



TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico San Luis Potosí

SISTEMAS Y COMPUTACIÓN

Desarrollo de un asistente personal para mejora, corrección y/o redacción de textos sencillos desde un chat de Telegram.

Nombre del docente:
Alvarado Cano Lázaro

Alumnos:
Araceli Hernandez De La Cruz
Adal Manuel Maldonado Bautista
Jesus Gerardo Mendoza Jonguitud
Victoria Elizabeth Ojeda Medina
Andrea Ortiz Molina
Maria Guadalupe Rojas Gonzalez

Materia:
Inteligencia Artificial

Hora: 17:00 – 18:00

Bot de Telegram - Autómata de corrección de texto.

Desarrollo de un asistente personal para mejora, corrección y/o redacción de textos sencillos desde un chat de Telegram.

Resumen.

El presente documento habla de un proyecto que tiene como objetivo el desarrollo de un asistente personal basado en un chat de Telegram que ayude al usuario en ciertas tareas básicas, como mejora, corrección y/o redacción de textos sencillos, además de responder a algunas interacciones básicas. Esto se logrará a través de 2 formas, una metodología consistente en el análisis de la información contenida en los textos suministrados por el usuario o bien desde una imagen mandada al chat, la cual se procesará para identificar los errores de redacción u ortografía, al mismo tiempo que se recopilan datos para el entrenamiento de un modelo de lenguaje natural (utilizando chatgpt).

Introducción.

El proyecto trata de un “asistente personal” que ayude en ciertas tareas básicas de mejora, corrección y/o redacción de textos sencillos, así como su traducción tanto de manera escrita por el usuario o por medio de imágenes en un chat de Telegram (el cual se mencionara como el bot) además de responder a algunas interacciones básicas.

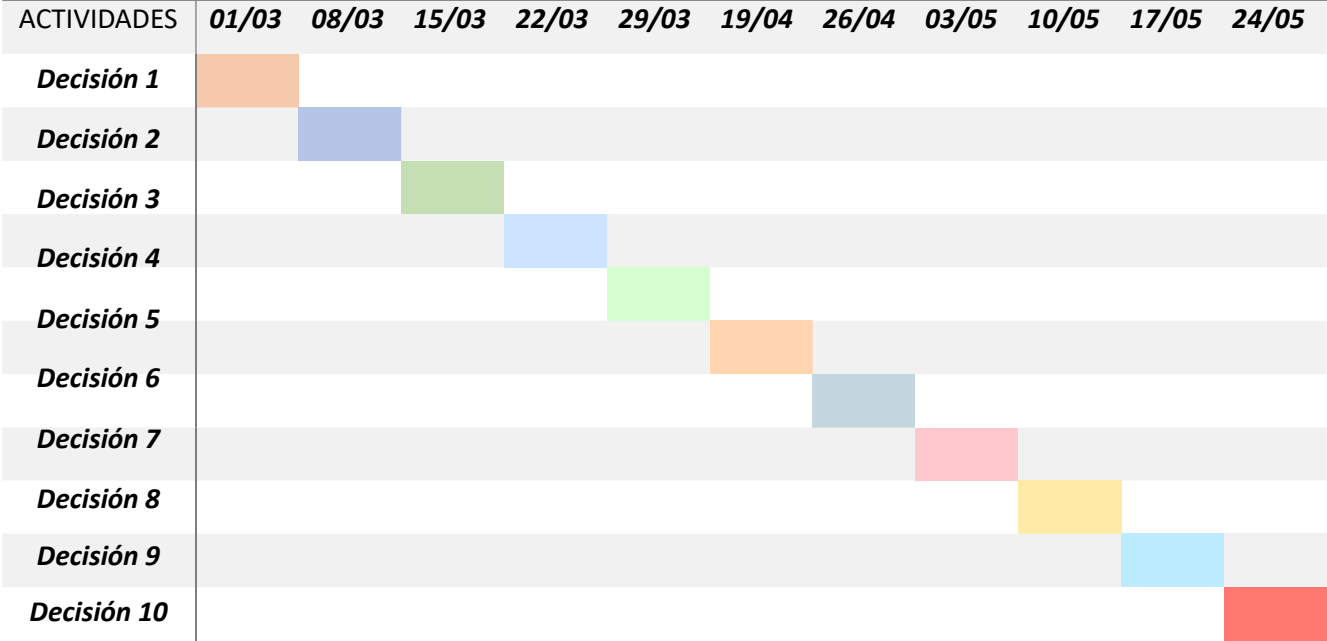
Objetivos.

El objetivo principal del presente proyecto es el diseño e implementación de un asistente personal basado en un chat de Telegram para mejorar, corregir y/o mejorar la redacción de textos sencillos. Esto se logrará a través del análisis y procesamiento de la información de los textos suministrados por el usuario y de la elaboración de un programa de software capaz de realizar las siguientes 10 decisiones autónomas:

Actividades	Descripción
DECISIÓN 1	<u>Contestar mensajes básicos como saludos y comandos sencillos de interacción</u>
DECISIÓN 2	<u>Recibir imágenes y guardarlas de manera temporal</u>
DECISIÓN 3	<u>Leer texto de una imagen</u>
DECISIÓN 4	<u>Mostar texto recopilado de una imagen</u>
DECISIÓN 5	<u>Mostar texto corregido de un texto mal escrito (faltas de ortografía), ya sea de una imagen o directamente algo escrito en el chat-bot</u>
DECISIÓN 6	<u>Mostar texto corregido de un texto mal escrito (redacción, palabras repetidas o mal escritas), ya sea de una imagen o directamente algo escrito en el chat-bot</u>
DECISIÓN 7	<u>Mostar texto parafraseado (redacción, palabras repetidas o mal escritas), ya sea de una imagen o directamente algo escrito en el chat-bot</u>
DECISIÓN 8	<u>Mostar texto traducido (redacción, palabras repetidas o mal escritas), ya sea de una imagen o directamente algo escrito en el chat-bot</u>
DECISIÓN 9	<u>Hacer de la actividad 2 a la 7 en conjunto con una muestra general al final dentro del chat</u>
DECISIÓN 10	<u>Como tal, no es una 10ma decisión si no que hace las tareas descritas de la actividad 2 a 7, con la diferencia de que debe estar escrito en el chat.</u>

El cronograma mostrado en esta tabla es sobre el tiempo estimado que se tiene para lograr cada objetivo de las actividades en la tabla anterior:

Grafica de Gantt - Fechas de entrega 2023 día/mes



Teoría, o consideraciones teóricas.

Actualmente existen variedad de formas de verificar texto en internet y otras herramientas que ayudan en la redacción de texto, a esto se le conoce como procesamiento de lenguaje natural (NLP), una tecnología de machine learning que brinda a las computadoras la capacidad de interpretar, manipular y comprender el lenguaje humano. Con esta tecnología la Inteligencia Artificial (IA) puede entender, procesar y aprender el lenguaje humano para crear un proceso comunicacional auténtico.

Pero ¿Qué es y cómo funciona el lenguaje natural?

Una base para comprender el lenguaje natural es el idioma que empleamos cuando hablamos. Al momento de tener el texto puede o no ser necesario cambiar parcialmente esta traducción.

Pero no solo se enfoca en ello, sino también en el sistema de corrección de palabras... de aquí sigue un léxico, una sintaxis y una estructura.

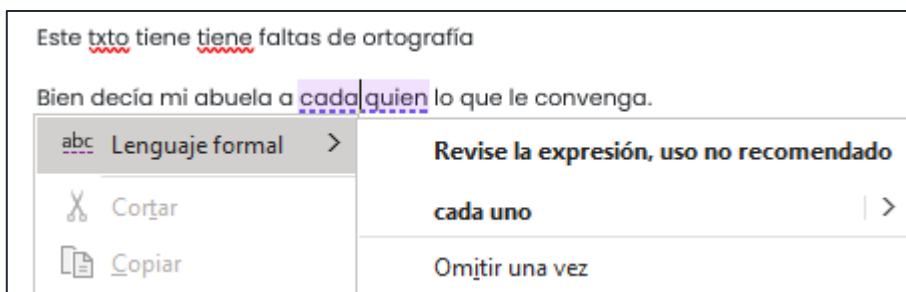
Ahora bien, si nos queremos referir a algún tipo de asistente los ejemplos que todos conocemos son las frases «Ok Google» o «Hey Siri» en los celulares a los cuales utilizamos cuando queremos resolver algo de express.

Los robots de *chat* funcionan de la misma manera: se integran con Slack, Microsoft Messenger y otros programas de *chat* donde leen el lenguaje que usamos, y se activan cuando escribimos una frase. Los asistentes de voz como Siri y Alexa también se activan cuando escuchan frases como "Hey, Alexa".

-Aplicaciones de corrección, ayuda en gramática, mejoramiento de redacción etc.

Empecemos por lo mas básico.

Usemos como ejemplo un documento de Word, mientras una persona esta escribiendo, el sistema de corrección de errores verifica el texto, su idioma y si tiene alguna falta de ortografía, falta de acentos, palabras repetidas o si está mal redactada alguna oración (por lo menos en sus versiones más recientes en 2023); si es el caso le marcará al usuario con una línea roja o azul según el tipo de error y le mostrará opciones de un diccionario para elegir como corregirlo o bien el documento lo hará por su cuenta.



Ahora, ¿Que pasa si lo que se quiere es mejorar un texto? Hoy en día existen cientos de páginas web que ayudan a parafrasear, resumir, reescribir o hasta hacer ensayos por su propia cuenta o de forma automática. Tan solo hay que introducir el texto y con un simple botón la tarea esta realizada en segundos.

Por mencionar alguno de estas aplicaciones existe “**Parafrasist**” (<https://parafrasist.com>), que hace justo lo que se mencionó en el párrafo anterior:

Ingresas un texto, seleccionas lo que quieres que haga.

PALABRAS: 50

CARÁCTERES: 295

Los robots de chat funcionan de la misma manera: se integran con Slack, Microsoft Messenger y otros programas de chat donde leen el lenguaje que usamos, y se activan cuando escribimos una frase. Los asistentes de voz como Siri y Alexa también se activan cuando escuchan frases como “Hey, Alexa”.

¿Qué quieres hacerle a tu texto?

☒ Parafrasear ☒ Resumir

Siguiente

Y despliega justo lo que se pidió:

← Procesar otro texto

El texto en rojo es el que se eliminó.

TEXTO ORIGINAL PARA COMPARAR:

Los robots de chat funcionan de la misma manera: se integran con Slack, Microsoft Messenger y otros programas de chat donde leen el lenguaje que usamos, y se activan cuando escribimos una frase. Los asistentes de voz como Siri y Alexa también se activan cuando escuchan frases como “Hey, Alexa”.

El texto cambiado aparece en color morado.

TEXTO RESUMIDO Y PARAFRASEADO

El robot de chat funciona de la misma manera. Están integrados con Slack, Microsoft Messenger y otros programas de chat, que están activos al leer y escribir frases.

Copiar Descargar pdf Descargar word

Palabras clave:

activan | alexa | chat | funcionan | programas | leen | lenguaje | escuchan | escribimos | frases |

Aun así, con la llegada de los modelos de inteligencia artificial esto se simplificó aún más. El más utilizado hasta ahora es ChatGPT. En este se introduce una instrucción pidiendo

que resuelva algo sencillo dentro del chat y por medio de una red neuronal avanzada procesara esa cadena de texto y mostrara lo que interpreto del mensaje

Este es un ejemplo utilizando el modelo de chat GPT3.5 en OpenAI Playground

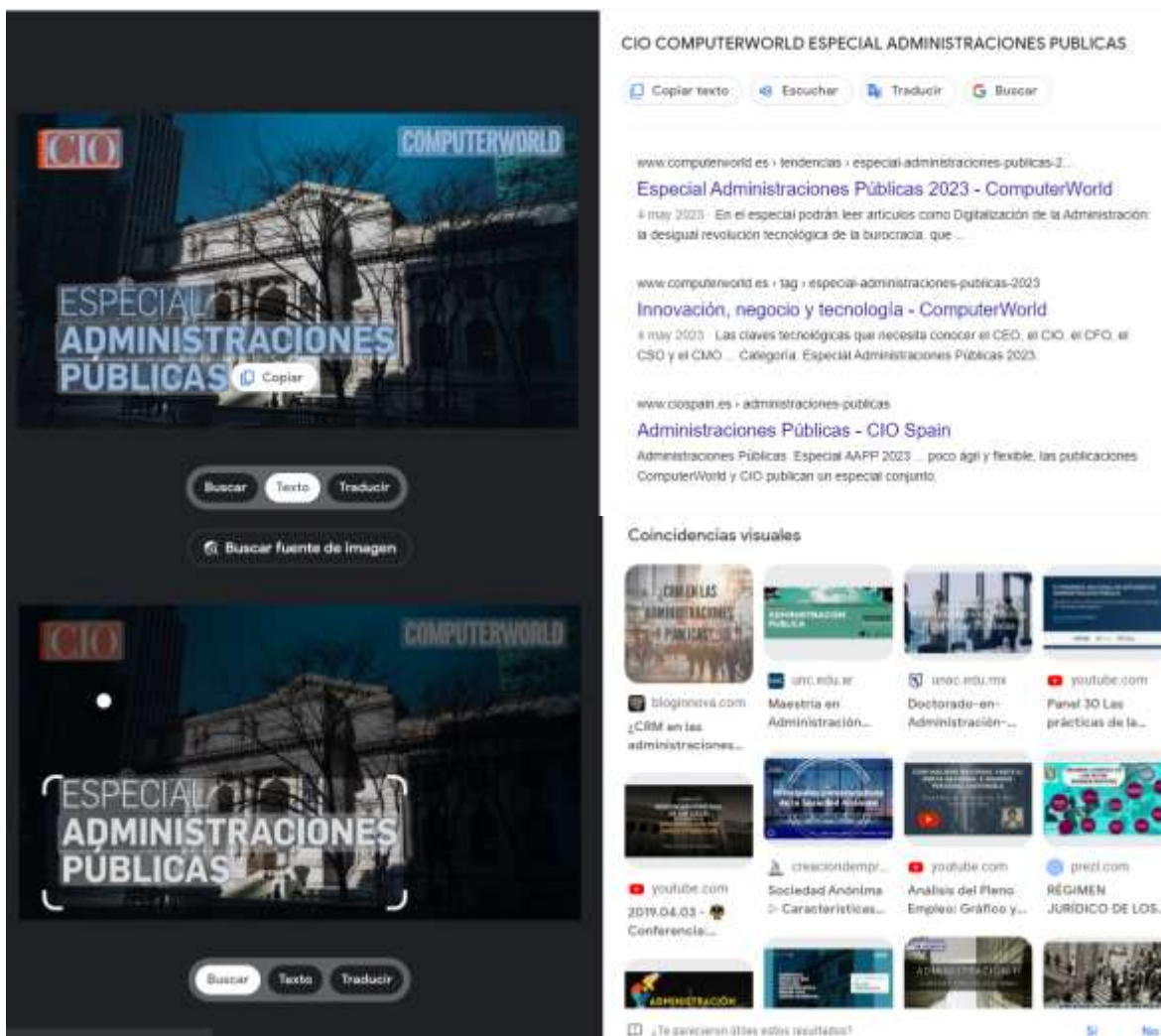
USER	Traduce este texto: Una base para comprender el lenguaje natural es el idioma que empleamos cuando hablamos.
ASSISTANT	A foundation for understanding natural language is the language we use when we speak.
USER	Gracias
ASSISTANT	¡De nada! Estoy aquí para ayudarte en lo que necesites.
USER	Dime que errores tiene este texto: El obgetivo princippal del presente proyecto es el diseño diseño e implementación de un asistente personal basado en un chat de Telegram para mejorar, corregir y/o mejorar la redacción de textos sencillos.
ASSISTANT	<p>El texto tiene los siguientes errores:</p> <ul style="list-style-type: none">- "obgetivo" debe ser "objetivo".- "princippal" debe ser "principal".- Hay un doble "diseño" en la primera parte de la oración, lo que indica una repetición innecesaria.- "testos" debe ser "textos". <p>La versión corregida del texto sería:</p> <p>El objetivo principal del presente proyecto es el diseño e implementación de un asistente personal basado en un chat de Telegram para mejorar, corregir y/o perfeccionar la redacción de textos sencillos.</p>

- Leer texto de imágenes

Existen diferentes formas de leer textos de imágenes, en internet hay diversas pagina donde se introduce una imagen y esta detecta con un scrip basado en OCR (Reconocimiento óptico de caracteres), un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos.

La herramienta más conocida de este tipo es Google Lens. La cual pide utilizar ya sea una imagen introducida desde el dispositivo en cuestión o por medio de una cámara, posteriormente hace un escaneo de la imagen (conocido como búsqueda inversa de imagen) y finalmente muestra las coincidencias o lo que encontró en la imagen.

También existen sitios parecidos como Yandex o Bing. Y otras páginas como WhatTheFont (verifica que fuente tiene el texto).



Método, metodología, o procedimiento experimental.

1. Análisis de datos y diseño

Requisitos

- Crear un bot de Telegram funcional

- El bot debe de cumplir cada objetivo planteado con una estructura organizada
- El programa del bot debe tener opciones a errores.
 - En caso de no entender el mensaje tener una contestación predeterminada.
 - Si falla el programa el bot no debe parar o ciclarse.

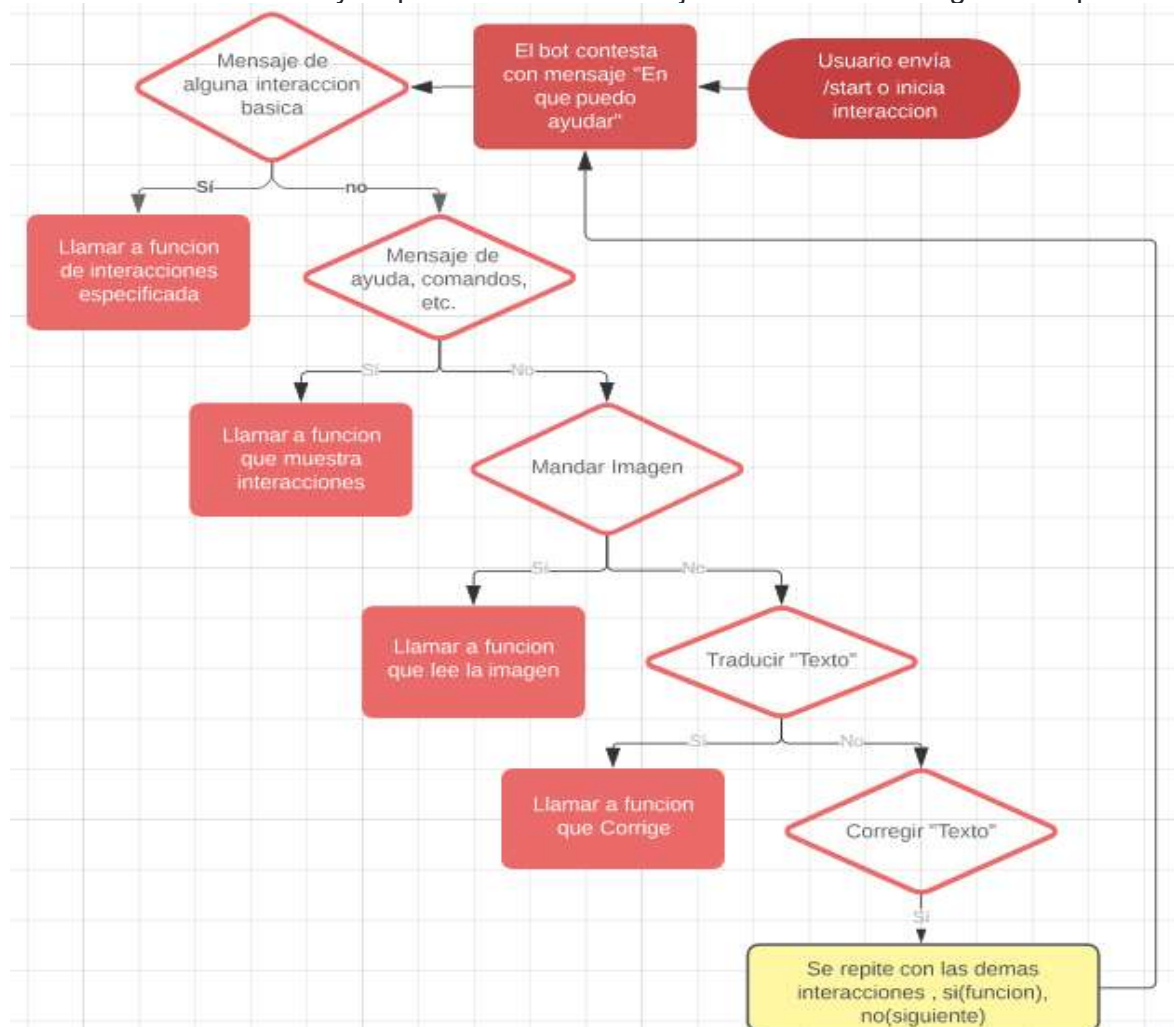
Especificaciones técnicas

Atributo	Descripción	
Recurso: Telegram 	Es una plataforma de mensajería y VOIP de origen ruso, desarrollada por los hermanos Nikolái y Pável Dúrov. La aplicación está enfocada en la mensajería instantánea, el envío de varios archivos y la comunicación en masa. 	
	Funciona tanto de manera Web como desde teléfono o aplicación de escritorio.	API: @BotFather Bot de telegram creado para generar cuentas de bots personalizables y administrables dentro de telegram y de forma externa por medio de una API-Key. 
Lenguaje: Python 	Lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo.	
	Librerías	
	pyTelegramBotAPI	Una implementación de Python simple pero extensible para Telegram Bot API , una interfaz basada en HTTP creada para desarrolladores interesados en crear bots para Telegram. Para aprender a crear y configurar un bot. https://pytba.readthedocs.io/en/latest/install.html#using-pip https://pypi.org/project/pyTelegramBotAPI/
	UUID (Universally Unique Identifier)	Se utiliza para generar id únicos de 128 bits que tienen muchos usos en bases de datos,

		<p>direccionamiento, generación de claves primarias, etc. Las claves generadas se utilizan para identificar objetos únicos de forma eficiente a través de una red sin tener que preocuparse por conflictos de nombres.</p> <p>https://pynative.com/python-uuid-module-to-generate-universally-unique-identifiers/</p>
	OS(Sistema Operativo)	<p>Proporciona una interfaz con el sistema operativo subyacente. Esta librería se utiliza para realizar tareas como obtener información del sistema, ejecutar programas externos, manipular archivos y directorios, etc.</p> <p>https://docs.python.org/es/3.10/library/os.html</p>
	OpenCV-Python	<p>Es una biblioteca de código abierto para el procesamiento de imágenes y visión por computadora. Está diseñado para proporcionar una interfaz sencilla para trabajar con imágenes digitales. Proporciona una amplia variedad de funciones para la manipulación de imágenes, como la transformación de imágenes, el análisis de imágenes, la detección de objetos y el reconocimiento facial. También proporciona herramientas para el desarrollo de aplicaciones de visión por computadora, como la detección de movimiento, la segmentación de imágenes y el reconocimiento de patrones.</p> <p>https://pypi.org/project/opencv-python/</p>
	Pytesseract	<p>Es una biblioteca de reconocimiento óptico de caracteres (OCR) para Python. Proporciona una interfaz a la biblioteca Tesseract OCR, que es una herramienta de código abierto ampliamente utilizada para la extracción de texto de imágenes. Esta biblioteca le permite procesar imágenes y extraer el texto que contienen. Pytesseract también incluye algunas utilidades adicionales para facilitar el procesamiento de imágenes y mejorar la precisión.</p> <p>Para que funcione correctamente se debe instalar el ejecutable de tesseract. https://github.com/UB-Mannheim/tesseract/wiki</p> <p>https://omes-va.com/como-instalar-tesseract-ocr-y-pytesseract-en-windows/</p>
	Googletrans2	<p>Implementado como la API de Google Translate. Esto utiliza la API Ajax de Google Translate para realizar llamadas a métodos tales como detectar y traducir. Ofrece la posibilidad de realizar traducciones en tiempo real, lo que significa que se puede traducir un texto mientras se escribe. Además, la librería googletrans también permite la traducción de fragmentos de texto, lo que significa que se pueden traducir frases y palabras específicas sin tener que traducir todo el texto completo.</p> <p>https://py-googletrans.readthedocs.io/en/latest/</p>
	OpenAI	<p>OpenAI es una organización sin fines de lucro dedicada a desarrollar tecnologías de Aprendizaje Automático y Inteligencia Artificial. Está financiada por inversores</p>

		<p>como Amazon, Microsoft, y Google. OpenAI Python ofrece herramientas para el desarrollo de aplicaciones de Inteligencia Artificial. Estas herramientas incluyen librerías de Machine Learning, herramientas de procesamiento de lenguaje natural, herramientas de optimización, entre otras. Estas herramientas permiten a los desarrolladores crear aplicaciones avanzadas de IA con facilidad. Estas aplicaciones pueden ser utilizadas para mejorar los procesos de toma de decisiones, optimizar los flujos de trabajo, mejorar la experiencia del usuario, entre otros.</p> <p>La biblioteca Python de OpenAI brinda un acceso conveniente a la API de OpenAI desde aplicaciones escritas en el lenguaje Python. Incluye un conjunto predefinido de clases para recursos de API que se inicializan dinámicamente a partir de respuestas de API, lo que lo hace compatible con una amplia gama de versiones de la API de OpenAI.</p> <p>https://platform.openai.com/docs/api-reference?lang=python</p>
		<p>Se utilizó OpenAI Playground como apoyo para codificación del programa https://platform.openai.com/playground</p> <p>Para conectar OpenAI con Python se utiliza además de la librería un API Key gratuito de la plataforma.</p> <p>https://platform.openai.com/account/api-keys</p> <p>Solo permite ciertos token o peticiones limitadas para mantener activo durante más tiempo es necesario pagar.</p>
VS Code		<p>Es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.</p>

El modelo del bot y opciones de mensaje llevarían el siguiente proceso:

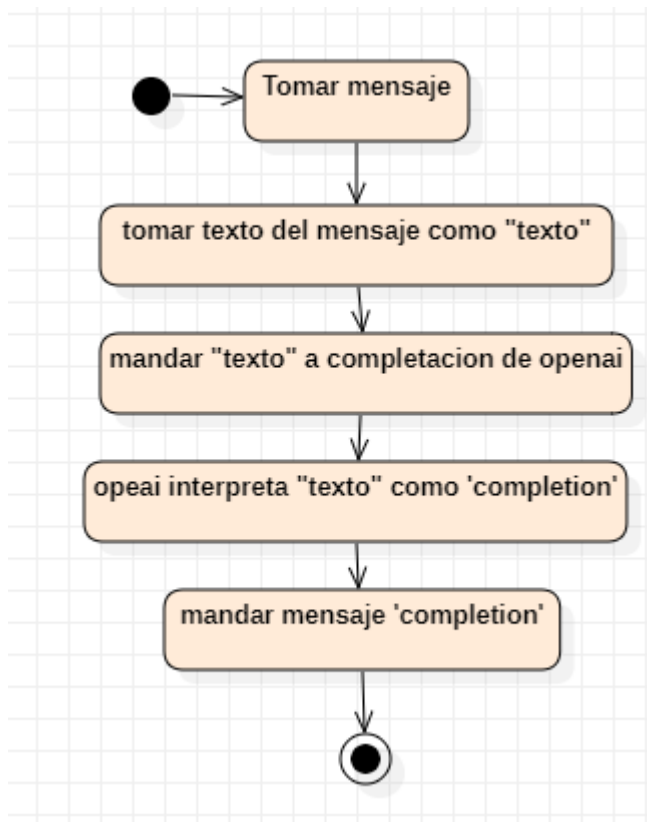


Al final del diagrama se hace referencia a “las demás interacciones”, se refiere a las interacciones especificadas en el objetivo. Más adelante se hablará a detalle de ellas.

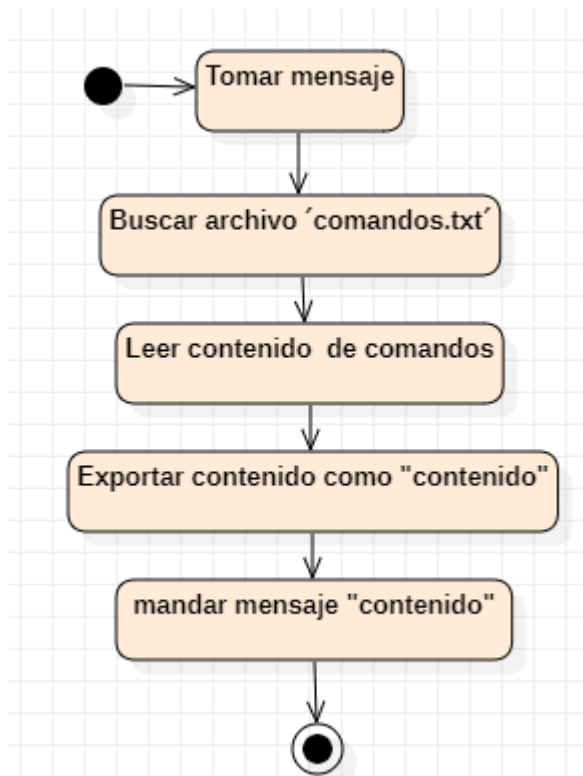
Funciones:

- Funciones de interacciones básicas: Solo manda saludos o pregunta si se necesita ayuda, si se pide una imagen manda una imagen que tiene por defecto del bot, si se pide un audio el bot manda un audio que tiene por defecto el bot.

Interacción con chat GPT para que mande saludos personalizados cuando se diga “hola” o algo por el estilo en el chat de telegram.



- Función que muestra interacciones: Manda en el chat la lista de interacciones posibles

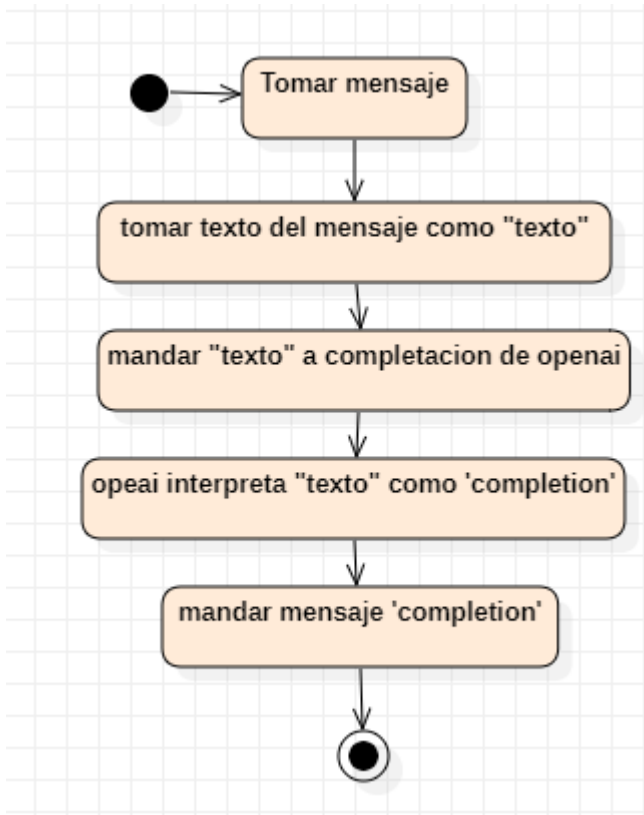


Interacción de comandos

- Función de la imagen: Cuando se mande una imagen al chat el bot responderá con: el texto leído de la imagen, su corrección, los errores, parafraseado, redacción mejorada y traducción de cada parte.

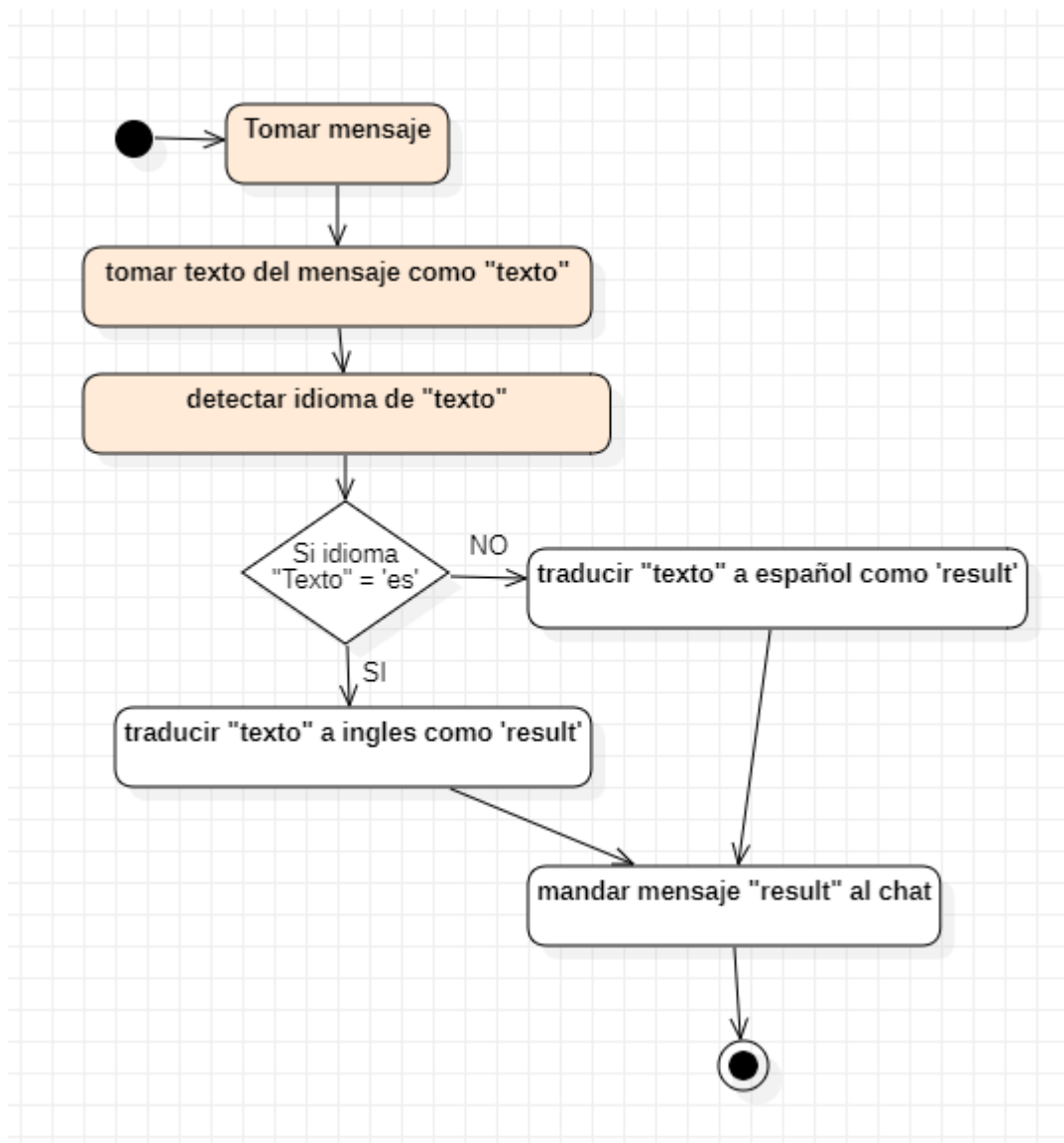
- Funciones individuales: Si se pide traducir, modificar, mejorar o verificar un texto desde el chat, este lo mostrara según la instrucción especificada.

- Corregir
- Modificar
- Parafrasear, Redacción y Errores.



Utilizando de la misma función de la primera tarea, pero solo cuando el texto comienza con alguna de las premisas de las funciones individuales

- Traducir



3. Codificación

Inicio: Se encuentra la librería principal, el api-key del bot, la conexión principal entre telegram y el programa y una variable con tokens para salida de comandos.

```
import telebot # libreria para interactuar en telegram - pip install pyTelegramBotAPI
# activar uso de scripts Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
token = '6161350290:AAHj7gRPh8tU7Cbei3Yclk0Nin9t6YHSAA4'
bot = telebot.TeleBot(token)

interacciones_alias = ['Ayuda', 'Comandos', 'Interacciones']
```

Llamada a cada función según lo que se mande al chat

```
# ----- Contesta mensajes -----
@bot.message_handler(content_types=['text', 'photo'])
def audio(message):
    #prueba audio
    if message.text == 'Audio':
        audio = open('audio.mp3', 'rb')
        bot.send_audio(message.chat.id, audio)
    #manda imagen
    elif message.text == 'Imagen':
        imagen = open('imagen.jpg', 'rb')
        bot.send_photo(message.chat.id, imagen)
    #interaccion con foto
    elif message.content_type == 'photo':
        try:
            mostrar = recibir_imagenes(message)
        except:
            bot.send_message(message.chat.id, 'No se ha podido reconocer el texto de la imagen')
    #muestra las interacciones
    elif message.text in interacciones_alias:
        mostrar = interacciones(message)
    elif message.text.startswith('Traducir'):
        mostrar = traducir(message)
    elif message.text.startswith('Corrige') or message.text.startswith('Mejora')
        ) or message.text.startswith('Parafrasea') or message.text.startswith('Dame')
        ) or message.text.startswith('Verifica'):
        mostrar = chatGPT(message)
    elif message.text.startswith('hola') or message.text.startswith('Hola')
        ) or message.text.startswith('Que tal') or message.text.startswith('que tal')
        ) or message.text.startswith('buen dia'):
        mostrar = chatGPT(message)
    else:
        bot.send_message(message.chat.id, "En qué te puedo ayudar?")
```

Función para comandos

```
# ----- Mensaje de ayuda [muestra las interacciones] -----
def interacciones(message):
    # abrir el archivo
    with open('comandos.txt', encoding='utf-8') as archivo:
        contenido = archivo.read()
        print(contenido)
        bot.send_message(message.chat.id, contenido)
```

Traducciones


```
def traducir(message):
    #Obtener el texto a traducir
    texto_a_traducir = message.text.split('Traducir ', 1)[1]
    print(texto_a_traducir)
    #Aquí iría el código para traducir
    traduccion = 'Aquí va la traducción:'
    # import googletrans
    from googletrans import Translator # pip install googletrans-py

    traductor = Translator()

    # Detectar el idioma del texto
    idioma = traductor.detect(texto_a_traducir).lang
    # Si el idioma es español, traducir al inglés
    if idioma == 'es':
        resultado = traductor.translate(texto_a_traducir, dest='en')
    # Si el idioma es distinto al español, traducir al español
    else:
        resultado = traductor.translate(texto_a_traducir, dest='es')

    #Enviar la traducción
    bot.reply_to(message, traduccion)
    bot.send_message(message.chat.id, resultado.text)
```

Uso de GPT3

Text-davinci-003 es el modelo que se utiliza para completar o responder mensajes

```
def chatGPT(message):
    import openai # pip install openai

    openai.api_key = "sk-eCuJYLJpBvDSU7SfBKHqT3B1bkFJPpIJmiso9L0r3LPkFQ0e"

    texto = message.text

    completion = openai.Completion.create(engine="text-davinci-003",
                                           prompt=texto,
                                           max_tokens=3000)
    print(completion.choices[0].text)
    bot.send_message(message.chat.id, completion.choices[0].text)
```

Uso de todas las decisiones en conjunto

Recibe las imágenes y crea un archivo temporal en lo que toma el texto detectado

```
# ----- Mensaje de imagen [interacción con foto] -----
def recibir_imagenes(message):

    import uuid # Libreria para nombre aleatorio
    import os   # Libreria para archivo temporal

    # Guardar la imagen
    file_id = message.photo[-1].file_id
    file_info = bot.get_file(file_id)
    downloaded_file = bot.download_file(file_info.file_path)
    # src = 'imagen.png' pero con nombre agregado
    src = 'imagen_{}.png'.format(uuid.uuid4())
    with open(src, 'wb') as new_file: # descarga la imagen para su uso
        new_file.write(downloaded_file)

    import cv2 # Libreria para interactuar con imagenes - pip install opencv-contrib-python
    import pytesseract # Libreria para Leer texto de imagen - pip install pytesseract

    pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

    image = cv2.imread(src) #lee la imagen
    text = pytesseract.image_to_string(image)
    # Eliminar la imagen después de un tiempo determinado
    os.remove(src)
```

Se despliega primera interpretación según idioma.

```
# Enviar mensaje al usuario

bot.send_message(message.chat.id, text)

from googletrans import Translator # pip install googletrans-py
traductor = Translator()
# Detectar el idioma del texto
idioma = traductor.detect(text).lang
# Si el idioma es español, traducir al inglés
if idioma == 'es':
    resultado = text
# Si el idioma es distinto al español, traducir al español
else:
    traduccion = traductor.translate(text, dest='es')
    resultado = traduccion.text
    bot.send_message(message.chat.id, resultado)
#Enviar la traducción
```

Utilizando el motor de OpenAI se hacen los pasos de 2 al 7 sin incluir traducción

```

import openai # pip install openai

openai.api_key = "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"
cadena_unida = "corrige el siguiente texto: " + " " + resultado
correct = openai.Completion.create(engine="text-davinci-003",
                                   prompt= cadena_unida,
                                   max_tokens=3000)
corregido = correct.choices[0].text

cadena_unida = "Dime los errores que tenga este texto: " + " " + resultado
errrs = openai.Completion.create(engine="text-davinci-003",
                                  prompt= cadena_unida,
                                  max_tokens=3000)
errores = errrs.choices[0].text

cadena_unida2 = "mejora la redaccion del sig texto: " + " " + corregido
redact = openai.Completion.create(engine="text-davinci-003",
                                   prompt= cadena_unida2,
                                   max_tokens=3000)
redaccion = redact.choices[0].text

cadena_unida2 = "parafrasea" + " " + corregido
parafr = openai.Completion.create(engine="text-davinci-003",
                                   prompt= cadena_unida2,
                                   max_tokens=3000)
parafrasear = parafr.choices[0].text

```

Se genera la traducción de las cadenas anteriores

```

from googletrans import Translator # pip install googletrans-py

traductor = Translator()

# Detectar el idioma del texto
idioma = traductor.detect(corregido).lang
idioma = traductor.detect(redaccion).lang
idioma = traductor.detect(parafrasear).lang
# Si el idioma es español, traducir al inglés
if idioma == 'es':
    trad = traductor.translate(corregido, dest='en')
    trad0 = traductor.translate(redaccion, dest='en')
    trad1 = traductor.translate(parafrasear, dest='en')
# Si el idioma es distinto al español, traducir al español
else:
    trad = traductor.translate(corregido, dest='es')
    trad0 = traductor.translate(redaccion, dest='es')
    trad1 = traductor.translate(parafrasear, dest='es')
#Enviar la traducción

```

Despliegue en cadena de los resultados obtenidos

```

bot.send_message(message.chat.id, "Corregido")
bot.send_message(message.chat.id, corregido)

bot.send_message(message.chat.id, "Errores")
bot.send_message(message.chat.id, errores)

bot.send_message(message.chat.id, "Traducción")
bot.send_message(message.chat.id, trad.text)

bot.send_message(message.chat.id, "Redacción")
bot.send_message(message.chat.id, redaccion)

bot.send_message(message.chat.id, "Traducción")
bot.send_message(message.chat.id, trad0.text)

bot.send_message(message.chat.id, "Parafraseado")
bot.send_message(message.chat.id, parafrasear)

bot.send_message(message.chat.id, "Traducción")
bot.send_message(message.chat.id, trad1.text)

```

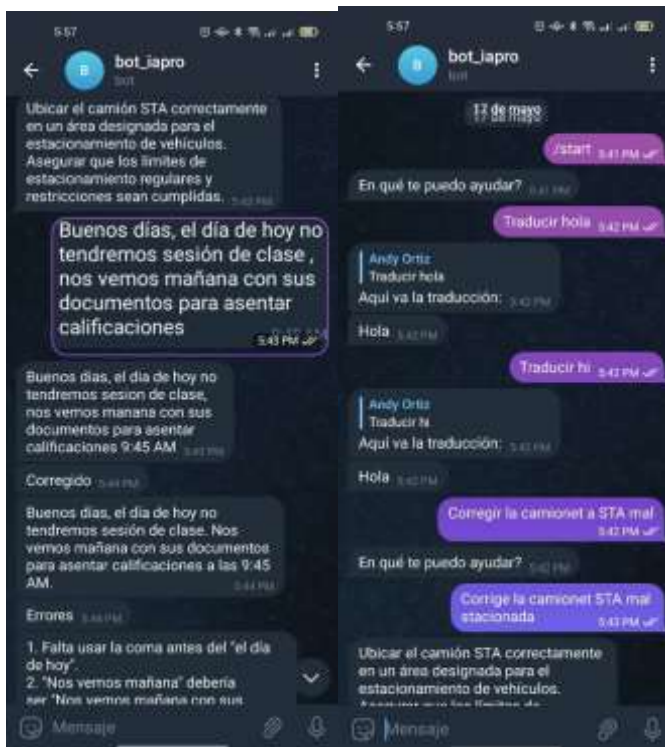
Cierre del programa

```
# -----
bot.polling()
while True: # Mantiene activo el bot
    pass
```

Resultados.

El programa de software diseñado es capaz de realizar las 10 decisiones autónomas mencionadas anteriormente en el documento, lo único que se necesita es escribir la instrucción en el chat y se despliega lo que se esta pidiendo.

Existieron problemáticas durante el proceso de creación del bot pero finalmente se logró un trabajo aceptable.



Discusión o interpretación.

Los resultados obtenidos demuestran que es posible diseñar e implementar un asistente personal basado en un chat de Telegram para mejorar, corregir y/o mejorar la redacción de textos sencillos. El uso de técnicas de procesamiento de lenguaje natural, el análisis sintáctico y semántico de los textos y la inteligencia artificial permiten al programa de software realizar las decisiones autónomas necesarias para mejorar la redacción y mostrar un buen resultado.

Conclusión o conclusiones.

El diseño e implementación de un asistente personal basado en un chat de Telegram para mejorar, corregir y/o mejorar la redacción de textos sencillos es una tarea factible. El programa de software diseñado es capaz de realizar las 10 decisiones autónomas

necesarias para mejorar la redacción, tales como la revisión de la ortografía, la corrección de errores de sintaxis, la adaptación del lenguaje a una audiencia específica, la verificación de la claridad y coherencia de los textos, entre otros.

Es algo que se puede hacer de otras maneras, como una aplicación móvil o una pagina web, pero como justificación se puede decir que tiene la facilidad de estar en cualquier momento cuando se inicia una conversación con el bot, además de la posibilidad de meterlo a grupos y que no es necesario hacer instalaciones externas.

Recomendaciones.

Se recomienda que el software diseñado sea probado en un entorno real para validar su eficiencia y robustez. Además, se sugiere que se realicen mejoras continuas en el programa para una mejor adaptación a los usuarios.

Bibliografía

<https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf>

[https://elipse.ai/blog/definicion-de-la-tecnologia-lenguaje-natural/#:~:text=El%20lenguaje%20natural%20se%20entiende,%2C%20chino%2C%20entre%20otros\).](https://elipse.ai/blog/definicion-de-la-tecnologia-lenguaje-natural/#:~:text=El%20lenguaje%20natural%20se%20entiende,%2C%20chino%2C%20entre%20otros).)

<https://www.google.com/search?client=opera-gx&q=tecnología+de+lenguaje+natural&sourceid=opera&ie=UTF-8&oe=UTF-8>