Descubrimiento de tópicos y análisis de sentimientos usando hastags relacionados al Covid-19

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

2th Jhonathan Camasca Huamán
Universidad ESAN
Código: 15100036
15100036@ue.edu.pe
Lima, Perú

***Abstract*—El presente documento consistió en desarrollar un modelo de LDA para realizar la detección de tópicos y un modelo de análisis de sentimiento para poder clasificar tweets extraídos usando keywords relacionados al Covid-19. Para lograr lo anterior, se dividió el trabajo en dos fases. En la primera fase, se construyó el modelo de LDA, en donde se hizo una comparación de dos modelos de LDA (usando Doc2bow y TF-IDF) de tal modo que el mejor modelo de LDA fue aplicando TF-IDF para lo que se uso una cantidad de tópicos igual a 24, con un Coherence Score de 0.4408 y Perplexity de -9.5593. La fase 2 consistió en crear un modelo de machine learning, uno de deep learning y uno que combine los 2 anteriores. Para escoger el mejor modelo de machine learning, se compararon 3 modelos diferentes(Regresión Logística, RandomForest y una Red Neuronal). Además, se construyeron estos modelos utilizando 3 técnicas de extracción de características diferentes (TF-IDF, N-GRAMS y Word2vect). El mejor modelo por cada tipo de técnica fue RandomForest usando TF-IDF con un accuracy de 54%, Logística con N-GRAMS con un Accuracy de 79% y Red Neuronal con Word2vect con un accuracy de 68%. El mejor modelo de los 3 mejores, fue la Regresión Logística con TF-IDF. El mejor modelo de machine learning fue utilizando Regresión Logística con N-GRAMS con un Accuracy de 79%. Para construir los modelos de Deep learning se realizaron 3 tipos de pipelines : modelos de deep learning con word embeddings pre-entrenados o sin entrenar, modelos de deep learning multicanal y modelos de deep learning con machine learnig usando tecnicas de voting y average. El mejor modelo de deep learning fue el que tenia el embedding Glove obteniendo un resultado comparable a los modelos multicanal.Por otro lado, el mejor mmodelos multicanal fue un modelo con un optimizador RSMprop, 100 epocas, 3 diferentes tipos de filtros de diferentes tamaños y una capa de 100 unidades de LSTM.**

Finalmente, se construyó un sistema usando StreamLit y Heroku en donde se introdujo el mejor modelo de LDA Y de análisis de sentimiento para poder introducir un Tweet en tiempo real y saber el tópico al cual pertenece y el sentimiento que posee. Además, se introdujeron gráficos en tiempo real que muestra la situación actual de los casos de infectados y fallecidos por Covid-19 en el mundo.

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

I. Introduction

En estos tiempos, alrededor del mundo se están viviendo momentos difíciles por el surgimiento del nuevo coronavirus conocido como Covid-19. Este virus letal surgido en china en diciembre del 2019 tiene a las personas viviendo con un profundo temor hacia esta nueva enfermedad dado que cada día se incrementan el número de contagiados y fallecidos por esta enfermedad. Como medida de prevensión, muchos gobiernos han tomado medidas drásticas con el objetivo de prevenir la propagación de esta enfermedad. Una de las medidas optadas por los países ha sido una cuarententa total con el objetivo de poder disminuir el transito de personas y así evitar la propagación del virus.Dado a lo último mensionado, las personas al estar mucho más tiempo en las casas, han usado las redes sociales para expresar sus emociones con respecto a todas las declaraciones o nuevas medidas optadas por el propio gobierno y con respecto al coronavirus en general.

Dada la cantidad gran cantidad 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 estan 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 utiliza la técnica de LDA para la detección de tópicos de tweets recolectados de twitter usando hashtags relacionados al coronavirus en el país de Perú. Además, se utiliza una técnica de análisis de sentimientos para la clasificación de los tweets en positivos o negativos. Para la elección de la técnica de análisis de sentimiento a usar, se hace una comparación de 3 modelos: 1 modelo de machine learning, 1 modelo de deep learning, un modelo de deep learning con machine learning. Se evaluarán los 3 modelos utilizando el Acuracy, Precision, Recall y F1-score con el objetivo de determinar qué modelo es el más adecuado para realizar el análisis de sentimiento.

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. Latent Dirichlet allocation (LDA) for topic modeling of the CFPB consumer complaints

Este primer trabajo consiste en la utilización de la técnica Latent Dirichlet Allocation conocida como LDA con el objetivo de detectar los tópicos presentes en las quejas que recibe la Ofinica para la protección Finaciera del Consumidor (CFPB por su siglas en inglés) y así agilizar el proceso de análisis de quejas dado que estas últimas han ido en aumento en los últimos años y el hacer el análisis de queja por queja por parte del personal de la CFPB se vuelve una tarea muy laboriosa y poco efectiva [1]. Es importante mensionar que el objetivo principal de la CFPB es recibir las quejas de todas las personas con respecto a los servicios financieros que ellos adquieren y asegurarse que los bancos, prestamistas y otras compañias brinden sus servicios de manera correcta y transparente [1]. La base de datos que se utilizó fueron al inicio de 615,273 quejas. Además de las quejas, se tenian los valores de otros campos importantes como la fecha de presentación de la queja, el código del consumidor, en nombre del banco infractor, entre otros. Dentro de la base de datos, se encontraron algunas filas que poseían el campo de la queja vacio y otras en donde se repetian la misma queja por lo que procedieron a eliminar estos tipos particulares de registros [1]. Luego de tener la data limpia, se obtuvo una nueva fuente de datos de 86,803 registros con los cuales se va a trabajar. La metodología que se utilizó consta de 5 pasos para el tratamiento de los datos antes utilizar LDA para detectar los tópicos presentes en el texto, la cual se encuentra presente en la figura 1.

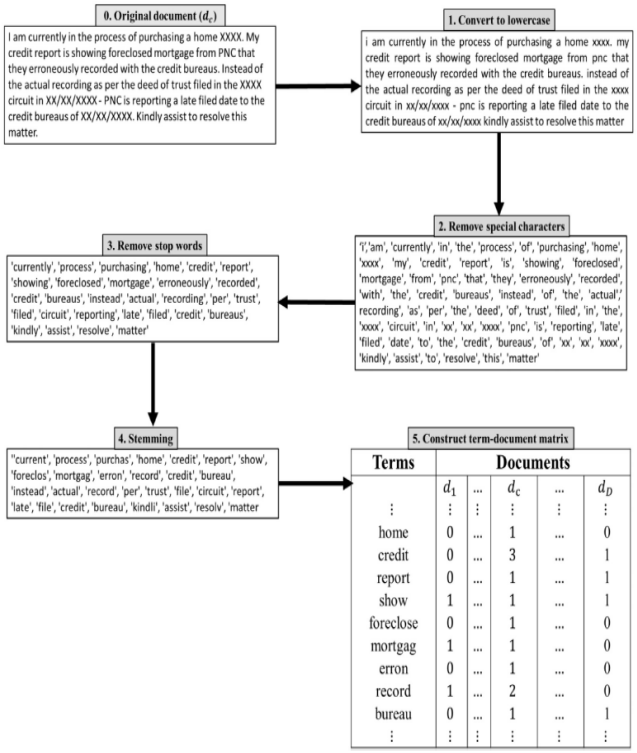

Fig. 1. Metodología empleada para el tratamiento de los datos

El paso cero consiste en la recopilación de la información. Esta data se obtuvo del portal web de la CFPB cuyos datos de las quejas de los clientes son públicos. Además, la eliminación de quejas duplicadas o en blanco. El paso 1 consiste en la conversión de todo el texto a minúsculas. El paso 2 consiste en remover todos los caractéres especiales como signos de exclamación, interrogación, numerales, entre otros y realizar tokenización a los datos. El paso 3 consiste en remover los Stopwords (palabras sin significado propio) como artículos, pronombres, etc. presentes en el documento. El cuarto paso consiste en realizar steamming a las palabras. Esta técnica consiste en obtener una palabra y quedarse solamente con la palabra raíz de la misma. Finalmente, el quinto y último paso consiste en la conversión de las palabras obtenidas a una matriz de términos en donde por cada palabra se ubica su frecuencia en cada documento.

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 K que es la cantidad de tópicos que uno desea extraer del texto [1]. Para el caso de los hiperparámetros, se utilizaron los valores de 0.1 tanto en alpha como en etha y un valor de K = 40. Por otro lado, los valores de N y D son la cantidad de palabras únicas en el documento y la cantidad total de documentos respectivamente. Además, W(d,n) representa cualquier palabra n dentro del documento d y Z(d,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.

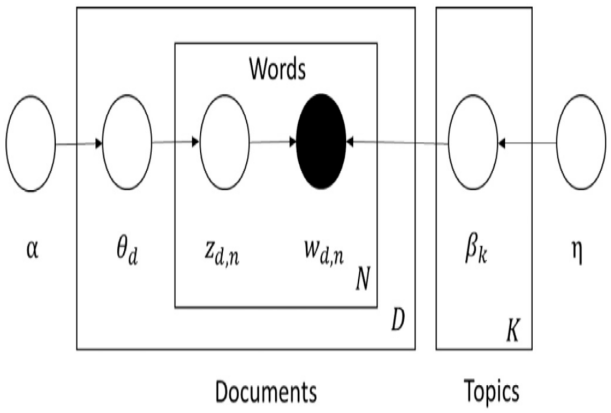

Fig. 2. Latent Dirichlet allocation (LDA) para detección de tópicos

Al implementar el algoritmo de LDA, cada uno de los 40 tópicos diferente era una colección de 10 palabras. A cada tópico se decidió darle una etiqueta en particular de manera manual para que sea más entendible por el usuario. Los resultados que se obtuvieron fueron presentados utilizando la herramienta de BI Tableau en donde cada usuario podía ingresar a la aplicación y observar por ejemplo por cada banco cual era el o los tópicos de quejas que más se repetian y así la persona podría decidir si desearia comprar un producto o servicio de un banco determinado. Por otro lado, otro reporte fue el agrupamiento de bancos o compañias en función al tipo de tópico, de tal forma que la CFPB podría darle prioridad a regular los servicios ofrecidos o productos ofrecidos por determinados bancos en función al tópico al cual pertenecen.

B. Analysis of Political Sentiment Orientations on Twitter

Este segundo avance consiste en la recopilación de tweets en la india con el objetivo de crear un modelo de aprendizaje automático con el fin de determinar a qué partido político tiene inclinación un tweet en cuestión. Este proyecto se elaboró durante las elecciones del 2019 en la india. Se utilizó la plataforma de twitter dado que durante las elecciones, el número de tweets acerca de las elecciones y partidos políticos en general, suelen incrementarse. La metodología consta de 5 pasos fundamentales: recolección de la data, preprocesamiento, Anotación, extracción de características y construcción de modelos de machine learning [2]. En la etapa de extracción de la data, se recopilaron 3886 tweets que corresponden a dos principales partidos políticos. Para la identificación de la orientación de tweet a un partido político determinado, se analizaron palabra por palabra identificando si dentro del tweet tenia palabras con las iniciales del partido político o con una keyword perteneciente al partido. Con respecto al pre procesamiento, se incluyen 4 pasos principales. En primer lugar, se remueven las mensiones, emoticones, hashtags y signos de puntuación no convencionales. Posteriormente, se realiza la tokenización de cada tweet [2]. El siguiente paso consite en la eliminación de los retweets y los tweets duplicados. El tercer paso consiste en la eliminación de las keywords. El último paso consiste en la conversión de todos los token limpios de mayúsculas a minúsculas. La etapa de anotación consitió en que tres profesionales analizaran los tweets limpios y decidieran a qué partido político pertenecen. En total fueron 3 clases diferentes llamadas P0, P1 y P2 [2]. Este procedimiento se utilizó dado que se necesita, en estos casos, de expertos que puedan identificar el sentimiento que posee un tweet con respecto a un partido político. Con respecto a la etapa de extracción de características, se utilizó TF-IDF, el cual es una medida estadística significativa utilizada para convertir un recopilación de documentos en bruto en una matriz en forma de frecuencia de término frente a frecuencia de documento inversa asignando un peso a cada palabra de los documentos [2]. Para la última etapa de construcción de modelos de machine learning, se construyeron 5 modelos diferentes: Support Vector Machines (SVM), Decision Tree Classifier (DTC), Logistic Regression (LR), Random Forest Classifier (RFC) y Long Short Term Memory (LSTM) [2]. Para la evaluación de los modelos, se utilizaron las métricas de Precision, Recall, F1 Score y Accuracy. Los principales resultados de el presente trabajo, fueron que el 55.46% de los tweets mostraron un sentimiento hacia la clase P0 y la clase con menor referencia fue la clase política P2 con un 2.07% [2]. Por otro lado, de los 5 modelos utilizados, el mejor modelo usando Tf-idfi fue Long Short Term Memory (LSTM) con un accucracy de 75%. Las conclusiones generales de este proyecto fueron que se lograron resultados prometedores con LSTM y Random Forests. Por otro lado, para la mejora de este tipo de análisis e inferencia, el número de tweets extraídos debe incrementarse a un tamaño bastante razonable. Así como se debe realizar un balance a los tweets para que se tenga la misma cantidad de tweets por cada clase [2].

C. Sentiment Analysis of US Airlines Tweets using LSTM/RNN

En este tercer trabajo se busco clasificar los sentimientos de los tweets sobre vuelos de aerolíneas con la finalidad de tener un insight sobre la satisfacción de los usuarios con respecto al servicio brindado. Para ello, se exploro los modelos de word embedding Word2vec y Glove en los tweets usando modelos de deep learning como Redes Neuronales Recurrentes (RNNs) y Long-Short Term Memory (LSTMs) obteniendo como resultado una precisión de clasificación mejor habiendo entrenado en la partición 80/20.Las redes sociales como Twitter han crecido día a día producto de los comentarios de los consumidores que expresan sus conformidades o disconformidades sobre varios producto o servicios. Muchos organizaciones atraen clientes con el uso de redes sociales medios de comunicación, es por ello es importante saber sobre la opinion de los consumidores para tomar decisiones mas inteligentes. Por otro lado, Deep Learning es una area del machine learning que tiene resultados excepcionales en tareas como reconocimiento de voz,texto y procesamiento de lenguaje natural (NLP). En este caso particular, una de las arquitecturas de deep learning, las redes neuronales recurrentes, son buenas para el tipo de dato secuencial, es por ello que se experimenta con estas redes.

LSTM es una red neuronal que contiene celdas de memoria que le permiten aprender las dependencias a largo plazo de una secuencia de palabras. Con respecto a la base de datos, se lograron recolectar 14640 tweets de 6 diferentes aerolíneas como Virgin America, United, US Airways, Delta y Southwest dataset obtenidos de Kaggle donados por CrowdFlower en forma de CSV.

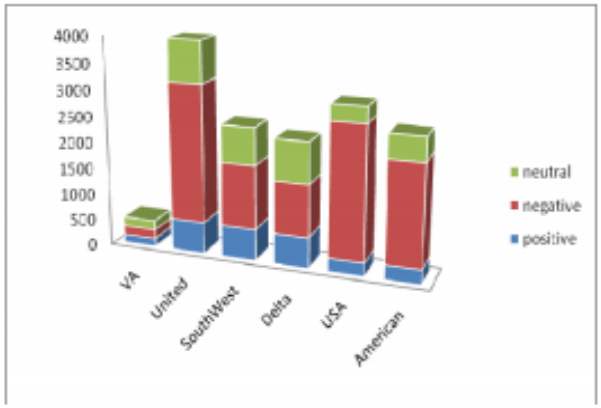

Fig. 3. Distribucion de los comentarios negativos, positivos y neutros

Como se visualiza en la Figura 3, el porcentaje de tweets negativos es mayor que los positivos en 91.78, podemos ver que los tweets positivos son los que tienen mayor cantidad los tweet negativos son los que tienen menor cantidad y los intermedios son los neutrales.

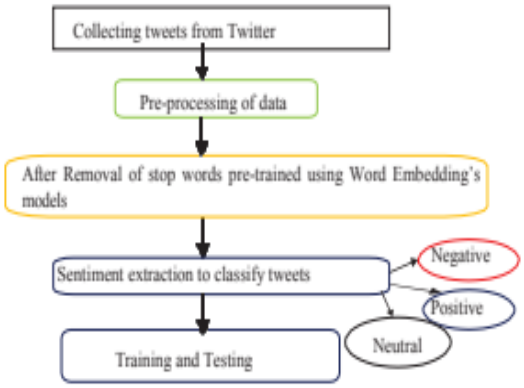

Fig. 4. Metodología del presente paper

La metodología de la Figura 4, se basa en usar en embeddings como el Glove que es una representación de palabras en vectores que tienen palabras incrustadas en los files del embedding, y sera comparado con Word2vec que son vectores de palabras en comparacion de Word2vec, los vectores de palabras estan en el forma de red neuronal poco profunda que intenta predecir la palabra pero en modelo GloVe hay una matriz de objetos calculados.Se siguieron los siguientes pasos, coleccionar la data de los tweets de tweeter luego de remover stop words pre entrenar usando modelos de word embeddings y luego la extracción de sentimientos para clasificar tweets en negativo, positivo y neutral y al final el entrenamiento y pruebas. En resumen, el texto recopilado de las redes sociales de preprocesado y limpiando, el modelo de incrustación de palabras aprende palabras, representaciones como vectores y mediante el uso de LSTM se aplica para la predicción de secuencia de palabras en oraciones.También se definió que una RNN es una clase de redes neuronales que recuerdan sus entradas debido a una memoria interna que involucra secuencia de datos, tales como, texto, genomas o datos numéricos de series de tiempo. Recordando palabras anteriores para predecir las siguientes. La palabra de la secuencia es bastante difícil en una tradicional red neural pero en RNN resuelve este problema con la ayuda de capas ocultas recuerdan información de secuencia a través de tiempo para producir la salida. Mientras el algoritmo LSTM es introducido como una solución para resolver el problema de gradiente sobre múltiples pasos de tiempo. Tiene tres reuniones y unidades multiplicativas.La arquitectura consta de un embedding de 2 capas dropout y una capa densa que devuelve la clasificación de las tres categorías y se puede ver a continuación en el siguiente gráfico. Para este trabajo se uso Pandas, la biblioteca Keras en el cuaderno Jupyter experimento con 200 dimensiones vectores de guantes. Panda es una biblioteca de código abierto que proporciona alto rendimiento para cargar herramientas de análisis y conjunto de datos

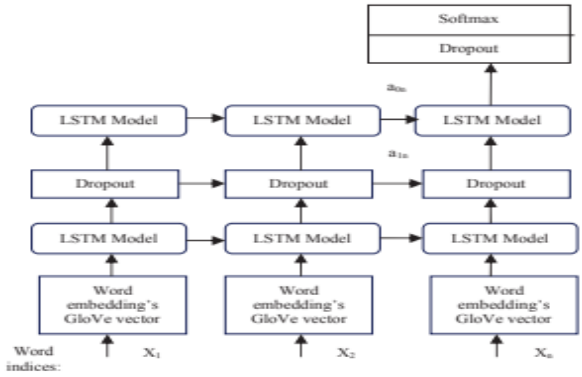

Fig. 5. Glove Embedding + LSTM

D. Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment Analysis

Las redes neuronales de aprendizaje profundo han logrado un progreso significativo en el área de análisis de imágenes y videos. Este éxito de las redes neuronales se puede dirigir hacia mejoras en la clasificación de sentimientos textuales.Sin embargo, las redes neuronales artificiales y las redes neuronales convolucionales no pueden analizar data secuencial como videos, dialogos o comentarios. Para ello un modelo de deep learning muy eficaz salio a la luz : redes neuronales recurentes que permiten el entrenamiento con data correlacionada (secuencia de datos) y realizar diversas tareas como analisis de sentimientos,etc. En este artículo, se describio una arquitectura que se baso en el uso de kernel combinado

de múltiples ramas de la red neuronal convolucional (CNN) con capas de memoria a corto plazo (LSTM). Obteniendo mayor precisión reportada en el conjunto de datos de opinión de revisión de Internet Movie Database (IMDb).La nueva arquitectura combina varias ideas utiles de varias redes y se inspiraron en una porcion del modelo de Google Inception, para lo cual usaron varias ramas(branches) de convolucion y agregaron capas LSTM al final, superando a otros modelos de machine learning en el mismo conjunto de datos de película, obteniendo al final resultaos superiores al 89% y logrando un mejor performance hasta la actualidad.

La base de datos que se utilizo para aprender vectores de palabras es la del ACL Internet Movie Database (IMDb). Dicha base de datos consta de 100,000 revisiones textuales de películas; la mitad (50,000) de las revisiones son para pruebas y no tienen etiqueta. Las otras (50,000) revisiones se combinan con una etiqueta de 0 o 1 para representar el sentimiento negativo y positivo, respectivamente. Las etiquetas son producto del mapeo del sistema de clasificación de estrellas de IMbd que permite la calificación por parte de los revisores del 1 al 10.Como podemos ver en la Figura 7, la base de datos que se uso esta balanceada siendo su distribución:12,500 comentarios positivos y 12,500 comentarios negativos para mantener los datos equilibrados.

La metodología empieza por el preprocesamiento de los comentarios empieza usando la libreria de Keras que toma las revisiones y las codifica en una secuencia de índices de palabras de acuerdo con un diccionario de las D palabras usadas con más frecuencia en el conjunto de datos, donde se proporciona D. Luego,las revisiones se rellenan para ajustarse a la longitud máxima de secuencia deseada.Después, las revisiones más largas se truncan y las revisiones más cortas se rellenan con ceros. Al final, la primera capa de la red neuronal convierte los índices en incrustaciones de dimensión E.Para este trabajo se limito el diccionario a 5,000 palabras con una secuencia máxima de 500 palabras. Los índices se integraron en 32 dimensiones.

	Positive	Negative
Training	12,500	12,500
Validation	12,500	12,500

Fig. 6. Distribución de la data de entrenamiento y validación

El método propuesto en este documento utilizo un CNN y un LSTM en la clasificación a nivel de palabra del conjunto de datos de opinión de revisión de IMDb. La novedad de la red propuesta radica en tener núcleos combinados a través de múltiples rama(branches) que aceptan los datos y realizan convolución. La salida de las ramas CNN se alimenta a un LSTM antes de concatenarse y enviarse a una capa completamente conectada para producir una única salida final. La red está entrenada y probada en batches entre 16 y 128. La arquitectura propuesta se puede visualizar en la Figura 8

La primera capa de la red acepta las revisiones de las pelicualas como una secuencia de índices e integra cada palabra en un vector de un tamaño específico.La capa de embedding es una matriz de pesos entrenables que, mediante la multiplicación de matrices, produce los vectores para cada índice de palabras. Posteriormente,la salida de la capa de embedding se da a cada rama(branch) que varian de 3 a 6 branches. Cada branch(rama) comienza con una capa de convolución c de tamaño variante entre 2 a 7, siendo el tamano del kernel 32 x c(c=2,3,4,5,6,7). Posteriormente, se aplica una activación de unidad lineal rectificada (ReLU) de la salida de la capa CNN reemplazando cualquier salida negativa con cero e introducir la no linealidad en la red. La salida de esta capa tiene la misma forma que la forma de entrada.Luego cada rama experimenta la aplicacion de un pooling, en este caso el tipo maxpooling cuyo resultado es una versión reducida de muestra de la entrada y reduciendo asi el sobreajuste.Luego se aplica a cada branch un capa de dropout que de manera aleatoria establece una parte de las entradas en 0 con la finalidad de prevenir el overfitting y permite generalizar la red para que no se centre en piezas de entrada específicas. Luego le sigue una capa de de batch normalization que normaliza la distribucion para cada batch despues del dropout y asi permitir que la convergencia llegue mas rapido.La capa final para cada rama es una capa LSTM con un número especificado de unidades.Se hizo uso del LSTM debido a la naturaleza de los datos secuenciales. La persistencia de la capa permite que el conocimiento de la entrada anterior (combinaciones de palabras enrevesadas) influya en la entrada posterior. Cada branch se fusiona al final por concatenación. Las salidas de las capas LSTM se combinan juntas en una matriz. La forma de salida de esta capa es igual a la suma de la salida de todas las ramas. La última capa es una capa fully-connected desde la entrada concatenada a una única salida. A la capa le sigue una función de activación sigmoidea simple para conformar la salida entre 0 y 1. El rendimiento final es una salida única. La compilacion de la red se realiza con una función de binary crossentropy que permite la clasificacion binaria calculando la pérdida con dos clases (0 y 1). Se denomino como 0 a un sentimiento negativo y 1 a uno positivo.La red se compila con los siguientes optimizadores: Adam, RMSprop y Stochastic Gradient Descent (SGD) . Cada optimizador se utilizó con diferentes tasas de aprendizaje y parámetros de disminución de la tasa de aprendizaje. Como concluciones, las diferentes ramas de convolución pudieron extraer información significativa de la información textual en una red poco profunda. Además, las capas LSTM permitieron usar la información para profundizar en la clasificación de las revisiones. La precisión del modelo propuesto de mejor rendimiento supero los modelos publicados anteriormente y mejora enormemente el modelo base CNN + LSTM. Si bien el sobreajuste fue un desafío para la mayoría de las versiones del modelo, las capas y los parámetros adecuados pudieron reducir el deterioro y alcanzar nuevos niveles de precisión en el conjunto de datos.

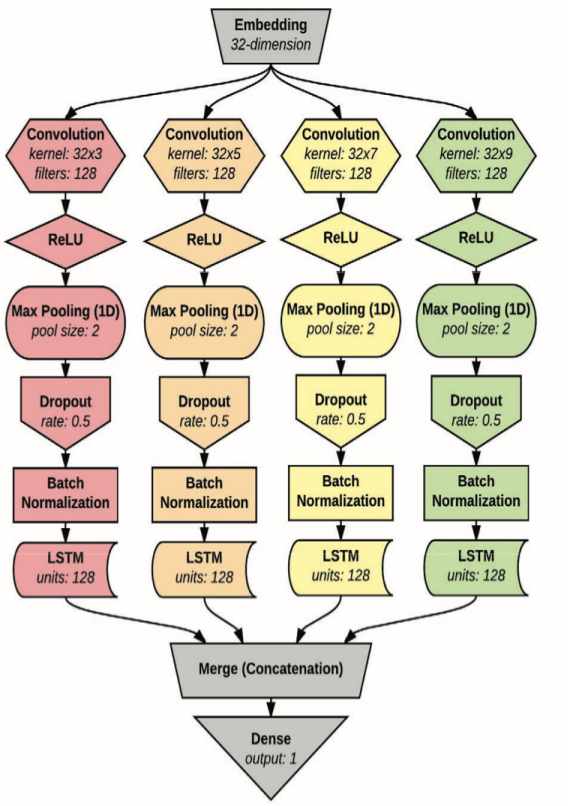

Fig. 7. Metodologia CNN+LSTM

Los resultados se pueden visualizar en la Figura 9, como podemos ver la capa LSTM de cada branch tiene 128 unidades, en caso se aumentara o reduciera una unidad se llega a reducir la precisión o aumentar el sobreajuste. El optimizador que mostro mejores resultados fue el RMSprop. Ademas el learning rate se establecio se subio en 0.01 y se disminuyo en 0.1 durante los experimentos. La red propuesta de mejor rendimiento alcanzó un 89,5% de precisión. La precisión alcanzada superó los resultados de otros modelos en el mismo conjunto de datos de opinión de revisión de IMDb. Tanto el modelo LSTM tradicional como modelos hibridos basados en aprendisaje supervisado o no supervisado. Uno de los problemas mas comunes en este tipo de arquitecturas era el overfitting.

Proposed Models	Convolution			Activation	Max Pooling	Branch Dropout	Batch Normalization	LSTM	Merge Dropout	Optimizer			Accuracy
	Branches / Kernel Sizes	*Filters*	*Kernel Regulaizer*	*Type*	*Pool Size*	*Rate*	*Present*	*Units*	*Rate*	*Type*	*Learning Rate*	*Learning Rate Decay*	*Maximum*
Model_08	3/4/5	32	None	ReLU	2	0	yes	100	0	Adam	0.001	0	0.8922
Model_16	3/4/5	32	L2(0.01)	ReLU	2	0.5	yes	100	0	Adam	0.001	0	0.8934
Model_43	2/3/4/5/6/7	128	L2(0.01)	ReLU	2	0	yes	128	0.8	Adam	0.001	0	0.8914
Model_57	3/5/7/9	128	L2(0.01)	ReLU	2	0.5	yes	128	0	RMSprop	0.001	0	0.8936
Model_63	3/5/7/9	128	L2(0.01)	ReLU	2	0.5	yes	128	0	RMSprop	0.01	0.1	**0.895**
Base Model	5	64	None	ReLU	4	0.25	no	70	0	Adam	0.001	0	0.8498

Fig. 8. Resultados de los 5 modelos de CNN+LSTM

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 del presente trabajo de investigación se encuentra presente en la Figura 9, Figura 10 y Figura 11.

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. 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 o hashtag 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. 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 mensionar que la API de twitter permite hacer la extracción solamente de los 7 últimos días, por lo que se hizo la extracción de manera díaria de 20000 en 20000 tweets utilizando hashtags relacionados al coronavirus como ”covid-19”, ”vacuna”, ”bono”, ”QuedateEnCasa” entre otros. Una vez extraidos los tweets, se procedió a convertir los resultados en 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

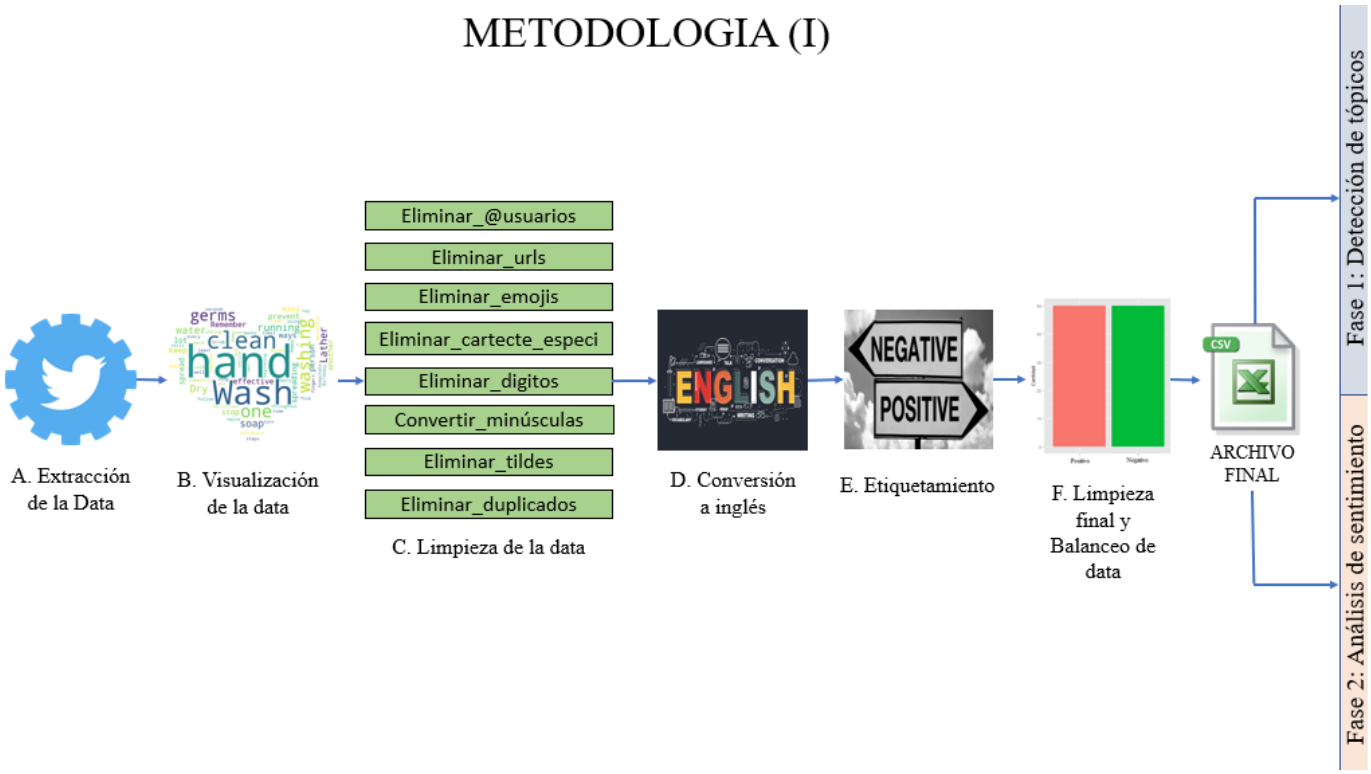

Fig. 9. Metodología del trabajo de investigación-I

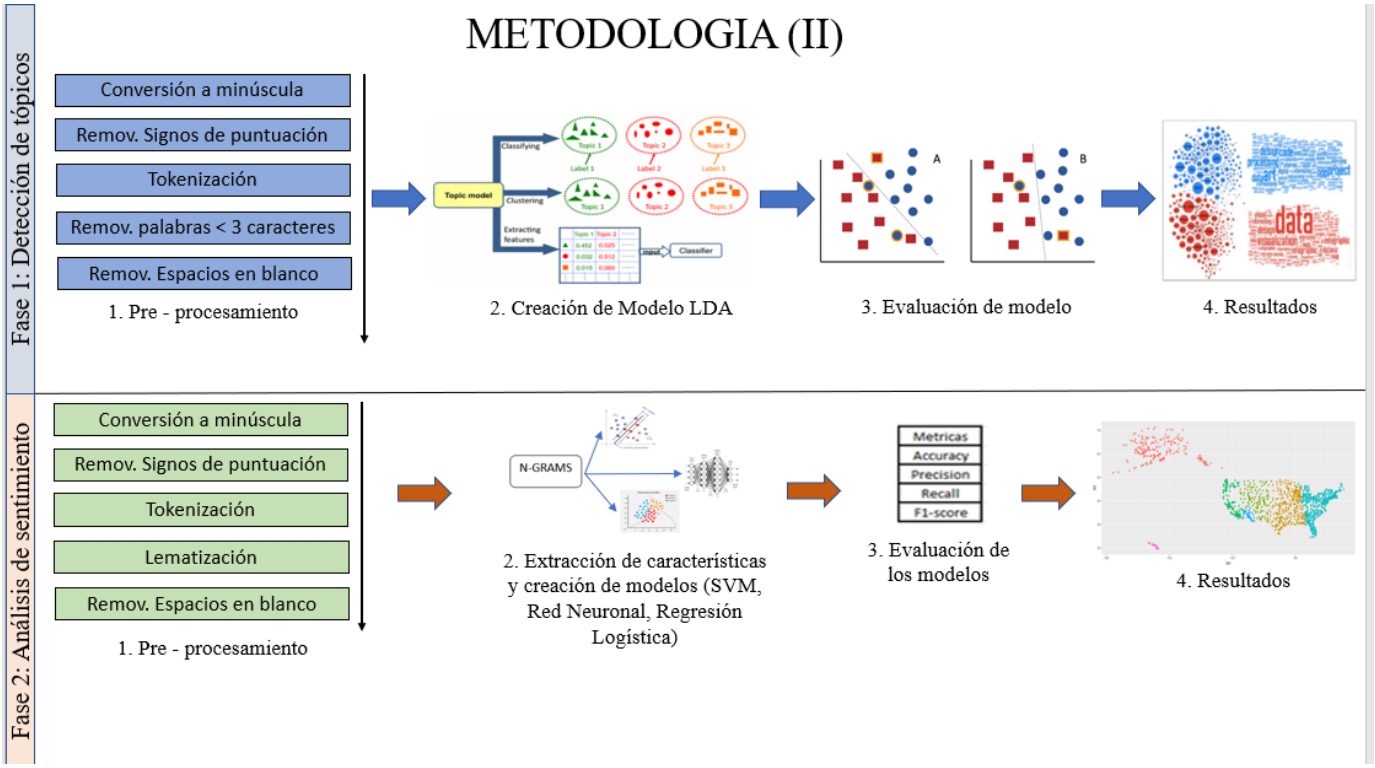

Fig. 10. Metodología del trabajo de investigación-II

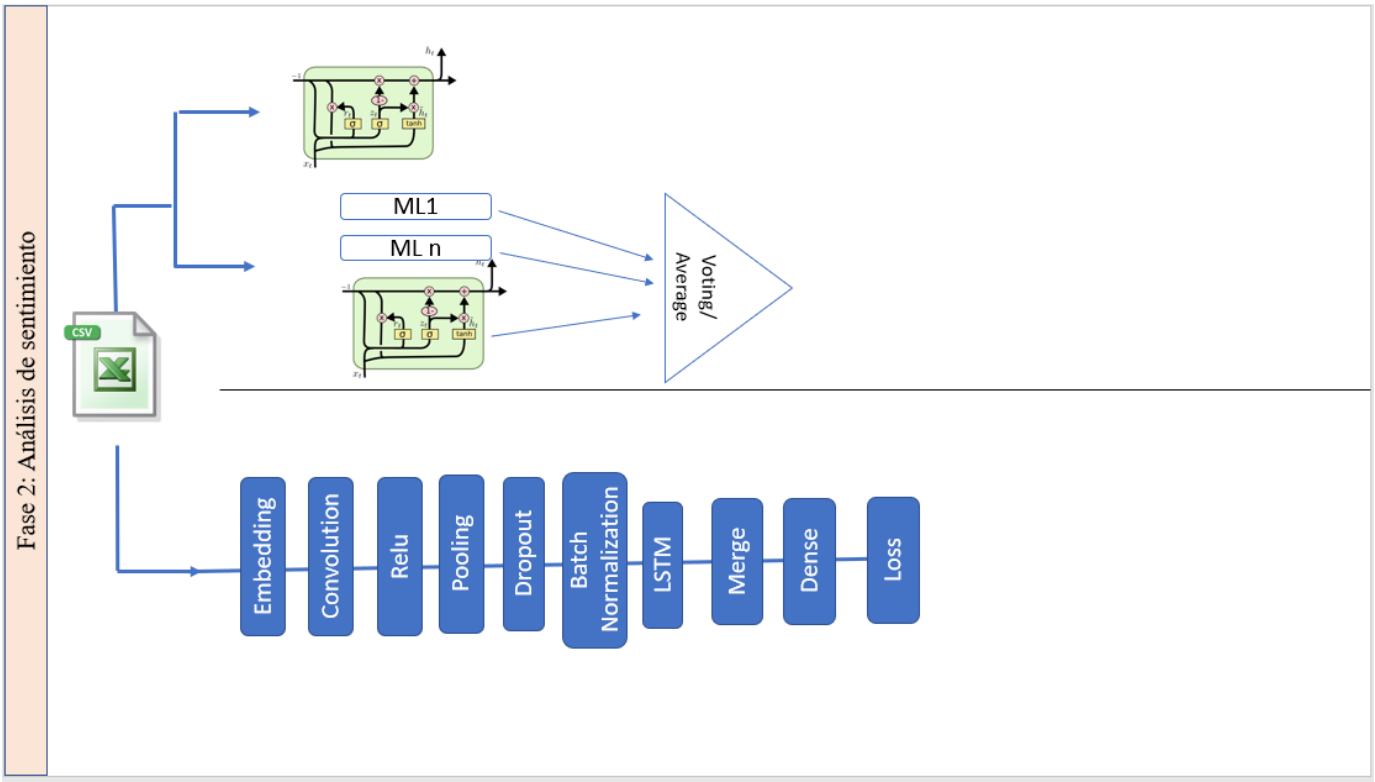

Fig. 11. Metodología del trabajo de investigación-III

el nombre del hashtag con el cual se extrajo para un mejor manero de los archivos. En total, se recopilaron 137874 tweets por un periodo de 6 semanas usando diversos hashtags relacionado a coronavirus en el Perú.

B. Visualización de la data

a data paso por un proceso de visualización que consistió en evaluar la distribución de los tweets con la finalidad de determinar si estaba correctamente balanceada, teniendo como resultado 45958 tweets positivos y 45958 negativos. Luego, se obtuvo la frecuencia del top 20 de palabras mas repetidas en todos los comentarios siendo en este caso palabras como el, sobre, dentro, coronavirus.Posteriormente, se realizo dos wordclouds para ver que palabras se repetían mas en las dos categorías (positivo y negativo), siendo coronavirus,casos, paises, hoy, govierno en el caso de los positivos y coronavirus, muertes,pandemia,covid en el caso de los negativos. Luego se removió los stop-words con la finalidad de eliminar palabras que no tengan mucho significado para el analisis de los comentarios y asi para poder visualizar cuales eran las palabras mas repetidas en todos los comentarios.Los resultados del del top 20 de palabras mas repetidas en este caso fue coronavirus, new, covid, muertes, personas, hoy, nosotros a nivel general y segmentandolo en categorias se obtuvo palabras como: coronavirus, nuevos, covid, hoy ,salud,bueno, nosotros, dia en los tweets positivos y casos, muertes, infectados, personas en el caso de los comentarios negativos.En el caso de los wordclouds se obtuvo palabras como hoy, casos, personas, nosotros, coronavirus en el caso de los comentarios positivos y en el caso de los negativos tenemos palabras como mundo, coronavirus, doctor, necesitamos y muerte .

C. Limpieza de la data

En la limpieza de los datos, se elimino los usuarios y los hashtags con la libreria re . Luego se procedio a eliminar los puntos, tildes y caracteres como simbolos de exclamacion,slash y numeros con la libreria string, re y unicode. Luego se elimino los emojis con la libreria demoji con la finalidad de dejar la data como un comentario sin ningun ruido o caracter extrano.

D. Conversión a inglés

Se convirtio la data en ingles haciendo uso de la api de google llamada googletranste, que es una biblioteca de Python gratuita e ilimitada que implementó la API de Google Translate. Esto utiliza la API de Google Translate Ajax para realizar llamadas a métodos como detectar y traducir. Sus caracteristicas son:Rápido y confiable,utiliza los mismos servidores que translate.google.com,detección automática de idioma,traducciones masivas,URL de servicio personalizable, etc.

E. Etiquetamiento

Para el etiquetamiento de la data se hizo uso de la libreria senti_classifier que esta basado en word sense disambiguation usando wordnet del corpus de reviews y que devuelve scores

en base a dos categorias positivo y negativo. Se implemento un algoritmo que compara los dos scores y determina la etiqueta a la cual pertenece, siguiendo la siguiente logica. Si el score positivo es mayor que el negativo, se determina como positivo, en caso contrario es negativo y en el caso que los dos scores sean iguales es neutro.

F. Limpieza final y balanceo

Luego de obtener los 91916 tweets en ingles, etiquetados y almacenados en un archivo llamado DATA_FINAL.csv, se procedió a realizar una limpieza antes de distribuir el archivo csv tanto para la fase 1 y para la fase 2. Esta limpieza consistió en realizar tokenización, remover stopwords, realizar lematización ,conversión a minúsculas y eliminación de signos de puntuación. Para la realización de estas dos últimas actividades descritas anteriormente, se utilizó la función utils.simple_preprocess() que se encuentra en la libreria gensim. Se obtuvieron nuevas oraciones limpias que en total fueron 91916. No obstante, se verificó que luego de la limpieza, habian tweets que habian quedado vacios por la limpieza y conservaban sus respectivas etiquetas. Por lo que se verificó cuántos tweets habian quedado vacios luego de la limpieza y en total fueron 164 tweets, los cuales fueron removidos. Luego se procedió a verificar la cantidad de tweets positivos y negativos los cuales fueron 45915 y 45837 respectivamente. Para balancear los datos, se eliminaron 78 tweets positivos, de tal modo que quedaron 91674 en total de los cuales 45837 fueron positivos y 45837 fueron negativos. Una vez obtenida la data balanceada, se exportó a un nuevo archivo csv llamado DATA_FINAL_LIMPIA la cual es la data que se utilizará para construir los modelos tanto para la fase 1 como para la fase 2.

G. Fase 1: Detección de tópicos

En esta sección se procederá a explicar de manera detallada el procedimiento para crear los 2 modelos de LDA para realizar la detección de tópicos y cómo evaluar cada modelo para saber qué modelo es mejor.

1) Modelo LDA usando Bag of words tradicional: Para realizar este primer modelo de LDA, 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 engrenaaiento. Para la evaluación del modelo, se utilizó las métricas de Perperxity y Coherence score.Para el calculo 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 nefativo 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 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.

H. Fase 2: Análisis de sentimiento

En esta sección se procederá a explicar de manera detallada el procedimiento para crear los modelos de machine learning, de deep learning y un modelo de machine nearling más deep learning.

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 llamda vectorizer. Para la creación de la matriz, se utilizó vectorizer.fit_transform y se les dio los tweets pre procesados. El resultado de aplicar esa función, fue una matriz cuyas dimensiones fueron de 91674 filas x 36284 columnas. 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 reliazó un fit.transform de tal modo que dio como resultado una matriz utilizando unigrams hasta trigrams que tuvo como dimensiones 91674 filas x 675271 columnas. 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 variabler 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 una matriz de 91674 filas x100 columnas. 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. En el presente proyecto de investigación, se utilizarán 4 diferentes modelos: RandomForest, una red neuronal, KNN y regresión logística y se crearán 3 modelos diferentes del mismo modelo utilizando las 3 diferentes matrices creadas, es decir que en total hubrán 12 diferentes modelos. Empezando por el 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 segundo 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. Para la construccion del modelo de KNN se le dio el parámetro de n_neighbors=5 que viene a ser el número de vecinos con los cuales trabajará el modelo y para 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. Elaborados los 12 modelos, 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.

2) Creación de modelos de Deep Learning: Una de las redes neuronales mas eficientes a la hora de clasificación de texto es la red neuronal recurrente. La red neuronal recurente a diferencia de otras redes, puede analizar secuencia de datos como vídeos, diálogos, comentarios.Esta red es buena por que puede usar datos con tamaño variable, puede analizar datos corelaccionados,etc.Algunas de las arquitecturas de RNN son : One to many(image captioning), many to one (clasificación de sentimientos), many to many (traductores automaticos).Por otro lado como aplicaciones tenemos: conversores de voz a texto, reconocimiento de escritura(LSTM), analisis de sentimientos, indexación de videos, generación de musica, detección de modificaciones en la secuencia de ADN. Para el presente trabajo de investigación se diseño varias redes neuronales recurentes que se diferenciaban unas de otras por la cantidad de bloques LSTM que tenían. Primero se realizo un one hot encoding a las clases de los tweets con la finalidad de que se puede expresar en terminos de 1 y 0 y asi realizar la clasificacion. Para ello se uso la libreria LabelEncoder y las funciones to_categorical. Posteriormente, se realizo la particion de la data en train y test, siendo la distribucion 80/20. Luego se calculo el tamano maximo de comentarios. Usamos la libreria Tokenizer que permite convertir la secuencia de strings en una secuencia de numeros, y le damos como parametro el numero maximo de palabras unicas que estan en el entrenamiento, este numero representa el numero maximo de palabras del vocabulario.Luego se usa la funcion fit_on_text que permite calcular la frecuencia de cada palabra en el corpus. Posteriormente usamos la funcion texts_to_sequences que convierte el array de strings en una secuencia de numeros. Para finalizar, convertimos la lista en una matriz usando el pad_sequences y le damos el tamano maximo de la sequencia como parametro(en este caso se obtuvo el tamano del comentario mas grande, siendo 166 en el trabajo de investigacion. Luego se procedio a difinir la arquitectura de la red neuronal recurrente, siendo en este caso LSTM. Primero se importo la funcion Sequential que permite la creacion de la RNN, luego se definio una capa Embedding que permite representar las palabras de forma numerica y mantener dichas palabras similares juntas en un espacio vectorial. Los parametros que se definieron en esta capa Embeding fueron: input_dim que especifica el numero de filas de la matriz embedding, output_dim que define el numero de columnas de las matriz embedding y el input_length que define el tamano maximo de la secuencia de entrada. Posteriormente se definio una capa del tipo de LSTM con 256 unidades o 256 celdas de memoria para que haga match con el tamano del output del embedding layer. Luego se definio una capa dropout para que se elimine el overfitting, cosa que le sucede mucho a una LSTM.Posteriormente, se compilara el modelo con un funcion de perdida llamada binary_crossentropy, un optimizador SGD y la metrica de accuracy. Se entrenara el modelo con 100, 200 y 300 epocas y con batch sizes de 32,64 y 128. Ademas la data del train sera partida en una de validacion y entrenamiento bajo la proporcion 80/20 Los experimentos que se haran a continuacion implican aumentan los bloques de LSTM y probar con diferentes optimizadores como Stocastic Gradient Descent(SGD), Adam y RMSprop.Se evaluarán los 4 modelos utilizando el Acuracy, Precision, Recall y F1-score con el objetivo de determinar qué modelo es el más adecuado para realizar el análisis de sentimiento.

3) Creación de modelos de Deep Learning + Machine Learning: Para la creacion de modelos de deep learning en conjunto de machine leaning, se trabajara con probabilidades obtenidas de los modelos de machine learnig con los modelos de deep learning. Para ser mas precisos se hara la suma producto de las probabilidades con los accuracies de los modelos de machine learning y deep learning, para despues dividirlos entre la suma de accuracies y mediante un umbral asignarle una clase siendo esta positva o negativa. Por otro lado tambien se implementara el metodo de voting o voto mayoritario que implica determinar la clase en base a los modelos anteriormente mencionados basandonos en la moda, es decir el resultado que mas se repite.Los experimentos que se realizaran involucran mediante la tecnica de voting analizar la union de modelos de deep learning de arquitecturas multicanal, arquitecturas multicanal y LSTMs, LSTMs con embeddings pre-entrenados y LSTM's sim embeddings pre-entrenados, modelos de machine learning con modelos de machine learning, y todos los modelos. Estos mismos experimentos se repetiran para la tecnica de average con la finalidad de poder comparar el resultado entre ambas tecnicas

4) Creación de modelo de LSTM + CNN (multicanal): Para el modelo de LSTM con CNN, se usara el mismo pipeline [4] con la finalidad de hacer varios modelos que usen la misma arquitectura y evaluar el resultado con respecto a las tres metodologias anteriormente mencionadas. La arquitectura que se usara es una compuesta por: embedding, 1 convolucion 1d, Activacion Relu, Pooling, Dropout, Batch Normalization, LSTM, merge, dense.

Se realizara cambios en los kernels,funciones de activacion, y optimizadores con la finalidad de realizar varios experimentos. Se evaluarán los 4 modelos utilizando el Acuracy, Precision, Recall y F1-score con el objetivo de determinar qué modelo es el más adecuado para realizar el análisis de sentimiento. Durante los experimentos se puede modificar los kernels : 3-4-5-6-7, filters= 32, 128, 84, kernel_regulizer= L2(0.1), tipo de activacion= ReLU, pool size= 2 y 4, dropout= 0.5,0.25, batch normalization, LSTM, optimizadores= Adam, RMSprop, learning rate=0.001, decay=0. Los modelos multicanal son modelos que son entrenados en paralelo y que al final se juntan con una capa merge. Tomaremos como referencia los modelos mutlicanal de [4] con la finalidad de examinar los resultados que dicho pipeline nos da. Por ende, realizaremos

un modelo base que tiene solo 1 canal con 5 kernels, 84 filtros en la convolucion, un regulizador de kernels, funcion de activacion ReLu, un max pooling de size 2 y un dropout rate de 0.5 con batchnormalization

IV. Resultados

En esta sección, se explicarán los resultados obtenidos luego de la realización de los experimentos con el objetivo de obtener un modelo óptimo de LDA para la derección de tópicos y un modelo de Machine Learning y Deep Learning para hacer la detección de sentimientos de un tweet.

A. Fase 1: Detección de tópicos

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 12 y la Figura 13.

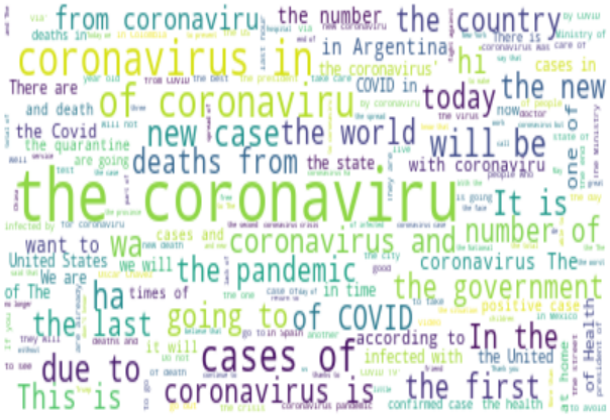

Fig. 12. Wordcloud de la data balanceada

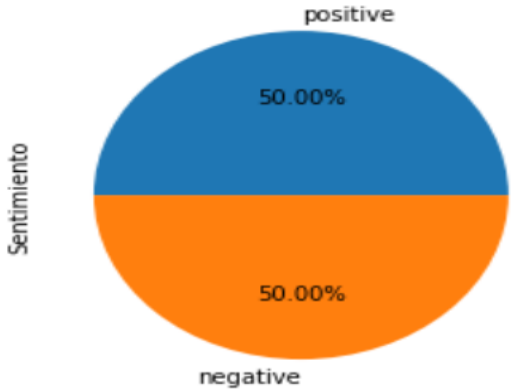

Fig. 13. Porcentaje de tweets positivos y negativos

Como se puede observar en la Figura 12, las palabras más resaltantes fueron ”coronavirus”, ”cases”, ”deaths”, ”world”, ”pandemic”, entre otros. Por otro lado, en la Figura 13, 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 0.2746 y Perplexity de -7.4734.

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 0.2780 y de Perplexity fue de -8.7840.

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 14.

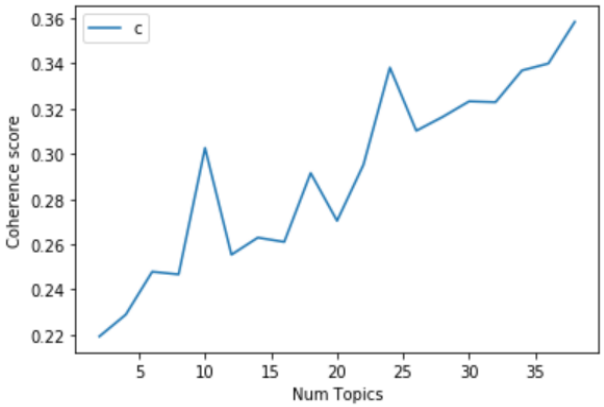

Fig. 14. Gráfica de la evolución del Coherence Score para el modelo de LDA-Doctobow

En la Figura 14, se observa que el mayor Coherence Score se da con una cantidad de tópicos igual a 24 y el valor del Coherence es de 0.3382. Sabiendo que el mejor modelo de DoctoBow tiene un K = 24, se procedió a calcular el valor de Perplexity y fue de -7.7615.

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 15, se observa el gráfico mencionado y se observa que el Coherence Score va en aumento, no se observó un pico determinado. Por tal motivo, se decidió obtener el mismo valor de K = 24 como en el modelo mejorado anterior para poder comparar los 2 mejores modelos.

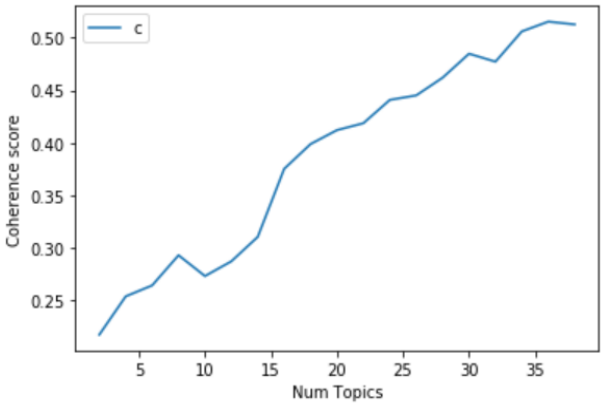

Fig. 15. Gráfica de la evolución del Coherence Score para el modelo de LDA-TF-IDF

Usando un K = 24 para el mejor modelo de LDA-TF-IDF, se obtuvo un Coherence Score de 0.4408 y una Perplexity de -9.5593. De tal modo que el mejor modelo de LDA y el cual se utilizará para realizar la detección de tópicos de los nuevos tweets será el modelo de LDA con TF-IDF con una cantidad de tópicos igual a 24.

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	0.2746	-7.4734
Tradicional LDA - TFIDF	10	0.2780	-8.7840
Mejorado LDA - Doctobow	24	0.3382	-7.7615
Mejorado LDA - TFIDF	24	0.4408	-9.5593

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.

B. Fase 2: Análisis de sentimientos

1) Resultados de modelos de machine Learning

Los modelos de machine learning que se utilizaron fueron Regresión Logistica, Random Forest y Red Neuronal. Además, por cada modelo, se creó uno por cada técnica de extracción de características (TF-IDF, NGRAMS y Word2Vect). Antes de crear los modelos respectivos, se realizó un tuneo para cada modelo probando con el train y test de tf-idf, Ngrams y Wordtovec. 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 7000, 14000 y 21000. En el caso de RandomForest, se utilizó el parámetro de n_estimators con valores de 200 y 400; max_features con valores de auto y sqrt y max_depth con valores de 4 y 5. 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 (40,3), (40,4) 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 = 7000, solver = 'newton-cg'. 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 Nueronal, se obtuvo unos valores de activation = 'logistic', hidden_layer_sizes = (40, 3), 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, se obtuvo unos valores de parámetros finales de activation = 'logistic', hidden_layer_sizes = (40, 4), 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, 3), learning_rate = 'constant'. En total, se crearon 9 modelos diferentes, cuyos resultados se muestran a continuación en la tabla III, la tabla IV y la figura 16.

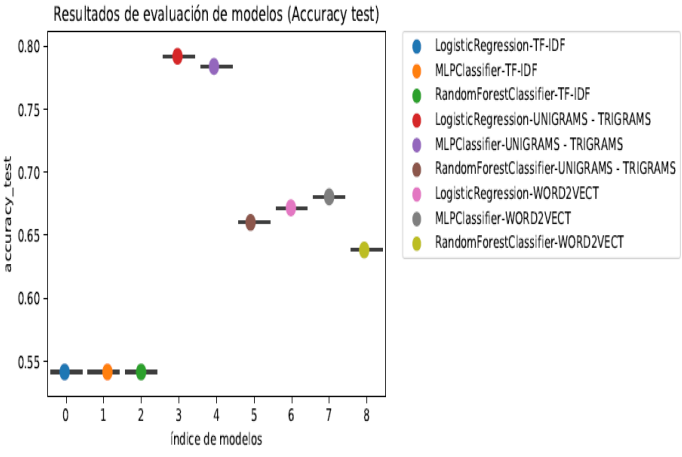

Fig. 16. Gráfica de los accuracies de cada modelo de Machine Learning

En dichas figuras antes mensionadas, se observa que con

Indice de tópico	Nombre de tópico
0	'Casos nuevos de infectados y fallecidos por coronavirus'
1	'Desafortunadas víctimas por coronavirus en el mundo'
2	'Disminución de criminalidad por coronavirus'
3	'Noticias falsas sobre disminución de tasas de muertes por coronavirus'
4	'Uso de materiales de limpieza para la prevención del covid-19'
5	'Noticias sobre posibles orígenes del Covid-19'
6	'Expectativas para el tratamiento y cura contra el coronavirus'
7	'Cancelaciones de vuelos por coronavirus'
8	'Supervivencia de extranjeros en época de pandemia'
9	'Lavado de manos como medida de prevención del coronavirus'
10	'Medidas ante el aumento de casos de covid-19 para disminuir la curva de contagios'
11	'Personas en extrema pobreza como principales afectados por el coronavirus'
12	'Suspensión de clases en los colegios para evitar contagios de covid-19 en niños y niñas'
13	'Evidencias de fallas en la prueba de descarte de coronavirus'
14	'Respuesta de América Latina a la segunda ola de coronavirus'
15	'Uso de mascarillas para protegerse del coronavirus'
16	'Discriminación por coronavirus'
17	'Noticias de muertes y crisis por coronavirus'
18	'Coronavirus: Usan el paseo de perros para salir a la calle'
19	'Quedarse en casa para protegerse del coronavirus'
20	'Personas afectadas por el coronavirus en el mundo'
21	'El peligro del coronavirus en animales'
22	'Lavado de manos para prevenir el coronavirus'
23	'Fallecimiento de sospechosos de coronavirus'

TABLE II
NOMBRE DE LOS TÓPICOS PERTENECIENTES AL MEJOR MODELO

respecto al accuracy, el mejor modelo de TFIDF, en este caso existe un empate entre los 3 modelos, por lo que para fines de comparación se escogió el modelo de RandomForest con un accuracy de 54.14 %. En segundo lugar, el mejor modelo de Ngrams, fue el de Regresión Logística con un accuracy de 79.2 %. Finalmente, el mejor modelo utilizando Wordtovect, fue la Red Neuronal con un Accuracy de 68.05 %. Además, analizando las métricas de Precision, Recall, F1-Score y Support podemos concluir que el mejor modelo de machine learning fue de Regresión Logística con Ngrams dado que los valores de las métricas antes mensionadas fueron de un 79 %.

Finalmente, para poder combinar el modelo ganador de Machine Learning con Deep Learning, se necesitaron colocar adicionalmente dos columnas dentro de los resultados de los modelos de machine learning en donde se hace la predicción utilizando el X_test y la probabilidad de predicción. Para realizar lo anterior mensionado, se realizó la predicción utilizando cada uno de los nueve modelos diferentes. Cada modelo tuvo un determinado tiempo para obtener la predicción por cada tweet dentro de X_test y su respectiva probabilidad. Los resultados de los tiempos para calcular la predicción, se presentan en la tabla V. En la tabla anterior mensionada, se puede observar que el modelo que se demoró más tiempo fue el modelo de RandomForest usando ngrams con un tiempo de demora de 4 horas y media y el modelo más ligero en encontrar las predicciones fue de Regresión logística con word2vect con una duración de 3 segundos. El motivo de los resultados mensionados es debido a la longitud de los vectores característicos. En el caso se Ngrams, el vector característico tiene una dimensión mayor que el de TF-IDF y Wordtovec por tal motivo el tiempo de ejecución es mucho mayor.

Realizadas la predicción utilizando cada modelo diferente y obtenida la probabilidad para cada predicción, se procedieron a almacenar los resultados para utilizarlos cuando se haga la combinación de un modelo de machine learning con deep learning, proceso que se detallará en las siguientes secciones.

2) Resultados de modelos de Deep Learning (DL)

Los modelos de Deep Learning estan basados en dos arquitecturas: LSTM y BiLSTM. Estas arquitecturas permiten aprovechar el contexto del tweet con la finalidad de realizar la prediccion de la clase del tweet. Podemos visualizar que los modelos de deep learning tuvieron mejores resultados que los modelos tradicionales de machine learning. En el caso del modelo llamadao LSTM_1 visualiza que los resutados oscilan entre el 90 a 91 %, en el modelo LSTM_2 que consta de 2 capas de LSTM se vizualiza que el performance en la precision decrece en 1 %, en el caso del modelo LSTM_3 se visualiza un resultado homogeneo en las cuatro metricas.En el modelo de Bi_LSTM que cuenta con los pesos cargados en su capa embedding del modelo GLoVE muestra resultados. En general los resultados de la Tabla III son optimos. El modelo LSTM_1 de la figura 17 es el que muestra el menor Loss durante el entrenamiento y la prueba con test obteniendo 26.5% de loss y 90% de accuracy y con ello se puede concluir que mas capas de LSTM no incrementa la precision del modelo.

Los resultados de los modelos de deep learnig son en su mayoría estables con respecto a las cuatro métricas(Accuracy, Precision, Recall, Fi-score) en el modelo LSTM1, el resultado de tener solo una capa de LSTM con un embedding sin inicializar(tranfer learning) es un resultado óptimo. Luego de haber probado con 2 capas de LSTM nos pudimos dar cuenta de que el resultado es mucho menos óptimo con respeto a la precisión y finalmente, el resultado de tener tres capas LSTM interconectadas demuestran un mejor resultado mas

Modelo	Técnica de extracción de característica	Accuracy Train	Accuracy Test
Regresión Logística	TF-IDF	0.5407	0.5414
Random Forest	TF-IDF	0.5407	0.5414
Red Neuronal	TF-IDF	0.5407	0.5414
Regresión Logística	NGRAMS	0.9981	0.7920
Random Forest	NGRAMS	0.6765	0.6601
Red Neuronal	NGRAMS	0.9964	0.7840
Regresión Logística	Word2vec	0.6745	0.6717
Random Forest	Word2vec	0.6445	0.6382
Red Neuronal	Word2vec	0.6927	0.6805

TABLE III
RESULTADOS GENERALES DE MODELOS DE MACHINE LEARNING (I)

Modelo	Técnica de extracción de característica	Precision	Recall	F1-Score	Support
Regresión Logística	TF-IDF	0.55	0.54	0.53	18335
Random Forest	TF-IDF	0.55	0.54	0.53	18335
Red Neuronal	TF-IDF	0.55	0.54	0.53	18335
Regresión Logística	NGRAMS	0.79	0.79	0.79	18335
Random Forest	NGRAMS	0.71	0.66	0.64	18335
Red Neuronal	NGRAMS	0.78	0.78	0.78	18335
Regresión Logística	Word2vec	0.67	0.67	0.67	18335
Random Forest	Word2vec	0.64	0.64	0.64	18335
Red Neuronal	Word2vec	0.69	0.68	0.68	18335

TABLE IV
RESULTADOS GENERALES DE MODELOS DE MACHINE LEARNING (II)

Modelo	Técnica de extracción de característica	Tiempo
Regresión Logística	TF-IDF	00:06
Random Forest	TF-IDF	10:31
Red Neuronal	TF-IDF	00:07
Regresión Logística	NGRAMS	00:07
Random Forest	NGRAMS	4:28:26
Red Neuronal	NGRAMS	00:08
Regresión Logística	Word2vec	00:03
Random Forest	Word2vec	09:42
Red Neuronal	Word2vec	00:07

TABLE V
RESUMEN DE TIEMPOS PARA EL CALCULO DE LA PREDICCIÓN POR CADA MODELO

homogenio que las versiones anteriores. Habiendo realizado los experimentos con word embedding pre entrenados(Glove) nos pudimos dar cuenta que nos dieron mejores resultados a comparación de los modelos que tuvieron un embedding sin pre entrenar. Obteniendo el mejor resultado con el 91.25 de accuracy y siendo comparable a los modelos multicanal. El mejor modelo de deep learning con presos no entrenadoses el LSTM con tres capas y el mejor modelo de deep con presos pre entrenados es el Bi LSTM.

3) Resultados de modelos de Deep Learning con machine learning Los resultados bajo el método voting indican que la unión de modelos de deep learning da un resultado robusto ya que podemos ver que las cuatro métricas son prácticamente homogenias en las mayoría de los casos es decir hay una variación de 1 por ciento, es decir la variación no es de mas de 1 o 2 por ciento entre métricas.Con respecto a los modelos de machine learniing se pudo determinar que dichos modelos al unirse no son robustos ya que sus resultados son demasiado diferentes. Con respecto a la unión entre modelos de machine learning y deep learning podemos ver que los resultados no son no tan altos ni mas bajos, siendo resultados promedio.Finalizando podemos visualizar que la unión de todos los modelos de machine learning con deep learning nos dan un resultado promedio.Por otro lado los resultados, de aplicar la técnica average entre los modelos de deep learning nos dan mejores resultados, siendo estos mayores en uno o dos por ciento, gracias a que se esta ponderando con sus respectivos accuracies y probabilidades.Con respecto a ;los modelos de machine learning podemos ver que el resultado sigue siendo mayor al de la técnica voting en uno o dos por ciento. Con respecto a la unión de todos los modelos de machine learning y deep learning por el método average podemos visualizar que son mejores resultados en comparación con el método voting en aproximadamente 1% por ciento.Con respecto a los modelos multicanal podemos ver que tienen un resultado robusto a comparación de los modelos de machine learning. Aplicando la técnica de voting podemos ver que su resultados no bajan del 99 por ciento en su mayoría en las cuatro métricas. Mientras que los resultados bajo la métrica average son mejores y no bajan del 91 por ciento en su mayoría.

4) Resultados de modelos multicanal (CNN+LSTM) Con respecto a las arquitecturas multicanal se siguio el pipeline encontrado en [4] con la finalidad de evaluar el performance de dichas arquitecturas en la base de datos de covid propia. Para ello, cabe destacar que los modelos cuyo optimizador era Adam se overfitteaba luego de la cuarta época, siendo esta una conducta del proceso de entrenamiento de la CNN-LSTM presente en el paper de investigación[4]. Mientras que los modelos con el optimizador RSMprop fueron los modelos con mejores resultados a excepción del modelo 1 (MM1) donde se obtuvo los peores resultados. El modelo con mejor performance alcanzo un 91.2 % de accuracy. Este

Modelo	Accuracy	Precision	Recall	F1-score
LSTM1	90.21	91	90	90
LSTM_2	90.37	89	90	90
LSTM_3	90.5	90	90	90
Glove+Bi_LSTM	91.25	0.92	0.9	0.91
Glove+LSTM	91.1	0.92	0.9	0.91

TABLE VI
RESULTADOS DE LOS MODELOS DE DEEP LEARNING

Modelo	Tecnica	Accuracy	Precision	Recall	F1-score
model_57 model_63 model_08 MM_2	Voting	0.9135	0.93	0.89	0.91
	Average	0.9127	0.92	0.91	0.91
model_base model_43 MM_2 MM_1	Voting	0.9017	0.89	0.92	0.9
	Average	0.91	0.93	0.89	0.91
model_base MM_1 lstm_3 model_8	Voting	0.8901	0.89	0.89	0.89
	Average	0.9125	0.93	0.89	0.91
LSTM_1 LSTM_2 LSTM_3 GLOVE BI_LSTM GLOVE LSTM	Voting	0.8901	0.89	0.89	0.89
	Average	0.9101	0.91	0.91	0.9
ML models	Voting	65.83	65	64	65
	Average	66.1	66	65	67
ALL MODELS	Voting	85.5	86	85	85
	Average	86.7	86	86	86

TABLE VII
RESULTADOS DE LOS MODELOS DE DEEP LEARNING Y MACHINE LEARNING BAJO LAS TECNICAS DE VOTING Y AVERAGE

resultado se obtuvo sin importar el desafio del overfitting mencionado anteriormente. Los demas modelos optimizados con el optimizador RSMprop, a excepcion de 1 reducieron el overfitting y generalizaron el aprendizaje. La figura 18 Y 19 muestra el performance del mejor modelo respecto a su loss y accuracy. Esta arquitectura muticanal esta basada en [4] no obstante se le incluye una variacion despues de la capa LSTM, cuyo nombre es Dropout. Esta capa es generalmente usada despues de una capa LSTM con la finalidad de reducir el overfitting, no obstante en el pipeline presente en [4] no figura esta variacion en ninguno de los casos. En terminos generales las capas pudieron ayudar a recudir el overfitting son: Max Pooling, Dropout y Batch Normalization. En el caso del maxpooling, al reducir la dimensionalidad= ayudó a generalizar el proceso de aprendizaje.En [4] se determino que las capas dropout eran más útiles después del max pooling y antes de la capa Batch Normalization. sin embargo en el presente trabajo se determino que eran mas eficientes despues de la capa de LSTM. Ademas la capa dropout en general produce un mejor resultado cuando esta establecido en 0.5 y la capa batch normalization mejoro mucho el accuracy y redujo el overfitting que se puede ver en el modelo base. Los parámetros de regularización, learning rate y decay jugaron un papel importante en la reducción del overfitting. El modelo que mejor esta posicionado es el modelo MM2 ya que es el que tiene mejores resultados

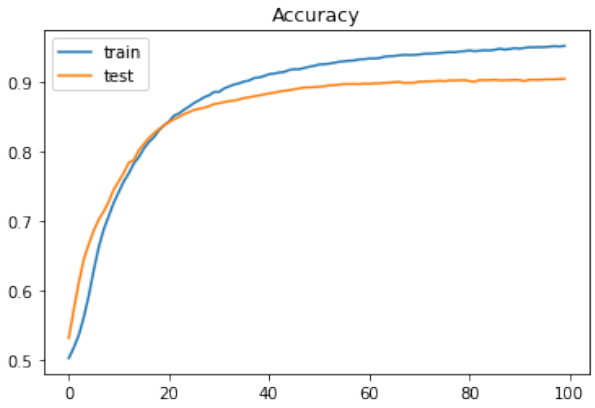

Fig. 17. Gráfica del accuracy del mejor modelo propuesto (MM2)

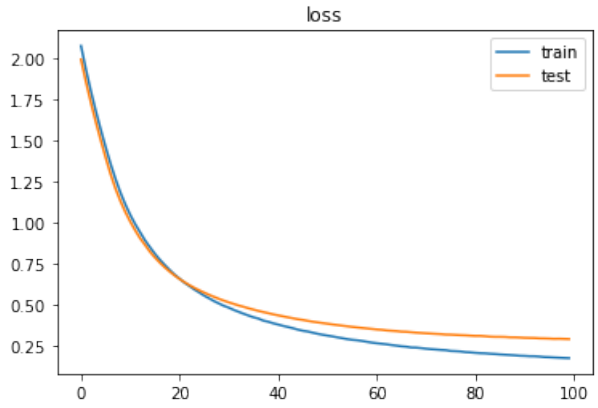

Fig. 18. Gráfica del loss del mejor modelo propuesto (MM2)

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 y su diccionario respectivo. Además, se creó una función que reemplaza el index del tópico del 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 19 , Figura 20 y Figura 21. El link para acceder al sistema es el siguiente: https://finalanalitica-g2.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 TF-IDF con una cantidad de tópicos igual a 24 y un Coherence Score de 0.4408.

Con respecto a la fase 2, los 3 mejores modelos de machine learning resultantes fueron RandomForest - TF-IDF, Regresion Logistica - Ngrams y Red Neuronal - Word2vect. El mejor modelo de machine learning en general fue el de Regresión logística - N-Grams. Com respecto a los modelos de deep learning, el mas estable es el modelo BiLSTM que cuenta con un embedding pre-entrenado (Glove) teniendo resultados comparables a los modelos de las arquitecturas multicanal en terminos de accuracy, recall, precision y F1-Score.Por el lado de los modelos multicanal se puede observar que el mejor optimizador es el RSMProp ya que evita que el modelo se overfitee durante las primeras epocas. Asimismo, el dropout, batch normalization y el kernel regulizer permiten que no se overfitee el modelo durante el entrenamiento. Por el lado de la union de los modelos de deep learning y machine learning bajo las tecnicas de voting y average demostraron que las combinaciones entre modelos de deep learning son mas robustas a comparacion de los de los modelos de machine learning. El modelo que es la union de todos los demas modelos tiende a ser robusto por los modelos de deep learning. En resumen, los modelos de deep learning mostraron mejor resultado al momento de clasificar los tweets en negativo o positivo y se debe a la capacidad de contextualizar el comentario.

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 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.Ademas, se puede usar los embeddings con la finalidad de poder usar modelos de deep learning pre entrenados y asi poder obtener un mejor performance. Asimismo se puede hacer mas combinaciones entre modelos de deep learning como CNNs y LSTMS inclyendo word embeddings pre entrenados.

References

[1] K. Bastani, H. Namavari and J. Shaffer, Latent Dirichlet allocation (LDA) for topic modeling of the CFPB consumer complaints, Expert Systems With Applications, 2019, vol. 127, 2019, pp. 256–271.

[2] M. Zeeshan Ansaria, M. B. Aziza, M. O. Siddiquib, H. Mehraa, K. P. Singha, Analysis of Political Sentiment Orientations on Twitter, Procedia Computer Science, 2020, vol. 167, 2020, pp. 1821-–1828.

[3] Lee, Ji Young and Franck Dernoncourt. “Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks.” ArXiv abs/1603.03827 (2016).

[4] A. Yenter and A. Verma, ”Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis,” 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, 2017, pp. 540-546, doi: 10.1109/UEMCON.2017.8249013.

Modelos propuestos	Convolution			Activation	Max pooling	Branch Dropout	Batch Normalization	LSTM
	Branches/Kernels Sizes	Filters	Kernel Regulizer	Type	Pool Size	Rate	Present	Units
model_08	3/4/5	32	None	ReLU	2	0	Yes	100
model_16	3/4/5	32	L2(0.1)	ReLU	2	0.5	Yes	100
model_43	2/3/4/5/6/7	128	L2(0.1)	ReLU	2	0	Yes	128
model_57	3/5/7/9	128	L2(0.1)	ReLU	2	0.5	Yes	128
model_63	3/5/7/9	128	L2(0.1)	ReLU	2	0.5	Yes	128
Base_model	5	84	None	ReLU	4	0.25	no	70
MM1	3/4/5	32	L2(0.1)	ReLU	2	0.5	Yes	100
MM2	3/4/5	32	L2(0.1)	ReLU	2	0.5	Yes	100
MM3	2/3/4/5/6/7	128	L2(0.1)	ReLU	2	0.5	Yes	128

TABLE VIII
ARQUITECTURA DE LOS MODELOS MULTICANAL

Modelos propuestos	Optimizer			Accuracy	Precision	Recall	F1-score	Tiempo (min)
	Type	Learning Rate	Learning Rate Decay					
model_08	Adam	0.001	0	91	92	91	91	650
model_16	Adam	0.001	0	91	92	88	90	650
model_43	Adam	0.001	0	81	73	96	84	800
model_57	RMSprop	0.001	0	90	95	81	88	500
model_63	RMSprop	0.01	0.1	90	91	90	90	600
Base_model	Adam	0.001	0	84	85	84	84	300
MM1	RMSprop	0.00001	0	89	89	89	89	400
MM2	RMSprop	0.00001	0	91	91	90	91	400
MM3	RMSprop	0.01	0	91	92	89	90	900

TABLE IX
RESULTADOS GENERALES DE LOS MODELOS DEEP LEARNING

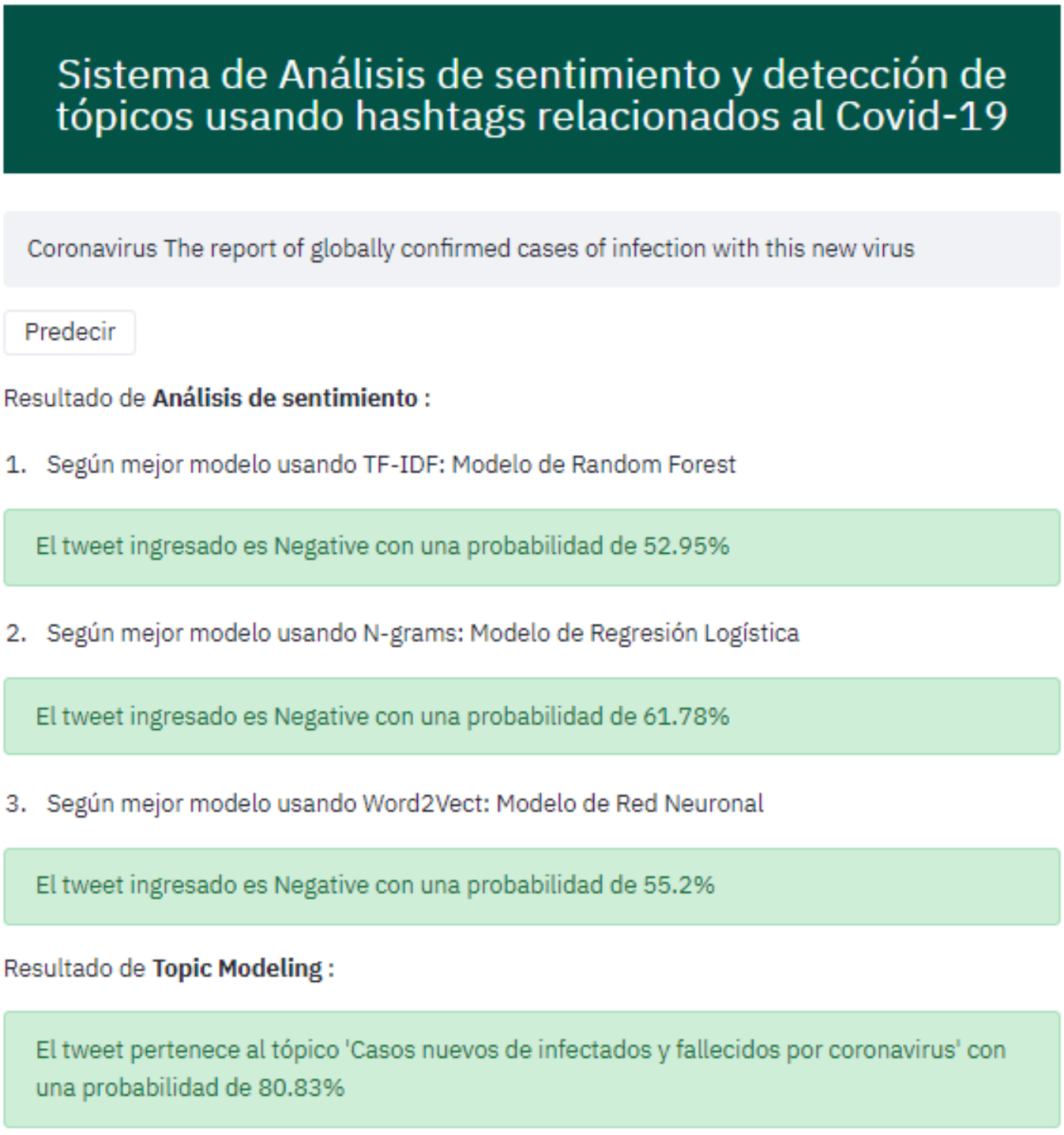

Fig. 19. Sistema de detección de tópicos y análisis de sentimiento - I

Mapa de casos confirmados de Covid-19 en el mundo

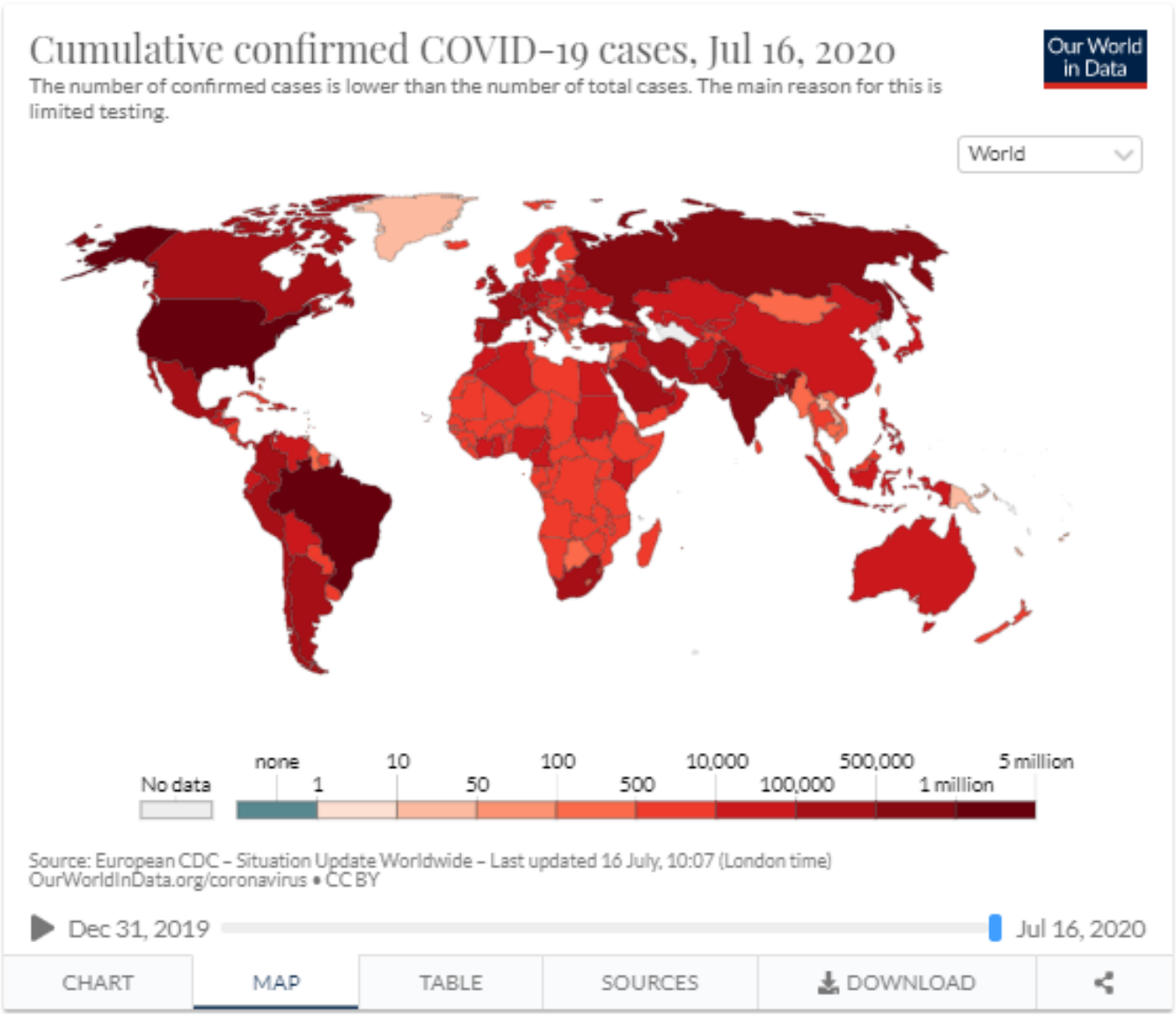

Fig. 20. Sistema de detección de tópicos y análisis de sentimiento - II

Crecimiento de casos confirmados de casos de Covid-19

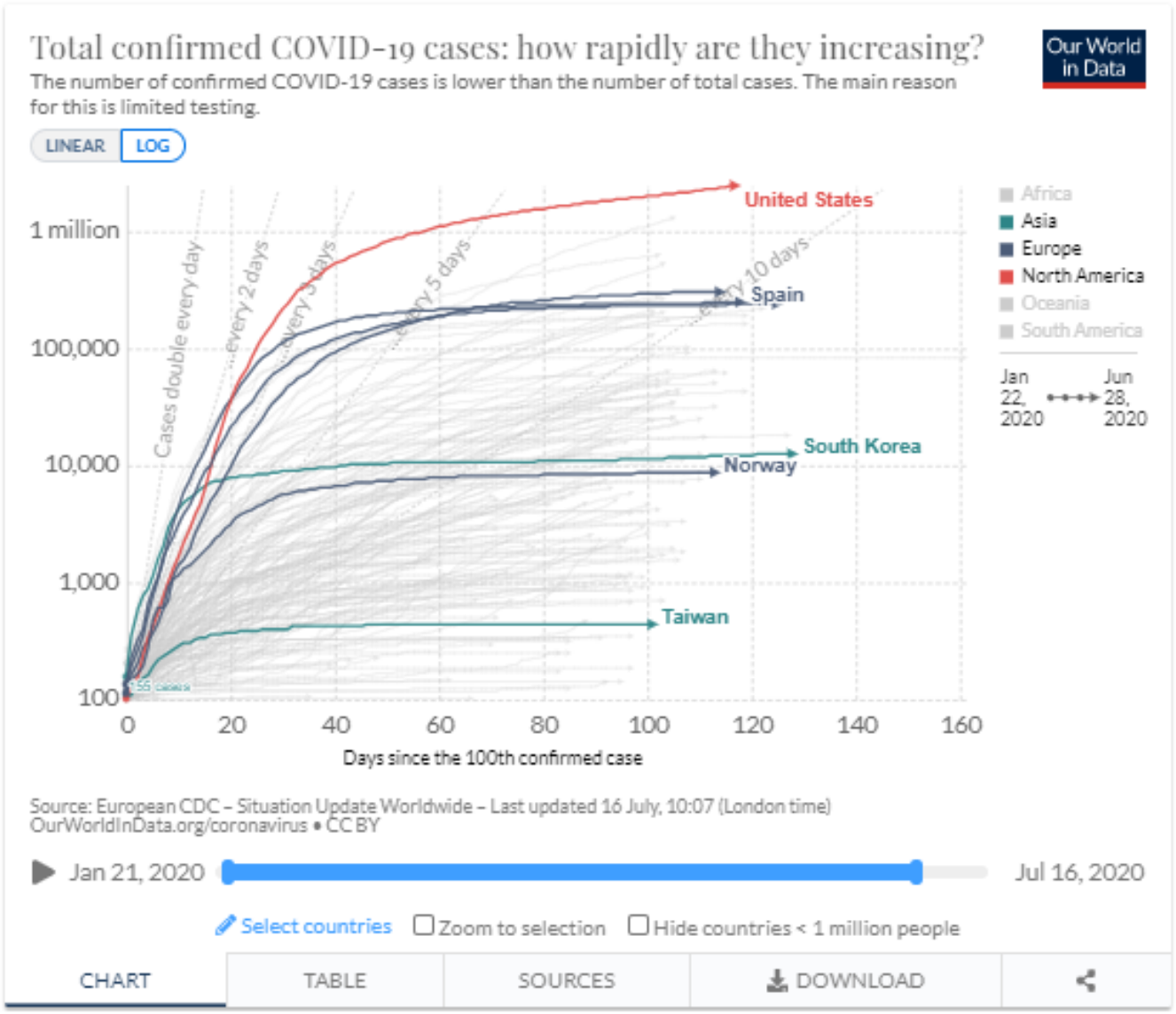

Fig. 21. Sistema de detección de tópicos y análisis de sentimiento - III