Automatización de Consultas en la Escuela de Ingeniería Informática: Implementación de un Chatbot para Docentes y Estudiantes

1st Phuyo Huaman Edson Leonid
Ingenieria Informatica y de sistemas
Universidad Nacional de San Antonio Abad del Cusco
Cusco,Peru
183078@unsaac.edu.pe

2nd Alex Sandi Mamani
Ingenieria Informatica y de sistemas
Universidad Nacional de San Antonio Abad del Cusco
Cusco,Peru
131050@unsaac.edu.pe

3rd George Alexander Zapana Flores
Ingenieria Informatica y de sistemas
Universidad Nacional de San Antonio Abad del Cusco
Cusco,Peru
192975@unsaac.edu.pe

4th Cristian Andy Cabrera Mejia
Ingenieria Informatica y de sistemas
Universidad Nacional de San Antonio Abad del Cusco
Cusco,Peru
201230@unsaac.edu.pe

Resumen

Este artículo detalla el proceso de análisis y diseño de un chatbot efectivo. Se exploran pasos clave, como la selección cuidadosa de herramientas de procesamiento de lenguaje natural, el preprocesamiento de texto para optimizar la calidad de las respuestas, y la arquitectura basada en similitud coseno para identificar respuestas relevantes. Tras pruebas y ajustes, el chatbot demuestra éxito en interacciones, ofreciendo respuestas coherentes y precisas. Este enfoque demuestra cómo las herramientas y técnicas de NLP pueden dar vida a un chatbot funcional y eficaz en una variedad de aplicaciones.

***Index Terms*—Palabras clave: chatbot, NLP, repositorio, corpus, Spacy, NLTK, json, automatizacion, procesamiento de lenguaje natural, análisis, diseño, selección de herramientas, preprocesamiento de texto, similitud coseno, relevancia, interacciones, calidad de respuestas, técnicas de NLP, funcional, eficiente, aplicaciones, tokenizacion, similitud coseno, optimizacion, TF-IDF, TfidfVectorizer.**

***Abstract*—This article outlines the process of analyzing and designing an effective chatbot. Key steps are explored, such as the careful selection of natural language processing tools, text preprocessing to enhance response quality, and an architecture based on cosine similarity to identify relevant answers. Following testing and adjustments, the chatbot proves successful in interactions, providing coherent and accurate responses. This approach showcases how NLP tools and techniques can breathe life into a functional and efficient chatbot across various applications.**

chatbot, NLP, repository, corpus, Spacy, NLTK, json, automation, natural language processing, analysis, design, tool selection, text preprocessing, cosine similarity, relevance, interactions, response quality, NLP techniques, functional, efficient, applications, tokenization, cosine similarity, optimization, TF-IDF, TfidfVectorizer.

I. Introduccion

Desde el nacimiento de ELIZA en los años 60 (El terapeuta virtual- joseph Weizenbaum - creado en MIT), ALICE en los años 90 (La Inteligencia en Auge - Richard Wallace) hasta la sofisticación de chatbots (2010) Siri (Apple), Alexa (Amazon) y Google Assistant (Google) en la última década, la historia de los chatbots revela un progreso constante en la comprensión y replicación del lenguaje humano. Con el lanzamiento de GPT-3 (Generative Pre-trained Transformer 3) por OpenAI, los chatbots han alcanzado nuevas alturas. GPT-3 es capaz de sostener conversaciones mucho más complejas y contextuales. Puede generar respuestas coherentes y realistas, incluso en temas especializados. Esta nueva generación de chatbots muestra un potencial emocionante para transformar la interacción entre humanos y máquinas. [1]

En el panorama de la tecnología actual, los chatbots se han establecido como programas de inteligencia artificial diseñados para llevar a cabo interacciones conversacionales, emulando conversaciones humanas. La esencia fundamental de un chatbot radica en su capacidad para responder preguntas, brindar información y ejecutar tareas específicas, todo ello a través de una interfaz de chat. Estos agentes virtuales se apoyan en algoritmos de procesamiento del lenguaje natural (NLP) para comprender y procesar los mensajes de los usuarios, permitiéndoles generar respuestas coherentes y contextualmente relevantes.

Una vision hacia el futuro, imaginemos un mundo donde los chatbots, impulsados por inteligencia artificial aún más avanzada, no solo comprendan nuestras palabras, sino también

nuestras emociones y necesidades más profundas. Con la capacidad de ofrecer apoyo emocional y consejos personalizados, podrían convertirse en compañeros virtuales que ayuden a mitigar problemas de salud mental y brindar orientación en momentos de crisis.

II. Analisis y Diseño de la solucion

Los pasos clave involucrados en el análisis y diseño de la solución se presentan a continuación:

A. Selección de Herramientas

Para la implementación del chatbot, se realizó una cuidadosa selección de herramientas de procesamiento de lenguaje natural (NLP). Se optó por utilizar librerías ampliamente reconocidas como Natural Language Toolkit (NLTK), spaCy y scikit-learn. NLTK se utilizó para tareas como tokenización y eliminación de palabras detenidas, mientras que spaCy proporcionó un modelo de procesamiento de lenguaje natural en español para análisis semántico. Además, se empleó el framework scikit-learn para la creación de vectores TF-IDF y el cálculo de similitud coseno.

B. Preprocesamiento de Texto

El preprocesamiento de datos es una etapa crítica en el análisis de lenguaje natural. Se implementó un proceso de preprocesamiento que incluyó la eliminación de acentos y caracteres especiales utilizando la librería ”unidecode”. Además, se aplicó tokenización a través de NLTK para dividir el texto en unidades significativas, y se eliminaron las palabras detenidas (stop words) para reducir el ruido y mejorar la calidad de las representaciones de texto utilizadas en el procesamiento posterior.

C. Diseño de la Arquitectura del Chatbot

El diseño de la arquitectura del chatbot se basó en la generación de vectores TF-IDF y la comparación de similitud coseno. Los patrones de respuestas del chatbot se transformaron en vectores TF-IDF utilizando el vectorizador TfidfVectorizer de scikit-learn. Esto permitió representar las respuestas en un formato numérico que reflejara la importancia de las palabras en el contexto del corpus. La similitud coseno se utilizó para comparar la entrada del usuario con los patrones de respuestas y determinar la relevancia.

D. Pruebas del Chatbot

Se realizaron interacciones de prueba con el chatbot utilizando una variedad de preguntas y se evaluaron las respuestas generadas. Las puntuaciones de similitud coseno se utilizaron como métrica para medir la relevancia de las respuestas generadas en función de la entrada del usuario.

E. Optimización y Ajustes del Chatbot

Durante el proceso de análisis y diseño, se realizaron ajustes y optimizaciones para mejorar el rendimiento y la precisión del chatbot. Se ajustaron los hiperparámetros del vectorizador TF-IDF y se experimentó con diferentes valores de umbral para determinar la similitud aceptable entre la entrada del usuario y los patrones de respuestas. Estos ajustes fueron fundamentales para garantizar respuestas relevantes y coherentes.

F. Conclusiones del Análisis y Diseño

El análisis y diseño detallados permitieron una implementación exitosa del chatbot en Google Colab. La elección cuidadosa de herramientas, el preprocesamiento de texto eficiente y la arquitectura basada en similitud coseno fueron factores clave para lograr un chatbot funcional. Sin embargo, se reconoce la necesidad de continuar mejorando el chatbot para comprender preguntas complejas y poco comunes, lo que abre puertas a futuras iteraciones y refinamientos.

III. Aplicacion de la Solucion

La implementación del chatbot se realizó utilizando el lenguaje de programación Python en la plataforma Google Colab. A continuación, se describe el proceso de aplicación de la solución, desde la preparación de datos hasta la interacción del usuario con el chatbot implementado.

A. Carga de Datos y Preprocesamiento

El chatbot se diseñó para interactuar con los usuarios utilizando patrones de preguntas y respuestas almacenados en un archivo JSON. Los datos de entrada se cargaron desde este archivo y se realizaron pasos de preprocesamiento para limpiar el texto y convertirlo en un formato adecuado para el procesamiento posterior. El preprocesamiento incluyó la eliminación de acentos, la tokenización y la eliminación de palabras detenidas (stop words) para reducir el ruido en los datos.

```
with open('corpus.json', 'r', encoding='utf-8'
    ) as archivo:
    corpus = json.load(archivo)

palabras_detenidas = set(stopwords.words('
    spanish'))

def preprocesar_texto(texto):
    doc = nlp(texto)
    tokens_filtrados = [
        token.lemma_ for token in doc
        if not token.is_punct
        and not token.is_space
        and not token.is_stop
        and not token.is_digit
        and not token.like_num
    ]
    return ' '.join(tokens_filtrados)
```

B. Creación de Vectores TF-IDF

Se implementó un vectorizador TF-IDF (Term Frequency-Inverse Document Frequency) utilizando la librería TfidfVectorizer de scikit-learn. Este vectorizador transforma los patrones de respuestas en una representación numérica basada en la importancia de las palabras en el contexto del corpus. El preprocesamiento de texto previamente realizado ayudó a mejorar la calidad de los vectores TF-IDF generados.

```
def crear_vectores_tfidf(corpus):

    todas_las_respuestas = []

    for seccion in corpus.values():
        todas_las_respuestas.extend(seccion['
            patrones'])

    vectorizador = TfidfVectorizer(
        preprocessor=preprocesar_texto)

    matriz_tfidf = vectorizador.fit_transform(
        todas_las_respuestas)
e
    return vectorizador, matriz_tfidf
```

C. Evaluación de Similitud Coseno

Para determinar la similitud entre la entrada del usuario y los patrones de respuestas, se utilizó la métrica de similitud coseno. Esta métrica mide la similitud entre dos vectores en un espacio n-dimensional y permite identificar las respuestas más relevantes para la pregunta del usuario. Los vectores TF-IDF generados se utilizaron para calcular las puntuaciones de similitud coseno.

```
def buscar_patrones_similares(entrada_usuario,
     vectorizador, matriz_tfidf, corpus):

    entrada_usuario_preprocesada =
        preprocesar_texto(entrada_usuario)

    vector_entrada = vectorizador.transform([
        entrada_usuario_preprocesada])

    puntuaciones_similitud = cosine_similarity
        (vector_entrada, matriz_tfidf)

    indices_mas_similares =
        puntuaciones_similitud.argsort()[0,
        ::-1]

    respuestas = []

    for indice in indices_mas_similares:
        similitud = puntuaciones_similitud[0,
            indice]
        if similitud >= 0.2:
            patrones = []

            for seccion in corpus.values():
                patrones.extend(seccion['
                    patrones'])

            patron_similar = patrones[indice].
                lower()

            patron_similar = unidecode(
                patron_similar)

            for clave, valor in corpus.items()
                :
                for patron in valor['patrones'
                    ]:
                    if unidecode(patron.lower
                        ()) == patron_similar:
                        respuestas = valor['
                            respuestas']
                        break
            break

    return respuestas
```

D. Interacción con el Usuario:

La aplicación del chatbot se logra mediante una interfaz de consola donde los usuarios pueden escribir preguntas. El chatbot analiza la entrada del usuario, busca patrones similares en el corpus y selecciona las respuestas más relevantes en función de la similitud coseno calculada. Además, el chatbot reconoce saludos y puede finalizarse cuando el usuario lo indique.

```
def chat():

    print(" BIENVENIDO  AL CHATBOT DE LA
        ESCUELA DE INGENIER A INFORM TICA! *
        ")
    print(" YACHAYBOT !")

    print(" Puedes escribir 'salir' en
        cualquier momento para terminar.\n")

    vectorizador, matriz_tfidf =
        crear_vectores_tfidf(corpus)

    while True:
        entrada_usuario = input("+ T  : ")
        print()
        r = saludos(entrada_usuario.lower())

        if entrada_usuario.lower() in ['salir'
            , 'adi s', 'chao', 'cerrar','
            adios','chau', 'hasta pronto','
            gracias', 'salir']:
            print(">> YachayBot:  Hasta  luego
                !")
            break

        respuestas = buscar_patrones_similares
            (entrada_usuario, vectorizador,
            matriz_tfidf, corpus)
        if (r != None):
          print(">> YachayBot: " + r)
          print("")
        else:
            if not respuestas:
              print(">> YachayBot: Lo siento,
                  no entiendo tu pregunta.\n")
            else:
                for respuesta in respuestas:
                    print(">> YachayBot:",
                        respuesta)
                print("")
```

IV. Interpretacion de Resultados

A. Evaluación del Rendimiento

En el contexto de este estudio, se ha desarrollado y evaluado el chatbot YachayBot. Los resultados obtenidos durante la

evaluación demuestran el éxito del chatbot en situaciones donde las consultas constan de hasta 10 palabras y contienen al menos una palabra clave presente en el corpus de patrones. En tales circunstancias, YachayBot exhibe un rendimiento cercano a la optimización, logrando responder de manera eficaz a las preguntas planteadas.

B. Validación mediante Pruebas

Para corroborar la eficacia del chatbot, se llevaron a cabo múltiples pruebas exhaustivas utilizando el corpus de patrones implementado. Estas pruebas involucraron una variedad de consultas y situaciones, y se realizaron comparaciones detalladas entre las respuestas proporcionadas por YachayBot y las respuestas esperadas basadas en el corpus. A continuación, se presentan los resultados de estas pruebas, respaldando así la coherencia y precisión del chatbot en su capacidad para interactuar de manera significativa con los usuarios.

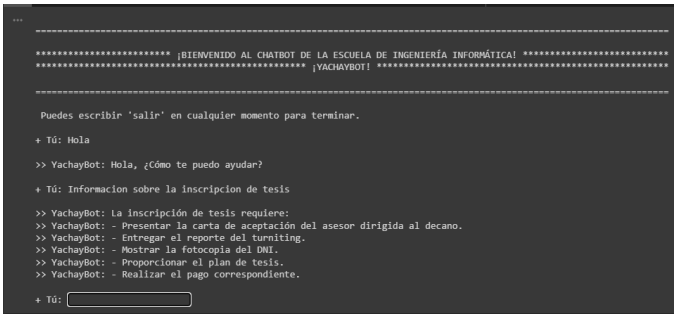

Fig. 1. Prueba de saludo y requisitos de inscripcion de tesis

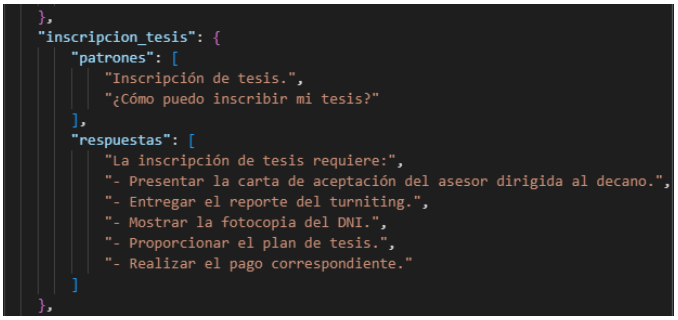

Fig. 2. Validación de la respuesta basandonos en el corpus

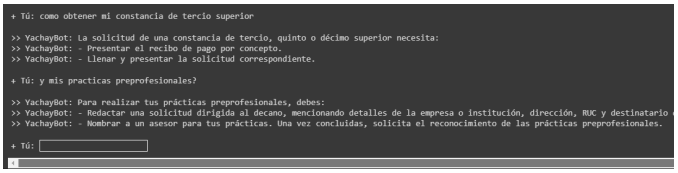

Fig. 3. Consultas sobre el tercio superior y sobre la prácticas preprofesionales

C. Posibles Problemas Durante la Ejecución

Durante el proceso de ejecución de nuestro chatbot, es esencial considerar una variedad de situaciones en las que pueden surgir problemas o fallos. Entre los desafíos identificados, se destaca la posibilidad de que no se encuentre al menos una palabra clave que coincida con alguna de las expresiones en el corpus de patrones, que la consulta sea una cadena que conste de más de 15 palabras. Estos escenarios pueden llevar a respuestas inadecuadas o nulas por parte de nuestro chatbot, lo que a su vez afecta la calidad de la interacción con los usuarios. A continuación, se presenta un ejemplo concreto de una prueba en la que se observa una respuesta fallida por parte de nuestro chatbot, conocido como ”YachayBot”.

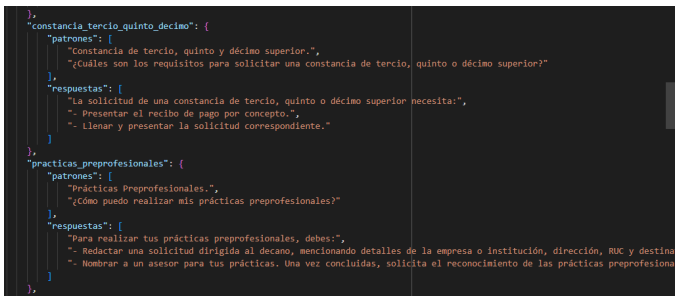

Fig. 4. Verificación sobre ambas consultas

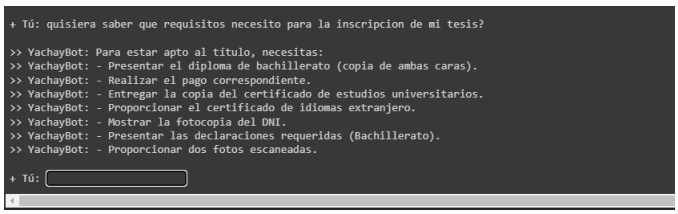

Fig. 5. Error sobre la consulta de requisitos de inscripcion de tesis

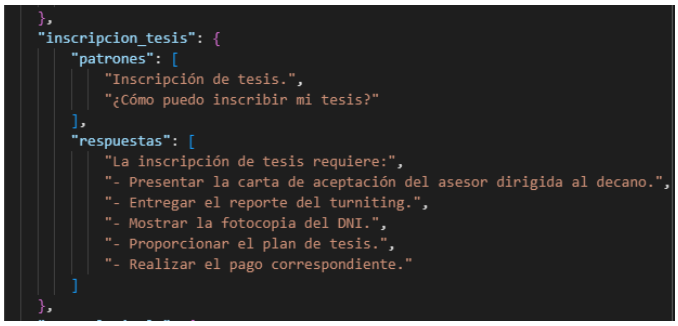

Fig. 6. Valores correctos en el corpus

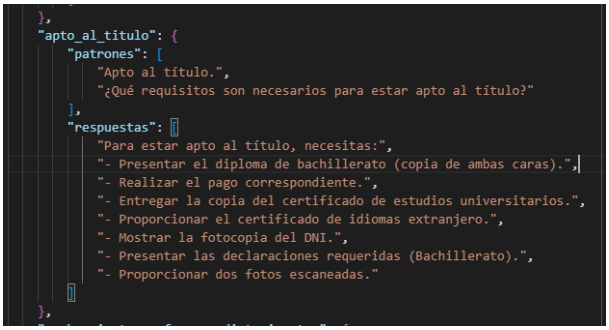

Fig. 7. Valores que muestra ya que encontro primero la palabra clave de requisitos

CONCLUSIONES

Finalmente el análisis de la situación actual de la Escuela Profesional de Ingeniería Informática y de Sistemas nos permite determinar la necesidad de implementar un chatbot en el proceso de atención a estudiantes y docentes.

El uso de tecnologías ha facilitado el desarrollo del chatbot, el cual, a través del procesamiento del lenguaje natural, nos permite comprender las consultas de los usuarios y resolver las dudas que puedan tener con respecto a los trámites documentarios realizados en la escuela profesional.

REFERENCES

[1] Bautista, O. V. (2023). Chatbots: la evolución de la atención al cliente en la era digital. Con-Ciencia Boletín Científico de la Escuela Preparatoria No. 3, 10(20), 24-27.

[2] Natural Language Toolkit. (s.f.). *NLTK - Natural Language Toolkit*. Recuperado de https://www.nltk.org/

[3] scikit-learn. (s.f.). *scikit-learn*. Recuperado de https://scikit-learn.org/stable/

[4] spaCy. (s.f.). *spaCy*. Recuperado de https://spacy.io/