Universidad Tecnologica de Honduras

Inge. Arnol Alfaro

Inteligencia Artificial

Carlos Luis Rivas Gomez

2018-100-402-62

Carlos Javier Perez Fuentes

2014-100-100-72

Allison Fabiola Chacon Flores

2019-100-600-46

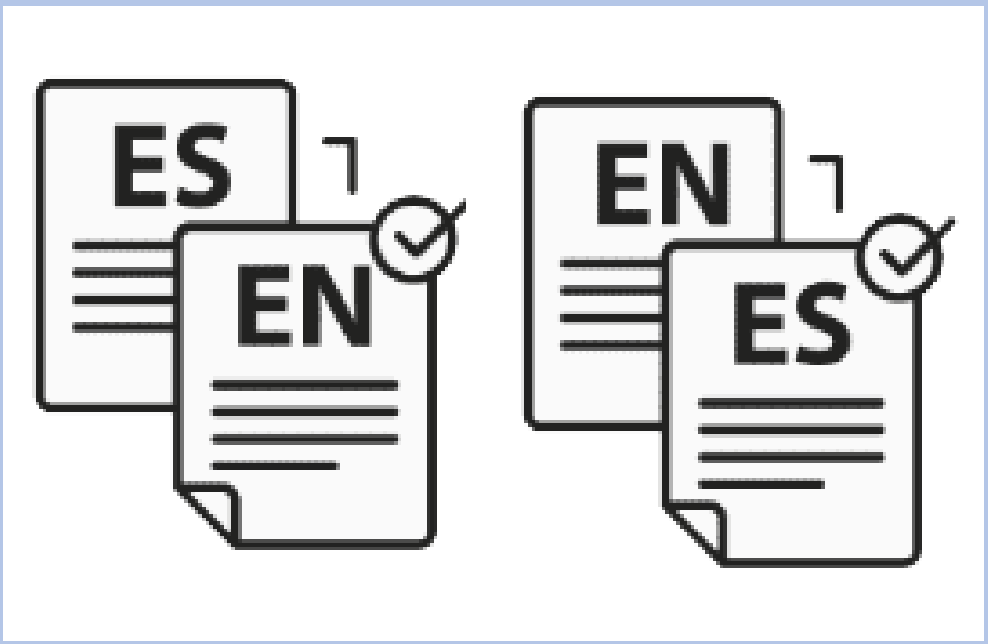
13 – 12 – 2023

Youtube: Proyecto Final traductor (youtube.com)

GitHub: https://github.com/carlosR2131/Proyecto_Final

Introducción.

El Traductor Español-inglés es una aplicación creada para facilitar la traducción de texto del español al inglés mediante el uso de redes neuronales implementadas con TensorFlow. Esta solución surge como respuesta a la necesidad de contar con una herramienta eficiente y precisa para la traducción automática entre ambos idiomas.



Problema

La barrera del idioma es un obstáculo común en la comunicación global. Traducir texto manualmente puede ser laborioso y propenso a errores, por lo que contar con una herramienta automatizada se vuelve fundamental. El proyecto aborda esta problemática al ofrecer una solución que permite a los usuarios traducir texto en español a inglés de manera rápida y precisa.

Solución

Al abordar el problema de traducción del español al inglés utilizando inteligencia artificial con Python y TensorFlow, se busca desarrollar un sistema capaz de capturar la estructura y el significado del texto en ambos idiomas para producir traducciones precisas y contextualmente relevantes.

Características Principales

Traducción Eficiente: Utiliza modelos de redes neuronales implementados con TensorFlow para lograr una traducción precisa y ágil.

Interfaz Amigable: Proporciona una interfaz sencilla e intuitiva para que los usuarios ingresen texto en español y obtengan su traducción al inglés de manera instantánea.

Precisión Mejorada: El sistema ha sido entrenado con conjuntos de datos extensos para mejorar la precisión y calidad de las traducciones.

Código app.py

```
import tensorflow as tf
import numpy as np
from flask import Flask, request, jsonify, render_template
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

app = Flask(__name__)

# Datos para entrenamiento
sentences = [
    "hola",
    "como estas",
    "buenos dias",
    "como te va",
    "el conocimiento es poder.",
    "la educacion es la clave del exito.",
    "libros",
    "El conocimiento es poder.",
    "La educación es la clave del éxito.",
    "El aprendizaje es un viaje de por vida.",
    "los maestros inspiran el futuro",
    "La lectura abre puertas a nuevos mundos.",
    "La curiosidad alimenta la educación.",
    "La naturaleza es nuestra casa, cuidémosla.",
    "en la naturaleza encontramos paz y armonia.",
    "La belleza de la naturaleza nos inspira.",
    "Cuidar el planeta es responsabilidad de todos.",
    "La biodiversidad es fundamental para nuestro futuro.",
    "Las montañas son testigos silenciosos de la grandeza de la
naturaleza.",
    "Los árboles son los pulmones de nuestro planeta.",
    "El océano es un tesoro que debemos proteger.",
    "Respetemos a los animales y su entorno natural.",
    "el aire puro es un regalo de la naturaleza."


]
translations = [
    "Hello",
    "How are you?",
    "Good morning",
    "how's it going?",
    "Knowledge is power.",
    "Education is the key to success.",
    "books",
    "Knowledge is power.",
    "Education is the key to success.",
    "Learning is a lifelong journey.",
```

```
"Learning is a lifelong journey.",
    "Teachers inspire the future.",
    "Reading opens doors to new worlds.",
    "Curiosity fuels education.",
    "Nature is our home, let's take care of it.",
    "In nature, we find peace and harmony.",
    "The beauty of nature inspires us.",
    "Taking care of the planet is everyone's responsibility.",
    "Biodiversity is crucial for our future.",
    "Mountains are silent witnesses of nature's greatness.",
    "Trees are the lungs of our planet.",
    "The ocean is a treasure we must protect.",
    "Let's respect animals and their natural habitat.",
    "Pure air is a gift from nature."


]

# Tokenización y preprocesamiento de datos
tokenizer_src = Tokenizer()
tokenizer_src.fit_on_texts(sentences)
sequences_src = tokenizer_src.texts_to_sequences(sentences)
input_data = pad_sequences(sequences_src)

tokenizer_tgt = Tokenizer()
tokenizer_tgt.fit_on_texts(translations)
sequences_tgt = tokenizer_tgt.texts_to_sequences(translations)
output_data = pad_sequences(sequences_tgt, padding='post')

# Definición del modelo
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer_src.word_index) +
1, output_dim=256, input_length=input_data.shape[1]),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=False)),
    tf.keras.layers.RepeatVector(output_data.shape[1]),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128,
return_sequences=True)),
    tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(len(tokenizer_t
gt.word_index) + 1, activation='softmax'))
])

# Compilación del modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Entrenamiento del modelo
model.fit(input_data, output_data, epochs=100)
```

```python
# Función para realizar predicciones de traducción
def translate(input_text):
    input_sequence = tokenizer_src.texts_to_sequences([input_text])
    padded_input = pad_sequences(input_sequence,
maxlen=input_data.shape[1])
    translation = model.predict(padded_input)
    predicted_sequence = np.argmax(translation, axis=-1)[0]
    predicted_text =
tokenizer_tgt.sequences_to_texts([predicted_sequence])[0]
    return predicted_text

# Ruta para cargar la página principal
@app.route('/')
def index():
    return render_template('index.html')

# Ruta para manejar la solicitud de traducción
@app.route('/translate', methods=['POST'])
def translate_text():
    user_input = request.form['input_text']
    predicted_translation = translate(user_input)
    return jsonify({'translation': predicted_translation})

if __name__ == '__main__':
    app.run(debug=True)
```