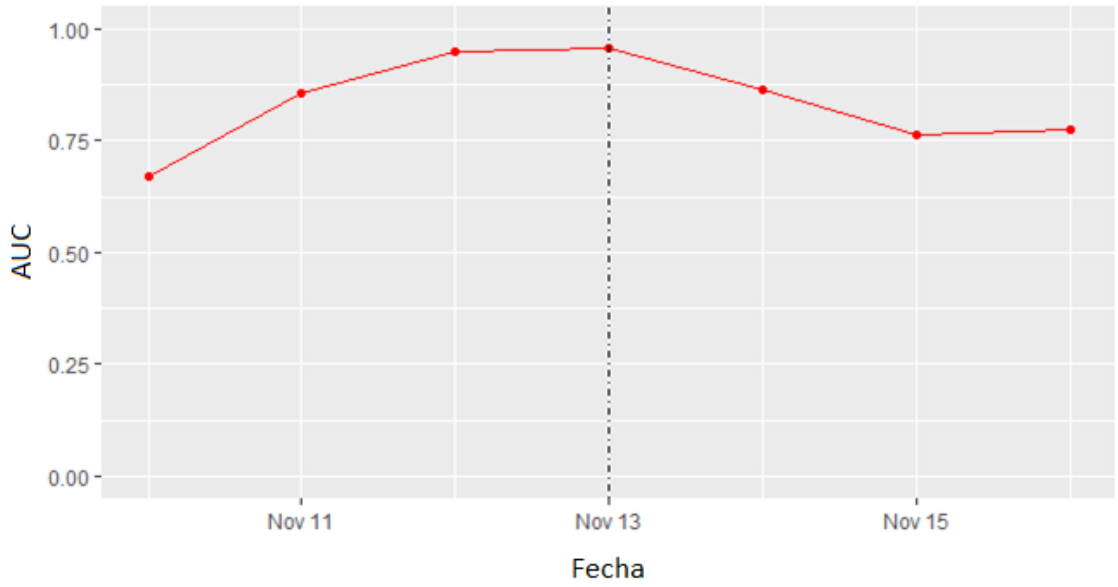

Fig. 3.1: AUC de las predicciones del modelo entrenado con 1 solo día

de cada semana del mes es decir, 3,5,10,12,17,19,24,26 de diciembre. En la figura. 3.2, graficamos el ROC AUC por día de cada modelo. Se observa que, para el modelo A, el AUC promedio es de 0.78 y su SD de 0.13 mientras que para el modelo B, el AUC promedio es de 0.82 y su SD de 0.129

Los resultados de este experimento nos muestran que es más beneficioso entrenar con distintos días de la semana en un intervalo cerrado que entrenar solamente con los extremos, ya que al entrenar con días contiguos podemos estar entrenando sobre un tópico, mientras que al generar el modelo utilizando distintos momentos, podemos minimizar este riesgo. Seleccionamos solamente este ejemplo como ilustración de que entrenar 2 días de la semana no contiguos obtiene en general mejores resultados, pero este mismo experimento se realizó también para los datasets de septiembre, octubre y noviembre con resultados similares.

3.1.2. Predecir los días futuros

En el experimento pasado, pudimos ver que seleccionar 2 días de cada semana nos permite obtener una mejor interpretación de la jerga que seleccionando los primeros 4 días y los últimos 4. En este experimento, nuestro objetivo es probar si esta misma estrategia nos resulta efectiva para predecir los días futuros. Para probar esto, entrenamos 2 modelos de 8 días cada uno con el objetivo de predecir los 7 días siguientes. El primer modelo lo entrenamos con 8 días anteriores a la semana que queremos predecir y el segundo, lo entrenamos utilizando 2 días de cada semana de las 4 anteriores.

En este caso, el Modelo A, utiliza los 8 días anteriores a la semana que queremos predecir, es decir, del 7 al 14 de noviembre y el Modelo B, utiliza 2 días de cada semana de las 4 anteriores, siendo estos los días 15, 17, 22 y 24 de octubre y 5,7,12 y 14 de noviembre y evaluamos sobre los siguientes 7 días. En la figura. 3.3, graficamos el ROC AUC por día

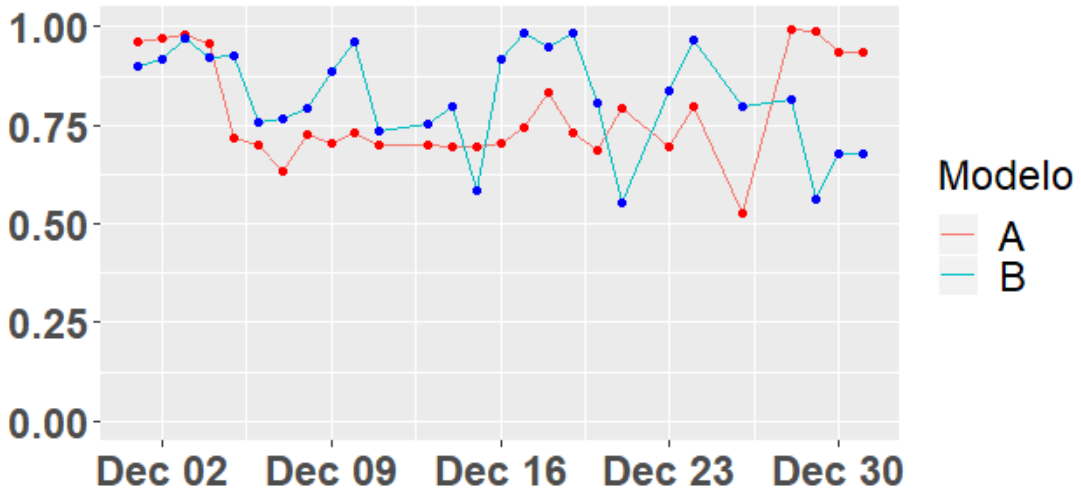

Fig. 3.2: AUC por día entrenando dentro del intervalo de predicción

de cada modelo. Se observa que, para el modelo A, el AUC promedio es de 0.73 y su SD de 0.105 mientras que para el modelo B, el AUC promedio es de 0.796 y su SD de 0.023

Los resultados obtenidos demuestran que utilizar los 2 días de cada semana para entrenar nuestro modelo nos permite predecir con mayor precisión los días futuros, lo cual respalda nuestra hipótesis inicial.

3.1.3. Aumentando la cantidad de datos

Este experimento consistió en verificar si aumentar la cantidad de días de entrenamiento ayuda a mejorar la capacidad de predecir del modelo.

Para esto entrenamos un modelo A, con todos los días de septiembre y octubre e intentamos predecir en noviembre y lo comparamos con otro B, usando solamente 2 días de cada semana de esos mismos meses, es decir los días 3,6,10,12,17,19,24,27 de septiembre y los días 8,10,15,17,22,24 y 29 de octubre. En la figura. 3.4, graficamos el ROC AUC por día de cada modelo. Se observa que, para el modelo A, el AUC promedio es de 0.716 y su SD de 0.054 mientras que para el modelo B, el AUC promedio es de 0.711 y su SD de 0.08.

Este experimento nos demuestra que agregar más días para entrenar el modelo no necesariamente redundará en mejores resultados. Se observa que utilizando una diferencia de 44 días entre los dos modelos, la diferencia obtenida fue tan solo de 0.005 a favor del modelo A.

3.2. Reentrenamiento de modelos

Como pudimos ver en los experimentos anteriores, al pasar el tiempo o intentar predecir las comunidades fuera del intervalo cerrado, la eficiencia del modelo empieza a disminuir. Como explicamos anteriormente, a lo largo del tiempo suelen aparecer cambios de tópicos, estos eventos modifican la jerga para siempre o simplemente por uno o varios días, el

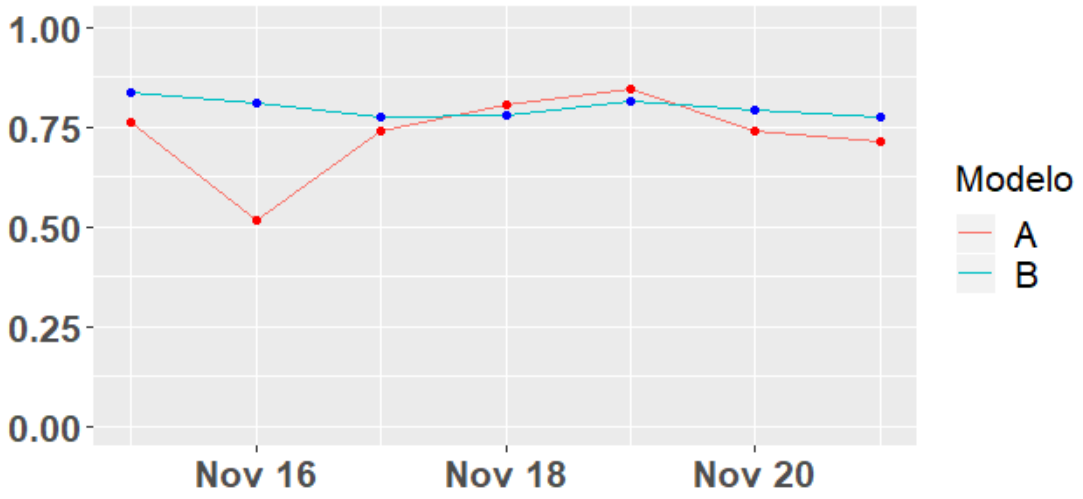

Fig. 3.3: AUC por día entrenando dentro del intervalo de predicción

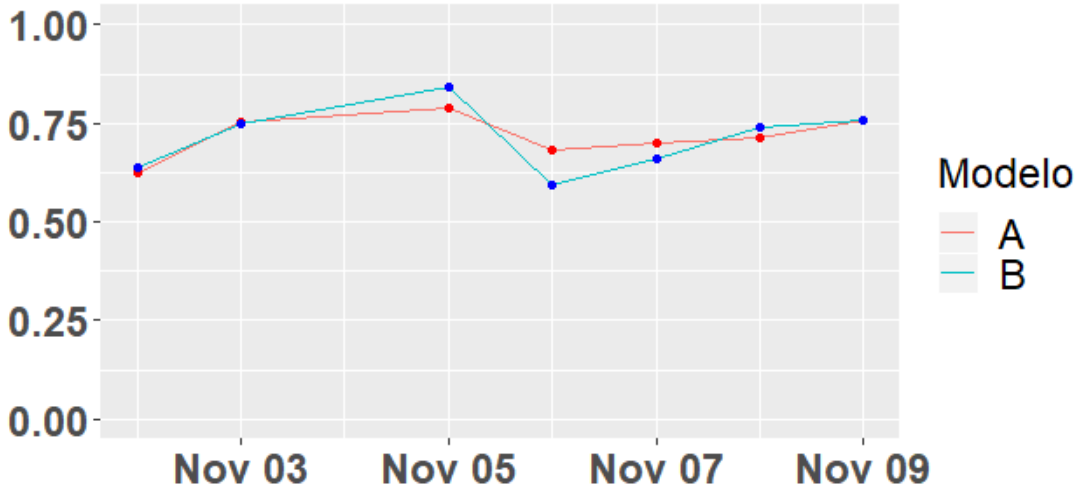

Fig. 3.4: AUC por día entrenando dentro del intervalo de predicción

problema es que es imposible determinar cuando van a ocurrir. Bajo esta premisa, necesitamos definir cuándo es que un modelo empieza a ser malo y necesita ser reentrenado para mantener la calidad de los resultados obtenidos. Decimos que un modelo predice mal cuando el AUC obtenido es inferior a 0.7.

En esta etapa vamos a realizar diversos métodos para determinar qué estrategia puede resultar más efectiva. En cada método se experimentará con los mismos datasets, estos están detallados en la tabla 3.1.

Dataset	Definición
DS-Septiembre	Contiene los tweets bajo el criterio *Macri* de los días de septiembre y octubre del 2019.
DS-Octubre	Contiene los tweets bajo el criterio *Macri* de los días de octubre y noviembre del 2019.
DS-Noviembre	Contiene los tweets bajo el criterio *Macri* de los días de noviembre y diciembre del 2019.
DS-Bolsonaro	Contiene los tweets bajo el criterio *Bolsonaro* de los días de junio y julio del 2020.

Tab. 3.1: Datasets utilizados para los experimentos de reentrenamiento.

3.2.1. Aprendiendo de los buenos y malos resultados

En este método intentamos evitar el problema de confundir que al disminuir la calidad de la predicción, esto sea producto de la aparición de un nuevo tópico o del cambio de la jerga, es por eso que consideramos que el modelo no está prediciendo correctamente al encontrar 2 días consecutivos con un AUC menor a 0.7. En este caso, conservamos el criterio de entrenar con días no contiguos, utilizamos entonces el último día que predijo correctamente como refuerzo de la jerga que se viene utilizando e incorporamos el segundo día con AUC menor a 0.7 a nuestro modelo, estableciendo que si la mala predicción se mantiene más de un día, es necesario incorporar la nueva terminología para poder seguir prediciendo correctamente.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.5 y 3.6, para DS-Septiembre este método no pudo ser aplicado porque en este período no ocurrió que 2 días consecutivos tengan un AUC menor a 0.7. Esto mismo ocurre en los próximos 3 métodos que utilizan una lógica similar. Para el DS-Octubre se logró un AUC promedio de 0.719, un SD de 0.109 en 5 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.72, un SD de 0.11 en 2 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.754, un SD de 0.07 en 2 reentrenamientos.

3.2.2. Experimento 3.2.1, Quitándole relevancia a los datos más viejos.

Al pasar el tiempo, puede que la jerga empiece a mutar producto de los cambios de tópico, es por eso que creemos que la forma de hablar en el pasado no sea la misma que en el futuro, nuestra intención entonces en esta etapa, es la de ir disminuyendo en un 10 % el

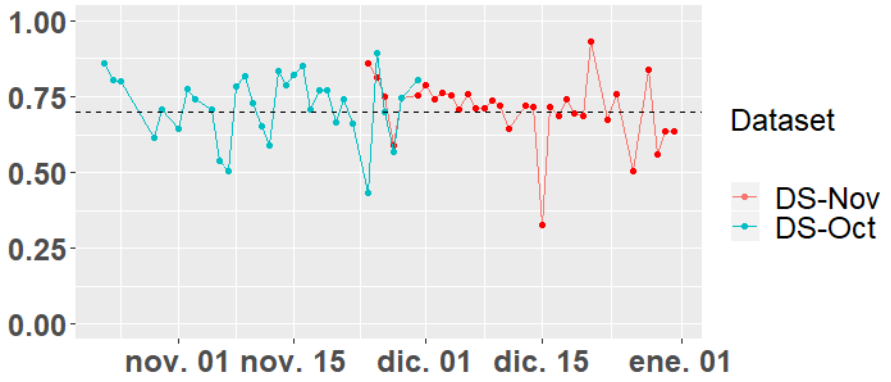

Fig. 3.5: AUC por día con los Datasets de las PASO del 2019 en Argentina

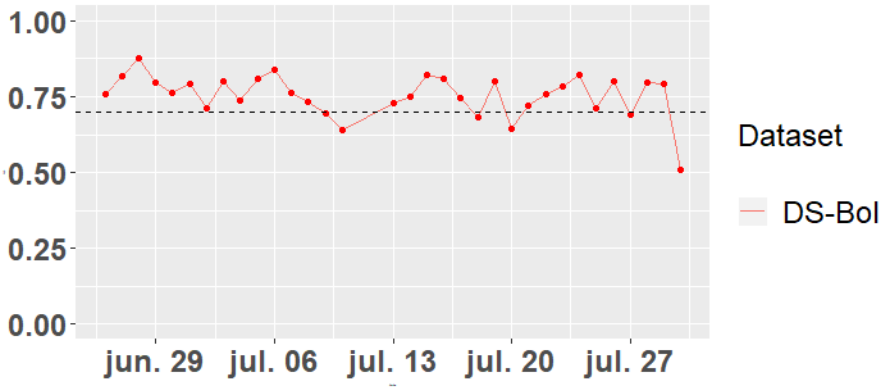

Fig. 3.6: AUC por día con el Dataset de Brasil en el 2020

porcentaje de datos utilizados de los días más viejos al agregar un día nuevo, es decir, al agregar 10 días el primer día de entrenamiento dejaría de utilizarse y así sucesivamente. En este experimento, utilizamos el mismo criterio de reentrenamiento que en el experimento pasado, es decir, utilizar para el reentrenamiento el último día cuyo AUC fue mayor al 0.7 y el segundo consecutivo menor a este valor, pero a diferencia del método anterior, disminuimos el porcentaje de datos utilizados de los tweets más viejos en un 10 % por día agregado.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.7 y 3.8. Para el DS-Octubre se logró un AUC promedio de 0.735, un SD de 0.088 en 4 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.707, un SD de 0.078 en 1 reentrenamiento. Para DS-Bolsonaro, un AUC de 0.731, un SD de 0.096 en 3 reentrenamientos. Como mencionamos anteriormente, para el DS-Septiembre este método no pudo ser aplicado.

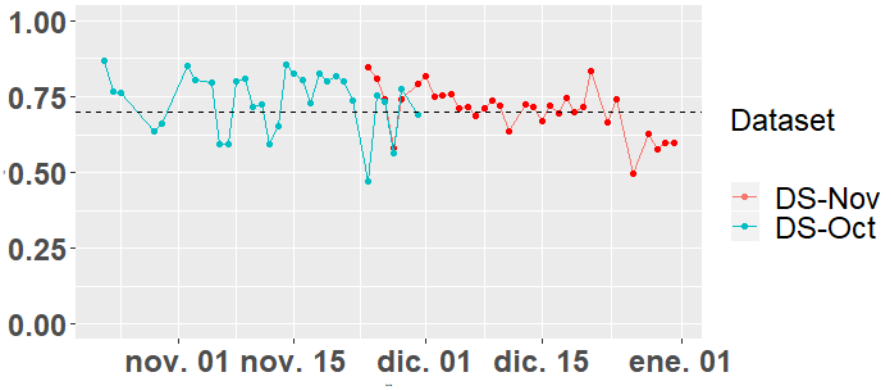

Fig. 3.7: AUC por día con los Datasets de las PASO del 2019 en Argentina

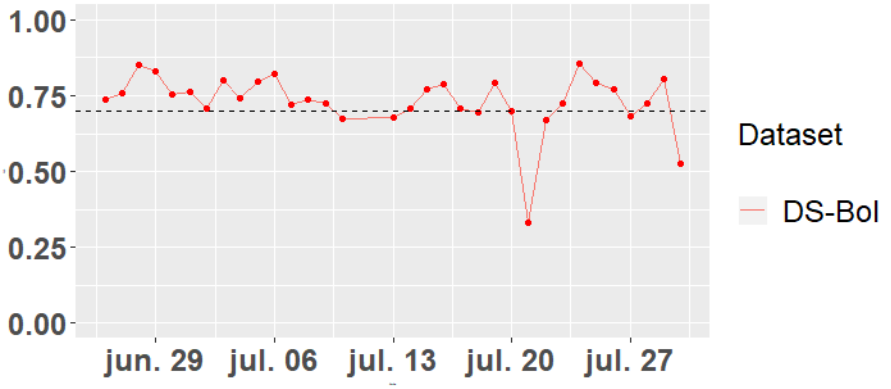

Fig. 3.8: AUC por día con el Dataset de Brasil en el 2020

3.2.3. Utilizando los primeros 2 días consecutivos con mal resultado

Para este método, nos inspiramos en la técnica de *Boosting* ya que ponderamos los datos mal clasificados del modelo anterior para generar mejores clasificadores. Basándonos en esto, al encontrar 2 días consecutivos con un AUC menor a 0.7, decidimos utilizar a estos 2 días para el reentrenamiento, para reforzar el modelo con los días en los que la predicción fue más débil.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.9 y 3.10. Para el DS-Octubre se logró un AUC promedio de 0.744, un SD de 0.095 en 4 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.735, un SD de 0.088 en 4 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.746, un SD de 0.089 en 2 reentrenamientos. Como mencionamos en el punto 3.2.1, para el DS-Septiembre este método no pudo ser aplicado.

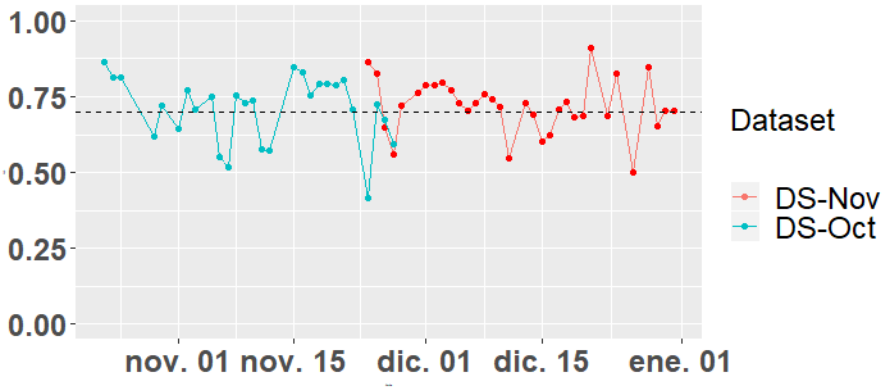

Fig. 3.9: AUC por día con los Datasets de las PASO del 2019 en Argentina

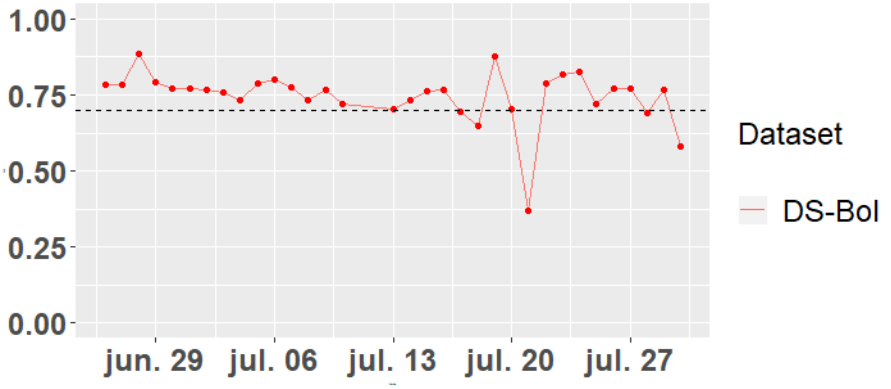

Fig. 3.10: AUC por día con el Dataset de Brasil en el 2020

3.2.4. Experimento 3.2.3, Quitándole relevancia a los datos más viejos.

Al igual que en el método anterior, decidimos utilizar la técnica de reentrenar nuestro modelo incorporando los últimos 2 días consecutivos con AUC menor a 0.7 y disminuir un

10 % el porcentaje de datos utilizados de los días más viejos por cada día agregado.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.11 y 3.12. Para el DS-Octubre se logró un AUC promedio de 0.714, un SD de 0.116 en 4 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.728, un SD de 0.072 en 2 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.741, un SD de 0.108 en 2 reentrenamientos. Como mencionamos en el punto 3.2.1, para el DS-Septiembre este método no pudo ser aplicado.

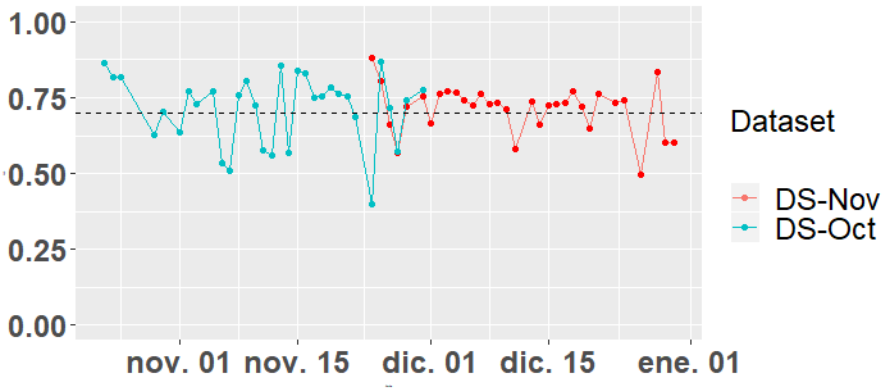

Fig. 3.11: AUC por día con los Datasets de las PASO del 2019 en Argentina

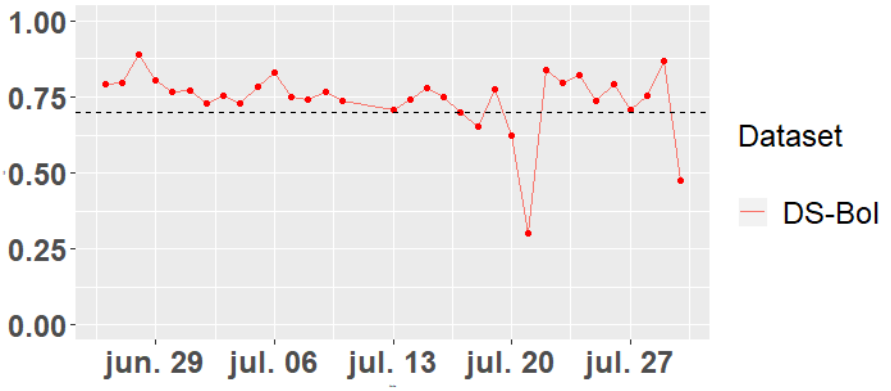

Fig. 3.12: AUC por día con el Dataset de Brasil en el 2020

3.2.5. Sliding Window los martes y jueves

En este experimento nos inspiramos en el concepto de Sliding Window. En este método se establece una ventana de x envíos simultáneos para no sofocar al receptor con demasiada información y se espera la respuesta para continuar la transmisión, nosotros en nuestro caso tenemos una ventana de 4 días, siendo estos los martes y jueves de cada semana, para no sobrecargar al modelo con información pasada, vamos eliminando la información del día más viejo al agregar los datos del nuevo día. El objetivo es verificar si entrenar siempre con los mismos días de la semana y eliminando los datos más viejos, se logre generar un

modelo que esté más actualizado a lo que está ocurriendo y pueda ser menos sensible a los cambios de tópico.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.13 y 3.14. Para el DS-Septiembre se logró un AUC promedio de 0.764, un SD de 0.084 en 9 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.725, un SD de 0.111 en 11 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.742, un SD de 0.082 en 11 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.758, un SD de 0.083 en 13 reentrenamientos.

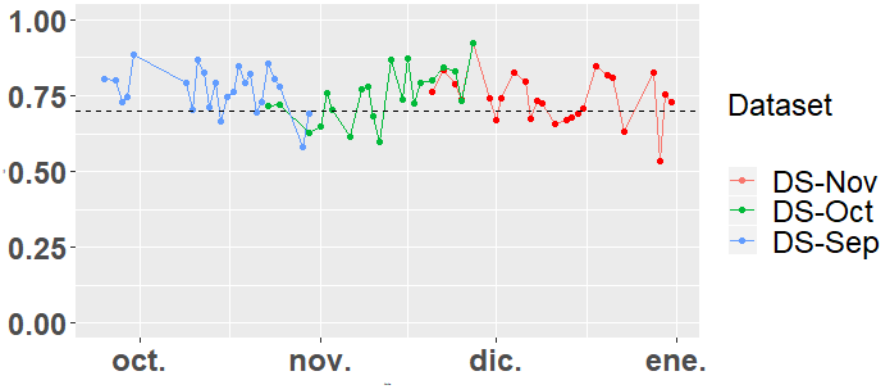

Fig. 3.13: AUC por día con los Datasets de las PASO del 2019 en Argentina

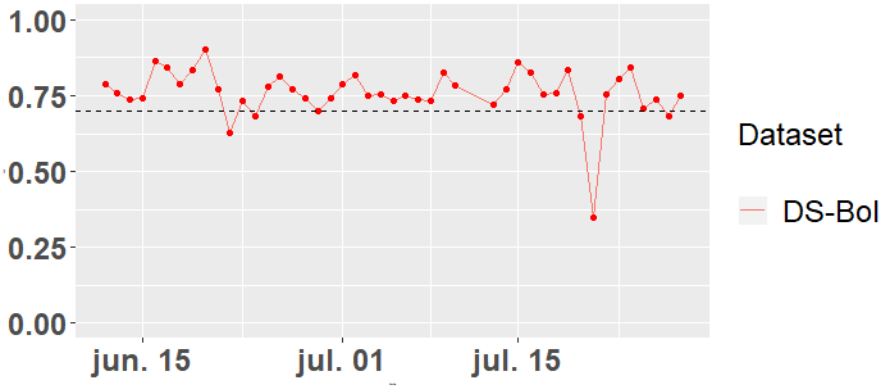

Fig. 3.14: AUC por día con el Dataset de Brasil en el 2020

3.2.6. Forzando reentrenamientos semanales

Cuando va pasando el tiempo, la calidad de la predicción empieza a disminuir, no siempre ocurre que 2 días seguidos el AUC sea menor que 0.7, muchas veces ese valor se mantiene muy cerca de esta linea, es por eso que decidimos realizar un reentrenamiento si al pasar una semana no fue necesario realizar uno, de esta manera buscamos mejorar el AUC obtenido y evitar un posible reentrenamiento futuro por un mal resultado. Para este reentrenamiento agregamos el último día de esta semana.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.15 y 3.16. Para el DS-Septiembre se logró un AUC promedio de 0.741, un SD de 0.054 en 4 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.726, un SD de 0.102 en 5 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.726, un SD de 0.085 en 5 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.758, un SD de 0.091 en 6 reentrenamientos.

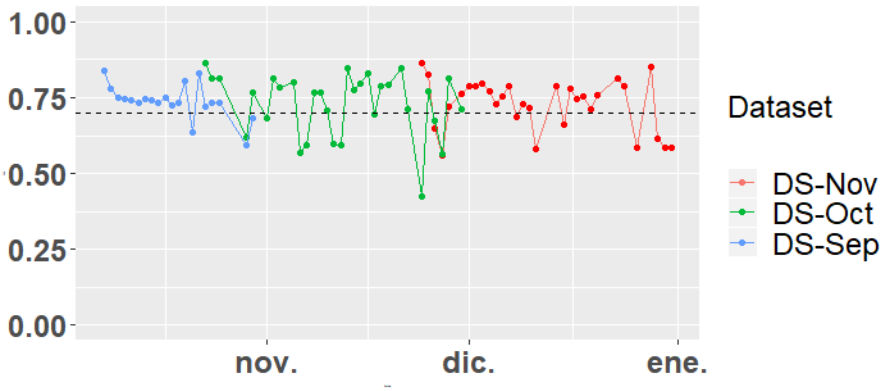

Fig. 3.15: AUC por día con los Datasets de las PASO del 2019 en Argentina

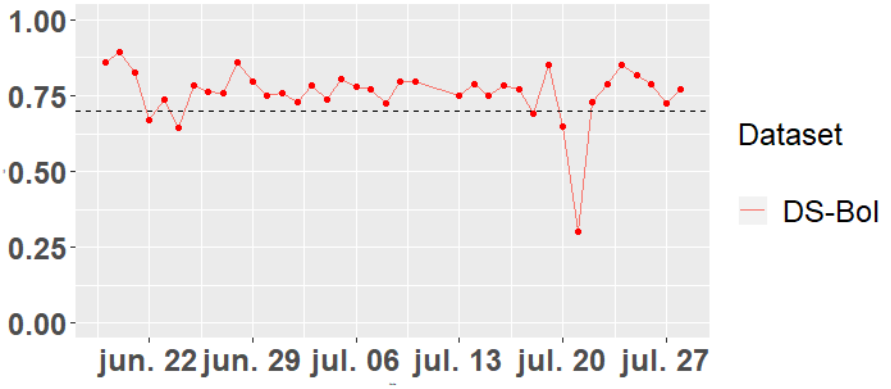

Fig. 3.16: AUC por día con el Dataset de Brasil en el 2020

3.2.7. Nueva métrica de performance a lo largo del tiempo

Siguiendo con el espiritu del caso anterior, no siempre ocurre que el AUC sea menor que 0.7 durante 2 días contiguos pero igual se puede detectar que la calidad del predictor empieza a disminuir y los valores de la predicción empiezan a oscilar entre el valor del punto de corte, obteniendo malos resultados pero no entrando en los planteados hasta el momento para reentrenar. Por otro lado a veces puede tener muy buenos resultados pero durante 2 días contiguos tiene malos llevándonos a un reentrenamiento cuando a lo mejor simplemente se trató de un tópico que duró más tiempo de lo común y terminamos incorporando a nuestro modelo información de un tópico, haciendo que este sea menos

performante en el futuro. Bajo esta lógica, introducimos una métrica que determina qué tanto está disminuyendo la calidad. Para eso, empezamos a almacenar la diferencia que hay entre el valor del AUC obtenido y nuestro umbral de 0.7, es decir, $\Delta = AUC - 0{,}7$. Cuando la sumatoria de los valores almacenados es menor que 0.05 realizamos un reentrenamiento agregando a nuestro modelo el último día que hizo que sea necesario el reentrenamiento y reseteamos el acumulador. Por ejemplo, si nuestro AUC fue de 0.76, al día siguiente 0.74 y al siguiente 0.63, nuestro $\Delta_1 = 0{,}06$ $\Delta_2 = 0{,}04$ $\Delta_3 = -0{,}07$ y finalmente $\sum_{i=0}^{3} \Delta_i = 0{,}03$ Como en este caso el valor resultante es menor a 0.05, realizamos un reentrenamiento.

El Δ se almacena hasta una semana para evitar que muy buenas predicciones del pasado hagan que el predictor prediga muy mal durante muchos días hasta que el valor del umbral sea menor a 0.05

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.17 y 3.18. Para el DS-Septiembre se logró un AUC promedio de 0.762, un SD de 0.070 en 3 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.744, un SD de 0.106 en 6 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.741, un SD de 0.070 en 7 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.743, un SD de 0.095 en 4 reentrenamientos.

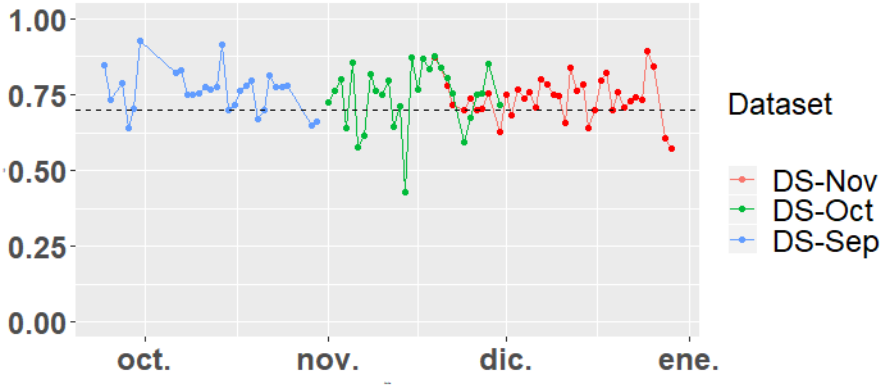

Fig. 3.17: AUC por día con los Datasets de las PASO del 2019 en Argentina

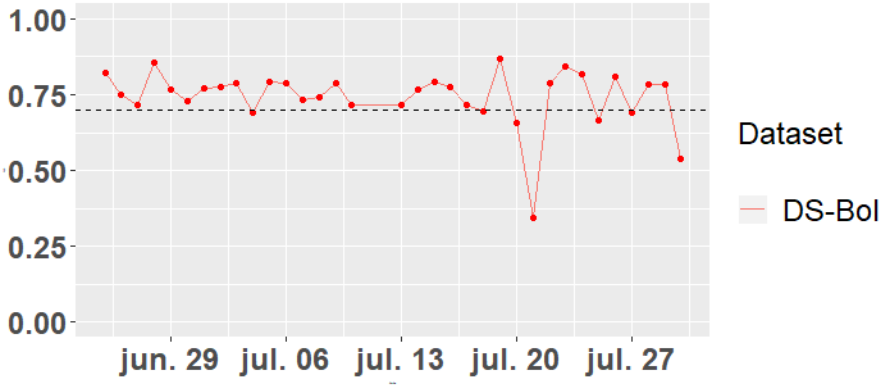

Fig. 3.18: AUC por día con el Dataset de Brasil en el 2020

3.2.8. Experimento 3.2.7, Quitándole relevancia a los datos más viejos

Al igual que en experimentos anteriores, la idea es repetir el mismo proceso que en el experimento anterior disminuyendo en un 10 % el porcentaje de datos utilizados en los días más viejos al agregar un día nuevo al conjunto de entrenamiento

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.19 y 3.20. Para el DS-Septiembre se logró un AUC promedio de 0.759, un SD de 0.079 en 3 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.737, un SD de 0.115 en 6 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.751, un SD de 0.089 en 8 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.743, un SD de 0.088 en 3 reentrenamientos.

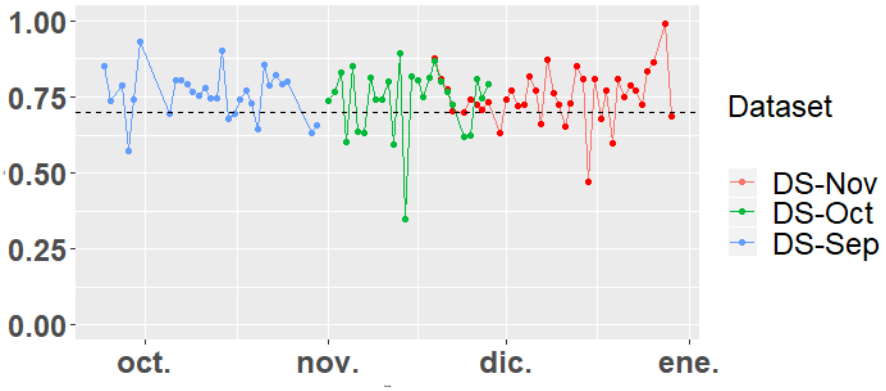

Fig. 3.19: AUC por día con los Datasets de las PASO del 2019 en Argentina

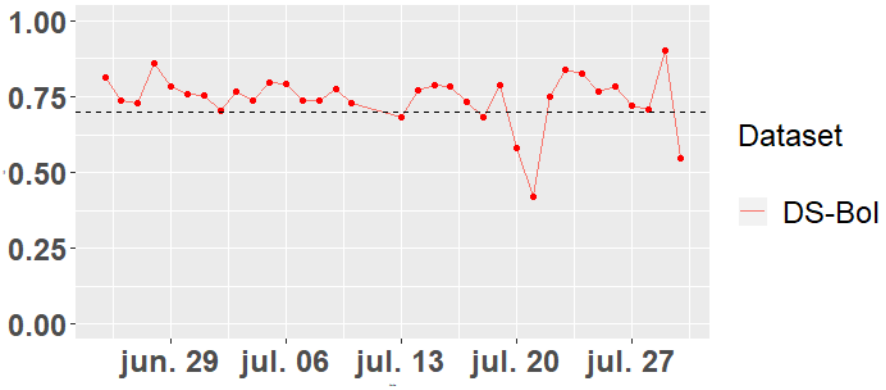

Fig. 3.20: AUC por día con el Dataset de Brasil en el 2020

3.2.9. Se hace más rigurosa la métrica establecida en el experimento 3.2.7

Siguiendo la misma linea que en experimento 3.2.7, se sube el umbral de 0.05 a 0.07 con el objetivo de aumentar la calidad de la predicción.

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.21 y 3.22. Para el DS-Septiembre se logró un AUC promedio de 0.771, un SD de 0.066 en 3 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.746, un SD de 0.113 en 6 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.751, un SD de 0.078 en 9 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.763, un SD de 0.089 en 8 reentrenamientos.

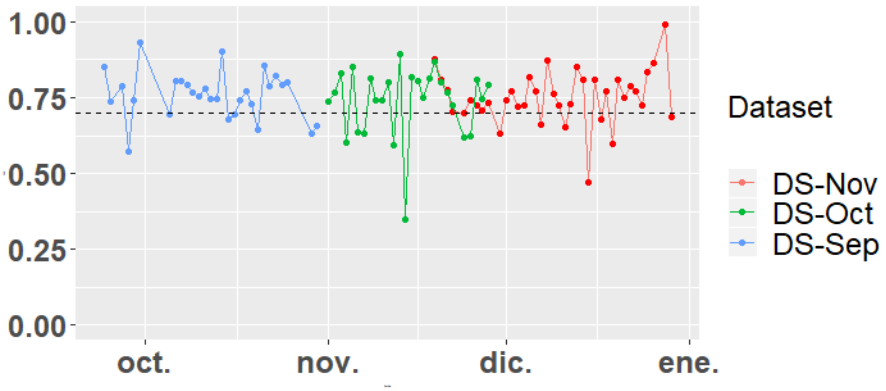

Fig. 3.21: AUC por día con los Datasets de las PASO del 2019 en Argentina

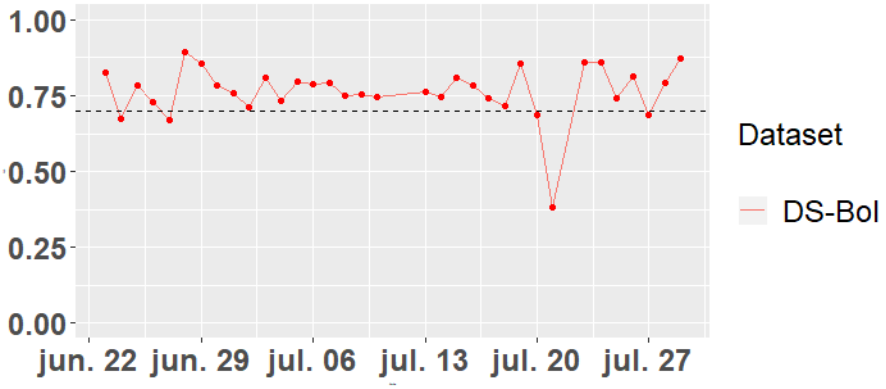

Fig. 3.22: AUC por día con el Dataset de Brasil en el 2020

3.2.10. Experimento 3.2.9, Quitándole relevancia a los datos más viejos

Volvemos a realizar el experimento anterior pero disminuyendo en un 10 % el porcentaje de los datos más viejos utilizados al agregar un nuevo día

Ejecutamos este método en nuestros datasets definidos anteriormente y generamos los gráficos 3.23 y 3.24. Para el DS-Septiembre se logró un AUC promedio de 0.768, un SD de 0.071 en 3 reentrenamientos. Para el DS-Octubre se logró un AUC promedio de 0.745, un SD de 0.101 en 6 reentrenamientos. Para DS-Noviembre un AUC promedio de 0.754, un SD de 0.09 en 10 reentrenamientos. Para DS-Bolsonaro, un AUC de 0.774, un SD de 0.05 en 7 reentrenamientos.

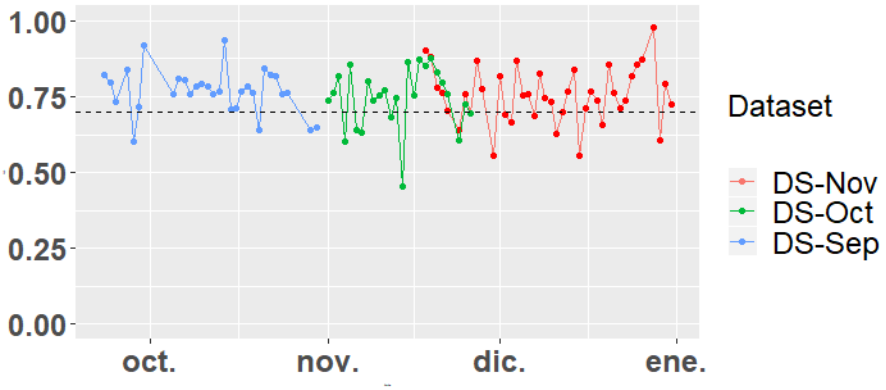

Fig. 3.23: AUC por día con los Datasets de las PASO del 2019 en Argentina

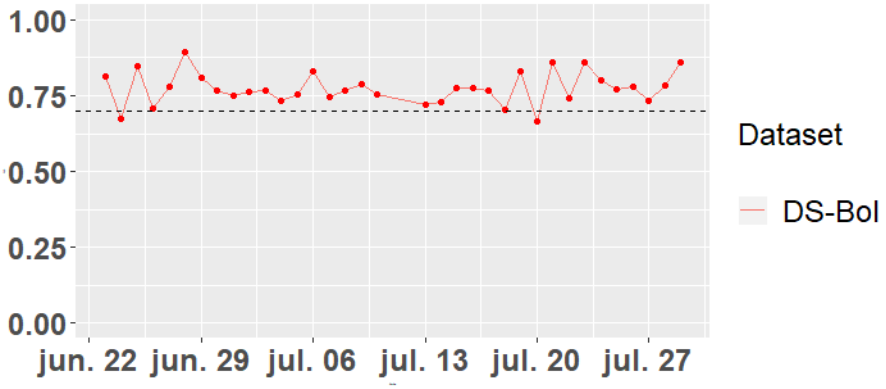

Fig. 3.24: AUC por día con el Dataset de Brasil en el 2020

3.2.11. Resultados de los reentrenamientos

Para analizar los resultados obtenidos en los distintos experimentos, dividimos el análisis en 3 ejes:

- Cantidad de iteraciones necesarias para procesar todos los datos.
- AUC promedio obtenido para cada uno de los experimentos.
- Desvío standard obtenido para cada uno de los experimentos.

Cantidad de iteraciones

En la tabla 3.25 comparamos la cantidad de reentrenamientos que fueron necesarios para poder analizar la totalidad de los datos de cada dataset. Cabe destacar que la cantidad de reentrenamientos necesarios no es algo necesariamente bueno o malo para la calidad

de la predicción, sino que simplemente depende de la estrategia de reentrenamiento seleccionada. En algunos casos siempre se reentrena utilizando la misma cantidad de días y en algunos es incremental. A medida que se incrementa la cantidad de datos, reentrenar modelos se puede volver más costoso, tanto en recursos como en tiempo, con lo cual hay que encontrar un equilibrio entre la cantidad de iteraciones, tamaño del conjunto de datos y precisión deseada para cada modelo.

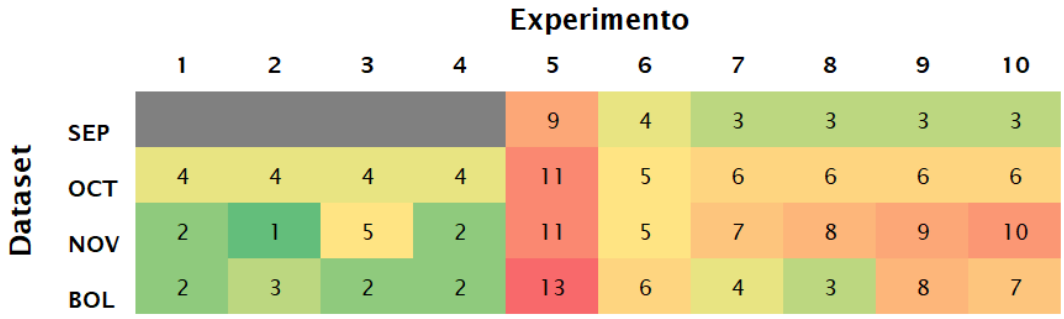

Fig. 3.25: Comparación de cantidad de reentrenamientos que fueron necesarios en los distintos experimentos

AUC promedio

En la tabla 3.26 comparamos el AUC promedio obtenido en cada uno de los experimentos. Se puede ver como los experimentos que utilizan la métrica introducida en el experimento 3.2.7 y ajustada en el experimento 3.2.9 y 3.2.10 son las que mejores resultados consiguieron.

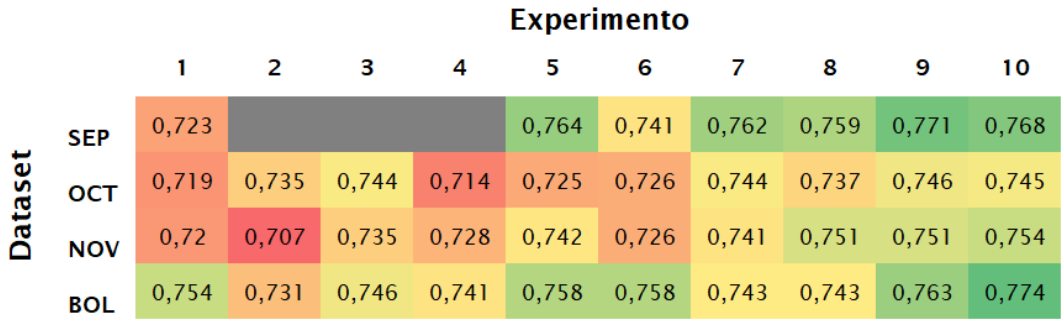

Fig. 3.26: Comparación de AUC obtenido en los distintos experimentos

Desvío standard

En la tabla 3.27 comparamos el desvío standard promedio de los resultados de los experimentos. Mientras más pequeño sea este valor, más estable va a ser nuestro predictor a lo largo del tiempo. Podemos ver que no hay ningún método más efectivo que otro en este punto. Esto se debe a que día a día puede haber mucha volatilidad en la forma de

comunicarse de las personas .

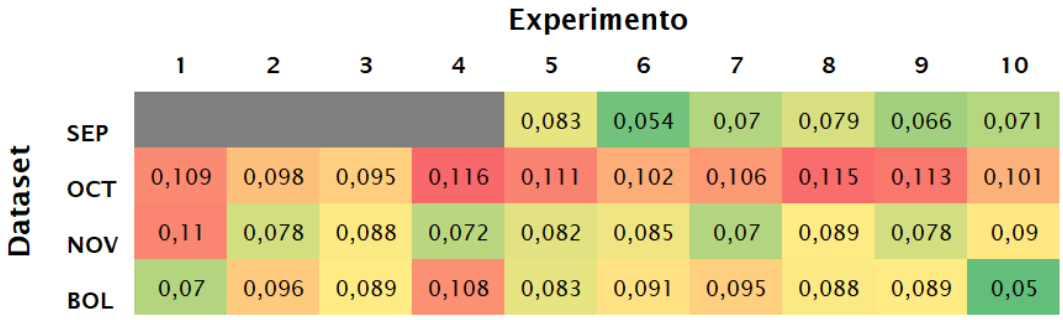

Fig. 3.27: Comparación del desvío standard de los distintos experimentos

4. CONCLUSIONES

En la sección de entrenamiento de los modelos pudimos verificar nuestra hipótesis inicial al analizar los tres experimentos. Si bien aumentar la cantidad de días es beneficioso para mejorar la performance del modelo, es importante elegir una buena estrategia para hacerlo. Esto lo podemos ver en los resultados de los primeros dos experimentos donde utilizando la misma cantidad de datos, la estrategia de seleccionar los días de entrenamiento de semanas distintas permite obtener un mejor resultado en la predicción y en el tercero, que a pesar de tener más datos y días de entrenamiento, el resultado no varía mucho.
En el experimento 3.1.1 notamos que predecir dentro del período de entrenamiento nos permite obtener un mejor resultado. Esto lo atribuimos a que los cambios de tópico no afectan tanto al predictor debido a que las jergas utilizadas en los tópicos son capturadas para el entrenamiento de los modelos dentro de este intervalo.
En la sección de reentrenamiento de modelos, pudimos ver que no hay una bala de plata para elegir el mejor método, es decir, los métodos que parecen obtener una mejor performance suelen requerir una mayor cantidad de reentrenamientos. Esto puede ser muy costoso en modelos muy grandes ya que los algoritmos de detección de comunidades empiezan a requerir muchos más recursos a medida que aumenta la cantidad de datos. En particular, Walktrap se vuelve muy costoso de correr muy rápidamente, lo cual obliga a utilizar algoritmos de optimización como Louvain. Por otro lado, en los experimentos también se puede ver que, al igual que en la sección de entrenamiento, tener más reentrenamientos o más datos no indican necesariamente que la performance del método vaya a ser superior. Consideramos que la métrica introducida en el método 3.2.9 es la que nos permite obtener un mejor resultado global. Esta, al tener una “memoria” de las predicciones pasadas, nos ayuda a evitar el reentrenamiento cuando aparece un nuevo tópico que dura unos pocos días y nos permite introducir nuevos datos cuando la calidad de la predicción empieza a disminuir, permitiendo encontrar un método estable a lo largo del tiempo.

En conclusión, nuestro trabajo nos sugiere que para obtener un modelo que pueda predecir con una buena probabilidad, debemos seleccionar una estrategia de entrenamiento de modelos que no utilicen días contiguos para la selección de datos y evitar así darle demasiada entidad a tópicos “cortos” tratados durante pocos días y para mantener a los modelos efectivos a lo largo del tiempo. Debemos ir reentrenando los modelos, incluso si estos no están prediciendo mal, para mantenerlos lo mas eficientes y más robustos posible ante la aparición de tópicos que se mantienen durante períodos más largos.

Como trabajo futuro se pueden seguir buscando nuevas estrategias para el entrenamiento y reentrenamiento de los modelos. En este trabajo, nosotros nos enfocamos en el ámbito de la política, en el cual los cambios de tópico son algo muy frecuente, motivo por el cual consideramos que luego de un tiempo iban a ser necesarios reentrenamientos para mantener el modelo útil a lo largo del tiempo. Sin embargo, la polarización en las redes sociales es un fenómeno que afecta a otros ámbitos de la sociedad con menos alteraciones de la jerga, por lo que se podría analizar si los métodos utilizados en este trabajo logran una mayor precisión en estos contextos. Teniendo esto en cuenta, un trabajo futuro podría ser el análisis de la aplicación de los métodos que desarrollamos para este trabajo al ámbito del fútbol en torno a discusiones polarizadas como "quién es mejor: Messi o Cristiano

Ronaldo, Messi o Maradona, Maradona o Pelé, Argentina o Brasil, etc."

Se puede seguir trabajando en los experimentos para encontrar si existe una cota óptima para el método 3.2.7 que maximice la calidad de la predicción. Podríamos analizar el uso de otros clasificadores de texto como por ejemplo BERT[21] que suele comportarse mejor al aumentar la cantidad de dimensiones utilizadas para los vectores de palabras o GPT[22]. En esta tesis utilizamos los algoritmos de clustering de Walktrap y Louvain, pero se podría comparar los resultados al utilizar otros como por ejemplo infomap[23], fast greedy[24] o label propagation[25]. En este trabajo experimentamos con la red social Twitter, se podría también analizar el comportamiento de las comunidades en distintas redes sociales, como Reddit o Facebook. Se podrían analizar también otros criterios para reentrenar los modelos, como por ejemplo, se podría estudiar qué palabras se utilizan más en una comunidad, cuándo aparecen y cuándo dejan de usarse para identificar cuándo conviene eliminarla de nuestros modelos predictivos. Se podría analizar también los cambios de tópico durante determinado tiempo para identificar si hay alguna manera de detectar si un tópico nuevo va a introducir lenguaje nuevo a la jerga que va a perdurar a lo largo del tiempo o va a ser algo pasajero.

Bibliografía

[1] ZACHARY, Wayne W. An information flow model for conflict and fission in small groups. Journal of anthropological research, 1977, vol. 33, no 4, p. 452-473.

[2] PARISER, Eli. The filter bubble: What the Internet is hiding from you.

[3] CHITRA, Uthsav; MUSCO, Christopher. Analyzing the impact of filter bubbles on social network polarization. En Proceedings of the 13th International Conference on Web Search and Data Mining. 2020. p. 115-123.

[4] MCPHERSON, Miller; SMITH-LOVIN, Lynn; COOK, James M. Birds of a feather: Homophily in social networks. Annual review of sociology, 2001, vol. 27, no 1, p. 415-444.

[5] ORTIZ DE ZÁRATE, Juan Manuel; FEUERSTEIN, Esteban. Identificación de comunidades a través del lenguaje. En IV Simposio Argentino de GRANdes DAtos (AGRANDA 2018)-JAIIO 47 (CABA, 2018). 2018.

[6] ZARATE, Juan Manuel Ortiz de, et al. Measuring controversy in social networks through nlp. En International Symposium on String Processing and Information Retrieval. Springer, Cham, 2020. p. 194-209.

[7] TRAN, Trang; OSTENDORF, Mari. Characterizing the language of online communities and its relation to community reception. arXiv preprint arXiv:1609.04779, 2016.

[8] RAMPONI, Giorgia, et al. Vocabulary-based community detection and characterization. En Proceedings of the 34th ACM/SIGAPP symposium on applied computing. 2019. p. 1043-1050.

[9] MIKOLOV, Tomas, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[10] JOULIN, Armand, et al. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016.

[11] PONS, Pascal; LATAPY, Matthieu. Computing communities in large networks using random walks. En International symposium on computer and information sciences. Springer, Berlin, Heidelberg, 2005. p. 284-293.

[12] CLAUSET, Aaron; NEWMAN, Mark EJ; MOORE, Cristopher. Finding community structure in very large networks. Physical review E, 2004, vol. 70, no 6, p. 066111.

[13] ROC: Rice, Marnie E and Harris, Grant T Comparing effect sizes in follow-up studies: ROC Area, Cohen's d, and r Law and human behavior, 2005, vol. 29, no. 5, pp. 615–620, Springer

[14] BERRAR, Daniel. Cross-validation. Encyclopedia of bioinformatics and computational biology, 2019, vol. 1, p. 542-545.

[15] ALBANESE, Federico, et al. Predicting shifting individuals using text mining and graph machine learning on twitter. arXiv preprint arXiv:2008.10749, 2020.

[16] ARUGUETE, Natalia; CALVO, Ernesto. Time to# protest: Selective exposure, cascading activation, and framing in social media. Journal of communication, 2018, vol. 68, no 3, p. 480-502

[17] BESSI, Alessandro, et al. Social determinants of content selection in the age of (mis) information. En International Conference on Social Informatics. Springer, Cham, 2014. p. 259-268.

[18] JACOMY, Mathieu, et al. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PloS one, 2014, vol. 9, no 6, p. e98679.

[19] JACOMY, Mathieu, et al. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PloS one, 2014, vol. 9, no 6, p. e98679.

[20] BLEI, David M.; NG, Andrew Y.; JORDAN, Michael I. Latent dirichlet allocation. the Journal of machine Learning research, 2003, vol. 3, p. 993-1022.

[21] DEVLIN, Jacob, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[22] BROWN, Tom B., et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

[23] ROSVALL, Martin; BERGSTROM, Carl T. Maps of random walks on complex networks reveal community structure. Proceedings of the national academy of sciences, 2008, vol. 105, no 4, p. 1118-1123.

[24] NEWMAN, Mark EJ. Fast algorithm for detecting community structure in networks. Physical review E, 2004, vol. 69, no 6, p. 066133.

[25] RAGHAVAN, Usha Nandini; ALBERT, Réka; KUMARA, Soundar. Near linear time algorithm to detect community structures in large-scale networks. Physical review E, 2007, vol. 76, no 3, p. 036106.