Aplicaciones de Procesamiento de Lenguaje Natural

Juan David Gonzales Mosquera1 and Juan Sebastian Montoya Franco1;
Juan_d.gonzales_m@uao.edu.co; juan_s.montoya_f@uao.edu.co
1Department of engineering, Informatic Engineering
Faculty of Engineering, Artificial Intelligence Universidad Autónoma de Occidente
Cali, Colombia

RESUMEN Este trabajo final examina la importancia de las Para el desarrollo de este proyecto se ha seleccionado el campo del análisis por sentimientos (opiniones) de textos, tanto para la aplicación de clasificación de textos como para el prototipo NLP. Es un campo de estudio que se enfoca en la extracción y clasificación de las emociones, actitudes y opiniones expresadas en el lenguaje escrito. Esta tarea es de suma importancia en el procesamiento del lenguaje natural y tiene una amplia gama de aplicaciones en diversas áreas.

La clasificación de textos por opiniones es crucial en el ámbito empresarial, ya que permite a las organizaciones comprender la percepción y el sentimiento de los clientes con respecto a sus productos o servicios. Al analizar grandes volúmenes de comentarios, reseñas y opiniones en línea, las empresas pueden obtener información valiosa sobre la satisfacción del cliente, identificar áreas de mejora y tomar decisiones estratégicas para optimizar sus productos o servicios

I. INTRODUCCIÓN

En la actualidad la clasificación de textos por opiniones es fundamental, ya que proporciona una forma eficaz de comprender las emociones y opiniones expresadas en el lenguaje escrito. Sus aplicaciones en el ámbito empresarial, político, social y académico hacen de este campo una herramienta valiosa para la toma de decisiones informadas, la comprensión de las necesidades del cliente y el desarrollo de sistemas inteligentes y personalizados.
En el ámbito del análisis de mercado, la clasificación de textos por opiniones juega un papel muy importante. Al analizar las opiniones y los sentimientos expresados por los consumidores en redes sociales, foros y sitios web, las empresas pueden evaluar la respuesta del público a nuevos productos o campañas publicitarias, identificar tendencias emergentes y tomar decisiones informadas para adaptarse al mercado.

En el ámbito de la política y la sociedad, la clasificación de textos por opiniones también es de gran relevancia. Permite analizar la opinión pública sobre temas candentes, evaluar la percepción de los ciudadanos hacia los líderes políticos y comprender la dinámica de los debates en línea. Esto puede ser especialmente útil para los gobiernos y los responsables de la toma de decisiones, ya que pueden tomar medidas basadas en una comprensión más profunda de las opiniones y preocupaciones de la población.

Además de estas aplicaciones comerciales y políticas, la clasificación de textos por opiniones tiene un papel importante en la investigación académica y en la creación de sistemas inteligentes. El análisis de sentimientos es una parte integral de la inteligencia artificial y del aprendizaje automático, y se utiliza en el desarrollo de chatbots, asistentes virtuales y sistemas de recomendación personalizados

II. METODOLOGÍA

Para el desarrollo de este proyecto fue necesario realizar una investigación de escritorio de aplicaciones de clasificación de texto, por lo que hemos escogido la clasificación por sentimiento/opiniones de textos, en este caso de un conjunto de datos provenientes de tweets. Posteriormente se buscan los modelos y conjuntos de datos que pueden ser de relevancia para el entrenamiento y clasificación de los textos para finalmente realizar la evaluación de los modelos y generar una interfaz de usuario amigable para su interacción.

III. PROBLEMÁTICA

A. CLASIFICACIÓN DE TEXTO POR OPINIÓN Y APLICACIÓN NLP

Para el desarrollo de este proyecto se ha utilizado la misma dinámica de clasificación por sentimientos para la investigación y entrenamiento del modelo y el desarrollo del aplicativo NLP.

Aquí mencionamos algunos contextos en los que puede ser aplicado este proyecto:

- **Análisis de la satisfacción del cliente:** Las empresas a menudo enfrentan el desafío de comprender y evaluar la satisfacción de sus clientes. La clasificación de textos por opiniones permite analizar comentarios, reseñas y feedback de los clientes para identificar patrones y tendencias, y así comprender mejor las necesidades y expectativas de los clientes. Esto puede ayudar a abordar problemas específicos, mejorar la calidad de los productos o servicios, y fortalecer las relaciones con los clientes.

* Revista Argentina de Trabajos Estudiantiles. Patrocinada por la IEEE.

- **Detección y gestión de opiniones negativas:** Las opiniones negativas pueden tener un impacto significativo en la reputación de una empresa. El análisis de sentimientos puede ayudar a detectar y gestionar rápidamente comentarios o reseñas negativas, permitiendo una respuesta oportuna y adecuada. Esto puede incluir la resolución de problemas, la atención al cliente mejorada y la implementación de medidas correctivas para mantener la satisfacción del cliente y preservar la imagen de la marca.

- **Evaluación de campañas de marketing:** Las empresas invierten en campañas de marketing para promover sus productos o servicios. El análisis de sentimientos puede evaluar la respuesta del público a estas campañas, identificando la efectividad de los mensajes y la percepción de los consumidores. Esto permite ajustar y mejorar las estrategias de marketing para lograr una mayor resonancia y alcanzar los objetivos deseados.

- **Identificación de tendencias y preferencias del mercado:** El análisis de sentimientos puede ayudar a las empresas a identificar tendencias emergentes y cambios en las preferencias del mercado. Al comprender las opiniones y los sentimientos de los consumidores expresados en diferentes plataformas, se pueden identificar oportunidades de negocio, lanzar productos o servicios innovadores y ajustar la oferta para adaptarse a las demandas del mercado.

- **Detección de noticias falsas y opiniones engañosas:** La propagación de noticias falsas y opiniones engañosas en línea es un desafío importante en la actualidad. El análisis de sentimientos puede contribuir a detectar y filtrar información falsa o engañosa, ayudando a los usuarios a tomar decisiones informadas y fomentando una mayor confianza en la información disponible en línea.

IV. PROTOTIPO DE APLICACIÓN NLP

A. MODELOS

- **Hugging Face:** se basa en el concepto de "transformer", que es un tipo de arquitectura de red neuronal diseñada para capturar relaciones a largo plazo en conjuntos de datos secuenciales, como texto. Estos modelos han logrado avances significativos en tareas de NLP, como el reconocimiento de entidades, la traducción automática y la generación de texto.

  La biblioteca "transformers" de Hugging Face proporciona una interfaz sencilla para acceder y utilizar una amplia variedad de modelos de lenguaje preentrenados, incluidos los modelos de transformer más conocidos, como BERT, GPT y RoBERTa. Estos modelos se entrenan en grandes cantidades de texto para aprender representaciones lingüísticas generales, lo que les permite realizar tareas de NLP con un rendimiento impresionante.

  El modelo Hugging Face es una biblioteca y plataforma que brinda acceso a modelos de lenguaje preentrenados de vanguardia y herramientas para el procesamiento del lenguaje natural. Ha ganado popularidad por su facilidad de uso y su capacidad para ayudar a los investigadores y desarrolladores a aprovechar el poder de los transformers en sus aplicaciones de IA

- **DistilBERT:** utiliza una técnica llamada "destilación de conocimiento" para comprimir la información relevante aprendida por BERT en un modelo más pequeño. En lugar de entrenar DistilBERT desde cero, se toma un modelo BERT previamente entrenado y se realiza un proceso de transferencia de conocimiento para que DistilBERT aprenda de los pesos y parámetros del modelo más grande.

  Aunque DistilBERT tiene menos capas y una menor cantidad de parámetros que BERT, logra mantener una calidad de representación de lenguaje similar. Esto se debe a que la destilación de conocimiento captura las características más importantes y las generaliza en un modelo más compacto. La reducción en la complejidad de DistilBERT lo hace más eficiente en términos de tiempo y recursos computacionales, lo que lo hace adecuado para aplicaciones con limitaciones de hardware o donde se requiere un procesamiento más rápido.

  DistilBERT es una versión más pequeña y eficiente del modelo BERT. Utiliza destilación de conocimiento para comprimir la información aprendida por BERT en un modelo más compacto, manteniendo una calidad de representación similar. DistilBERT es adecuado para aplicaciones con limitaciones de recursos computacionales y ha demostrado ser efectivo en una variedad de tareas de procesamiento del lenguaje natural

- **Wav2vec2:** es un modelo de aprendizaje automático utilizado para la transcripción automática de voz a texto. Aprovecha las redes neuronales convolucionales y los modelos de lenguaje de transformadores para aprender representaciones significativas directamente de la señal de audio. Mediante el uso de una tarea previa de predicción de audio enmascarado, el modelo se entrena para transcribir voz a texto sin requerir transcripciones previas. Wav2vec2 ha demostrado un rendimiento sólido y se utiliza como punto de partida para mejorar otros sistemas de procesamiento del lenguaje natural

B. BACKEND

Para el desarrollo del Backend se ha utilizado el lenguaje de programación Python, instalado en un ambiente local, para ello algunas de las siguientes librerías son necesarias:

```
pip install tensorflow
pip install torch
pip install torchaudio
pip install transformers
pip install --no-cache-dir transformers sentencepiece
pip install kenlm
pip install https://github.com/kpu/kenlm/archive/master.zip
pip install pyctcdecode
pip install flask
pip install librosa
```

Ilustración 1/ Librerías necesarias para el proyecto

Posteriormente incluimos el paso de los datos hacia el modelo:

```
def obtain_audio_information(archivo_audio):
    print('Audio information')
    #utilizamos la biblioteca librosa  para obtener la informacion del audio
    audio, frecuencia_muestreo = librosa.load(archivo_audio)
    # Retorna la duración del audio en segundos
    duration = len(audio) / frecuencia_muestreo

    #obtenemos los modelos sentiment-analysis y automatic-speech-recognition
    Clasificador = pipeline('sentiment-analysis')
    transcripcion = pipeline('automatic-speech-recognition')

    #Realizamos la transcripción y el analisis de sentimientos
    text_process = transcripcion(audio)['text']
    clasification_res = Clasificador(text_process)
```

Ilustración 2 / Función modelos

Y la respuesta que este nos brinda para poder visualizar los datos en el Frontend.

```
response = {
    "type":"success",
    "status": 200,
    "data": {
        "duration": f'{duration} segundos',
        "text_process": text_process,
        "clasification": clasification_res
    }
}
```

Ilustración 3 / Respuesta del modelo

C. FRONTEND

Para el desarrollo de la parte visual, el colectivo ha optado por realizar una página web en donde se adjunta por medio de un selector, el archivo de audio, este se envía al modelo, donde se analiza y transcribe, como resultado final obtenemos el score, el texto traducido, la duración que tiene el audio, la clasificación y el testo transcrito.

A continuación, se presenta la GUI desarrollada para el proyecto:

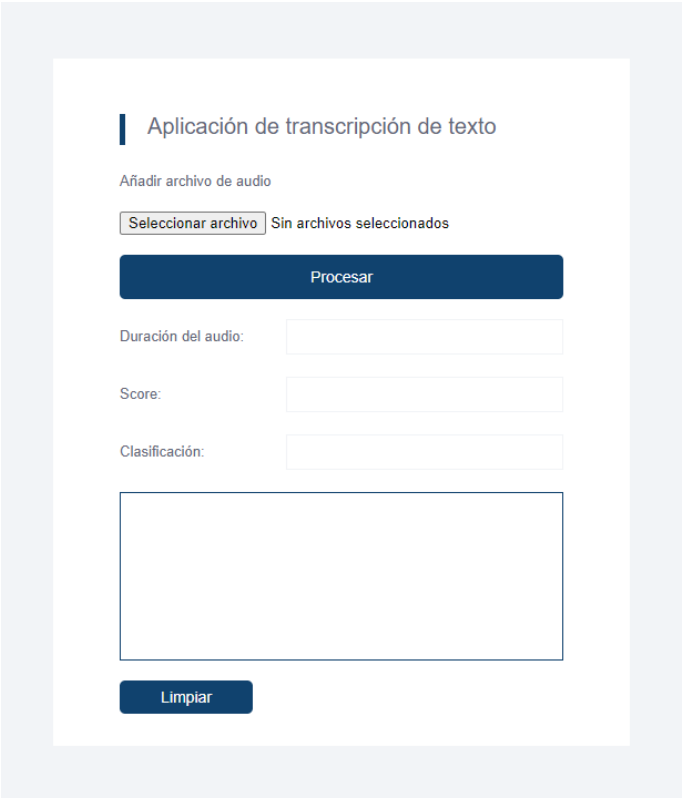

Ilustración 4 / Visualización Frontend

V. RESULTADOS

A. CLASIFICACIÓN DE TEXTOS

a. LSTM (Long Short-Term Memory)

Recordemos que el dataset utilizado para este proyecto fue el de Sentiment140 dataset con 1.6 millones de tweets, aquí hay varias cosas para tener en cuenta:

1. El dataset tiene más columnas irrelevantes para la predicción, por lo que hay que retirar lo que no es de interés (sólo dejando el sentimiento y el texto)
2. Se tienen en los textos muchos signos y palabras que no son importantes para la clasificación, por ello hay que eliminarlos. Estos textos son tweets, por lo que tienen muchos enlaces, emojis o puntuaciones, además de que los Tweets suelen llevar menciones con @ o tildes, sin mencionar palabras derivadas que la gente utiliza para abreviar o palabras mal escritas, por lo que hay que realizar Word Emdedding
3. La salida del modelo se basa en definir si el texto es positivo o negativo, esto en código son 0 o 1, por lo que hay que realizar un tokenizado a las frases o palabras

Para este modelo se ha definido la siguiente estructura de capas:

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding_sequences = embedding_layer(sequence_input)
x = SpatialDropout1D(0.2)(embedding_sequences)
x = Conv1D(64, 5, activation='relu')(x)
x = Bidirectional(LSTM(64, dropout=0.2, recurrent_dropout=0.2))(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
outputs = Dense(1, activation='sigmoid')(x)
model = tf.keras.Model(sequence_input, outputs)
```

Ilustración 5 / Modelo LSTM

- Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32'): Esta línea crea una capa de entrada para los datos de secuencia. La forma de entrada es (MAX_SEQUENCE_LENGTH,), Los elementos de las secuencias se representan como números enteros.

- Embedding_sequences = embedding_layer(sequence_input): Aquí, los datos de entrada de secuencia se pasan a una capa de embedding predefinida llamada embedding_layer. Esta capa transforma los números enteros en vectores de embedding, que son representaciones densas de baja dimensionalidad de los elementos de la secuencia.

- x = SpatialDropout1D (0.2) (embedding_sequences): Se aplica una capa de dropout espacial 1D con una tasa de dropout del 20% a los vectores de embedding. Esto ayuda a evitar el sobreajuste y mejora la capacidad de generalización del modelo.

- x = Conv1D (64, 5, activation='relu') (x): Se aplica una capa de convolución 1D con 64 filtros y un tamaño de kernel de 5. La función de activación utilizada es ReLU, que introduce no linealidad en la salida de la capa convolucional.

Este entrenamiento realizado en 10 épocas da como resultado un Accuracy (Precisión) del 78% y para el Accuracy de validación también un 78%. Sin embargo, en las pruebas se obtuvo una buena clasificación con el texto "¡This is a Good movie!":

```
# Example usage:
text = "This is a good movie!"
sentiment = predict_sentiment(text)
print("Sentiment:", sentiment)
```

```
1/1 [==============================] - 1s 689ms/step
Sentiment: Positive
```

Ilustración 6 / Pruebas modelo LSTM

A continuación, las gráficas de la precisión y la pérdida:

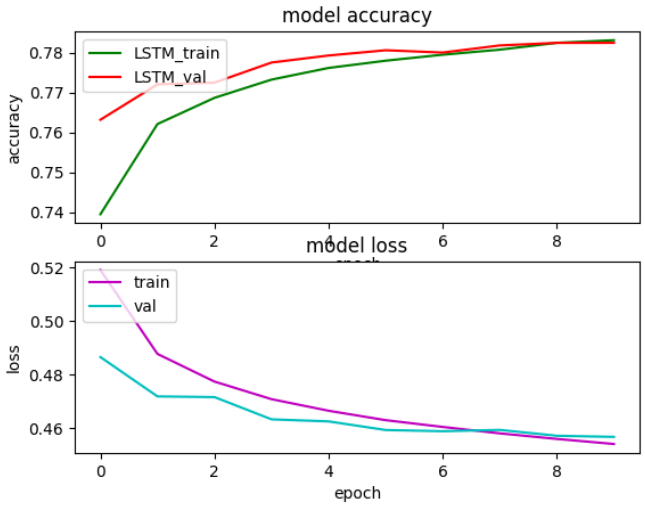

Ilustración 7 / Gráficas modelo LSTM

b. Transformers

Para este modelo se ha definido la siguiente estructura con un modelo pre-entrenado como lo es BERT (Bidirectional Encoder Representations from Transformers) el cual tiene la capacidad para comprender el contexto de las palabras en una oración al tomar en cuenta tanto las palabras anteriores como las posteriores a través del uso de la atención bidireccional en la arquitectura de Transformers.

Arquitectura del modelo:

```
def get_model(**kwargs):
    max_seq_length = kwargs.get('max_seq_length',55)

    input_ids = tf.keras.Input(shape=(max_seq_length,),dtype='int32')
    attention_mask = tf.keras.Input(shape=(max_seq_length,),dtype='int32')

    transformer = model({'input_ids':input_ids, 'attention_mask':attention_mask},
                        training=False)
    pooler_output  = transformer["pooler_output"]

    # Model Head
    h1 = tf.keras.layers.Dense(128, activation='relu')(pooler_output)
    dropout = tf.keras.layers.Dropout(0.2)(h1)
    output = tf.keras.layers.Dense(1, activation='sigmoid')(dropout)

    new_model = tf.keras.models.Model(inputs = [input_ids, attention_mask],
                                      outputs = output)
    new_model.compile(tf.keras.optimizers.Adam(lr=1e-4),
                      loss='binary_crossentropy',
                      metrics=['accuracy'])

    return new_model
```

Ilustración 8 / Modelo Transformers

Con esto obtuvimos los siguientes resultados:

```
Epoch 10/10
175/175 [==============================] - 106s 608ms/step - loss: 0.6932 - accuracy: 0.4938 - val_loss: 0.6931 - val_accuracy: 0.5014
```

Con un accuracy del 49% y un accuracy de validación del 50%, no muy buenos resultados, esto puede ser por la cantidad de datos con los que se entrena, ya que el dataset tiene 1.6 billones de tweets y aquí le estamos psando poco mas de 3000, debido a la capacidad de Google Colab.

B. APLICACIÓN NLP

Para la aplicación de NLP y los modelos empleados obtenemos los siguientes resultados mediante la GUI realizada:

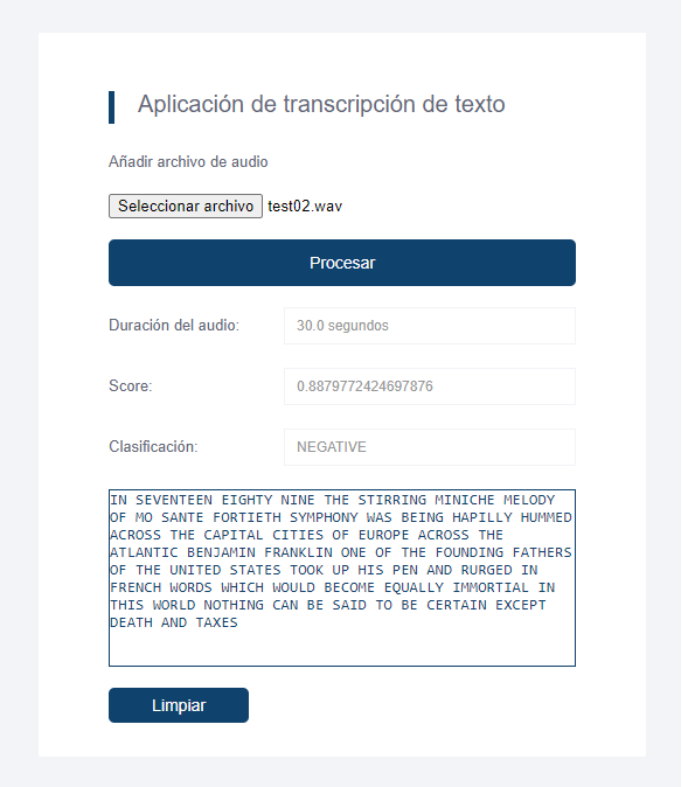

Ilustración 9 / Ressultados en Fronted

En donde le pasamos un archivo en formato .Wav (Audio), el modelo lo procesa y devulve la duración del audio (30 segundos), la precisión-score (88%), la clasificación de sentimiento (NEGATIVO) finalmente el texto transcrito

VI. CONCLUSIONES

Desarrollar un proyecto basado en la clasificación de textos por opiniones proporciona beneficios tangibles, como la mejora de la satisfacción del cliente, la toma de decisiones informadas, la gestión de la reputación y la identificación de oportunidades de mercado. Además, contribuye a una mayor eficiencia en la utilización de recursos y ayuda a las organizaciones a mantenerse actualizadas en un entorno empresarial en constante evolución. Por lo tanto, la importancia de desarrollar un proyecto con esta tecnología radica en su capacidad para impulsar el crecimiento y el éxito sostenible de las empresas en diversos sectores.

En el entrenamiento y clasificación del modelo podemos observar que hay bastante diferencia en el entrenamiento a través de LSTM Y Transformers, los resultados discrepan mucho entre ellos, puede ser debido a las capas utilizadas o épocas de entrenamiento. Sin embargo, debido a las limitaciones por la plataforma de Google Colab no se pudieron mejorar los resultados por el consumo de recursos.

Los resultados obtenidos por LSTM, fueron bastante buenos, ya que le hemos pasado varios tipos de texto como pruebas y a pesar de tener una precisión del 78% ha clasificado muy bien las pruebas.

El aplicativo de NLP puede estar limitado debido a el idioma en que procesa los audios, ya que solo lo hace con audios en inglés.

VII. REFERENCIAS

[1] *Hugging Face "distilbert-base-uncased-finetuned-sst-2-english. Available in: https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english*
[2] Hugging Face "facebook/wav2vec2-base-960h". Available in: https://huggingface.co/facebook/wav2vec2-base-960h
[3] ΜΑΡΙΟΣ ΜΙΧΑΗΛΙΔΗΣ ΚΑΖΑΝΟΥΑ. Sentiment 140 dataset with 1.6 million tweets. Available in: https://www.kaggle.com/datasets/kazanova/sentiment140
[4] Juan David Gonzalez Mosquera Github. Available in: https://github.com/TOYCRESJDGM
[5] Juan Sebastian Montoya Franco. Github. Available in: https://github.com/JuanSebastianMontoyaFranco

VIII. AUTORES:

JUAN DAVID GONZALEZ MOSQUERA actualmente se desempeña como desarrollador backend, con gran interés en el desarrollo de aplicaciones web, está finalizando una especialización en inteligencia artificial en la Universidad Autónoma de Occidente.

JUAN SEBASTIAN MONTOYA FRANCO actualmente se desempeña como desarrollador Backend en plataformas de eCommerce, está finalizando una especialización en inteligencia artificial en la universidad Autónoma de Occidente