Clasificacion de texto - Comentarios virales de YouTube

Mario García García

11 de mayo de 2023

Resumen

En este proyecto se presenta un modelo de clasificación en version beta de comentarios de YouTube que permite identificar aquellos que tienen mayor probabilidad de volverse virales. Para lograr esto, se utilizó una combinación de análisis de sentimientos y técnicas de aprendizaje automático, que permiten identificar patrones y características relevantes en el texto. Se recopilaron y etiquetaron datos de comentarios de la velada 2 del famoso Youtuber Ibai Llanos previamente viralizados y se utilizaron para entrenar y validar el modelo propuesto. Este enfoque puede ser útil para empresas de publicidad y marketing que buscan identificar y aprovechar comentarios virales para promocionar productos o servicios en línea antes de que los humanos digan a cierto comentario si les ha gustado o no.

1. Introducción

Este proyecto tiene como objetivo aprovechar el gran volumen de comentarios disponibles en YouTube para desarrollar un modelo de clasificación que permita a los creadores de contenido y las empresas identificar los comentarios con mayor potencial para volverse virales. Esto puede ser de gran ayuda para la promoción de productos y servicios, ya que los comentarios virales tienen el potencial de influir en la percepción de los usuarios y aumentar la visibilidad de una marca. En este caso, se ha tenido en cuenta todos los comentarios del famoso evento de la Velada del año 2 del famoso YouTuber y streamer Ibai Llanos, con el objetivo de que cuando se celebre la Velada del Año 3, seamos capaces de conseguir comentarios virales.

El proceso de desarrollo del modelo implica varias etapas, desde la recolección y limpieza de datos, analísis de datos hasta la implementación y evaluación del modelo final. El análisis de texto y la limpieza de datos son importantes para garantizar que el modelo esté trabajando con datos de alta calidad y reducir el ruido en los datos que podrían afectar la precisión del modelo.

El análisis de sentimientos utilizando transformers tipo BERT [6] y nltk [3] es una técnica poderosa que permite comprender el tono y el sentimiento de los comentarios. La combinación de esta técnica con el aprendizaje automático y las redes neuronales recurrentes permite identificar patrones en los datos y realizar clasificación de texto, lo que permite identificar los comentarios que tienen el mayor potencial para volverse virales.

En resumen, el desarrollo de un modelo de clasificación de comentarios de YouTube que permita identificar aquellos que tienen mayor probabilidad de volverse virales es un proyecto emocionante y desafiante que combina técnicas de análisis de texto, aprendizaje automático y redes neuronales. Si se logra, este modelo podría tener un gran impacto en el marketing y la publicidad en línea, así como en la percepción de los usuarios sobre productos y servicios, no solo para este caso en concreto.

2. Explicación del proyecto

2.1. Recolección de datos

Para desarrollar un modelo de clasificación de comentarios de YouTube que permita identificar aquellos con mayor probabilidad de volverse virales, se necesitan datos relevantes y representativos. En este proyecto, se ha creado un script en JavaScript utilizando Google Sheets para la recolección de datos de comentarios de YouTube. Los datos se han obtenido de todos los comentarios de la Velada del Año 2.

De los datos recopilados, se han seleccionado únicamente dos columnas: el contenido del comentario y el número de likes que ha recibido ese comentario. La selección de estas columnas se debe a que son los datos más relevantes y útiles para identificar comentarios con mayor probabilidad de volverse virales. Además, no se ha utilizado la columna que representa el nombre del usuario que ha dejado el comentario para mantener una perspectiva general y no hacer distinciones entre los usuarios de YouTube.

También se han eliminado los comentarios que responden al primer comentario, ya que se considera que pueden estar influenciados por el comentario inicial y podrían no tener la misma probabilidad de volverse virales por sí solos. Además, se ha eliminado el hilo de comentarios en su totalidad, ya que muchos usuarios de YouTube prefieren leer solo el comentario original en lugar de pasar por todo un hilo de comentarios y respuestas.

El enfoque de seleccionar solo los datos más relevantes y eliminar los comentarios que pueden estar influenciados o sesgados, permite obtener una muestra más representativa de comentarios de YouTube para el entrenamiento del modelo de clasificación de comentarios. De esta manera, se espera que el modelo pueda identificar patrones y características relevantes para la predicción de comentarios virales con mayor precisión y generalidad.

El código de este script se puede obtener directamente desde mi GitHub.

2.2. Análisis de los datos

La definición de un comentario viral puede variar de persona a persona y no hay un límite claro para determinar hasta qué punto un comentario ha sido viral o no. Sin embargo, en este proyecto, se ha establecido que cualquier comentario que reciba más de 20 likes se considera un comentario viral. Este criterio se ha adoptado para fines prácticos de predicción y permitirá clasificar los comentarios en dos categorías: viral y no viral. De esta manera, el modelo de clasificación de comentarios de YouTube podrá identificar patrones y características relevantes que puedan contribuir a la predicción de comentarios virales con mayor precisión.

A la luz de este criterio, conviene mostrar como la influencia latinoamericana supera a la española:

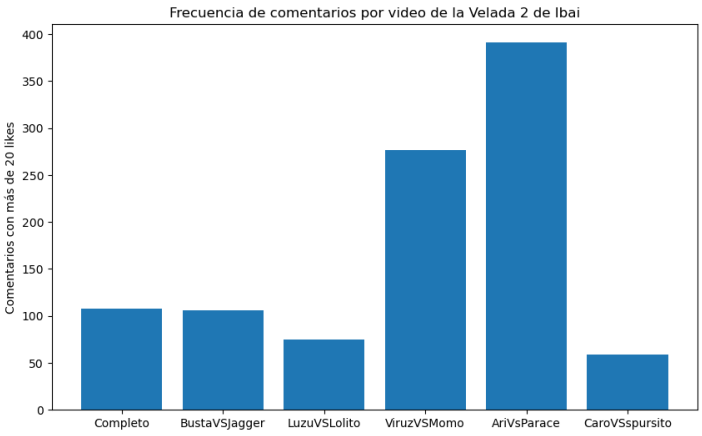


Figura 1: *En el eje x representamos en cada video la frecuencia de comentarios con más de 20 likes junto con la pelea que se desarrolló. El nombre "Completo.es el video completo sin procesar*

esto se puede explicar a que en latinoamerica hay de manera estimada 237 millones de personas en la franja de 10-24 años, en comparación con España que el número total de habitantes en cualquier franja de edad es de 47 millones, casi 6 veces más.

Tras juntar todos los comentarios en una misma base de datos, se ha hecho dos tipos de recuentos; el más significativo que es ver que palabras son las más repetidas en todos los videos y el número de preposiciones, artículos, conjunciones...etc

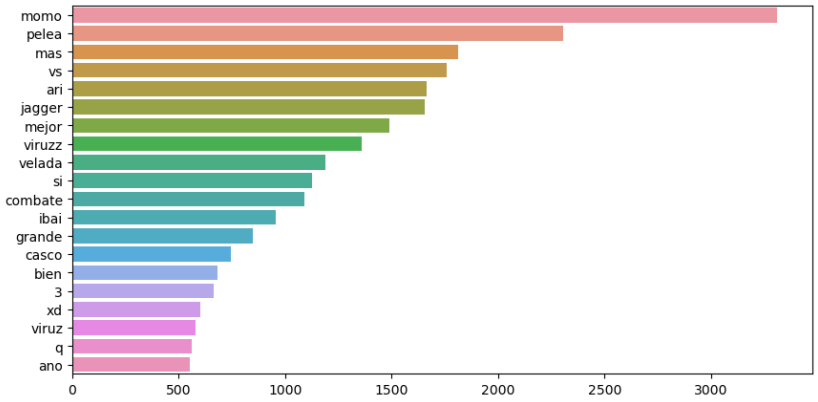


Figura 2: *Frecuencia de palabras más usadas en la Velada del Año 2*

Se puede observar aqui también el apoyo y la viralidad que ha implicado Momo y Arigameplays en esta velada. Más datos interesantes es la impresionante repercusión que ha tenido byViruZz en segundo lugar con momo, puesto que se escribe viruz y viruzz de manera similar. Finalmente destacar como palabra clave "casco", quizás por la preocupación que hubo al disponer cada boxeador de uno diferente.

Aqui se muestra todo el número de preposiciones, artículos, conjunciones...etc más comunes en estos comentarios y que se han eliminado.

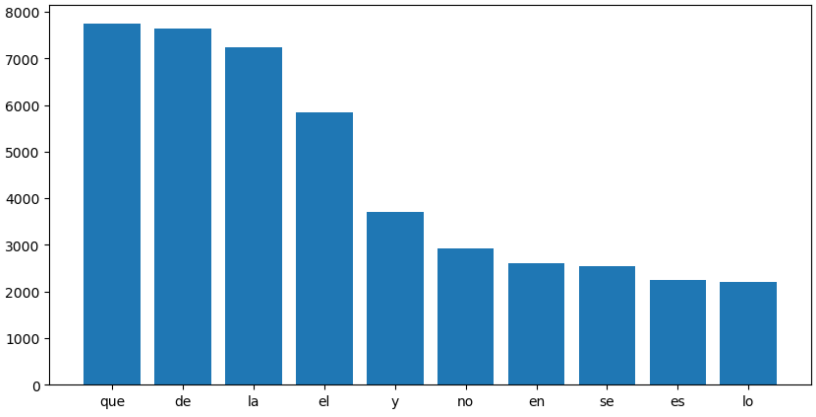


Figura 3: *Frecuencia de stopwords más usados en la Velada del Año 2*

2.3. Preprocesamiento de datos

También se ha explorado la posibilidad de que un comentario original con respuestas sea más viral que aquellos sin ellas, ya que podría proporcionar información valiosa para nuestro modelo. Sin embargo, se ha determinado que la presencia de respuestas en un comentario no es necesariamente un indicador de su viralidad. Por lo tanto, esta idea no se ha incluido en el modelo de predicción.

En el procesamiento del lenguaje natural, es común encontrarnos con textos que contienen elementos que no son relevantes para el análisis que queremos realizar. Por ejemplo, los caracteres especiales y la puntuación, como signos de exclamación o comas, pueden ser útiles para el significado de un texto, pero no para nuestro objetivo de análisis. Por esta razón, una de las técnicas de limpieza más populares consiste en eliminar estos elementos.

Asimismo, los números tampoco suelen ser relevantes en el análisis del lenguaje natural, a menos que se esté haciendo un análisis específico sobre ellos. Por lo tanto, eliminar los números del texto puede facilitar el procesamiento de los datos.

Otra técnica de limpieza muy útil es la eliminación de palabras vacías o stopwords, que son palabras muy comunes en un idioma y que no aportan significado relevante al análisis. Estas palabras suelen ser artículos, preposiciones, conjunciones, entre otras. Al eliminarlas del texto, podemos reducir el ruido y obtener una versión más clara y concisa del mismo.

En la actualidad, los emojis se han vuelto muy populares en la comunicación escrita, especialmente en las redes sociales. Sin embargo, en muchos casos no son relevantes para el análisis que queremos realizar. Por esta razón, una técnica de limpieza consiste en eliminar los emojis del texto.

Es común convertir todo el texto a minúsculas para evitar que el procesamiento del lenguaje natural considere dos palabras iguales como diferentes solo por estar escritas en mayúsculas o minúsculas diferentes. De esta manera, podemos garantizar que nuestro análisis sea más preciso y consistente.

También se han eliminado algunos caracteres específicos que no aportan información relevante para el análisis. En particular, se han eliminado caracteres como "", que son etiquetas HTML que se utilizan para cerrar negritas en los comentarios de YouTube o bien referenciar ciertos momentos del video de YouTube. También se han eliminado comentarios que contienen referencias a páginas web mediante el uso de "httpz etiquetas"</br>", que se utilizan para hacer saltos de línea.

La eliminación de estos caracteres específicos se ha realizado porque no aportan información relevante para la identificación de patrones o características en los comentarios virales. Además, la presencia de estos caracteres puede dificultar el análisis de los datos y su interpretación posterior. Por lo tanto, su eliminación ayuda a simplificar los datos y a mejorar la calidad de los resultados obtenidos.

Por último, es importante mencionar que en el uso del lenguaje a menudo podemos encontrar palabras con acentos ortográficos, como por ejemplo "león.º "león". A pesar de que nuestro cerebro puede reconocer la palabra correctamente en ambos casos, es una buena práctica eliminar las tildes para mantener una coherencia y uniformidad en los comentarios. De esta manera, se evitan confusiones y se facilita el procesamiento del lenguaje natural para la tarea de predicción de comentarios virales en YouTube.

A la luz de los cambios en la estructura de la base de datos, se ha llevado a cabo un análisis de sentimiento utilizando un modelo de aprendizaje profundo llamado BERT [6] [2], el cual ha permitido la creación una nueva columna que proporciona información valiosa sobre el sentimiento de los comentarios. Cabe destacar que BERT es un modelo muy complejo que requiere una gran cantidad de tiempo y recursos para procesar todos los comentarios de la base de datos. Debido a que la variable a predecir (viralidad) presenta un desbalanceo en los datos, se ha optado por reducir la base de datos a 8700 registros, manteniendo el mismo número de comentarios virales y eliminando 20000 registros de comentarios no virales. Esto ha permitido acelerar significativamente el proceso de análisis de sentimiento, reduciendo el tiempo de ejecución de varios días a 15 horas. Esta decisión se tomó con el objetivo de obtener resultados más rápidos y para una primera aproximación del modelo.

Tambien se ha creado otra columna de sentimientos, esta vez más sencillo, utilizando NLTK. Así durante el entrenamiento tendré tanto comentarios como columnas de sentimientos.

Uno de los problemas comunes que se pueden encontrar al trabajar con modelos de aprendizaje automático es el desequilibrio de clases en los datos. Esto significa que hay una clase minoritaria que tiene muy pocos ejemplos en comparación con la clase mayoritaria, lo que puede afectar negativamente el rendimiento del modelo y su capacidad para predecir correctamente la clase minoritaria.

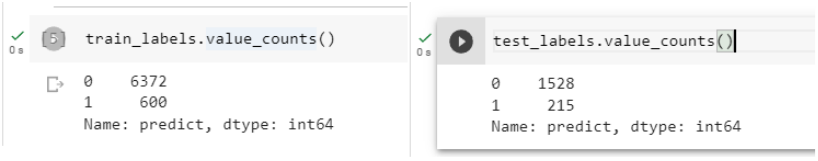


Figura 4: *Desbalanceo en la variable a predecir. "0"si no es viral "1"si es viral*

por lo que aunque en la division de nuestra base de datos, especifiquemos que lo queremos de manera "stratify", el desbalanceo sigue existiendo.

Una forma de corregir este problema es utilizando técnicas de sobremuestreo como SMOTE (Synthetic Minority Over-sampling Technique).

SMOTE es una técnica de sobremuestreo que crea nuevos ejemplos sintéticos para la clase minoritaria mediante la interpolación de ejemplos existentes. En lugar de simplemente duplicar los ejemplos existentes, SMOTE genera ejemplos sintéticos que son combinaciones lineales de ejemplos adyacentes. Esto permite crear ejemplos que son más representativos de la clase minoritaria y reduce la probabilidad de sobreajuste.

En el contexto de nuestro proyecto de aprendizaje automático, se ha utilizado SMOTE para abordar el problema de desequilibrio de clases en la variable a predecir de viralidad. Para hacer esto, se ha aplicado SMOTE solo al conjunto de entrenamiento y se ha generado un número de ejemplos sintéticos para la clase minoritaria, con el objetivo de igualar el número de ejemplos en ambas clases. Esto ha permitido entrenar un modelo más preciso y equilibrado que puede predecir con mayor precisión la clase minoritaria.

2.4. Creación del modelo:

Una vez que hemos recopilado y analizado nuestros datos, podemos proceder a la creación del modelo de clasificación de comentarios virales en YouTube. En primer lugar, se llevará a cabo una división de nuestros datos en dos conjuntos: entrenamiento y prueba. El conjunto de entrenamiento se utilizará para ajustar el modelo, mientras que el conjunto de prueba se utilizará para evaluar el rendimiento del modelo en datos no vistos previamente. En primer lugar, se ha probado con algoritmos como regresión logística, AdaBoostClassifier, MultinomialNB, ExtraTreesClassifier, RandomForestClassifier, XGBClassifier consiguiendo en todas ellas un resultado de sobreajuste. En segundo lugar, decidí utilizar un clasificador utilizando BERT, sin embargo, me encontré con que excede de la capacidad computacional que ofrece tanto mi gráfica local como google colab, encontrando el siguiente error a la hora de hacer un "fine-tuning"

> .ºutOfMemoryError: CUDA out of memory. Tried to allocate 300.00 MiB (GPU 0; 14.75 GiB total capacity; 13.68 GiB already allocated; 104.81 MiB free; 13.87 GiB reserved in total by PyTorch) If reserved memory is ≫ allocated memory try setting max split size mb to avoid fragmentation. See documentation for Memory Management and PYTORCH CUDA ALLOC CONF"

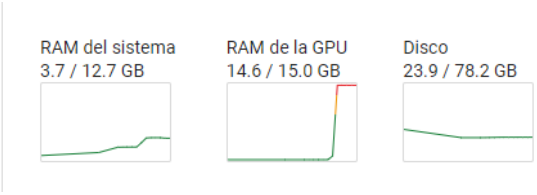


Figura 5: *Se puede observar como la memoria de la GPU está saturada.*

Es por tanto que en vez de pagar por utilizar un servicio de AWS para conseguir recursos y poder entrenar mi modelo, decidí optar por crear una red neuronal de tipo RNN.

Las RNN (Redes Neuronales Recurrentes) son un tipo de redes neuronales diseñadas específicamente para trabajar con datos secuenciales, como texto, audio o series temporales. Una de las principales ventajas de las RNN es su capacidad para capturar la información contextual y la relación temporal entre los datos.

Existen varios tipos de RNN, entre los que se encuentran SimpleRNN, GRU (Gated Recurrent Unit) y LSTM (Long Short-Term Memory). Cada uno de ellos tiene sus propias características y se utilizan para diferentes tipos de problemas.

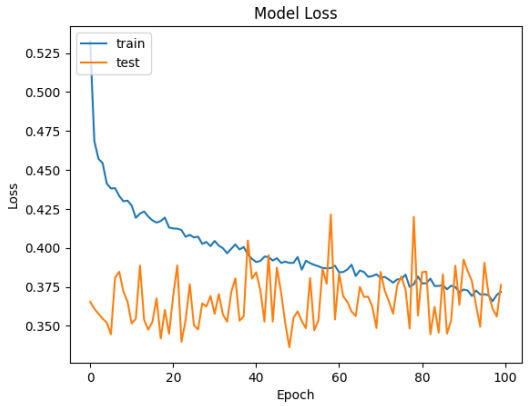
En este caso, al no contar con los recursos suficientes para pagar por un servicio en la nube como AWS, se optó por utilizar las RNN disponibles en las librerías de Python, como Keras y TensorFlow. Esto permitió entrenar el modelo utilizando los recursos disponibles en la computadora local, evitando así los costos adicionales de utilizar un servicio en la nube.

Es importante destacar que aunque el uso de RNN puede ser una alternativa más económica, es posible que no se logre la misma eficiencia y escalabilidad que se tendría al utilizar un servicio en la nube. Por lo tanto, se debe evaluar cuidadosamente si esta opción es adecuada para el proyecto en cuestión.

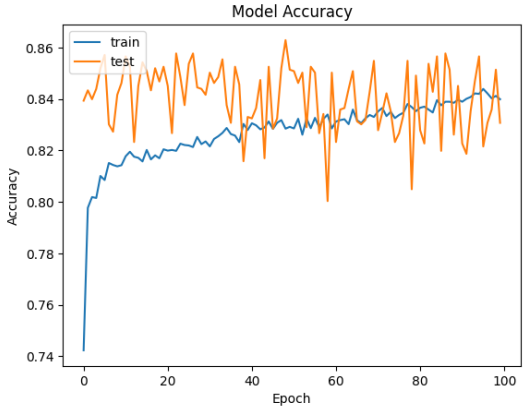
Finalmente, el modelo que generaliza para mis datos de entrenamiento y se comporta en condiciones para mis datos de test es una red neuronal basada en SimpleRNN. Uno podria esperar que fuese LSTM al ser una estructura de red neuronal recurrente más poderosa, sin embargo para este problema en concreto, tenemos comentarios que tienen pocas palabras y las LSTM están diseñadas para manejar secuencias de datos largas y complejas, y si los datos son demasiado cortos, las capas adicionales de una LSTM, basadas en puertas input, puertas olvido y puertas salida, pueden no ser necesarias. Si la tarea en cuestión no requiere una memoria a largo plazo, una SimpleRNN puede ser más efectiva que una LSTM.

2.5. Validación del modelo

Las gráficas que resultan del modelo basado en SimpleRNN son las que se muestran. El resto de curvas de aprendizaje, incluidas las LSTM, mostraban casos de sobreajuste, por lo que se omite su presentacion. En el aprendizaje automático y la inteligencia artificial, la precisión y la pérdida son dos métricas importantes para evaluar el rendimiento de un modelo. La precisión mide la capacidad del modelo para predecir correctamente las etiquetas de los datos de prueba, mientras que la pérdida mide la diferencia entre las etiquetas predichas por el modelo y las etiquetas reales de los datos de prueba.



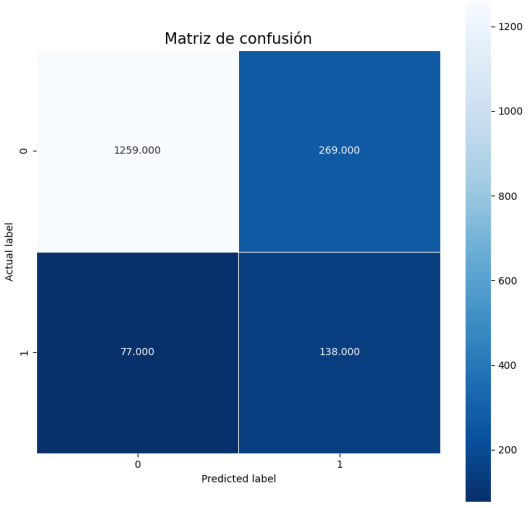
(a) Curva aprendizaje para precisión.



(b) Curva aprendizaje para precisión.

En particular, es importante notar que la gráfica de precisión muestra un aumento constante en la precisión a medida que se entrena el modelo, lo cual es una señal prometedora de que el modelo está mejorando con el tiempo. Además, la gráfica de pérdida muestra una disminución constante en la pérdida, lo cual indica que el modelo está minimizando la diferencia entre las etiquetas predichas y las etiquetas reales de los datos de prueba.

Tambien podemos ver otros estimadores relevantes cuando se hace clasificación, como es la matriz de confusión



(a) Matriz de confusión.

```python
from sklearn.metrics import precision_score, recall_score,f1_score

precision = precision_score(test_labels, predictions)
recall = recall_score(test_labels, predictions)
f1 = f1_score(test_labels, predictions)


print(f'Accuracy: {round(precision*100,2)}%')
print(f'recall: {round(recall*100,2)}%')
print(f'f1_score: {round(f1*100,2)}%')
```

```
Accuracy: 36.81%
recall: 55.81%
f1_score: 44.36%
```

(b) Precision, recall y f1-score

podemos ver que tu modelo predijo correctamente 1259 casos donde la etiqueta real era 0 y el modelo también predijo 0. Esto se encuentra en la esquina superior izquierda de la matriz. Además, el modelo predijo incorrectamente 289 casos donde la etiqueta real era 0, pero el modelo predijo 1. Esto se encuentra en la esquina superior derecha de la matriz. Por otro lado, el modelo predijo incorrectamente 77 casos donde la etiqueta real era 1, pero el modelo predijo 0. Esto se encuentra en la esquina inferior izquierda de la matriz. Finalmente, el modelo predijo correctamente 138 casos donde la etiqueta real era 1 y el modelo también predijo 1. Esto se encuentra en la esquina inferior derecha de la matriz. La matriz de confusión indica que el modelo tiene un alto rendimiento en la predicción de casos donde la etiqueta real es 0, ya que predice correctamente la mayoría de estos casos (1259) y solo tiene una pequeña cantidad de falsos positivos (289). Sin embargo, parece tener más dificultades para predecir correctamente los casos donde la etiqueta real es 1, ya que tiene un número significativo de falsos negativos (77) y un número más pequeño de verdaderos positivos (138). Es importante tener en cuenta que esta es solo una evaluación preliminar del rendimiento del modelo y que se necesitaría más información para hacer una evaluación más completa ya que en los datos de test tenemos las clases desbalanceadas.

Un dato importante es que cuando las clases están desbalanceadas, hay estimadores que son más significantes que la precisión, uno de ellos es recall donde observamos un 55,81 %. Recall es una métrica donde nos mide la sensibilidad de nuestro modelo a predecir la proporción de verdaderos positivos que se han identificado correctamente. Es una métrica útil cuando estamos más interesados en evitar los falsos negativos. En este caso los falsos negativos seria cuando un comentario que es viral se ha detectado como no viral. Aunque este número puede parecer bajo, si que abre las puertas a la creación de un mejor modelo que prediga de manera más precisa cuando un comentario va a ser viral o no.

2.6. Implementación del modelo

Hay que tener en cuenta que se está introduciendo los comentarios ya tokenizados, por lo que si tratamos introducir una cadena de texto con nuestro comentario, va a resultar en un error. Además, tambien se introduce el sentimiento tanto para BERT como para nltk, por lo que también se tendría que conseguir esta informacion de la cadena de texto. Para ello se ha realizado el codigo que se puede encontrar en mi GitHub.

Se ha creado la funcion "frase a predecir"tal que, el resultado final devolverá la probabilidad de que el comentario sea viral.

```
texto = 'dale like para que ibai participe en la siguiente velada'

frase = frase_a_predecir(texto)

model.predict(frase)
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
1/1 [==============================] - 0s 51ms/step
array([[0.31621855]], dtype=float32)
```

Figura 8: *Predicción de viralidad del comentario que se guste*

Una vez entrenado el modelo, se puede descargar este modelo con librerias como Joblib o pickle e introducirlo en una página sencilla de Streamlit, ya que utilizando esta funcion, debería ser simple de realizar.

3. Conclusión y futuros pasos

Posibles mejoras a este modelo seria realizar la clasificacion de texto basandose en BERT [1] [2] [6] , sin embargo, para BERT quedo limitado por GPU, ya que BERT es un modelo de gran tamaño con millones de parámetros, y su re-entrenamiento, requiere una gran cantidad de potencia de procesamiento y memoria. Por lo tanto, se recomienda usar al menos una GPU con al menos 12 GB de memoria para entrenar modelos BERT de tamaño mediano en conjuntos de datos de tamaño moderado. Para conjuntos de datos más grandes o modelos más grandes, se puede requerir una GPU con 16 GB o más de memoria, de la cual ni usando Google Colab puedo disponer. Tambien el uso de AutoGPT [5] [4], para hacer analisis de sentimiento utilizando su API "text-davinci-002.[en] el caso de analisis de sentimiento. Sin embargo, tambien quedo limitado por el numero de llamadas que puedo hacer a la API. Más mejoras al modelo seria conseguir más datos de comentarios virales realizando un cierto histórico. Por ejemplo, no solo basarse en comentarios de La Velada del Año, si no tener todo el historico de su canal de YouTube ya que en los comentarios nos daría información de todo el tráfico y público que tiene. Así, se tendría más comentarios virales y nos daría información de qué tipo de comentario es el más buscado. Otra posible mejora sería realizar más entrenamientos con diferentes conjuntos de hiperparámetros, con el objetivo de encontrar la combinación óptima de parámetros para mejorar el desempeño del modelo.

En resumen, esta metodología de análisis de sentimiento de comentarios podría ser útil para cualquier estrategia publicitaria que involucre la presentación de reseñas de productos o servicios. En lugar de simplemente mostrar los comentarios más populares, esta metodología permite identificar los comentarios que son más relevantes y valiosos para la audiencia. Además, puede ser especialmente útil en las primeras horas después de la publicación de una reseña, cuando aún no hay muchos votos, para ordenar y optimizar los comentarios que se muestran. Una vez pasado el tiempo, mostrar también aquellos comentarios con grandes tendencias a ser virales y no que quede en el olvido para así dar una mejor experiencia al usuario. En definitiva, la aplicación de esta metodología puede mejorar significativamente la efectividad y eficiencia de la presentación de reseñas y comentarios en una estrategia publicitaria.

Referencias

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[2] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Brew, J. (2019). Hugging Face's Transformers: State-of-the-art Natural Language Processing. *ArXiv preprint arXiv:1910.03771*.

[3] Bird, S., Klein, E., Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

[4] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33.

[5] OpenAI. (2021). *Introducing AutoGPT: A Framework for Automatic Architecture and Hyperparameter Search for Transformers*. Recuperado el 11 de mayo de 2023, de https://openai.com/blog/introducing-autogpt/.

[6] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.