Informe sobre el Sistema de Predicción de Sentimientos Basado en Inteligencia Artificial

1. Introducción

Este informe describe el desarrollo de un sistema de predicción de sentimientos basado en inteligencia artificial, cuyo propósito es analizar y clasificar textos en tres categorías: **Positivo, Neutral y Negativo**. El proceso de desarrollo se divide en dos fases principales. En la primera, se entrenó un modelo de aprendizaje profundo utilizando redes neuronales recurrentes para el análisis de sentimientos. En la segunda, se implementó una aplicación web en Flask que permite la interacción con el modelo, proporcionando predicciones en tiempo real y permitiendo el análisis de múltiples frases a partir de archivos de texto.

El sistema se construyó utilizando **TensorFlow y Keras** para el entrenamiento del modelo, mientras que Flask fue la tecnología elegida para la implementación de la aplicación web. La combinación de estas herramientas permitió la creación de una solución funcional y eficiente para la clasificación de sentimientos. A continuación, se detalla cada una de las fases del desarrollo, desde la preparación de los datos hasta la implementación del modelo en un entorno accesible para los usuarios.

2. Generación del Conjunto de Datos

Para entrenar el modelo, se utilizó un conjunto de datos sintético almacenado en un archivo de texto llamado datos_entrenamiento2.txt. Este conjunto de datos contiene frases etiquetadas con su respectivo sentimiento, lo que permite a la red neuronal aprender patrones en los textos y asociarlos con categorías específicas. La carga de estos datos se realizó a través de un script en Python que procesó el archivo línea por línea, eliminando caracteres problemáticos y separando las frases de sus etiquetas:

```
def cargar_datos(archivo):
    preguntas = []
    respuestas = []
    with open(archivo, 'r', encoding='utf-8') as file:
        for line in file:
            try:
                line = re.sub(r'[\u200b\u200c\u200d\u200e\u200f]', '', line)  # Eliminar ca
                pregunta, respuesta = line.strip().split(':')  # Separar frase y etiqueta
                preguntas.append(pregunta.strip())
                respuestas.append(respuesta.strip())
            except ValueError:
                continue  # Ignorar líneas mal formateadas
    return np.array(preguntas), np.array(respuestas)
```

Una vez extraídas las frases y etiquetas, se procedió a la conversión de las etiquetas en valores numéricos utilizando LabelEncoder, una técnica común en el procesamiento de datos categóricos. Para garantizar que el modelo recibiera un conjunto de datos equilibrado, se aplicó un filtro que eliminó aquellas clases con menos de dos muestras:

```
from collections import Counter
class_counts = Counter(y)
min_samples_per_class = 2
valid_classes = [key for key, value in class_counts.items() if value >= min_samples_per_class
filtered_data = [(pregunta, respuesta) for pregunta, respuesta in zip(preguntas, y) if respue
preguntas_filtradas, respuestas_filtradas = zip(*filtered_data)
```

Posteriormente, se realizó la división del conjunto de datos en entrenamiento y prueba, asignando el 80% de las muestras para el entrenamiento del modelo y el 20% restante para su evaluación:

```
from sklearn.model_selection import train_test_split

x_train_raw, x_test_raw, y_train, y_test = train_test_split(
    preguntas_filtradas, y_reindexado, test_size=0.2, random_state=42, stratify=y_reindexado
)
```

3. Entrenamiento del Modelo de Predicción de Sentimientos

El modelo de predicción fue construido utilizando una arquitectura basada en redes neuronales recurrentes, específicamente una **Bidirectional LSTM** (Long Short-Term Memory), una variante de las redes LSTM que permite capturar patrones en secuencias de texto en ambas direcciones. La tokenización del texto fue un paso esencial en la preparación de los datos, por lo que se empleó TextVectorization para convertir las frases en secuencias de enteros, asignando un máximo de 10,000 tokens únicos:

```
import tensorflow as tf
text_vectorizer = tf.keras.layers.TextVectorization(output_mode='int', max_tokens=10000)
text_vectorizer.adapt(x_train_raw)
```

La estructura del modelo incluyó una capa de **embedding**, seguida por una capa **Bidirectional LSTM** con mecanismos de dropout y recurrent dropout. Además, se agregaron dos capas densas con activación **ReLU** y regularización **L2**. Finalmente, se incluyó una capa de salida con activación **softmax** para la clasificación multiclase:

```
from keras import regularizers

def construir_modelo(text_vectorizer, num_clases=3):
    model = tf.keras.Sequential([
        text_vectorizer,
        tf.keras.layers.Embedding(input_dim=len(text_vectorizer.get_vocabulary()) + 1, output_d
        tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16, dropout=0.5, recurrent_dropout=0
        tf.keras.layers.Dense(10, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
        tf.keras.layers.Dense(10, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
        tf.keras.layers.Dense(num_clases, activation='softmax')
    ])
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'
    return model
```

Finalmente, el modelo fue guardado en el formato TensorFlow para su posterior uso en la aplicación Flask:

```
modelo.save("modelo_sentimiento", save_format="tf")
```

4. Implementación en Flask para Predicción de opiniones

La aplicación Flask carga el modelo previamente entrenado y permite recibir textos como entrada para analizar su sentimiento:

```
from flask import Flask, render_template, request, jsonify
import tensorflow as tf
import numpy as np

app = Flask(__name__)

modelo = tf.keras.models.load_model("modelo_sentimiento")

etiquetas = {0: "Negativo", 1: "Neutral", 2: "Positivo"}

def predecir_sentimiento(model, texto):
    entrada = [texto]
    probabilidades = model.predict(entrada, verbose=0)
    indice_pred = np.argmax(probabilidades, axis=1)[0]
    return etiquetas[indice_pred], texto
```

Además, se implementó la funcionalidad para analizar archivos de texto subidos por el usuario. Se procesan las frases por lotes y se genera un gráfico de distribución de sentimientos:

```python
def generar_grafico_pie(conteo):
    import matplotlib.pyplot as plt
    import io

    labels = list(conteo.keys())
    sizes = list(conteo.values())
    fig, ax = plt.subplots()
    ax.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['#FFC107' , '#F44336', '#4CAF50'])
    ax.axis('equal')

    img = io.BytesIO()
    plt.savefig(img, format='png')
    img.seek(0)
    plt.close()

    return img.read().hex()
```

5. Conclusión

Este sistema de análisis de sentimientos combina inteligencia artificial y desarrollo web para ofrecer una herramienta funcional y precisa. La arquitectura basada en **LSTM bidireccionales** permite capturar el contexto de las frases, mientras que la aplicación Flask proporciona una interfaz accesible para la predicción y análisis de textos.

Futuras mejoras pueden incluir el entrenamiento con conjuntos de datos más grandes, la integración de modelos más avanzados como transformers, y el despliegue en la nube para mayor escalabilidad.