Documentación del Proyecto de Inferencia Narrativa con IA

1. Descripción del Proyecto

El Proyecto de Inferencia Narrativa con IA tiene como objetivo entrenar un modelo de aprendizaje automático para clasificar el nivel de inferencia necesario para comprender una historia narrativa. Los niveles de inferencia son:

- **Literal (Nivel 1)**: Comprensión directa de los hechos explícitos.
- **Causal (Nivel 2)**: Entender relaciones de causa y efecto en la narrativa.
- **Crítica o Evaluativa (Nivel 3)**: Interpretar emociones, intenciones o implicaciones más profundas.

El proyecto utiliza un dataset basado en el formato *Story Cloze Test*, que incluye historias cortas con un final correcto y un final incorrecto, junto con su nivel de inferencia. El modelo entrenado puede predecir el nivel de inferencia de un nuevo texto a través de una API Flask.

1.1. Aplicaciones

- **Educación**: Evaluar la complejidad de textos para estudiantes.
- **Procesamiento de Lenguaje Natural (PLN)**: Clasificación y generación de narrativas.
- **Evaluación Automática**: Determinar el nivel de comprensión necesario para textos.

2. Estructura del Proyecto

- `/datos`: Contiene el archivo `ROCStories_Ejemplo_Combinado.xlsx` con los datos de entrenamiento.
- `/modelo`: Almacena el modelo entrenado (`modelo_inferencia.pkl`) y el vectorizador (`vectorizador.pkl`).
- `/api`: Incluye el microservicio Flask (`microservicio_flask.py`) para realizar predicciones.

3. Requisitos

Para ejecutar el proyecto, necesitas:

- Python 3.8 o superior.
- Dependencias (instálalas con `pip`):

```
pip install pandas scikit-learn joblib flask
```

- Un entorno con acceso al archivo Excel en `/datos`.

4. Instalación

1. **Clona el repositorio** (si aplica) o asegúrate de tener los archivos del proyecto.
2. **Crea un entorno virtual** (opcional, pero recomendado):

```
python -m venv venv
source venv/bin/activate  % En Windows: venv\Scripts\activate
```

3. **Instala las dependencias**:

```
pip install -r requirements.txt
```

   Si no tienes un archivo `requirements.txt`, instala manualmente:

```
pip install pandas scikit-learn joblib flask
```

4. **Verifica la estructura**: Asegúrate de que las carpetas `/datos`, `/modelo` y `/api` estén en el directorio raíz, junto con los archivos `entrenar_modelo.py` y `microservicio_flask.py`.

5. Uso

5.1. Entrenar el Modelo

El script `entrenar_modelo.py` entrena un modelo de regresión logística usando el dataset en `/datos`. Sigue estos pasos:

1. Asegúrate de que el archivo `ROCStories_Ejemplo_Combinado.xlsx` esté en `/datos`.
2. Ejecuta el script:

```
python entrenar_modelo.py
```

3. Esto genera dos archivos en `/modelo`:
   - `modelo_inferencia.pkl`: El modelo entrenado.
   - `vectorizador.pkl`: El vectorizador TF-IDF.

El script combina las oraciones de cada historia, las vectoriza usando TF-IDF y entrena un modelo de regresión logística para predecir el nivel numérico (1, 2 o 3).

5.2. Ejecutar la API

El microservicio Flask permite realizar predicciones en tiempo real. Para usarlo:

1. Asegúrate de que los archivos `modelo_inferencia.pkl` y `vectorizador.pkl` estén en `/modelo`.
2. Ejecuta el script:

```
python api/microservicio_flask.py
```

3. La API se ejecutará en `http://localhost:5000`.

5.3. Hacer Predicciones

Envía una solicitud POST a la ruta `/inferir` con un JSON que contenga el texto de la historia. Ejemplo:

```
curl -X POST http://localhost:5000/inferir -H "Content-Type:
    application/json" -d '{"texto": "Pedro se quedó callado durante la
    reunión. Todos notaron su comportamiento extraño. Él no levantó la
    mirada ni una vez. Cuando terminó, salió sin decir nada. Estaba
    decepcionado con la decisión tomada."}'
```

Respuesta esperada:

```
{ "nivel": 3 }
```

Esto indica que el texto requiere una inferencia crítica (nivel 3).

6. Detalles Técnicos

6.1. Dataset

El archivo `ROCStories_Ejemplo_Combinado.xlsx` contiene historias con las siguientes columnas:

- **ID_Historia**: Identificador único.
- **Oración_1 a Oración_4**: Partes de la historia.
- **Final_Correcto**: Final lógico de la historia.
- **Final_Incorrecto**: Final ilógico (no usado en el entrenamiento).
- **Nivel_Numérico**: Nivel de inferencia (1 = Literal, 2 = Causal, 3 = Crítica).
- **Nivel_Eduquest**: Descripción del nivel.

Ejemplo:

ID	Oración 1	Oración 2	Oración 3	Oración 4	Final Correcto
1	Luis se despertó tarde...	Saltó de la cama...	Salió corriendo...	Tomó el primer bus...	Llegó a tiempo...

6.2. Proceso de Entrenamiento

1. **Carga de Datos**: Se lee el archivo Excel con `pandas`.
2. **Preprocesamiento**: Las oraciones y el final correcto se concatenan en un solo texto.
3. **Vectorización**: Se usa `TfidfVectorizer` para convertir los textos en vectores numéricos.
4. **Entrenamiento**: Se entrena un modelo de regresión logística (`LogisticRegression`) para clasificar el nivel numérico.
5. **Guardado**: El modelo y el vectorizador se guardan con `joblib`.

6.3. API Flask

- **Endpoint**: `/inferir` (POST)
- **Entrada**: JSON con un campo `texto` (string).
- **Salida**: JSON con el campo `nivel` (entero: 1, 2 o 3).
- **Funcionamiento**: El texto se vectoriza con el vectorizador guardado, y el modelo predice el nivel de inferencia.

7. Ejemplo Completo

1. Entrena el modelo:

```
python entrenar_modelo.py
```

2. Inicia la API:

```
python api/microservicio_flask.py
```

3. Envía una solicitud:

```
curl -X POST http://localhost:5000/inferir -H "Content-Type:
    application/json" -d '{"texto": "Marta encontró una billetera en
     la calle. La abrió y vio una cédula adentro. Buscó el número en
     redes sociales. Llamó al dueño para devolvérsela. El dueño la
    agradeció con una sonrisa."}'
```

4. Recibe:

```
{ "nivel": 1 }
```

8. Notas Adicionales

- **Escalabilidad**: Para datasets más grandes, considera optimizar el modelo (por ejemplo, usar redes neuronales) o ajustar los hiperparámetros de `TfidfVectorizer` o `LogisticRegression`.
- **Limitaciones**: El modelo depende de la calidad y tamaño del dataset. Un dataset pequeño puede limitar la generalización.
- **Mejoras Futuras**:
  - Incorporar modelos más avanzados (e.g., transformers como BERT).
  - Ampliar el dataset con más historias.
  - Agregar validación cruzada para mejorar la robustez del modelo.

9. Preguntas Frecuentes

£Qué pasa si el texto enviado no sigue el formato de cinco oraciones? El modelo puede procesar textos de cualquier longitud, pero su precisión depende de que el texto sea similar a los datos de entrenamiento (historias narrativas claras).

£Cómo puedo contribuir al proyecto?

- Agrega más datos al archivo Excel.

- Experimenta con otros algoritmos de clasificación en `entrenar_modelo.py`.
- Mejora la API con nuevas funcionalidades, como validación de entrada.