Propuesta de proyecto: Sistema de redacción avanzado

Link modelo base: https://github.com/area-de-informatica/ICD_PA_Writter_IA.git

Nombres:

José Camilo Orozco Avilez - jorozcoavilez@correo.unicordoba.edu.co

Jaime cervantes Arrieta - jcervantesarrieta86@correo.unicordoba.edu.co

Johan Mercado Fernández - jmercadofernandez82@correo.unicordoba.edu.co

Deiker Alexánder Domínguez Sinitave - ddominguezsinitave09@corre.unicordoba.edu.co

Leonardo Triana Coronado - ltrianacoronado18@correo.unicordoba.edu.co

Descripción del Proyecto:

Este proyecto consiste en Desarrollar un Sistema de Redacción Avanzado basado en inteligencia artificial, que sea capaz de sugerir, expandir y completar oraciones a partir de entradas breves proporcionadas por el usuario (ya sea una palabra o una frase corta).

Este sistema se perfila como una herramienta útil para facilitar la redacción de textos más extensos, coherentes y bien estructurados, como ensayos, artículos académicos, informes y documentos en general.

Al aprovechar modelos de lenguaje pre entrenados y personalizables, el sistema permitirá:

- Autocompletar oraciones con sentido lógico y gramatical.
- Generar propuestas de redacción según el estilo del usuario.
- Sugerir mejoras en la formulación de ideas.

Tecnologías Utilizadas

Tecnología	Descripción
Python	Lenguaje de programación utilizado para el desarrollo de la aplicación.
Streamlit	Framework para crear aplicaciones web interactivas de forma sencilla.
OpenRouter	API que permite el acceso a modelos de lenguaje de inteligencia artificial.
mistralai/mistral-7b-instruct	Modelo de lenguaje utilizado para generar las reescrituras.
Re (regex)	Librería utilizada para identificar y separar opciones generadas por el modelo.

Estructura del Sistema

Entrada del Usuario

- **Texto a reescribir**: Campo de texto (máx. 2048 caracteres).
- **Tono deseado**: Selección entre Profesional, Formal, Amistoso, Informal o Diplomático.
- **Cantidad de variantes**: 1 o 2.
- **Modificar estructura**: Activar o desactivar reestructuración de ideas.

Procesamiento

- Se construye un prompt con las preferencias seleccionadas.
- Se envía el prompt y el texto original al modelo vía API.
- Se recibe la respuesta con 1 o 2 versiones escritas.

Salida

- Muestra el resultado en pantalla como texto enriquecido.
- Si hay 2 opciones, las separa claramente como **Opción 1** y **Opción 2**.

Código del aplicativo

```python
import streamlit as st
from openai import OpenAI
import re

# Clave de API
api_key = "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"
client = OpenAI(api_key=api_key, base_url="https://openrouter.ai/api/v1")

# Configuración de página
st.set_page_config(page_title="Reescribir texto", layout="centered")
st.markdown("## 📝 Reescribe el siguiente texto...")

# Entrada de texto
user_input = st.text_area("Texto:", height=150, max_chars=2048)
st.caption(f"{len(user_input)} / 2048 caracteres")

# Controles: Tono, Variantes, Estructura
col1, col2, col3 = st.columns([2, 2, 2])

with col1:
    tono_opciones = {
        "👔 Profesional": "profesional",
        "🏛️ Formal": "formal",
        "😊 Amistoso": "amistoso",
        "😎 Informal": "informal",
        "🤝 Diplomático": "diplomático"
    }
    tono_seleccionado = st.selectbox("Tono", list(tono_opciones.keys()), index=0)

with col2:
    cantidad = st.selectbox("Generar", ["1 variante", "2 variantes"])

with col3:
    editar_estructura = st.toggle("Editar estr  ura", value=False)
```

```
# Estilo del botón
st.markdown("""
<style>
div.stButton > button:first-child {
    background-color: #ff7900;
    color: white;
    font-weight: bold;
    border-radius: 5px;
    height: 3em;
}
</style>
""", unsafe_allow_html=True)

# Botón de acción
if st.button("✨ Reescribir párrafo"):
    if not user_input.strip():
        st.warning("Por favor, escribe algo.")
    else:
        with st.spinner("Reescribiendo..."):
            try:
                n_variantes = 2 if "2" in cantidad else 1
                estructura_msg = "Modifica también la estructura del texto." if editar_estructura

                if n_variantes == 2:
                    prompt = (
                        f"Reescribe el siguiente texto en español con un tono {tono_opciones[tono_
                        f"{estructura_msg} Devuelve exactamente dos versiones claramente separadas
                        "No des ninguna explicación adicional."
                    )
                else:
                    prompt = (
                        f"Reescribe el siguiente texto en español con un tono {tono_opciones[tono_
                        f"{estructura_msg} Devuelve solo una versión, sin usar etiquetas como 'Op
                        "No des ninguna explicación adicional."
                    )
```

```
        response = client.chat.completions.create(
            model="mistralai/mistral-7b-instruct:free",
            messages=[
                {"role": "system", "content": prompt},
                {"role": "user", "content": user_input}
            ],
            temperature=0.7
        )

        resultado = response.choices[0].message.content.strip()

        if "Opción 1:" in resultado and "Opción 2:" in resultado and n_variantes == 2:
            match = re.search(r"Opción 1:(.*?)Opción 2:(.*)", resultado, re.DOTALL)
            if match:
                opcion_1 = match.group(1).strip()
                opcion_2 = match.group(2).strip()
                st.markdown("### 📝 Opción 1")
                st.success(opcion_1)
                st.markdown("### 📝 Opción 2")
                st.success(opcion_2)
            else:
                st.markdown("### 📝 Resultado")
                st.success(resultado)
        else:
            st.markdown("### 📝 Versión mejorada")
            st.success(resultado)

    except Exception as e:
        st.error(f"Error: {str(e)}")
```

Consideraciones

El sistema desarrollado depende de una conexión estable a internet, ya que utiliza la API de OpenRouter para comunicarse con el modelo de lenguaje que realiza las reescrituras. Por lo tanto, si hay fallos de red o problemas con la plataforma externa, la funcionalidad del sistema puede verse limitada o interrumpida temporalmente.

Por razones de seguridad, se debe tener en cuenta que la clave de acceso a la API está escrita directamente en el código. Esto representa un riesgo potencial si se comparte el proyecto públicamente. Lo recomendable es almacenar estas credenciales en variables de entorno o en archivos de configuración no expuestos.

Finalmente, el sistema no almacena un historial de textos reescritos ni ofrece retroalimentación educativa. Su enfoque es práctico y directo, orientado a ofrecer resultados inmediatos. No obstante, en futuras versiones se podría considerar agregar funciones como guardar el historial del usuario o permitir análisis lingüísticos más detallados.

Mejoras futuras

Una posible mejora para versiones futuras del sistema es la incorporación de un módulo de autenticación de usuarios. Esto permitiría personalizar la experiencia, almacenar historiales de reescritura y generar estadísticas de uso. Además, se podría implementar una base de datos para guardar textos procesados y permitir que los usuarios revisen sus versiones anteriores o retomen trabajos inconclusos.

Otra mejora relevante sería la ampliación del soporte multilingüe, de modo que el sistema no sólo reescribe textos en español, sino también en inglés u otros idiomas. También se podría incluir la posibilidad de elegir entre diferentes modelos de IA según el nivel de profundidad o estilo de reescritura requerido, como GPT-4, Claude u otros disponibles en la plataforma Open Router.

Finalmente, se pueden agregar funciones complementarias como análisis de estilo, sugerencias de mejora por secciones o exportación de los textos en formatos como PDF o Word. Estas opciones aumentan la utilidad del sistema en contextos educativos, académicos y profesionales, haciendo del proyecto una herramienta aún más completa y versátil.