Descubrimiento de tópicos y análisis de sentimientos usando palabras de Twitter relacionadas al Covid-19

1st Alvaro Machuca Breña
Universidad ESAN
Código: 15101364
15101364@ue.edu.pe
Lima, Perú

2nd Fiorela Vargas Ignacio
Universidad ESAN
Código: 14100015
14100015@ue.edu.pe
Lima, Perú

3rd Berk Yanar Alarcón
Universidad ESAN
Código: 15100509
15100509@ue.edu.pe
Lima, Perú

***Abstract*—En el presente documento se extrajeron 3 043 171 tweets con keywords relacionados al coronavirus con el objetivo de desarrollar 3 diferentes modelos de detección de tópicos: Asignación Latente de Dirichlet (LDA), Procesos jerárquicos de Dirichlet (HDP) e Índice Semántico Latente (LSI), de tal modo que se escogió el mejor modelo el cual fue el LDA con 12 tópicos para realizar la detección de tópicos de un determinado tweet (Fase 1). Por otro lado, se crearon 9 modelos de machine learning con 3 clasificadores (Regresión Logística, RandomForest y Red Neuronal) con 3 técnicas de extracción de características diferentes (TF-IDF, Ngrams y Word2vect) con el objetivo de obtener el mejor modelo y poder realizar análisis de sentimiento (Fase 2). El mejor modelo fue la regresión logística con TF-IDF con un accuracy de 79%. Finalmente, se creó un sistema para detectar el tópico al cual pertenece un determinado tweet y el sentimiento que posee en tiempo real.**

***Index Terms*—Covid-19, Coronavirus, LDA, HDP, LSI, Text Mining, Sentiment Analysis.**

I. Introduction

Actualmente, todo el mundo se encuentra luchando contra un nuevo virus surgido en china en diciembre del 2019 conocido como Covid-19. Este nuevo tipo de coronavirus ha obligado a todos los paises a tomar medidas drásticas con el objetivo de disminuir el número de contagiados. Una de las medidas optadas por los países ha sido la cuarententa total con el objetivo de poder disminuir el tránsito de personas y así evitar la propagación del virus. Debido a esto último, las personas pasan mucho más tiempo en sus casas, y así han hecho uso de las redes sociales para expresar sus emociones con respecto a todas las declaraciones yo nuevas medidas optadas por el propio gobierno con respecto al coronavirus en general.

Dada la cantidad gran cantindad de información que se genera por las personas en estos tiempos de coronavirus, es dificil dar seguimiento a cada uno de los comentarios que se van publicando en las distintas redes sociales como Twitter o Facebook con el objetivo de determinar si las personas están a favor o en contra con respecto a algún tipo de situación relacionada al coronavirus o de qué tópico suelen estar hablando mucho más en sus comentarios. No obstante, con el surgimiento de nuevas técnicas para el análisis de información y para la detección de tópicos, es posible la realización de un modelo que permita dado un comentario de entrada, poder detectar a qué tópico pertenece y qué tipo de sentimiento tiene (positivo o negativo).

En el presente proyecto se realizará, en primer lugar, la recolección de una gran cantidad de tweets de Perú utilizando palabras clave relacionadas a este nuevo virus como ”Coronavirus”, ”pandemia”, ”vacuna”, entre otros, durante un periodo de 6 semanas utilizando la API de twitter. Esta cantidad de tweets que se reunirá, cumplirá por lo menos con 2 de las 3Vs de Big Data (Volumen, Varidad y Velocidad). En segundo lugar, se realizará un análisis comparativo de las técnivas de LDA, LCI y HDP para la detección de tópicos con el objetivo de utilizar una de las 3 anteriores mencionadas en función a la que posea mejor resultado en la evaluación de las métricas de cada uno de los 3 modelos. El modelo de detección de tópicos que resulte ser el mejor, se utilizará para poder detectar los temas presentes en un tweet. Finalmente, se utiliza una técnica de análisis de sentimientos para detectar si un tweet tiene un sentimiento positivo o negativo. De igual manera se evaluarán 3 modelos para realizar análisis de sentimientos que serán Regresión Logística, RandomForest y una red neuronal. Se eligirá uno de los 3 modelos anteriormente mencionados en función a los resultados del análisis comparativo con el objetivo de poder clasificar un tweet en función al sentimiento que este posea.

II. Estado del arte

En esta sección, se procederán a explicar cuatro artículos relacionados con las técnicas que se aplicarán en este documento, la metodología que siguieron los autores y los resultados obtenidos que servirán de base para el desarrollo del presente proyecto de investigación.

A. Opinion Mining on Mandalika Hotel Reviews Using Latent Dirichlet Allocation

Este primer trabajo consiste en la utilización de la técnica Latent Dirichlet Allocation conocida como LDA con el ob-

jetivo de detectar los tópicos presentes en los comentarios con respecto al hotel ”Mandalika Hotel” en indonesia. Es importante mencionar que la actividad del turismo aporta en gran cantidad a la economía de un país. En el caso de indonesia, de enero a junio del 2018, la cantidad de turistas se incrementó en 13,08%. No obstante, aún se encuentra por debajo de paises como Singapur, Malasia y Tailandia [1]. Por otro lado, existe un problema sobre las reseñas que se suben a internet sobre este hotel en particular dado a que al gerente y trabajadores en general les gustaría saber sobre qué temas especificamente hablan los clientes con respecto al hotel con el objetivo de crear nuevas estrategias para mejorar los servicios que se ofrece y poder atender de una mejor manera a todas las personas que deseen visitar este hotel [1]. Por lo tanto, se reunieron 1187 reseñas diferentes sobre Mandalika Hoyel a través de Python 3.6 para realizar el modelo de detección de tópicos. La metodología que se utilizó para realizar el pre-procesamiento de los datos consta de 5 pasos que se pueden apreciar en la figura 1.

Stage	Example
Raw text	Comfortable places to stay
Case folding	comfortable places to stay
Word tokenizing	“comfortable”, “places”, “to”, “stay”
Filtering	“comfortable”, “places”, “stay”
Stemming	“comfortable”, “place”, “stay”

Fig. 1. Metodología empleada para el tratamiento de los datos

En la figura 1, se puede apreciar el primer paso que es Raw text, este paso consiste en la recopilación de toda la data que son de 1187 reseñas en total y almacenarlas en un vector para su posterior limpieza reseña por reseña. El paso 2 es conocido como Case Folding, este paso consiste en la eliminación de cualquier caractér especial que no sea letra y la conversión de la oración ya limpia a minúsculas. El tercer paso de la figura 1, es la word tokenizing. Este caso consiste en realizarle una partición a la reseña por palabras de tal forma que la oración quede dividida. El cuarto paso consiste llamado Filtering consiste en la eliminación de Stop Words (palabras que no tienen significado propio como ”the”, ”of”, ”to”) con el objetivo de que quede un vector con las palabras que tienen un significado por si solas como los sustantivos y verbos. El último paso es conocido como Steamming, en este paso lo que se busca es reemplazar cada palabra de la reseña por su palabra raíz dado que existen palabras que son muy similares entre sí que en el fondo expresan lo mismo [1]. Luego de haber realizado la respetiva limpieza a los datos, se utilizó bigram y trigrams con el objetivo de encontrar dos palabras y tres palabras que se repiten más dentro del documento. El paso final antes de realizar el modelo de LDA, consiste en convertir las palabras en una matriz numérica que será el input para el modelo de detección de tópicos.

Para la realización del algoritmo LDA cuya representación se encuentra en la figura 2, es necesario definir 3 valores. De estos 3 valores, 2 de ellos son los hiperparámetros alpha y etha y el último viene a ser la cantidad de tópicos que uno desea extraer del texto [1]. Por otro lado, los valores de N y M son la cantidad de palabras únicas en el documento y la cantidad total de documentos respectivamente. Además, W(m,n) representa cualquier palabra n dentro del documento M y Z(m,n) representa la asignación del tema o tópico por cada palabra en el texto. Finalmente, βk y θd representan las distribuciones sobre las palabras de cada tópico y la distribución por tema presente en el documento [1].

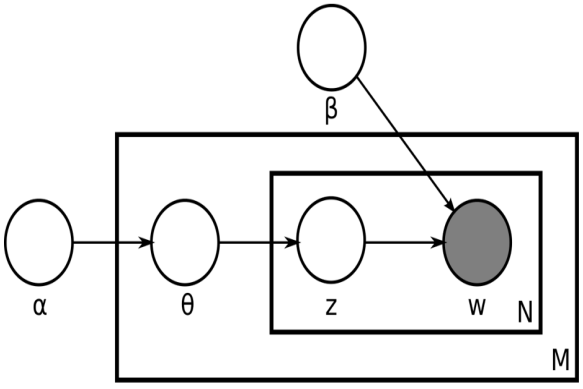

Fig. 2. Latent Dirichlet allocation (LDA) para detección de tópicos

Para saber cuál es la cantidad de tópicos óptima para el modelo se utilizaron dos métricas que son Perplexity y Coherence Score. El criterio para escoger la cantidad de tópicos en función a la primera métrica es que la cantidad de tópicos será mejor entre más bajo sea el valor de la Perplexity [1]. Los resultados usando la métrica de Perplexity se presentan en la figura 3.

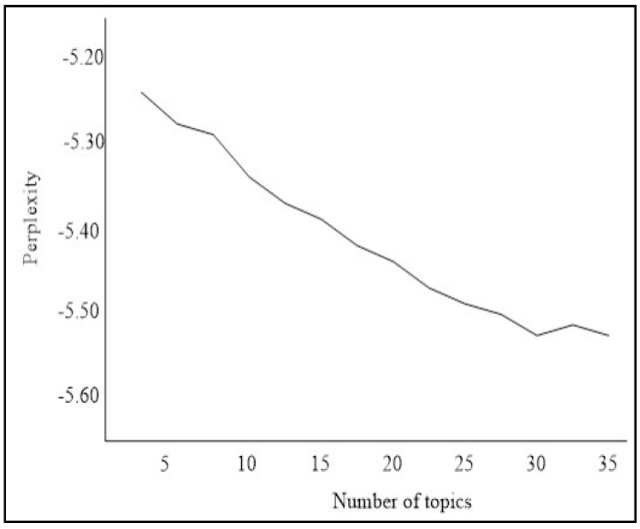

Fig. 3. Perplexity graph.

En la figura 3, se puede observar que la cantidad de tópicos optima es alrededor de 35 dado que devuelve un Perplexity de -5.50. No obstante. el tener tanta cantidad de tópicos puede disminuir la calidad de la clasificación dado que en un mismo tópico o en más de uno pueden haber palabras que se repiten [1]. Por lo tanto, se utilizó una segunda métrica conocida como Coherence Score. El criterio para determinar la cantidad de tópicos en función a esta es que entre mayor sea el Coherense Score, mejor será el número de tópicos ligado a esta.

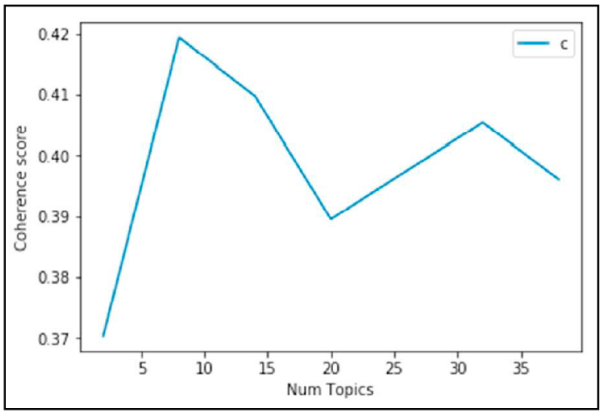

Fig. 4. Coherence score graph.

Como se puede apreciar en la figura 4 , el valor más alto para el Coherence Score es para una cantidad de tópicos igual a 8 dado que después de este número, el Score empieza a disminuir. Por lo tanto, la cantidad de tópicos que se utilizó fueron 8. Cada tópico es una colección de 10 palabras ordenadas por probabilidad de mayor a menor [1]. Los 8 tópicos resultantes fueron: Satisfacción con la habitación, Facilidad, Desayuno, Ubicación del hotel a la playa, Experiencia del turista acerca del hotel, Atmósfera, hopitalidad y baño. Los resultados de este trabajo sirve como retroalimentación para el propio hotel. Con estos 8 temas que son los que más discuten los turistas, se pueden crear estrategias para mejorar áreas o servicios específicos que ofrece el propio hotel con el objetivo de incrementar el número de turistas y clientes en general.

B. Sub-story detection in twitter with Hierarchical Dirichlet Processes

El problema principal en este artículo es sobre detectar sub-historias asociadas a una historia, tópico o evento principal. Así, hacen uso de los procesos jerárquicos de Dirichlet o HDP (modelo probabilístico) como un método para la detección automática de sub-historias. [2]

Un ejemplo para entender mejor la diferencia entre detección de tópicos y de sub-tópicos sería tener como tópico o evento los disturbios de Ferguson en 2014 que se basaron en manifestaciones luego del asesinato de un hombre a manos de la policía y como sub-tópicos o historia secundarias están los mostrados en la Figura 5 los cuales hablan sobre ciertas acciones de la policía, comentarios generales de las personas y también sobre rumores del hecho ocurrido(como que la víctima fue parte de un robo). En este sentido, la detección de tópicos en sí de esta información no produce buenos resultados, debido a que el vocabulario es compartido entre todas las historias secundarias. [2]

Description of major stories in the Ferguson data set.

Sub-story id	Description
1	M. Brown was involved in a robbery before being shot
2	Ferguson police are leading a smear campaign or character assassination of M. Brown
3	Initial contact between police officer and M. Brown was not related to the robbery
4	Ferguson police to release name of police officer who shot M. Brown today
5	Ferguson police are lying about the circumstances leading up to M. Brown's death
6	Ferguson police once beat up a man and charged him for bleeding on their uniforms
7	M. Brown was stopped by police for walking in the middle of the street
8	Fox News is not covering the Ferguson protests

Fig. 5. Sub-tópicos de los disturbios de Ferguson

Las historias secundarias tienen como característica principal la superposición en el tiempo, esto es porque hay muchas conversaciones que van evolucionando durante el tiempo. En este sentido, pueden tanto ser un tema latente de forma temporal y luego convertirse en un tema principal. [2]

En este estudio, utilizaron 5 bases de datos, de las cuales 3 fueron usadas principalmente para los experimentos de detección de sub-historias: Disturbios en Ferguson, Tiroteo en Ottawa, y Disturbios en Inglaterra. Las otras dos bases tomadas en cuenta fueron datasets públicas creadas específicamente para la detección de historias que fueron usadas con fines comparativos. A continuación pasamos a describir brevemente cada base de datos.

El primer dataset (Ferguson) consistía en tweets recolectados entre agosto y septiembre de 2014 relacionados al disturbio ocurrido en Ferguson, estos fueron clasificados manualmente por periodistas terminando en 45 historias secundarias diferentes. Luego de quitar sub-historias con menos de 10 tweets la base de datos consistió finalmente en 6598 tweets etiquetados y distribuidos en 35 sub-historias. La segunda base (Ottawa) consistió en 6414 tweets relacionados al titoreo en el parlamenteo canadiense ocurrido en octubre del año 2014 y distribuidos en 39 sub-historias. La última base usada para experimentos (Disturbios de Londres) consistió en 2,5 millones de tweets relacionados al evento que tuvo lugar en agosto de 2011, de estos habían 10 mil tweets etiquetados pertenecientes a 7 sub-historias. A parte de esas tres bases, usaron las de ”First story detection” que tenía 2400 tweets etiquetados y distribuidos en 27 sub-historias pertenecientes al periodo de junio-septiembre del año 2011, mientras que en la de ”Copa FA”(Football Association) habían 7 mil tweets asociadas a una partidos de fútbol y separados en 13 sub-historias. [2] Hemos de recalcar que cada dataset contenía tweets completos junto con los que seguían la conversación del tweet original, es por este motivo que los autores realizan dos tipos de experimentos: con todos los tweets, y solamente con los tweets originales (llamados también tweets de origen)

En su metodología hicieron un preprocesamiento de los tweets para eliminar caracteres inusuales, menciones de usuarios, hashtags, URLs y palabras de detención, así mismo realizaron stemming. Luego experimentaron con Procesos de

Dirichlet jerárquicos (HDP) para modelar eficazmente la detección de sub-historias, así mismo, compararon sus resultados con otros dos modelos: agrupación espectral (SC) y hashing sensible a la localidad (LSH). [2]

Lo que hace HDP es detectar palabras comunes para cada tweet, luego calcula la puntuación de similitud con un subtema a través de la suma de las probabilidades asociadas a las palabras. Así, el tweet se asigna a un subtema con el cual tenga el mayor puntaje de similitud. En el caso de Agrupamiento espectral, se basa en la idea de que si dos palabras aparecen consistentemente en el mismo tweet entonces estas pertenecen a un mismo tema, mientras que en LSH, crea cubos (depósitos de tweets similares), luego asigna tweets entrantes a ciertos cubos (disminuyendo el espacio de búsqueda) en los que se temina buscando a los tweets vecinos más cercanos en un tiempo constante y finalmente se asigna al grupo si la similitud del coseno es mayor que el umbral predefinido. [2]

Para evaluar los resultados de sus experimentos, hicieron uso de las métricas de precisión (micro), recuento(micro), medida F(micro), e información mutua ajustada. Se usa los micro-promedio debido a los diferentes tamaños de cada subtema.Los resultados se pueden observar en las figuras 6 y 7 y de estos se puede concluir que HDP es el método con mejor rendimiento (medida F) en la comparación de modelos para las dos primeras bases de datos (Ferguson y Ottawa), mientras que LSH funcionar mejor con respecto a la precisión del modelo (esto debido a que crea agrupaciones de pequeñas cantidades de tweets),y SC tiene el mejor recuento (en este caso SC no es capaz de diferenciar dos sub-historias similares y por ende los agrupa en un pequeño grupo creando varios de estos, generando el aumento de recuento y la disminución de su precisión). [2] Los resultados usando todos los tweets se pueden observar en las figuras 6 y 7

Method	Ferguson			Ottawa		
	P_{micro}	R_{micro}	F_{micro}	P_{micro}	R_{micro}	F_{micro}
HDP (k200)	0.0536	0.0889	0.0668	0.1799	0.1431	**0.1594**
HDP (k300)	0.1366	0.1057	**0.1191**	0.2182	0.1249	0.1588
SC (k400)	0.0131	**0.1622**	0.0242	0.0519	**0.1821**	0.0807
SC (k2000)	0.0422	0.0861	0.0566	0.0873	0.1244	0.1025
LSH (k12h56b10)	**0.3441**	0.0301	0.0554	**0.4797**	0.0314	0.0589
LSH (k13h71b10)	0.3430	0.0407	0.0728	0.3768	0.0285	0.0529

Fig. 6. Resultados de evaluación para las datasets de Ferguson y Ottawa

Method	London riots		
	P_{micro}	R_{micro}	F_{micro}
HDP (k50)	0.4188	**0.2759**	**0.3326**
HDP (k100)	0.4194	0.2013	0.2720
SC (k50)	0.1833	0.2666	0.2172
SC (k100)	0.4522	0.2539	0.3252
LSH (k12h56b10)	**0.5948**	0.2258	0.3273
LSH (k13h71b10)	0.4976	0.2323	0.3167

Fig. 7. Resultados de evaluación para el dataset de Londres

Así mismo, los resultados usando solamente los tweets de origen se pueden observar en la Figura 8, usando los tweets de conversación en la Figura 9, y usando los temas más destacados en la Figura 10

	Ferguson			Ottawa		
	P_{micro}	R_{micro}	F_{micro}	P_{micro}	R_{micro}	F_{micro}
HDP (k200)	0.2578	0.3042	0.2790	0.4615	0.4172	0.4382
HDP (k300)	0.3482	0.3688	**0.3582**	0.4212	**0.5248**	**0.4673**
SC (k400)	0.0508	**0.3691**	0.0893	0.1574	0.3055	0.2077
SC (k2000)	0.1614	0.2617	0.1996	0.1373	0.3055	0.1894
LSH (k12 h56 b10)	**0.6083**	0.2402	0.3444	0.6417	0.2603	0.3703
LSH (k13 h71 b10)	0.5591	0.2508	0.3462	**0.6975**	0.2451	0.3627

Fig. 8. Resultados de evaluación para las datasets de Ferguson y Ottawa usando los tweets de origen

Method	Ferguson			Ottawa		
	P_{micro}	R_{micro}	F_{micro}	P_{micro}	R_{micro}	F_{micro}
HDP (k200)	0.273	0.3674	0.3132	0.4749	0.4612	0.4679
HDP (k300)	0.3822	**0.4199**	**0.4001**	0.4398	**0.5691**	**0.4968**
SC (k400)	0.0722	0.4091	0.1227	0.1786	0.3581	0.2383
SC (k2000)	0.2149	0.3034	0.2515	0.1588	0.3143	0.2109
LSH (k12 h56 b10)	**0.5589**	0.3087	0.3977	0.5428	0.3038	0.3895
LSH (k13 h71 b10)	0.5079	0.3106	0.3854	**0.7777**	0.2877	0.4200
Baseline	1.0	0.2545	0.4057	1.0	0.1696	0.2900

Fig. 9. Resultados de evaluación para las datasets de Ferguson y Ottawa usando los tweets de conversación

Method	Sub-story 1			Sub-story 2			Sub-story 3		
	P	R	F	P	R	F	P	R	F
Ferguson									
HDP (k300)	0.96	**0.26**	**0.41**	0.45	**0.40**	**0.42**	**0.59**	**0.74**	**0.66**
SC (k400)	0.32	**0.26**	0.28	0.16	0.19	0.17	0.23	0.29	0.26
LSH (k12h56b10)	**0.99**	0.16	0.28	**1**	0.17	0.29	0.31	0.26	0.28
Ottawa									
HDP (k300)	**0.99**	**0.65**	**0.78**	0.82	**0.66**	**0.73**	**0.46**	**0.47**	**0.46**
SC (k400)	0.95	0.27	0.42	0.15	0.31	0.20	0.11	0.28	0.16
LSH (k13h71b10)	**0.99**	0.44	0.61	**1**	0.15	0.26	0.4	0.16	0.23

Fig. 10. Resultados de evaluación para las datasets de Ferguson y Ottawa usando las historias más destacadas

Para finalizar sus experimentos, también realizaron las evaluaciones para los datasets de FSD y Copa FA, los cuales son mostrados en las figuras 11 y 12

	Ferguson			Ottawa		
	P_{micro}	R_{micro}	F_{micro}	P_{micro}	R_{micro}	F_{micro}
HDP (k200)	0.2578	0.3042	0.2790	0.4615	0.4172	0.4382
HDP (k300)	0.3482	0.3688	**0.3582**	0.4212	**0.5248**	**0.4673**
SC (k400)	0.0508	**0.3691**	0.0893	0.1574	0.3055	0.2077
SC (k2000)	0.1614	0.2617	0.1996	0.1373	0.3055	0.1894
LSH (k12 h56 b10)	**0.6083**	0.2402	0.3444	0.6417	0.2603	0.3703
LSH (k13 h71 b10)	0.5591	0.2508	0.3462	**0.6975**	0.2451	0.3627

Fig. 11. Resultados de evaluación para el dataset de FSD

Method	London riots		
	P_{micro}	R_{micro}	F_{micro}
HDP (k50)	0.4188	**0.2759**	**0.3326**
HDP (k100)	0.4194	0.2013	0.2720
SC (k50)	0.1833	0.2666	0.2172
SC (k100)	0.4522	0.2539	0.3252
LSH (k12h56b10)	**0.5948**	0.2258	0.3273
LSH (k13h71b10)	0.4976	0.2323	0.3167

Fig. 12. Resultados de evaluación para el dataset de Copa FA

Así mismo, como se mencionó previamente también utilizarón la métrica de información mutua ajustada todos los datasets, los resultados para esta métrica se observan en las figuras 13 y 14, en estas se demuestra que HDP se desempeñó mucho mejor que el resto de modelos, por lo tanto, es un modelo muy útil para detección de temas secundarios. [2]

Method	Ferguson	Ottawa	FSD	FAcup
HDP (k100)	0.46	0.59	**0.70**	0.10
HDP (k200)	0.46	0.55	0.67	**0.11**
HDP (k300)	**0.47**	**0.60**	0.65	0.10
SC (k200)	0.40	0.39	0.65	0.07
SC (k400)	0.38	0.43	0.45	0.07
SC (k2000)	0.39	0.42	0.31	0.08
LSH (k12 h56 b10)	0.40	0.46	0.23	0.08
LSH (k13 h71 b10)	0.40	0.47	0.24	0.09

Fig. 13. Resultados de evaluación para Fergurson, Ottawa, FSD y Copa FA

Method	LondonRiots
HDP (k25)	**0.32**
HDP (k50)	0.31
HDP (k100)	0.29
SC (k50)	0.31
SC (k100)	0.31
SC (k200)	0.28
LSH (k12 h56 b10)	0.29
LSH (k13 h71 b10)	0.30

Fig. 14. Resultados de evaluación para data de Disturbios de Londres

Para concluir HDP resulta muy útil para realizar un seguimiento de la evolución de los temas secundarios con respecto a un tema principal. Además, los autores en [2] consideran que la estructura de las conversaciones de Twitter es la que ha ayudado a que su modelo rastree mejor los tópicos secundarios, lo cual es necesario para detectar rumores tempranamente.

C. Big data analytics and international negotiations: Sentiment analysis of Brexit negotiating outcomes

Este trabajo consiste en la aplicación de Analisis de Sentimiento (SA) a fin de saber las opiniones y sentimientos de los usuarios mediante la red social Twitter para el caso de las negociaciones de Brexit entre el Reino Unido y la UE. [3] Si bien es cierto, la desvinculación total del Reino Unido de la Union Europea, o también conocida como brexit, oficialmente fue en 31 de enero 2020, las discussiones sobre este caso se llevaron a cabo más de 4 años. Este proceso inició tras un referendum en el 23 de junio del año 2016 donde el 51,9 por ciento de la población apoyó a favor de la salida de Reino Unido de la UE. Este trabajo hace referencia en que las encuestas en los días previos a la votación del referendum consideraba la votación ”Remain” con una ventaja sin tener consideración los ultimos cambios en la opinión pública. [4] De esta manera este trabajo resalta que sería una fuente poderosa el Twitter para poder proyectar el resultad. De hecho, los estudios posteriores demostraron que este medio había indicado el voto de ”Salida”. [5] [6]

En base a esto, el estudio se realizó en el intervalo de 5 de Mayo hasta 7 de Noviembre del año 2018 (es considerado como los períodos más activos de las negociaciones del Brexit.), para visualizar toda la interacción y/o reacción del público ante esta situación haciendo el uso de la plataforma Twitter. [3] Para la implementación se utilizó un enfoque basado en léxico [8] la cual analiza los textos y retorna 4 (3+1) scores de polaridad:

- Score positiva para la polaridad positiva
- Score negativa para la polaridad negativa
- Score neutro que representa la neutralidad
- Y el score compuesta indica la combinación de todo los scores anteriores

En pocas palabras, las puntuaciones positivas, negativas y neutrales representan la proporción de texto que cae en estas categorías, mientras que la puntuación compuesta calcula la suma de todas las clasificaciones de léxico, que se han normalizado entre -1 y +1.

Adicionalmente hacen uso de la herramienta ”Valence Aware Dictionary for Sentiment Reasoning” (VADER) la cual es una herramienta de análisis de sentimientos basada en reglas y léxico que está relacionado con los sentimientos expresados en las redes sociales y funciona bien en textos de otros dominios. El autor indica que esta herramienta genera más valor ,cuando se trabaja con los tweets, en comparación a otros herramientas de analisis de sensibilidad porque tambien detecta los sentimientos de los emojis y las jergas.

El estudio se basa en 2 tipos de fuente de datos:

- Los momentos claves donde ocurrió un evento con relación Brexit
- 13,018,367 tweets que se publicaron durante este mismo periodo.

Se utilizó el Twitter Streaming Api para generar este dataset trabajado en el estudio. A continuación se muestra la tabla de relación de los hastags y la cantidad de los tweets recolectados.

Brexit negotiations' outcome hashtags.

Brexit negotiations' outcome hashtags	Volume of tweets	Brexit negotiations' outcome hashtags	Volume of tweets
#Brexit	1.243.972	*#SingleMarket*	8,311
#HardBrexit	6.038	*#SecondReferendum*	4.283
#SoftBrexit	398	*#NoDeal*	24.502
#StopBrexit	158.081	*#CustomsUnion*	6.727

Fig. 15. Los hashtags del resultado de las negociaciones del brexit.

Para la captura y analisis en tiempo real del sentimiento se utilizó código de Python la cual se conectó a Twitter API Streaming. Solo fueron consideradas los tweets en Ingles que contengan la palabra ”Brexit” y/o los hastaghs mencionados en la fig 15.

Luego podemos visualizar tambien la fig 16 donde nos indica los acontesimientos importantes y sus analisis de sentimiento por cada hastag elegido anteriormente.

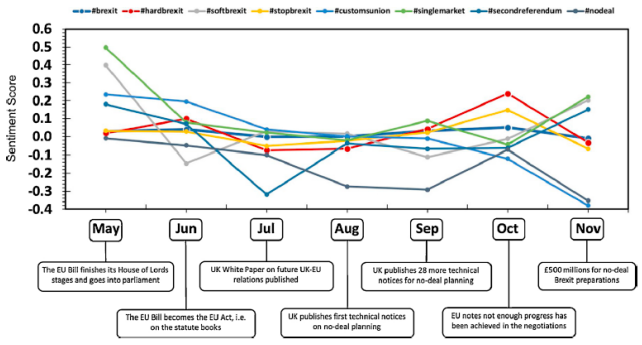

Fig. 16. Grafica de hashtags de sentimiento semanal sobre las negociaciones Brexit .

En conclusion, los autores recalcan que el analisis de sentimiento junto a BDA tienen el potencial de mejorar la toma de decisiones y las estrategias hasta en una sitaucion tan compleja como el Brexit. Adicionalmente, indican que los resultados de Brexit preferidos o menos preferidos podrían haber sidos inferidos por las emociones positivas o negativas expresadas por los usuarios.

D. Pre-trained Contextualized Representation for Chinese Conversation Topic Classification

En esta investigación se propone una arquitectura de red neuronal basada en un modelo BERT pre-entrenado que se utiliza para construir multiples modelos de redes neuronales para la clasificación de temas de conversación en chino. [9] Para ello se utilizó una dataset de 5 clases conversacionales de chino, recopilada desde varias paginas web de aprendizaje de ingles en linea. Este dataset consiste en 1994 conversaciones y 13590 oraciones que contiene 5 temas, los cuales son:

- banco
- dieta
- sentimiento
- compras
- viajes

Durante el proceso esta data se separó en datasets de Train, Dev y Test con un ratio 3:1:1. Se aplicó el modelo BERT pre-entrenada como linea de base para clasificación de topicos. Para ello se concateno los enunciados en una conversación en un texto largo y se incorporó al modelo BERTbase. Para esta investigación se encontró valores optimos de los hiperparametros. Estos son:

- Batch size: 8
- Learning rate (Adam): 2e-5
- Number of training epochs: 100
- Max sentence length: 512

TABLE I
THE HYPERPARAMETERS OF OUR MODELS IN THE ARCHITECTURE

Parameters	Choice	Experimental Range
Max utterances per Conversation	10	10, 15, 20
Max length per Sentence	140	100, 140, 200, 600
Max sequence length of BERT	140	140, 200, 512
Dropout rate	0.4	0.4, 0.5
Epoch size	5	5, 10
Mini-batch size	32	8, 16, 32, 64

Fig. 17. Hiperparametros del modelo en la architectura

En la fig 17 se demuestra en detelle los ajustes hechos en el hiperparametro del modelo en el conjunto de datos de validación. Para este caso se aplicó 20 epochs y se estableció un tasa de abandono en 0.4 para obtener un mejor rendimiento.

El proceso de entrenamiento consiste en 2 fases. La primera fase es adaptar el modelo BERTbase chino pre-entrenado para la clasificación de temas con textos de conversación etiquetados. La siguiente etapa es aprender un clasificador avanzado basado en el modelo BERT. Una conversación tiene dos o más oraciones. Si existiera dos personas conversando sobre diferentes temas la conversación se expresaría de la siguiente forma:
SL = {As1,Bs1 , ...,Asm,Bsn}
donde:

- A,B : dos personas hablando
- m,n : # de oraciones pronunciados por las personas A y BERT
- L : # total de enunciados

Se realiza metodos de aprendizaje supervisado para clasificar cada conversacion en chino.

- ci : tema predefinida para conversacion
- k : # de temas
- C = {c1, c2, ..., ck}

El objetivo de esta investigación es utilizar la arquitectura de red neronal para entrenar, evaluar y predecir en la base de datos. Esta arquitectura se puede visualizar en el imagen.

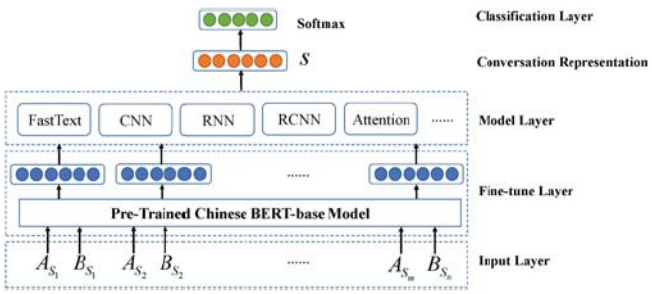

Fig. 18. Arquitectura de red neuronal basada en una representación BERT

Como se puede visualizar en la fig 18, la arquitectura se descompone en las siguientes capas:

- Input layer:
  En esta investigacion existen 2 formas de entrada. Uno es mediante concatenacion de los enunciados como un ”long input” y secuencia de los enunciados.
- Fine-tune layer:
  Para la entrada se le aplica el modelo Bert pre entrenado para adaptar los parametros de clasificacion de topicos supervisados.
- Model layer:
  Para el modelamiento se utilizo varios modelos como FastText,CNN,RNN, RCNN, RCNN, BLSTM
- Clasification layer:
  La capa de modelamiento como output da las conversaciones, tal como figura en la fig 18, Se agrega un nuevo parametro, W, la cual es un output linear para predecir la etiqueta y. La capa linear transforma la conversación S en un vector cuya dimensión es el número de etiqueta. Las probabilidades de etiqueta se calculan con un softmax estándar de la siguiente manera:

$$p(y||x1, x2, ..., xm) = softmax(CwT)$$

TABLE II
RESULTS IN PERCENT OF WEIGHTED-AVERAGE PRECISION (P), RECALL (R) AND F_1-SCORE (F_1)

Methods	P	R	F_1
Baseline ($BERT_{base}$)	90.4	89.2	89.4
FastText + BERT	90.5	89.7	89.9
CNN + BERT	90.5	89.7	89.9
LSTM + BERT	91.0	90.0	90.1
BLSTM + BERT	91.1	90.2	90.4
RCNN + BERT	91.2	90.7	90.8
BLSTM-Attention-single + BERT	91.1	90.2	90.4
BLSTM-Attention + BERT	**91.5**	**91.5**	**91.5**

Fig. 19. Resultados en porcentaje de Precision(P), Recall(R) y F1 Score (F1)

La fig 19. nos muestra el rendimiento de metodos y los modelos que se evalua mediante el precision, recall y F1-Score. Los modelos procesados en esta investigacion utilizaron el fine-tuned BERT para representar la conversión china. Mientras que el modelo BERT pre-entrenado para clasificacion de temas. Esta tabla muestra que todos los modelos son efectivos para la clasificación de temas. Se puede visualizar que el modelo de linea base tiene un buen rendimiento; sin embargo, todos los modelos utilizados superan a está.

TABLE III
F_1-SCORE (PRECISION, RECALL) BY PERCENT OF EACH INDIVIDUAL TOPIC

No.	Topic	Test size	RCNN + BERT	BLSTM-Attention + BERT
1	bank	62	**99.2 (98.4,1.00)**	98.4 (96.9, 1.00)
2	diet	93	91.6 (95.3, 88.2)	**93.5 (94.5, 92.5)**
3	sentiment	72	84.1 (77.6, 91.7)	**87.1 (85.3, 88.9)**
4	shopping	86	**90.5 (92.7, 88.4)**	89.9 (91.6, 88.4)
5	travel	86	**89.9 (91.6, 88.4)**	89.5 (89.5, 89.5)

Fig. 20. F1 Scores (F1) por cada topico

La fig 19. muestra que los modelos RCNN y BLSTM están clasificados entre los dos primeros modelos. Estos dos fueron comparados, como se puede visualizar en la fig 20., sus resultados a clasificación en cada tema en particular. Para temas de bancos, compras y viajes el modelo RCNN + BERT logra un mejor rendimiento que el BLSTM-Attention + BERT. Por otro lado, este supera al primero en temas de dieta y sentimientos.

Los autores concluyen que a pesar de que el conjunto de datos procesados no es muy grande, los modelos logran un buen rendimiento en el conjunto de datos. Resaltan que el modelo BERT pre-entrenado es un enfoque de aprendizaje de transferencia efectivo que puede utilizarse para entrenar un modelo para una tarea específica con un numero no muy grande de datos etiquetados.

III. METODOLOGÍA

En esta sección del presente documento, se detallará la metodología a utilizar para poder detectar el tópico al cual pertenece cada tweet y el sentimiento que posee. La metodología general para realizar el proyecto, se encuentra en la Figura 21 y en la Figura 22 presentes a continuación.

A. Extracción de la data

Para la extracción de la data se necesitó contar con una cuenta de desarrollador para usar la API de twitter. Una vez creada la cuenta, se procedió a crear una nueva Twitter Application. Al momento de la creación de nuestra nueva app, se nos otorga principalmente valores para 4 principales campos que son el consumer key, cosumer secret key, access token y el access token secret. Estos 4 campos con sus respectivos valores, servirán de autentificación para poder conectarnos a twitter y poder extraer los tweets más adelante. Posteriormente, se procedió a instalar R junto con R studio versión 1.2.1335 para poder realizar la extracción de tweets. Las librerías que se utilizaron fueron ”twitterR” para la extracción de los datos , ”rtweet” para activar la extracción por zonas específicas y ”readr” para la exportación e importacion de archivos. Se procedió a importar las 3 librerías y se proceció a crear 4 variables llamadas api_key,api_secret_key,access_token y access_token_secret en donde cada una almacenaba el valor proporcionado por la aplicación de twitter cuando se creó. Luego, para la autentificación, se utilizó la función setup_twitter_oauth() proporcinada por la librería ”twitterR” y dentro de esta función se agregaron las 4 variables mensionadas. Para la extracción de los tweets, se creó una variable llamada ”tweets” que almacenaria todos los tweets que se iban extrayendo. La función searchTwitter, permitió hacer la conexión por twitter para realizar la extracción. Además, hubieron otros argumentos que intervinieron en la función de extracción.

El primer argumento para la extracción es la propia palabra que se desea extraer, como segundo argumento esta la función lookup_coords en donde dentro de esta función proporcionada por la librería ”rtweet” se introducía el pais del cual se deseaba hacer la extracción que en este caso es solamente de Perú.

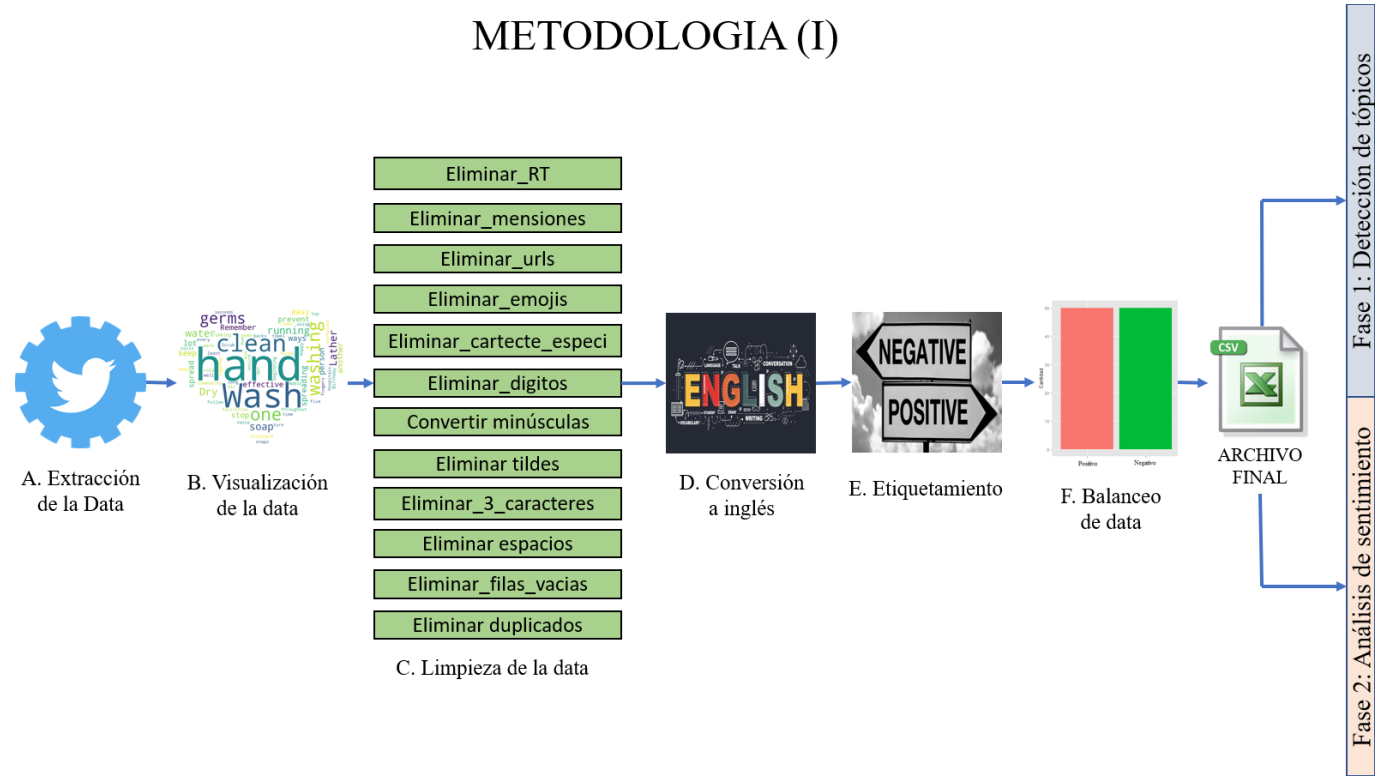

Fig. 21. Metodología del presente trabajo de investigación (I)

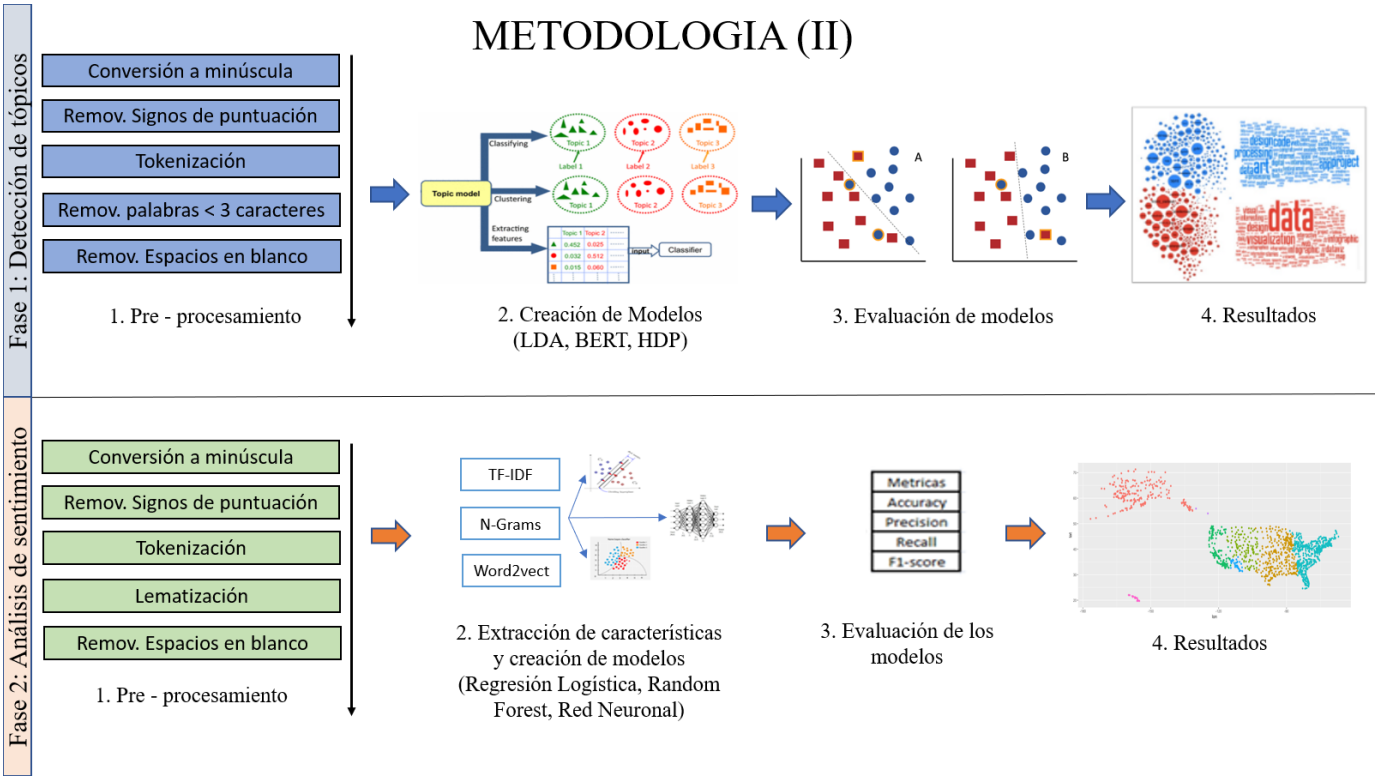

Fig. 22. Metodología del presente trabajo de investigación (II)

El siguiente argumento es el n que viene a ser la cantidad de tweets que se desea extraer. Seguidamente, se encuentra el argumento ”lang” que es el idioma con el cual se desea que tengan los tweets. En este caso esta variable lleva el valor de ”es” porque los tweets serán extraidos en el idioma español.

Finalmente están los parámetros de ”since” y ”until” en donde se introduce desde qué fecha a qué fecha se desea hacer la extracción. Es preciso mencionar que la API de twitter permite hacer la extracción solamente de los 7 últimos días, por lo que para reunir una cantidad considerable en un periodo considerable de tiempo 2 integrantes extrajeron tweets de 20 000 en 20 000 y un último integrante de 50 000 en 50 000. Para todas las extracciones, se utilizaron palabras referentes al coronavirus en el Perú como ”Covid-19”, ”infectados”, ”vacuna”, ”pandemia”, entre otros.

Una vez extraidos los tweets, se procedió a convertir los resultados en un dataframe con la función twListToDF y almacenado en una nueva variable.

Finalmente, con la función write_csv proporcionada por la librería ”readr”, se exportó el dataframe a un archivo csv cuyo nombre del archivo era el nombre de la palabra con el cual se extrajo junto con la fecha de extracción para un mejor manero de los archivos. Al final todos los archivos csv que se recopilaron de manera individual por cada integrante, se juntaron para formar un solo archivo y así trabajar de una mejor manera. En total, se recopilaron 3 043 171 tweets por un periodo de 7 semanas usando las diversas palabras relacionadas al coronavirus mensionadas anteriormente.

B. Exploración de la data

En este caso, decidimos generar un pie usando la librería matplotlib para chequear cuántos de los tweets que se extrajeron son Retweets, esta información se resume en forma porcentual en la Figura 23

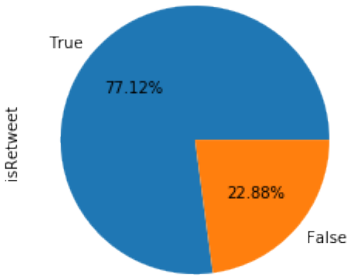

Fig. 23. Visualización de recuento de Retweets en valores porcentuales

C. Limpieza de la data

Para la limpieza de los datos se utilizaron las librerías re (expresiones regulares) y unidecode y demoji. A continuación detallaremos las funciones utilizadas para la limpieza de los datos.

- Eliminación de retweets: En los tweets se eliminó el prefijo ”RT” de todos los tweets que lo tenían, esto se debió a que podría existir la posibilidad de haber extraído retweets pero no su tweet original, por ese motivo nos quedamos con el resto del texto y en caso de estar realmente repetido sería eliminado posteriormente en eliminación de duplicados.
- Eliminación de menciones: En los tweets descargados se tienen menciones iniciando con , este tipo de texto no era necesario para las tareas a realizar, por ende, usando la librería re se terminaron eliminando de los tweets.
- Eliminación de URLs:se usó la librería de expresiones regulares para eliminar los hipervínculos dado que no eran relevantes para este estudio
- Elimación de emojis: para esta función se hizo uso de la librería demoji, la cual requiere de una descarga inicial de datos del repositorio de códigos emoji del Consorcio Unicode, además de la librería re y su módulo replace para hacer reemplazar el emoji a nulo (vacío).
- Eliminación de caracteres especiales: en esta función también se utilizó la librería re para quitar todos los caracteres dado que estos eran completamente irrelevantes para este trabajo
- Eliminación de dígitos: en esta parte, se eliminaron los números o cifras encontradas en cualquier parte del texto a través de la librería re.
- Conversión a minúsculas: dado que pueden existir palabras iguales pero escritas en mayúsculas y minúsculas en toda la data, se decidió homogenizar las palabras.
- Eliminación de tildes: haciendo uso de la librería unidecode se pasaron las vocales con tilde a vocales.
- Eliminación de palabras con menos de 3 caracteres: dado que hay palabras usadas comúnmente y que no son útiles para el texto y que no necesariamente son stop words se decidió eliminarlas con el fin de no obstaculizar la traducción del tweet.
- Eliminación de espacios vacíos: haciendo uso de ’strip’ se eliminaron los espacios vacíos al inicio y al final de los textos que fueron dejados luego de la eliminación de emojis.
- Eliminación de filas vacías: en esta función se tuvo en cuenta que pudieron existir ciertos traspases o errores al unir la data descargada en diferentes archivos y que por ende podrían existir filas sin información.
- Eliminación de duplicados: esta última función también permitió eliminar errores en la unificación de la información y los posibles retweets que existían como tal dentro de la data.

Todas las funciones mencionadas arriba fueron unificadas en una sola para aplicar la limpieza de forma automática a los datos.De esta forma, de los 3043171 de tweets que se tenían, terminaron quedándose 892734 tweets.

D. Conversión a inglés

Una vez que que se obtuvo el input del proceso de limpieza de la data, se procedió a realizar la traducción de los tweets al idioma ingles. Esta conversión es importante dado que en el siguiente proceso de etiqeutamiento el ”etiquetador de

sentimiento" sentiword solo trabaja con el idioma ingles. Para nuestro caso, en este proceso de conversión, se trabajó con un input de 892734 tweets, la cual fue partida en 3 diferentes grupos para poder trabajarlo, simultaneamente en distintas maquinas y de esta forma optimizar el tiempo de ejecución. Durante la ejecución se identificó la necesidad de hacer nuevamente una partición de estos 3 grupos y se volvieron a partir en 3 formando en total 9 distintos grupos. Los 6 de estos 9 grupos se trabajaron desde google colab mientras que los 3 grupos restantes se trabajó en maquina local desde anaconda. Para la conversión de los tweets al idioma ingles tanto en la maquina local como en google colab se utilizaron las librerias googletrans y Translator. Luego de importar el csv se creó una variable data_ingles donde se almacenó todos los tweets traducidas en el proceso. Para los grupos que se ejecutaron en google colab adicionalmente se trabajó con la libreria google colab. Para ejecutar códigos en este ambiente es necesario tener una cuenta de google y autenticarse mediante una contraseña de google. Una vez terminado el proceso, las bases traducidas se exportaron a csv para luego ser enviado al proceso de etiquetamiento. Dada las limitaciones del tiempo y caracteristicas/especificaciones de cada PC solo llegamos a procesar y tradujir una cantidad total de 549185 tweets la cual fue enviada al proceso de etiquetamiento.

E. Etiquetamiento

Para realizar el etiquetamiento de los tweets, se utilizó la libreria senti_classifier. Se importó el csv con los 549185 almacenados en una variable la cual se convirtió en una lista usando la funcion tolist(). Lo primero que se hizo fue obtener el score tanto positivo como negativo de un mismo tweet. La obtención de la polaridad, fue realizada gracias a la función polarity_scores() de la libreria senti_classifier. Finalmente, en una lista vacía llamada "data", se almacenaron cada tweet con su respectiva polaridad positiva y negativa. Luego, se creó un nuevo ciclo for para recorrer cada elemento de la lista "data" y poder clasificar un tweet como "Positivo" o "Negativo". El criterio para decidir si un tweet es positivo o negativo, fue si el score_positivo es mayor que el score_negativo de un determinado tweet, entonces el tweet es positivo. Por el contrario, si el score_negativo es mayor que el score_negativo, entonces el tweet es negativo. Este proceso se repitió por cada uno de los 549185 tweets. Se almacenó en una lista vacia cada tweet con su respectiva etiqueta. Finalmente, todos los tweets etiquetados se exportaron a un csv que contenia dos columnas: tweet y sentimiento para el siguiente paso que es el balanceo de tweets de tal forma que exista la misma cantidad de tweets positivos como negativos.

F. Balanceo de la data

Luego de tener la data etiquetada, se observó que existían 326261 tweets positivos y 222924 tweets negativos. Dado que esto es equivalente a un 59.1% y 40.59% de la data respectivamente, se decidió balancearla de tal forma que quedase la misma cantidad para ambas etiquetas. La diferencia en cantidad entre ambas etiquetas era de 103337 tweets, así que se decidió eliminar los primeros 103337 tweets que tengan como etiqueta al sentimiento positivo. Esto se hizo obteniendo una lista de con los primeros 103337 índices cuyo sentimiento es postivo, y pasándola como parámetro a la función 'drop'. Hemos de recalcar que no se hizo la eliminación de forma random dado que al realizar experimentos en diferentes fechas podrían modificarse los resultados y genera confusión. Finalmente se obtuvo la data balanceada con un total de 445848 tweets: 222924 tweets para cada sentimiento.

Para finalizar, se exportó los tweets etiquetados y balanceados a un archivo csv llamado DATA_MASTER_BALANCEADA con el cual se construirán tanto los modelos de detección de tópicos como los modelos de machine learning para realizar el análisis de sentimiento.

G. Fase 1: Detección de tópicos

En esta sección se procederá a explicar de manera detallada el procedimiento para crear los 3 modelos para realizar la detección de tópicos y cómo evaluar cada modelo para saber qué modelo es mejor y poder utilizar ese único modelo para realizar la detección de tópicos de nuevos tweets.

1) Modelo de detección de tópicos usando LDA: En este proyecto de investigación se realizarán dos versiones de LDA: Un modelo de LDA usando Bag of words tradicional (Doc2bow) y otro modelo de LDA usando TF-IDF. Al final se compararán los dos modelos de LDA para saber cuál es mejor y quedarnos solamente con uno.

1. Modelo de LDA usando Doc2bow: Para realizar el primer modelo de LDA usando Doc2bow, se cargó el csv que contenia los tweets que se iban a utilizar. Para limpiar los tweets antes de la creación del modelo, se realizó una conversión de todas las oraciones a minúsculas, se removieron todos los signos de puntuación, se tokenizacion las oraciones, se removieron palabras menores a 3 caractéres y se removieron los espacios en blanco de tal modo que quedó un corpus con el cual se puede crear el modelo de LDA. El siguiente paso fue la creación de un diciconario usando la función gensim.corpora.Dictionary. Esto permitirá colocarle un identificador a cada palabra dentro del corpus. Una vez creado el diccionario, el siguiente paso, el la creación de un bag of words (BOW) en donde las palabras en las oraciones se reemplazan con su identificación respectiva proporcionada por este diccionario. Para la creación del BOW tradicional, se utilizará la función Doc2bow por cada documento que se encuentre dentro del corpus. Una vez creado el BOW usando doc2bow, se procedió a crear el modelo de LDA usando la función gensim.models.LdaMulticore() proporcionado por la libreria gensim cuyos parámetros del modelo fueron bow_corpus el cual es la bolsa de palabras que ya se había creado, num_topics=10 que es número de tópicos que el modelo encontrará, id2word=dictionary que es el diccionario que se ha creado anteriormente, passes=2 el cual es número de veces que recorrerá el corpus durante el entrenamiento del modelo y workers=2 que es el número de workers que se utilizarán para el proceso de paralelización del entrenamiento. Para la evaluación del modelo, se utilizó las métricas de Perperxity y Coherence score. Para el cálculo de la primera

métrica se utilizó la función log_perplexity() dandole como parámetro el corpus creado. La perperxibilidad se analiza en función a la negatividad del número. Entre más negativo sea la perperxibilidad, el modelo es mejor. Por otro lado, para al cálculo del coherence score, se utilizó la función CoherenceModel() que recibe como parámetros el modelo construido, el corpus creado y el diccionario. Para analizar el Coherence score, se debe de analizar cuán cercano a 1 es el valor. Entre más cerca a uno sea el valor del coherence score del modelo, este será mejor. Para la obtención del valor óptimo de tópicos que debe de poseer el modelo, se creó una función llamada compute_coherence_values() que recibe como parámetros el diccionario creado, el corpus, los tweets procesados, un valor start = 2 que es el inicio del gráfico y un step = 2 que es de cuánto en cuánto incrementará el número de tópicos por modelo creado. Esta función permitirá retornar una lista de modelos de LDA creados con sus respectivos coherence score. La libreria matplotlib.pyplot permitirá visualizar de manera gráfica qué modelo de LDA usando BOW es el más óptimo.

2. Modelo de LDA usando TF-IDF: Para la creación del modelo de LDA usando TF-IDF, se utilizará la función models.TfidfModel() que recibe como parámetro el bag of word creado de tal manera que en vez de obtener un corpus en función a la frecuencia de palabras con sus identificadores, se obtienen una colección de pares ordenados cuyos valores son el id de la palabra y el peso correspondiente por cada palabra almacenado en una variable llamada tfidf. Para la creación del nuevo corpus se utilizará la nueva variable llamada tfidf a la cual se le da como parámetro el bow_corpus creado para el modelo anterior, de tal manera que se cree un nuevo corpus llamado corpus_tfidf. seguidamente, se procedió a crear el modelo de LDA usando la función gensim.models.LdaMulticore() proporcionado por la libreria gensim cuyos parámetros del modelo fueron corpus_tfidf el cual es la nueva bolsa de palabras creada, num_topics=10 que es número de tópicos que el modelo encontrará, id2word=dictionary que es el diccionario que se ha creado anteriormente, passes=2 el cual es número de veces que recorrerá el corpus durante el entrenamiento del modelo y workers=2 que es el número de workers que se utilizarán para el proceso de paralelización del entrenamiento. Para la visualización del modelo se utilizan las librerias pyLDAvis y pyLDAvis.gensim asi como matplotlib.pyplot. La función que se utilizará será pyLDAvis.gensim.prepare() que recibe como parámetro el modelo que se desea graficar, el corpus y el dicccionario. Todo almacenado en una variable llamada vis y para la visualización se utiliza la función pyLDAvis.display() que recibe como entrada la variable vis. Para evaluar al modelo de LDA con tf idf, se utilizarán las métricas de Perperxibilidad y Coherence Score descritos anteriormente. Para la obtención del valor óptimo de tópicos que debe de poseer el modelo,se utilió la misma función llamada compute_coherence_values() que recibe como parámetros el diccionario creado, el corpus, los tweets procesados, un valor start = 2 que es el inicio del gráfico y un step = 2 que es de cuánto en cuánto incrementará el número de tópicos por modelo creado. Esta función retorna una lista de modelos de LDA - TF-IDF creados con sus respectivos coherence score de tal modo que se podrá obtener el valor valor óptimo de K en función a la Coherence Score.

2) Modelo de detección de tópicos usando HDP: Se realizarán dos versiones de HDP: un modelo usando ngrams y otro usando TF-IDF. Al final los dos modelos serán comparadas para escoger el mejor de ambos. Para ambos modelos, se preprocesó el archivo csv con la data balanceada; es decir, se tokenizaron los textos haciendo uso de la librería nltk, se eliminaron los stop words -también usando nltk y su corpus de stopwords predefinidos- y palabras con menos de 3 caracteres, y finalmente se lematizó la data usando la librería spacy. Así mismo se importó la librería gensim junto a sus modelos y corpus para utilizar HdpModel correctamente en ambos modelos.

1. Modelo de HDP usando bag of words: Primero se hizo uso de la librería gensim y su módulo corpora, del cual se usó Dictionary para generar el diccionario con las palabras únicas de todo el documento, dándole como parámetro los datos preprocesados como lista de listas. Así mismo, se generó el corpus usando doc2bow y un loop for, ambos fueron datos retornados a través de una función (bag_words)que recibía como parámetro a los datos.

Finalmente se crea el modelo HDP con la función HdpModel de gensim, el cual requiere de dos parámetros principales: el corpus (matriz de vectores característicos) y el diccionario, datos que ya fueron determinados en la función anterior, así mismo se colocaron los parámetros T=10 (número de tópicos), K=1 (segundo nivel de número de tópicos) y un random_state =1 (semilla) para una prueba inicial y finalmente se hizo una muestra de los tópicos obtenidos con show_topics() y se realizó una prueba del modelo generado con un nuevo texto. Dado que se necesitaba conocer el mejor modelo, se creó una función (compte_coherence_values) en la cual entran como parámetros el diccionario, el corpus, los textos, inicio, límite y pasos (estos últimos tres son para generar la cantidad de tópicos en un rango determinado), con el fin de crear varios modelos que serían guardados en una lista junto a sus coherence_values (métrica con la cual se evalúa el modelo). De esta forma, se imprimen el número de tópicos, su valor para la métrica y la posición en la cual se encuentran los modelos usando una función (print_cv) y por otro lado, también se crea otra función que permite mostrar un gráfico de esos resultados. De esta forma, se genera una nueva variable para almacenar el modelo con mayor coherence y así, se imprimen los tópicos de este modelo para que finalmente se visualicen los resultados usando pyLDAvis.

2. Modelo de HDP usando TF-IDF: Para este modelo, se usará la librería TfidfModel de gensim, y Dictionary. Primero, se crea el diccionario (índice:palabra) de toda la data preprocesada, luego se crea el corpus usando ese diccionario creado y doc2bow, después se usa la librería TfidfModel para 'entrenar' el corpus y se aplica el modelo a todo el corpus. Finalmente se crea el modelo HDP con la función ya mencionada, ingresando como parámeotros el corpus y el diccionario creado por TF-IDF y también los valores de K, T

y random_state mencionados anteriormente.

Así como en el modelo con Bag of words, se busca el número óptimo de tópicos, de los cuales se queda el que tenga mayor coherence y al final este modelo pasa a una nueva variable que será usada para imprimir los tópicos y para generar el gráfico dinámico usando pyLDAvis.

3) Modelo de detección de tópicos usando LSI: Para realizar el modelo de LSI utilizando el doc2bow, primero se cargó la base limpia y balanceada al entorno. Para que se pueda trabajar con el modelo LSI, luego de haber cargado la base se realizó una limpieza adicional a la fase anterior (fase pre-procesamiento) la cual contenia pasar la data a minuscula, eliminación de las ,mayusculas y de las palabras con tamaños menores a 3 letras, tokenizacón y eliminacion de los espacios. Luego de haber validado la data limpia, se procesó a crear un diccionario mediante la función de gensim.corpora.dictionary al igual que el modelo LDA, se realizó la creación de los Bag of words utilizando la función Doc2bow. Una vez terminada se procesó a crear el modelo usando la función LsiModel de la libreria gensim models. Como parametro se procesó el dictionary creada anteriormente, el corpus luego de aplicar el BOW y numero de topicos. Para evaluar el modelo y poder hacer comparaciones entre los modelos anteriores (LDA y HDP) se utilizó la función Coherence Score y se realizó la iteración para poder encontrar el mejor modelo, la cual se detallará en este paper en la parte de resultados.

H. Fase 2: Análisis de sentimiento

En esta sección se procederá a explicar de manera detallada el procedimiento para crear los 3 modelos de machine learning que será un modelo de regresión logística, un modelo de RandomForest y una red neuronal por cada técnica de extracción de característica (Tf-idf, N-grams y Word2vect). Para la evalución de los modelos, se evaluarán por cada técnica de extracción de caracterísyicas hasta obtener el mejor modelo usando Tf-idf, N-grams y Word2vect. Finalmente, se evaluarán estos 3 mejores modelos para solamente obtener un solo modelo que vendría a ser el mejor. Este modelo servirá para la clasificación de tweets en positivos y negativos.

1) Creación de modelos de Machine learning: En primer lugar, se hizo una visualización de los tweets que se tenia usando la libreria Wordcloud con uns cantidad máxima de palabras, en donde se muestra que las palabras que mayor frecuencia dentro del corpus era ”coronavirus”, ”pandemic”, ”goverment”, ”cases”, entre otros. Además, se realizó un diagrama circular para visualizar el porcentaje de tweets positivos y negativos. Posteriormente, se realizó el pre procesamiento de los tweets. Para este proceso se crearon dos funciones llamadas lemmatize y preprocess. La primera sirve para realizar lematización y la segunda realiza los procesos de conversión a minúsculas, tokenización, elimininación de signos de puntuación y remover espacios en blanco. Una vez realizado el pre procesamiento a los tweets, se procedió a realizar la etapa de extracción de características. En esta etapa, se utilizaron 3 técnicas de extracción de características diferentes: TF-IDF, N-GRAMS(Unigrams-Trigrams) y Word2vect. En primer lugar, para crear la primera matriz utilizando TF-IDF, se utilizó la libreria sklearn, específicamente la función feature_extraction.text en donde se utilizó la función que se encuentra del mismo llamada TfidfVectorizer almacenada dentro de una variable llamada vectorizer. Para la creación de la matriz, se utilizó vectorizer.fit_transform y se les dio los tweets pre procesados. Luego de obtener la matriz TF-IDF, se estandarizaron los datos utilizando la función StandardScaler. En segundo lugar, para la obtención de la matriz usando N-Grams, se utilizó la funcion de CountVectorizer con un rango de ngrams de 1 al 3. Lo que significa que se tomarán desde unigrams hasta trigrams. Usando esta configuración, se realizó un fit.transform de tal modo que dio como resultado una nueva matriz utilizando unigrams hasta trigrams. Luego de obtener la matriz, se realizó la estandarización de los datos, usando nuevamente la función StandardScaler. Para la construcción de la matriz usando word2vect primero se tuvo que tokenizar todos los tweets con el objetivo de obtener una lista de listas y almacenarla en una variable llamada TweetsVec. Con esta variable creada, se procedió a crear el modelo de word2vect dandole como parámetro la varible anterior mensionada, un min_count de 1 y una longitud del vector (size) igual a 100. Una vez el modelo construido, se procedió a crear la matriz. En este caso, la matriz tuvo que construirse utilizando 2 bucles for. La lógica utilizada fue que el modelo de word2vect representaba cada palabra como un vector de tamaño 100. Lo que se elaboro fue un bucle que recorrira cada tweet tokenizado y realizar una suma de cada vector (palabra) y dividir la suma entre el total de palabras que tenga el tweet. De tal modo, que se obtenia un nuevo vector que representaba a todo el tweet. Con este procedimiento, se obtuvo la tercera matriz usando word2vect. Obtenida la matriz, se procedió a estandarizar los datos, utilizando nuevamente StandardScaler. Obtenida las 3 matrices normalizadas, se procedió a particionar la matriz en data train y data test (80% para el conjunto train y 20% para el conjunto test). Luego de la división de la data en train y test por cada matriz, se procedió a construir los modelos estadísticos. Como se mensionó al inicio de esta sección, se utilizarán 3 diferentes modelos: Regresión Logística, RandomForest y una red neuronal y se crearán 3 modelos diferentes del mismo modelo utilizando las 3 diferentes matrices creadas, es decir que en total habrán 9 diferentes modelos. Para iniciar con el modelo de regresión logística, se trabajó con 3 parámetros los cuales fueron solver =’lbfgs’ el cual es el algoritmo a utilizar para la optimización, max_iter=7600 el cual es el número máximo de iteraciones y class_weight = ”balanced” para establecer el peso que tendrá cada clase. Al usarse ”balanced”utiliza los valores de la clase para ajustar automáticamente los pesos inversamente proporcionales a las frecuencias de clase en los datos de entrada en el modelo. El segundo modelo de RandomForest, cuyos parámetros fueron n_estimators = 200 que es el número total de arboles que habrá dentro del modelo, max_depth = 3 que equivale a la profundidad que tendrá cada arbol y random_state= 0 para controlar la aleatoriedad en la elección de la muestra. El tercer modelo es una red neuronal cuyos parámetros fueron solver =

'lbfgs' para la actualización de los pesos, un alpha=1e-5 que es el parámetro de penalización, hidden_layer_sizes =(15,) que representa el número total de neuronal en la capa oculta y un random_state=1 para controlar la aleatoriedad en la elección de la muestra. Elaborados los 9 modelos diferentes, se procedió a utilizar las métricas de Acuracy, Precision, Recall y F1-score con el objetivo de determinar qué modelo es el más adecuado para realizar el análisis de sentimiento. Además se obtuvo la matriz de confusión por cada modelo para determinar el número de tweets correctamente clasificados por cada modelo. Para la evaluación de los modelos, se realizarán en bloques en función a cada técnica de extracción de características, es decir todos los modelos que utilizan tf-idf son comparados entre ellos de tal manera que se obtenga el mejor modelo usando tf-idf. De igual modo se realizará con N-grams y con Word2vect. Al finalizar la evaluación, se obtendrá el mejor modelo por cada técnica de extracción de característica para finalmente, ser comparadas entre los 3 modelos con el objetivo de obtener el mejor modelo de machine learning que servirá para realizar el análisis de sentimiento de manera individual y para combinarlo con el modelo de deep learning.

IV. RESULTADOS

En esta sección, se explicarán los resultados obtenidos luego de la realización de los experimentos con el objetivo de obtener el mejor modelo de detección de tópicos y de machine learning para descubrir tópicos y analizar el sentimiento de un determinado tweet.

A. Fase 1: Detección de tópicos

En este apartado de explicarán los resultados obtenido con respecto a los 3 modelos de detección de tópicos.

1) Modelo LDA: Para escoger, el mejor modelo de LDA, se construyeron 4 modelos diferentes. Los 2 primeros modelos fueron con una cantidad de tópicos igual a 10 y que correspondían al modelo de LDA con Doctobow y LDA usando TF-IDF. Para empezar, se realizó una visualzación de los datos con los cuales se va a trabajar tal como se muestra en la Figura 24 y la Figura 25.

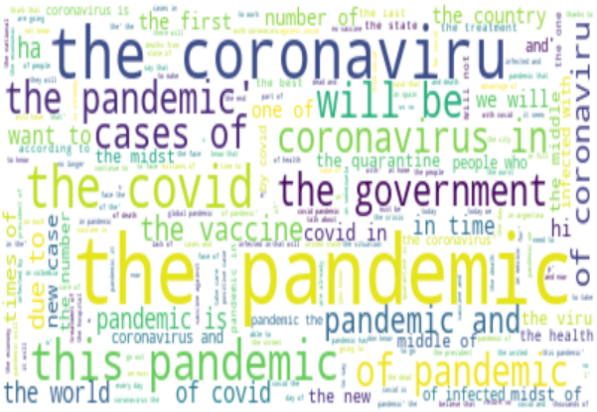

Fig. 24. Wordcloud de la data balanceada

Fig. 25. Porcentaje de tweets positivos y negativos

Como se puede observar en la Figura 24, las palabras más resaltantes fueron "coronavirus", "cases", "deaths", "world", "pandemic", "government", entre otros. Por otro lado, en la Figura 25, se observó que la cantidad de tweets positivos y tweets negativos son la misma.

1) Modelo tradicional LDA-Doctobow: Para la creación del modelo tradicional de usando Doctobow, se escogio un K = 10 solamente de prueba para verificar un valor inicial del Coherence que nos podía ofrecer. Los resultados de este modelo en particular fueron que obtuvo un Coherence Score de 0.2484 y Perplexity de -8.1315.

2) Modelo tradicional LDA-TF-DFI: Para realizar el modelo de TF-IDF, se transformó el corpus de Doctobow, de tal manera que se obtuvo un nuevo corpus que posea el identificador de cada palabra con su respectivo peso asignado. De igual manera, este modelo se realizó con un K = 10 de prueba para poder tener una idea de los valores resultantes que podrian salir. Con respecto a su Coherence Score fue de 0.2556 y de Perplexity fue de -9.4336.

Analizando los dos primeros modelos tradicionales, se observa que en términos de Coherence Score, el modelo de LDA-TF-DFI es mejor que el modelo de LDA-DoctoBow con una diferencia mínima. Por otro lado, en términos de Perplexity, el modelo de LDA con TDF-IDF es mejor que el modelo de LDA-DoctoBow dado que posee el valor de su Perplexity es más negativa.

3) Modelo Mejorado LDA-Doctobow: Para realizar el primer modelo mejorado de LDA, usando Doctobow se construyó una función que construyó 19 modelos de LDA-Doctobow, obteniendo el Coherence Score de cada uno con su respectiva Perplexity. La gráfica de los 19 experimentos en donde se observa la evolución del Coherence Score se muestra en la Figura 26.

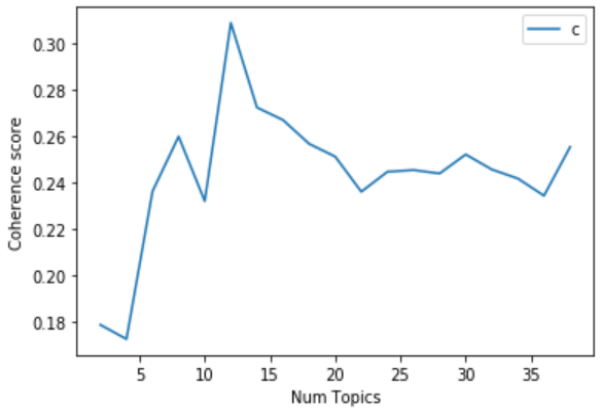

Fig. 26. Gráfica de la evolución del Coherence Score para el modelo de LDA-Doctobow

En la Figura 26, se observa que el mayor Coherence Score se da con una cantidad de tópicos igual a 12 y el valor del Coherence es de 0.3089. Sabiendo que el mejor modelo de DoctoBow tiene un K = 12, se procedió a calcular el valor de Perplexity y fue de -8.3718.

4) Modelo Mejorado LDA-TF-IDF: Para crear el mejor modelo con TF-IDF, se realizaron 19 experimentos con el corpus de tf-idf de tal manera que se crearon 19 modelos diferentes y se observó la evolución del Coherence Score en cada iteración. En la Figura 27, se observa el gráfico mencionado y se observa que el Coherence Score va en aumento hasta aproximadamente un K = 32, luego empieza a tener una disminución del Coherence Score. Por tal motivo, se decidió trabajar con un modelo K = 32 para LDA con TF-IDF.

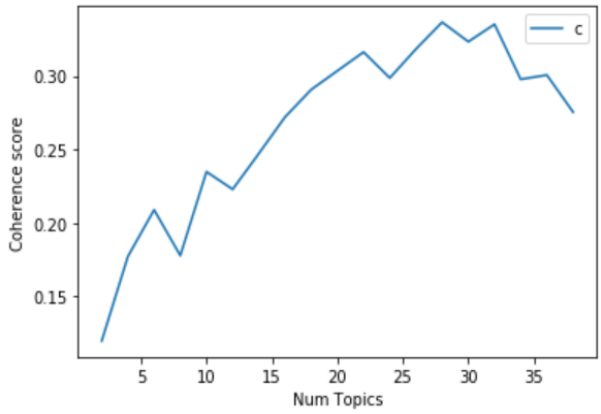

Fig. 27. Gráfica de la evolución del Coherence Score para el modelo de LDA-TF-IDF

Usando un K = 32 para el mejor modelo de LDA-TF-IDF, se obtuvo un Coherence Score de 0.335 y una Perplexity de -12.9630. De tal modo que el mejor modelo de LDA, en función tanto del Coherence Score como Perplexity es el LDA usando Doctobow dado que posee un Coherence Score mucho más elevado usando la menor cantidad de tópicos posibles.

A modo de resumen, se puede observar en la Tabla I , los diferentes modelos que se han construido con su respectiva cantidad de tópicos, el Coherence Score y Perplexity.

Modelo	Tópicos	Coherence Score	Perplexity
Tradicional LDA - Doctobow	10	0.2484	-8.1315
Tradicional LDA - TFIDF	10	0.2556	-9.4336
Mejorado LDA - Doctobow	12	0.3089	-8.3718
Mejorado LDA - TFIDF	32	0.335	-12.9630

TABLE I
RESUMEN DE MODELOS DE LDA DESARROLLADOS

Como se puede observar en la Tabla I, se construyeron 4 diferentes modelos: 2 de manera tradicional asignando manualmente una cantidad de tópicos específica y 2 mejorados en donde se encontraba el número óptimo de tópicos para cada modelo.

Finalmente, en la Tabla II, se pueden observar los tópicos que contiene el modelo. El criterio para la elección del nombre para cada tópico fue observar las palabras pertenecientes a cada tópico y poner las palabras que tienen mayor peso dentro del título de cada modelo.

2) Modelo HDP: Para el modelo HDP-BOW se corrieron un total de 25 modelos, iniciando en 5 tópicos y terminando en 30, esta corrida tomó un tiempo de 3 horas 25 minutos aproximandamente, de estos se obtuvieron los resultados mostrados en la Figura 28

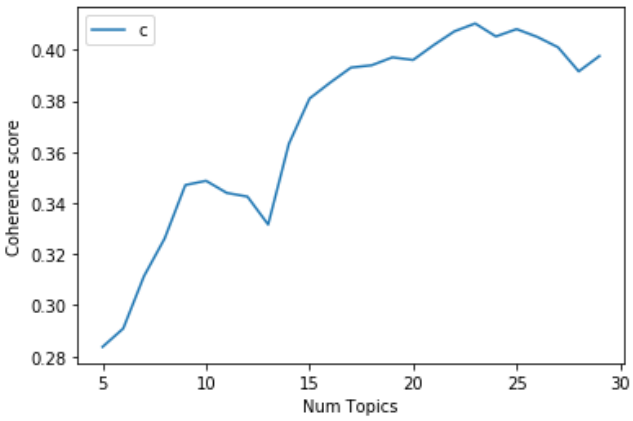

Fig. 28. Número de tópicos y coherence score para HDP-BOW

Así, como se puede observar el pico más alto es con 23 tópicos y un coherence de 0.4104, esto se puede analizar mejor en los resultados impresos mostrados en la Tabla III

Indice de tópico	Nombre de tópico
0	'Personas desean tratamiento para poner fin a la pandemia'
1	'Atención a pacientes con coronavirus y conocimiento sobre prevención para no contraer coronavirus'
2	'Compras de vacunas por parte del gobierno para el retorno a las escuelas para niños y niñas'
3	'Medidas de prevención para continuar con el trabajo en tiempos de pandemia'
4	'Medidas planteadas por los gobiernos para enfrentar al coronavirus'
5	'Noticias sobre incremento de casos de muertes por coronavirus'
6	'Noticias sobre la lucha para preservar la vida de las personas con coronavirus'
7	'Busqueda de una vacuna efectiva contra el nuevo coronavirus y confinamiento de niños en tiempos de pandemia'
8	'Noticias sobre los hospitales del Ministerio de salud para el cuidado de pacientes con Covid-19'
9	'Conocimiento de las personas acerca de la cuarentena general'
10	'Desarrollo de nuevos estudios y creación de vacuna contra el Covid-19'
11	'Informe sobre nuevos casos positivos de coronavirus'

TABLE II
NOMBRE DE LOS TÓPICOS PERTENECIENTES AL MEJOR MODELO DE LDA

#Topics	Coherence value	Position
20	0.3961	15
21	0.4019	16
22	0.4073	17
23	**0.4104**	**18**
24	0.4053	19
25	0.4081	20
26	0.4051	21
27	0.4010	22
28	0.3916	23
29	0.3976	24

TABLE III
TOPICS AND COHERENCES HDP-BOW

#Topics	Coherence value	Position
20	0.1492	15
21	0.1557	16
22	0.1562	17
23	0.1560	18
24	0.1578	19
25	**0.1589**	**20**
26	0.1588	21
27	0.1582	22
28	**0.1589**	**23**
29	0.1547	24

TABLE IV
TOPICS AND COHERENCES HDP-TFIDF

De la misma forma, se realizó el mismo procedimiento para HDP-TFidf obteniendo la Figura 29 y la tabla IV.

De ambos modelos, el mejor sería HDP-BOW, debido a que el valor de coherence score se acerca más a 1 que para el otro modelo. De esta forma, se tienen los primeros 5 tópicos mostrados en la Figura 30 y los temas asignados a estos .

3) Modelo LSI: Para LSI se trabajó con un modelo, la cual es el LSI con BOW y el detalle de sus resultados son lo siguiente. En la Figura 31 se puede visualizar que hay una caida con respecto a la relación entre numero de tópicos con el coherence value.

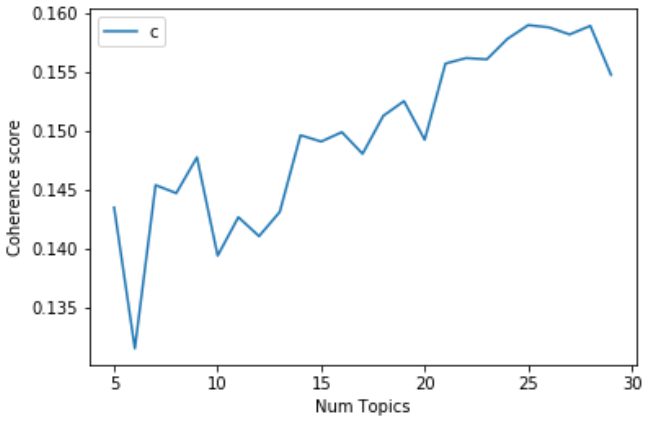

Fig. 29. Número de tópicos y coherence score para HDP-TFidf

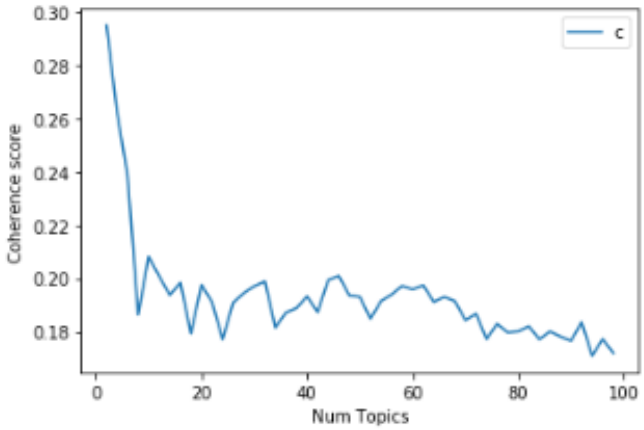

Fig. 31. Número Tópicos por coherence value en el modelo LSI

```
Topic 0: Nuevos casos de personas infectadas por coronavirus
pandemic*0.0288 + coronavirus*0.0248 + covid*0.0183 + infect*0.0085 + people*0.0075 + case*0.0075 + vaccine*0.0060 +
health*0.0057 + quarantine*0.0055 + government*0.0052 + country*0.0051 + know*0.0048 + take*0.0048 + today*0.0045 +
time*0.0045 + death*0.0045 + work*0.0038 + say*0.0036 + make*0.0035 + treatment*0.0035

Topic 1: Búsqueda de vacuna para el covid
pandemic*0.0479 + vaccine*0.0166 + treatment*0.0115 + covid*0.0067 + time*0.0057 + take*0.0055 + good*0.0055 +
people*0.0055 + know*0.0055 + coronavirus*0.0053 + want*0.0047 + like*0.0045 + well*0.0042 + make*0.0042 + come*0.0041 +
think*0.0040 + quarantine*0.0039 + would*0.0038 + thing*0.0035 + without*0.0034

Topic 2: Tratamiento para el covid
pandemic*0.0067 + treatment*0.0041 + coronavirus*0.0033 + vaccine*0.0022 + bonus*0.0022 + covid*0.0019 + believe*0.0016 +
people*0.0014 + quarantine*0.0014 + like*0.0014 + think*0.0012 + time*0.0011 + infect*0.0010 + would*0.0010 + need*0.0010
+ leave*0.0010 + make*0.0010 + government*0.0009 + today*0.0009 + know*0.0009

Topic 3: El gobierno emite bono para las personas que necesitan
pandemic*0.0006 + coronavirus*0.0006 + bonus*0.0005 + bond*0.0004 + quarantine*0.0004 + people*0.0003 + say*0.0003 +
government*0.0003 + work*0.0003 + take*0.0002 + want*0.0002 + need*0.0002 + today*0.0002 + believe*0.0002 + give*0.0002 +
make*0.0002 + face*0.0002 + president*0.0002 + problem*0.0002 + call*0.0002

Topic 4: Personas que trabajan solicitan bono
want*0.0006 + bonus*0.0005 + pandemic*0.0005 + covid*0.0003 + coronavirus*0.0003 + thank*0.0003 + work*0.0002 +
people*0.0002 + bond*0.0002 + country*0.0002 + shampoo*0.0002 + government*0.0002 + quarantine*0.0002 + diver*0.0002 +
anti*0.0001 + shame*0.0001 + agents*0.0001 + go*0.0001 + reconfuse*0.0001 + yellow*0.0001

Topic 5: Noticias del gobierno sobre la pandemia
bonus*0.0002 + pandemic*0.0002 + coronavirus*0.0002 + vaccine*0.0002 + treatment*0.0002 + nicaragua*0.0001 + accept*0.0001
+ even*0.0001 + quarantine*0.0001 + cloned*0.0001 + infect*0.0001 + covid*0.0001 + diadelkinesiologo*0.0001 +
president*0.0001 + mean*0.0001 + infected*0.0001 + discipline*0.0001 + obrasprivadas*0.0001 + copywriter*0.0001 +
fala*0.0001
```

Fig. 30. Top 5 de tópicos para HDP-BOW

Para esta investigación se decidió trabajar con un minimo de 6 topicos en adelante. Por lo tanto, se consideró el mejor coherence score encontrado luego de haber llegado a 6 tópicos. En la tabla 32 se detalla los coherence values por cada incremento en los tópicos.

#Topics	Coherence Value	Position
2	0,2951	0
4	0,2627	1
6	0,2387	2
8	0,1865	3
10	0,2082	4
12	0,2008	5
14	0,1938	6
16	0,1985	7
18	0,1794	8
20	0,1977	9
22	0,1912	10

Fig. 32. Coherence Value Per Topics LSI

Según esta tabla, identificamos que en numero de tópicos igual a 10 tenemos un coherence score 0,2082 la cual es mayor a los demas. Por lo tanto, como el mejor modelo nos quedamos con esta cantidad de topicos (numero topicos = 10). En el imagen 33 se detalla el top 5 de los topicos con sus respectivas porcentajes.

B. Fase 2: Análisis de Sentimientos

Para la segunda fase se utilizaron tres modelos para vectores característicos: TF-idf, N-grams y Word2vect. Se realizó la partición de la data en 80% y 20% para entrenamiento y prueba para cada vector. Para realizar la construcción de modelos, primero se hizo uso GridSearchCV para que se encuentren los parámetros que permitan a los modelos tener los mejores resultados. En el caso de Regresión Logística, se utilizaron los parámetro de ”C” con valores de 0.1 y 0.5; solver con parámetros de newton-cg, lbfgs y sag y max_iter con valores de 14000 y 21000. En el caso de RandomForest, se utilizó el parámetro de n_estimators con valores de 200 y 300; max_depth con valores de 7 y 9 min_sample_leaf con valores 0.5 y 1. Finalmente, para el caso de la Red Neuronal, se utilizó el parámetro de learning_rate con valores de constant, invscaling, hidden_layer_sizes con valores de (10,1), (10,2) y activation con valores de ”logistic” y ”relu”. Los resultados luego de realizar el tuneo respectivo fueron en primer lugar para TF-IDF en donde el modelo de Regresión Logística tiene unos valores de C = 0.1, max_iter = 14000, solver = ’lbfgs’. Para el caso de RandomForest, se obtuvieron unos valores de max_depth = 4, max_features = ’auto’, n_estimators = 200. Finalmente, para el caso de la Red Neuronal, se obtuvo unos valores de activation = ’logistic’, hidden_layer_sizes = (10, 1), learning_rate = ’constant’). En segundo lugar, usando N-grams, el modelo de Regresión Logística obtuvo unos valores de C = 0.1, max_iter = 21000, solver = ’sag’. Para el caso de RandomForest, max_depth = 4, max_features = ’auto’, n_estimators = 400 y finalmente, para la Red Neuronal,

```
Topic: 0
Words: 0.932*"pandemic" + 0.200*"covid" + 0.174*"coronavirus" + 0.087*"people"
+ 0.053*"government" + 0.047*"health" + 0.045*"cases" + 0.043*"times" + 0.042
*"today" + 0.041*"infected"
Topic: 1
Words: 0.875*"coronavirus" + -0.272*"pandemic" + 0.263*"covid" + 0.166*"cases"
+ 0.133*"infected" + 0.082*"deaths" + 0.069*"vaccine" + 0.052*"people" + 0.048
*"health" + 0.039*"confirmed"
Topic: 2
Words: -0.894*"covid" + 0.375*"coronavirus" + 0.149*"pandemic" + -0.092*"infect
ed" + -0.090*"cases" + -0.085*"vaccine" + -0.044*"health" + -0.037*"treatment"
+ -0.031*"patients" + -0.029*"today"
Topic: 3
Words: 0.732*"infected" + 0.565*"people" + -0.187*"covid" + -0.153*"coronaviru
s" + 0.149*"health" + 0.092*"number" + 0.091*"dead" + 0.080*"quarantine" + 0.06
5*"today" + -0.061*"pandemic"
Topic: 4
Words: 0.779*"vaccine" + 0.404*"people" + -0.345*"infected" + -0.166*"cases" +
0.108*"treatment" + 0.075*"know" + 0.071*"like" + 0.067*"virus" + 0.066*"quaran
tine" + -0.066*"deaths"
```

Fig. 33. Top5 Topics of LSI

se obtuvo unos valores de parámetros finales de activation = 'logistic', hidden_layer_sizes = (10, 1), learning_rate = 'constant'. En tercer y último lugar, utilizando Word2vec, en el caso de Regresión Logística unos valores de C = 0.1, max_iter = 14000, solver = 'lbfgs', para RandomForest unos valores de max_depth = 5, max_features = 'auto', n_estimators = 200 y para la Red Neuronal, unos valores de activation = 'relu', hidden_layer_sizes = (10, 2), learning_rate = 'constant'. Luego de obtener estos datos, se relizó fimalmente la creación de los 9 modelos y se comprararon para encontrar al mejor modelo predictivo. Dado que se trabajó con grandes volumenes de datos, a cada integrante se le asignó un modelo en particular para que lo creara por cada tipo de técnica de extracción de característica. Es decir, cada integrante debía realizar 3 diferentes modelos utlizando el mismo clasificador pero con diferente vector característico. Los resultados de cada tipo de modelo, se presentarán en los siguientes apartados.

1) Regresión Logística: Para el modelo de regresión logística, se realizaron 3 modelos diferentes con el mismo clasificador. Un modelo de regresión logistica con Tf-idf, otro con Ngrams y finalmente uno con Wordtovec. Los resultados de los experimentos, se presentan en la tabla V y tabla VI. Como se puede observar, el mejor modelo de regresión logística, es utilizando el vector de TF-IDF. El segundo mejor es utilizando Ngrams y en último lugar, se encuentra el modelo de regresión logistica utilizando Wordtovec.

Para una mejor visualización de los resultados de los accuracies tanto del train como del test, se presentan 2 gráficos en las figura 34 y figura 35 , respectivamente.

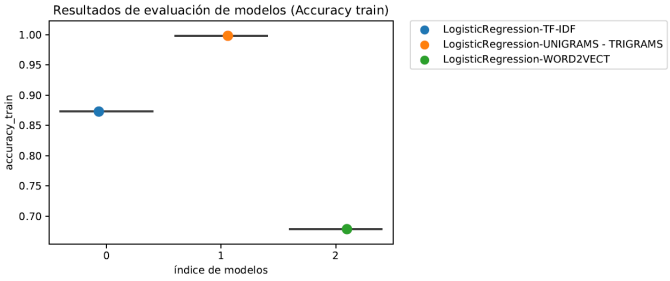

Fig. 34. Gráfica de los accuracies train de modelos de regresión logística

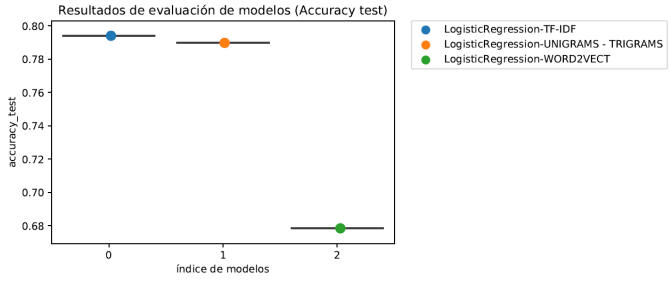

Fig. 35. Gráfica de los accuracies test de modelos de regresión logística

Por otro lado, analizando las 4 métricas para evaluar cada modelo según la tabla VI, existe un empate entre los modelos de regresión logística con TF-IDF y Ngrams. Por tal motivo, analizando el accuracy de cada modelo, se puede llegar a la conclusión que el mejor modelo de regresión logística es utilizando TF-IDF con un accuracy de 0.7940.

2) Random Forest: Para este modelo, se colocaron como parámetros n_estimators, max_depth y min_sample_leafs, para estos, se realizaron pruebas en las que se notó que el accuracy promedio mejoraba si si tanto el n_estimators como el max_depth aumentaba; finalmente se probaron 6 combinaciones: 200 y 300 estimadores, 7 y 9 para la profundidad y

Modelo	Técnica de extracción de característica	Accuracy Train	Accuracy Test
Regresión Logística	TF-IDF	0.8727	0.7940
Regresión Logística	NGRAMS	0.9979	0.7898
Regresión Logística	Word2vec	0.6783	0.6784

TABLE V
RESULTADOS GENERALES DE MODELOS DE REGRESIÓN LOGÍSTICA (I)

Modelo	Técnica de extracción de característica	Precision	Recall	F1-Score	Support
Regresión Logística	TF-IDF	0.79	0.79	0.79	89105
Regresión Logística	NGRAMS	0.79	0.79	0.79	89105
Regresión Logística	Word2vec	0.68	0.68	0.68	89105

TABLE VI
RESULTADOS GENERALES DE MODELOS DE REGRESIÓN LOGÍSTICA (II)

los valores de 0.5 y 1 para el mínimo número de muestras. De esta forma se obtuvieron los mejores parámetros mostrados en la Tabla VII

Model	**n_estimators**	**max_depth**	**min_sample_leafs**
RF-TFidf	300	9	1
RF-ngrams	300	9	1
RF-Word2vect	200	9	1

TABLE VII
BEST PARAMETERS FOR RANDOM FOREST

De esta forma se crearon tres modelos de Random Forest, cada uno con sus extractores de características y sus mejores parámetros, que finalmente fueron evaluados usando el accuracy como métrica principal. Estos resultados se muestran en las Figuras 36 y 37

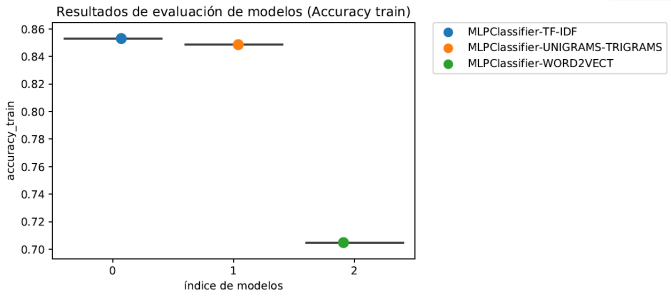

Fig. 36. Resultados de accuracy para Train

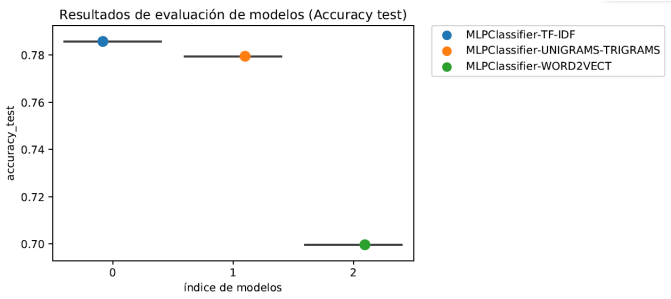

Fig. 37. Resultados de accuracy para Test

Finalmente también se crearon el reporte de clasificación y la matriz de confusión, dando también las métricas de precision, recall y F1-score y soporte, las cuales son presentadas en la siguiente tabla (resultados para test)

Modelo	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Support**
RF-TFidf	0.75	0.75	0.75	0.74	89105
RF-ngrams	0.74	0.74	0.74	0.74	89105
RF-Word2vect	0.68	0.68	0.68	0.68	89105

TABLE VIII
RESULTADOS PARA LOS MODELOS DE RANDOM FOREST

De esta forma, se concluye que para Random Forest, el mejor modelo estaría entre los modelos que usaron TF-idf y ngrams dado que sus valores para las métricas son similares, sin embargo, por unas décimas, es el que utiliza TF-idf como vector característico el que tiene el mayor valor de accuracy con 0.7454 ganando al segundo lugar cuyo valor era de 0.7427.

3) MLP: En el modelo MLP, al igual que los 2 modelos presentados previamente, se realizaron combinaciones con los modelos Tf-idf, Ngrams y Word2vec. En la figura 38 podemos visualizar los resultados de las combinaciones del modelo MLP con las tecnicas de extracción de caracteristicas mencionados previamente.

		Train				Test			
Modelo	**Tecnica**	**Accuracy**	**Presicion**	**Recall**	**f1-score**	**Accuracy**	**Presicion**	**Recall**	**f1-score**
MLP	Word2Vec	0,7048	0,71	0,7	0,7	0,6997	0,71	0,7	0,7
	TF-IDF	0,8531	0,85	0,85	0,85	0,7858	0,79	0,79	0,79
	Ngrams	0,8486	0,85	0,85	0,85	0,7795	0,78	0,78	0,78

Fig. 38. Tabla Consolidado del Modelo MLP con las técnicas de extracción de caracteristicas

Para encontrar la mejor combinación evaluamos el accuracy y lo graficamos para un mejor visualización. A continuación se muestran las mejores accuracies en el train y test respectivamente en las gráficas 39 y 40

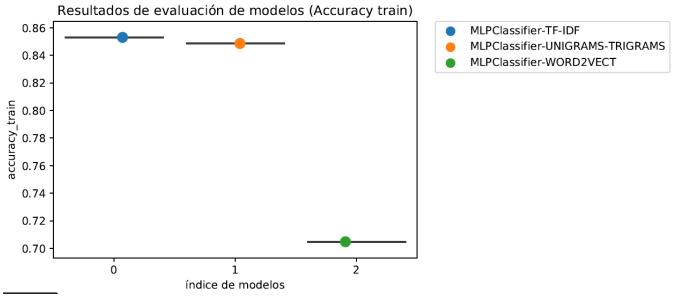

Fig. 39. Grafica Modelo MLP train y técnicas de extracción de caracteristicas

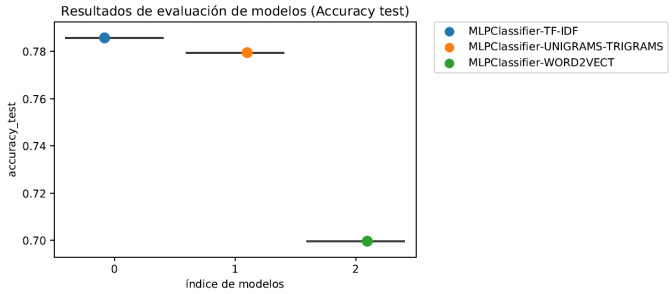

Fig. 40. Grafica Modelo MLP test y técnicas de extracción de caracteristicas

En conclusión, según lo que podemos visualizar en la gráfica 41, la mejor técnica de extracción de datos para el modelo MLP es el TF-IDF con un accuracy 0,8531 para train y 0,7858 para test.

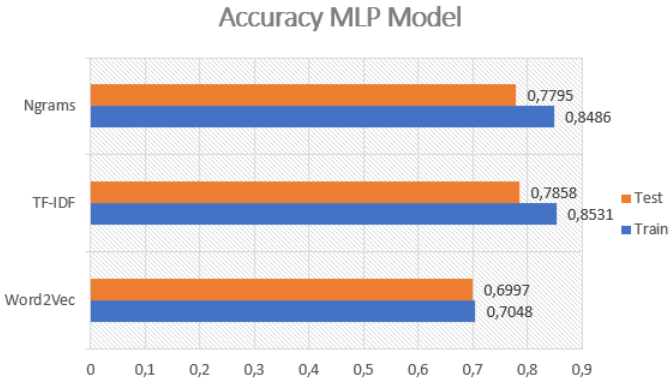

Fig. 41. Grafica Consolidada del Modelo MLP y técnicas de extracción de caracteristicas

4) Resultados finales de modelos de machine learning: Para resumir los resultados anteriores descritos y luego de realizar un análisis por cada modelo, se presenta en la tabla IX y la tabla X los mejores modelos de machine learning por cada tipo de extracción de características.

- Creación de sistema de clasificación de tópicos y análisis de sentimiento: Finalmente, se creó un sistema utilizando Streamlit para la creación del sistema e impresión de resultados, Sublime Text para la edición y Heroku con el objetivo de realizar el despligue del sistema. Este sistema se realizó con el objetivo de realizar análisis de sentimientos y detección de tópicos introduciento un tweet determinado en tiempo real. Para la construcción del sistema, se exportaron los 3 mejores modelos de machine learning en formato .pkl utilizando la librería pickle de python. Además, se exportaron en archivo .pkl, los objetos vectorizer, ngrams y el modelo de word2vect dado que estos objetos transforman el tweet ingresados en un vector de una longitud determinada. Además, se introdujo una función de limpieza del tweet antes que este se convierta en un vector de características. En segundo lugar, para realizar la predicción del tópico al cual pertenece un tweet, se exportó el mejor modelo de LDA que fue LDA usando TFIDF, el mejor modelo de HDP y el mejor modelo de LSI con su diccionario. Además, se creó una función que reemplaza el index del tópico de cada modelo por su respectivo nombre para que el usuario pueda tener conocimiento del tópico al cual pertenece su tweet. Además, se introdujeron gráficos en tiempo real con respecto a las actualizaciones de los datos de casos nuevos de coronavirus, número actual de infectados y de fallecidos alrededor del mundo. Finalmente, se introdujo un apartado de música clásica para mejorar la experiencia del usuario con respecto a su navegación dentro de la pagina desarrollada. El sistema descrito anteriormente, se puede observar en la Figura 42, Figura 43, Figura 44, Figura 45, Figura 46 y Figura 47. Para exportarlo en Heroku, debido al peso de los archivos, solamente se exportó el mejor modelo de detección de tópicosy el mejor modelo para realizar análisis de sentimiento. El sistema anteriormente mensionado, se puede encontrar en el siguiente link: https://app-finalbd2.herokuapp.com/.

V. Conclusiones

Con respecto a la fase 1, se desarrollaron 19 experimentos tanto para LDA-Bow y LDA-Tfidf dando como resultado el LDA con Doctobow con una cantidad de tópicos igual a 12 y con un Score de 0.3089.

Así mismo para HDP, se realizaron 24 experimentos para HDP-BOW y otros 24 para HDP-TFidf, de estos se obtuvo que el modelo HDP-BOW con 23 tópicos era el mejor entre ellos, con un coherence score de 0.4104.

Por ultimo, para el modelo LSI se realizó experimento con DocToBow. Se obtuvo el numero optimo de topicos de 10 con un coherence value de 0,2082.

Con respecto a la fase 2, los 3 mejores modelos de machine learning resultantes fueron Regresion Logistica - TF-IDF con un acuraccy de 0.7940, Regresion Logistica - Ngrams con un acuraccy de 0.7898 y Red Neuronal - Word2vect con un acuraccy igual a 0.6996. El mejor modelo de machine learning en general fue el de regresión logística con Tf-idf.

VI. Recomendaciones y trabajos a futuro

Como recomendaciones, se sugiere disminuir la longitud de los vectores característicos, en especial si se trabaja con Ngrams que lleguen hasta Trigrams dado que la longitud del vector se incrementará desmesuradamente trayendo como consecuencia una lenta creación de los modelos de machine learning. Como trabajos a fututo, se plantea la creación de un sistema de analisis de sentimiento pero analizando otros

Modelo	Técnica de extracción de característica	Accuracy Train	Accuracy Test
Regresión Logística	TF-IDF	0.8727	0.7940
Regresión Logística	NGRAMS	0.9979	0.7898
Red Neuronal	Word2vec	0.7047	0.6996

TABLE IX
RESULTADOS DE MEJORES MODELOS POR CADA TÉCNICA DE EXTRACCIÓN DE CARACTERÍSTICAS (I)

Modelo	Técnica de extracción de característica	Precision	Recall	F1-Score	Support
Regresión Logística	TF-IDF	0.79	0.79	0.79	89105
Regresión Logística	NGRAMS	0.79	0.79	0.79	89105
Red Neuronal	Word2vec	0.70	0.70	0.70	89105

TABLE X
RESULTADOS DE MEJORES MODELOS POR CADA TÉCNICA DE EXTRACCIÓN DE CARACTERÍSTICAS (II)

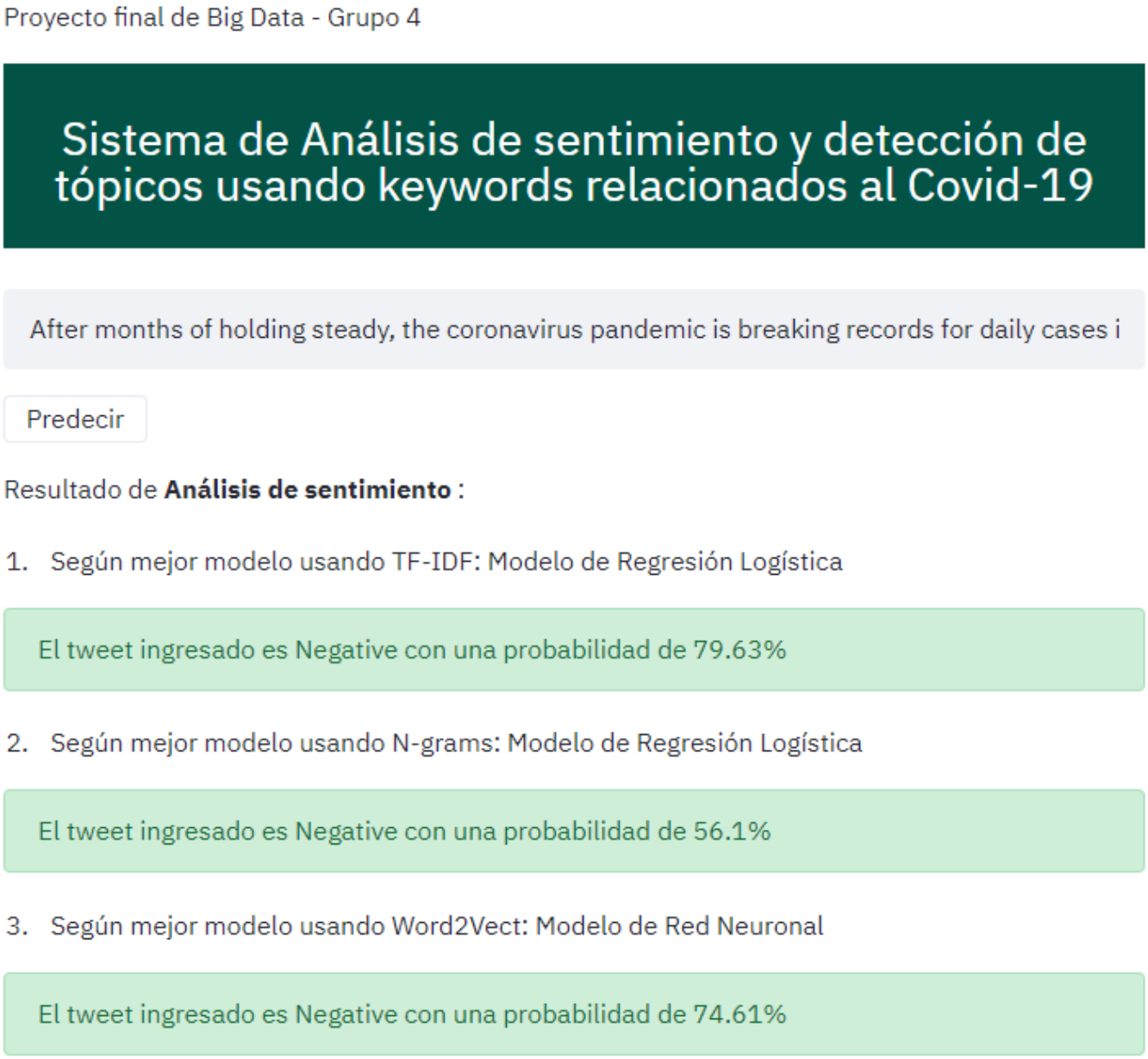

Fig. 42. Sistema de detección de tópicos y análisis de sentimiento - I

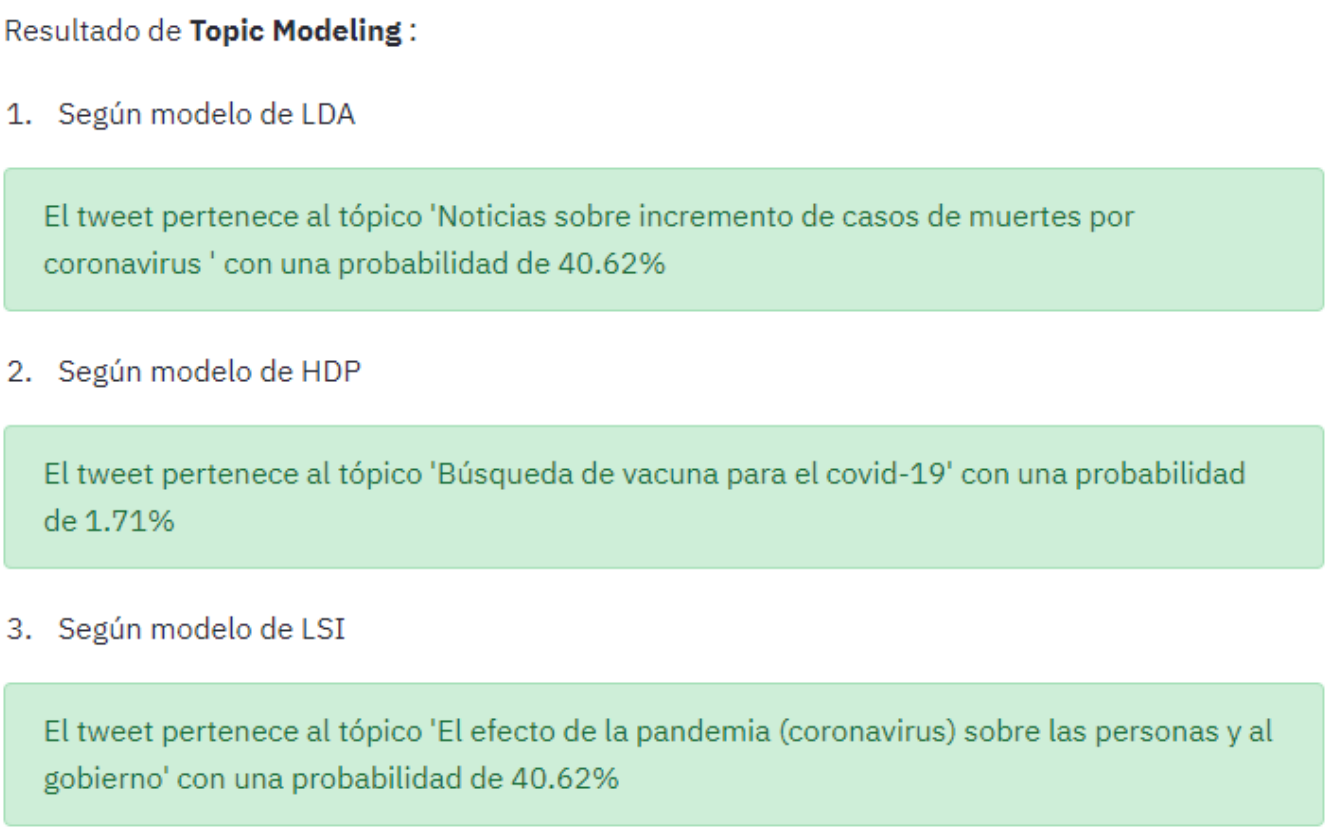

Fig. 43. Sistema de detección de tópicos y análisis de sentimiento - II

modelos de machine learning que no se han visto en el presente trabajo y poder determinar qúe modelo puede ser el mejor con el objetivo de clasificar un tweet como positivo o negativo.

Además, estos algoritmos podrían ser aplicados a otra data de la cual se requiera conocer sus tópicos y determinar el sentimiento de las personas hacia esto, lo cual es usado actualmente para la información que es publicada en internet.

References

[1] R. Annisa, I. Surjandari and Zulkarnain, Opinion Mining on Mandalika Hotel Reviews Using Latent Dirichlet Allocation, Procedia Computer Science, 2019, vol. 161, 2019, pp. 739–746.

[2] Srijith, P. K., Hepple, M., Bontcheva, K., & Preotiuc-Pietro, D. (2017). Sub-story detection in twitter with hierarchical dirichlet processes doi:https://doi.org/10.1016/j.ipm.2016.10.004

[3] Elena Georgiadou, Spyros Angelopoulos and Helen Drake, International Journal of Information Management,

[4] Amador, J., Collignon-Delmar, S., Benoit, K., Matsuo, A. (2017). Predicting the Brexit vote by tracking and classifying public opinion using twitter data. Statistics, Politics and Policy, 8(1), 85–104.

[5] Hurlimann, M., Davis, B., Cortis, K., Freitas, A., Handschuh, S., Fernández, S. (2016). A twitter sentiment gold standard for the Brexit referendum. the Proceedings of the 12th international conference on semantic systems, 193–196.

[6] Hall, W., Tinati, R., , Jennings, W. (2018). From Brexit to Trump: Social media's role in democracy. Computer, 51(1), 18–27

[7] Llewellyn, C., Cram, L. (2016). Brexit? Analyzing opinion on the UK-EU referendum within twitter. Association for the advancement of artificial intelligence. Available online:

[8] Giachanou, A., & Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods. ACM Computing Surveys (CSUR), 49(2), 28.

[9] Yujun Zhou1, Changliang Li2, Saike He3,4, Xiaoqi Wang1, Yiming Qiu1 1Jiangsu Jinling Science and Technology Group Co., Ltd, Nanjing, China 2Kingsoft AI Lab, Beijing, China (2016). Pre-trained Contextualized Representation for Chinese Conversation Topic Classification

Evolución del total de casos de fallecidos por Covid-19

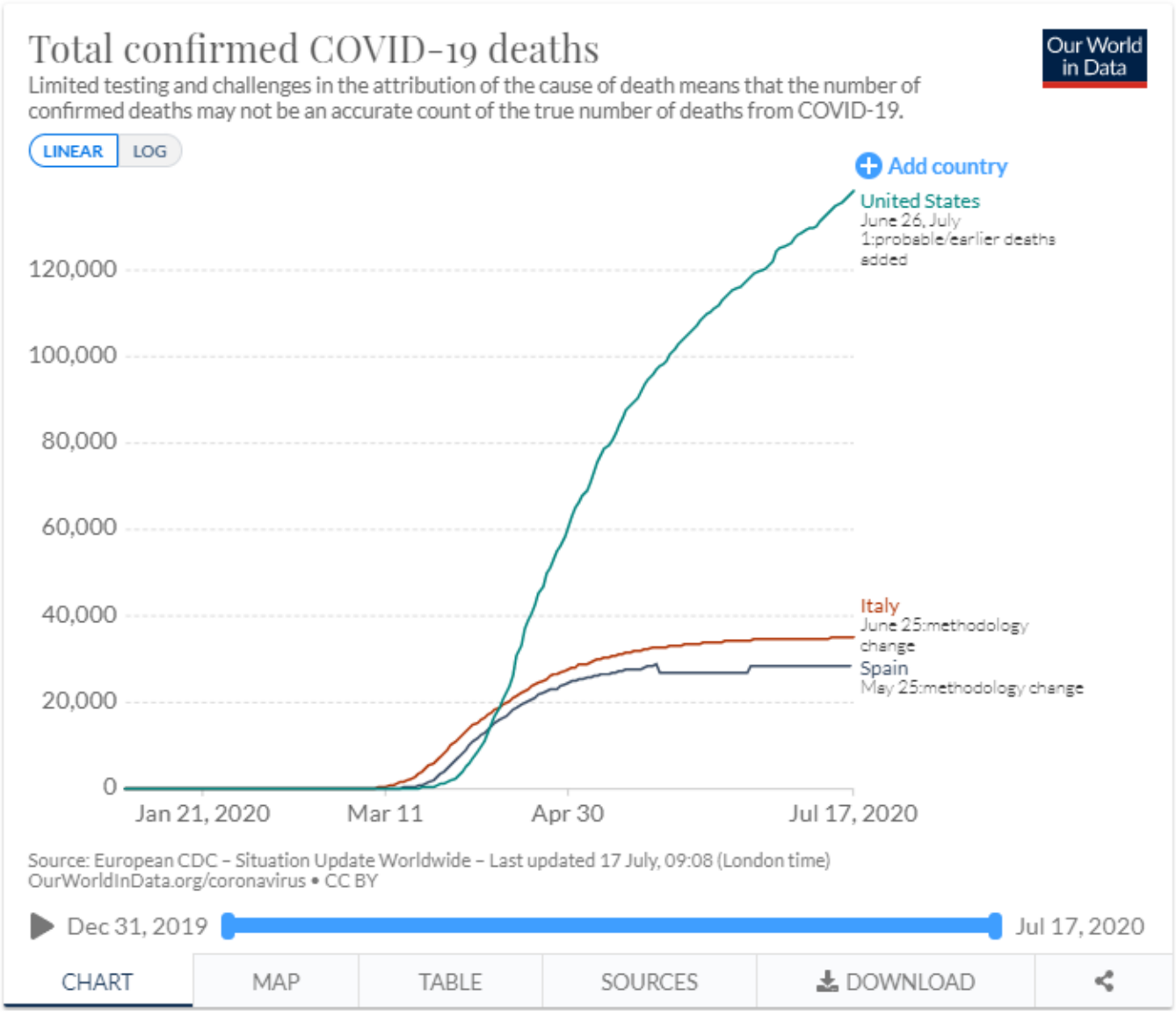

Fig. 44. Sistema de detección de tópicos y análisis de sentimiento - III

Crecimiento de casos confirmados de casos de Covid-19

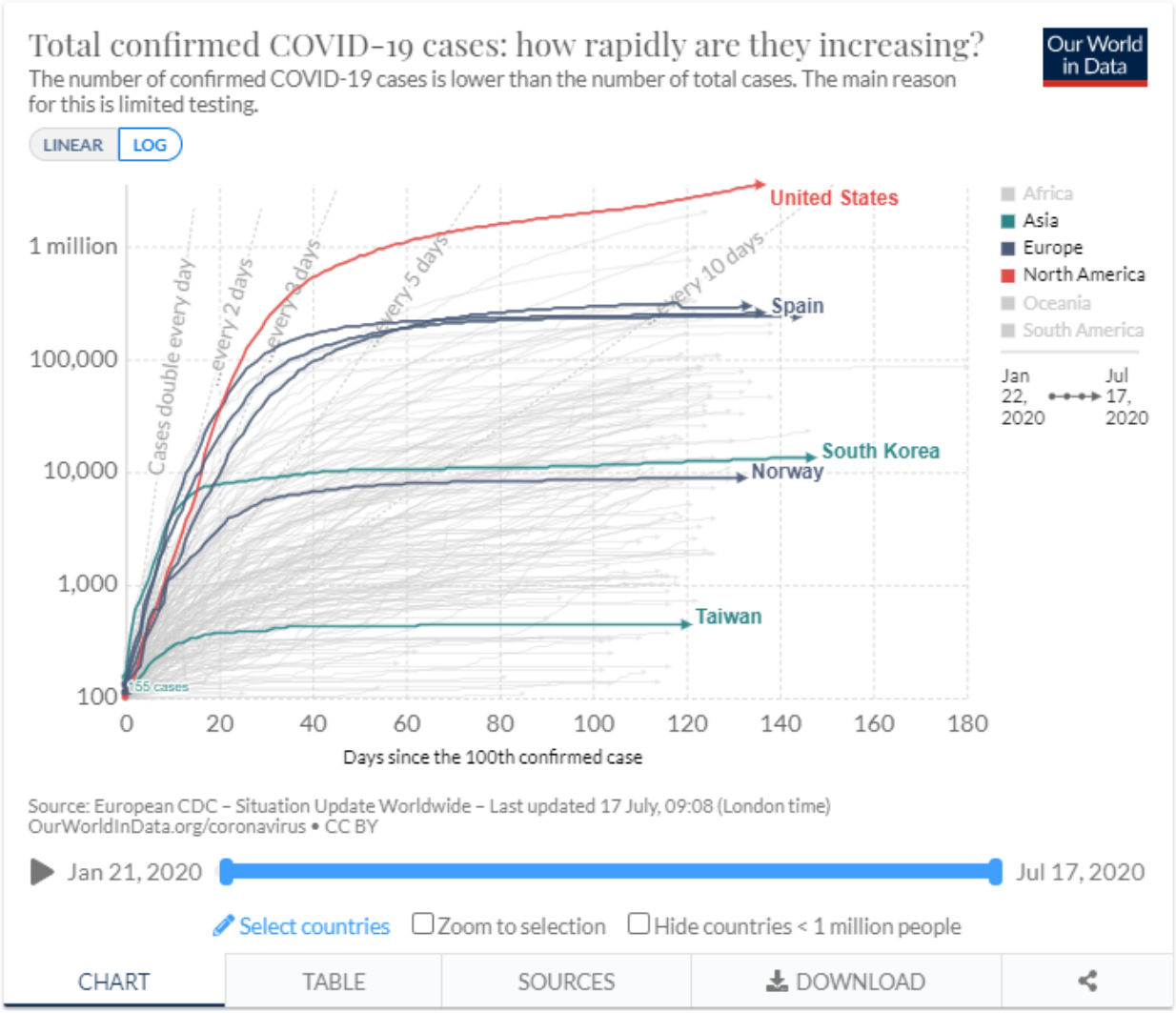

Fig. 45. Sistema de detección de tópicos y análisis de sentimiento - IV

Total de pruebas rápidas para detección de Covid-19 por cada 1000 personas

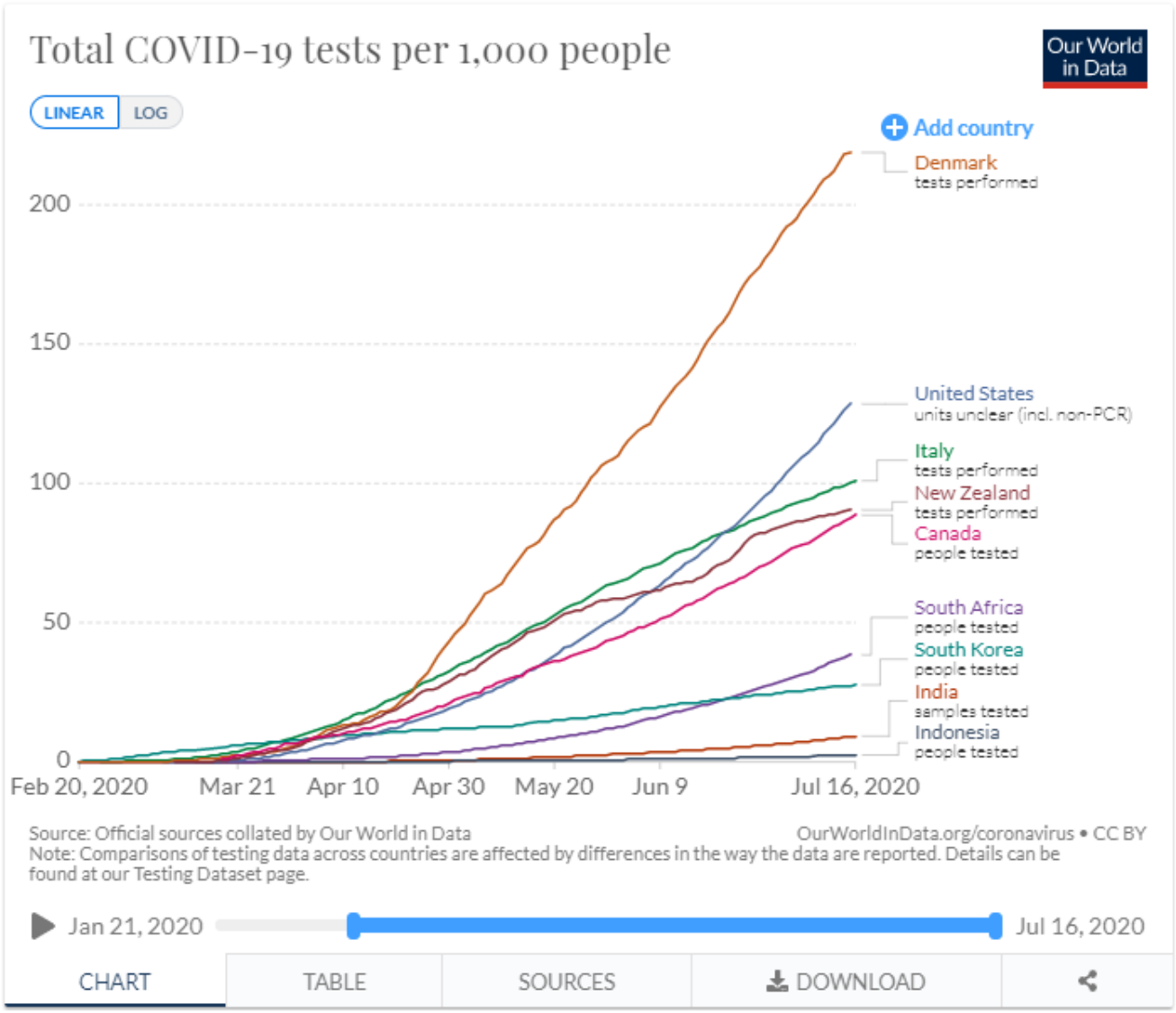

Fig. 46. Sistema de detección de tópicos y análisis de sentimiento - V

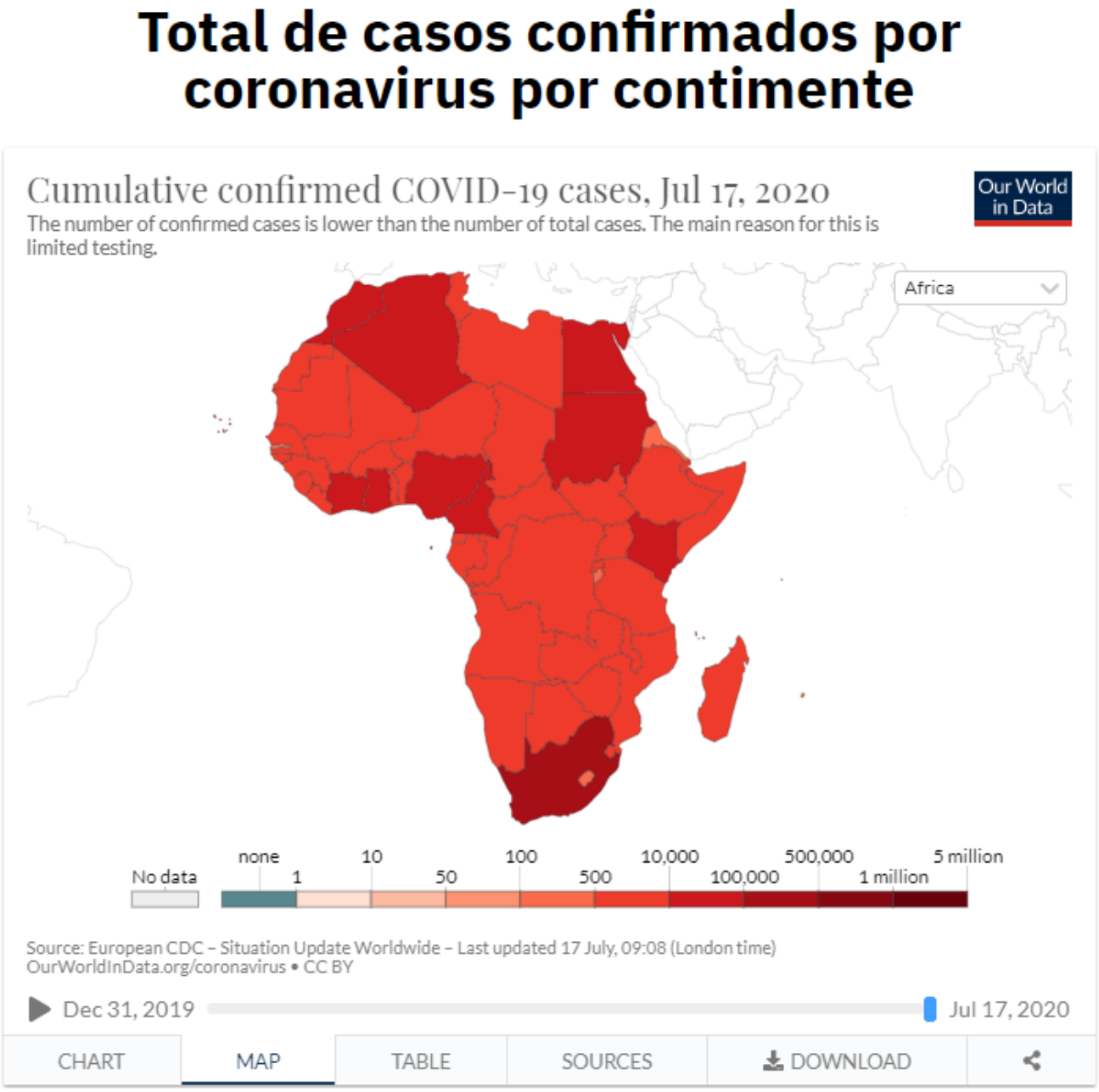

Fig. 47. Sistema de detección de tópicos y análisis de sentimiento - VI