DICIEMBRE 2024 AUX. JOSE MONTENEGRO

MANUAL TECNICO

PROYECTO FASE 2

GRUPO 11
Erick Abdul Chacon Barillas - 201807169
Nataly Saraí Guzmán Duarte - 202001570
Ana Belén Contreras Orozco - 201901604

Introducción

En el presente documento se describen los pasos, herramientas y procesos necesarios para desarrollar un modelo de inteligencia artificial (IA) que interprete entradas de texto en español e inglés y proporcione respuestas coherentes. Este proyecto, asignado a estudiantes de Ingeniería en Ciencias y Sistemas, busca aplicar conocimientos adquiridos en clases magistrales y laboratorios, fomentando el aprendizaje incremental a través de la implementación práctica de un modelo funcional. Además, se contempla el diseño de una interfaz gráfica que facilite la interacción del usuario con el modelo, permitiendo una experiencia intuitiva y eficiente.

Objetivos

Objetivo General

Implementar un modelo de inteligencia artificial utilizando bibliotecas de JavaScript, capaz de entender entradas de texto en español e inglés y responder de manera coherente a las mismas, aplicando conocimientos teóricos y prácticos adquiridos durante el curso.

Objetivos Específicos

1. Seleccionar una biblioteca de JavaScript adecuada para desarrollar el modelo de inteligencia artificial.
2. Diseñar y entrenar un modelo que interprete entradas textuales en español e inglés y genere respuestas coherentes.
3. Implementar una interfaz gráfica interactiva para facilitar la interacción entre el usuario y el modelo.
4. Documentar de manera técnica el desarrollo del modelo y la interfaz gráfica, garantizando claridad y replicabilidad.

Descripción del Problema

En un entorno donde la interacción natural entre humanos y sistemas computacionales es cada vez más demandada, se plantea el reto de desarrollar modelos de inteligencia artificial que interpreten y respondan entradas textuales de manera efectiva en diferentes idiomas. Este desafío ofrece una oportunidad para que los estudiantes apliquen técnicas de programación, aprendizaje automático y desarrollo de interfaces, fortaleciendo sus competencias en el diseño de soluciones tecnológicas innovadoras.

El desarrollo de este modelo se llevará a cabo en un periodo de un mes, utilizando un enfoque incremental. Una vez finalizada la primera fase, se priorizará la mejora de la capacidad del modelo para responder en inglés, asegurando un desempeño óptimo en ambos idiomas.

Solución

El desarrollo del modelo de inteligencia artificial se llevó a cabo utilizando **TensorFlow**, una biblioteca robusta de aprendizaje automático, junto con **TensorFlow.js** para su posterior implementación en navegadores. El proceso incluyó la preparación de datos, diseño del modelo, entrenamiento, evaluación y exportación para uso web. A continuación, se detalla cada componente:

1. **Instalación y Configuración de Librerías:**

   Se utilizaron estas líneas para gestionar las versiones de TensorFlow y TensorFlow.js. Esto asegura compatibilidad entre el modelo entrenado en Python y su conversión al formato web.

   - **TensorFlow**: Biblioteca principal para construir y entrenar el modelo.
   - **TensorFlow.js**: Herramienta para portar modelos entrenados a entornos web.
   - **Keras**: API para simplificar la creación de redes neuronales.

```
!pip install tensorflow==2.17.0

!pip install tf-keras --upgrade

!pip uninstall tensorflow tensorflowjs keras -y

!pip install tensorflow==2.14 tensorflowjs==4.22.0 keras
```

2. **Preparación de los Datos:**

Carga del archivo `intents.json`

```python
with open('intents.json') as content:
    data = json.load(content)
```

Se cargaron los datos de entrenamiento desde un archivo JSON que contiene patrones de texto y respuestas asociados a etiquetas (`tags`). Este archivo define las intenciones que el modelo debe reconocer. Conteniendo un contenido similar a este, dividiendo los tags en español e ingles:

```json
{
    "tag": "saludo_es",
    "patterns": [
        "¡Hola!",
        "¿Cómo estás?",
        "Buenas tardes",
        "Hola, ¿qué tal?",
        "Hey!",
        "¡Qué tal!"
    ],
    "responses": [
        "¡Hola! ¿En qué puedo ayudarte?",
        "¡Hola! ¿Cómo puedo asistirte hoy?",
        "¡Hola! ¿En qué te puedo ayudar?"
    ]
    },
    {
    "tag": "saludo_en",
    "patterns": [
        "Hello",
        "Hi",
        "How are you?",
        "Good morning",
        "Hey there"
    ],
    "responses": [
        "Hi! How can I assist you today?",
        "Hello! How can I help you?",
        "Hey! How can I help you?"
    ]
},
```

Extracción de Patrones y Etiquetas:

```python
for intent in data['intents']:
    responses[intent['tag']] = intent['responses']
    for line in intent['patterns']:
        patterns.append(line)
        tags.append(intent['tag'])
```

- **patterns:** Frases de ejemplo que el modelo debe interpretar.
- **tags:** Etiquetas que identifican la intención de cada patrón.
- **responses:** Respuestas predefinidas para cada intención.

Preprocesamiento de Texto

```
def preprocess_text(text):
    text = ''.join([char for char in text if char not in string.punctuation])
    text = text.lower()
    return text

data['patterns'] = data['patterns'].apply(preprocess_text)
```

El texto se convierte a minúsculas y se eliminan los signos de puntuación para normalizar los datos y mejorar la capacidad del modelo de generalizar.

3. Tokenización y Codificación

Tokenización

```
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(data['patterns'])
train = tokenizer.texts_to_sequences(data['patterns'])
x_train = pad_sequences(train)
```

Se tokenizan las frases (se convierten a representaciones numéricas) y se ajustan todas las secuencias al mismo tamaño usando *padding*.

Codificación de Etiquetas

```
le = LabelEncoder()
y_train = le.fit_transform(data['tags'])
```

Las etiquetas (tags) se convierten a valores numéricos para que el modelo pueda procesarlas.

4. Diseño y Entrenamiento del Modelo

Arquitectura del Modelo

```
i = Input(shape=(input_shape,))
x = Embedding(vocabulary + 1, 10)(i)
x = LSTM(10, return_sequences=True)(x)
x = Flatten()(x)
x = Dense(output_length, activation='softmax')(x)
model = Model(i, x)
```

- **Capa de Entrada:** Define la longitud de las secuencias de entrada.

- **Capa de Embedding:** Convierte palabras en vectores densos para su procesamiento.
- **Capa LSTM:** Procesa las secuencias de texto, detectando patrones temporales.
- **Capa Dense:** Genera probabilidades para cada etiqueta.
- **Función de Activación `softmax`:** Escoge la etiqueta más probable como predicción.

Compilación y Entrenamiento

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=700, validation_split=0.1, batch_size=5, verbose=1)
```

- **Pérdida:** Se utilizó `sparse_categorical_crossentropy` para comparar predicciones con etiquetas.
- **Optimizador:** `adam` para un aprendizaje eficiente.
- **Entrenamiento:** Se configuraron 700 épocas con lotes pequeños para mejorar la precisión.

5. Exportación del Modelo

Guardado del Modelo y Componentes

```
model.save('my_model.h5')
with open('tokenizer.json', 'w') as f:
    json.dump(json.loads(tokenizer.to_json()), f)
with open('label_encoder.json', 'w') as f:
    json.dump({"classes": list(le.classes_)}, f)
```

Se guardaron el modelo, el tokenizador y el codificador de etiquetas para reutilización. Esto permite integrar el modelo en aplicaciones web.

Conversión a TensorFlow.js

```
!tensorflowjs_converter --input_format keras ./my_model.h5
./modelo_python
```

El modelo entrenado se convirtió al formato necesario para su uso en navegadores con **TensorFlow.js**.

6. Evaluación y Visualización:

Gráfica de Precisión

```
plt.plot(history.history['accuracy'], label='Precisión de entrenamiento')
plt.plot(history.history['val_accuracy'], label='Precisión de validación')
plt.xlabel('Épocas')
plt.ylabel('Precisión')
plt.title('Rendimiento del modelo')
plt.legend()
plt.show()
```

Se generó una gráfica para analizar el desempeño del modelo durante el entrenamiento, evaluando la precisión en los datos de entrenamiento y validación.

7. **Pruebas Interactivas:**

   El modelo se probó con un bucle interactivo, donde se procesan las entradas del usuario y se predicen las etiquetas correspondientes. Las respuestas se seleccionan en función de la etiqueta predicha:

```
while True:
    prediction_input = input("You: ")
    if prediction_input.lower() == "exit":
        print("Chatbot: ¡Adiós!")
        break
```

8. **Diseño de una interfaz gráfica:**

   Implementar una interfaz web que permita al usuario interactuar fácilmente con el modelo.

   La interfaz web para interactuar con la inteligencia artificial se realizó en React + Vite, para luego ser desplegada en GithubPages.

   Para realizar la publicación en GithubPages se realizó lo siguiente:

   En el archivo *package.json* se agregaron las siguientes líneas:

```
"homepage": "https://zenthic22.github.io/IA1_Proyecto_11"

"scripts": {

  "predeploy": "npm run build",

  "deploy": "gh-pages -d build"

}
```

   Luego para desplegar la aplicación se corrio la instrucción

```
npm run deploy
```

Conclusiones

- La realización de este proyecto permitirá a los estudiantes consolidar sus habilidades en el diseño e implementación de modelos de inteligencia artificial, así como en el desarrollo de interfaces gráficas interactivas.
- Se promoverá el aprendizaje colaborativo, la gestión eficiente de proyectos y el uso de herramientas modernas como GitHub.
- El éxito del proyecto demostrará la capacidad de los estudiantes para integrar conocimientos teóricos y prácticos, desarrollando soluciones innovadoras a problemas tecnológicos complejos.