Análisis del Sentimiento del lenguaje en comentarios de películas de Rotten Tomatoes

Carolina García Patiño

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Tutor
Javier Fernando Botia Valderrama, Doctor (PhD)

Universidad de Antioquia
Facultad de Ingeniería

Especialización en Analítica y Ciencia de Datos

Medellín, Antioquia, Colombia

2021

RESUMEN

El presente trabajo aborda el desarrollo de un modelo de analítica para el procesamiento natural del lenguaje, más específicamente la clasificación multiclase para los comentarios de películas extraídas de la plataforma de reseñas *Rotten Tomatoes*. La importancia de este tipo de algoritmos, reside en el conocimiento de la expectativa y experiencia de los usuarios frente al consumo de un bien o servicio. Esto genera una migración de los modelos tradicionales de evaluación de satisfacción del cliente, donde se otorgan puntuaciones en escalas cualitativas, hacia una retroalimentación personal y detallada frente a su experiencia.

El dataset inicial consta de 156.060 comentarios en inglés con clases desbalanceadas, adicional, como se menciona en la descripción en la página de competición *Kaggle* (Kaggle, 2014) presenta particulares obstáculos frente al sarcasmo, ambigüedad en el lenguaje y la brevedad en las reseñas.

Se plantea abordar el problema con la metodología planteada para procesamiento del lenguaje en una revisión de la literatura por Jain et al. (2021), la cual abarca, en el marco del procesamiento de los datos, tokenización, remoción de stopwords y lematización sobre el remanente de palabras. Posteriormente durante la extracción de características, se usan dos tipos de metodologías, seleccionadas de acuerdo con el tipo de modelo aplicado, para los modelos denominados como *soft clasiffier* se aplica la vectorización del vocabulario a través de un *Term Frecuency Inverse Document Frecuency* (TF-IDF), mientras que para el modelo de *Deep Learning* se aplica una red tipo *Embedding*.

Como resultados generales, se obtiene un modelo con un accuracy del 73.02% y una tasa de F1-Score Micro y Macro del 73.01% y 72.11% respectivamente.

Palabras clave: Machine Learning, Sentiment Analysis, Natural Language Processing, RNN LSTM

TABLA DE CONTENIDO

1. INTRODUCCIÓN

De acuerdo con Jain et al., 2021, los volúmenes de datos de comentarios de retroalimentación de satisfacción del cliente han presentado un incremento, en gran medida a la incursión de diferentes plataformas de *e-commerce* como Amazon o del uso de las redes sociales como medio de divulgación de productos o procesos de soporte técnico. En este sentido, los datos describen opiniones, sentimientos y actitudes en torno al uso del producto y pueden ser claves en términos del negocio tanto en el soporte post venta como en la atracción de futuros compradores, en los escenarios en donde las valoraciones previas de clientes antiguos pueden influir en la decisión de compra de clientes futuros.

En el presente trabajo se abordará un modelo de clasificación multiclase para la predicción del sentimiento generado en los comentarios de películas de la página de *Rotten Tomatoes.* Los datos se obtienen de la competencia de kaggle denominada *Sentiment Analysis on Movie Reviews,* publicada hace 7 años en la plataforma. Cuenta con 876 implementaciones, con un valor máximo de ajuste del 0.76526.

El dataset se enmarca en un problema clásico de análisis de sentimiento del lenguaje, con una única variable de entrada: las reseñas. La complejidad de este, radica en la preparación previa de los datos debido a la cantidad de registros duplicados generados tras eliminación de *stopwords,* adicional las frases presentan ambigüedad y desbalanceo de clases, donde el 50% de los datos se concentran en la clase neutra. Se plantean cinco modelos entre ellos un Multi Naive Bayes, Random Forest Classifier, K Neighbors Classifier, Voting Classifier y una red neuronal de corta memoria LSTM.

Finalmente, la relevancia en este tipo de soluciones radica en el posicionamiento del producto o servicio basado en el incremento de opiniones positivas del consumidor. En la era digital actual se ha presentado una transformación de las variables influyentes en la decisión de compra de un consumidor. No sólo el precio, calidad o relevancia de marca hacen eco, si no, la experiencia de otros clientes frente al producto (Ali Acar & Puntoni, 2016). Claro ejemplo de esto se evidencia en el uso de aplicaciones hoteleras, donde las opiniones de los usuarios, juegan un factor determinante para la selección de hospedaje, posterior a la evaluación de precios ofrecidos. Las interacciones compañía-consumidor deberán ser rápidas, continuas y personales, lo que lleva a que este tipo de análisis sean el primer filtro para la mejora de procesos y la cercanía con el cliente.

2. MARCO TEÓRICO

El análisis de sentimiento, se basa en la extracción de características del texto que permitan la clasificación de la opinión del cliente en positivo, negativo o neutro, brindando de alguna manera una estructura de datos más descriptiva y generalizada frente a las percepciones del cliente.

Una de las desventajas que plantea Jain et al., 2021, en este tipo de aplicaciones se debe generar un entendimiento de contexto en el uso de las palabras, lo cual requiere una gran cantidad de datos de entrenamiento y un particular dominio de los datos de estudio.

Una metodología estándar aplicada en el desarrollo de modelos de procesamiento de lenguaje se describe en la Figura 1. Se compone de 3 macroprocesos, el preprocesamiento de los datos o *Data Preprocessing*, la extracción de características o *Feature Extraction* y finalmente la Implementación y evaluación del modelo o el *Decision makin*g. (Jain et al., 2021)

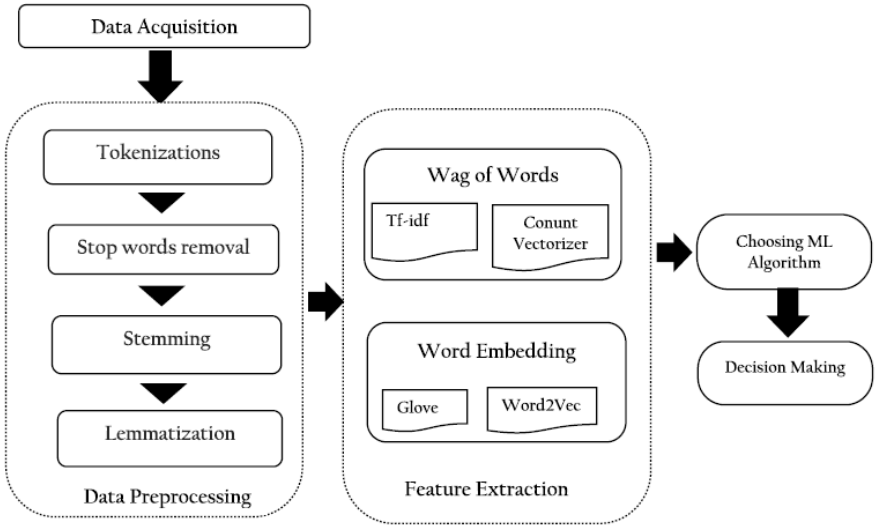

FIGURA 1. DIAGRAMA DE PROCESO ANÁLISIS DE SENTIMIENTO. TOMADO DE JAIN ET AL., 2021.

Dentro de los pasos del preprocesado se frecuentan los siguientes métodos:

- *Tokenizations*: División de las frases entre palabras denominadas tokens.

- *Stopwords*: Palabras que no tienen significancia semántica dentro del contexto de la oración, como artículos.

- *Stemming*: Reducir una palabra a su palabra raíz.

- *Lemmatization*: Reducir una palabra a su palabra raíz, garantizando que esta pertenezca al lenguaje.

Para los pasos de la extracción de características o *features* se dividen en dos tipos de técnicas, *Bag of words* y *Word embedding*. Para cada una de ellas, se tienen dos técnicas ampliamente aplicadas: TF-IDF y *CountVectorizer* para *bag of words* y *Word2Vec* y *Glove* para el Word embedding.

- *TF-IDF*: Por sus siglas en inglés Term Frequency-Inverse Document Frequency y su variación TF-IDF *weight*. Es una medida estadística que evalúa la importancia de una palabra dentro del documento en una colección de documentos o *corpus*. De acuerdo con TF-IDF determina que palabras en el corpus son relevantes a través de calcular una proporción inversa entre la frecuencia de la palabra en un documento especifico contra el porcentaje de documentos en el que la palabra aparece.

- *CountVectorizer*: Sparce matrix que representa en un vector la frecuencia de las palabras en una frase.

- *Word2Vec*: Se trata de una red neuronal de dos capas que procesa texto mediante "vectorización" de palabras, donde la entrada es un corpus de texto y la salida es un conjunto de vectores: vectores de características que representan palabras en ese corpus. Convierte texto en una forma numérica que las redes neuronales pueden procesar.

- *Glove*: representan sus siglas *Global Vectors for Word Representation*. Es un algoritmo de aprendizaje no supervisado, el cual genera una matriz de co-ocurrencia lineales de palabra a palabra en un *corpus*, como por ejemplo la co-ocurrencia entre Mujer-Hombre.

3. METODOLOGÍA

Como se expuso previamente, el problema se centra en una clasificación de sentimiento basado en comentarios. De manera general se abordan las etapas propuestas por Jain et al., 2021, ver Figura 1, resaltando que en esta metodología las relaciones entre etapas son secuenciales. Sin embargo, se adapta para este presente trabajo, una aplicación no secuencial en la etapa de eliminación de stopwords y lematización frente a la selección del diccionario final. Esto se debe a que los datos se ven influenciados por palabras propias de su contexto que no agregaran valor a la tarea propuesta. Por ejemplo, la palabra *movie*, no es una palabra clave para la tarea de clasificación, pero presenta una ocurrencia alta. En este caso se deberá eliminar, por la razón que no agregará valor en el diccionario.

En este apartado se analizará con detalle el step 1, ya que los demás serán abordados en los siguientes numerales.

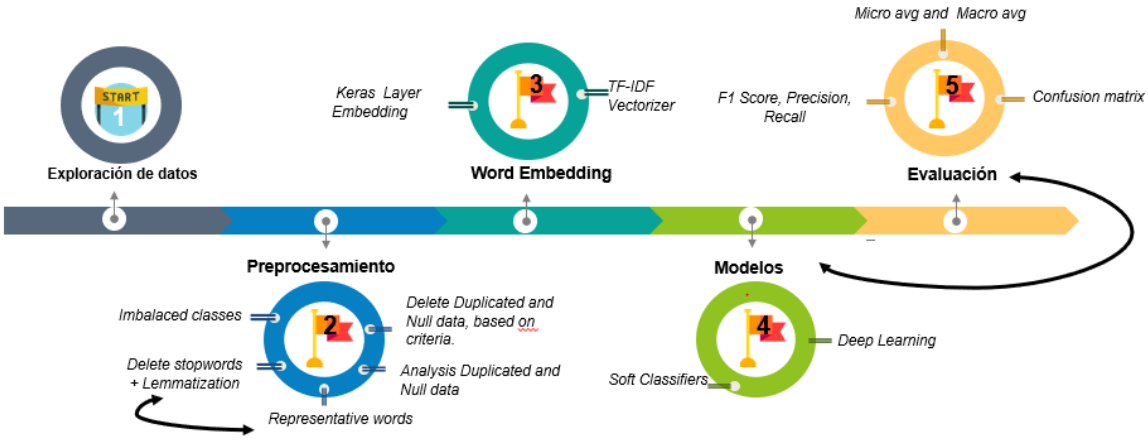

FIGURA 2. METODOLOGÍA SOLUCIÓN PROBLEMA. FUENTE: ELABORACIÓN PROPIA.

1) Exploración de datos: El primer paso es la exploración de los datos, de manera general, allí se identificaron el tamaño de las frases sin ningún procesamiento, la duplicidad de datos, valores nulos y la cantidad de datos por clase.

2) Preprocesamiento: En esta segunda etapa, se realizan ciertas estrategias para solventar algunas de las inconsistencias halladas en el paso 1, como el tratamiento para las clases desbalanceadas, la eliminación de valores nulos y la eliminación de valores duplicadas basado en un criterio decisión. Adicionalmente, se realizan procesos propios de esta etapa como la eliminación de stopwords y la lematización de las palabras.

3) Word Embedding: En este paso, se genera la creación del diccionario de términos o corpus con la metodología de TF-IDF (Por sus siglas en inglés *Term Frequency-Inverse Document Frequency*), la cual es una medida estadística que nos permite evaluar la frecuencia de una palabra dentro de una frase o documento y cuál es su frecuencia en los demás documentos.

4) Modelos: Se plantea un ajuste inicial de los denominados *Soft classifier*, trabajando con *Naive Bayes*, *Random Forest*, *KNN* y un Voting Classifier. Adicional, se plantea el desarrollo con el algoritmo RNN *LSTM*, Red neuronal recurrente de corta memoria. En el siguiente literal se ampliará los parámetros utilizados para cada uno de ellos.

5) Evaluación: Se utilizan métricas de evaluación como *el F1- Score, Recall, Precision* para el modelo completo como para cada una de sus clases, acompañado de la matriz de Confusión.

Adicional se analizan las métricas del *Micro Average precision scores y Macro Average precision scores* como evaluadores multiclase. La primera de ellas nos otorga una evaluación general del modelo en cuanto a la detección de los verdaderos positivos o *True positive* de todas las clases en conjunto, dividido por el total de predicciones positivas, es decir *False Positive* más *True Positive*, así mismo de cada clase.

La métrica macro, representa una media aritmética simple de la precisión obtenida en cada clase.

4. DESCRIPCIÓN DEL EXPERIMENTO

4.1 Exploración de datos

La cantidad de observaciones iniciales son 156.060, consta de 4 variables, dos de ellas representan identificadores, uno para la frase (*PhraseId*) y otro para la oración (*SentenceId*), con la particularidad que esa última presenta valores duplicados como se detalla en la Figura 3.Las dos variables restantes contienen el comentario y la etiqueta numérica asociada al comentario.

En la Figura 3, se observa una representación de las cinco primeras observaciones del *dataset*.

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

FIGURA 3. REPRESENTACIÓN DEL DATASET

4.1.1. Histograma de palabras por frase

Se realizó un primer acercamiento a los datos, analizando la cantidad promedio de palabras en las frases. De lo anterior, se obtuvo que el promedio de las palabras dentro de cada comentario tiene es de 7 palabras por frase, siguiendo una distribución con sesgo hacia la derecha, como indica la Figura 4, donde el tamaño máximo es cercano a 40.

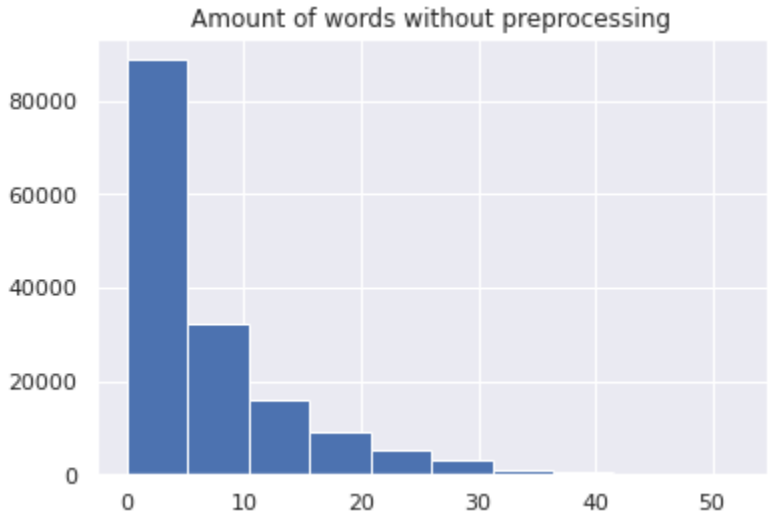

FIGURA 4. CANTIDAD DE PALABRAS POR FRASE ANTES DE PREPROCESAMIENTO.

4.1.2 Datos nulos y duplicados

No se detectan valores perdidos, pero sí 84 valores con duplicidad teniendo en cuenta sólo palabras o caracteres espaciales contenidos en las frases.

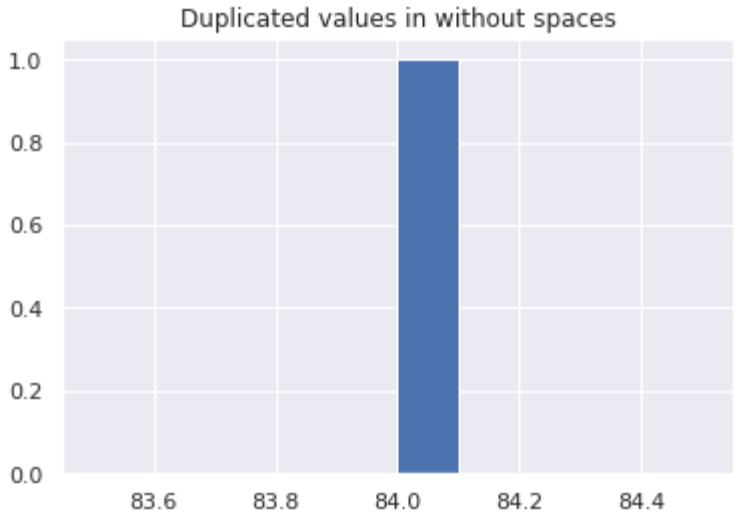

FIGURA 5. VALORES DUPLICADOS.

4.1.3. Descripción de clases

Por último, se tiene un se evidencia en la un desbalanceo de las 5 clases que presenta el problema. Aproximadamente el 51% de la información está clasificada como clase 2 o "*Neutral*", seguido de un 21% para la clase *"Some positive*", 17% para "*Some negative*", 6% para "*Positive*" y 5% para la clase "*Negative*".

FIGURA 6. CANTIDAD DATOS POR CLASE ANTES

4.2. Preprocesamiento

4.2.1. Clases desbalanceadas

En el literal anterior, se evidenció la existencia de clases desbalanceadas, se decidió una unificación de clases de muy positivo con positivo y muy negativo con negativo, ver Tabla 1.

TABLA 1. BALANCEO DE CLASES

Clase Inicial	*Nueva clase*	*Label*	*Porcentaje*
0 y 1	1	Negative	22.01
2	2	Neutral	50.99
3 y 4	3	Positive	27.00

Fuente: Elaboración propia.

El resultado de esto fue generar que el 49 % de las clases se concentra en las negativas y positivas y el otro porcentaje restante en la clase neutra, tal como describe la tabla anterior.

4.2.2. Limpieza de datos

El primer paso es la eliminación de las palabras stopwords en inglés a través de la librería NLTK, a esta lista se le adicionó eliminación de caracteres especiales que no estaban contemplados inicialmente en la librería. Seguido de ello, se aplicó lematización del remanente de palabras.

El conteo de palabras tras aplicar la limpieza se redujo a un promedio de 4 palabras por frase, lo que se concluye que en promedio 3 palabras estaban contenidas dentro de la lista de *stopwords*, la distribución conserva el sesgo, pero el máximo de palabras es de 20.

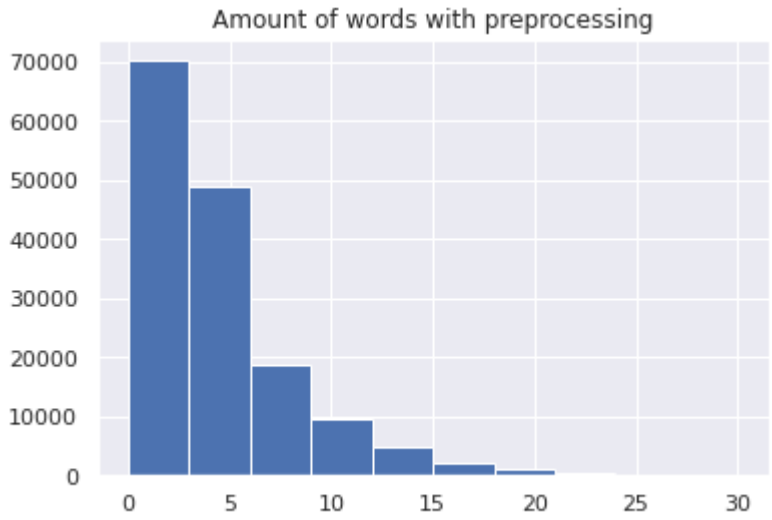

FIGURA 7. DISTRIBUCIÓN CANTIDAD DE PALABRAS DESPUÉS DE LA LIMPIEZA.

4.2.3. Datos nulos y duplicados

Previamente en la exploración de datos se obtuvo 84 muestras con duplicidad, tras aplicar la limpieza se genera un valor alto de valores duplicados equivalente a un 43% de las muestras iniciales, es decir 67.100 muestras o instancias.

Dentro de las más comunes, se identifican las monopalabras como *movie*, *film*, las cuales serían sinónimos y representan un 0.10% y 0.08% respectivamente del total de frases con duplicidad. En un 90% aproximadamente, estas palabras están asociadas a la clase 2.

Surgen otras palabras interesantes como *bad y better*, que, a pesar de estar en el top de palabras duplicadas más frecuentes, claramente podrían generar una distinción fuerte de las clases. Cabe resaltar que, se presentan 2 muestras donde *bad* recibe una etiqueta de positivo y 11 casos donde es un comentario neutro de 40 duplicados. Algo similar ocurre para la palabra *better* donde 25 ocurrencias de la palabra, 3 ocasiones es clasificada como negativa, y un poco menos de la mitad como neutra.

La estrategia implementada en el tratamiento de estos valores duplicados fue eliminar aquellas frases sólo con un conteo de duplicados igual a dos y los valores nulos. Tras esta limpieza el valor final de registros es 126.161 representando un 81% aproximadamente del valor inicial. Cabe resaltar, que, tras el proceso de eliminación, las clases ya redistribuidas, no sufren un impacto alto, ya que se conservan las proporciones del literal 5.2.1

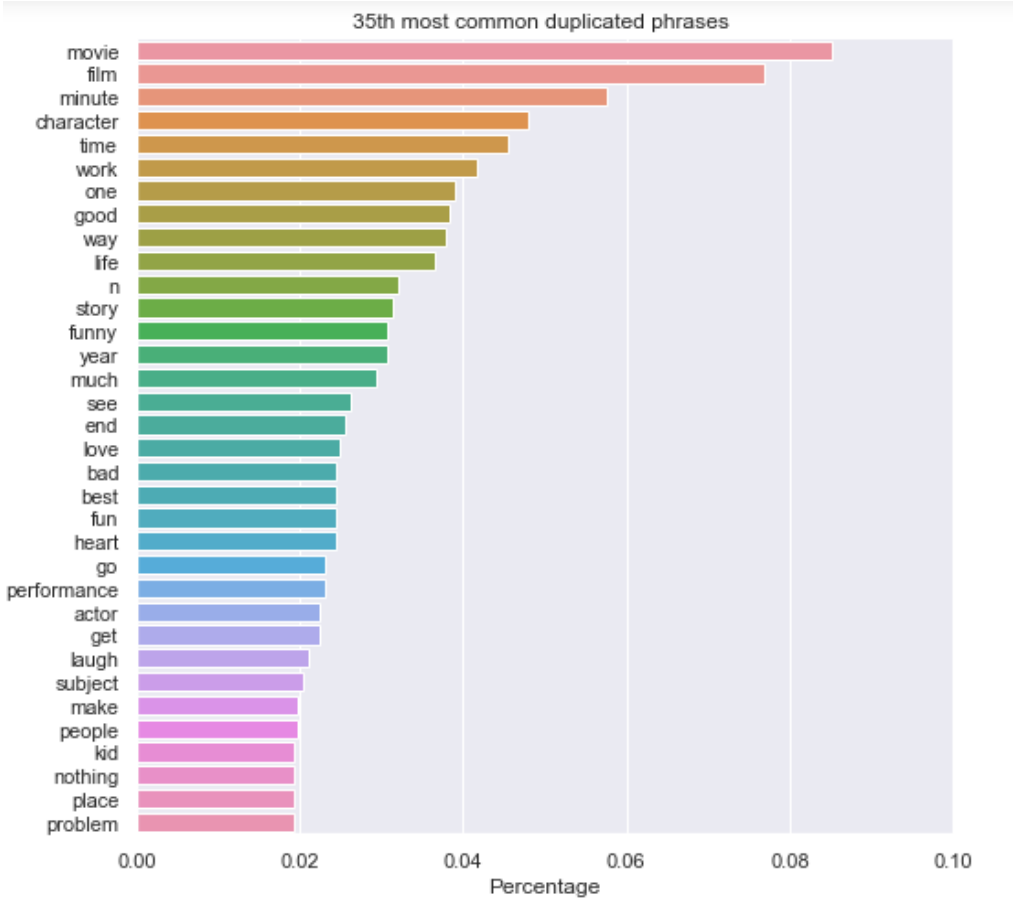

FIGURA 8. TOP 35 PALABRAS MÁS COMUNES.

En la posición 11 de la Figura 8, se identifica la letra *n* como una palabra recurrente, ahondando en las frases iniciales que contenían dicha palabra, se identificó que esta pertenece a una negación y está presente en frases como *was n' t, do n' t have to, is n´ t just*.
La cual tiene un contexto semántico de negación y siempre está dada en la misma estructura. Para dicha palabra se usó una estrategia de replace en la frase original, intercambiando *n´ t* por *not*. Una vez aplicada la estrategia esta frase salió del top de palabras frecuentes.

4.2.4 Word Embedding:

Como se describió previamente en la metodología, se dividieron los modelos en dos tipos: Machine learning de tipo supervisado y Deep Learning con una red neuronal de arquitectura Tensorflow.

Para los modelos tipo *Machine learning* se realizó la vectorización de las palabras a través de la librería de *Sklearn* con el módulo *feature*_extraction y el método *TfidfVectorizer*. Una

ventaja en el uso de este embedding, es la obtención de una métrica estadística entre la frecuencia de ocurrencia de la palabra en el texto en particular y la relación de la misma en otros textos. En este sentido, si la frecuencia de la palabra es repetitiva a lo largo de los documentos, se podrá relativizar su uso, ya que no permitirá agregar valor en la distinción entre clases. La configuración general, se presenta a continuación:

```
tfidf_vectorizer = TfidfVectorizer(max_features=df_word_count.shape[0]
,stop_words='english', use_idf=True, ngram_range=(1,3))
```

Para la red neuronal, se hizo uso de una capa *embedding* de la librería de Tensor Flow, como se presenta a continuación, con un diccionario de 13.725 palabras y un tamaño máximo de número de palabras de 30 por cada comentario.

```
model1.add(layers.Embedding(max_words ,max_length))
```

4.2.3 Modelos Machine Learning

El proceso de Split se asume de manera transversal con la función *train_test_split,de la* librería *Sklearn,* haciendo una partición de 80% para datos de entrenamiento y el porcentaje restante para el proceso de prueba.

Se realiza un *Grid search* para el *tunning* de los parámetros de cada modelo, seleccionando el f1_score y un *StratifiedShuffleSplit* como parámetros dentro de la *grid* como se describe a continuación:

```
st = StratifiedShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
nb_clf = GridSearchCV(estimator=nb_classifier, param_grid=parameters,
cv=st, scoring=f1)
```

En la Tabla 2, se describen cada uno de los modelos seleccionados con la respectiva búsqueda de hiperparámetros analizados en el proceso de tunning de cada uno de los modelos, así como las mejores hiperparámetros hallados y las principales métricas obtenidas. Cabe resaltar, que en el literal posterior se abordará con mayor detalle los resultados.

TABLA 2. MÉTRICAS GENERALES MODELOS IMPLEMENTADOS

Modelo ML	Parámetros GridSearch	Best Estimator	Métricas Globales Obtenidas
Multi Naive Bayes	{'alpha':[0.1, 0.5,1], 'class_prior':[[0.35,0.3,0.35] ,[0.4,0.3,0.3],[0.4,0.2,0.4]] }	MultinomialNB(alpha=1, class_prior=[0.35, 0.3, 0.35], fit_prior=True)	Accuracy = 0.55 BalancedAccuracy= 0.61 F1 Score micro=0.55 F1 Score macro=0.55
RandomForest Classifier	{'n_estimators': [60,80,100,200,300,400], 'max_depth':[2,4,6,8,10,20]}	class_weight='balanced_subsample', criterion='gini', max_depth=20, n_estimators=300	Accuracy = 0.62 BalancedAccuracy= 0.55 F1 Score micro=0.62 F1 Score macro=0.57
KNN	{'n_neighbors':[5, 7, 9, 11,13,15], 'metric':['minkowski'], 'p':[1,2]}	KNN_clf.best_estimator_ KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform')	Accuracy = 0.68 BalancedAccuracy= 0.64 F1 Score micro=0.68 F1 Score macro=0.65
Voting Classifier	clf1 = RandomForestClassifier(n_estimators=300, random_state=1,max_depth=20) clf2 = AdaBoostClassifier(n_estimators=400) clf3 = KNeighborsClassifier(metric='minkowski',n_neighbors=5, p=2)	No Aplica	Accuracy = 0.64 BalancedAccuracy= 0.53 F1 Score micro=0.64 F1 Score macro=0.55
RNN		No aplica	Accuracy = 0.73 BalancedAccuracy= 0.72 F1 Score micro=0.73 F1 Score macro=0.72

Fuente: Elaboración propia**.**

5. RESULTADOS Y ANÁLISIS

Para los datos de prueba, se cuenta con una muestra de 25.233, cuya clase más prominente es la 2, correspondiente a la etiqueta de neutro. Esta representa 12.778 muestras, es decir, un 50% del total de los datos, seguido de la clase 3 o positivo con 6.852 registros, equivalente al 27% y el 23% restante es clase 1 o negativo. Intuitivamente, la evaluación podría estar

generando una alta tasa de falsos positivos en la clase 2, debido a ser la clase con el mayor número de registros asociados. A continuación se detallarán los resultados de la evaluación de cada uno de los modelos, basados en una matriz de confusión multiclase y la métrica F1 Score.

En el modelo de Multi Naive Bayes las tasas de falsos positivos son mayores para las clases 1 y 3, con valores respectivos de 21,46% y 15.41%, mientras que esta métrica tiene un valor de 8.15% en la clase 2, lo cual lo hace positivo dado que, siendo la clase más representativa, no está asignando un gran número de datos, pertenecientes a las clases 1 y 3 como clase 2.

Sin embargo, el caso contrario ocurre para los falsos negativos, donde el valor más alto está dado para la clase 2, con un 31.33% y de alrededor de 6.5% para las demás clases. Ver Figura 9. Se obtiene un F1-Micro Score de 55% y un F1-Macro Score de 55%, ver Tabla 2.

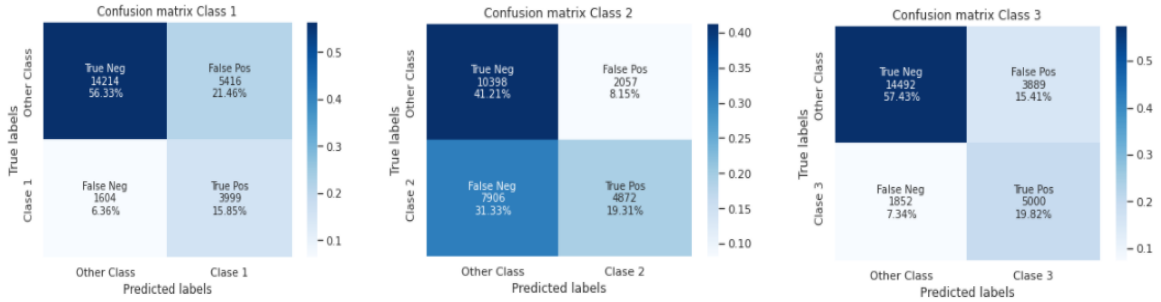

FIGURA 9. MATRIZ DE CONFUSIÓN POR CLASE MULTI NAIVE BAYES

En la Figura 10, las tasas de falsos positivos en el modelo *de Random Forest Classifier* son menores para las clases 1 y 3, con valores respectivos de 7,02% y 6.89%, mientras que esta tiene un valor de 24.17% en la clase 2. Lo cual indica que está intentando clasificar las demás clases como 2, para aumentar las clasificaciones verdaderas.

Sin embargo, para los falsos negativos, los valores están similares para las 3 clases, entre 13.56%, 10.05% y 14.47% respectivamente en clase 1,2,3. Un punto a resaltar, son los verdaderos negativos, donde en las clases “positivas” y “negativas” es decir 1 y 3, se nota un alto porcentaje de registros clasificados como tal, lo cual a pesar de que las métricas son mejor que el primer modelo, no parece estar creando una separación generalizada de las características de los datos. Se obtiene un F1-Micro Score de 62% y un F1-Macro Score de 57%, ver Tabla 2.

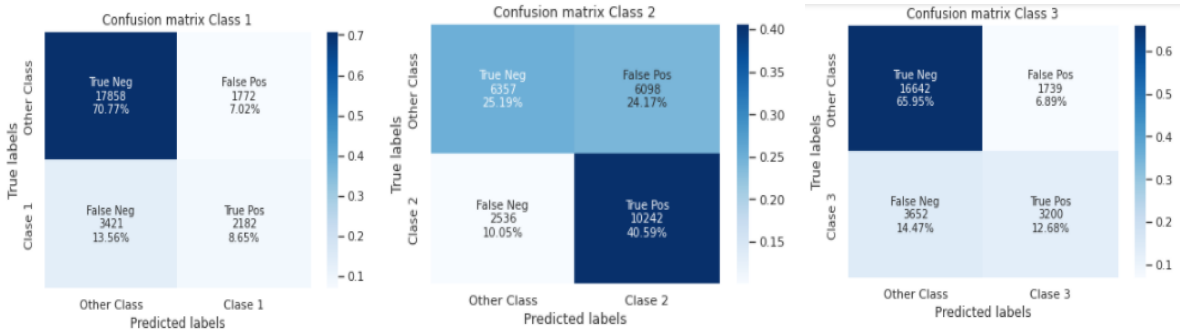

FIGURA 10. MATRIZ DE CONFUSIÓN RANDOM FOREST CLASSIFIER

En la Figura 11, las tasas de falsos positivos en el modelo de KNN son menores para las clases 1, seguido de la clase 3 y 2, con valores respectivos de 5,62%, 7.33% y 19.06%. A diferencias de los falsos negativos, donde los valores son similares para las 3 clases, entre 7.33%, 19.06% y 5.62% respectivamente en clase 1,2,3. Estos valores bajos de clasificaciones erróneas, lleva a un valor en la diagonal principal de 83% aproximadamente para las clases 1,3 y del 70% en la clase 2. De acuerdo con la Tabla 2, se obtiene un F1-Micro Score de 68% y un F1-Macro Score de 65%.

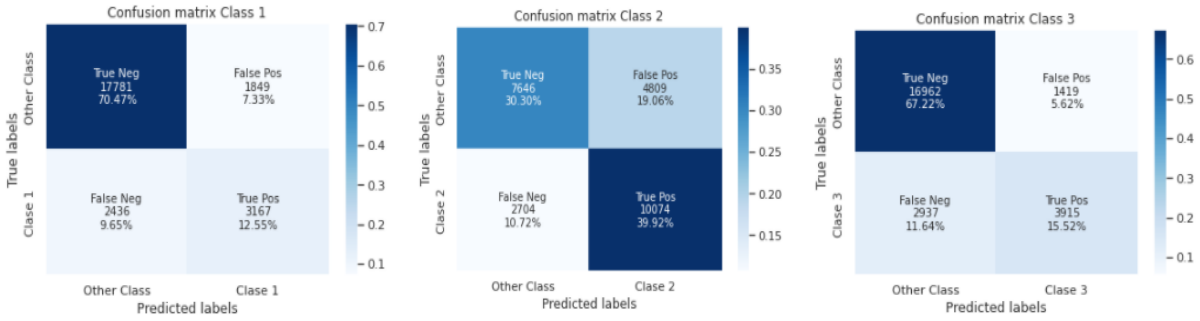

FIGURA 11. MATRIZ DE CONFUSIÓN KNN

En las matrices de confusión obtenidas para la implementación de *Voting Classifier* ver Figura 12, se evidencia que las tasas de falsos positivos para la clase 2 es cercana al 19%, por lo que el modelo intenta asignar la mayoría de las instancias a esta clase más prominente. En general, las diagonales referentes a los valores verdaderos, ver Tabla 3, son cercanos al 80% para las clases de positivo y negativo, pero este valor disminuye para la clase con la mayor cantidad de observaciones, con un valor del 70% aproximadamente. Como se indicó, la razón que más influye en este valor es la tasa de falsos positivos. De acuerdo con la Tabla 2, se obtiene un F1-Micro Score es de 64% y un F1-Macro Score de 55% para el modelo de votación.

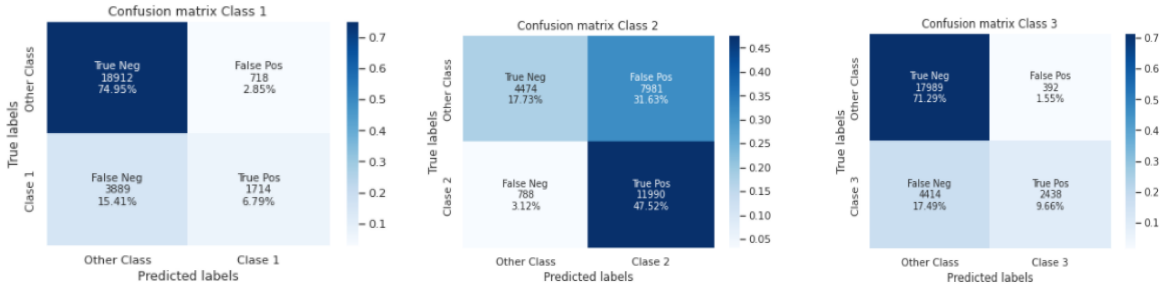

FIGURA 12. MATRIZ DE CONFUSIÓN VOTING CLASSIFIER

Finalmente se tiene la evaluación del modelo de LSTM ver Figura 13, en este se presentan, tasas de falsos negativos y falsos positivos en promedio por debajo del 10% para todas las clases, lo cual es altamente satisfactorio en la clase 2 ya que representa el menor valor de todos los modelos ejecutados. El valor de la diagonal para los valores verdaderos, son del 85% para la clase positiva y negativa. Para las observaciones neutras, el valor de la diagonal disminuye a un 75%, ver Tabla 3. Los valores F1-Micro Score y F1-Macro Score obtenidos son de 73% y un de 72% respectivamente.

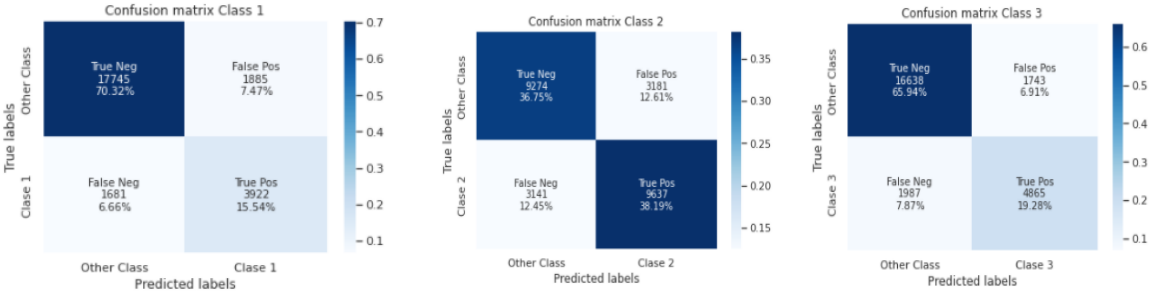

FIGURA 13. MATRIZ DE CONFUSIÓN RNN

TABLA 3. RESUMEN TASAS CLASIFICACIONES VERDADERAS

Modelo	***Clase1 Negativo***	***Clase 2 Neutro***	***Clase 3 Positivo***
Multi Naive Bayes	72.18%	60.52%	77.25%
RandomForest Classifier	79.42%	65.78%	78.68%
KNN	83.02%	70.22%	82.74%
Voting Classifier	81.74%	65.25%	80.85%
RNN	85.86%	74.94%	85.22%

6. DISCUSIÓN DE LOS RESULTADOS

A la vista de los resultados generados, a través de las métricas descritas en la Tabla 2. *Métricas Generales modelos implementados* y la matriz de confusión multiclase de la Figura 13, con la red neuronal tipo *Long Short–Term memory (LSTM*) implementada, se obtienen las métricas con mejor balance entre el objetivo individualizado de clasificación por cada clase y la métrica generalizada de clasificación, cuyos valores de precisión se obtienen por encima de 70% y las diagonales principales de las clasificaciones verdaderas, se ajustan a valores del 85% para las clases 1 y 3 y de un 75% para la clase 2, ver Tabla 3. Otro punto a resaltar, es que la precisión del modelo en el *dashboard* de liderazgo de la competencia en *Kaggle* (Kaggle, 2014), está en un aproximado del 76%, 6 puntos porcentuales por encima del obtenido con la implementación de la red neuronal LSTM.

Cabe resaltar que esta arquitectura implementada consta de una red básica de *Keras* con una única capa LSTM y una salida densa multiclase con 3 neuronas, la sencillez de la red se debe a la limitante computacional en tiempo por sesión ofrecida por el recurso *opensource* de *Google Colab*. El tiempo de ejecución en la tarea de entrenamiento fue de 6 horas aproximadamente con 60 épocas, versus un tiempo total de 7 horas en promedio disponible en *Colab*.

Por otro lado, el desempeño del modelo KNN, presenta un resultado similar para sus matrices de confusión, con valores en la matriz principal del 83% y 83% para clases 1,3 y del 70% aproximadamente en la clase 2. El coste computacional es similar a la red neuronal y obtiene un F1-Micro score del 68%.

Para finalizar, la red neuronal implementada tiene oportunidad de robustecerse, aumentando el número de capas intermedias LSTM y las épocas en el entrenamiento.

7. CONCLUSIONES Y TRABAJOS A FUTURO

- En una etapa inicial de exploración de datos, no se contaban con registros duplicados y tan sólo 84 registros sin valores. Posterior a aplicar la limpieza de datos con stopwords y lematización, se detecta un equivalente a 43% de los registros como valores duplicados, correspondientes a 67.100 instancias.

- De lo anterior, cabe resaltar la importancia de la aplicación de la limpieza de datos en un modelo de predicción de sentimiento del lenguaje, ya que la identificación de estos valores duplicados puede evitar un sobreajuste del modelo en el entrenamiento.

- La duplicidad de una frase, en este caso de estudio particular, podía presentar más de 2 ocurrencias, por lo cual, y dado que los valores repetidos se ubican en un orden cercano al

50%, se decide no trabajar con aquellos registros que tuviesen menos de dos ocurrencias, puesto que representaban el 18% del total de los registros iniciales. Solventando con ello, un posible sobreajuste y el hecho de no tener la suficiente cantidad de muestras para el desarrollo del modelo.

- Tras aplicar la eliminación de registros nulos, correspondientes a un 1% sobre el total inicial de muestras y la eliminación del 18% de valores con dos duplicados, se obtiene un remanente de 126.161 registros, lo que es equivalente a un 81% del total de datos netos.

- Como ya se hizo mención, el paso de la limpieza de datos en este caso permitió analizar duplicidad y crecimiento en los registros iniciales nulos, sin embargo, es recomendable no sólo implementar las palabras del diccionario de "*stopwords*" que la librería trae implementado si no ahondar en palabras claves que se adecuen al contexto específico del problema. En este caso en particular, se detecta que es recurrente el uso de las negaciones en inglés como contracciones con espacios entre sí, es decir, el uso de la palabra not como n 't. Por lo tanto, la estrategia utilizada fue identificar este tipo de palabras y reemplazarlas previamente por su respectiva negación.

- El número de clases iniciales que presentan el problema son 5, divididas como: Negativo, Algo Negativo, Neutro, Algo positivo y Positivo, con porcentajes de representación respectivamente del 4%, 17%, 51%, 21% y 6%. Originado un evidente desbalanceo de las clases con predominancia de la clase neutra. La estrategia implementada para solventar el desbalance es la unión de clases dado que, si algo es algo positivo o negativo, pues ya de entrada cabe en una clase general ya sea positiva o negativa. El resultado final, de estas agrupaciones generan un 22% para clase Negativa y un 27% para clase Positiva. Este comportamiento de la distribución de clases se conserva tras aplicar la limpieza de los datos.

- Otra estrategia para el desbalanceo de datos, posterior al agrupamiento preveniente mencionado, y para este tipo de problemática en particular, es el *resample* o remuestreo. En esta técnica se controla la cantidad de clases que se utilizan para el desarrollo del modelo, sin embargo, la utilización de esta no se aplica ya que reduce la cantidad de instancias en un 34% sobre el dataset con el preprocesamiento, pasando de tener 126.161 registros a 83.266 aproximadamente. Esto asumiendo muestras de clases similares a la clase menos prominente, lo cual equivale a valores de 27.000 registros por clase.

- El coste computacional en el entrenamiento para este tipo de modelos de texto es alto, asumiendo la premisa de todo modelo de Machine Learning de que, al conservar la mayor cantidad de muestras posibles para entrenar, nos arroja modelos más robustos y con mayor capacidad de generalización. En promedio el entrenamiento de los modelos tipo *soft classifier* y la red neuronal LSTM básica tardaban entre 6 y 8 horas aproximadamente. Esto sin ahondar en búsquedas de hiper parámetros más amplias o técnicas más robustas como Procesos Gaussianos.

- A la vista de los resultados generados, a través de las métricas descritas en la matriz de confusión multiclase, con la red neuronal tipo *Long Short–Term memory* (LSTM) implementada, se obtienen las métricas con mejor balance entre el objetivo individualizado

de clasificación por cada clase y la métrica generalizada de clasificación, cuyos valores de precisión se obtiene por encima de 70%. Estos resultados son alentadores a la luz del uso de una red neuronal simple, con una sólo capa tipo LSTM, además de contar con unos datos de prueba donde se tiene predominancia de la clase neutra.

- Se plantea, para trabajos futuros sobre este problema en particular, ahondar en el modelo de red neuronal LSTM, aumentando su complejidad en las capas intermedias de este tipo de memoria corta con capas densas.

- Otro modelo que ha comenzado a entrar en auge en el contexto de Procesamiento Natural del Lenguaje es el modelo BERT, por sus siglas en inglés de *Bidirectional Encoder Representations from Transformers*. La idea tras este algoritmo pre-entrenado es la aplicación de un entrenamiento bidireccional permitiendo un profundo entendimiento del contexto y el flujo del contexto del lenguaje, opuestos a los algoritmos secuenciales unidireccionales como la LSTM (Colón-Ruiz Cristóbal & Segura-Bedmar Isabel, 2020).

8. REFERENCIAS BIBLIOGRÁFICAS

Kaggle. (2014). Obtenido de Sentiment Analysis on Movie Reviews: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews

Ali Acar, O., & Puntoni, S. (2016). CUSTOMER EMPOWERMENT IN THE DIGITAL AGE. In *Forthcoming in Journal of Advertising Research* (Vol. 1).

Colón-Ruiz Cristóbal, & Segura-Bedmar Isabel. (2020). Comparing deep learning architectures for sentiment analysis on drug. *Journal of Biomedical Informatic*, 110.

Jain, P. K., Pamula, R., & Srivastava, G. (2021). A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews. In *Computer Science Review* (Vol. 41). Elsevier Ireland Ltd. https://doi.org/10.1016/j.cosrev.2021.100413

9. ANEXOS

En el siguiente link de GitHub se almacenan los datos utilizados para en entrenamiento y prueba, así como los notebooks implementados en el desarrollo de cada algoritmo aplicado: https://github.com/carolina-garcia5/Monograf-a_UdeA.git