Universidad Nacional Experimental de Guayana

Vicerrectorado Académico

Proyecto de Carrera: Ingeniería en Informática

Unidad Curricular: Simulación Informática

Semestre: 2023 - 2

Aplicación Automática de Generación de Texto.

Docente:

Sergio Romero

Alumnos:

Gabriel Ramos C.I: 29.907.387

José Yepez C.I: 27.296.261

Ciudad Guayana, Febrero de 2024.

Índice

Introducción

La propuesta de proyecto se base en una Aplicación Automática que permite la generación de texto de forma automática, por lo tanto se le refiere a edición de texto a que a través de IA se permita generar un texto a partir del entrenamiento de una Red Neuronal a la cual se le ingresa información de cierto tema específico para luego generar un texto formal del tema en cuestión, con el fin de ser logrado mediante una Red Neuronal entrenado mediante TensorFlow y permitiendo el desarrollo de un generador de texto o también denominado “text generator” mediante el entrenamiento por un dataset (Información del tema) ingresado por el usuario.

La aplicación en cuestión cuenta con Interfaz Gráfica básica y amigable para el usuario la cual consta de módulos donde el usuario ingresara un lote de información sobre el tema que quiere generar un texto formal y amigable y otro apartado para visualizar el texto generado mediante información que ingresó el usuario.

Bases Teóricas

TensorFlow

Es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos.

Además de trabajar con redes neuronales, TensorFlow es multiplataforma. Trabaja con GPUs y CPUs e incluso con las unidades de procesamiento de tensores (TPUs).

Modelo

Los modelos de IA son el motor que impulsa la innovación en campos como la visión por computadora, el procesamiento de lenguaje natural y el aprendizaje automático. Los modelos de IA se utilizan para analizar patrones de datos sofisticados. Más allá del reconocimiento de patrones, también utilizan algoritmos de toma de decisiones para aprender y acercarse cada vez más a dominar sus actividades y objetivos asignados. El proceso de entrenamiento, recopilación de datos y análisis de datos es fundamental para su desarrollo.

Redes Neuronales

Es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano. Se trata de un tipo de proceso de Machine Learning llamado aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente. De esta forma, las redes neuronales artificiales intentan resolver problemas complicados, como la realización de resúmenes de documentos o el reconocimiento de rostros, con mayor precisión.

Text Generator

Un generador de texto en IA es una herramienta que utiliza modelos de lenguaje basados en redes neuronales recurrentes (RNN) para producir texto automáticamente. Estos modelos son capaces de generar contenido coherente siguiendo las instrucciones del usuario.

Pasos Implementados para desarrollar el Generador de Texto.

1. **Preparación de Datos:**

   En este paso se guarda el conjunto de datos de texto ingresado por el usuario, como escritos de artículos, páginas web entre otras fuentes.

2. **Características del Modelo**

   Se modifica el modelo con ciertas características con el fin de que funcione de forma correcta a la hora de generar el texto con los datos ingresados por el usuario mediante procesos como Caracteres como Unidades de Entrada y

Redes Neuronales Recurrentes.

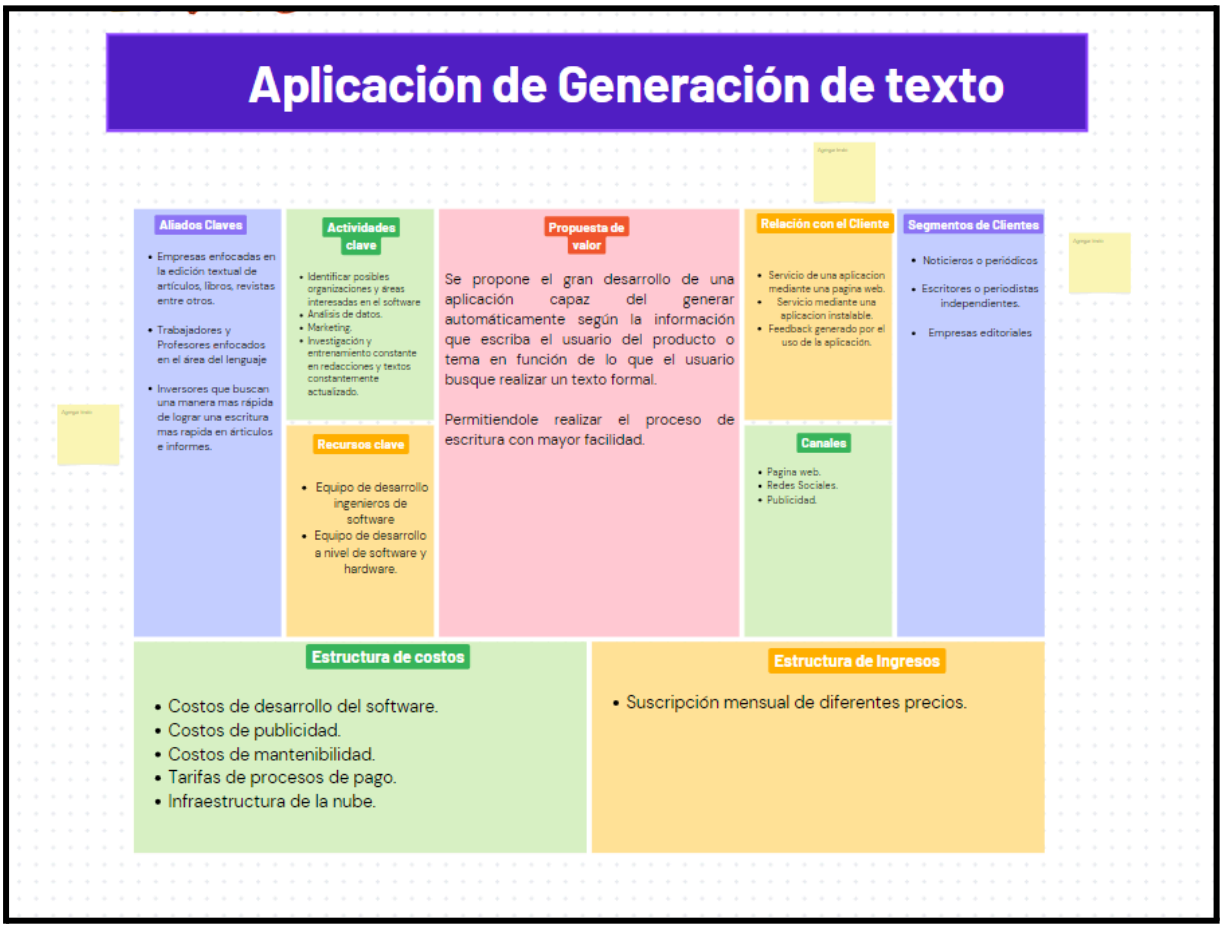
3. **Entrenamiento del Modelo**

   En este paso de ajustan un peso general para la RNN según un número de caracteres para minimizar la diferencia entre las predicciones y los caracteres reales. De igual forma en este paso el modelo aprende a predecir el siguiente carácter en función de los caracteres anteriores en la secuencia.

4. **Generación del texto.**

   Este paso es implicado de forma automática donde una vez el modelo es entrenado el modelo tendrá la capacidad de generar el texto por sí mismo, donde el usuario le ingresara una cierta palabra clave o texto relacionado con el tema en cuestión que fue ingresado en la preparación de datos para luego generar un texto formal a el usuario.

Canvas Business Model



Fragmentos de código en desarrollo de la Aplicación

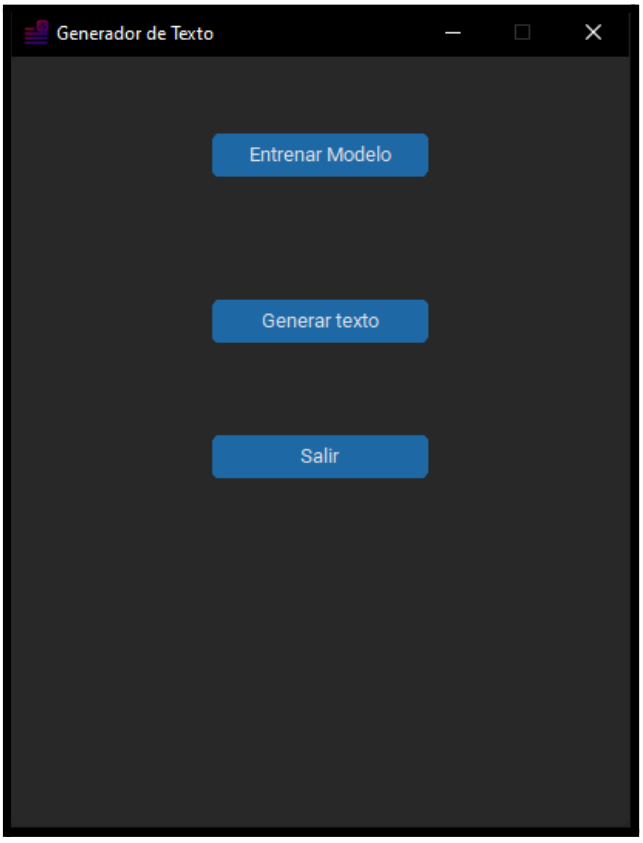
```python
def Train_new_model():
    joined_text = open("data/text.txt", "r", encoding="latin-1").read()

    partial_text = joined_text[:250000]

    tokenizer = RegexpTokenizer(r"\w+")
    tokens = tokenizer.tokenize(partial_text.lower())

    unique_tokens = np.unique(tokens)
    unique_token_index = {token: index for index, token in enumerate(unique_tokens)}
    n_words = 10  #El modelo funciona leyendo n Palabras de un texto y de alli saca la mas posible siguiente palabra
    input_words = []
    next_word = []

    for i in range(len(tokens) - n_words):
        input_words.append(tokens[i:i + n_words])
        next_word.append(tokens[i + n_words])

    X = np.zeros((len(input_words), n_words, len(unique_tokens)), dtype=bool)  # para cada muestra, se da n Palabras entrante luego un valor booleano para cada posible siguiente palabra
    y = np.zeros((len(next_word), len(unique_tokens)), dtype=bool)  # Por cada muestra, un booleano por cada posible siguiente palabra
```

```python
    for i, words in enumerate(input_words):
        for j, word in enumerate(words):
            X[i, j, unique_token_index[word]] = 1
        y[i, unique_token_index[next_word[i]]] = 1

    model = Sequential()
    Layer1= model.add(LSTM(128, input_shape=(n_words, len(unique_tokens)), return_sequences=True))
    Layer2 = model.add(LSTM(128))
    Layer3 = model.add(Dense(len(unique_tokens)))
    Layer4 = model.add(Activation("softmax"))

    optimizer = RMSprop(learning_rate=0.01)
    model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
    history = model.fit(X, y, batch_size=128, epochs=10, shuffle=True).history
    model.save("text_gen_model2.h5")

    save ('data/unique_tokens', unique_tokens)
    save ('data/unique_token_index', unique_token_index)

def Train_current_model (model_name, text): #Funcion para seguir entrenando el modelo com mas textos, hasta ahora no esta completo
```

```python
def generate_text(input_text, n_words, creativity=3): #Genera un texto
    unique_tokens =load('data/unique_tokens.npy')
    tokenizer = RegexpTokenizer(r"\w+")
    word_sequence = input_text.split()
    current = 0
    for _ in range(n_words):
        sub_sequence = " ".join(tokenizer.tokenize(" ".join(word_sequence).lower()))[current:current+n_words])
        try:
            choice = unique_tokens[random.choice(predict_next_word(sub_sequence, creativity))]
        except:
            choice = random.choice(unique_tokens)
        word_sequence.append(choice)
        current += 1
    return " ".join(word_sequence)
```

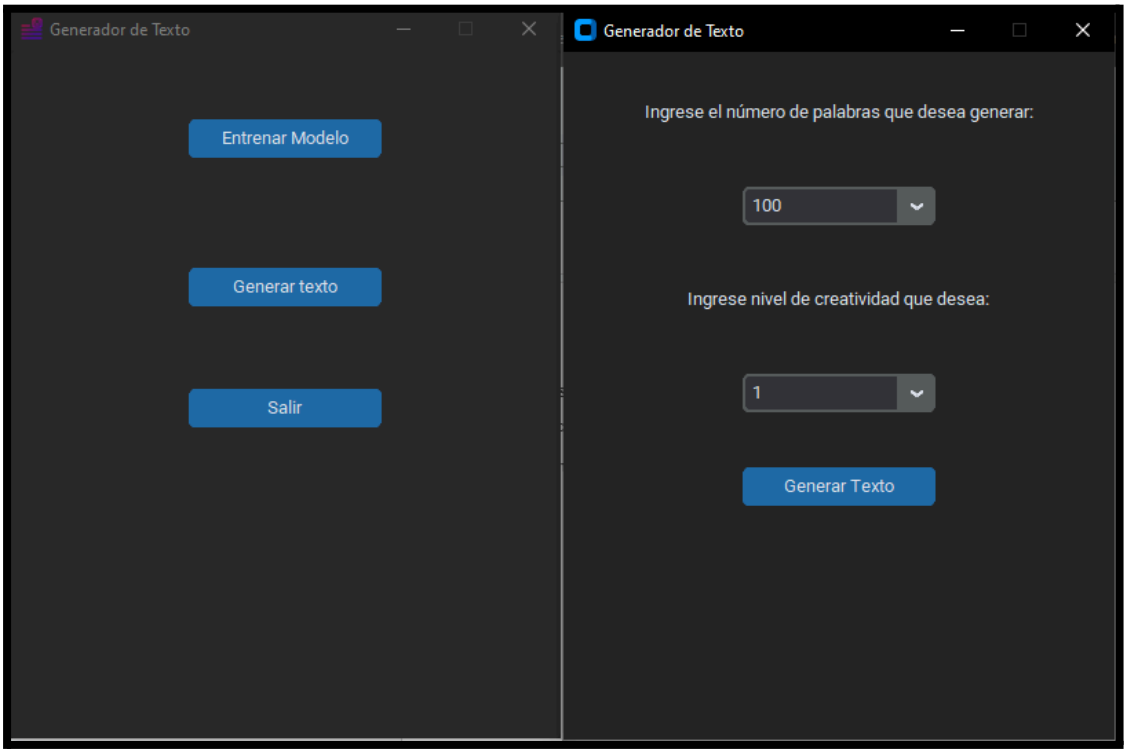
```python
def predict_next_word(input_text, n_best):

    unique_tokens =load('data/unique_tokens.npy')
    unique_token_index = load('data/unique_token_index.npy')
    model = load_model("text_gen_model2.h5")

    input_text = input_text.lower()
    X = np.zeros((1, n_words, len(unique_tokens)))
    for i, word in enumerate(input_text.split()):
        X[0, i, unique_token_index[word]] = 1

    predictions = model.predict(X)[0]
    return np.argpartition(predictions, -n_best)[-n_best:]
```

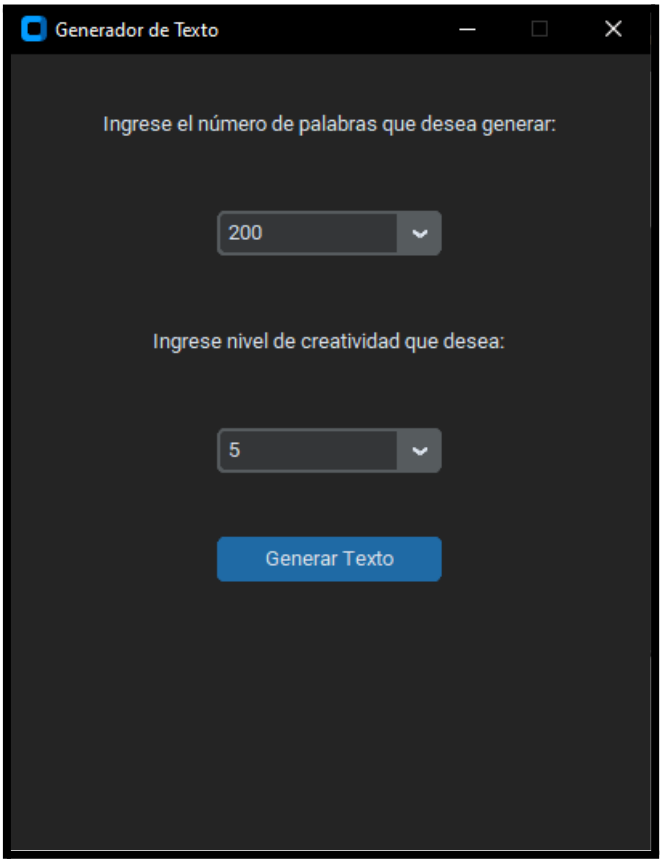
Imagenes de Funcionamiento de la Aplicación desarrollada:

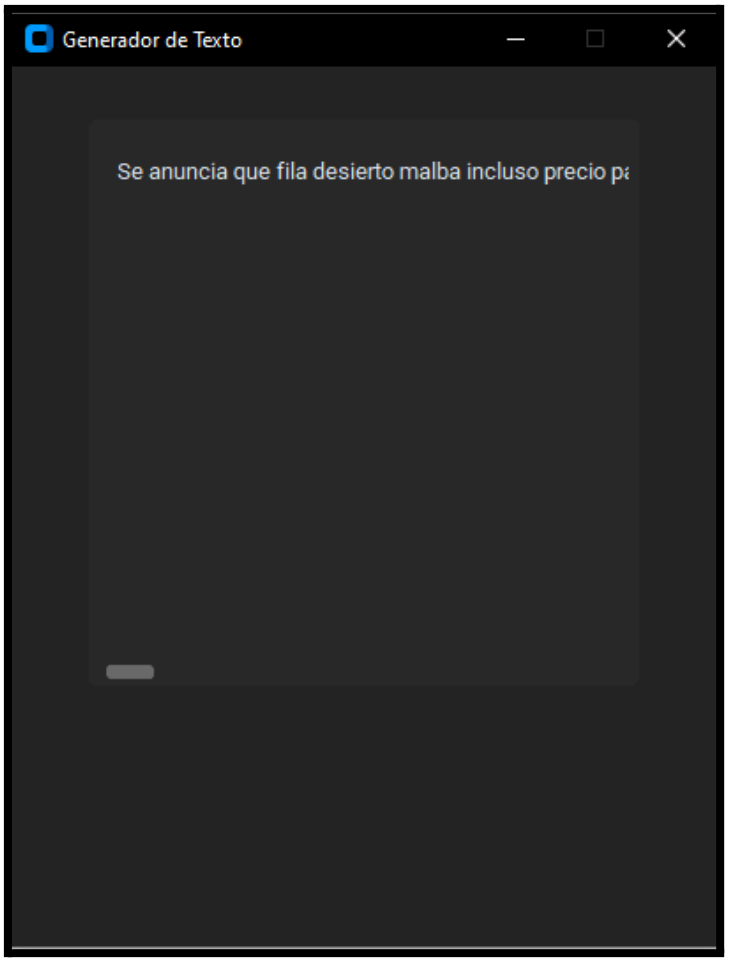


La aplicación consta de tres opciones la primera realiza el entrenamiento del módulo que extrae un archivo con información con la capacidad de entrenar la Red Neuronal con el fin de que adquiera conocimientos fundamentales sobre la economía actual.

La segunda opción permite generar texto según la cantidad de palabras que desee el usuario y la creatividad que quiera que se le aplique al texto que será generado.







Link de Github de desarrollo de la aplicación:

https://github.com/gabe-cloud/Generador-Texto-IA

Conclusión

El proyecto realizado permitió adquirir conocimientos sobre el funcionamiento de las redes neurales el funcionamiento de los algoritmos necesarios para realizar los módulos de entrenamiento necesarios para la Red Neuronal, de igual forma el proyecto permite y cumple con la iniciativa planteada en el Canvas Model teniendo la capacidad de generar texto formal mediante la entrada de un texto ingresado por el usuario.

Por ultimo, tambien tiene la posibilidad de tener un entrenamiento conste según la información que se le vaya ingresando a la Red Neuronal mediante el módulo de la primera opción antes mencionada.