UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

OPTIMIZACIÓN DEL TRÁMITE DOCUMENTARIO VIRTUAL EN LA UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO, MEDIANTE LA IMPLEMENTACIÓN DE UN CHATBOT

PRESENTADO POR:
- BR. CRISTIAN FERNANDO BECERRA YARIN
- BR. EBER BRYAN CHAVEZ MONTESINOS

PARA OPTAR AL TÍTULO PROFESIONAL DE INGENIERO INFORMÁTICO Y DE SISTEMAS

ASESOR:
DR. RONY VILLAFUERTE SERNA

Cusco - Perú
2024

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: Optimización del trámite documentario virtual en la Universidad Nacional de San Antonio Abad del Cusco mediante la implementación de un chatbot.

presentado por: Cristian Fernando Becerra Yarin con DNI Nro.: 73197763

presentado por: Eber Bryan Chávez Monteiros con DNI Nro.: 71559442

para optar el título profesional/grado académico de Ingeniero Informático y de Sistemas

Informo que el trabajo de investigación ha sido sometido a revisión por 3 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 2 %.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 31 de enero de 2024

Firma

Post firma: Dr. Rony Tillahuante Serna

Nro. de DNI: 23957778

ORCID del Asesor: 0000-0003-4607-522X

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: OID : 27259 : 318602727

turnitin

Identificación de reporte de similitud: oid:27259:318602727

NOMBRE DEL TRABAJO

chatBotUNSAACv3

AUTOR

Cristian & Eber Becerra & Chavez

RECUENTO DE PALABRAS

18825 Words

RECUENTO DE CARACTERES

104209 Characters

RECUENTO DE PÁGINAS

85 Pages

TAMAÑO DEL ARCHIVO

4.8MB

FECHA DE ENTREGA

Jan 30, 2024 9:51 PM GMT-5

FECHA DEL INFORME

Jan 30, 2024 9:53 PM GMT-5

● 2% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base c

- 2% Base de datos de Internet
- Base de datos de Crossref
- 2% Base de datos de trabajos entregados
- 0% Base de datos de publicaciones
- Base de datos de contenido publicado de Crossr

● Excluir del Reporte de Similitud

- Material bibliográfico
- Coincidencia baja (menos de 20 palabras)
- Material citado
- Bloques de texto excluidos manualmente

Resumen

Dedicatorias

Dedicado a toda mi familia, a mi padre que no está conmigo pero aún vive en mis recuerdos y especialmente a mi madre, quien ha sido mi fuente de inspiración y quien me ha enseñado a ser único, determinado, a creer en mí mismo y a perseverar siempre.

Cristian Fernando Becerra Yarin

Dedico esta tesis a mi familia, especialmente a mis padres y abuelos que siempre me dieron su apoyo.

Eber Bryan Chávez Montesinos

Agradecimientos

Agradecemos a Dios por ser nuestra fortaleza en los momentos más difíciles y permitirnos lograr nuestro objetivo.
Agradecemos a nuestro asesor Dr. Rony Villafuerte Serna, ya que cada vez que nos encontrábamos con un problema o teníamos una pregunta sobre nuestra investigación, nos guió en la dirección correcta cuando lo necesitabamos, agradecidos por la enseñanza durante nuestros estudios y a todos los que participaron en nuestra carrera universitaria.
Terminar nuestra tesis requirió más que apoyo académico, tenemos muchas personas a quienes agradecer por escucharnos y, en ocasiones, tener que tolerarnos durante los últimos dos años. No podemos comenzar a expresar nuestra gratitud y aprecio por su amistad.

Resumen

La pandemia causada por el virus SARS-CoV-2 ha generado que muchas actividades se desarrollen de manera remota. De esta manera la aplicación de chatbots ha logrado un impulso exponencial en todo ámbito. En el caso de trámite documentario, la UNSAAC usa un chatbot basado en reglas para dar asistencia virtual a los usuarios. Dicho chatbot es poco intuitivo y carece de autonomía, por lo que depende de la asistencia humana, la cual a su vez está sujeta a más limitaciones(horarios, disponibilidad, respuestas inmediatas, estado de ánimo), por ende no cumple su propósito de manera óptima. Para optimizar la asistencia virtual se implementó un chatbot basado en Inteligencia Artificial utilizando el framework RASA del lenguaje de programación Python. Para la implementación se utilizó funcionalidades de RASA, las cuales aseguraron la usabilidad y autosuficiencia del modelo. Es así que los resultados obtenidos demostraron que el modelo propuesto obtuvo una alta aprobación por parte de los usuarios. Basado en estos resultados, se concluye que se optimizaría el proceso de trámite haciendo uso de este chatbot.

Palabras clave: Chatbot, inteligencia artificial, procesamiento del lenguaje natural, RASA.

Abstract

The pandemic caused by the SARS-CoV-2 virus has caused many activities to be carried out remotely. Thus, the application of chatbots has gained exponential momentum in all areas. In the case of document processing, the UNSAAC uses a rule-based chatbot to provide virtual assistance to users. This chatbot is not very intuitive and lacks autonomy, so it depends on human support, which in turn is subject to more limitations (schedules, availability, immediate responses, mood), and therefore does not fulfill its purpose optimally. To optimize virtual assistance, we implemented an AI-based chatbot using the RASA framework of the Python Programming Language. RASA functionalities were used for the implementation, which ensured the usability and self-sufficiency of the model. Thus, the results obtained showed that the proposed model obtained a high approval by the users. Based on these results, it is concluded that the process would be optimized by using this chatbot.

Keywords: Chatbot, artificial intelligence, natural language processing, RASA.

Introducción

En la era actual, caracterizada por la rápida evolución tecnológica y la creciente digitalización de los servicios, los chatbots han emergido como una herramienta fundamental para mejorar la interacción entre usuarios y sistemas automatizados. Estos asistentes virtuales, alimentados por inteligencia artificial, han ganado prominencia en una variedad de sectores, desde el ámbito empresarial hasta el educativo, proporcionando soluciones eficientes y accesibles.

En este contexto, la presente tesis busca demostrar cómo un chatbot diseñado a medida puede potenciar la eficiencia y la calidad de las interacciones en el proceso de trámite documentario en la UNSAAC. A través de una combinación de investigación teórica, desarrollo práctico y evaluación crítica, este trabajo aspira a ofrecer una contribución significativa al campo de la inteligencia artificial aplicada a los asistentes virtuales.

El desarrollo del trabajo se estructura por capítulos: En el primer capítulo, se realiza una breve introducción al proyecto, cuales son sus objetivos y la motivación que ha empujado a su desarrollo. Asimismo, se describe la metodología de trabajo empleada durante toda su realización y una breve descripción de la estructura del documento. En el segundo capítulo, se da a conocer la base teórica en la que se basa el proyecto: se describe el estado actual del arte, investiga en los detalles a considerar en el diseño del chatbot. En el tercer capítulo, se muestra todo el proceso del desarrollo del proyecto, desde el análisis de requisitos hasta el diseño e implementación, así como las pruebas realizadas. Finalmente, en el cuarto capítulo, se lleva a cabo una evaluación conclusiva del chatbot para verificar el cumplimiento de todos los objetivos establecidos al inicio del proyecto.

Listado de Abreviaturas

- NLP: Natural Language Processing. Campo de la inteligencia artificial que se encarga del procesamiento del texto para que sea legible por las máquinas.
- IA: Inteligencia Artificial.
- HTTP: Hypertext Transfer Protocol.
- REST: Representational State Transfer.
- API: Application Programming Interface.
- YAML: Acronimo recursivo: YAML(Yet Another Markup Language) Ain't Markup Language
- Bag of words: Representación vectorial de un texto que describe cuántas veces aparece cada palabra dentro de un documento.
- DIET: Arquitectura transformer multitarea que maneja la clasificación de intenciones y el reconocimiento de entidades.
- CDD: Conversation Driven Development
- Endpoint: Extremo final de un canal de comunicación donde una API envía una solicitud y emite la respuesta.
- Entidad: Término relevante del mensaje del usuario.
- Intención: Propósito del mensaje del usuario.
- Policy: Componente que decide qué acción ejecutar en cada paso de un diálogo.
- History: Representación de una conversación entre un usuario y el chatbot. Las entradas del usuario se expresan con las intenciones correspondientes (y entidades cuando es necesario).
- LSTM: Long Short Term Memory, tipo de red neural recurrente capaz de aprender dependencias a largo plazo.
- Pipeline: Conjunto de técnicas para el procesamiento de datos conectados en serie de manera que la salida de cada método es la entrada de alguna siguiente.
- Chatbot: Programa informático que simula una conversación humana, formado a partir de la unión de chat y robot.
- Token: Puede ser una palabra, un carácter o una subpalabra.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Aspectos Generales

1.1. Planteamiento del Problema

1.1.1. Descripción del problema

En la actualidad, la aplicación de los chatbot se ha ido volviendo muy popular en todos los rubros a nivel mundial, por ejemplo: asistentes personales virtuales, finanzas, educación, climáticas, agrícolas, logística y transporte, sanidad, comercial y entretenimiento. No sólo están presentes en páginas web sino también en aplicaciones móviles y de escritorio, de esta manera los chatbots siguen creciendo exponencialmente en todo ámbito.

En el Perú existen distintas empresas que desarrollan chatbots construidos con Inteligencia Artificial; por ejemplo, BotCenter, que ofrecen servicios bajo la modalidad de "Software como servicio"(SaaS) y se encargan de la gestión utilizando chatbots. Además, cuentan con un equipo humano que supervisa el correcto funcionamiento de su bot y puede tomar control de la conversación para el manejo de situaciones complejas (BotCenter, 2021).

El 13 de octubre del 2021, el diario Andina.pe publicó un artículo titulado "El 58 % de usuarios peruanos se comunicó mediante chatbots durante este año" donde señalan que debido a la pandemia del covid-19, las marcas han fortalecido su presencia digital con el uso de las redes sociales y sitios web como canales de comunicación. De hecho, el 58 % de usuarios peruanos afirma haber interactuado con chatbots en lo que va del presente año, según una encuesta de la agencia de marketing digital Play Group (Andina, 2021).

En la región de Cusco existen empresas que en sus páginas web, parte tecnológica fundamental de una institución, no cuentan con chatbots o estos son muy limitados en la interacción con los usuarios. Una de estas instituciones es la Universidad Nacional de San Antonio Abad del Cusco, que actualmente ha optado por adquirir un chatbot basado en reglas, administrado por un personal de trabajo encargado de responder en caso se realice una consulta distinta a las opciones predeterminadas que ofrece. Esto provoca que la asistencia virtual dependa de una sola persona, la cual no es la más óptima ya que está limitada a la capacidad humana. Por ejemplo, este personal se puede tardar en responder si hay varios usuarios consultando paralelamente o si simplemente no se encuentra disponible, lo cuál causa que la interacción se vea prolongada. Además este personal está limitado a su horario, por lo que la asistencia

virtual no tiene disponibilidad continua. De esta manera, se produce cierta disconformidad por parte de los usuarios, lo que podría generar reclamos de su parte e incluso errores en sus trámites. Debido a estas razones, este chatbot no cumple su funcionalidad al 100 %.

1.1.2. Identificación del problema

El chatbot en la página web de trámite documentario de la UNSAAC presenta limitaciones y depende en gran medida de la intervención humana, lo que afecta su eficiencia y funcionalidad.

1.2. Formulación del Problema

1.2.1. Problema General

El chatbot en la página web de trámite documentario de la UNSAAC no es óptimo por ser limitado y depender de una persona.

1.2.2. Problemas Específicos

- El chatbot actual es poco intuitivo.
- La asistencia virtual depende de la disponibilidad de un encargado, esto implica limitaciones humanas.
- La usabilidad y eficiencia del chatbot son limitadas.

1.3. Objetivos

1.3.1. Objetivo General

Optimizar el proceso de trámite documentario de la UNSAAC mediante la implementación de un chatbot basado en inteligencia artificial.

1.3.2. Objetivos Específicos

- Desarrollar un modelo capaz de manejar una conversación natural.
- Establecer un chatbot autosuficiente con disponibilidad constante.
- Obtener un grado de usabilidad satisfactorio en el modelo implementado.

1.4. Justificación

1.4.1. Conveniencia

La conveniencia de implementar un chatbot basado en inteligencia artificial en la plataforma de trámite documentario de la UNSAAC se fundamenta en la necesidad de mejorar la experiencia y eficiencia del usuario. En la actualidad, el chatbot existente, basado en reglas y gestionado por personal administrativo, presenta limitaciones en disponibilidad de horarios, generando demoras y disconformidad en los usuarios. La opción propuesta de un chatbot basado en inteligencia artificial ofrece respuestas de mayor calidad al generar texto en función de los datos ingresados por el usuario.

1.4.2. Relevancia

La adopción de esta solución no solo impactará positivamente a los usuarios de la página de trámites de la UNSAAC, sino que también servirá como un ejemplo motivador para otras instituciones en la región del Cusco, fomentando la adquisición de chatbots basados en inteligencia artificial y promoviendo así una transformación tecnológica y social más amplia en la gestión de trámites documentarios.

1.4.3. Implicancias Prácticas

Las implicancias prácticas de estudiar y aplicar chatbots basados en inteligencia artificial incluyen mejoras significativas en la eficiencia operativa, la optimización de recursos, la reducción de errores, una experiencia mejorada para el usuario y el fomento de la innovación tecnológica a nivel institucional y regional.

1.4.4. Valor Teórico

Este proyecto proporciona un marco para comprender, diseñar e implementar chatbots basados en inteligencia artificial de manera efectiva, maximizando su valor en el contexto específico de trámites documentarios universitarios.

1.4.5. Utilidad Metodológica

Este proyecto está desarrollado para una solución específica; sin embargo, se puede aplicar a diferentes áreas en cualquier institución.

1.5. Delimitación de estudio

1.5.1. Delimitación Espacial

La delimitación espacial abarca la UNSAAC, ubicada en Cusco-Perú, donde se llevó a cabo la recopilación de datos, interacción con los usuarios y análisis de resultados.

1.5.2. Delimitación Temporal

Esta propuesta fue desarrollada durante los años 2022 y 2023.

1.6. Método

1.6.1. Alcance

- Este proyecto propone optimizar la asistencia virtual en el trámite documentario mediante la implementación de un chatbot para el servicio de atención al usuario en el sitio web de trámite virtual de la UNSAAC, dicho chatbot es un prototipo por lo que no fue desplegado.
- Los resultados beneficiarán a los usuarios del sitio web, lo cual servirá como base a que también se pueda implementar para otras áreas de la universidad que buscan mejorar su servicio.
- El chatbot tiene disponibilidad 24/7.

1.6.2. Diseño

Debido a la naturaleza de la tesis, se utilizó la metodología aplicada (Tamayo, 2004), ya que se recopilará información que permitirá lograr un objetivo concreto, además se desarrolla la investigación en entorno específico al implementar un chatbot inteligente utilizando conceptos de IA.

El conjunto de fases que se siguen son:

1. **Revisión de literatura:** En esta etapa se recolecta y analiza todas las investigaciones referentes a los chatbots, especialmente los implementados usando el framework Rasa, así como también la documentación oficial de dicho framework.
2. **Generación de dataset:** En esta fase, se crea el dataset con las casuísticas de los diferentes tipos de consultas que se le hacen al chatbot. Es importante usar diferentes oraciones para una misma intención, para que se obtenga un dataset más variado y amplio.

3. **Implementación:** Para esta etapa seguimos la metodología mostrada en la figura 1.1, en la cual se detalla el proceso a seguir para obtener un modelo.

4. **Evaluación:** El mejor modelo obtenido en la fase anterior es evaluado por los investigadores.

5. **Análisis de resultados:** En esta etapa se analiza los resultados obtenidos de la evaluación para determinar si se logró los objetivos propuestos o no.

1.6.3. Para el desarrollo de la parte informática

Para la implementación del chatbot inteligente, se utilizó la metodología recuperada de (Pinto and Quispetupa, 2019), puesto que se basa en el desarrollo de un chatbot. No obstante, se aplicaron algunas modificaciones para adaptarlo al proyecto de investigación.

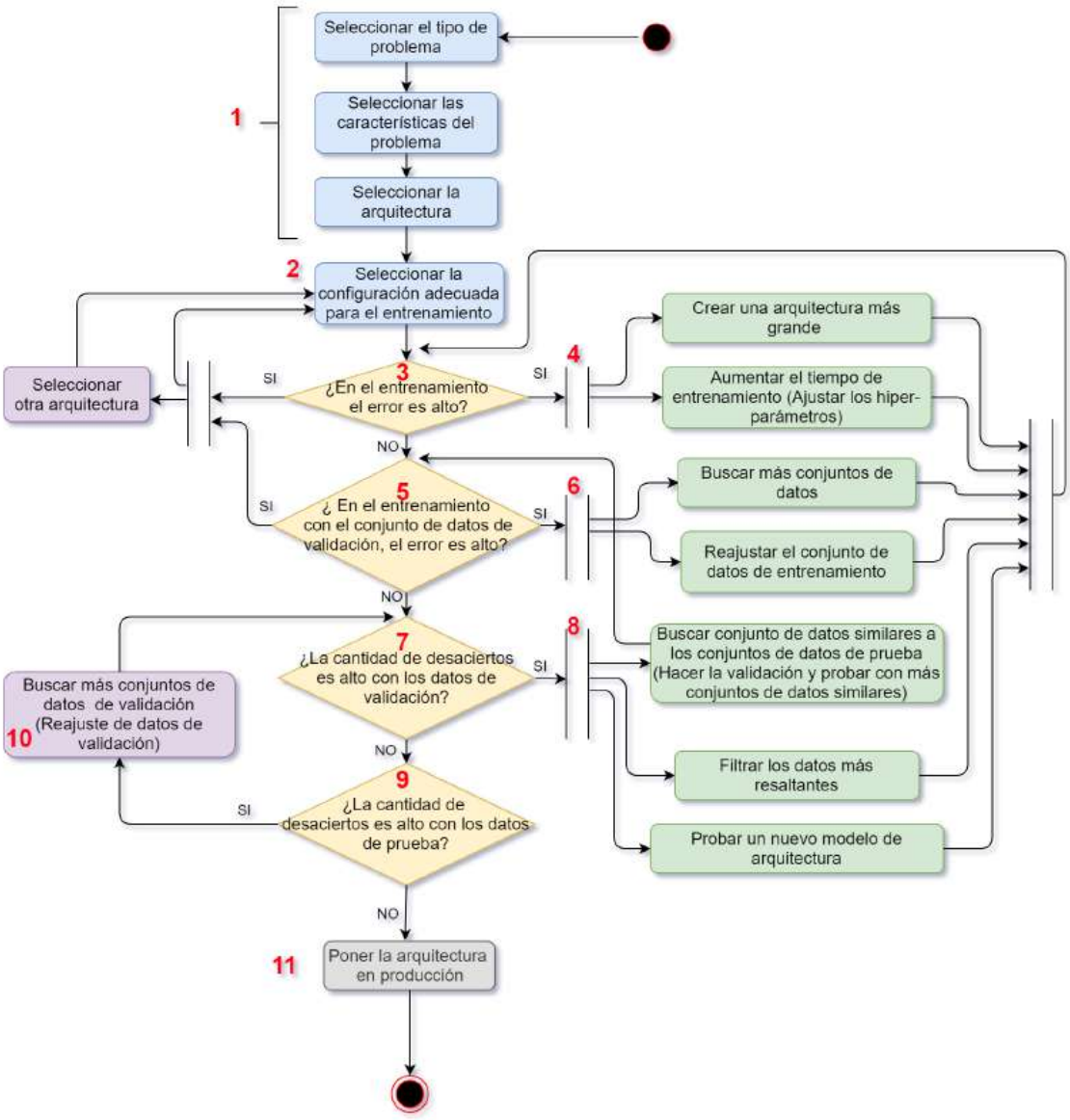


Figura 1.1: Metodología para la implementación del chatbot.
Fuente: (Pinto and Quispetupa, 2019)

Esta metodología consta de las siguientes etapas:

1. En esta etapa, primero se define el problema y sus características para luego seleccionar el modelo de arquitectura adecuada, en el caso del framework Rasa esto se define en el NLU pipeline y el dialogue management (Rasa Core).

2. Aquí se realizará la primera configuración en la arquitectura de la red neuronal, así como el preprocesamiento de los conjuntos de datos y el ajuste de los hiper-parámetros.

3. En esta etapa se realizará el entrenamiento, y de acuerdo al error obtenido se tomará las siguientes decisiones.

4. Si el error en el paso 3 es alto, entonces crearemos una arquitectura más grande (agregaremos más capas) o realizaremos un re-ajuste de los hiper-parámetros y volveremos al paso 3.

5. En caso que el error en el paso 3 sea mínimo, entonces, continuaremos el entrenamiento con el conjunto de datos de validación.

6. Si el error obtenido en el paso 5, es alto, entonces buscaremos más conjuntos de datos o re-ajustaremos el conjunto de datos de entrenamiento y volveremos al paso 3.

7. En caso que el error obtenido en el paso 5 no es alto, realizaremos la comprobación con los datos de validación.

8. Si la cantidad de desaciertos es alta en el paso 7, entonces buscaremos un conjunto de datos similares al conjunto de datos de prueba para luego volver al paso 5, o filtraremos los datos más resaltantes y volveremos al paso 3.

9. En el caso que la cantidad de desaciertos no es alta en el paso 7, entonces comprobaremos con los datos de prueba.

10. Si la cantidad de desaciertos es alta en el paso 9, entonces buscaremos más conjuntos de datos de validación, y volveremos al paso 7.

11. En el caso que, la cantidad de desaciertos no es alta en el paso 9, entonces podemos decir que hemos encontrado la arquitectura adecuada con sus respectivos hiper-parámetros y estará listo para entrar en producción.

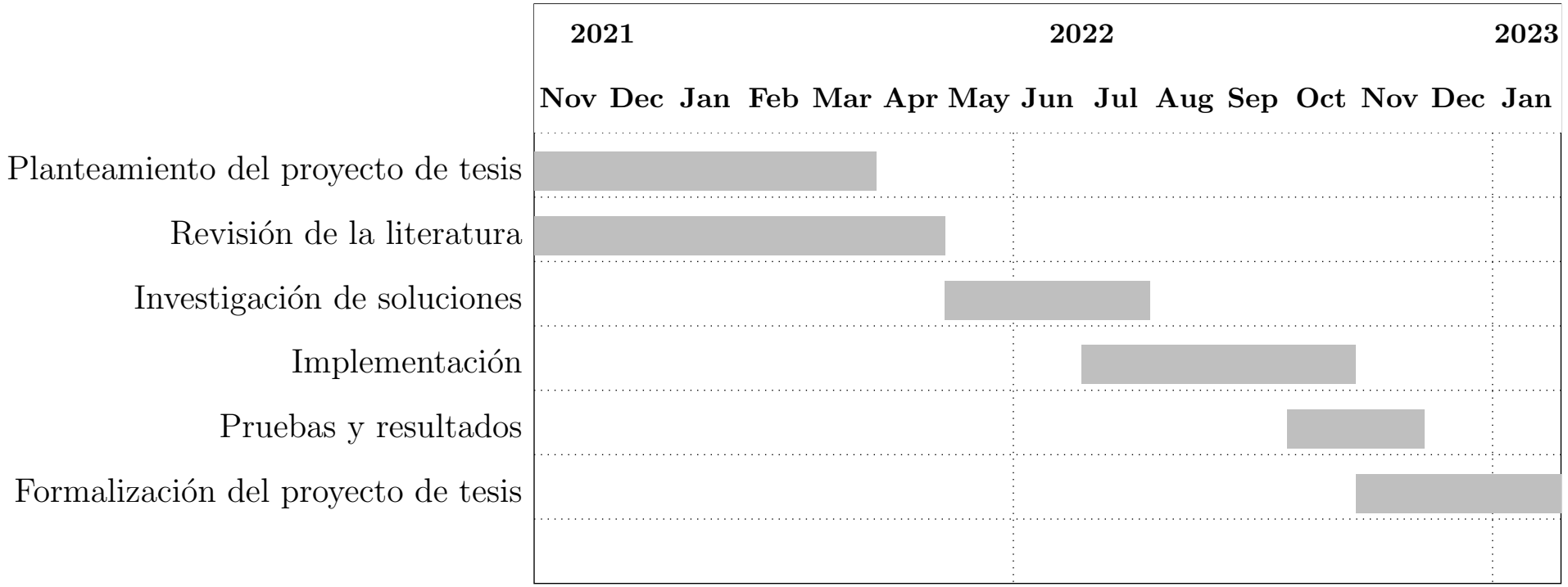
1.6.4. Cronograma de Actividades

Tabla 1.1: Cronograma de actividades

Actividades	**Duración (Semanas)**	**Fecha Inicio**	**Fecha Fin**
Planteamiento del proyecto de tesis	19	01/11/2021	08/03/2022
Revisión de la literatura	23	22/11/2021	29/04/2022
Investigación de soluciones	10	02/05/2022	08/07/2022
Implementación	13	11/07/2022	7/10/2022
Pruebas y resultados	7	10/10/2022	25/11/2022
Formalización del proyecto de tesis	6	28/11/2022	06/01/2023

Fuente: Elaboración Propia

Figura 1.2: Diagrama de Gantt.



Fuente: Elaboración Propia.

Capítulo 2

Marco Conceptual

2.1. Antecedentes

A continuación, se mencionan algunos antecedentes de la bibliografía revisada y que son útiles en la investigación:

2.1.1. Antecedentes Internacionales

(Bavishi, 2019), "IMPLEMENTING A COLLEGE ENQUIRY CHATBOT", Tesis consultada de California State University, Sacramento.
Conclusiones:

- Para concluir, College Enquiry Chatbot es útil para guiar a los estudiantes con fuentes de información actualizadas y verídica. Es ventajoso para consultas de solicitantes internacionales tales como pago de cuotas y asuntos académicos. Los estudiantes pueden obtener información en la palma de sus manos en lugar de ir a la universidad. Esto mejora la eficiencia al hacerse cargo de tareas en las que los humanos no son esenciales.
- El análisis de sentimientos implementado en College Enquiry Chatbot reconoce correctamente la consulta del usuario, como positiva, negativa y neutral, almacenando todas las conversaciones en la base de datos. Sin embargo, el sistema fue parcialmente exitoso en agregar empatía ya que el alcance de estas consultas es amplio y el sistema requiere datos más rigurosos para manejar todas las consultas que están fuera de contexto. Sin embargo, el aprendizaje activo ayuda a mejorar el rendimiento del bot para estas consultas.
- Para mejorar en el futuro las funcionalidades actuales de College Enquiry Chatbot, el alcance del chatbot se puede aumentar insertando datos de todos los departamentos, entrenando el bot con datos variados, haciendo pruebas en un sitio web en linea, y en base al feedback insertando más datos de entrenamiento.

Comentario: Esta tesis de postgrado nos sirvió de ejemplo para implementar nuestro chatbot, ya que utiliza uno para resolver consultas similares en la universidad de California.

Además, utiliza análisis de sentimientos para reconocer la consulta del usuario y generar su respuesta en base a dicho análisis, lo cual hace que se asemeje más al lenguaje natural humano.

(Ingale et al., 2021), "College Enquiry CHATBOT using RASA", Paper consultado de la Dr. D. Y. Patil School of Engineering, India
Conclusiones:

- Los chatbots se están convirtiendo en una pieza básica del mundo computarizado.
- Los estudiantes y trabajadores pueden realizar públicamente sus preguntas.
- El bot ofrece respuestas rápidas y eficaces a las consultas y obtiene información significativa de estas.
- El objetivo del sistema es ayudar a los estudiantes a permanecer actualizados con su universidad.
- El motivo principal del proyecto es reducir la carga de trabajo del personal de oficina de la universidad y reducir el tiempo de respuesta a las consultas de los usuario.

Comentario: El paper nos ayudó a entender de mejor manera el funcionamiento de chatbots para responder consultas administrativas de la universidad, debido a que fue implementado con el framework RASA y este es el que utilizamos en nuestro proyecto.

(Teodoro, 2021), "VIHRTUAL-APP: Un chatbot para la divulgación médica del VIH", Tesis consultada de la UNIVERSITAT POLITÈCNICA DE VALÈNCIA ESCOLA POLITÈCNICA SUPERIOR DE GANDIA, GANDIA.
Conclusiones:

- Durante la realización de este trabajo se han alcanzado con éxitos los objetivos planteados inicialmente. Se ha explorado el estado actual de creación de chatbots y se han estudiado distintos principios de diseño para aplicarlos al proyecto. Posteriormente, todo este conocimiento adquirido se ha plasmado en el diseño e implementación del chatbot, implementando todas las funcionalidades inicialmente previstas.
- Tras su publicación, se han podido recopilar las conversaciones mantenidas por los usuarios con el servicio, de manera que ha sido posible trabajar en base a esta información para ir corrigiendo y mejorando el servicio. Este modo de trabajo ha producido una buena sinergia con los colaboradores de la Unidad de Enfermedades Infecciosas del Hospital General de Elche, que en última instancia, ha repercutido positivamente en el desarrollo. Gracias a la recogida de datos y su colaboración, ha sido posible recopilar y responder hasta 40 nuevas preguntas que inicialmente no estaban previstas, añadiendo una notable mejora a la base de conocimiento del chatbot.

Comentario: El chatbot implementado en este trabajo de investigación nos sirvió de referencia para entender el funcionamiento del framework RASA utilizando el idioma español y también la aplicación del CUQ que está específicamente dirigido a los chatbots que nos permitió recibir información muy importante sobre algunos aspectos de su funcionamiento.

2.1.2. Antecedentes Nacionales

(Pinto and Quispetupa, 2019), "Chatbot generativo en el idioma español utilizando la arquitectura de red neuronal Transformer", UNSAAC, PERÚ.
Conclusiones:

- Para este proyecto se utilizó dos dataset full opensubtitle y tiny opensubtitle siendo este último un subconjunto del primero; por consiguiente, el preprocesamiento fue de la misma manera para ambos; sin embargo, no podemos precisar que se haya quitado todas las irregularidades, dado que ambos dataset son de tamaño grande y de carácter ruidoso.
- La arquitectura predeterminada de vanilla transformer tiene buenos resultados cuando se entrena con grandes y pequeños dataset (full opensubtitle 88,147,503 lineas y tiny opensubtitle 100,730 lineas), pero obtiene mejores resultados cuando se disminuye la cantidad de capas ocultas para pequeños conjuntos de datos (Tabla B.2.1). Por otro lado, la arquitectura preestablecida de universal transformer tiene buenos resultados cuando se entrena con grandes y pequeños conjuntos de datos (Tablas B.1.3 y B.2.3). En general, se ha visto que los modelos entrenados con mayores cantidades de datos, responden mejor.
- La estrategia de entrenar en forma paralela a partir de los 200K steps hasta 1M steps con average checkpoint en periodos de 50K steps para la arquitectura 1 (Sección 3.4.1) ayudo a que nuestro modelo tenga una mejor precisión, obteniendo un 60 % de respuestas buenas y un 76 % de respuestas coherentes en la fase de evaluación (Figura 4.3.2).
- Para nuestros mejores modelos obtenidos en la fase de pruebas (arquitectura 1 y 3) se implementó prototipos de chat utilizando tecnologías basadas en la web (Figura 4.2.1); esto permitió el acceso al chat desde cualquier dispositivo, generando una mejor interacción entre los evaluadores y el chatbot.

Comentario: Esta tesis nos sirvió de referencia bibliográfica, ya que nos ayudó a obtener los conceptos básicos de los chatbots y su implementación. Además dentro de sus trabajos futuros encontramos la motivación para implementar un chatbot específico, en este caso enfocado a trámites administrativos.

2.2. Bases teóricas

2.2.1. Inteligencia Artificial

En informática, es la inteligencia expresada por máquinas, sus procesadores y sus softwares, que serían los análogos al cuerpo, el cerebro y la mente, respectivamente, a diferencia de la inteligencia natural demostrada por humanos y ciertos animales con cerebros complejos. En ciencias de la computación, una máquina «inteligente» ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea.([Russell and Norvig, 2021]) diferencian varios tipos de inteligencia artificial:

- Sistemas que piensan como humanos: Estos sistemas tratan de emular el pensamiento humano; por ejemplo las redes neuronales artificiales. La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas y aprendizaje.
- Sistemas que actúan como humanos: Estos sistemas tratan de actuar como humanos; es decir, imitan el comportamiento humano; por ejemplo la robótica (El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor).
- Sistemas que piensan racionalmente: Es decir, con lógica (idealmente), tratan de imitar el pensamiento racional del ser humano; por ejemplo, los sistemas expertos, (el estudio de los cálculos que hacen posible percibir, razonar y actuar).
- Sistemas que actúan racionalmente: Tratan de emular de forma racional el comportamiento humano; por ejemplo los agentes inteligentes, que está relacionado con conductas inteligentes en artefactos.

2.2.2. Machine Learning

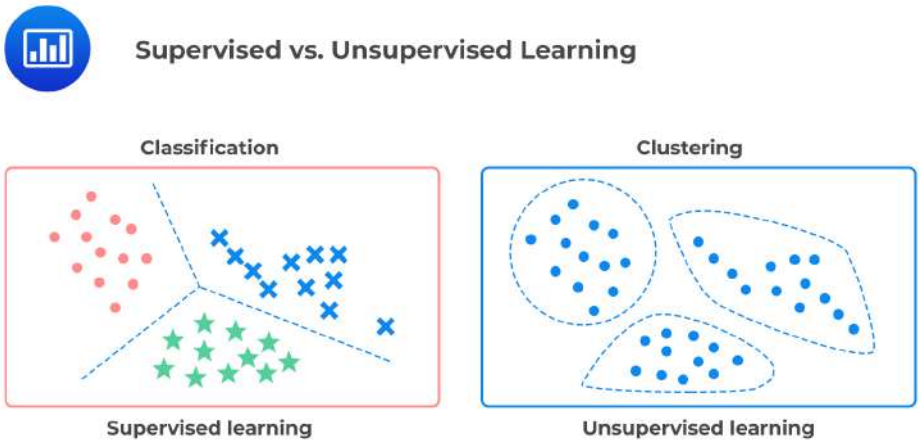
Machine learning (aprendizaje automático) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento. “En el aprendizaje de máquinas un computador observa datos, construye un modelo basado en esos datos y utiliza ese modelo a la vez como una hipótesis acerca del mundo y una pieza de software que puede resolver problemas”([Russell and Norvig, 2021]).

2.2.2.1. Tipos de Aprendizaje Automático

En ([AnalystPrep, 2021]), los enfoques de machine learning se dividen tradicionalmente en tres categorías amplias, que corresponden a paradigmas de aprendizaje, según la naturaleza de la “retroalimentación”disponible para el sistema de aprendizaje:

- **Aprendizaje supervisado:** A la computadora se le presentan entradas de ejemplo y sus salidas deseadas, dadas por un "maestro", y el objetivo es aprender una regla general que asigna entradas a salidas. De esta forma es como el algoritmo va "aprendiendo" a clasificar las muestras de entrada comparando el resultado del modelo, y la etiqueta real de la muestra, realizando las compensaciones respectivas al modelo de acuerdo a cada error en la estimación del resultado. Por ejemplo, el aprendizaje supervisado ha sido utilizado para la programación de agentes virtuales de atención al cliente.
- **Aprendizaje no supervisado:** Los algoritmos de aprendizaje no supervisado trabajan de forma muy similar a los supervisados, con la diferencia de que no se asignan etiquetas al algoritmo de aprendizaje, dejándolo solo para encontrar una estructura en su entrada. El aprendizaje no supervisado puede ser un objetivo en sí mismo (descubrir patrones ocultos en los datos) o un medio para lograr un fin (aprendizaje de funciones).
- **Aprendizaje por refuerzo:** Un programa de computadora interactúa con un entorno dinámico en el que debe realizar un objetivo determinado (como conducir un vehículo o jugar un juego contra un oponente). A medida que navega por su espacio problemático, el programa recibe retroalimentación que es análoga a las recompensas, que trata de maximizar. Este tipo de métodos pueden usarse para hacer que los robots aprendan a realizar diferentes tareas.

Figura 2.1: Ilustración de Aprendizaje Supervisado y no Supervisado



Fuente: (AnalystPrep, 2021)

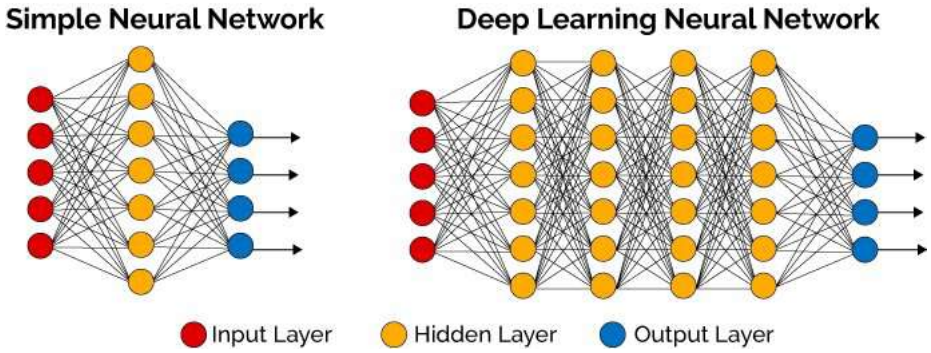
Para aplicar ML se necesita crear un modelo, que se entrena en algunos datos de entrenamiento y luego puede procesar datos adicionales para hacer predicciones. Se han utilizado e investigado varios tipos de modelos para los sistemas de aprendizaje automático:

- Artificial Neural Networks (ANN)
- Arboles de Decisión
- Support-Vector Machines (SVM)
- Análisis de Regresión
- Redes Bayesianas
- Algoritmos Genéticos

2.2.3. Deep Learning

Deep Learning (aprendizaje profundo) es una amplia familia de técnicas de machine learning basadas en redes neuronales artificiales que se usan para el aprendizaje de características. Este aprendizaje puede ser supervisado, semi-supervisado o no supervisado.
La palabra "profundo"se refiere al hecho de que los circuitos normalmente se organizan en muchas capas, lo que significa que las rutas de cálculo desde las entradas hasta las salidas tienen muchos pasos. (Torres, 2017)

Figura 2.2: Representación de Deep Learning



Fuente: (Torres, 2017)

El aprendizaje profundo es actualmente el enfoque más utilizado, sus arquitecturas como redes neuronales recurrentes (RNN), redes neuronales convolucionales (CNN) y transformers son ampliamente utilizadas para aplicaciones tales como el reconocimiento visual de objetos, traducción automática, reconocimiento de voz, síntesis de voz y síntesis de imágenes, bioinformática, diseño de fármacos, análisis de imágenes médicas, videojuegos; en los que han producido resultados comparables y, en algunos casos, superiores al rendimiento de expertos. También juega un papel importante en las aplicaciones de aprendizaje por refuerzo.

2.2.4. Redes Neuronales Artificiales

2.2.4.1. Feedforward Neural Networks

Para (Russell and Norvig, 2021), una red feedforward, como su nombre indica, se mueve en una única dirección: adelante. De los nodos de entrada, a través de los nodos escondidos (si los hay) hacia los nodos de salida. Cada nodo calcula una función de sus entradas y pasa el resultado a sus sucesores en la red, sin bucles.
Cada nodo dentro de una red se denomina unidad. Tradicionalmente, siguiendo el diseño propuesto por McCulloch y Pitts, una unidad calcula la suma ponderada de las entradas del predecesor nodos y luego aplica una función no lineal para producir su salida. Denotemos a_j la salida de la unidad j y sea $W_{i,j}$ el peso asociado al enlace de la unidad i a la unidad j; entonces tenemos:

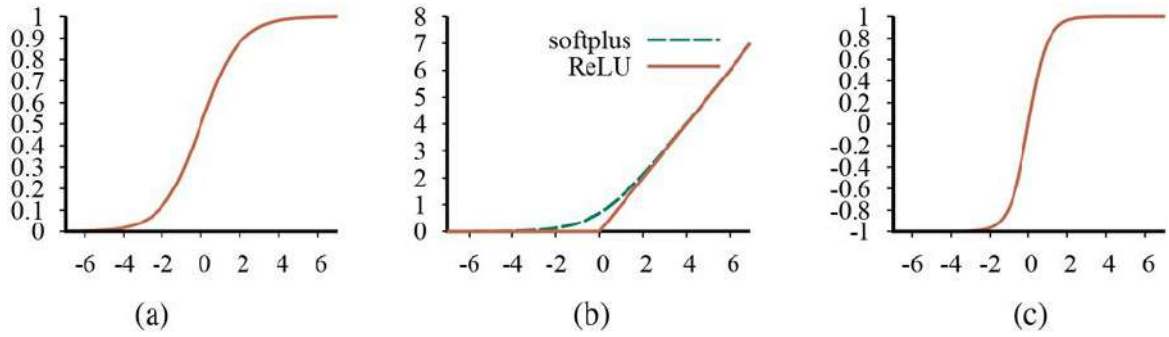
$$a_j = g_j(\sum_i w_{i,j} a_i)$$

donde g_j es una función de activación no lineal asociada con la unidad j, y esta se multiplica por la suma ponderada de las entradas a dicha unidad.

Se utiliza una variedad de diferentes funciones de activación. Los más comunes son las siguientes:

- La función logística o sigmoidea, que también se utiliza en la regresión logística:
$$\sigma(x) = 1/(1 + e^{-x}).$$
- La función ReLU, cuyo nombre es una abreviatura de Rectified Lineal Unit (Unidad Lineal Rectificada):
$$ReLU(x) = max(0, x).$$
- La función softplus, una versión fluida de la función ReLU:
$$softplus(x) = log(1 + e^{x}).$$
La derivada de la función softplus es la función sigmoidea.
- La función tanh:
$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}.$$

Figura 2.3: Gráfica de funciones de activación usadas comúnmente en Deep Learning



Fuente: (Russell and Norvig, 2021)

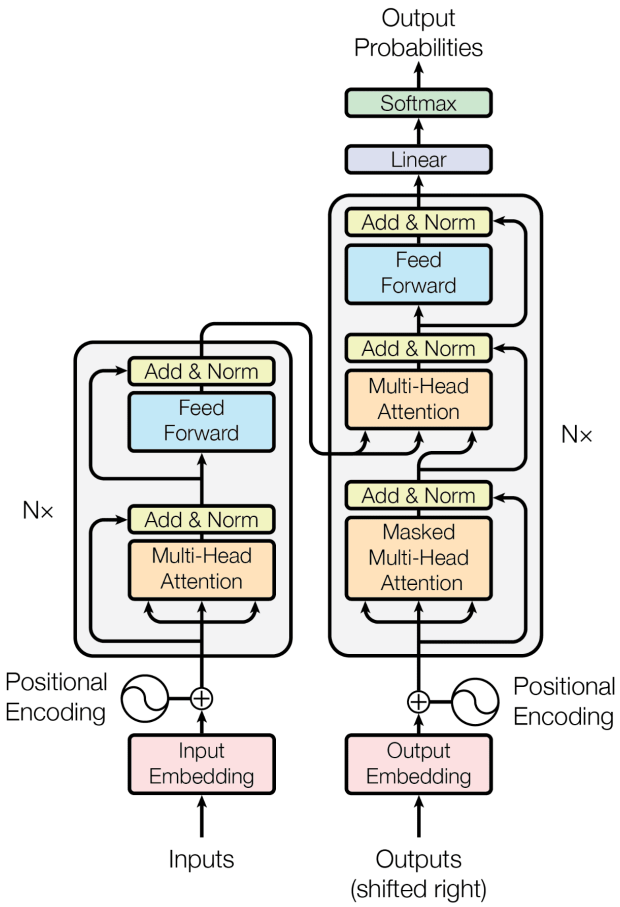
2.2.5. Transformers

El influyente artículo "Attention is all you need" (Vaswani et al., 2017) introdujo la arquitectura transformer, que utiliza un mecanismo de autoatención que puede modelar contexto a larga distancia sin una dependencia secuencial.

2.2.5.1. Arquitectura del Modelo Transformer

La mayoría de los modelos mas utilizados y eficaces de redes neuronales artificiales tienen una estructura de encoder-decoder. Aquí, el encoder mapea una secuencia de entrada de representaciones de símbolos $(x_1, ..., x_n)$ a una secuencia de representaciones continuas $z = (z_1, ..., z_n)$. Dado z, el decoder genera una secuencia de salida $(y_1, ..., y_m)$ de símbolos un elemento a la vez. En cada paso el modelo es autorregresivo, consumiendo los símbolos generados previamente como entrada adicional al generar el siguiente.(Vaswani et al., 2017) El modelo Transformer sigue esta arquitectura general utilizando autoatención apilada y conexión punto-punto de capas tanto para el encoder como para el decoder, como se muestra respectivamente en las mitades izquierda y derecha de la Figura 2.4.

Figura 2.4: Arquitectura del modelo Transformer

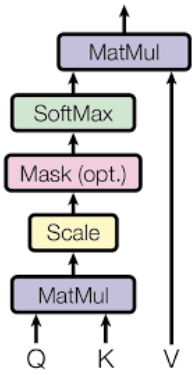


Fuente: (Vaswani et al., 2017)

2.2.5.2. Scaled Dot-Product Attention

Anteriormente, en los modelos sequence-to-sequence, la atención se aplicaba desde la RNN destino a la RNN origen. La autoatención extiende este mecanismo para que cada secuencia de los estados ocultos también se atiendan a sí mismos. Esto permite al modelo capturar adicionalmente el contexto de larga (y corta) distancia dentro de cada secuencia.

Figura 2.5: Representación gráfica de Scaled Dot-Product Attention



Fuente: (Vaswani et al., 2017)

La auto atención particularmente usada en los modelos transformer se llama "Scaled Dot-Product Attention", ver figura 2.5. Esta función que genera la matriz de salidas se define como:

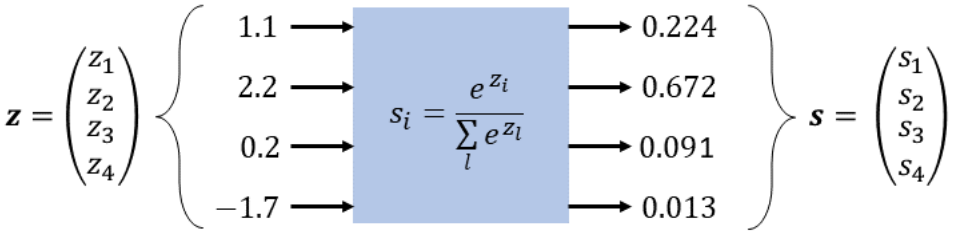
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

(Kurbiel, 2021) Donde:

- **Q**: Una matriz de consultas de dimensión d_k
- **K**: Una matriz de claves de dimensión d_k
- **V**: Una matriz de valores de dimensión d_k
- d_k: La dimensión de la matriz de claves. Este es un hiperparámetro elegido en tiempo de diseño.

La función softmax en las redes transformes tiene como objetivo tomar una serie de números reales arbitrarios (positivos y negativos) y convertirlos en números positivos que suman 1:

Figura 2.6: Ejemplo de funcion Softmax.



Fuente: (Kurbiel, 2021)

2.2.5.3. MultiHead Attention

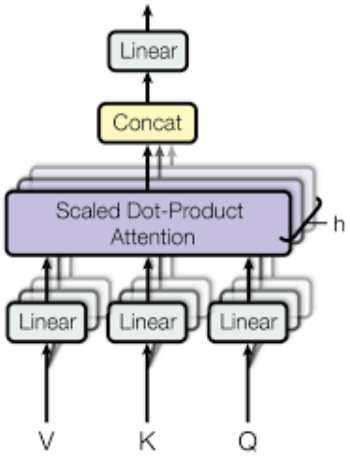
(Moreno, 2021), en lugar de realizar una única función de atención con claves, valores y consultas con dimensiones d_k, se encontró beneficioso proyectar linealmente las consultas, claves y valores h veces con diferentes proyecciones lineales de dimensiones d_k, d_k y d_v respectivamente. En cada una de estas versiones proyectadas de consultas, claves y valores se realiza la función de atención en paralelo, produciendo salidas de dimensión d_v. Estos son concatenados y nuevamente proyectados, dando como resultado los valores finales, como se muestra en la Figura 2.7.
La multi-head attention permite que el modelo atienda conjuntamente la información de diferentes subespacios en diferentes posiciones. Con una sola cabeza de atención, el promedio inhibe esto.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Donde las proyecciones son matrices de parámetros $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ y $W_i^Q \in \mathbb{R}^{hd_v \times d_{model}}$.

Figura 2.7: Representación gráfica de Multi-Head Attention

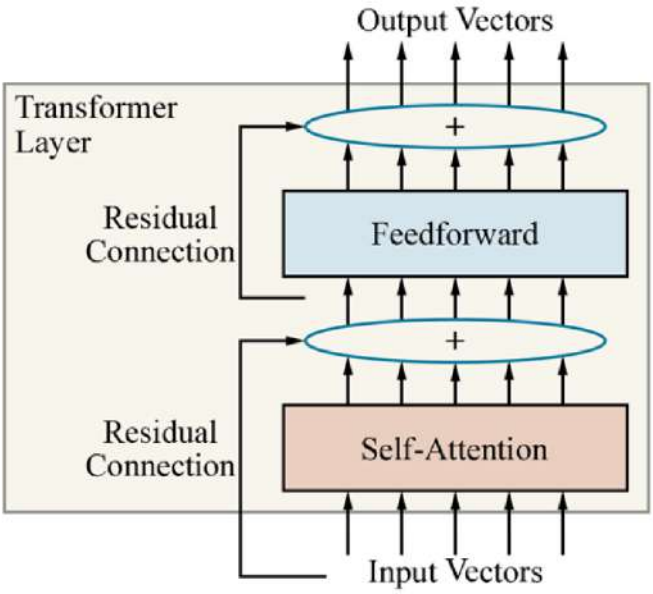


Fuente: (Vaswani et al., 2017)

2.2.5.4. Self-Attention De la autoatención a transformer

La autoatención es solo un componente del modelo transformer. Cada capa de transformer consta de varias subcapas. En cada capa del transformer, primero se aplica la autoatención. La salida del módulo de atención se alimenta a través de capas feedforward, donde las matrices de peso feedforward se aplican de forma independiente en cada posición. Una no lineal función de activación, típicamente ReLU, se aplica después de la primera capa de realimentación. Con el fin de abordar el problema potencial del gradiente de fuga, dos conexiones residuales (se añaden a la capa del transformer. Un transformer de una sola capa se muestra en la figura 2.8 En la práctica, los modelos de transformers suelen tener seis o más capas. Al igual que con los otros modelos que hemos aprendido, la salida de la capa se utiliza como entrada a la capa.) (Russell and Norvig, 2021)

Figura 2.8: Transformer de una sola capa consta de auto atención, una red de realimentación y conexiones residuales.

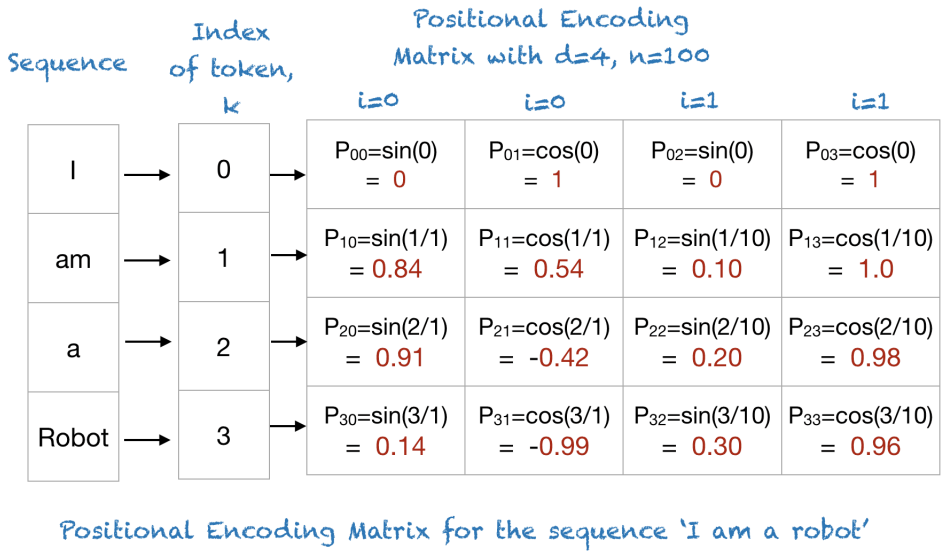


Fuente: (Russell and Norvig, 2021)

2.2.5.5. Positional Encoding

Describe la posición de una entidad en una secuencia para que a cada posición se le asigne una representación única. Hay muchas razones por las que un solo número, como el valor del índice, no se utiliza para representar la posición de un elemento en los modelos transformers. Para secuencias largas, los índices pueden crecer en magnitud. Si normaliza el valor del índice para que se encuentre entre 0 y 1, puede crear problemas para las secuencias de longitud variable, ya que se normalizarían de manera diferente.
Los transformers utilizan un esquema de positional encoding inteligente, donde cada índice se asigna a un vector. Por lo tanto, la salida de la capa de codificación posicional es una matriz, donde cada fila de la matriz representa un objeto codificado de la secuencia sumada con su información posicional. En la figura 2.9 se muestra un ejemplo de la matriz que codifica solo la información posicional. (Top Big Data, 2022)

Figura 2.9: Ejemplo de matriz de codificación posicional.



Fuente: (Brownlee, 2022)

2.2.6. Procesamiento de lenguaje natural

Es el campo de conocimiento de la Inteligencia Artificial que se ocupa de investigar la manera de comunicar las máquinas con las personas mediante el uso de lenguas naturales, como el español, el inglés o el chino.
La empresa mundial de software de analítica Statistical Analysis Systems Institute Inc afirma que el procesamiento de lenguaje natural toma elementos prestados de muchas disciplinas, incluyendo la ciencia de la computación y la lingüística computacional, en su afán por cerrar la brecha entre la comunicación humana y el entendimiento de las computadoras.
El procesamiento del lenguaje natural incluye diferentes técnicas para interpretar el lenguaje humano, que van desde los métodos estadísticos y del aprendizaje basado en máquina hasta los enfoques basados en reglas y algorítmicos. Necesitamos una amplia variedad de métodos porque los datos basados en texto y en voz varían ampliamente, al igual que las aplicaciones prácticas.
En términos generales, las tareas PLN dividen el lenguaje en piezas elementales más cortas,

intentan entender las relaciones entre las piezas y exploran cómo funcionan las piezas juntas para crear significado.(Statistical Analysis Systems Institute Inc, 2021)

2.2.6.1. Aplicaciones del PLN

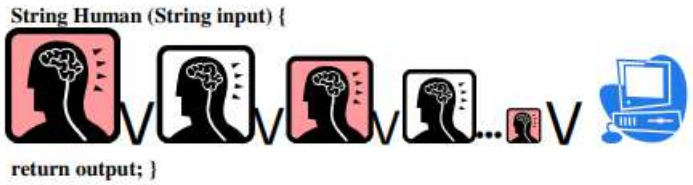
El PLN tiene aplicación en cualquier sector que disponga de grandes cantidades de información no estructurada:(IIC, 2021)

- Búsqueda avanzada de información
- Named-entity recognition (NER).
- Anonimización de documentos
- Detección de topics, similitudes o anomalías en los textos
- **Asistentes Inteligentes - Chatbots**
- Clasificación automática de documentos y mensajes
- Análisis de sentimiento y de la opinión(IIC, 2021)

2.2.7. Natural Language Understanding (NLU)

Natural Language Understanding (comprensión del lenguaje natural) es una sub-rama de NLP e IA, que utiliza software de computadora para comprender input en forma de oraciones usando texto o voz.
Según la Encyclopedia of Artificial Intelligence publicada en 1992, se considera que NLU es un problema AI-Complete (Shapiro, 2003).

Figura 2.10: Teoría de AI-Completeness



Shapiro (2003)

El objetivo de NLU (Natural Language Understanding) es extraer información estructurada de los mensajes de los usuarios. Esto generalmente incluye el intent (intención) del usuario y las entities (entidades) que contenga su mensaje. (RASA, 2022a).

2.2.8. Chatbot

Los bot de charla o bot conversacional (en inglés, chatbot), son aplicaciones software que surgen en los años 60, y que simulan mantener una conversación con una persona al proveer

respuestas automáticas, las cuales son previamente establecidas por un conjunto de expertos a entradas realizadas por el usuario. Estos bot, también conocidos como sistemas expertos, utilizan el razonamiento basado en casos (CBR: case base reasoning) (Troncoso, 2012). Los chatbots tienen su origen en 1966, cuando el profesor Joseph Weizenbaum creó el primer chatbot basado en inteligencia artificial llamado ELIZA, el cual actuaba como terapeuta (Weizenbaum, 1966). (Sansonnet et al., 2006) describen tres funciones principales que debe realizar un chatbot:

- Agente Racional: Para poder manejar las preguntas del usuario, el chatbot debe ser capaz de interactuar con una base de datos para llevar a cabo un razonamiento heurístico, de tal manera que el usuario juzga cuan competente es el chatbot.
- Agente personificado: El chatbot debe ser lo más natural y auténtico posible. Tiene que ser lo más parecido posible a una conversación humana, por lo que tiene que añadir personalidad.
- Agente de diálogo: Debe entender la solicitud del usuario. El chatbot debe recibir entradas de texto o voz que permitan al usuario ingresar solicitudes de ayuda, y este debe poder analizarlos utilizando herramientas adecuadas de PLN para responder adecuadamente.

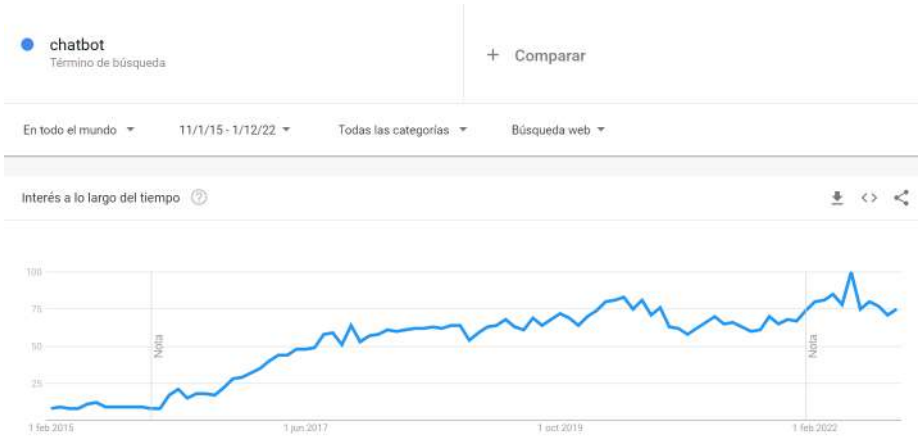
Figura 2.11: Descripción gráfica de un chatbot.

Fuente: (Sansonnet et al., 2006)

Según Google Trends, el interés de búsqueda de chatbots estuvo creciendo de manera ascendente durante los últimos 7 años.

La principal razón de su popularidad es que no es un software complicado y cualquiera lo puede usar.

Figura 2.12: Búsqueda Google Trends, últimos 7 años.



Fuente: Elaboración Propia.

2.2.8.1. Tipos de chatbots

Los chatbots de ITR no suelen utilizar Inteligencia Artificial. Funcionan de una manera muy sencilla, con menús y árboles de decisión predefinidos, a modo de navegación. Aunque este tipo de chatbot no sirve para cuestiones complejas es muy útil de cara a que el usuario o cliente pueda realizar una auto-gestión de algunos servicios sin necesidad de atención humana, como reportar incidencias.

Los chatbots de aprendizaje automático utilizan Inteligencia Artificial, y por lo tanto, su desarrollo y entrenamiento es más complejo. Son capaces de mantener conversaciones naturales y aprender de la experiencia, es decir utilizan las tecnologías de NLP (Natural Language Processing) y de ML (Machine Learning) y suponen un salto cualitativo muy importante, tanto en tecnología como en funcionalidad. Los chatbots de reconocimiento de palabras clave están en un punto intermedio. No utilizan Inteligencia Artificial, sino que funcionan identificando palabras clave en el entorno conversacional y proporcionando respuestas programadas ante esas palabras.

Por último, los chatbots cognitivos, también basados en la Inteligencia Artificial y el Machine Learning, van un paso más allá y son capaces no sólo de entender el lenguaje natural y las intenciones del usuario sino de interpretarlas dentro de un contexto mucho mayor. Esto contribuye a un uso mucho más amplio del chatbot, pues permite al usuario cambiar de contexto lingüístico en cualquier momento, en otras palabras cambiar de tema cuando él considere sin que el asistente 'pierda el hilo' de la conversación (Aunoa, 2020).

2.2.8.2. Clasificación principal de Chatbots

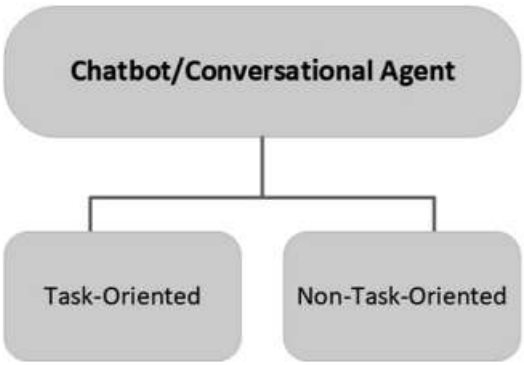
En los últimos años, el campo de los chatbots se ha ido popularizando con la llegada de nuevas tecnologías se van creando chatbots con diferentes propósitos específicos. Esto no permite realizar una clasificación exacta. Pero, en general, los chatbots se clasifican en dos categorías principales según los objetivos:

1. **Task-oriented:** Estos están diseñados para una tarea en específico y están configurados para tener conversaciones breves, generalmente dentro de un tema cerrado.

2. **Non-task-oriented:** Estos chatbots están diseñados para conversaciones prolongadas, configurados para imitar la característica conversacional no estructurada de interacción humano-humano, en lugar de centrarse en una tarea particular como reservar vuelos de avión; también son capaces de simular una conversación con una persona y parecen realizar charlas con fines de entretenimiento en temas abiertos. Las arquitecturas de chatbot no orientadas a tareas se dividen en 2, son las siguientes:

   - Generativos, como los modelos de secuencia, generan respuestas durante la conversación y pueden generar respuestas más adecuadas que no existen en el dataset.
   - Basado en recuperación, aprende a seleccionar respuestas fluidas para la conversación actual desde un dataset con algoritmos de selección. Se puede aplicar una variedad de heurísticas para elegir una respuesta adecuada. Este concepto se puede interpretar de una forma bastante simple al estar basada en reglas o puede ser tan compleja como usar clasificadores de Machine Learning.

Figura 2.13: Clasificación principal de Chatbots



Fuente: (Sianaki and Ababneh, 2019)

2.2.8.3. Ventajas de los chatbots

Los bots conversacionales tienen diversas ventajas, algunas de ellas son: proporcionar servicio las 24 horas del día, los 7 días de la semana; rapidez de respuesta, y gestión simultánea de múltiples clientes. Por ejemplo, la implantación de bots conversacionales en el ámbito empresarial resulta muy beneficiosa ya que ofrecen a su clientela una alternativa de servicio automatizado inteligente. Esto aumenta la participación del cliente, mejora la experiencia de la marca y brinda información útil para la empresa (Solutions, 2020).
Según Amazon Web Service, estos son los beneficios de utilizar chatbots:

1. **Eficiencia a través de la automatización:** Esto debido a que pueden optimizar y automatizar tareas repetitivas comunes a través de unas pocas solicitudes de forma verbal o texto, lo que reduce el tiempo de ejecución y mejora la eficiencia del negocio.

2. **Flexibilidad:** Se pueden construir para responder a la voz o al texto en el idioma nativo del usuario. Puede incorporar chatbots personalizados en los flujos de trabajo cotidianos para interactuar con la fuerza de trabajo de sus empleados o con los compromisos de los consumidores.

3. **Compromiso con el cliente:** Una experiencia de cliente ganadora puede ser un diferenciador significativo para un modelo de negocio. Los chatbots se pueden implementar en los canales en los que sus clientes potenciales ya están comprometidos, como Facebook Messenger, para que pueda comunicarse con ellos en entornos familiares para responder a sus solicitudes más rápido y cumplir sus expectativas (Amazon Web Service, 2021a).

2.2.8.4. Casos de uso comunes de chatbots

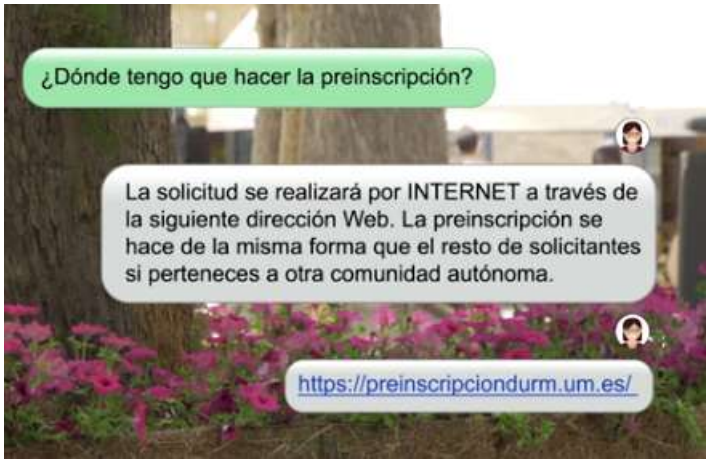
- **Productividad empresarial:** Permite que los chatbots se pueden integrar con los sistemas back-end de la empresa, como un CRM, un sistema de administración de inventario o un sistema de recursos humanos. Los chatbots se pueden crear para comprobar las cantidad de ventas, el rendimiento de marketing, el estado del inventario o la incorporación de empleados.
- **Asistentes personales:** Pueden simplificar y agilizar el proceso de las actividades personales cotidianas, como pedir zapatos nuevos o alimentos, reservar citas médicas o hacer reservas de viajes, desde su dispositivo móvil, navegador o plataforma de chat favorita.
- **Aplicaciones de call center:** Al utilizar un chatbot en una aplicación de centro de llamadas, sus clientes pueden realizar tareas como cambiar una contraseña, revisar el saldo en una cuenta o programar una cita, sin la necesidad de hablar con un agente. Los chatbots mantienen el contexto y administran el diálogo, ajustando dinámicamente las respuestas en función de la conversación (Amazon Web Service, 2021b).

2.2.8.5. Chatbot en el contexto de trámites administrativos universitario

- **Lola - Universidad de Murcia:** El chatbot que nace para guiar y ayudar a los futuros estudiantes de la UMU y de la Universidad Politécnica de Cartagena con todos sus trámites de preinscripción y matrícula. El chatbot aprende a medida que interactúa con los usuarios. Cuanto más cosas diferentes le pregunten, más será capaz de responder.

  Lola utiliza la tecnología Dialogflow de Google que es la que le permite ir aprendiendo de sus interacciones y la mejora continua. Solo el año pasado el SIU atendió 28.000 consultas relacionadas con el proceso de admisión a la Universidad, consultas que a partir de ahora podrán ser resueltas por Lola. El acceso a Lola está disponible a través de la web del SIU y mediante la aplicación DURM para móviles que utilizan los futuros estudiantes universitarios del distrito único de la Región de Murcia. (La verdad, 2018)

Figura 2.14: Chatbot Lola - Universidad de Murcia



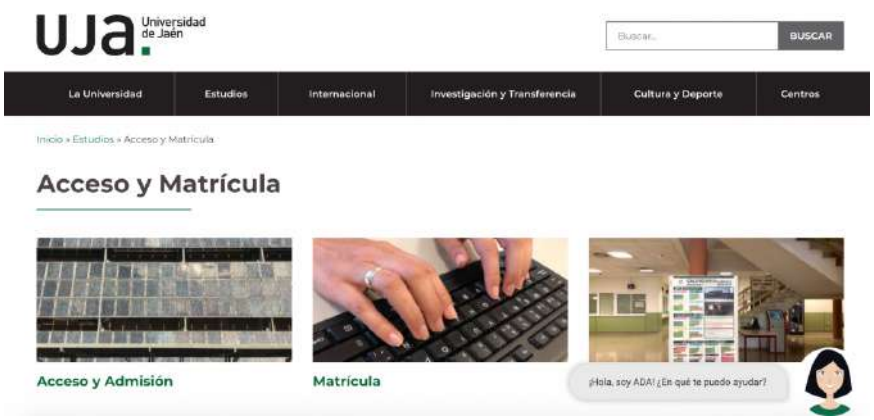
Fuente: (La verdad, 2018)

- **Ada - Universidad de Jaén:** Es el nuevo chatbot para dar cobertura a los alumnos que quieran ingresar en la Universidad. Su función es ofrecer soporte al nuevo alumnado de la Universidad de Jaén en ámbitos como:

  1. Pruebas Evaluación Bachillerato para Acceso a la Universidad.
  2. Acceso a la Universidad.
  3. Preinscripción.
  4. Grados de la Universidad.
  5. Matrícula en la Universidad.
  6. Reconocimiento de créditos.
  7. Dudas generales acerca del estudio en la Universidad.
  8. Información y contacto.

  Alrededor de 200 preguntas relacionadas con los estudios, acceso, selectividad y matricula hacen de ADA un instrumento de ayuda al alumnado que permitirá identificar con claridad las necesidades de información del mismo. ADA se convierte en una respuesta normalizada a la exigencia de minimizar la presencialidad de estudiantes y profesores en la era post-covid. (1MillionBot, 2020)

Figura 2.15: ADA - Universidad de Jaén



Fuente: (1MillionBot, 2020)

- **Isidra - Universidad de Alcalá:** Es un chatbot que utiliza inteligencia artificial, es decir, que aprende de cada conversación y perfecciona sus respuestas a partir de las preguntas que le planteamos. A partir de su puesta en marcha, seguirá mejorando para que al inicio del nuevo año académico, ofrezca mayor información, más completa y personalizada. El principal objetivo es acercar la Universidad a todos los interesados en la vida universitaria, bien porque forman parte de la comunidad, o porque están interesados en lo que hace y ofrece la universidad desde el punto de vista académico. Su lanzamiento surge de la necesidad, por parte de los usuarios, de acceder a información adicional e inmediata durante la pandemia.(UAH, 2020)

Figura 2.16: ISIDRA - Universidad de Alcalá

Fuente: (UAH, 2020)

2.2.8.6. Intents

Un intent representa una tarea o acción que el usuario desea realizar. Es la intención expresada detrás de el utterance de un usuario. En la mayoría de los casos, los intents se pueden identificar buscando verbos en los diálogos de los usuarios. Pero a veces se usa la oración completa para determinarlo. Por ejemplo, una aplicación de viajes tendría varias intenciones:
Hay otro tipo de intents llamados casual intents. Estos son en su mayoría los que inician y terminan la conversación como "hola","gracias", "adiós", etc. Los casual intents también pueden ser de afirmación o negación como "sí", "no", "no, gracias", "sí, claro", etc (Microsoft, 2022b).

Tabla 2.1: Ejemplo de intents

Aplicación de viaje (Intents)	**Ejemplos (Utterances)**
Reservar un vuelo	"Resérvame un vuelo a Río la semana que viene" "Llévame a Río el 24" "Necesito un boleto de avión el próximo domingo a Brasil"
Saludo	"Hola" "Buenos días"
Comprobar Clima	"¿Cómo está el clima en Boston?" "Muéstrame el pronóstico para este fin de semana"

Fuente: Elaboración Propia.

2.2.8.7. Entities

Los entities (entidades) son piezas estructuradas de información dentro de un mensaje de usuario, es la información proporcionada sobre el intent en un diálogo. Por ejemplo en la siguiente consulta "Quiero pedir una pizza pequeña", la intención del usuario es Ordenar-Pizza, mientras que la entidad sería Tamaño y en este caso "pequeña".
Además se puede tener varias entidades en una misma consulta. En el siguiente ejemplo: "Deseo comprar 3 boletos a Nueva York", "3", es una instancia de la entidad NroTickets y "New York.[es] una instancia de la entidad Destino.
Para decidir qué entidades se necesita extraer, se debe pensar en qué información necesita el asistente para responder correctamente las consultas de usuario. El usuario puede proporcionar información adicional que el chatbot no necesita, esta información no tiene que ser extraída como entities (Microsoft, 2022a).

Tabla 2.2: Ejemplo de entities

Utterance	**Intent**	**Entity**	**Tipo de Entity**
Hola, como estás?	Saludo	-	Ningún entity.
Quiero pedir una pizza pequeña	HacerPedido	'pizza', 'pequeña'	Entities 'Producto' y 'Tamaño'
Apaga la luz del dormitorio.	ApagarLuz	'dormitorio'	Entity 'Habitación'.
Comprar 3 pasajes a Nueva York	ComprarPasajes	'3', 'New York'	Entities 'Cantidad' y 'Destino'

Fuente: (Microsoft, 2022a)

2.2.8.8. Utterances

Los utterances son entradas de los usuarios que el chatbot necesita entender e interpretar. Como por ejemplo:

- "¿Cómo consigo una computadora?"

- "¿Dónde consigo una computadora?"
- "Quiero comprar una computadora, ¿cómo lo hago?"
- "¿Cuándo podré tener una computadora?"

Estos ejemplos tienen el mismo intent: comprar un equipo. Sin embargo, el entity computadora no es variado. Por lo que usar alternativas como laptop, PC, equipo o incluso solo máquina es mejor para generar variedad. Recopilar utterances que los usuarios ingresarán es muy importante para construir un buen chatbot. Para esto, se deben incluir utterances que significan lo mismo pero se construyen de varias maneras: (Microsoft, 2022c)

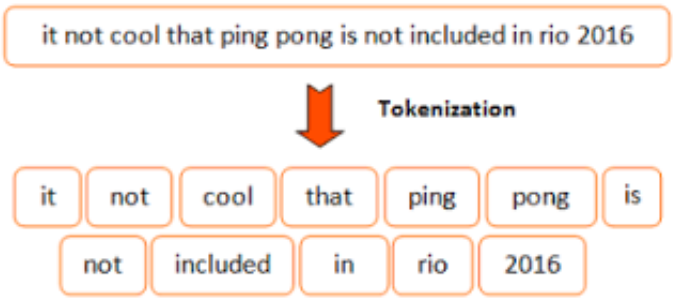
- Longitud del utterance: corta, media y larga
- Longitud de palabra y frase
- Ubicación de palabras: entity al principio, en medio y al final de la expresión
- Gramática
- Pluralización
- Derivación
- Elección de sustantivos y verbos
- Puntuación: usando gramática correcta e incorrecta

2.2.8.9. Tokenization

La tokenización es una forma de separar un fragmento de texto en unidades más pequeñas llamadas tokens. Aquí, los tokens pueden ser palabras, caracteres o subpalabras. Por lo tanto, la tokenización se puede clasificar en términos generales en 3 tipos: tokenización de palabras, de caracteres y de subpalabras (n-gramas).
La tokenización es una tarea común en el procesamiento del lenguaje natural (NLP). Es un paso fundamental tanto en los métodos tradicionales de NLP como Count Vectorizer, así como en las arquitecturas basadas en Advanced Deep Learning como los Transformers.
Un tokenizer es una pieza de código que realiza el proceso de tokenizar. Esto se debe hacer antes de que el texto sea procesado por algoritmos de machine learning, por lo que generalmente es el primer paso a seguir en un pipeline de NLU.
Por lo general, su output es una lista de palabras, puesto que en el idioma español así como en inglés se usa el espacio en blanco como separador.

Se debe tener en cuenta que los tokenizers no cambian el texto subyacente, solo separan el texto en tokens.
Para el idioma español es una tarea más sencilla ya que se puede separar las palabras por espacios y caracteres que no pertenecen al alfabeto. En el caso del inglés, se convierte en algo más complejo. Si se tiene la expresión don't el algoritmo tiene que identificar las locuciones do y n't.

Figura 2.17: Ejemplo de Tokenizer.

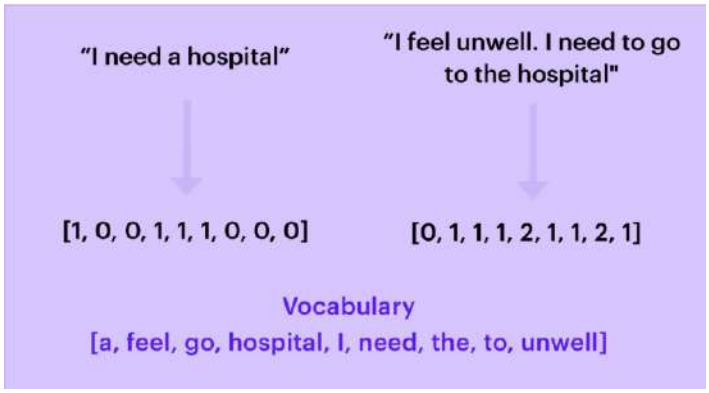


(Notebook Community, 2016)

2.2.8.10. Featurization

Un featurizer es un fragmento de código que transforma los datos de entrada sin procesar en una forma procesada adecuada para el aprendizaje automático. Los métodos de aprendizaje automático a menudo necesitan que los datos sean pre-procesados previamente. El siguiente diagrama muestra un ejemplo de un featurizer que cuenta la frecuencia con la que ciertas palabras de los datos de entrenamiento aparecen en un mensaje:

Figura 2.18: Ejemplo de Featurizer.

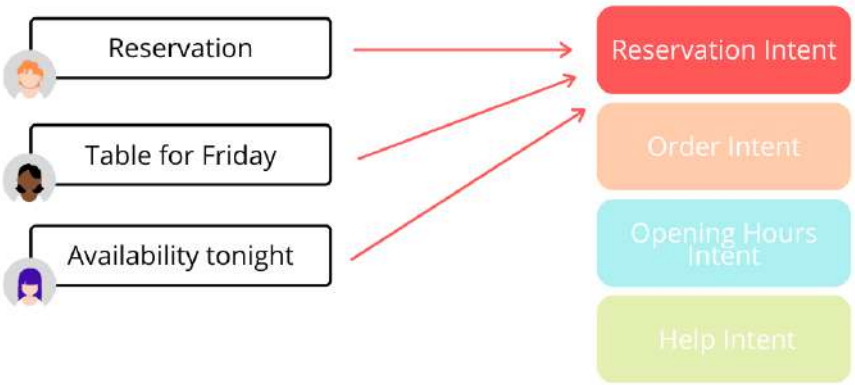


(RASA, 2021a)

2.2.8.11. Intent Classification

La clasificación de intenciones es la categorización automatizada de datos de texto en función de los objetivos del cliente.
En esencia, se analiza automáticamente los textos de entrada y se categoriza en intenciones como comprar, vender, cancelar suscripción, reservar, etc. Esto es útil para comprender lo que está realmente detrás de las consultas realizadas, puesto que cada interacción con el cliente y/o usuario tiene un propósito, un objetivo o una intención.
La clasificación de intenciones utiliza modelos de aprendizaje automático y procesamiento de lenguaje natural para asociar automáticamente palabras o expresiones con una intención en particular. Estos modelos reciben el nombre de intent classifiers, y lo que realizan es tomar el output del featurizer como input y procesarlo para hacer una predicción sobre qué intención coincide con el mensaje del usuario.

Figura 2.19: Ejemplo de Clasificación de Intenciones.



(RASA, 2021b)

Sin embargo, los classifiers deben entrenarse primero con ejemplos de texto, también conocidos como datos de entrenamiento. Cuantos más información se use para entrenar el modelo, más inteligente será el clasificador, ya que tiene más información de la que aprender.
Clasificar las intenciones de los usuarios de manera correcta permite que las empresas ofrezcan un mejor servicio, especialmente en áreas como la atención al cliente y ventas online; por ende, crear un buen modelo de classifier es una pieza clave en el NLU pipeline.

2.2.8.12. Entity Extraction

Un named entity (entidad nombrada) es, en términos generales, cualquier cosa a la que puede ser referida con un nombre propio: una persona, un lugar, una organización. La tarea del reconocimiento de entidades nombradas (Named Entity Recognition o NER) es encontrar tramos de texto que constituyen nombres propios y etiquetar el tipo de entidad nombrada.
Los tipos de entidades más comunes son: PER (persona), LOC (ubicación), ORG (organización) o GPE (entidad geopolítica). Sin embargo, el término entidad nombrada es comúnmente extendido para incluir cosas que no son entidades en sí, incluyendo fechas, tiempos, y otro tipo de expresiones temporales, e incluso expresiones numéricas como precios (Jurafsky and Martin, 2020).
En el ejemplo a continuación, el sistema NER encontró ocho entidades que han sido clasificadas en cuatro categorías diferentes: persona, organización, fecha y nacionalidad o grupo religioso o político (NORP).

Figura 2.20: Ejemplo de Named Entity Recognition.

(Chavan, 2020)

Los dos modelos de extracción de entidades más usados comúnmente son los siguientes:

- Deep Neural Network Extractors: también conocidos como "extractores estadísticos", se utilizan principalmente para identificar entidades que se pueden especificar; tal es el caso de personas, lugares y organizaciones. Por ejemplo, Dakota puede referirse a una persona o un lugar. Mediante el uso de estos modelos, los extractores pueden categorizar con precisión una entidad.
- Pattern Matching Extractors: estos se utilizan para reconocer expresiones comunes como fechas, horas, localizadores de recursos uniformes (URL), direcciones de correo electrónico, números de teléfono, números de tarjetas de crédito y etiquetas de redes sociales. Por ejemplo, si se tiene una cadena que cuenta con un dominio de nivel superior (TLD) y una subcadena de caracteres entre el símbolo @ y el punto, esta se identificaría como una dirección de correo electrónico.

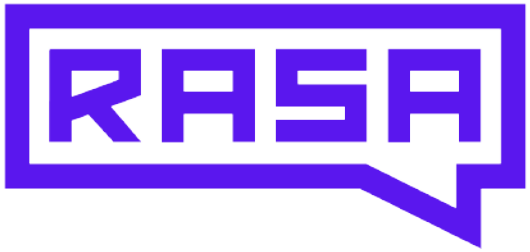
Figura 2.21: Ejemplo de Pattern Matching Extractor.

(Techslang, 2020)

2.2.9. RASA

Rasa es un marco de trabajo de código abierto para crear chatbots de IA conversacional. Proporciona herramientas y bibliotecas para desarrollar e implantar chatbots basados en texto e impulsados por IA capaces de mantener conversaciones en lenguaje natural con los usuarios. Rasa permite a los desarrolladores crear experiencias de chatbot interactivas y dinámicas que pueden comprender las intenciones del usuario, extraer entidades y proporcionar respuestas significativas (RASA, 2023).

Figura 2.22: RASA



(RASA, 2023)

Capítulo 3

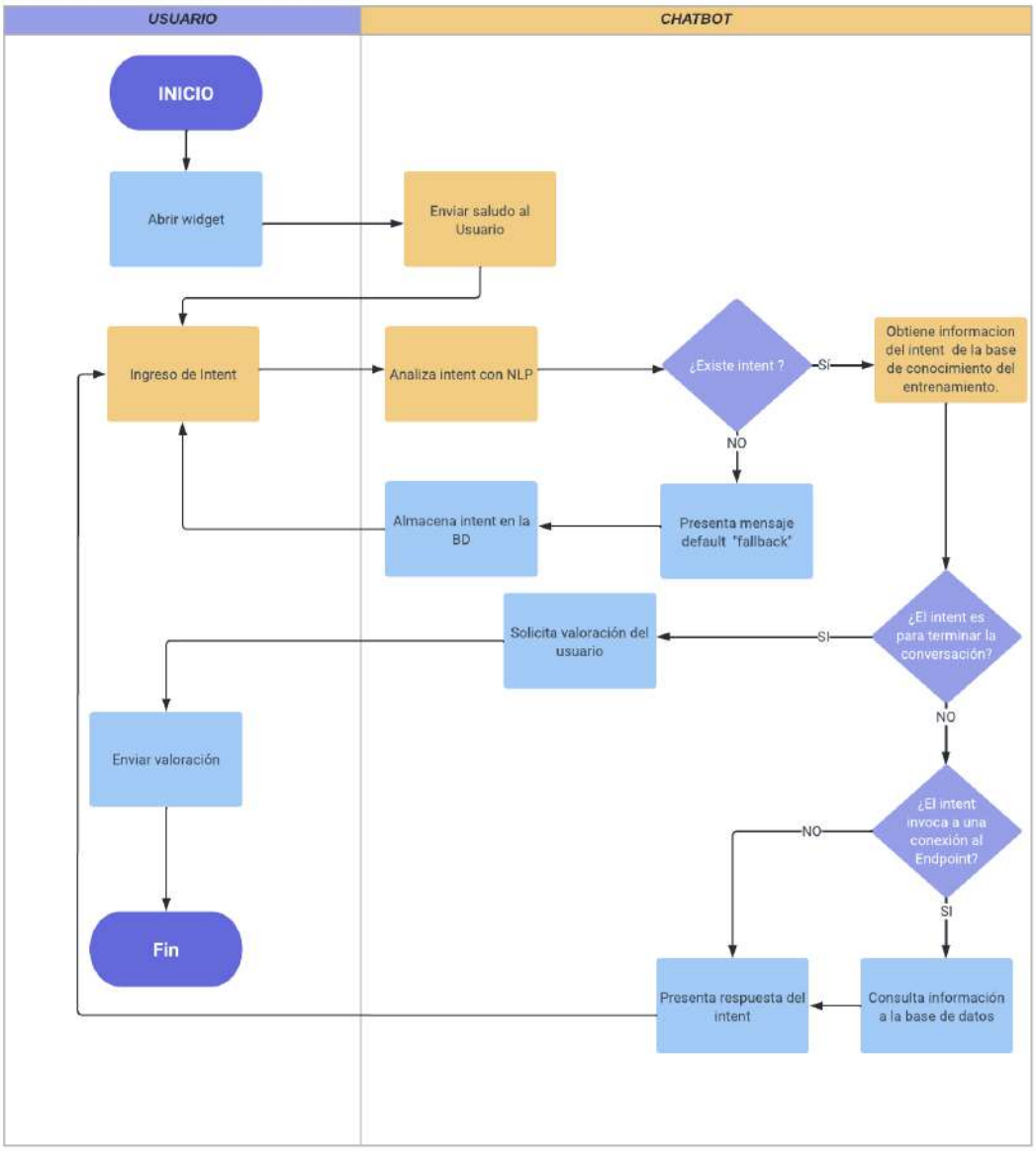
Desarrollo del proyecto

3.1. Análisis del modelo

La propuesta fundamental fue que el chatbot sea capaz de responder a preguntas especificas bajo el criterio de como funciona el actual chatbot de la plataforma de trámite documentario PLADDES (Sistema de Trámite Documentario Virtual), pero de una forma más óptima y natural, ya que los usuarios nuevos o no frecuentes tienen consultas que requieren ser atendidas en un lapso de tiempo corto. Además, se implementaron las características que hicieron que el chatbot obtenga personalidad y naturalidad durante el flujo de sus conversaciones.
Seguidamente mencionamos el alcance del chatbot, las funcionalidades y también se muestra el diagrama de flujo del chatbot en la figura 3.1:

- Es capaz de responder de manera natural a mensajes básicos como saludo, despedida.
- En caso de no haber entendido el mensaje del usuario, el chatbot responde indicando que no entendió la consulta pero que la tendrá en cuenta a futuro para seguir aprendiendo.
- Es capaz de comprender si se le está consultando por los requisitos, monto o descripción de X trámite y responder correctamente.
- Si se requiere información sobre trámites de un usuario, el bot es capaz de pedir la información necesaria para posteriormente responder correctamente.
- Tiene la capacidad de entender mensajes con errores ortográficos y/o abreviaciones informales de palabras.
- Es capaz de preguntar al usuario si su asistencia virtual fue de ayuda (feedback).

Figura 3.1: Diagrama de flujo del chatbot



Fuente: Elaboración Propia.

Habiendo mencionado el alcance y funcionalidades del modelo, para lograr el objetivo, se optó por utilizar como base el software de código abierto Rasa de Python, ya que está orientado específicamente a la construcción de agentes conversacionales basados en inteligencia artificial y así lograr los resultados correctos.

3.1.1. Conceptualización del modelo

- ¿Qué no debe hacer explícitamente el chatbot?
  El chatbot está diseñado para atender a un público general y, por lo tanto, no debe abordar casos demasiado específicos de los usuarios, se centrará en brindar explicaciones generales y no personalizadas para poder abarcar las necesidades de un amplio espectro de usuarios. Su objetivo principal es proporcionar información y respuestas claras que sean aplicables a la mayoría de los usuarios, manteniendo así un enfoque amplio y general.

- ¿El chatbot va a interactuar con otro software?
  El chatbot interactuará con los usuarios a través de una interfaz de tipo web, además que se conectará al Api Rest del Pladdes y SQL Server.

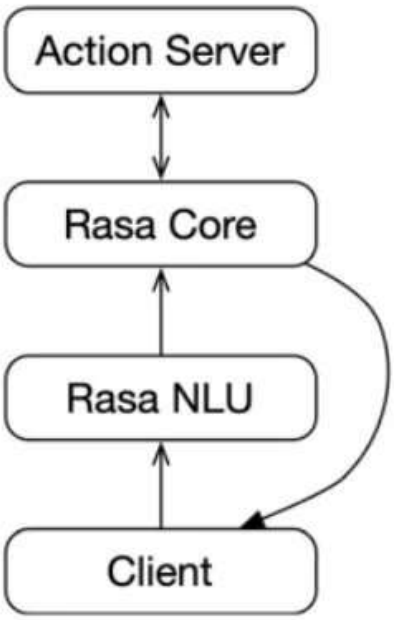
- ¿Cuál es el propósito por el que se esta implementando el chatbot?
  El propósito del desarrollo de este chatbot es solucionar la problemática de brindar respuestas a las preguntas más comunes que los usuarios tienen al momento de ingresar a la plataforma web de trámite documentario. Con este fin, el chatbot proporcionará respuestas a consultas generales relacionadas a los trámites administrativos de la universidad.

- ¿Que papel desempeña el chatbot?
  El chatbot desempeñará el papel de un personal administrativo, ofreciendo respuestas a las consultas planteadas por los usuarios. Las preguntas que los usuarios dirigirán al chatbot se centrarán en los procesos o trámites administrativos de la universidad. Además, se espera que el chatbot mantenga un tono cercano y amigable en su forma de expresarse, evitando un lenguaje excesivamente informal. El nombre seleccionado para este chatbot es **HatunBot** .

3.2. Estructura de un modelo en Rasa

Es muy importante comprender que los chatbots en Rasa constan de dos modelos: un modelo NLU y un modelo Core. El modelo NLU es responsable de reconocer las intenciones de las consultas del usuario y de extraer las entidades presentes en ellas. El modelo Core es responsable de administrar el flujo de la conversación: recordar las entidades, generar la respuesta y/o acción correcta, comprender cuándo es momento de esperar el siguiente mensaje del usuario, etc.
Aunque el Core decide cuál es la acción correcta a realizar, en realidad no la ejecuta. Hay un servidor independiente que se encarga de ejecutar las acciones. Cuando el chatbot predice una acción personalizada, el servidor del Core envía una solicitud POST al servidor de Acciones con una carga json que incluye el nombre de la acción prevista, el ID de la conversación, el contenido del Tracker y el contenido del dominio. Cuando el servidor de acciones termina de ejecutar la acción recibida, devuelve un payload en formato json de respuestas y eventos. Luego, el servidor del Core devuelve las respuestas al usuario y agrega los eventos al Tracker.3.2

Figura 3.2: Estructura de un modelo en Rasa.

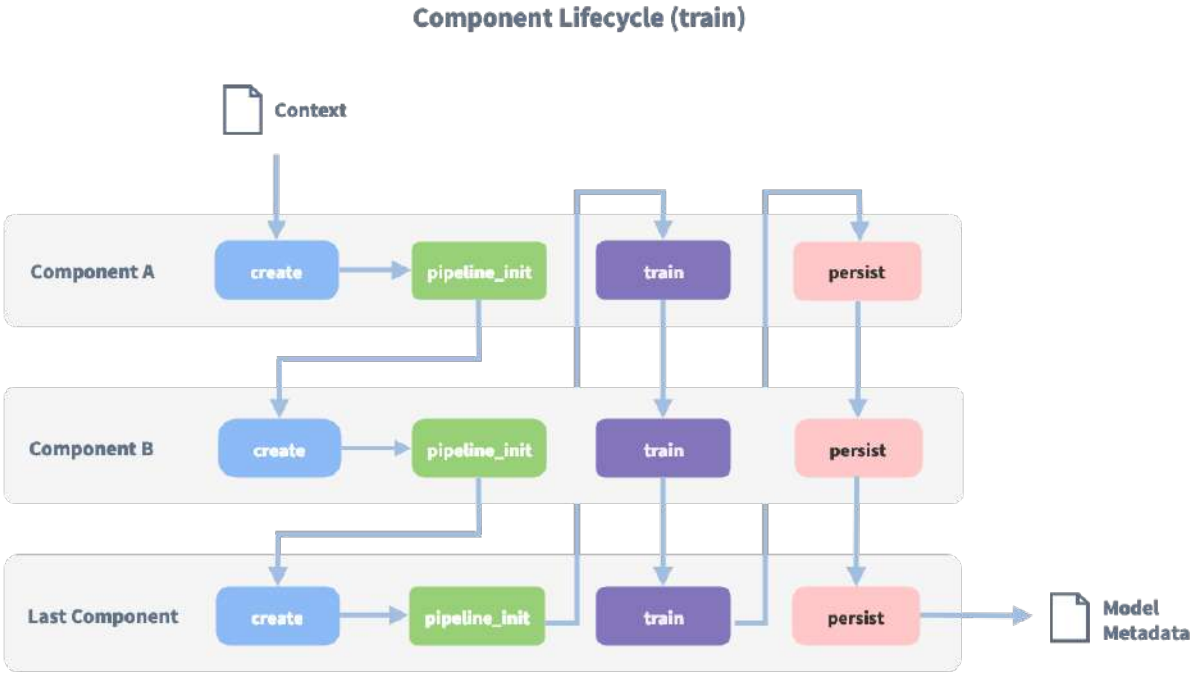


Fuente: (Omdena, 2022)

3.2.1. Arquitectura Rasa NLU

La arquitectura del NLU se basa en un pipeline(tubería) de procesamiento. Este pipeline define las etapas de procesamiento por las que tendrán que pasar los mensajes de usuario entrantes hasta que se produzca el output, en este caso una intención y opcionalmente uno, varios o ninguna entidad.
En la Figura 3.4 se muestran todos los componentes de la arquitectura de Rasa NLU. Debido a que esta es muy flexible, se puede configurar para cualquier tipo de chatbot dependiendo de su objetivo. Técnicamente, se realizan los procesos secuencialmente, donde los más comunes son los ya descritos tokenization, featurization, intent classification y entity extraction. Cabe recalcar que se pueden utilizar distintas operaciones y no necesariamente utilizar solo uno en cada paso. Para los 2 primeros pasos utilizamos algoritmos de NLU como lo son WhiteSpaceTokenizer (separador por espacios en blanco), CountVector de palabras y de caracteres, ya que estos pasos son de preprocesamiento de información. La figura 3.3 muestra el orden de llamada durante el entrenamiento de este pipeline:

Figura 3.3: Ciclo de vida de los componentes



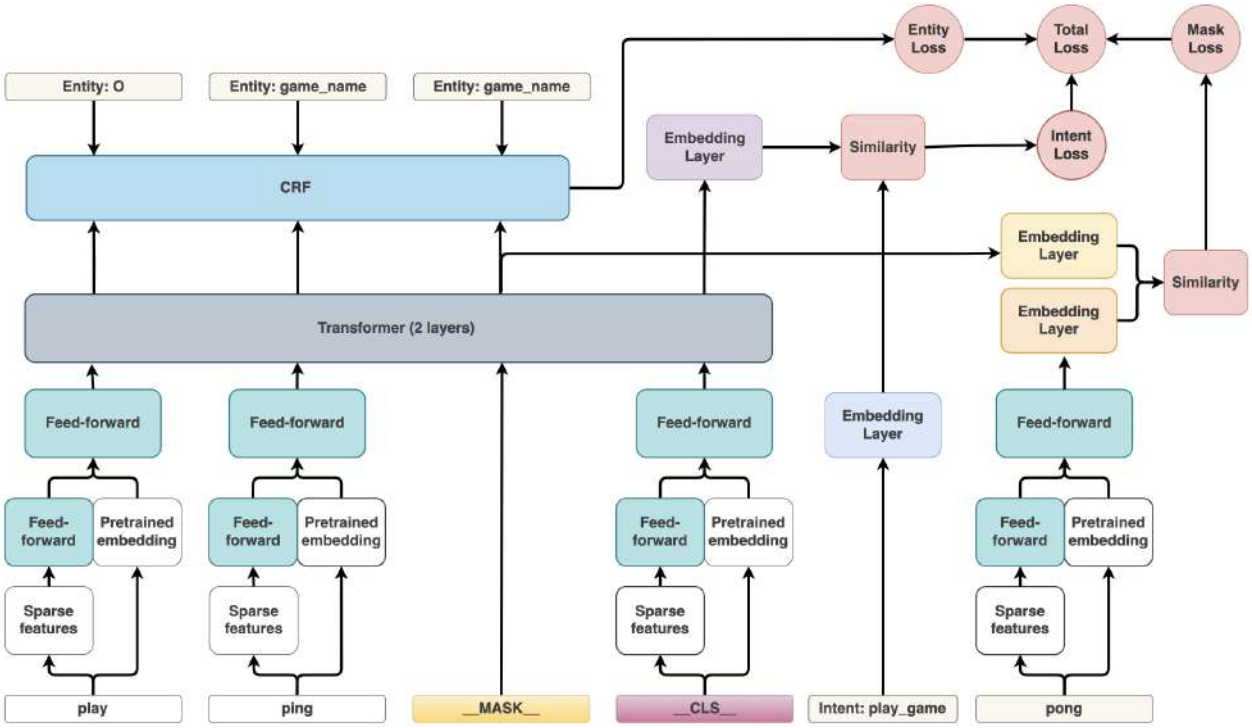
Fuente: (RASA, 2022b)

En el componente Transformer se utilizó uno llamado DIET (Dual Intent Entity Transfromer), debido a que este es muy flexible, por lo que se puede configurar para cualquier tipo de chatbot dependiendo de su objetivo. Por lo tanto, el modelo final entrenado para el pipeline seleccionado se muestra en la figura 3.4. Esta arquitectura se basa en transformers, permite realizar simultáneamente la clasificación de intenciones y la extracción de entidades. Tiene componentes diferentes que se pueden modificar, lo que la convierte en una arquitectura muy flexible. Los modelos de lenguaje en su mayoría son computacionalmente exigentes en cuanto a potencia de cálculo, porque se basan esencialmente en RNN. DIET se diferencia en los siguientes aspectos:

- Aprende 6 veces más rápido que dicho modelos.
- Los resultados son similares a modelos pre-entrenados en relación a rendimiento y precisión.
- Arquitectura modularizada que proporciona a los desarrolladores una mayor flexibilidad.

DIET usa un modelo de secuencia que considera el orden de las palabras, lo que le permite lograr un rendimiento superior. Además, destaca por ser una estructura más compacta y cuenta con una arquitectura modular plug and play, lo que facilita su implementación. Por ejemplo, es capaz de llevar a cabo tanto la clasificación de intenciones como la extracción de entidades, o bien, podríamos configurarlo para llevar a cabo solo una de estas funciones.

Figura 3.4: Arquitectura Rasa NLU



Fuente: (Botfront, 2020)

En nuestro caso, al ser un chatbot personalizado y con tareas específicas, en la tabla 3.1 elegimos la arquitectura con los siguientes hiper-parámetros:

Tabla 3.1: Configuración de Hiper-parámetros de modelo Rasa NLU

Hiper-parámetro	**Arquitectura**
Epochs	150
Masked language model	False
Embedding dimensions	30
Transformer layers	2
Transformer size	256
Attention heads	4
Hidden layers	0
Attention drop rate	0.1

Fuente: Elaboración Propia.

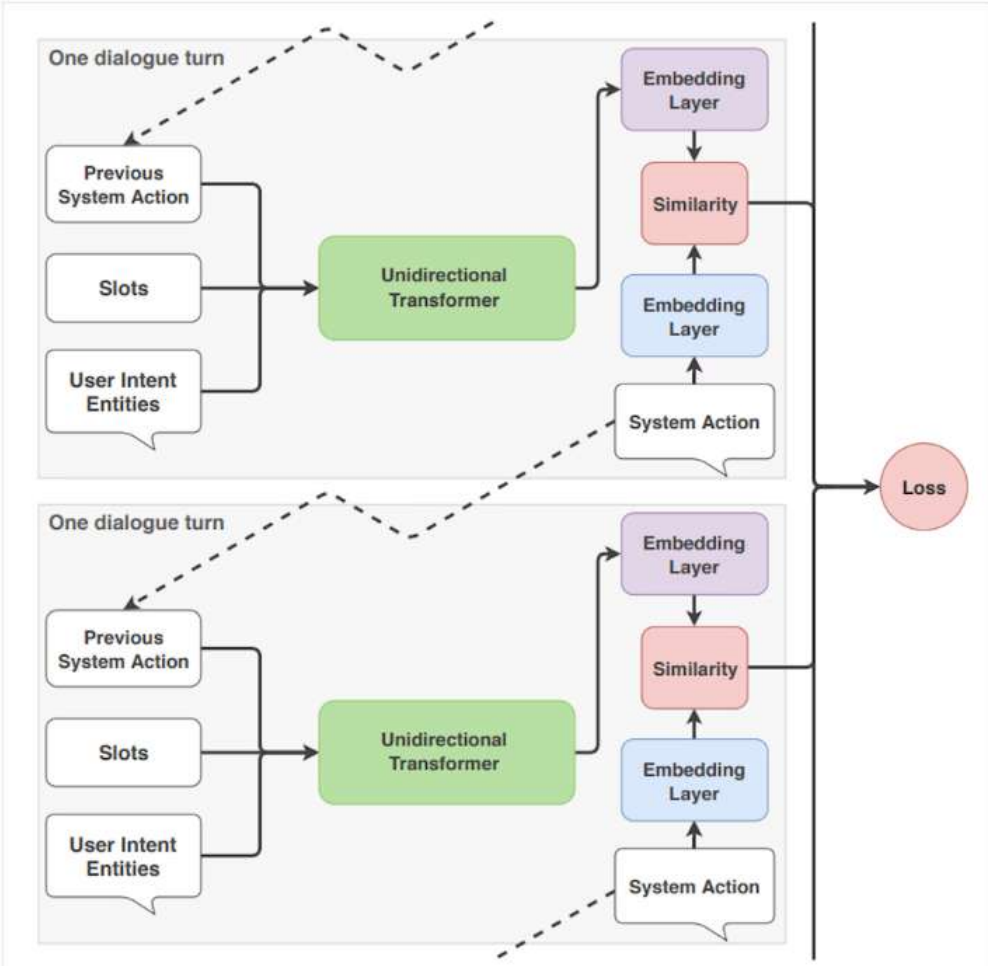
3.2.2. Arquitectura Rasa Core

La arquitectura del Core se basa en el uso de policies (políticas) para decidir qué acción tomar en cada paso de una conversación. Estas policies se pueden usar en conjunto, donde cada una genera su propio output y la que prediga con la mayor confianza (valores entre 0 y 1) decide la próxima acción a ejecutar.
La arquitectura de la política TED (ver figura 3.5) comprende los siguientes pasos:

- Concatenar los vectores de
  - Entrada del usuario (intents y entities)
  - Action previa que realizó el modelo
  - Slots (entities que se guardan durante la conversación)

  para cada intérvalo de tiempo t en un solo vector que será el input $\mathbf{t}$
- Pasar el input por el Transformer
- Pasar el output del Transfromer por una dense layer para obtener un vector "dialogue"de tamaño distinto
- Pasar cada Action a_t por una dense layer que genere un vector "actions" del mismo tamaño del paso anterior
- Calcular la similitud entre el vector dialogue y el vector actions.

Figura 3.5: Arquitectura TED Policy



Fuente: (White, 2020)

Para nuestro modelo, utilizamos una arquitectura con los siguientes hiper-parámetros.

Tabla 3.2: Configuración de Hiper-parámetros de modelo Rasa Core

Hiper-parámetro	**Arquitectura**
Epochs	150
Max History	8
Transformer layers	1
Transformer size	256

Fuente: Elaboración Propia.

3.3. Implementación

Como se mencionó anteriormente el proyecto se ha dividido en 4 partes a lo largo de la investigación, siendo las 3 primeras partes la implementación del modelo descrito en los respectivos capítulos y la última los resultados obtenidos.
En esta sección se muestra cómo se implementó el modelo para las tareas del chatbot, el conjunto de datos que se usó y porque, como se entrenó el modelo, también se explicará como se probó el modelo, cuál fue el criterio utilizado para la evaluación.
En la Figura 3.8 se muestra el diseño de la arquitectura propuesta para el prototipo, está compuesta por el frontend y el backend. El frontend esta compuesto por la interfaz del chatbot y la consulta a la API, para luego mostrar la respuesta de la API. El backend comprende la recepción de los datos capturados por el chatbot para procesarla de acuerdo a la configuración de entrada del modelo, este modelo procesa y luego se devuelve una respuesta procesada.
Todo el código para la implementación del modelo base se escribió en Python, también consta de una interfaz web, este está disponible en el apéndice C .

3.3.1. Construcción del modelo cognitivo

Las tecnologías cognitivas actuales tienen la capacidad de extraer conocimiento a través del reconocimiento de relaciones que pueden hallarse contenidas de forma explícita (datos estructurados) (de Águeda, 2020). El proceso que caracteriza la construcción de nuestro modelo cognitivo es el siguiente:

- Generar una intención según las intenciones de los usuarios.
- Añadir entre 20 a 30 expresiones de ejemplo adecuadas en la entrada de los usuarios.
- Clasificar las expresiones de ejemplo.
- Separar los datos en subentidades.
- Añadir particularidad a las subentidades.
- Añadir particularidad a las intenciones.

3.3.2. Entrenamiento del NLU

Puesto que el asistente tiene un dominio específico y personalizado, se creó un dataset de entrenamiento propio, donde se recabaron aproximadamente 300 utterances. Dichas frases representan las dudas mas comunes y frecuentes que puede tener un usuario, escritas de distintas formas y utilizando diferentes palabras, esto hace que el bot tenga un vocabulario más amplio y variado, y por consiguiente prediga con mayor precisión los intents y entities respectivos.
Así mismo, se incluyeron frases básicas de charlas triviales como lo son saludos, despedidas, afirmaciones y negaciones para que el asistente pueda seguir con un flujo de conversación simple de manera natural.

Tabla 3.3: Detalles del entrenamiento

Intención de la aplicación	**Expresiones de ejemplo**
Saludo	"Hola" "Buenos dias"
Plazo de atención	"Cuanto demora un trámite?" "Demora mucho los trámites?"
Consultar trámites	"Que tramites tengo?" "Quiero verificar mis trámites"
Solicitud online	"Puedo hacer un trámite en línea?" "¿Cómo puedo hacer un trámite en línea?"
Horario de atención	"A qué hora puedo realizar mi trámite?" "A que hora está abierto el sistema?"
Consultar descripción	"Qué es el pase para escolares" "Qué es el pase para escolares?"
Consultar monto	"Quiero pagar mis cursos dirigidos en Maestría o Doctorado" "Cuanto esta el duplicado de Acta de Evaluación?"
Consultar requisitos	"Que se necesita para sacar la constancia de estudios?" "Que me piden para el reinicio de estudios?"
Consultar pago	"Donde pago?" "Como pago?"
Ubicación	"Dónde está la UNSAAC?" "Como llego a la UNSAAC?"

Fuente: Elaboración Propia.

3.3.3. Entrenamiento del Core

De igual modo que en el entrenamiento del NLU, para el Core se utilizaron stories y rules propias y personalizadas. Las stories son representaciones de conversaciones entre un usuario y un chatbot, donde por la parte del usuario los mensajes se representan con intents y opcionalmente entities, mientras que por la parte del bot con actions que responden a estos intents. En nuestro proyecto, basamos la mayoría de información de entrenamiento en stories, puesto que el bot puede predecir donde aplicar attention y así no seguir un flujo predeterminado de conversación.
Por otra parte, las rules son conversaciones que se cumplen bajo ciertas condiciones, por lo que están más orientados a asistentes basados en reglas. Las únicas reglas utilizadas en el proyecto fueron que el bot responda que es un chatbot cada vez que alguien haga una consulta con ese intent, y que se despida cuando alguien se despida de el.

3.3.3.1. Actions

Después de cada mensaje de usuario, el modelo Core predice una acción que el asistente debe realizar a continuación, estas pueden ser de diferentes clases. Las que utilizamos en el proyecto fueron:

- Responses (respuestas) que el asistente envía al usuario para consultas o mensajes simples, donde se requiere enviar información general.(Ver tabla 3.4)

Tabla 3.4: Ejemplos de responses del chatbot

Utterance	Response
hola buenos dias	Hola!, soy un chatbot y estoy aquí para ayudarte en lo que necesites.
a qué hora puedo realizar mi trámite?	El horario de atención es de 8:00 a.m. - 1:00 p.m. y de 3 p.m. - 5 p.m.
adios	Adiós, que tengas un buen día!

Fuente: Elaboración Propia.

- Custom Actions (acciones personalizadas) que pueden ejecutar código, se usa para hacer una llamada API o para consultar una base de datos. (Ver tabla 3.5)

Tabla 3.5: Custom Actions implementadas

Action	Descripción
ActionConsultarDescripcionTramite	Retorna descripción a detalle de un trámite “X”
ActionConsultarMontoTramite	Retorna el monto de un trámite “X”
ActionConsultarRequisitosTramite	Retorna los requisitos de un trámite “X”
ActionEstadoTramite	Se listan los trámites de un usuario

Fuente: Elaboración Propia.

3.3.3.2. Custom Actions

Para guardar y extraer la información que solicita el servidor de actions, se consideraron 2 endpoints que corresponden a lo siguiente:

- Una base de datos constituida por 2 tablas: la primera guarda el nombre de un trámite, así como su descripción, requisitos y monto respectivos recopilados del TUPA 2021 de la UNSAAC; mientras que la segunda guarda todas las consultas que no pudo resolver (fallback) para poder agregarlas posteriormente al dataset de entrenamiento.

Tabla 3.6: Detalles tabla - base de datos

CAMPO	DETALLES
Nombre	Nombre del trámite según TUPA.
Descripcion	Descripción del trámite.
Requisitos	Requisitos del trámite.
Monto	Monto del trámite.

Fuente: Elaboración Propia.

- En el segundo endpoint el chatbot es capaz de interceptar el tráfico de red del PLADDES, lo cual permite analizar, aceptar o rechazar todas las solicitudes y respuestas de la aplicación.

Tabla 3.7: Detalles Endpoint Pladdes

Método	Url	Parametros	Descripcion
POST	https://tramite.unsaac.edu.pe/ tramite/buscar_expediente_x_codigo	anio codigo	Lista los trámites realizados por un estudiante

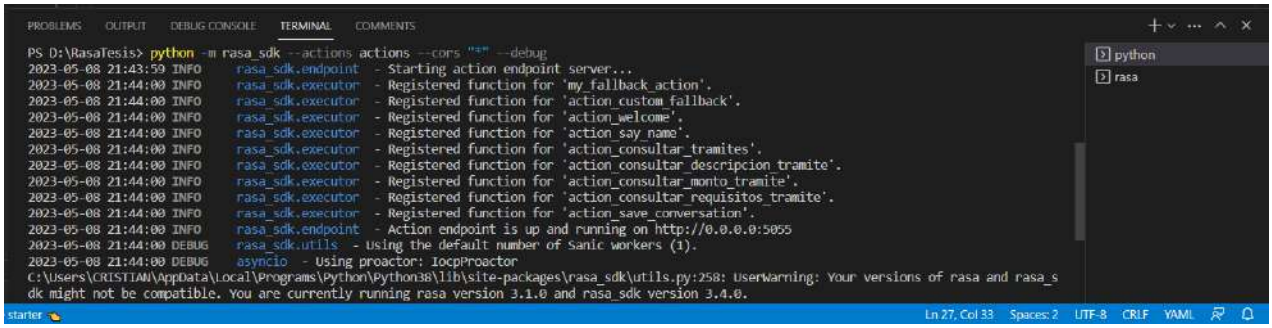
Fuente: Elaboración Propia.

3.3.4. Ejecución y Correcciones del Chatbot

Después de haber terminado el entrenamiento del chatbot y tener el modelo generado, seguidamente tendremos que alistar el servidor de acciones del SDK de RASA con el modelo entrenado en la sección 3.4.1 y 3.4.2. Para ejecutar el chatbot como una API y poder realizar peticiones POST, debemos interactuar con 2 terminales distintas en las que tendremos que ejecutar los siguientes comandos:

- Ejecución 1(Inicia servidor del SDK): Ya que haremos uso de 2 endpoints y es necesario acceder a ellas. Ejecutamos el siguiente comando
  python -m rasa_sdk - -actions actions - -cors "*" - - debug

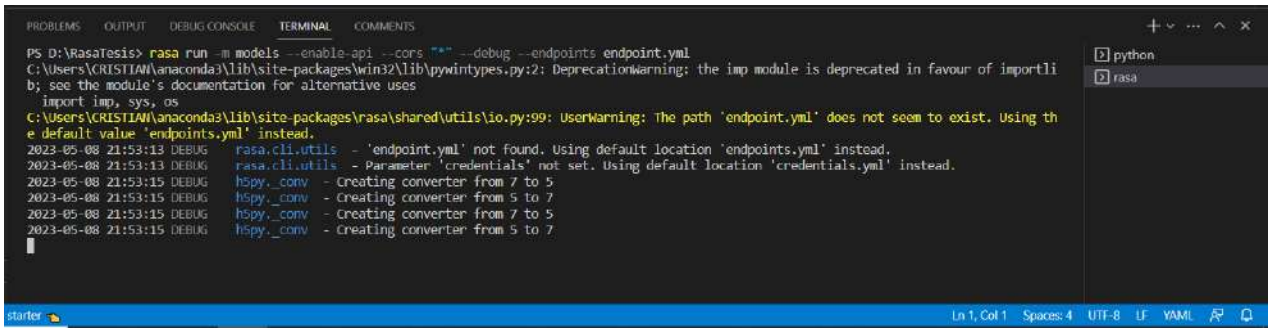
Figura 3.6: Comando para arrancar el servidor de acciones



Fuente: Elaboración Propia.

- Ejecución 2(Inicia el servidor de Rasa con el modelo previamente entrenado): Este servidor se encarga de recibir las llamadas HTTP por parte del canal que procesa el modelo. Ejecutamos el siguiente comando
  rasa run -m models - -enable-api - -cors "*" - - debug - - endpoints endpoint.yml

Figura 3.7: Comando para arrancar el servidor principal de Rasa



Fuente: Elaboración Propia.

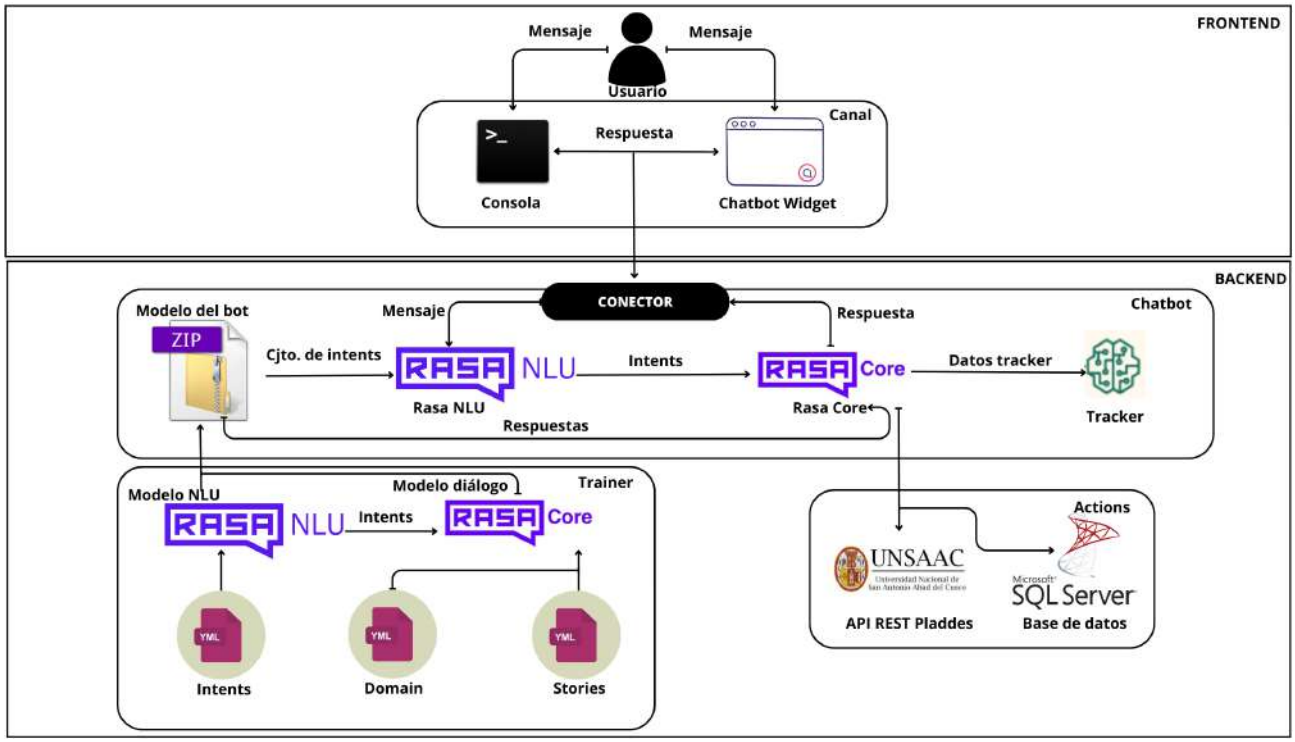
Una vez que los dos servidores estén iniciados, debemos realizar algunas pruebas para asegurarnos de que el flujo de la conversación sea correcto y que el chatbot responda como se esperaba.
Para realizar las pruebas necesarias, utilizamos Postman, que habilita y simplifica el uso de la API ingresando la URL de la API y el texto en formato Json, tal como se interactuará con el widget. Hacemos solicitudes POST a la URL http://localhost:5005/webhooks/rest/webhook y dentro del cuerpo se agrega la identificación del user junto al mensaje, que es la entrada recibida por el chatbot.
La figura D.1 muestra un ejemplo enviado mediante Postman, la conversación inicia con el mensaje "que es el traslado interno", lo que correspondería a la intención consultar_descripcion.

3.3.5. Prototipo implementado

En esta etapa presentamos el desarrollo del prototipo del chatbot para que ayude a los usuarios y de esta manera optimizar la asistencia virtual del trámite documentario. En la figura 3.8 se evidencia la arquitectura de referencia para el prototipo donde resalta el frontend y backend. El frontend lo compone ChatBot-Widget de código abierto para conectar el bot

a través de Canales Rest y la consulta al API para luego mostrar la secuencia de mensajes generados por el modelo. El backend se encarga de recibir el texto en lenguaje natural para procesarla de acuerdo a configuración de entrada de los modelos descritos anteriormente, estos modelos se encargan de procesar la entrada y seguidamente generar una respuesta.

Figura 3.8: Diagrama de arquitectura



Fuente: Elaboración Propia.

3.3.5.1. Conversation-Driven Development

Existen muchos detalles a considerar al momento de implementar un chatbot, uno de los principales problemas es que inicialmente el conjunto de datos suele ser creado por los desarrolladores, lo que significa que está sesgado y alejado de un contexto real.
Tal como se explica en (RASA, 2020), Conversation-Driven Development(CDD) o el desarrollo basado en conversaciones es el proceso de escuchar a los usuarios y utilizar esa información para mejorar el chatbot con IA, es el enfoque general de mejores prácticas para el desarrollo de chatbots. CDD incluye las siguientes acciones:

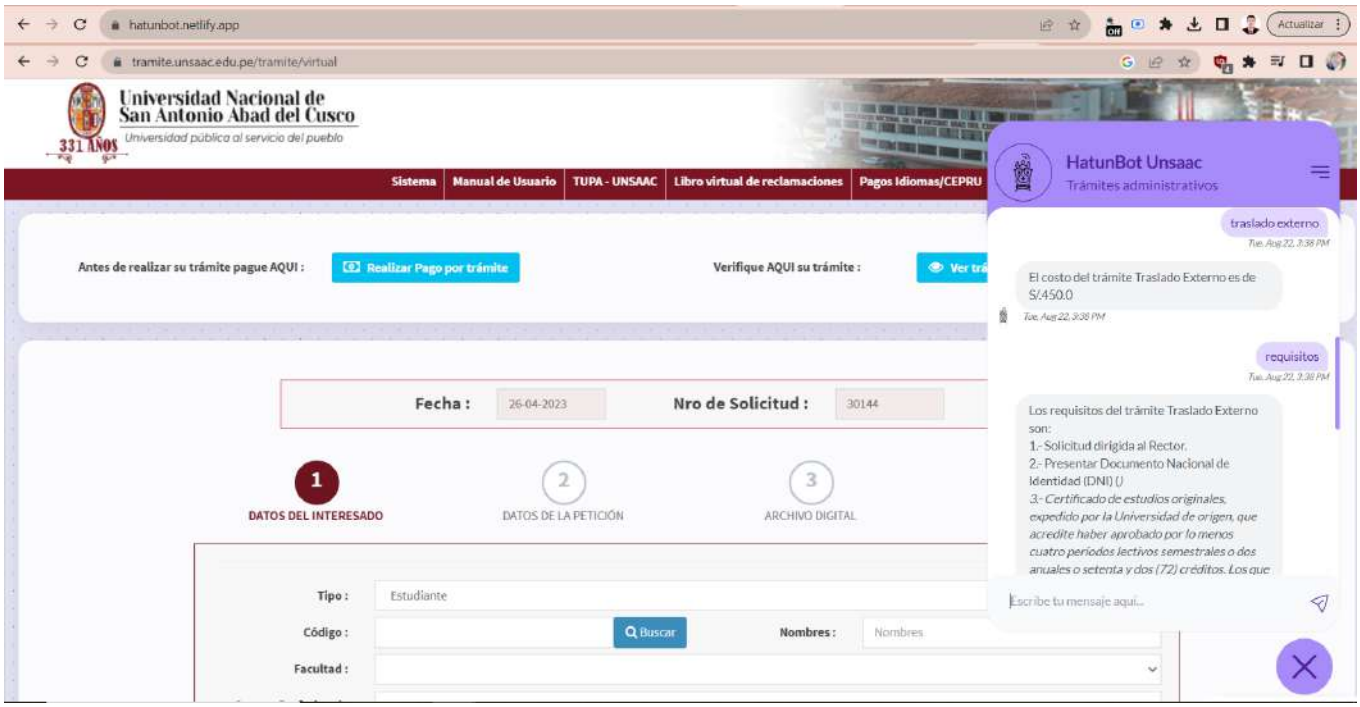
1. Compartir el prototipo del chatbot con los usuarios para que lo puedan probar lo antes posible.
2. Revisar las conversaciones de forma regular, desde el prototipo hasta la producción.
3. Anotar los mensajes y utilizarlos como datos de entrenamiento de NLU.
4. Probar que el chatbot siempre se comporta como esperamos usando conversaciones completas como pruebas.
5. Realizar un seguimiento de las fallas del chatbot y medir su rendimiento a lo largo del tiempo.

6. Corregir la forma en que el chatbot maneja las conversaciones fallidas y verificar las conversaciones exitosas para convertirlas en pruebas de inmediato.

3.3.5.2. Frontend

Para implementar la interfaz de usuario, se utilizó Chatbot-Widget, que es una biblioteca OpenSource, que permite integrar el chatbot con la API REST del servidor Rasa. Realiza solicitudes HTTP entre el servidor e internet, envía mensajes estándar y recibe la respuesta proporcionada por el modelo entrenado.

Figura 3.9: Prototipo implementado

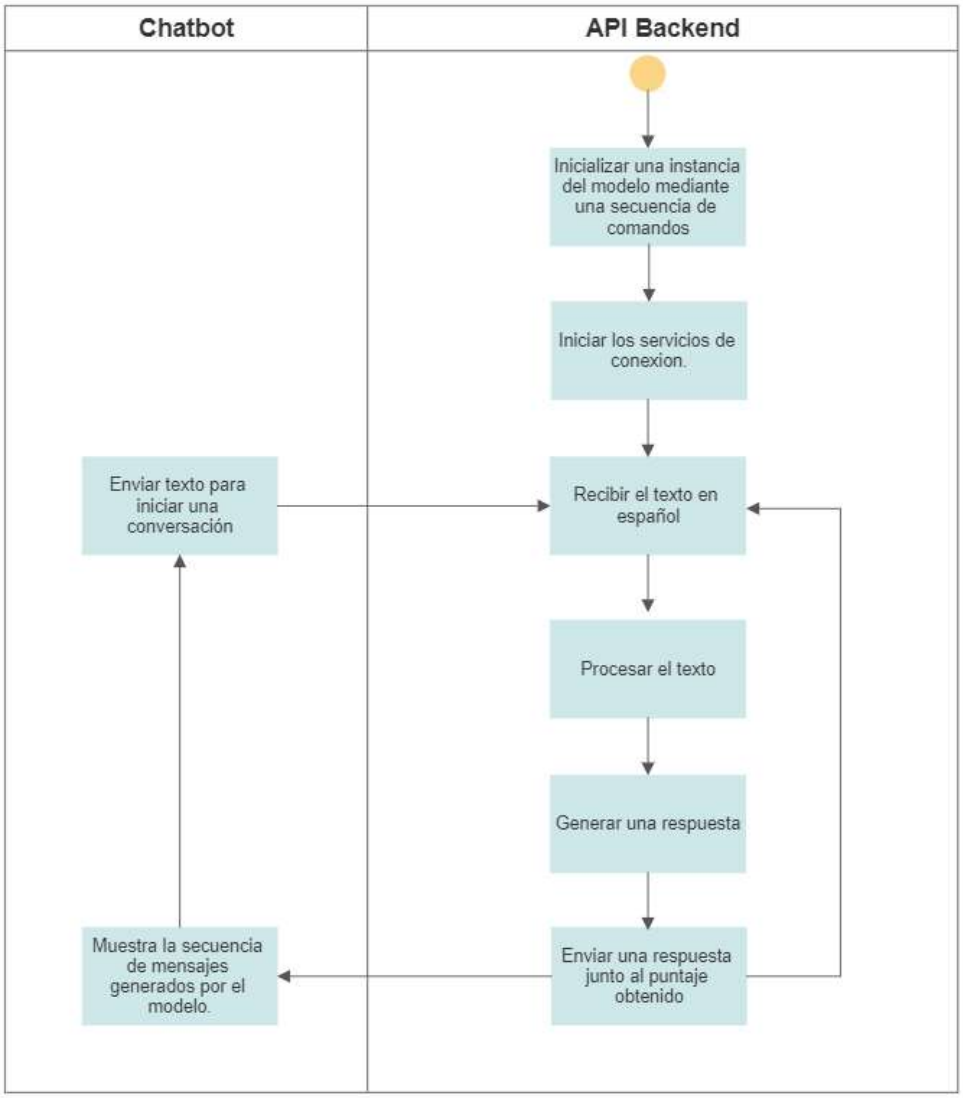


Fuente: Elaboración Propia.

3.3.5.3. Backend

El backend se encarga de procesar el texto ingresado por el usuario para así generar una secuencia de mensajes e interactuar con el mismo. Para la implementación de los endpoints se usó el entorno de desarrollo de RASA usando Python. En la figura 3.10, vemos el proceso que sigue el API para mostrar la secuencia de mensajes generados por el modelo.

Figura 3.10: Proceso de inferencia de la API

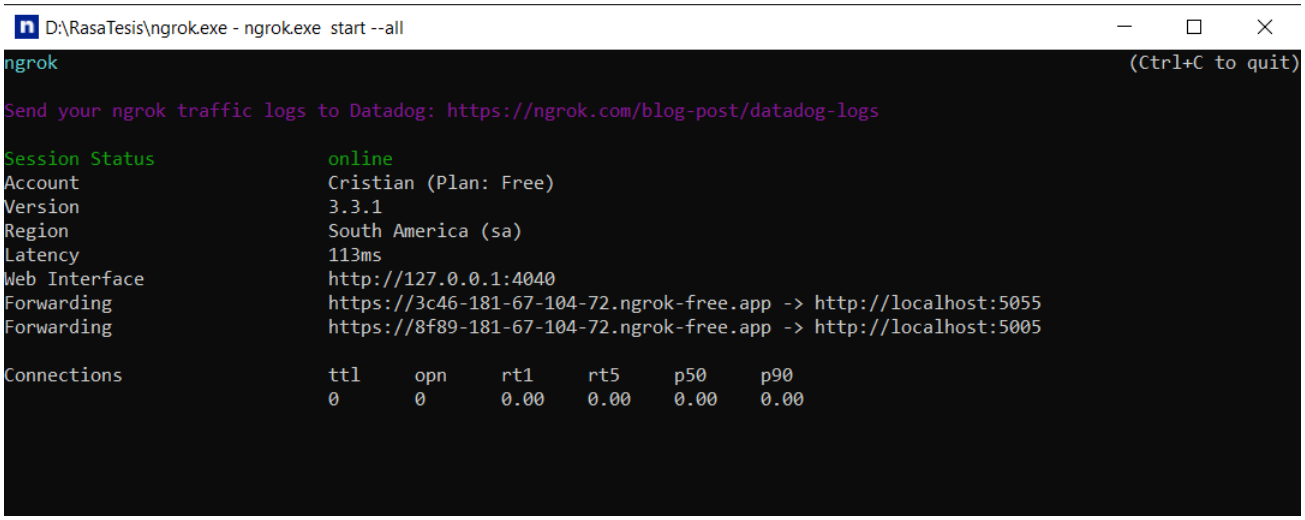


Fuente: Elaboración Propia.

3.3.6. Despliegue del Chatbot

Si bien es cierto los servidores se ejecutan un poco lento, es por el uso de las librerías y los recursos limitados que hay en la máquina virtual, apartir del momento de su ejecución su tiempo de respuesta es mas acelerada.
Para finalizar vamos ha usar el programa ngrok en una máquina virtual, este programa permite establecer túneles http para exponer la dirección local al exterior, seguidamente tenemos el servidor de las acciones y el servidor Rasa en los puertos 5055 y 5005, para las acciones y el modelo entrenado respectivamente, el cual recibe y envía solicitudes POST en formato JSON al framework RASA. Además, en la misma máquina se encuentra un servidor de base de datos(SQL Server), importante para la realización del proyecto. La interacción actual con RASA se realiza a través de una integración directa con la plataforma web, debido a que los servidores necesarios están libres mediante ngrok (ver figura 3.11), esto ayuda a realizar pruebas en un entorno web.

Figura 3.11: Activación del ambiente de publicación URL – NGROK.



Fuente: Elaboración Propia.

El sistema sigue un flujo en el que el usuario solicita información al chatbot a través de la plataforma alojada en HatunBot 2.0. La solicitud del usuario se envía a Rasa, donde se evalúa el intent correspondiente. Si el intent tiene un webhook habilitado, se crea un JSON con el nombre del intent y los valores de las entidades identificadas. Este JSON se envía a través de una petición POST a una URL proporcionada por ngrok, que redirige el tráfico al servidor web en Netlify. Allí, se procesa la solicitud, se accede a los endpoints y se genera una respuesta en formato JSON que se envía de vuelta a RASA. Finalmente, RASA envía la respuesta al usuario a través del widget incrustado en la simulación de la plataforma web de trámite documentario, así como se muestra en la figura 3.9.

3.3.7. Pruebas

Se han realizado pruebas exhaustivas durante el desarrollo del chatbot para garantizar que todo funcione correctamente. La parte más importante del proceso es interactuar directamente con el bot para ver cómo reacciona, ver si el bot está interpretando correctamente el lenguaje natural y comprobar si funciona la integración externa; y de esta manera evidenciar la aceptación del modelo.

3.3.7.1. Pruebas de integración

Para las pruebas de integración, realizamos las necesarias para asegurar que todos los elementos con los que interactúa el bot funcionan correctamente. En la tabla 3.8, la prueba Nro 3 se ejecuta en todos los casos posibles porque es parte fundamental para las consultas al chatbot, porque se requiere interactuar con los endpoints. Esto prueba la integración entre el chatbot y la base de datos.

3.3.7.2. Pruebas del modelo y aceptación

El éxito del desarrollo de la solución para implementar un chatbot basado en inteligencia artificial se basa en la aceptación del usuario hacia el modelo adaptado al prototipo. La meta es lograr un producto exitoso que satisfaga las necesidades del usuario y proporcione respuestas útiles a través del chatbot, es por tal motivo que se ha entregado el chatbot a un número de usuarios para que puedan probar lo que consideren oportuno, asegurándose de que el modelo cumple con todas las expectativas y está libre de errores. En la tabla 3.8 podemos observar los resultados de la interacción entre el chatbot y el usuario, respondiendo las consultas para la cual fue entrenado, debemos señalar que será posible llamarlas con expresiones parecidas por la aplicación de inteligencia artificial.

Tabla 3.8: Pruebas de aceptación

N° prueba	Nombre	Descripción	Entrada	Salida esperada	Resultado de prueba
PRU01	Saludar usuario	Se define el saludo al usuario.	Buenos días	Hola, soy HatunBot. soy un asistente virtual y estaré para poder resolver dudas como: - Qué tramites tengo - Requisitos, descripción y monto a pagar de un trámite Entre otras...	Correcto
PRU02	Solicitud online	Consultar acerca de la solicitud online.	¿Puedo hacer un trámite en linea?	Claro que sí, lo puedes hacer directamente en esta página, solo llena el formulario.	Correcto
PRU03	Solicitud online sin conexión	Consultar acerca de la solicitud online, se verifica la capacidad de interactuar con el chatbot sin conexión a los endpoints.	¿Puedo hacer un trámite en linea?	Desafortunadamente, estoy teniendo un problema de conexión :D . Te agradecería si pudieras volver a intentarlo más tarde.	Correcto
PRU04	Horario de atención	Consultar acerca del horario de atención.	¿a que hora puedo realizar mi trámite?	Puedes realizar tus trámites entre las 8:00 a.m. - 1:00 p.m. y 3 p.m. - 5 p.m.	Correcto
PRU05	Consultar descripción de un trámite	Consultar la descripción de un trámite.	¿para que necesito la Constancia de no ser deudor a la Universidad?	El trámite Constancia de no ser deudor a la Universidad es un Procedimiento a través del cual, se establece la existencia de alguna carencia pendiente de pago bien utilizado no devuelto y que el administrado necesita poseer para realizar trámites de su interés.	Correcto
PRU06	Consultar monto a pagar por un trámite	Consultar por el monto a pagar por un trámite.	Cuánto es el costo del trámite Traslado Externo?	El costo del trámite Traslado Externo es de S/.450.0	Correcto
PRU07	Consultar requisitos de un trámite	Consultar por los requisitos necesarios para realizar un trámite.	Cuales son los requisitos para el tramite de Silabos de Estudios en Maestrias o Doctorados?	Los requisitos del trámite Sílabos de Estudios en Maestr´ıas o Doctorados son: 1.- Solicitud dirigida al Rector. 2.- Pago por derechos de sílabos de Maestría o Doctorado Notas: 1.- Pago por derechos de reinicio de estudios: S/ 22.00 Por Asignatura	Correcto
PRU08	Consultar monto, descripción o requisitos de un trámite	Consultar monto, descripción o requisitos de un trámite, durante la prueba de interacción con el chatbot, se simula un escenario en el que no se especifica un trámite en particular.	cuanto cuesta un tramite?	Entiendo, ¿indícame el nombre del trámite?	Correcto
PRU09	Consultar mis trámites	Consultar los trámites que realicé durante el año	que trámites tengo?	Cual es tu codigo universitario?	Correcto
PRU10	Consultar donde se realiza el pago	Consultar donde realizar el pago del trámite.	como hago para pagar mi tramite?	Puedes pagar aqui http://servicios.unsaac.edu.pe/recaudacion/	Correcto
PRU11	Consultar ubicación	Consultar donde se ubica la UNSAAC	donde se encuentra la UNSAAC?	La UNSAAC se encuentra en la Av. de La Cultura 773.	Correcto
PRU12	Consultar si tiene otra consulta	Consultar si tiene otra consulta	(En cualquier situación que el usuario ya haya hecho una consulta)	¿Tienes otra duda en la que te pueda ayudar?	Correcto
PRU13	Consultar descripción con memoria	Consultar la descripción de un trámite recordando el nombre mencionado con anterioridad	para que sirve	El trámite Traslado Externo es un Procedimiento a través del cual el administrado (estudiante de otra universidad del país o del extranjero) solicita traslado externo para realizar estudios en una determinada Escuela Profesional de la UNSAAC.	Correcto

Fuente: (UNSAAC, 2021)

Capítulo 4

Análisis y discusión de resultados

En esta etapa se presentan las pruebas que se realizaron al modelo y demostramos haber realizado los objetivos establecidos en este proyecto. Primero verificamos que el modelo entiende y responde de una manera natural, y por otro lado, se utilizó un cuestionario validado para medir la usabilidad del chatbot por parte de usuarios reales.

4.1. Análisis de resultados del modelo

4.1.1. Evaluación del modelo NLU

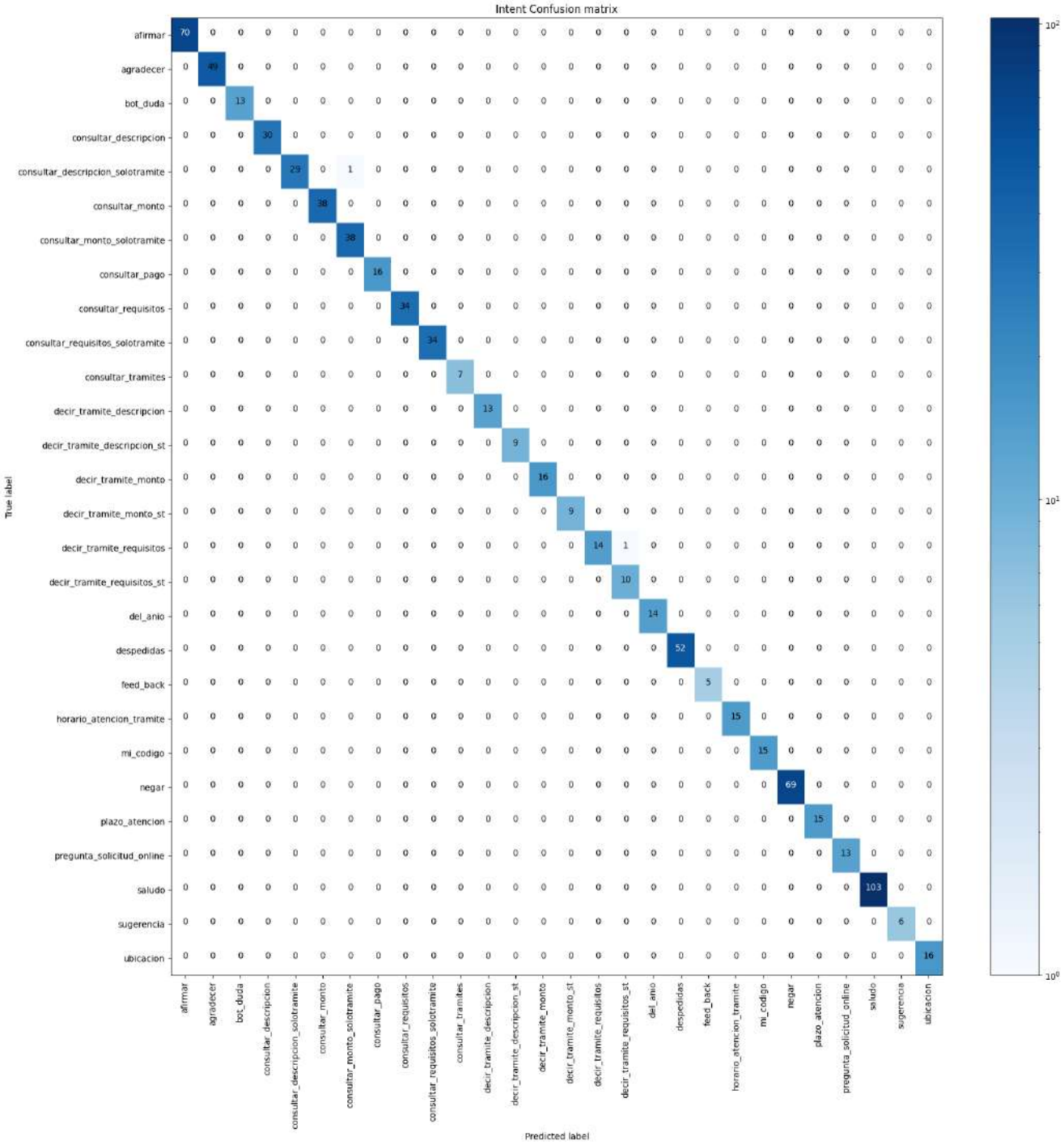
Como se describió en la tabla 3.1, se utilizó una configuración con valores idóneos en los hiper-parámetros para el modelo NLU y se corrió un test aplicando lo siguiente:

- Se dividió el dataset en 2 partes, 80 % de entrenamiento y 20 % de prueba.
- Se excluyó un cierto porcentaje de los datos de entrenamiento.
- Se entrenaron los modelos para cada configuración con el porcentaje de datos de entrenamiento restantes.
- Se evaluó cada modelo con el 100 % de los datos de prueba.

El proceso anterior se repitió con diferentes porcentajes (0 %, 25 %, 50 % y 75 %) de datos de entrenamiento en el paso 2 para tener una idea de cómo se comportaría la arquitectura si aumentara la cantidad de datos de entrenamiento. Dado que el entrenamiento no es completamente determinista, todo el proceso se repitió tres veces para dicha configuración.
Luego de aplicar la prueba al modelo, se generó una matriz de confusión de intents. En esta matriz, los datos que se encuentran fuera de la diagonal son los intents que se confundieron con otros intents, esto se debe a que al generar datos de prueba, es necesario buscar términos menos comunes. Además, tenemos claro que los usuarios no ingresarán algo muy diferente a los datos de entrenamiento y por consiguiente ocurran errores en la clasificación. Lo ideal es conseguir que la mayoría de datos se encuentren en la diagonal, por lo que dicha matriz sirvió para identificar los intents mal clasificados, y realizar las modificaciones necesarias a

los datos de entrenamiento.
Posteriormente se volvió a realizar el test varias veces, hasta conseguir una matriz con dispersiones mínimas como se muestra en las figuras 4.1 y 4.2 , (visualizamos las matrices de confusión de las intenciones en el entrenamiento y validación).

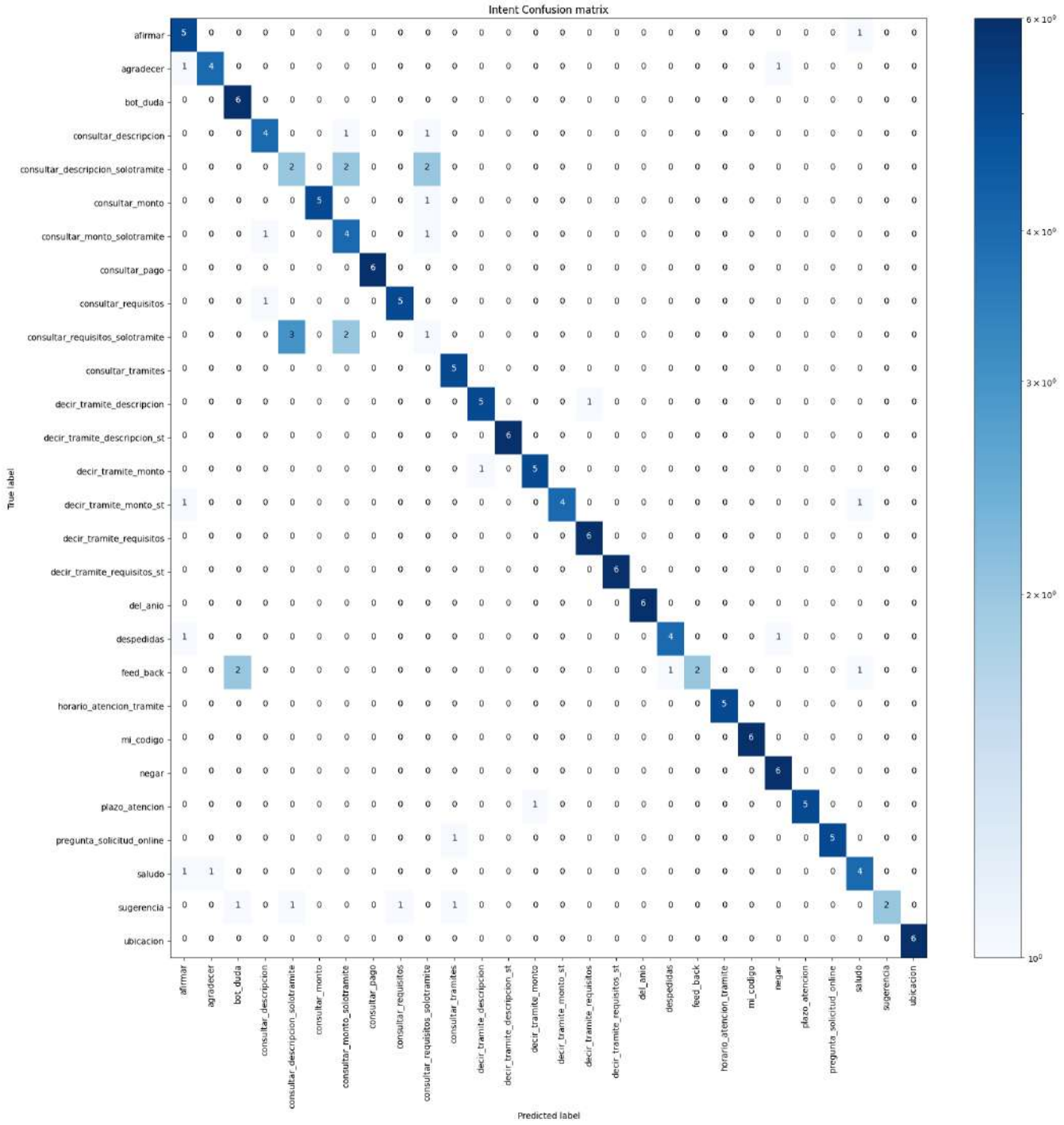
Figura 4.1: Matriz de confusión de los intents en **train**



Fuente: Elaboración Propia.

Observamos que en el entrenamiento los predice sin ningún error, sin embargo, durante la validación, seguimos encontrando algunos errores en algunas intenciones como saludo y despedidas.

Figura 4.2: Matriz de confusión de los intents en **test**

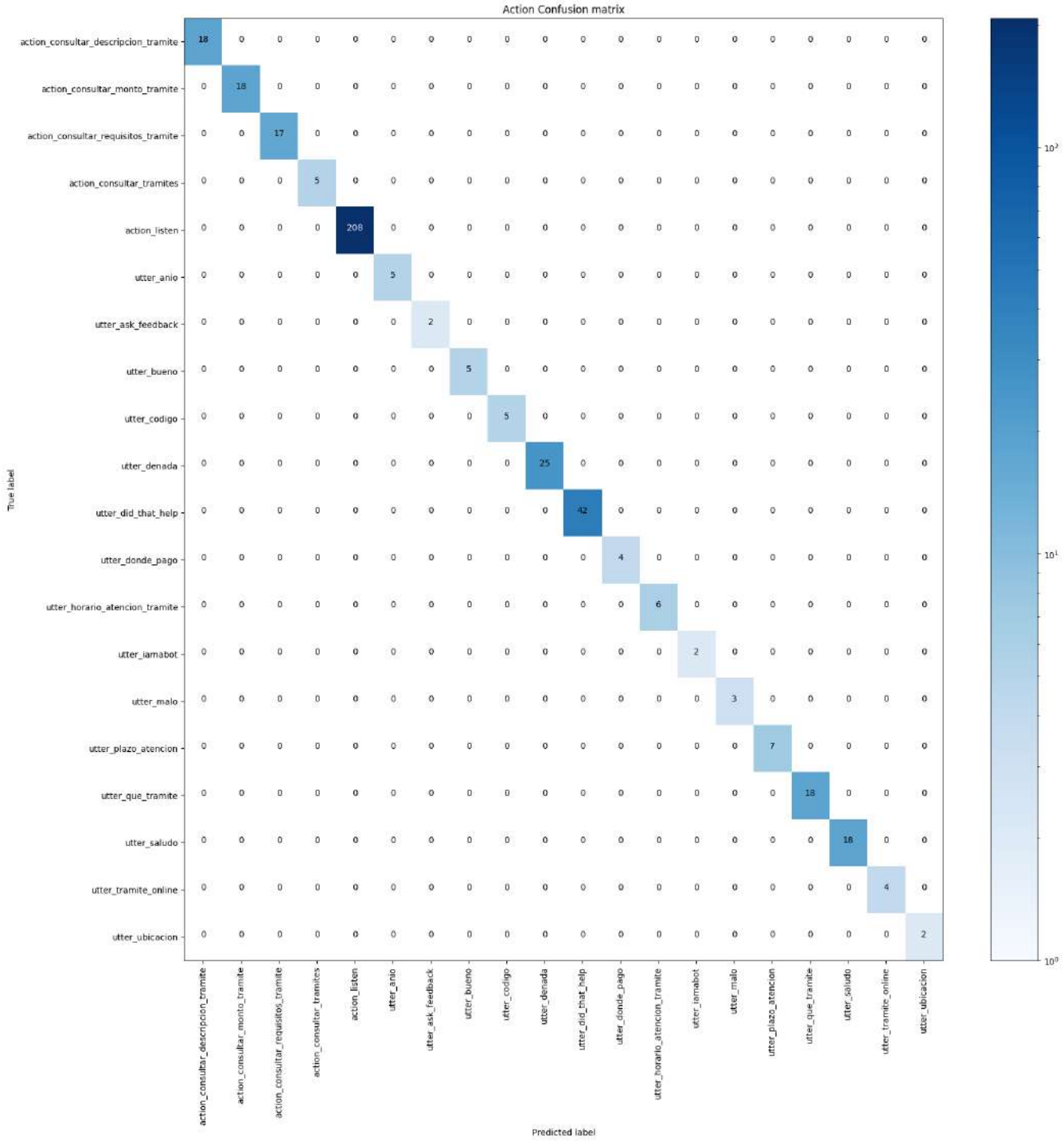


Fuente: Elaboración Propia.

4.1.2. Evaluación del modelo Core

Así como el modelo NLU, el Core también se configuró con una arquitectura descrita en la tabla 3.2; no obstante la forma de realizar el test fue diferente respecto al test del NLU. Se creó un dataset de pruebas con flujos de conversaciones similares a los flujos de entrenamiento, y se corrió el test. Luego de esto, se obtuvo una matriz de confusión con la clasificación correcta de todas las acciones, tanto para los datos de entrenamiento y de validación, tal como se muestra en las figuras 4.3 y 4.4.

Figura 4.3: Matriz de confusión de los utterences en **train**

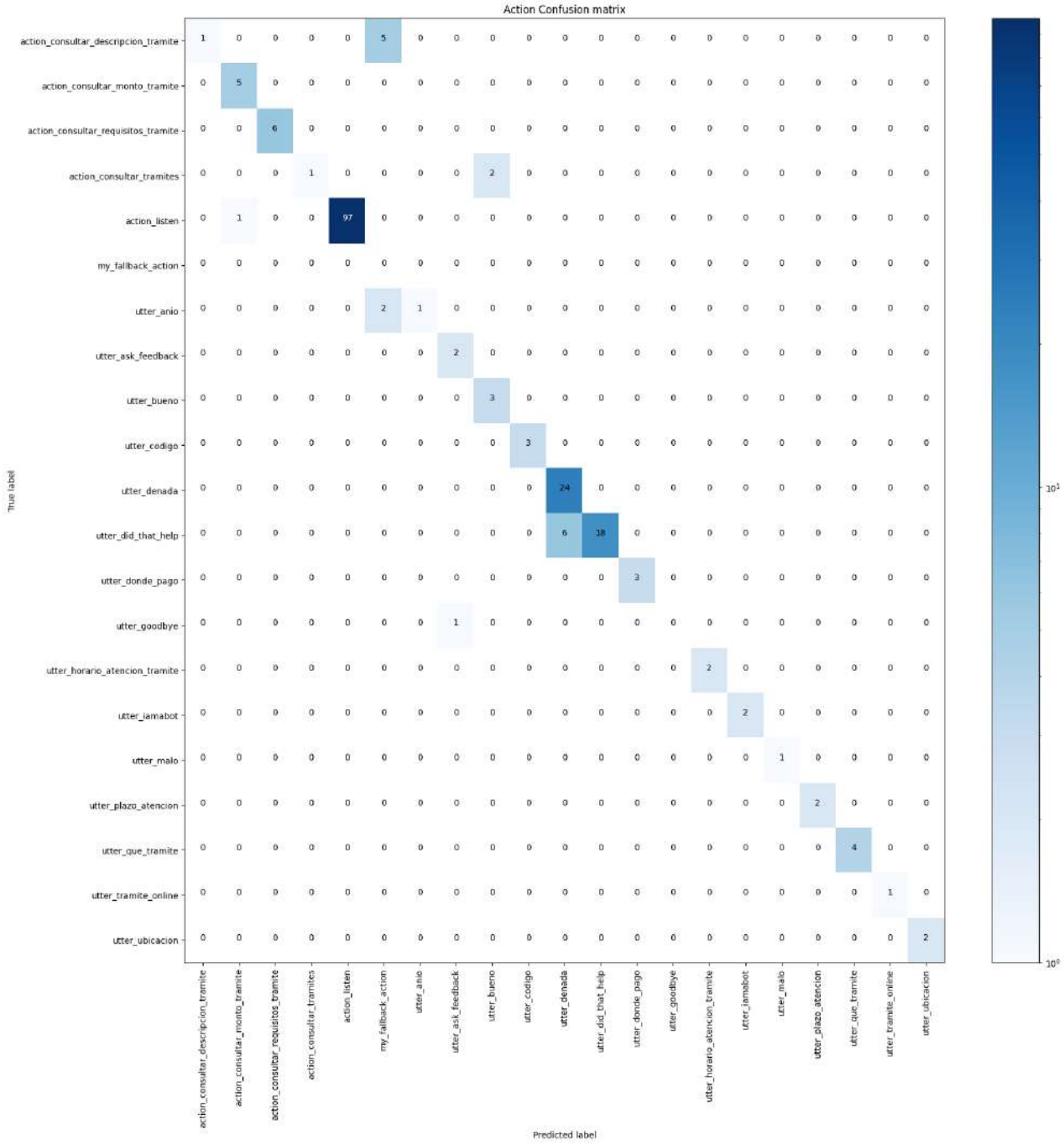


Fuente: Elaboración Propia.

La interpretación de esta matriz es la misma que la ya descrita en el punto anterior, por lo que se repitió el proceso varias veces hasta que la matriz tenga la mayor cantidad de datos en la diagonal.

La razón principal por la que este modelo es impecable es que, dada una intención determinada y slots requeridos, siempre esperamos que responda con la acción propuesta. Por ejemplo, cuando el usuario dice hola, el chatbot debe responder con un saludo, si no proporcionamos el nombre del trámite para uno específico, debe solicitarlo primero antes de continuar con la siguiente acción, y así sucesivamente. Cuando el modelo "comprende" dichos patrones, este debe responder con la acción propuesta.

Figura 4.4: Matriz de confusión de los utterences en **test**



Fuente: Elaboración Propia.

4.1.3. Resultados del modelo

Para probar el modelo, se escribieron diferentes tipos de frases(intents) y se crearon posibles conversaciones(stories), estos archivos se encuentran en el directorio results. Se introdujeron errores ortográficos y oraciones abreviadas para probar si los pequeños cambios en el conjunto de entrada afectan el rendimiento del chatbot, si bien no es una solución universal de clasificación de datos dentro del campo de la IA donde los datasets de entrenamiento y validación son numéricamente equivalentes, son cruciales para la implementación de los chatbots. La tabla 4.1 muestra la confianza promedio obtenida de todas las intenciones, para datos de entrenamiento y validación.

Tabla 4.1: Confianza media para las intenciones.

Intención	**Confidence en train**	**Confidence en test**
saludo	0.99	0.72
despedida	0.98	0.58
afirmar	0.96	0.55
plazo_atencion	0.99	0.76
consultar_tramites	0.96	0.81
pregunta_solicitud_online	0.97	0.78
horario_atencion_tramite	0.93	0.85
consultar_descripcion	0.98	0.79
consultar_monto	0.94	0.77
consultar_requisitos	0.97	0.76
consultar_pago	0.96	0.79

Fuente: Elaboración Propia.

Se puede observar que las intenciones con puntuaciones bajas en el test de validación fueron las básicas, frente a las intenciones relacionadas a los trámites administrativos que obtuvieron puntuación superior a 0,75. Este es un buen resultado porque, por ejemplo, como se mencionó, hay diferentes formas de solicitar trámites, requisitos, montos, etc y las ha identificado de manera correcta. Sin embargo, en algunas intenciones se cometieron errores debido al uso de expresiones inusuales en el lenguaje escrito.
Tomar en consideración que al haber un número reducido de datos de entrenamiento, y por tanto de validación, los errores tienen un efecto notorio.
Se escribieron algunas preguntas para lo cual el chatbot no fue entrenado, así poder garantizar que el puntaje de confianza de la intención con la que clasifica una oración se reduce considerablemente. Como puede verse en la tabla 4.2, las solicitudes mencionadas obtuvieron un nivel de confianza que no supera los 0.40, para estos ejemplos devolverá el **FallBack Action**. Aquí queremos señalar que cuando se le hacen preguntas fuera de su alcance, incluso añadiendo entidades con los que ha sido entrenado, demuestra que son funciones que aún no se han implementado, porque el nivel de confianza es muy bajo.

Tabla 4.2: Confianza para frases fuera del alcance del bot.

Entrada de texto	**Confidence**
¿Qué hora tienes?	0.39
Déjame un correo de contacto	0.37
Cómo te consideras?	0.32
Brindame el nro de contacto de área de trámite	0.35
Creame un trámite para el código 144996	0.28

Fuente: Elaboración Propia.

Aunque se presenten errores en las frases de validación, esperamos que los usuarios no las escriban, pues la acción no se realizaría correctamente. Por ejemplo, el bot no entendería de que trámite quiere los requisitos, descripción, monto, si está escrito incorrectamente. En consecuencia, podemos determinar que toda entidad es extraída de manera satisfactoria.
Las 2 principales razones de la adecuada identificación de las intenciones y la extracción de las entidades realizada por el modelo DIET son las siguientes:

- Se introdujeron expresiones para identificar entidades, lo que facilitó la identificación y contribuyó a la correcta implementación del modelo.
- Por el momento, las capacidades del bot son limitadas, no se realizan muchas tareas en él y esto se puede identificar fácilmente, porque hay que usar palabras clave para preguntar, como: trámites, requisitos, monto, descripción de X trámite, etc; el modelo si es capaz de reconocer estas palabras, por lo que el entrenamiento muestra las intenciones más comunes para clasificarlas correctamente.

Por último, para iniciar una conversación con el chatbot, requerimos de diferentes comandos expuestos anteriormente. La conversación resultante de prueba se puede ver en el apéndice A.1, se comprueba cuán satisfactorio es el resultado de la conversación propuesta y si contesta de manera correcta a todas las solicitudes realizadas. También, le realizamos una pregunta fuera de contexto y resuelve a la perfección, devolviendo una frase para solucionar perfectamente la situación.

4.2. Análisis de resultados de usabilidad

Para verificar el correcto funcionamiento del modelo, se realizó una prueba de usabilidad con usuarios reales. Esto ayudó a que los errores pudieran ser detectados y corregidos.

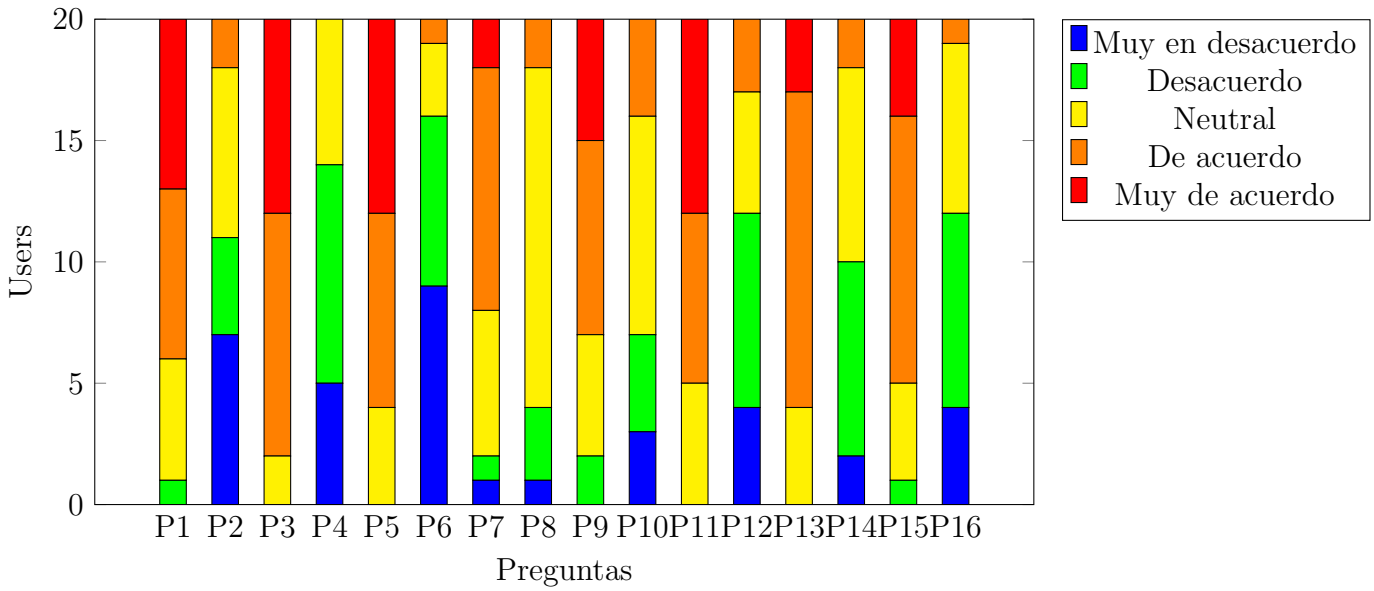
4.2.1. Evaluación de usabilidad

Para medir qué tan bueno fue el modelo, se utilizó Chatbot Usability Questionnaire (CUQ) porque es un cuestionario diseñado específicamente para medir la usabilidad de un chatbot. Este fue elaborado por un equipo interdisciplinario de la Universidad de Ulster. Así mismo, CUQ fue validado como parte de una investigación de doctorado en agosto de 2019 en dicha universidad. (Holmes et al., 2019)
Los enunciados del CUQ fueron traducidos y utilizados en el cuestionario que se llevó a cabo de manera online a un grupo de 20 personas, estos fueron principalmente estudiantes de la UNSAAC. El cuestionario constó de los siguientes enunciados:

1. El chatbot tenia una personalidad realista y atractiva.
2. El chatbot parecía demasiado robótico.
3. El chatbot fue acogedor al momento de presentarse.
4. El chatbot era poco amigable.

5. El chatbot explicó su alcance y propósito bien.
6. El chatbot no dio ninguna indicación sobre su propósito.
7. El chatbot era fácil de seguir.
8. Sería fácil confundirse al usar el chatbot.
9. El chatbot me entendió bien.
10. El chatbot no pudo reconocer muchos de mis mensajes.
11. Las respuestas del chatbot fueron útiles, apropiadas e informativas.
12. Las respuestas del chatbot fueron irrelevantes.
13. El chatbot pudo actuar bien ante errores o equivocaciones.
14. El chatbot parecía incapaz de manejar errores.
15. El chatbot fue muy fácil de usar.
16. El chatbot era muy complejo.

Figura 4.5: Representación gráfica de la escala de Likert

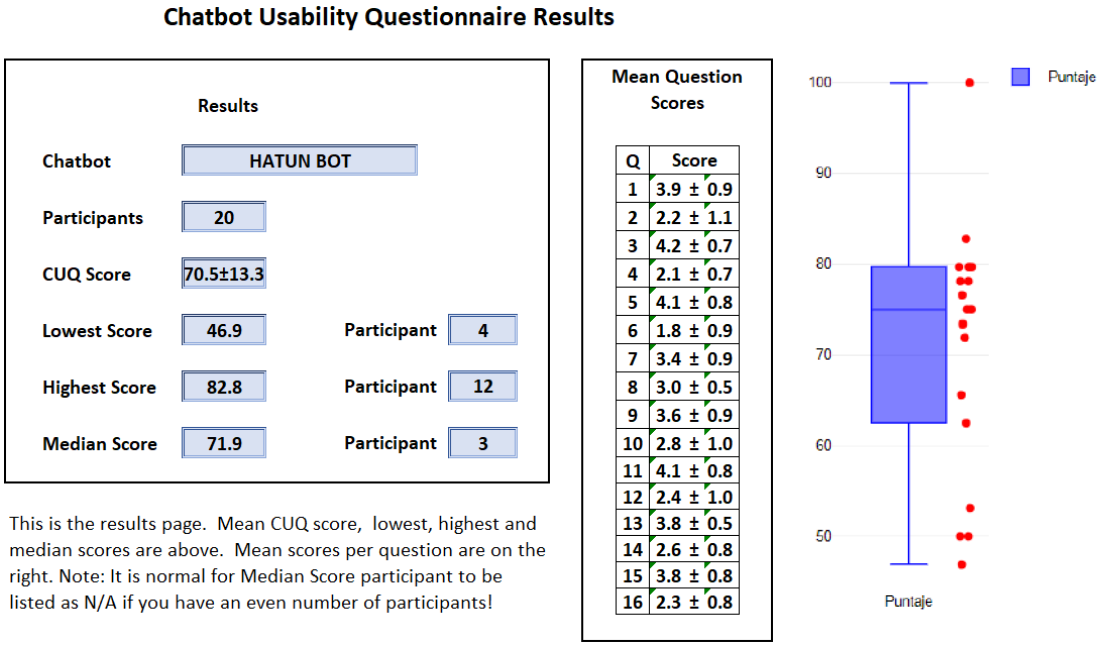


Fuente: Elaboración Propia.

4.2.2. Resultados de usabilidad

Tal como se muestra en la figura 4.6, las dieciséis preguntas se calificaron utilizando una escala de Likert con puntuaciones del 1 al 5, donde 1 significaba "muy en desacuerdo", y 5 "muy de acuerdo". La puntuación promedio obtenida fue de 70.5 en base a 100 puntos, esto demostró que el chatbot implementado funcionó de manera satisfactoria y cumplió con su propósito.

Figura 4.6: Diagrama de caja de los puntajes CUQ obtenidos



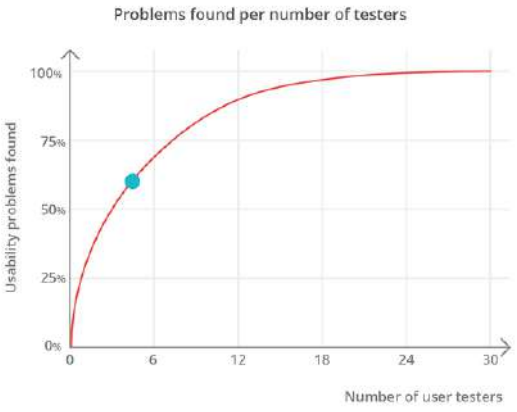
Fuente: Elaboración Propia.

Para la evaluación final se realizó una única pregunta:

1. ¿La solución propuesta optimizaría el trámite documentario en la universidad?

Los resultados obtenidos de dicha pregunta alcanzaron el 80 % de aprobación (ver figura 4.8). Esto guarda relación con los resultados del CUQ, por lo que podemos interpretar que el chatbot optimizaría el proceso de trámite documentario (reduciendo tiempo de permanencia en la página, disminuyendo reclamos, minimizando errores, entre otros).
(Nielsen and Landaue, 1993) concluyeron que 5 usuarios son suficientes para obtener buenos resultados, demostrando, que con esta cantidad de usuarios encontraremos el 84 % de los problemas que existen con la UX para cualquier interfaz y creen que hacer pruebas con más personas no es necesario.

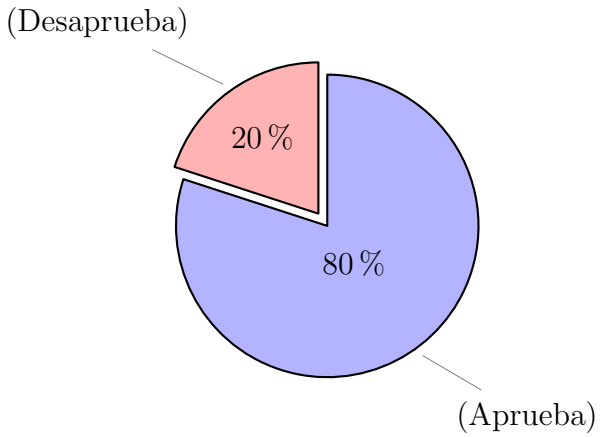
Figura 4.7: Problemas de usabilidad encontrados por usuario



Fuente: (Nielsen and Landaue, 1993).

Es importante mencionar que aunque la muestra fue pequeña, nos brindó información relevante para concluir con el propósito del trabajo de investigación . La encuesta se realizó a un grupo de personas relacionadas con la universidad entre un rango de edad de [20, 35] años, puesto que la gran mayoría de usuarios del PLADDES son usuarios con esa edad que en algún momento estarán propensos a necesitar asistencia en sus trámites; por lo que dicha muestra reflejó información relevante y válida para el proyecto.

Figura 4.8: Porcentaje de aceptación del modelo



Fuente: Elaboración Propia.

4.3. Discusión de resultados

En los resultados obtenidos en nuestro trabajo de investigación podemos observar que la implementación de un buen chatbot basado en inteligencia artificial contribuye a optimizar consultas y procesos administrativos de una universidad, coincidiendo en ese sentido con las investigaciones de (Bavishi, 2019) y (Ingale et al., 2021).
Debemos indicar que fue ventajoso implementar el chatbot en un entorno de código abierto frente a plataformas de desarrollo listas para usar. (Bavishi, 2019) encontró limitaciones en las funcionalidades que brinda la suscripción gratuita de Azure y esta sujeto al vendor lock-in, es por este motivo que para el desarrollo de nuestro proyecto se eligió el framework RASA de python. Esto nos permitió implementar libremente, y adicionalmente desplegar el modelo de manera local para luego exponerlo a internet manteniendo sus componentes, y teniendo control total y privacidad de los datos. No obstante, (Bavishi, 2019) logró aplicar análisis de sentimientos a su agente, característica que no fue implementada para nuestro caso, haciendo que su bot tenga más personalidad.
Como se indicó al inicio del proyecto, nuestro objetivo general era el de optimizar el proceso de trámite en la UNSAAC, coincidiendo con (Ingale et al., 2021), puesto que el motivo principal de su proyecto fue reducir la carga de trabajo del personal de oficina de la universidad y reducir el tiempo de respuesta a las consultas de los usuarios. Respecto a la implementación de su prototipo, este consta de botones para interactuar con los usuarios, por lo que no les da libertad. Por otro lado, nuestro prototipo permite al usuario escribir libremente, haciéndolo mas completo y con un alcance más amplio.
De acuerdo con (Teodoro, 2021), la aplicación de Conversation-Driven Development(CDD) en la validación de la usabilidad del prototipo genera resultados positivos, puesto que se logra involucrar a los usuarios en el proceso de implementación para obtener información tanto de nuevas funciones como de mejora de las existentes. Resultado de ello es que el chatbot recibió un score alto en el CUQ, así como en la pregunta final de evaluación.
(Pinto and Quispetupa, 2019) recomendaron implementar chatbots orientados a temas específicos utilizando arquitecturas transformer, esto debido a que obtuvieron resultados favorables en su proyecto. Siguiendo esta recomendación fue que logramos implementar nuestro chatbot usando transformers, y logramos verificar que esta arquitectura es óptima para la implementación de chatbots específicos.

Conclusiones

1. La implementación de este chatbot ha contribuido significativamente a mejorar la experiencia del usuario en el ámbito de la asistencia virtual, habiendo obtenido un 80 % de aprobación en la encuesta final. Las mejoras específicas incluyen tiempos de respuesta más rápidos, una mejor comprensión de las necesidades de los usuarios y una experiencia más personalizada, lo que resulta en una optimización del proceso.

2. El chatbot superó las limitaciones de la interacción del lenguaje natural, permitiendo una comunicación más fluida y realista. La implementación de algoritmos avanzados de procesamiento de lenguaje natural ha contribuido en gran medida a mejorar la capacidad del chatbot para comprender las consultas de los usuarios y responderlas de manera coherente.

3. Se logró que el chatbot sea autosuficiente, gestionando consultas y solicitudes sin intervención humana adicional integrándolo con plataformas de hosting de alta disponibilidad. Esto garantiza que el chatbot esté siempre disponible para ayudar, mejorando la experiencia del usuario al brindar respuestas rápidas y precisas en cualquier momento.

4. Se obtuvo un grado de usabilidad satisfactorio en el chatbot (70.5/100 en el score del CUQ), lo que significa que dicho chatbot fue intuitivo y fácil de usar. En consecuencia, la interacción con este fue más eficaz y eficiente.

Recomendaciones

- Las funcionalidades actuales del bot todavía necesitan algunas mejoras potenciales respecto a los datasets utilizados, puesto que fue implementado desde cero. Así mismo, se recomienda utilizar un dataset pre-entrenado para ver cómo afectaría al desempeño del chatbot.
- Para agregar la función de comunicación por voz, Rasa es compatible con Audiocodes VoiceAI connect, el cual es un portal que utiliza Speech-to-Speech engines. Cabe mencionar que esta funcionalidad solo está disponible con Rasa Pro, la cuál es un servicio de pago.
- Rasa soporta la integración a diversas aplicaciones como WhatsApp mediante Twilio, Facebook Messenger, Telegram, etc., por lo que se podría integrar el chatbot a dichas plataformas.
- Un aspecto intrigante también podría ser investigar las autocorrecciones, debido a que para este estudio sólo se utilizaron errores ortográficos simples.
- Se recomienda adicionalmente utilizar un Pipeline que incluya un analizador de sentimientos para que el chatbot responda según el input del usuario, ya que el nuestro solo responde de forma amigable sin tomar en cuenta esta característica.

Bibliografía

1MillionBot (2020). Uja: chatbot post-covid para estudios, acceso y matrícula a la universidad. url: https://ost.torrejuana.es/uja-chatbot-post-covid-estudios-acceso/.

Amazon Web Service (2021a). Benefits of a chatbot. url: https://aws.amazon.com/es/what-is-a-chatbot/.

Amazon Web Service (2021b). Common chatbot use cases. url: https://aws.amazon.com/es/what-is-a-chatbot/.

AnalystPrep (2021). Supervised machine learning, unsupervised machine learning, and deep learning. url: https://analystprep.com/study-notes/cfa-level-2/quantitative-method/supervised-machine-learning-unsupervised-machine-learning-deep-learning/.

Andina (2021). andina. url: https://andina.pe/. [recuperado el 10-03-2022].

Aunoa (2020). ¿que tipos de chatbot existen? url: https://aunoa.ai/que-tipos-de-chatbot-existen/. [recuperado el 12-04-2020].

Bavishi, U. K. (2019). *Implementing a college enquiry chatbot*. California State University.

BotCenter (2021). Botcenter. url: https://www.botcenter.pe/. [recuperado el 10-03-2022].

Botfront (2020). Better intent classification and entity extraction with dietclassifier pipeline optimization. url: https://botfront.io/blog/better-intent-classification-and-entity-extraction-with-diet-classifier-pipeline-optimization.

Brownlee, J. (2022). A gentle introduction to positional encoding in transformer models, part 1. url: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/.

Chavan, A. (2020). Complete tutorial on named entity recognition (ner) using python and keras. url: https://www.aitimejournal.com/@akshay.chavan/complete-tutorial-on-named-entity-recognition-ner-using-python-and-keras.

de Águeda, F. (2020). Inteligencia artificial cognitiva y sus aplicaciones multisectoriales. url: https://www.linkedin.com/pulse/inteligencia-artificial-cognitiva-y-sus-aplicaciones-de-

Holmes, S., Moorhead, A., Bond, R., Zheng, H., Coates, V., , and Mctear, M. (2019). Usability testing of a healthcare chatbot: Can we use conventional methods to assess conversational user interfaces? url: https://dl.acm.org/doi/10.1145/3335082.3335094.

IIC, I. (2021). Procesamiento del lenguaje natural las máquinas analizan nuestro lenguaje en texto. url: https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/.

Ingale, N., Anand, J. T., Ritin, D., and Vishal, K. B. (2021). College enquiry chatbot using rasa. *International Journal of Scientific Research in Computer Science*, 1(6):6.

Jurafsky, D. and Martin, J. H. (2020). *Speech and Language Processing*. Stanford University and University of Colorado at Boulder.

Kurbiel, T. (2021). Transformer networks: A mathematical explanation why scaling the dot products leads to more stable gradients. url: https://towardsdatascience.com/transformer-networks-a-mathematical-explanation-why-scaling-the-dot-products-leads-to-more-stable-414f87391500.

La verdad (2018). Así es la nueva inteligencia artificial que ayuda a los alumnos de la umu y la upct. url: https://www.laverdad.es/murcia/funciona-nueva-asistenta-20180705124322-nt.html.

Microsoft (2022a). Entity types. url: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/entities.

Microsoft (2022b). Intents. url: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/intents.

Microsoft (2022c). Utterances. url: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/utterances.

Moreno, B. S. (2021). Transformers bert para question-answering sobre covid-19. url: http://e-spacio.uned.es/fez/eserv/bibliuned:master-ETSInformatica-TL-Bsiguenza/SiguenzaBernardo$_T FM.pdf$.

Nielsen, J. and Landaue, T. K. (1993). A mathematical model of the finding of usability problem. url: https://dl.acm.org/doi/epdf/10.1145/169059.169166.

Notebook Community (2016). Tokenization doesn't have to be slow ! url: https://notebook.community/huggingface/pytorch-transformers/notebooks/01-training-tokenizers.

Omdena (2022). How to build a chatbot using rasa: Use case of an ai driving assistant. url: https://omdena.com/blog/how-to-build-a-chatbot-using-rasa/.

Pinto, D. P. and Quispetupa, M. U. (2019). *Chatbot generativo en el idioma español utilizando la arquitectura de red neuronal Transformer*. Repositorio UNSAAC.

RASA (2020). Conversation-driven development. url: https://rasa.com/blog/conversation-driven-development-a-better-approach-to-building-ai-assistants/.

RASA (2021a). Featurizer. url: https://rasa.com/blog/the-rasa-masterclass-handbook-episode-4/.

RASA (2021b). Intent classifiers. url: https://rasa.com/blog/intents-entities-understanding-the-rasa-nlu-pipeline/.

RASA (2022a). Nlu training data. url: https://rasa.com/docs/rasa/nlu-training-data/.

RASA (2022b). Tuning your nlu model. url: https://rasa.com/docs/rasa/tuning-your-model/.

RASA (2023). What is rasa? url: https://medium.com/@bharathreddy028/what-is-rasa-c6f2790a38b7.

Russell, S. and Norvig, P. (2021). *Artificial Intelligence A Modern Approach Fourth Edition*. Pearson Education, Inc.

Sansonnet, J.-P., Leray, D., and Martin, J.-C. (2006). Architecture of a framework for generic assisting conversational agents. *International Workshop on Intelligent Virtual Agents,*, 1(6):145–156.

Shapiro, S. C. (2003). *Encyclopedia of Artificial Intelligence*. John Wiley, New York.

Sianaki, O. A. and Ababneh, N. (2019). A survey on conversational agents/chatbots classification and design techniques. url: https://www.researchgate.net/publication/331746678$_{AS}urvey_on_{C}onversational_{A}gentsChatbots_{C}las$

Solutions, A. (2020). Chatbots:the definitive guide (2020). url: http://marketing.artificial-solutions.com/rs/177-TDV-970/images/Chatbots-the-definitive-guide-2020.pdf.

Statistical Analysis Systems Institute Inc (2021). ¿cómo funciona nlp? url: https://www.sas.com/es$_a$r/*insights/analytics/what − is − natural − language − processing − nlp.html*.

Tamayo, M. (2004). *El proceso de la investigación cientifica*. Limusa Noriega Editores.

Techslang (2020). What is named entity recognition (ner)? url: https://www.techslang.com/definition/what-is-named-entity-recognition-ner/.

Teodoro, J. C. M. (2021). *VIHRTUAL-APP: Un chatbot para la divulgación médica del VIH*. UNIVERSITAT POLITÈCNICA DE VALÈNCIA ESCOLA POLITÈCNICA SUPERIOR DE GANDIA.

Top Big Data (2022). Una introducción suave a la codificación posicional en modelos de transformadores, parte 1. url: https://topbigdata.es/una-introduccion-suave-a-la-codificacion-posicional-en-modelos-de-transformadores-parte-1/.

Torres, G. V. (2017). Qué es deep learning. url: https://openwebinars.net/blog/que-es-deep-learning/.

Troncoso, M. A. A. (2012). *Sistema para el Aprendizaje del Mapudungun.Incluyendo características de reconocimiento de voz y bot conversacional.* Pontificia Universidad Católica de Valparaíso Facultad de Ingeniería Escuela de Ingeniería Informática.

UAH, P. C. (2020). Isidra, el nuevo asistente virtual para la página web de la uah. url: https://portalcomunicacion.uah.es/diario-digital/actualidad/la-uah-estrena-asistente-virtual-para-su-pagina-web.html.

UNSAAC (2021). Texto Único de procedimientos administrativos – tupa de üniversidad nacional de san antonio abad del cusco". url: http://transparencia.unsaac.edu.pe/links/planeamiento/documentos/TUPAUNSAAC$_{N}uevoFormat$

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser, and Polosukhin, I. (2017). Attention is all you need.

Weizenbaum, J. (1966). *ELIZA—a computer program for the study of natural language communication between man and machine*. ACM.

White, K. (2020). Unpacking the ted policy in rasa open source. url: https://rasa.com/blog/unpacking-the-ted-policy-in-rasa-open-source/.

Apéndice A

Conversación con el chatbot

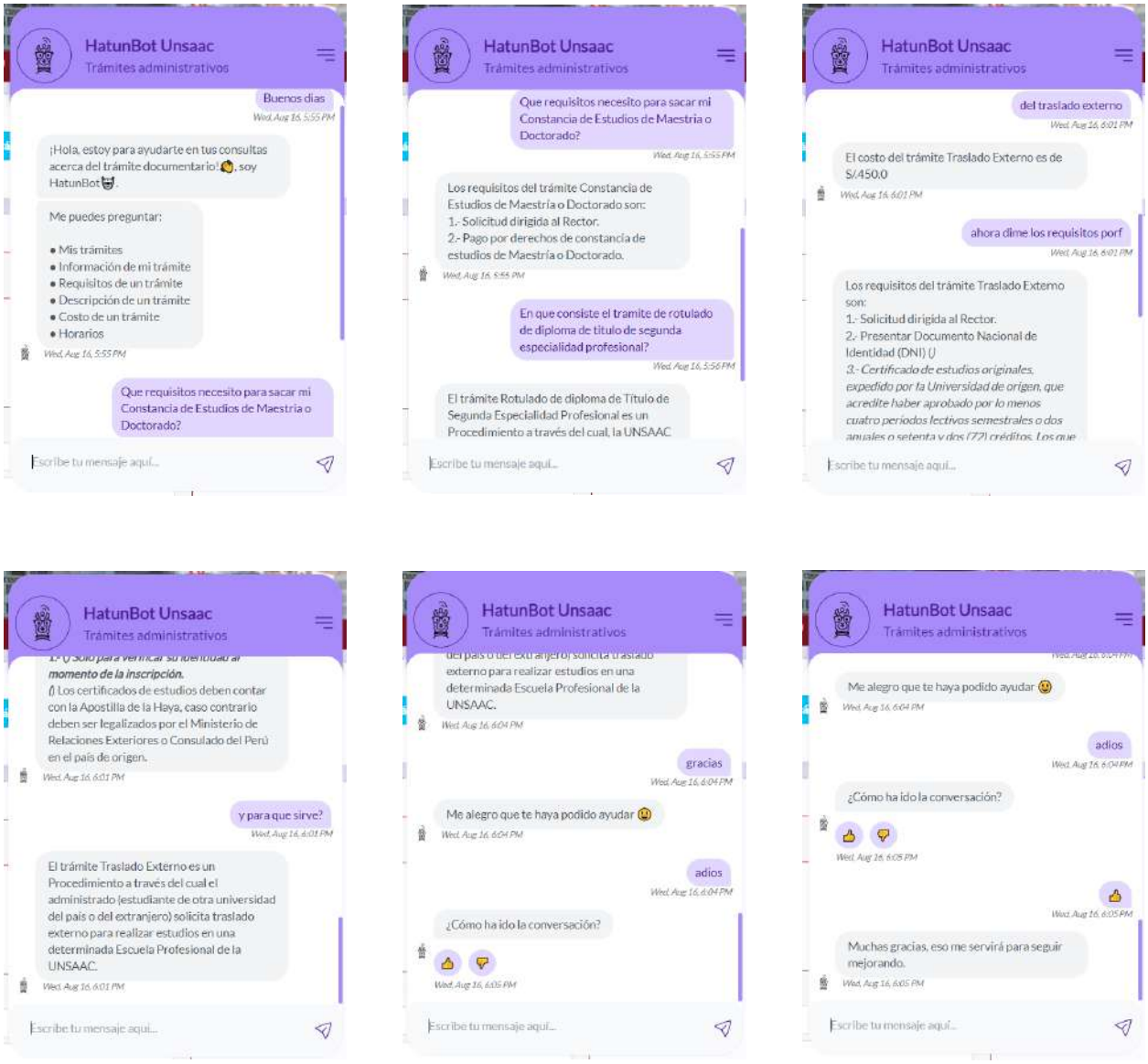
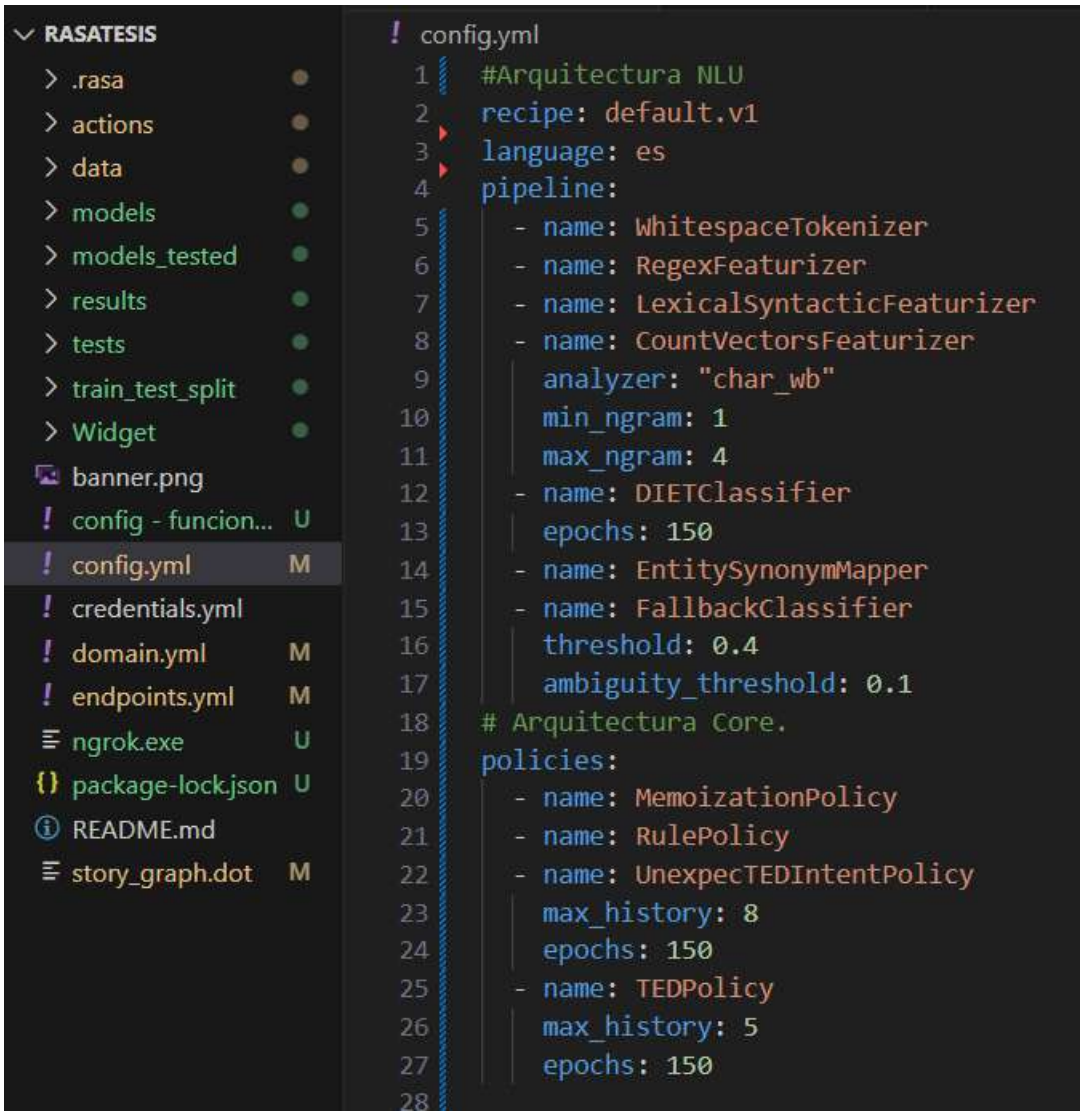


Figura A.1: Conversación con el chatbot.
Fuente: Elaboración Propia.

Apéndice B

Estructura del proyecto

Figura B.1: Estructura del proyecto



Fuente: Elaboración Propia.

Apéndice C

Repositorio del proyecto

Para el trabajo de investigación, se crearon dos directorios diferentes, en la primera parte encontraremos el chatbot (RASA) y en la otra un sitio web (React JS). Para realizar un seguimiento de los cambios realizados, el código de ambos proyectos está almacenado en Github.

- URL del chatbot desarrollado en Rasa:
  `https://github.com/Becessj/Tesis-Chatbot`
- URL de la aplicación web desarrollada en React JS:
  `https://github.com/Becessj/Tesis-Chatbot/tree/main/Widget`

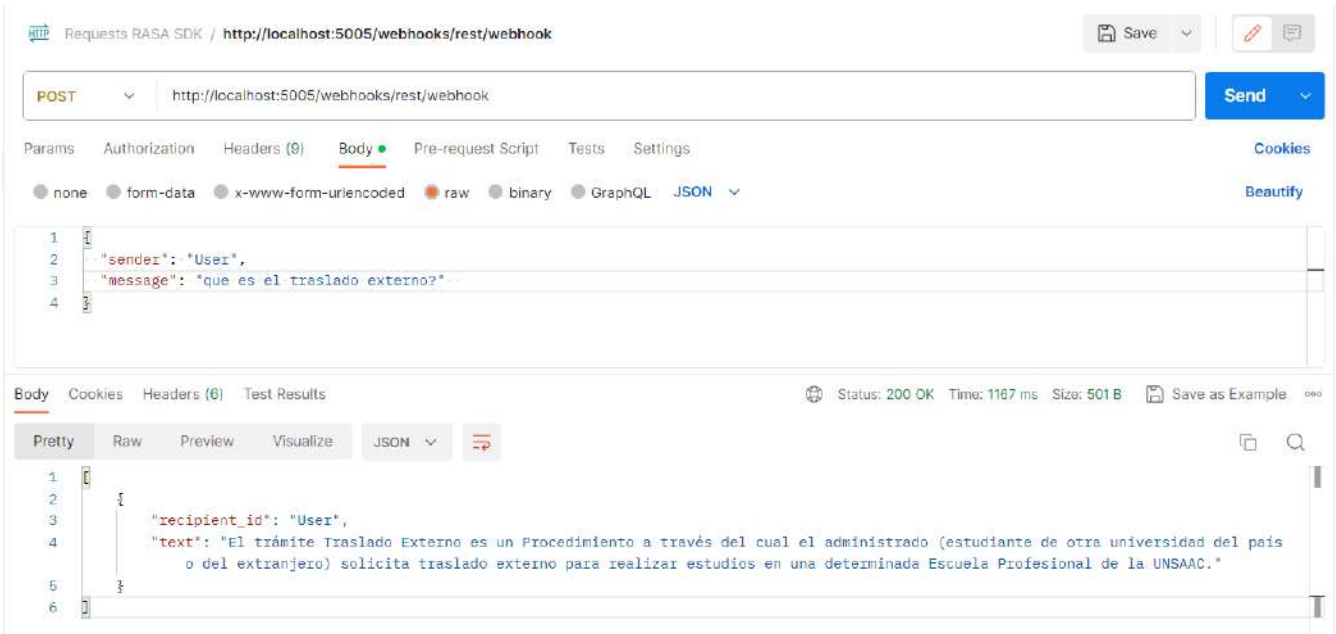
Apéndice D

Interacción con la API

Una forma de comunicarse con el servidor Rasa es a través de la API HTTP esta permite recibir peticiones REST, esta puede recibir mensajes de diferentes tipos de aplicaciones, al crear un chatbot, debemos asegurarnos de que funcione correctamente. Entonces, usamos Postman para probarlo, enviamos diferentes mensajes al chatbot y comprobamos si responde correctamente, hacemos esto enviando un mensaje tipo "POST" a la siguiente dirección: `http://localhost:5005/webhooks/rest/webhook`

- **Consultar sobre un trámite de manera directa**

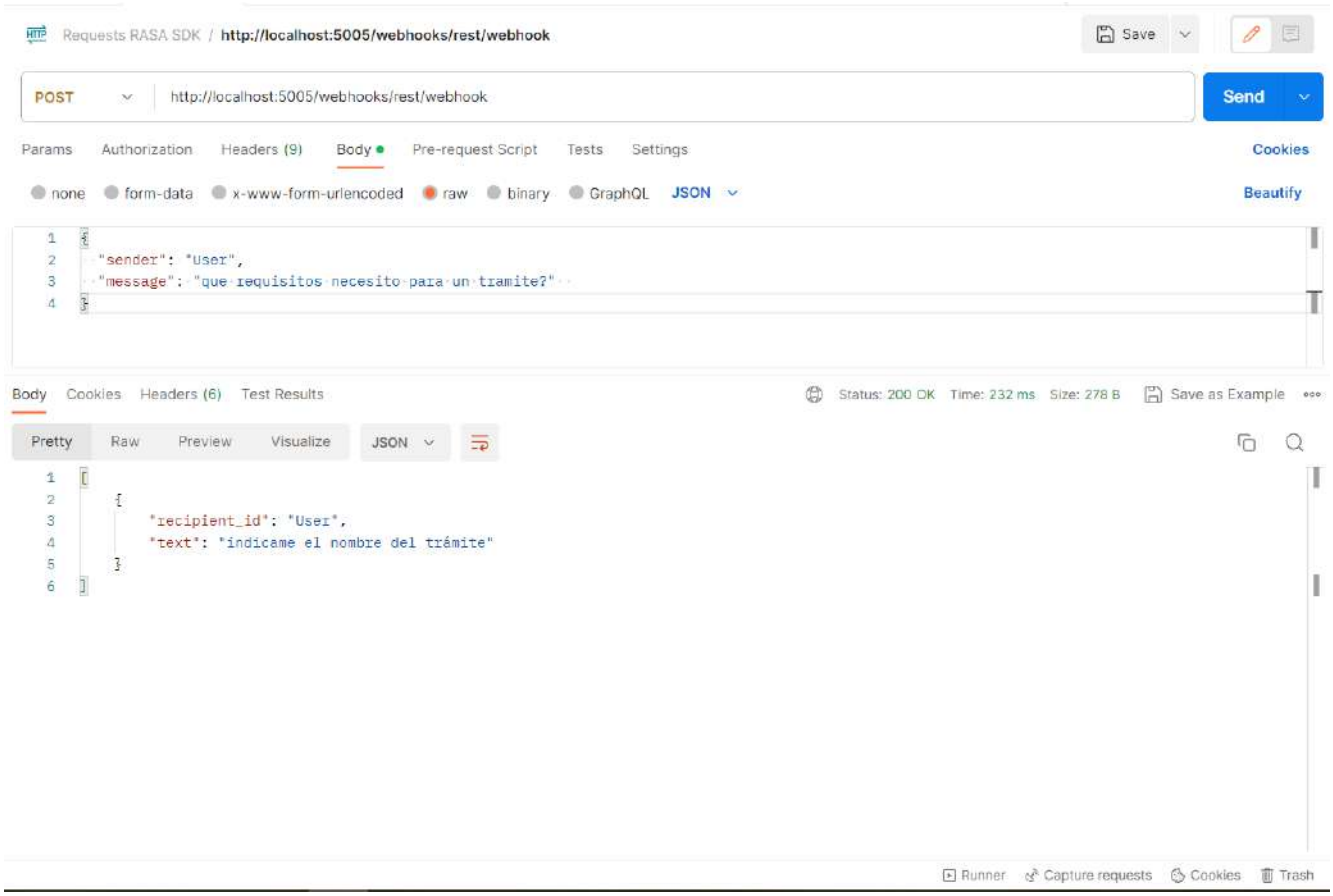
Figura D.1: Solicitud a la API vía Postman

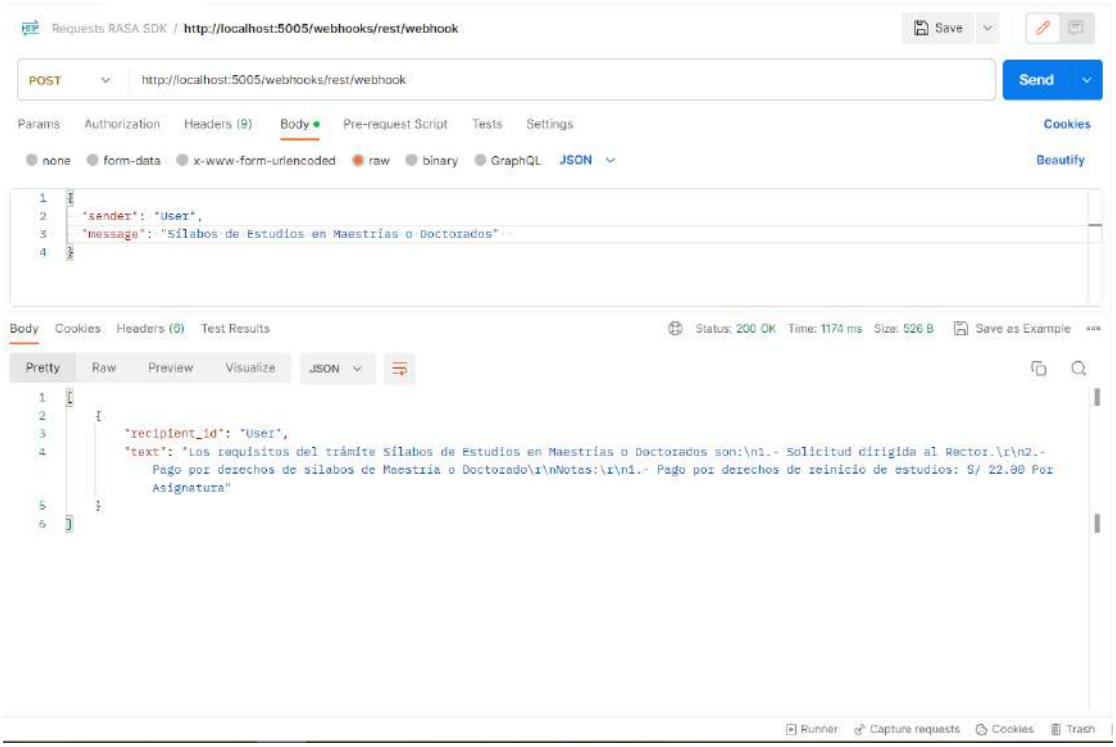


Fuente: Elaboración Propia.

- **Contextualizar la conversación y memoria del bot**

Figura D.2: Aplicación de la memoria del chatbot





Fuente: Elaboración Propia.

Apéndice E

Herramientas utilizadas

- **Software**
  - Lenguaje de programación: Python 3
  - Framework inteligencia artificial: RASA
  - Librería Rasa SDK
  - Librería PyODBC, Pandas, Json, Requests.
  - React JS , Webpack
  - Visual Studio Code
  - SQL Server 2018
  - Ngrok
- **Hardware**
  - GPU: NVIDIA GeForce GTX 1060