ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

UNIDAD DE TITULACIÓN

DESARROLLO DE UN ASISTENTE CONVERSACIONAL PARA GESTIÓN DE COMPRA-VENTA DE ROPA EN LÍNEA USANDO RASA OPEN SOURCE

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SISTEMAS DE INFORMACIÓN

XAVIER IVAN AGUAS HARO
xavier.aguas@epn.edu.ec

Director: MARCO E. BENALCÁZAR, Ph.D.
marco.benalcazar@epn.edu.ec

Quito, Junio 2022

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación “Desarrollo de un asistente conversacional para gestión de compra-venta de ropa en línea usando Rasa Open Source” desarrollado por Xavier Iván Aguas Haro estudiante de la Maestría de Sistemas de Información, mención en Inteligencia de Negocios y Analítica de Datos Masivos, habiendo supervisado el desarrollo de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la defensa oral.

Marco E. Benalcázar, Ph.D.
DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Xavier Iván Aguas Haro declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Xavier Iván Aguas Haro

DEDICATORIA

A todas aquellas personas que llegaron a mi vida, No hay nada más valioso que una persona que te ofrece su tiempo y su escucha atenta y amorosa, ¡gracias!

"Hagas lo que hagas, hazlo bien. Hazlo tan bien que cuando la gente te vea hacerlo quiera volver y verte hacerlo de nuevo, y querrán traer a otros y mostrarles lo bien que lo haces"
Walt Disney

Xavier Iván Aguas Haro

AGRADECIMIENTO

Papá y mamá, ambos son la bendición más bonita en mi vida. No estaría donde estoy hoy sin su apoyo e inspiración.

Me gustaría agradecer de forma muy especial a mi director de tesis PhD. Marco E. Benalcázar por todo su apoyo, continuo seguimiento y retroalimentación brindada para la culminación de este trabajo.

Finalmente, mi caluroso agradecimiento a mis compañeros de JRTEC, que más que compañeros se han convertido en amigos muy cercanos. Hemos aprendido muchísimo en este camino duro del emprendimiento.

Xavier Iván Aguas Haro

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

La tecnología ha generado un impacto significativo en la vida de las personas y ha favorecido el desarrollo del comercio electrónico. Desde que se produjo la crisis sanitaria del COVID-19, tanto el distanciamiento social como el cierre de varios negocios provocaron que la actividad comercial aumentara a través de canales electrónicos en Ecuador. A pesar de que varios negocios optaron por contratar más personal, no fue suficiente debido al gran volumen de preguntas que recibían. En este aspecto, la tecnología ha otorgado a los negocios una serie de herramientas para mejorar la atención al cliente y automatizar los procesos de compra. Uno de los más utilizados son los chatbots. A pesar de que se han realizado esfuerzos para mejorar la efectividad de los chatbots, a través de algoritmos de aprendizaje de máquina, las conversaciones aún se mantienen rígidas y con respuestas, en algunos casos, indeseables. Esto se debe a que no se ha considerado el contexto y el historial de la conversación entre el chatbot y el usuario. Por esta razón, en este trabajo se propone el desarrollo de un asistente conversacional capaz de acompañar al cliente en todo el proceso de descubrimiento de ropa, a través de la utilización de Rasa; un framework que combina la comprensión y procesamiento de lenguaje natural basado en redes neuronales tipo Transformer permitiendo mejorar la efectividad en las respuestas de dicho asistente. Además, Rasa fue combinado con un motor de búsqueda conocido como Algolia para potencializar las habilidades del asistente para encontrar la ropa solicitada y entregar resultados instantáneos. Finalmente, los resultados de este trabajo demostraron que a través de Rasa se puede crear asistentes con una precisión de clasificación de intenciones y extracción de características mayor al 90% y que incorporar procesos de capacitación y evaluación continua con usuarios reales posibilita que el modelo pueda generalizarse a escenarios del mundo real, aumentando así la tasa efectividad. Se obtuvo más de 600 ejemplos de conversaciones para crear el corpus de entrenamiento y se aplicó la metodología CRISP-ML para el diseño e implementación del asistente propuesto.

PALABRAS CLAVE: Rasa Open Source, Asistentes, Chatbots, Algolia, Redes Neuronales Artificiales, Transformers, Comercio Electrónico.

ABSTRACT

Technology has generated a significant impact on people's lives and has favored the development of e-commerce. Since the COVID-19 health crisis, both social distancing and the closure of various businesses have caused commercial activity to increase through electronic channels in Ecuador. This became a challenge due to users write at any time of the day, always looking for an immediate response. Although several businesses chose to hire more staff, it was not enough due to the high volume of questions they received. In this regard, technology has given businesses a series of tools to improve customer service and automate purchasing processes. One of the most used are chatbots. Although efforts have been made to improve the effectiveness of chatbots, through machine learning algorithms, conversations remain rigid and with responses, in some cases, undesirable. This is because the context and history of the conversation between the chatbot and the user have not been considered. For this reason, this work proposes the development of a conversational assistant capable of accompanying the client throughout the clothing discovery process, using Rasa; a framework that combines the comprehension and processing of natural language based on transformer-type neural networks, allowing to improve the effectiveness of the responses of said assistant. In addition, Rasa was combined with a search engine known as Algolia to enhance the assistant's abilities to find the requested clothing and deliver instant results. Finally, the results of this work showed Rasa can create assistants with an accuracy of intent classification and feature extraction greater than 90% and that incorporating continuous training and evaluation processes with real users allows the model to be generalized to real-world scenarios, thereby increasing the effectiveness rate. More than 600 examples of conversations were obtained to create the training corpus and the CRISP-ML methodology was applied for the design and implementation of the proposed assistant.

KEYWORDS: Rasa Open Source, Assistants, Chatbots, Algolia, Artificial Neural Networks, Transformers, E-commerce.

1. INTRODUCCIÓN

La utilización de chatbots en negocios y empresas ha comenzado a popularizarse en la actualidad y cada vez está tomando más fuerza, a pesar de que está tecnología lleva épocas desarrollándose [1]. Los chatbots aparecieron por primera vez en la época de los 60s y hasta el momento se han logrado establecer dos tipos de chatbots [2].

La primera clasificación de chatbots se caracteriza porque están basados en un conjunto de reglas conocida como máquina de estados. Las máquinas de estado funcionan con determinadas entradas para pasar de un estado a otro, de tal manera que la conversación empiece a fluir entre el chatbot y el usuario. Un chatbot por máquinas de estado no podrá pasar al siguiente estado de la conversación si no recibe la entrada esperada. Esto representa una de las principales dificultades para que un chatbot pueda recuperarse ante una entrada inesperada [3].

Por otro lado, la segunda clasificación de chatbots funciona con inteligencia artificial, pero han dejado de llamarse chatbots y ahora se les conoce como asistentes conversacionales. Los asistentes no solo entienden el mensaje del usuario sino el significado de este, generando una larga interacción con el usuario; no solo con conversaciones sino con una amplia variedad de tareas que puede ejecutar el asistente [4]. Por ejemplo, conectarse con sistemas externos a través de Application Programming Interfaces (APIs, por sus siglas en inglés), motores de búsquedas, bases de datos, servicios web, etc. Por lo tanto, se puede decir, que un asistente conversacional está orientado al usuario mientras que un chatbot está orientado al negocio o empresa.

Existen herramientas como IBM Watson, Google Dialogflow, BotKit y Microsoft LUIS para la creación de asistentes conversacionales aplicando inteligencia artificial [3]. Sin embargo, para obtener todos los beneficios que ofrece cada una de estas herramientas en la creación de asistentes se debe contratar un plan, que en algunos casos llega a ser muy costoso dependiendo de las habilidades y personalización que se requiera.

En base a lo anterior, se decidió escoger un framework de código abierto. Este framework está basado en bibliotecas de aprendizaje automático como Tensorflow y SpaCy; también conocido como Rasa Open Source, el cual empezó, como un proyecto de GitHub en el año 2016. Con el pasar del tiempo se ha perfeccionado y se ha convertido en un conjunto de

herramientas centradas en la creación de asistentes conversacionales basados en inteligencia artificial más popular en los últimos años [5].

De acuerdo con la Cámara Ecuatoriana de Comercio Electrónico (CECE), en 2020 se evidenció un crecimiento del 800 % en visitas a diferentes sitios web y un aumento de 40 % en órdenes de compra en comparación al 2019. Por lo tanto, el 2020 marcó un punto de inflexión en el comercio electrónico en Ecuador debido a la crisis sanitaria del COVID-19 [6], [7]. Los usuarios que antes realizaban sus compras de manera presencial empezaron a utilizar distintas plataformas digitales para la compra de alimentos y bebidas, medicamentos, ropa, entre otros; provocando que la venta digital se convierta en un desafío las 24 horas del día [8].

Por esta razón, se decidió en este trabajo desarrollar un asistente conversacional con Rasa para gestionar la comunicación de forma organizada, atendiendo rápidamente a las consultas y sobre todo cubrir la demanda en momentos donde se satura la capacidad de respuesta de los empleados. Para complementar las habilidades de este asistente se usa un motor de búsqueda llamado Algolia para mejorar la efectividad en encontrar la ropa solicitada y entregue resultados en milésimas de segundo.

Lo que se espera con este asistente conversacional es alivianar la carga a los empleados, ahorrar costos de personal y sobre todo recopilar datos para conocer mejor a los usuarios a través del uso de herramientas de código abierto.

La Sección 1 detalla los aspectos generales que se deben tener en cuenta cuando se habla de asistentes conversacionales. La Sección 2 muestra la metodología aplicada para la creación y diseño de un asistente conversacional con Rasa. La Sección 3 describe los resultados que se obtuvieron en el desarrollo y evaluación del asistente. Finalmente, las conclusiones y recomendaciones son presentadas en la Sección 4.

1.1 Objetivo general

El objetivo general de este trabajo de titulación es:

- Desarrollar un asistente conversacional para agilizar y brindar asistencia personalizada en un proceso de compra-venta de ropa en línea usando Rasa Open Source.

1.2 Objetivos específicos

Los objetivos específicos son:

- Diseñar un asistente conversacional orientado a la compra-venta de ropa en línea que sea capaz de ejecutar las tareas de saludar, despedirse, agradecer, revisar si un producto existe en stock, llamada a la acción de compra y devolver un mensaje por defecto en caso de no haber entendido alguna entrada de texto inesperada.

- Modelar e implementar una base de datos en un motor de búsqueda para el manejo de información sobre productos considerando colores, stock, categoría y género.

- Evaluar la exactitud del asistente conversacional en la ejecución de las siguientes tareas: saludar, despedirse, agradecer y revisar si un producto existe en el stock.

1.3 Alcance

Crear un asistente conversacional capaz de acompañar al cliente en todo el proceso de descubrimiento de ropa en línea a través de un punto de contacto. El asistente será desarrollado en un framework de código abierto basado en aprendizaje de máquina. Además, dicho asistente estará conectado a un motor de búsqueda conocido como Algolia para ofrecer resultados de manera rápida y aplicando cierta relevancia en los productos.

1.4 Marco Teórico

Cuando se habla de lenguaje natural, es importante tomar en cuenta que lo que se escribe no necesariamente significa lo que se quiere interpretar. Las personas pueden escribir o hablar de distintas formas para conseguir algo en concreto, pueden cometer errores, introducir palabras erróneas, decir o escribir oraciones fragmentadas e incluso tener una mala pronunciación [9]. Tomando en cuenta lo anterior, aparecieron frameworks de desarrollo de asistentes conversacionales basados principalmente en dos conceptos conocidos como el procesamiento de lenguaje natural (NLP, por sus siglas en inglés) y compresión de lenguaje natural (NLU, por sus siglas en inglés), pero dependiendo, el objetivo de la aplicación se pueden utilizar indistintamente [10].

Cuando se crea asistentes conversaciones, por más simples o sofisticados que sean, el procesamiento de lenguaje natural es una parte primordial ya que otorga el poder de reconocer o clasificar la intención de un mensaje. Por otro lado, el componente de compresión de lenguaje natural otorga al asistente comprender el significado de la intención en cada expresión durante la conversación. A continuación, se detalla cada uno de estos conceptos.

1.4.1 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural es un subconjunto de la inteligencia artificial y tiene como objetivo procesar datos de lenguaje; esto implica descomponer el lenguaje en elementos más pequeños para comprender las relaciones que tienen entre ellos y cómo funcionan entre sí [11]. En otras palabras, el NLP convierte el lenguaje natural en datos estructurados para realizar tareas como análisis y reconocimiento de voz, etiquetado y extracción de información.

1.4.2 Compresión del lenguaje natural

Por otro lado, la compresión del lenguaje natural es un subconjunto del NLP y que se encarga de interpretar y comprender el lenguaje de las personas. Es decir, busca el significado de un mensaje ante sutiles variaciones de lenguaje [12]. Se utiliza en tareas como, por ejemplo, el análisis de sentimiento, semántica, contexto, etc.

1.4.3 Asistentes conversacionales

Un asistente conversacional es aquel que maneja las conversaciones basadas en el contexto y que adicionalmente puede realizar tareas más complejas para ayudar a los

usuarios en sus transacciones. A este tipo de asistentes, les importa lo que el usuario ha dicho antes, cuándo, dónde o cómo lo dijo. Considerar el contexto también significa tener la habilidad de comprender y responder a entradas diferentes e inesperadas [13].

Crear un asistente conversacional en primera instancia puede resultar un desafío, ya que implica tomar en cuenta las preguntas y tareas que se relacionan directamente con la actividad del negocio. La creación de este tipo de asistentes requiere aprendizaje automático, librerías sofisticadas para procesar lenguaje natural y datos en forma de conversaciones de usuario. Todo esto, para comprender el contexto y realizar la acción correcta, a pesar de tener entradas ambiguas por parte del usuario, interjecciones, bromas y otras posibles maneras de que una conversación pueda desviarse de la ruta ideal [14].

Una aplicación conversacional que contenga inteligencia artificial independientemente de las capacidades que se le otorgue está compuesta por varias capas que se detalla a continuación:

- **Capa de cálculo:**

Es la capa fundamental sobre la cual se basan las capas superiores. En ella aparecen lenguajes de programación como Python y distintos frameworks de aprendizaje automático como TensorFlow, Spacy y PyTorch para realizar procesamiento de lenguaje natural.

- **Capa de infraestructura**

Esta capa está compuesta de una infraestructura estándar que proporciona herramientas de aprendizaje de máquina (algoritmos) y componentes básicos para crear asistentes conversacionales.

- **Capa de herramientas y servicios**

La capa de herramientas y servicios permite mejorar al asistente para mantener conversaciones y ejecutar cierto tipo de tareas que le permitan seguir aprendiendo de usuarios del mundo real. Existen soluciones que permiten recopilar, visualizar y revisar conversaciones de usuarios para un mejoramiento continuo.

- **Capa de aplicación**

Finalmente, la capa de aplicación es donde el asistente conversacional está desplegado y que le permite interactuar con otro tipo de sistemas y aplicaciones web. Por ejemplo, el

asistente puede conectarse con servicios de correo, motores de búsqueda y bases de datos, convirtiéndole en una herramienta multi-task y enfocada al usuario final.
Por lo tanto, siempre que se inicie con el diseño de un asistente es importante definir las capacidades mínimas y las rutas de conversación ideales que permitan solicitar información al usuario y cumplir una tarea en específico (Asistente mínimo viable). Posteriormente, el asistente debe seguir mejorándose, a través del entrenamiento con conversaciones reales, de tal manera que pueda anticiparse ante entradas inesperadas o mensajes que no cumplan con la lógica comercial establecida.

Para realizar el mejoramiento continuo del asistente aparece una técnica denominada "Conversation-Driven Development", el cual describe el proceso de escuchar a los usuarios y utilizar esas conversaciones para un mejoramiento continuo. Este enfoque se ha convertido en una de las mejores prácticas para el desarrollo de asistentes.

1.4.4 Conversation-Driven Development:

Desarrollar asistentes conversacionales robustos es un desafío ya que los usuarios siempre dirán o escribirán mensajes que no se espera y anticiparse ante cualquier entrada es prácticamente imposible. El principio detrás de esta técnica es obtener conversaciones de usuarios en sus propias palabras y determinar lo que exactamente quieren, de tal manera que el asistente se oriente hacia el lenguaje y comportamiento real de los usuarios que interactúan [10].

Las acciones que comprenden el mejoramiento del asistente a través de esta técnica se ilustran en la Figura 1 y se describen a continuación:

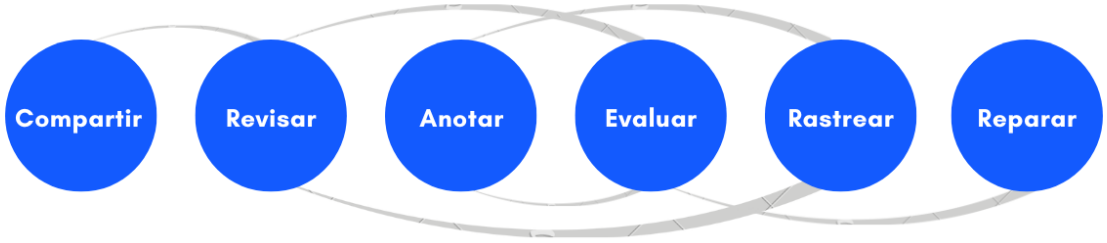


Figura 1 - Técnica Conversation-Driven Development.

1. **Compartir:** Desplegar el asistente mínimo viable a los usuarios para que lo prueben lo antes posible. No es recomendable, utilizar herramientas para generar conversaciones de prueba.

2. **Revisar:** Analizar las conversaciones que se generaron entre los usuarios con el asistente. Esta tarea es vital en cada fase de desarrollo para el diseño de nuevas características para el asistente.
3. **Anotar:** Etiquetar conversaciones reales para mejorar el modelo clasificación de mensajes y extracción de información.

4. **Evaluar:** Usar conversaciones reales completas para determinar la precisión de lo que el usuario quiere decir y sobre todo evaluar la extracción de información clave en los mensajes.

5. **Rastrear:** Identificar conversaciones exitosas y no exitosas para depurar la capacidad de responder a entradas inesperadas y aumentar ejemplos para mejorar el modelo.

6. **Reparar:** Basado en el estudio de conversaciones ya sea exitosas como fallidas; determinar la acción de mejoramiento del asistente y realizar las correcciones pertinentes en el código. Por ejemplo, la necesidad de conseguir más datos de entrenamiento para detectar la intención de un determinado mensaje.

Como se pudo observar en la Figura 1, esta técnica no sigue un proceso lineal, pero garantiza que el asistente se adapte a lo que realmente quiere el usuario, en lugar de que el usuario se adapte al comportamiento del asistente.

La incorporación de esta técnica en el mejoramiento continuo de un asistente es clave en el desarrollo de nuevas habilidades y acceso a funcionalidades más complejas, de tal manera, que los asistentes no solo respondan preguntas, sino que realmente ayuden a los usuarios en tareas concretas.

2. METODOLOGÍA

La metodología aplicada para el desarrollo de este proyecto de tesis se basa en un estándar conocido como **Cross Industry Standard Process for Machine Learning** (CRISP-ML), el cual busca cumplir con todas las expectativas establecidas por una organización, teniendo en cuenta el objetivo de que se incremente la eficiencia y éxito de proyectos de aprendizaje automático [15]. Esta metodología está comprendida de cinco etapas, las cuales se listan a continuación:

- Entendimiento del problema
- Preparación de datos
- Diseño
- Evaluación
- Despliegue

Gracias a estas fases, se puede identificar y mitigar los riesgos que ocurren en la ejecución de un proyecto de manera temprana, asegurando así la calidad de los resultados al momento del despliegue.

2.1 Entendimiento del problema

2.1.1 Problema: El comercio electrónico en Ecuador

El año 2020 marcó un punto de inflexión en el comercio electrónico en Ecuador debido a la crisis sanitaria ocasionada por el COVID-19. Las restricciones impuestas provocaron el distanciamiento social y el cierre de varios negocios, impulsando la actividad comercial a través de canales electrónicos. Varios negocios fueron obligados a adaptarse a esta nueva normalidad [16]. La relación entre los negocios y compradores cambió. Ahora, los usuarios investigan a través de Internet, buscan nuevos productos, promociones e incluso solicitan información detallada antes de realizar una compra.

En este sentido, el trabajo que realizan los negocios en canales digitales se convirtió en un gran desafío, ya que deben responder preguntas a cualquier hora y sobre todo brindar respuestas inmediatas. Algunos negocios han optado por contratar más personal en atención al cliente, pero, aun así, no ha sido suficiente, ya que las preguntas aparecen a cualquier momento.

Para tratar de solventar dicho problema y brindar atención inmediata a los usuarios, varios negocios recurrieron a los chatbots. El uso de chatbots ofrece muchas ventajas tanto a los usuarios como a los negocios. Por ejemplo, con un chatbot se puede gestionar las consultas de forma organizada y rápida, cubrir la demanda en momentos donde se satura la capacidad de respuesta de los empleados y sobre todo conseguir atención al cliente las 24 horas del día. Un chatbot tendrá éxito siempre y cuando genere respuestas precisas y brinde una atención inmediata al responder dudas [17].

La mayoría de chatbots utilizan solamente NLP para detectar intenciones en los mensajes de usuarios. La principal desventaja de utilizar solamente NLP es que se generan conversaciones rígidas y que ante mensajes inesperados se produzcan respuestas, que en algunos casos son, indeseables.

A pesar de esta limitación, aparecieron herramientas y plataformas para el desarrollo de asistentes, y así, solventar el problema de conversaciones rígidas con aprendizaje de máquina (IBM Watson, Google Dialogflow, BotKit y Microsoft LUIS). Esto puede conllevar a contratar un plan de desarrollo con todos los beneficios y que en algunos casos llega a ser costoso dependiendo la personalización en el diseño del asistente. Por otro lado, existe un framework llamado Rasa, el cual permite la creación de asistentes conversacionales gratuitos centrados en inteligencia artificial, desarrollados en Python.

Las ventajas más representativas que ofrece Rasa respecto a otras plataformas para construir asistentes conversacionales son las siguientes: Los datos que genera el asistente conversacional pertenecen al negocio y no están alojados en servidores de plataformas de terceros. Por otro lado, con Rasa se puede personalizar el asistente conversacional como se desee, incluyendo cuantas acciones personalizadas se requiera [4], como por ejemplo conectarse a servicios en la nube, bases de datos y a plataformas de terceros.
Rasa empezó como un proyecto de GitHub en el año 2016 y con el paso del tiempo se ha ido perfeccionando hasta convertirse en uno de los frameworks más importantes para la creación de asistentes conversacionales. Rasa ya es utilizado por miles de desarrolladores en todo el mundo y con resultados prometedores en el área de servicio al cliente en industrias de seguros, salud y por supuesto en el comercio electrónico [18].

2.1.2 Solución: Asistente conversacional con Rasa

Rasa no solamente utiliza NLP, sino que incorpora otro concepto conocido como compresión del lenguaje natural. El objetivo de utilizar NLU es clasificar y entender la

intención de usuario, ofrecer respuestas más acertadas y sobre todo brindarle una conversación personalizada al usuario. Gracias al uso del NLU, crear un asistente conversacional sin contemplar todos los posibles casos que se pueden generar en las conversaciones es una realidad; debido a que dentro de su estructura interna tiene un sistema de seguimiento de conversaciones capaz de analizar el contexto de las conversaciones y así proporcionar una respuesta adecuada, incluso cuando la conversación no coincide exactamente con ninguno de los ejemplos que se utilizaron para entrenar [19].

Rasa está formado de dos componentes principales: Rasa NLU y Rasa Core. El componente Rasa NLU se encarga de interpretar los mensajes del usuario para obtener la intención y extraer entidades disponibles en dicho mensaje a través de configurar procesos de análisis (Pipelines). Por otro lado, Rasa Core es un gestor de diálogo basado en un modelo de probabilidad para decidir la respuesta más acertada en función de los mensajes anteriores del usuario.

La Figura 2 ilustra los componentes principales de Rasa, donde Rasa NLU se encarga de interpretar los mensajes del usuario y Rasa Core decide la siguiente acción que debe suceder.

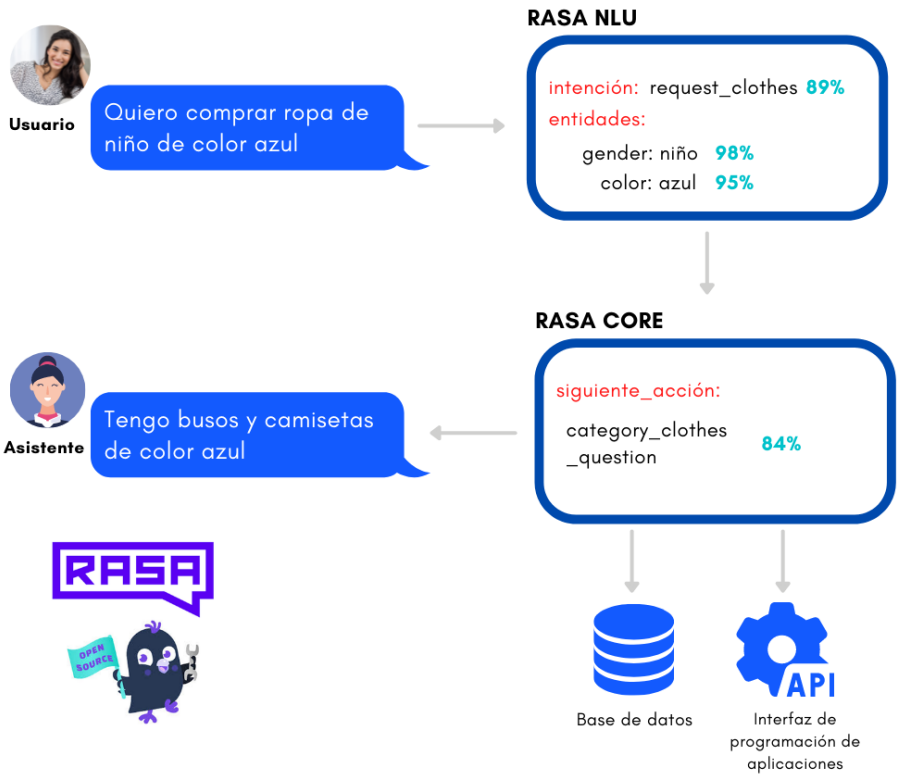


Figura 2 - Componentes principales de Rasa.

Los componentes principales de Rasa se detallan a continuación:

2.1.3 Rasa NLU: Pipeline

Cuando se diseña un asistente con Rasa se debe definir el proceso llamado NLU pipeline. Este proceso consiste en añadir diferentes componentes para convertir los mensajes no estructurados provenientes del usuario en intenciones y entidades.

La Figura 3 muestra el proceso por el cual un mensaje es analizado para obtener la intención del usuario y extraer información clave.

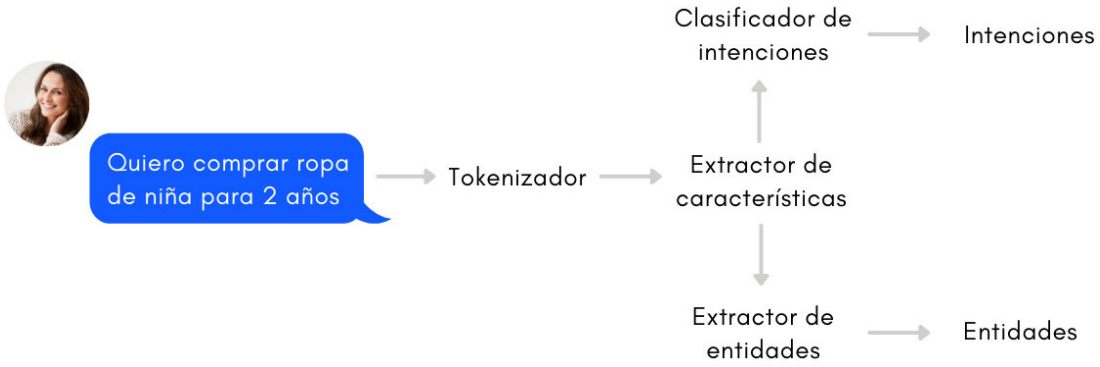


Figura 3 - Proceso de análisis de mensajes.

Los componentes que se añaden al proceso de clasificación de intenciones y extracción de información (entidades) son los siguientes:

- Tokenizador
- Extractor de características
- Clasificador de intenciones
- Extractor de entidades

A continuación, se describe la funcionalidad de cada componente dentro del proceso de análisis de mensajes:

1. **Tokenizadores**

El primer paso en el proceso de análisis de mensajes provenientes del usuario es la tokenización. Este componente se encarga de dividir el mensaje en pequeños fragmentos de texto, conocido como token. Cada token puede ser obtenido a través de una técnica o herramienta de tokenización. Por ejemplo, para la realización de este trabajo se utilizó la

técnica **WhiteSpaceTokenizer,** la cual separa las palabras identificando espacios en blanco [20].

La Figura 4 presenta el resultado de la técnica WhiteSpaceTokenizer para obtener un listado de palabras provenientes del mensaje. Los tokens resultantes son utilizados para la extracción de características. Es importante recalcar que muchas de las características que son extraídas se derivan de las palabras que del mensaje.

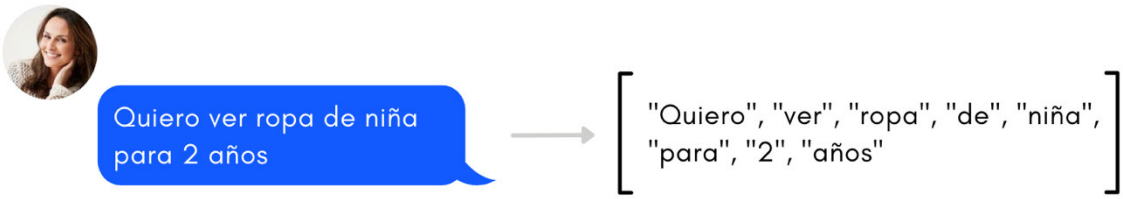


Figura 4 - Tokenización de una oración.

2. Extractor de características

La extracción de características es el proceso de transformar los tokens (palabras) en números o vectores representativos. Existen dos tipos de características que pueden ser extraídas de las palabras, las características dispersas y las características densas. Rasa permite a los desarrolladores que puedan elegir entre estos dos tipos de características, como también combinarlas libremente entre ellas.

Las características dispersas son generadas usualmente por la técnica **CountVectorizer**. Esta técnica recolecta las palabras y las contabiliza dependiendo la frecuencia de aparición en la oración sin importar el orden. Por otro lado, las características densas son generadas a partir de modelos pre-entrenados provenientes de la librería **SpaCyFeaturizer** o **Huggingface**. En el caso, de que se utilice modelos pre-entrenados es necesario leer la documentación respectiva de cada modelo y elegir el tokenizador compatible [21].

Además de tener características por cada palabra, también se tiene una característica correspondiente a la oración completa. El token que representa a todo el mensaje del usuario se le conoce como token **CLS**.

La característica del token CLS es obtenida a partir de la suma de las características dispersas y la suma ponderada de las características densas. En la Sección 2.3 se muestra los tipos de extractores de características que se utilizaron para crear el asistente propuesto.

La Figura 5 muestra una representación de cómo se genera las características correspondientes a cada palabra y a la oración completa.

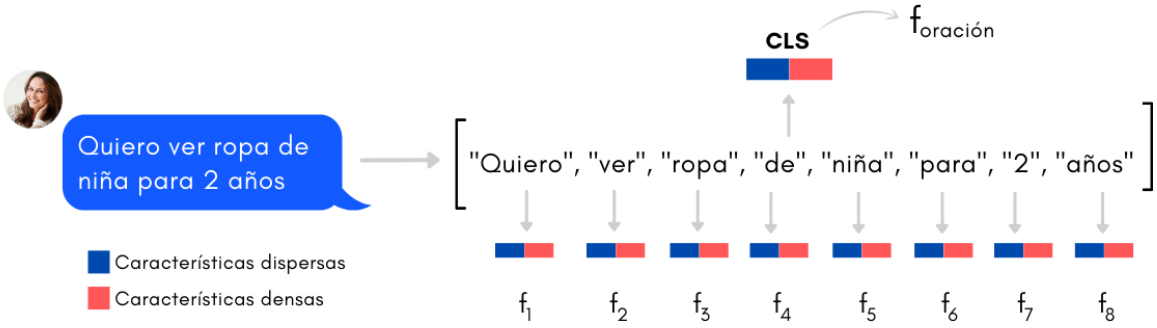


Figura 5 - Extracción de características dispersas y densas.

3. Clasificador de intenciones

Una vez que hayan extraído las características tanto para los tokens como para la oración completa, pasan a un modelo de clasificación de intenciones. Rasa recomienda utilizar el modelo **DIET (Dual Intent Entity Transformer)**, un modelo capaz de extraer fragmentos de información (entidades) de la oración y reconocer intenciones al mismo tiempo [22]. La Figura 6 muestra un diagrama de como las características generadas ingresan al modelo DIET para obtener intenciones y entidades.

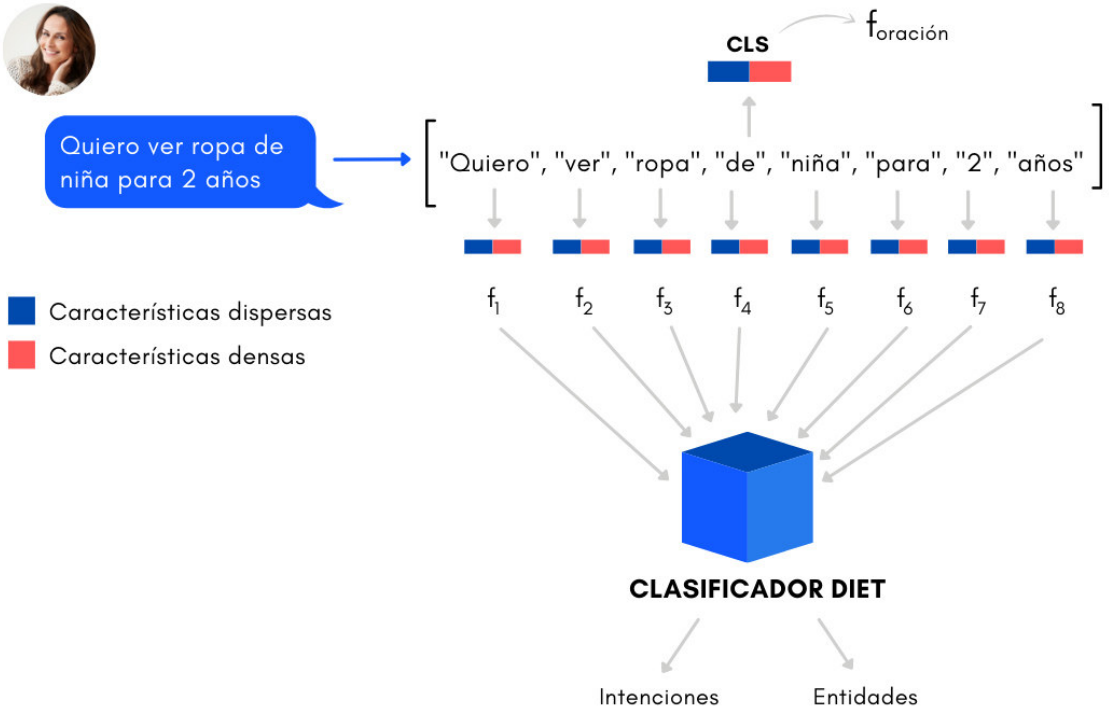


Figura 6 - Proceso para reconocer intenciones y extraer entidades.

El modelo DIET es un tipo de red neuronal tipo “Transformers” basada en la auto atención. Tiene una arquitectura multitarea que le permite clasificar intenciones y extraer entidades. DIET utiliza un modelo de secuencia que tiene en cuenta el orden de las palabras, por lo que ofrece un mejor rendimiento. También, es un modelo más compacto con una arquitectura modular plug-and-play, de modo que brinda la capacidad de conectar con modelos pre-entrenados como ConveRT, BERT, entre otros, mostrando así, resultados prometedores y sobre todo, flexibilidad en la experimentación.

4. Extractor de entidades

Como ya se mencionó anteriormente una entidad es un fragmento de información clave dentro de un mensaje de usuario. Para que las entidades puedan ser extraídas, se deben especificar en los datos de entrenamiento del modelo o definir expresiones regulares para extraerlas en función de un patrón de caracteres. A pesar de que el modelo DIET puede extraer entidades, no es recomendable utilizarlo para cada tipo de entidad [10]. Existen herramientas que pueden extraer entidades siguiendo un patrón estructurado. Por ejemplo, **DucklingEntityExtractor,** permite extraer entidades comunes como fechas, cantidades de dinero, distancias, pesos, entre otras.

La Figura 7 presenta la extracción de entidades a partir de un mensaje de usuario. En este caso, la entidad de *gender* y *age* (con sus respectivos valores niña y 2) fueron extraídas.

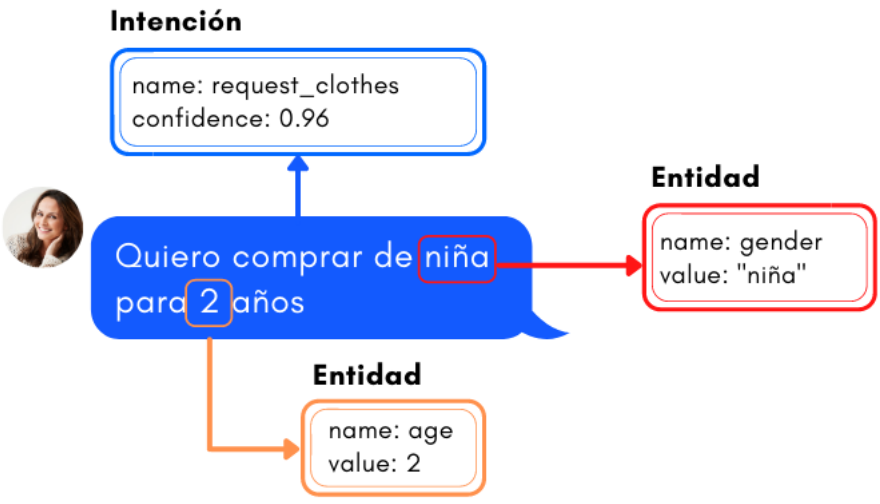


Figura 7 - Extracción de dos entidades en el mensaje de usuario.

Para realizar la configuración de este proceso, se debe crear o editar el archivo denominado `config.yml` añadiendo los distintos tipos de extractores de entidades qye fueron descritos anteriormente.

2.1.4 Rasa Core: Gestión de Diálogo

Rasa ha introducido en su arquitectura políticas para decidir la acción que debería tomar el asistente en cada instante de la conversación. Existen políticas complejas que se basan en aprendizaje automático y que toman en cuenta el contexto de la conversación. Por otro lado, existen políticas más simples las cuales utilizan solamente reglas. Las políticas desarrolladas en Rasa pueden combinarse entre ellas, en caso de que se requiera [23].

A continuación, se describe la característica de cada política que decide la acción que el asistente debería realizar en cada paso de la conversación:

1. **RulePolicy:** maneja las conversaciones que coinciden con reglas previamente definidas en el desarrollo del asistente.

2. **MemoizationPolicy:** asegura que el asistente siga las rutas de conversación de ejemplo que fueron detalladas en los datos de entrenamiento.

3. **Transformer Embedding Dialogue Policy (TEDPolicy) :** esta es una política predeterminada basada en aprendizaje automático. Dentro de su arquitectura tiene una red neuronal tipo “Transformer” que supera a las redes recurrentes debido a sus resultados cuando se tiene un corpus disperso o mensajes inesperados provenientes del usuario.

Las políticas mencionadas anteriormente se ejecutan en una jerarquía basada en prioridades como se muestra en la Figura 8. La configuración estándar que maneja Rasa menciona que RulePolicy se considera antes que MemoizationPolicy, que a su vez se considera antes que TEDPolicy.

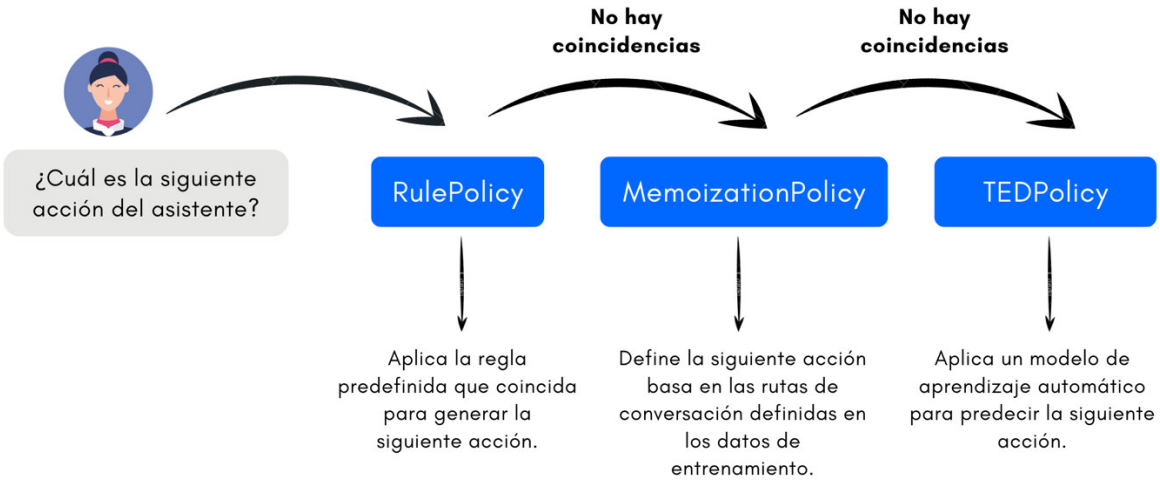


Figura 8 - Políticas de manejo de diálogo basadas en jerarquías.

Una vez que se ha dado a conocer los componentes principales de Rasa. Crear asistentes conversacionales inteligentes con este framework otorga un diseño sin tantas conversaciones de entrenamiento; ahorrando tiempo y dinero. Por esta razón, se decidió crear un asistente capaz de acompañar al usuario en todo el proceso de descubrimiento de ropa, respondiendo preguntas, asistiéndole y brindándole una experiencia personalizada a través de un punto de contacto. Además, dentro del desarrollo de dicho asistente se ha contemplado conectar con un motor de búsqueda llamado Algolia para mejorar la efectividad en la búsqueda de ropa solicitada y entregar resultados instantáneos.

La Figura 9 muestra el flujo del asistente conversacional diseñado para mostrar ropa y fomentar las ventas.

Figura 9 - Esquema del objetivo principal del asistente conversacional.

2.2 Preparación de datos

Esta sección presenta como se obtuvieron y prepararon los datos para la creación de la base de datos de ropa en Algolia. Así, también se indica el procedimiento para la creación del conjunto de entrenamiento correspondiente al asistente conversacional a través de la técnica conocida como **Mago de Oz**. A continuación, se describe el procedimiento que se realizó para crear la base de datos en Algolia como para desarrollar el corpus de entrenamiento del asistente conversacional.

2.2.1 Construcción de la base de datos de ropa en Algolia

Para la creación de la base de datos de ropa en Algolia se adoptó los pasos descritos en la Figura 10, que van desde solicitar acceso a la ropa del negocio hasta realizar las configuraciones idóneas para que el asistente sea capaz de encontrar la ropa requerida por el cliente.

Figura 10 - Pasos para la construcción de una base de datos de la ropa.

Pero antes, de explicar cómo se construyó la base de datos es importante definir lo que es Algolia. Algolia es un motor de búsqueda que brinda a los usuarios el poder de encontrar contenido en milésimas de segundos sin importar que tan complejo sea encontrarlo debido a su arquitectura basada en inteligencia artificial. A diferencia de otros motores de búsqueda, Algolia es compatible con varios lenguajes y frameworks de programación, otorgándole flexibilidad para ser implementada en cualquier solución.

Por lo tanto, para la creación de la base de datos, el paso primordial fue buscar un negocio que permitiera que sus prendas de vestir sean mostradas en Internet a través del asistente conversacional, en este caso “Creaciones Jasmin”.

Una vez, que se tuvo el acceso a ciertas prendas de vestir, se empezó con la sesión fotográfica, organización y etiquetado de la ropa de acuerdo con los aspectos relevantes que fueron consultados al dueño del negocio como categoría, género, edad, color y stock. Dentro del conjunto de ropa escogido, se tiene ropa de distintos tipos para niño y niña que va de 1 hasta 5 años. Por ejemplo, para mujer se tiene blusas, leggins y pijamas; por otro lado, para niños existe camisetas, pantalones y buzos. Toda la organización de ropa fue

plasmada en un archivo tipo ".csv"; de acuerdo con el modelo especificado por Algolia para ser cargado en sus servidores. Los datos deben cumplir ciertas especificaciones para que el motor de búsqueda pueda procesarlos y que los resultados sean optimizados en velocidad y calidad.

Para cargar los datos dentro de los servidores de Algolia, se creó una aplicación de acuerdo con las características de búsqueda. En este caso, se optó por el plan FREE que brinda un almacenamiento de 10000 registros y 10000 solicitudes de búsqueda por mes. Configurada la aplicación en Algolia, se continua con la carga de contenido, importando un total de 170 registros con sus respectivos atributos los cuales son descritos en la Tabla 1.

Tabla 1 - Descripción de los atributos de cada prenda de vestir.

Atributo	Descripción
objectID	Identificador único
name	Nombre descriptivo
category	Tipo de ropa (pijama, blusas, buzo, legging, camiseta, pantalón)
gender	Identificador de ropa de niño o niña
age	Identificador de talla
color	Identificador de color de ropa
price	Precio de la ropa

Con los datos cargados en Algolia se configura los atributos de búsqueda que otorgan un resultado más refinado (gender, age, color, category) y los atributos de visualización para las respuestas del asistente, es decir, la URL de la imagen, stock y precio.

Una vez configurada la búsqueda, se crea una instancia de cliente con las credenciales (appID & apiKey) otorgadas por Algolia para consumir datos a través de una API. Una API es una especie de definición o protocolo que permite conectar aplicaciones o servicios para intercambiar datos, en este caso la API conecta al motor de búsqueda Algolia con el asistente conversacional. En el Anexo II y III se muestra algunas fotografías de la ropa seleccionada que se incluyó en la base de datos.

2.2.2 Técnica del Mago de Oz (WoZ)

La técnica de Mago de Oz fue mencionada por primera vez por John F. Kelly en 1980, donde la intervención del ser humano en el flujo de trabajo de una aplicación de procesamiento natural de lenguaje fue introducida. Este método obtuvo su nombre de la historia **El maravilloso Mago de Oz**, donde una persona común se esconde detrás de una

cortina y pretende, mediante el uso de la tecnología, ser un mago poderoso. Basado en este método, los usuarios iniciales que interactúan con el asistente, aparentemente autónomo, le escriben y cuyas funciones que aún no han sido implementadas en realidad son simuladas por un operador humano conocido como mago. El usuario no sabe esto y piensa que está interactuando con una computadora, lo que lo convierte en el entorno de prueba perfecto para obtener resultados válidos. La Figura 11 muestra la técnica de mago de oz, en la cual intervienen el usuario, el desarrollador y el asistente conversacional.

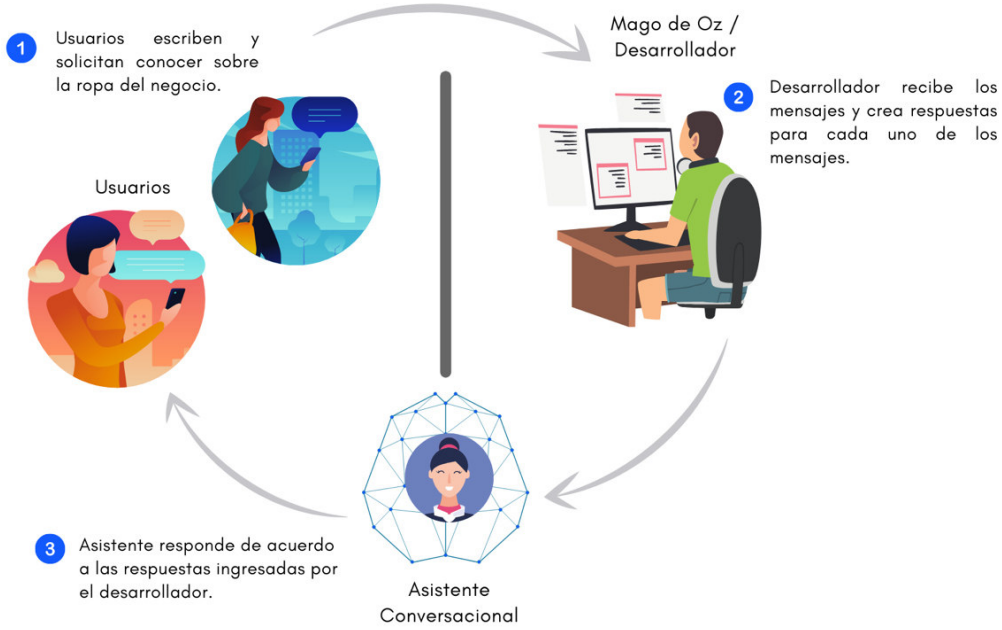


Figura 11 - Diagrama de la técnica del Mago de Oz.

Esta técnica fue aplicada para crear el primer conjunto de entrenamiento y así generar el modelo junto con la planificación de las posibles rutas de conversación. El conjunto de entrenamiento inicial contiene alrededor de 600 ejemplos clasificados para cada una de las intenciones propuestas. En la Sección Resultados se presenta, los resultados del primer modelo entrenado con los datos obtenidos al aplicar este método.

2.3 Diseño

Para realizar el diseño del asistente conversacional primero se definieron los alcances de acuerdo con los objetivos establecidos en este trabajo de titulación. Por lo tanto, el alcance del asistente conversacional conlleva las siguientes funcionalidades:

- El asistente debe entender los saludos y despedidas para responder con un mensaje adecuado según corresponda.

- El asistente debe realizar búsquedas de ropa bajo los parámetros pre-establecidos que ha seleccionado el usuario y ejecutar una llamada a la acción para la compra del producto.
- El asistente debe ser capaz de dar las gracias en caso de que se le agradezca y devolver un mensaje por defecto en caso de no haber entendido alguna entrada de texto inesperada.

A las funcionalidades mencionadas anteriormente se le añadieron actividades complementarias que tienen como propósito que el asistente pueda mostrar detalles del negocio, obtener un registro de las personas que realizan búsquedas y conseguir información para el mejoramiento futuro de dicho asistente.

Las actividades complementarias que se establecieron son las siguientes: solicitar el nombre y el correo electrónico al usuario que por primera vez le escribe al asistente, tener la posibilidad de responder a ciertas preguntas frecuentes respecto al negocio, guardar retroalimentaciones del usuario para mejoras futuras y enviar un correo de agradecimiento al generar una búsqueda exitosa.

La Figura 12 ilustra las funcionalidades propuestas en color azul y las actividades complementarias que se añadieron en color morado al asistente.

Figura 12 - Habilidades del asistente conversacional propuesto.

Para realizar el diseño e implementación del asistente conversacional a través de framework de desarrollo llamado Rasa se ejecutaron los siguientes pasos:

2.3.1 Instalación

Antes de realizar el diseño del asistente conversacional es necesario instalar Rasa con todas sus dependencias. Es recomendable crear un entorno virtual para aislar el proyecto de otros ya existentes e instalar las librerías requeridas. Para realizar la instalación se necesita tener instalado previamente Python y Pip, el cual es un sistema de gestión de paquetes para administrar librerías de Python. Los pasos para instalar Rasa son los siguientes:

1. Revisar los paquetes previos como Python y Pip:
   ```
   python3 --version
   pip3 –version
   ```

2. Crear un entorno de virtual para instalar Rasa y las dependencias adjuntas (Opcional):
   ```
   python3 -m venv ./venv
   source ./venv/bin/activate
   ```

3. Instalar Rasa Open Source
   ```
   pip3 install rasa
   ```

En el caso, de utilizar otros algoritmos de aprendizaje de máquina que no sean propios de Rasa se debe instalar paquetes adicionales de Python a través de Pip. Por ejemplo, para que este asistente pueda extraer fácilmente los nombres propios de persona se instaló Spacy como herramienta de extracción de entidades en este caso particular.

```
pip3 install spacy
```

4. Para iniciar con el desarrollo y diseño del asistente conversacional se debe crear un proyecto de Rasa a través del siguiente comando:
   ```
   rasa init
   ```

Con este comando se crea los archivos necesarios para diseñar el asistente y un ejemplo simple de cómo usarlo. El formato de los archivos es Yet Another Markup Language, (YAML, por sus siglas en inglés), un lenguaje de marcado que facilita la escritura y edición

del documento al momento de desarrollar el asistente. La Tabla 2 presenta la descripción de los archivos que se generan cuando se crea un nuevo proyecto en Rasa.

Tabla 2 - Descripción de los archivos para configurar el asistente.

Archivo	Descripción
`__init__.py`	Archivo vacío que facilita encontrar los archivos de Python a través del editor de código.
`action.py`	Archivo para crear las acciones personalizadas del asistente.
`config.yml`	Archivo para configurar el proceso de NLU y la gestión de diálogo.
`credentials.yml`	Archivo para conectar con otros servicios.
`data/nlu.yml`	Archivo que contiene los datos de entrenamiento.
`data/stories.yml`	Archivo que contiene conversaciones de ejemplo para entrenamiento.
`domain.yml`	Archivo que define el mundo del asistente, lo que conoce, lo que entiende y lo que puede hacer.
`endpoints.yml`	Archivo para configurar los canales de contacto con los usuarios.
`models/<timestamp>.tar.gz`	Archivo comprimido que contiene el modelo inicial.

Una vez descrito los archivos de Rasa, en primer lugar, se necesita crear el conjunto de entrenamiento para el NLU. Existen herramientas de generación de texto para aumentar de una manera más rápida el corpus de entrenamiento. Sin embargo, esta alternativa no es recomendada por Rasa, ya que los mensajes sintéticos generados no se parecen a los mensajes de usuarios que realmente envían, por lo tanto, el rendimiento del modelo será inferior. Para evitar este inconveniente se optó por recopilar la mayor cantidad de mensajes de usuarios reales que han escrito a algunos chats del negocio. Además, se aplicó la técnica del Mago de Oz y se añadieron mensajes adicionales para usarlos como datos de entrenamiento, de tal manera, que el corpus sea más robusto. Generalmente, los mensajes provenientes de usuarios reales suelen ser confusos en algunos casos, pueden contener errores ortográficos y no parecerse a ejemplos “ideales” de las intenciones que se han determinado. Sin embargo, con estos mensajes, permitirá crear un modelo más eficiente ante escenarios del mundo real. Por otro lado, el código completo se encuentra en el siguiente repositorio de GitHub, de modo que sirva como una guía de diseño de asistentes conversacionales con herramientas de código abierto.

(Enlace: https://github.com/kabirivan/Ecommerce-Assistant-Jasmine)

2.3.2 Intenciones y entidades

En el archivo `nlu.yml` se define los posibles mensajes que podrían escribir los usuarios al asistente. Esto permite que el asistente sea capaz de identificar la intención y extraer información importante del mensaje del usuario. Al añadir los mensajes de entrenamiento se debe indicar la intención a la que corresponde y las entidades que se van a extraer. Además, se tiene dos apartados adicionales como **Lookup Tables** que permite mejorar el reconocimiento de entidades basado en coincidencias. Por ejemplo, se utilizó Lookup Tables para reconocer la mayor cantidad de colores. Por otro lado, el apartado de **Synonyms**, se utilizó para asignar ciertos ejemplos de entidad a una única entidad normalizada. Por ejemplo, algunos usuarios a la ropa le dicen ropita, prendas, atuendo, etc.

La Figura 13 muestra un fragmento de código donde se define las intenciones y entidades para crear el conjunto de entrenamiento del modelo de NLU.

```yaml
version: "2.0"
nlu:
- intent: request_clothes
examples: |
- quiero comprar ropa
- que colores no mas tiene
- quiero comprar ropa de [niño](gender)
- me gustaria comprar [pantalones](category) para [niñas](gender)
- intent: inform
examples: |
- muestrame ropa [niña](gender)
- muestrame ropa [niño](gender)
- muestrame ropa de color [azul](color)
- muestrame ropa de color [amarillo](color)
- synonym: niña
examples: |
- nina
- niñas
- ninas
- mujer
- mujeres
- mujercitas
- lookup: color
examples: |
- rojo
- amarillo
- azul
```

Figura 13 - Ejemplo de intenciones y entidades del conjunto de entrenamiento.

En cuanto, a los datos de conversación que se requiere, contienen historias y reglas que permiten entrenar el modelo para la gestión de diálogo. Tener conversaciones bien definidas permite que el asistente siga de manera confiable las rutas de conversación establecidas y generalizar las rutas inesperadas. Al diseñar historias es importante considerar las rutas unHappy, estas rutas son aquellas en donde el usuario se desvía con otras preguntas de la conversación que se está llevando a cabo. Poco a poco, estas rutas unHappy deben ser integradas para mejorar la efectividad en las respuestas del asistente.

En el archivo `stories.yml` se añade las posibles conversaciones reales que pueden existir entre el asistente y el usuario para encontrar el producto requerido (Ver Figura 14). Para crear una conversación de ejemplo se debe añadir la intención correspondiente al mensaje de usuario seguido de la respuesta a dicha intención. Las respuestas a las intenciones se definen en el archivo `domain.yml` y se detalla en la Sección de dominio.

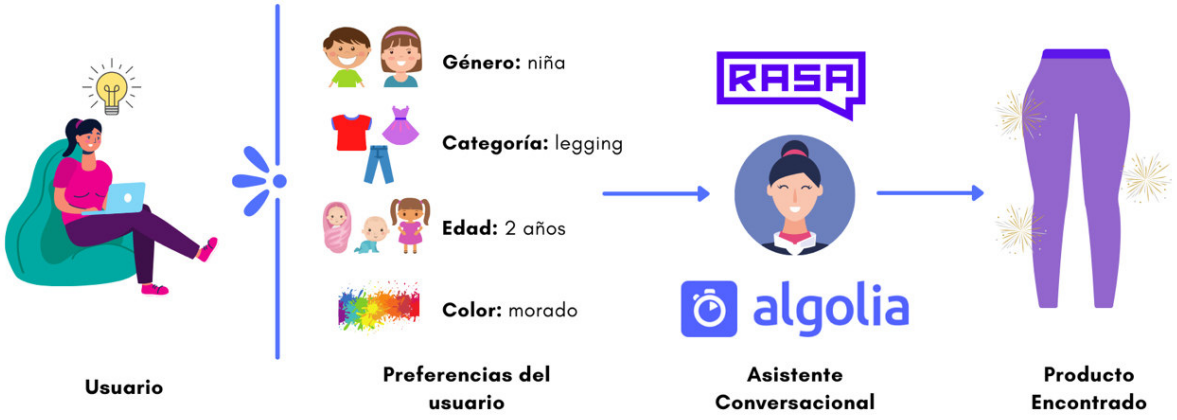


Figura 14 - Interacción entre el usuario y asistente para encontrar un producto.

Por otro lado, las reglas permiten al asistente gestionar pequeños fragmentos de conversación que deben seguir la misma ruta. En este caso, las reglas se utilizaron para asignar respuestas fijas a mensajes de saludar, despedirse y dar las gracias. Las reglas fueron definidas en el archivo `rules.yml`

Para encontrar un producto requerido por parte del usuario a través de sus preferencias se han establecido cuatro casos de conversación que pueden suscitarse en una primera versión del asistente propuesto. A continuación, se detallan las rutas de conversación que se consideraron para diseñar el asistente:

- **Caso 1: Usuario quiere comprar ropa**

En este caso el usuario requiere comprar ropa, pero en su mensaje no brinda ninguna información adicional de lo que desea. Por lo tanto, el asistente solicita la información necesaria para obtener los parámetros de búsqueda y generar resultados. Los parámetros de búsqueda que se definieron para encontrar una prenda de vestir son el género, categoría, edad y color (opcional). Una vez que el asistente haya conseguido dicha información se comunica a través de una Api con Algolia y se realizan la búsqueda determinada. Luego de obtener los resultados, el asistente presenta la ropa junto con un llamado a la acción para la compra. Esto con el objetivo de generar más conversiones de venta. La Figura 15 ilustra un ejemplo de conversación para obtener información de la ropa sin que el usuario le haya dicho alguna característica previamente al asistente.

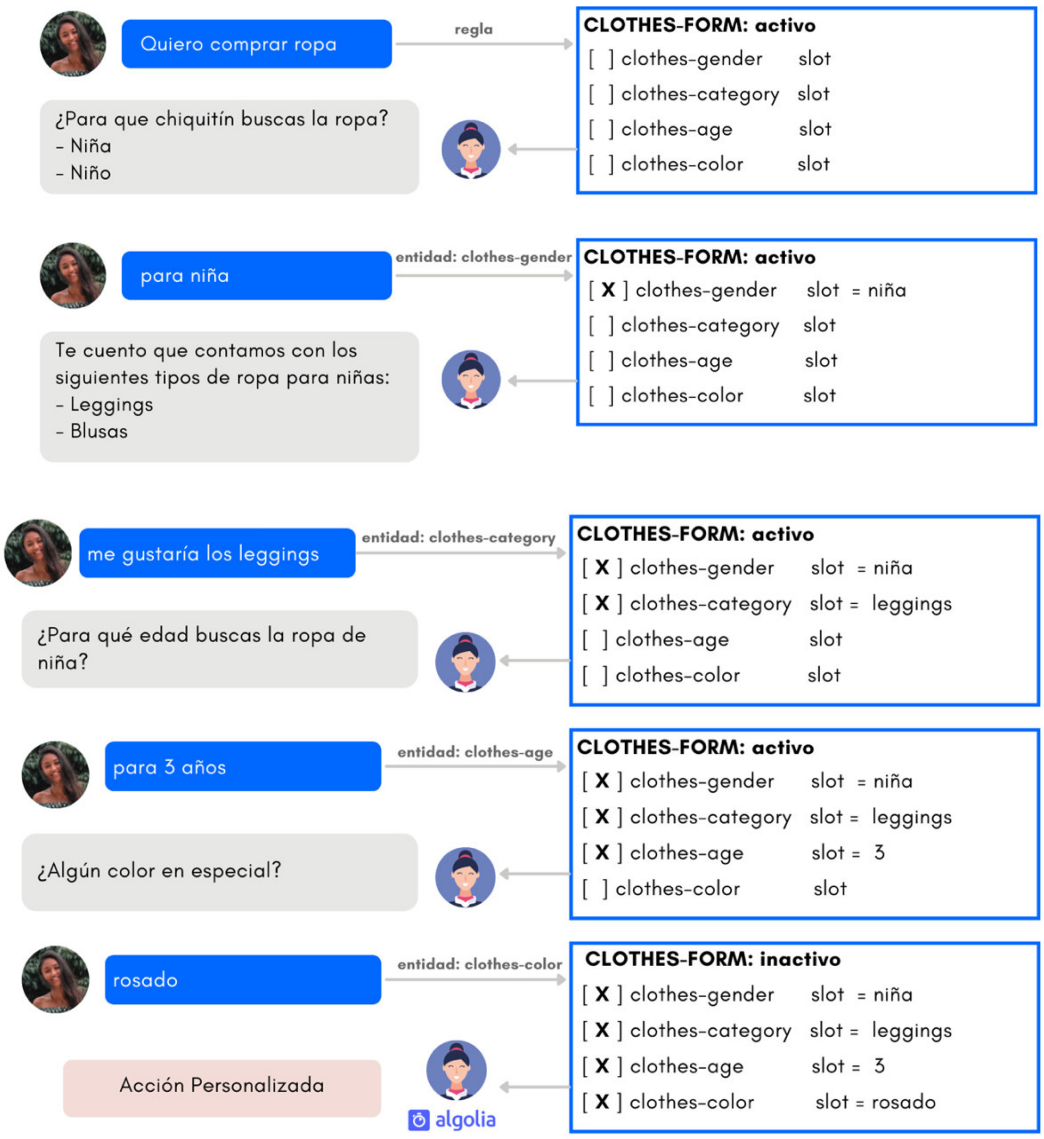


Figura 15 - Caso 1: Usuario quiere comprar ropa.

- **Caso 2: Usuario quiere comprar ropa con características previas**

Para este caso, el usuario empieza la conversación solicitándole al asistente ropa con características de antemano. Como el usuario, ya ha brindado cierta información, el asistente solamente realiza las preguntas necesarias para completar los parámetros de búsqueda faltantes. La Figura 16 presenta un ejemplo de conversación donde el usuario requiere una prenda con ciertas características de primera mano.

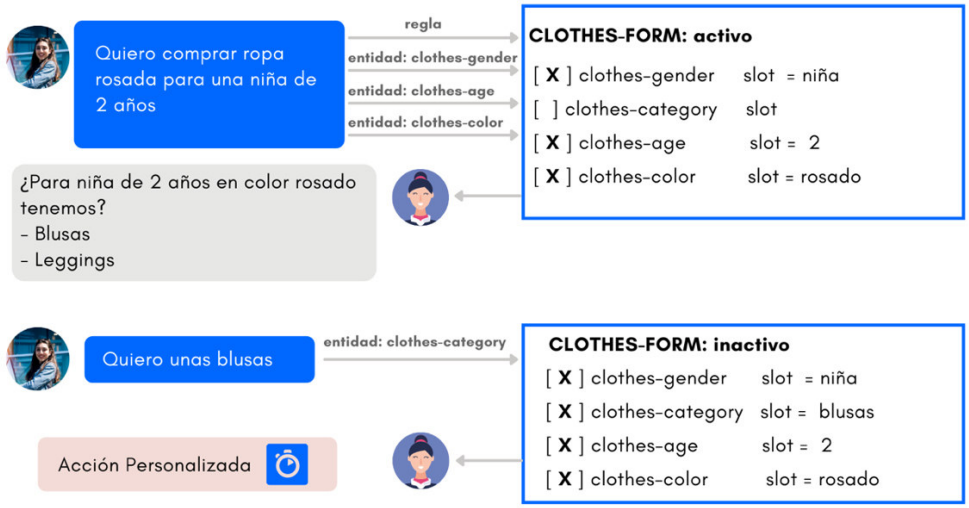


Figura 16 - Caso 2: Solicitud de ropa con ciertas características definidas.

- **Caso 3: Usuario ingresa valores fuera del alcance del asistente**

En esta ocasión, el usuario solicita un producto con ciertas características. Mientras el asistente realiza las preguntas para obtener la información faltante, puede que el usuario brinde una respuesta que no esté dentro de los alcances de búsqueda definidos. Entonces, el asistente devuelve un mensaje de ayuda diciéndole que el parámetro ingresado no está disponible y sugiriéndole opciones para completar la información (Ver Figura 17).

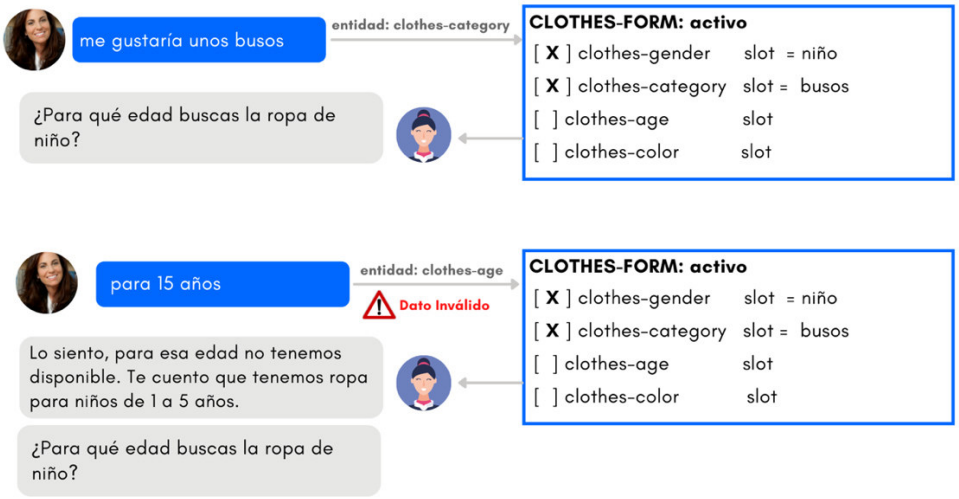


Figura 17 - Caso 3: Solicitud de ropa con características fuera del alcance de búsqueda.

- **Caso 4: Usuario no quiere continuar con la conversación**

Normalmente, el asistente ya no realiza preguntas cuando el usuario haya proporcionado todos los parámetros de búsqueda requeridos. Pero en el caso, de que el usuario quiera terminar de forma prematura la conversación, el asistente debe estar preparado para manejar esta situación cuando se da de forma inesperada. Entonces, el asistente genera una respuesta y cancela la búsqueda. La Figura 18 muestra el flujo de la conversación terminada de formar inesperada por parte del usuario.

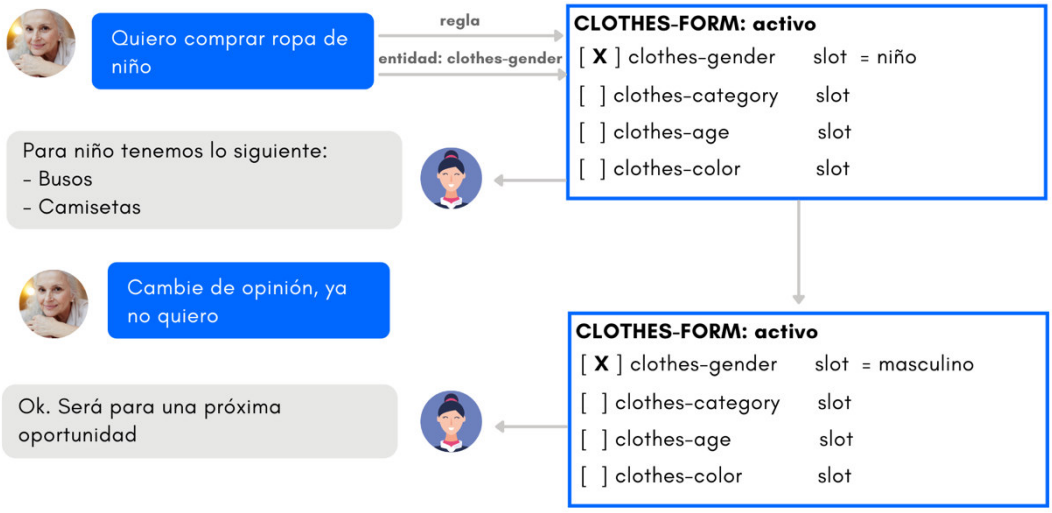


Figura 18 - Caso 4: Usuario termina la conversación de forma inesperada.

Hasta este momento, el asistente propuesto sería capaz de manejar solamente mensajes provenientes de usuarios basados en el contexto de la conversación. Para complementar las habilidades del asistente, se han añadido acciones personalizadas que permitirán buscar ropa, conectarse con servicios en la nube, etc. Las acciones personalizadas propuestas se detallan a continuación:

2.3.4 Acciones personalizadas

Las acciones personalizadas se refieren a aquellas acciones que no pueden ser manejadas directamente con los archivos descritos anteriormente por Rasa y necesitan un código más específico y personalizado. Cuando un asistente requiere realizar una acción personalizada envía una solicitud a un servidor de acciones. A través de este servidor se puede realizar llamadas a APIs, cálculos, consultar bases de datos, etc. Para realizar una acción personalizada se debe escribir código en Python en el archivo `actions.py`. Las acciones personalizadas que se definieron para el asistente son las siguientes:

- **Búsqueda de ropa:** Servicio creado para buscar ropa a través de parámetros de búsqueda con Algolia.
- **Registro de ropa:** Servicio creado para guardar información de las búsquedas de ropa para cada usuario con AirTable.
- **Registro de comentarios/sugerencias:** Servicio que permite guardar los comentarios de cada usuario con AirTable.
- **Envío de correos:** Servicio que se utiliza para enviar correos electrónicos a través del Protocolo simple de transporte de correo, (SMTP) después de que el usuario haya dejado un comentario.

2.3.5 Dominio

El archivo más importante dentro de la creación y diseño de un asistente conversacional es `domain.yml`. En este archivo se define toda la información que ya se estableció en los anteriores archivos como las intenciones, entidades, acciones, formularios y respuestas. También, se define la configuración para las sesiones de conversación.
Dentro del archivo `domain.yml`, las intenciones como las entidades son listadas. Además, aparecen otros apartados como los slots, los cuales son espacios de memoria del asistente que sirve para guardar información de las conversaciones. Por ejemplo, el nombre del usuario, las características del producto y los comentarios de retroalimentación fueron utilizados para que se mantenga una conversación personalizada. Además, se define las respuestas para cada una de las intenciones definidas y finalmente se configura el tiempo de duración de la conversación realizada entre el usuario y el asistente con un tiempo igual a 60 minutos.

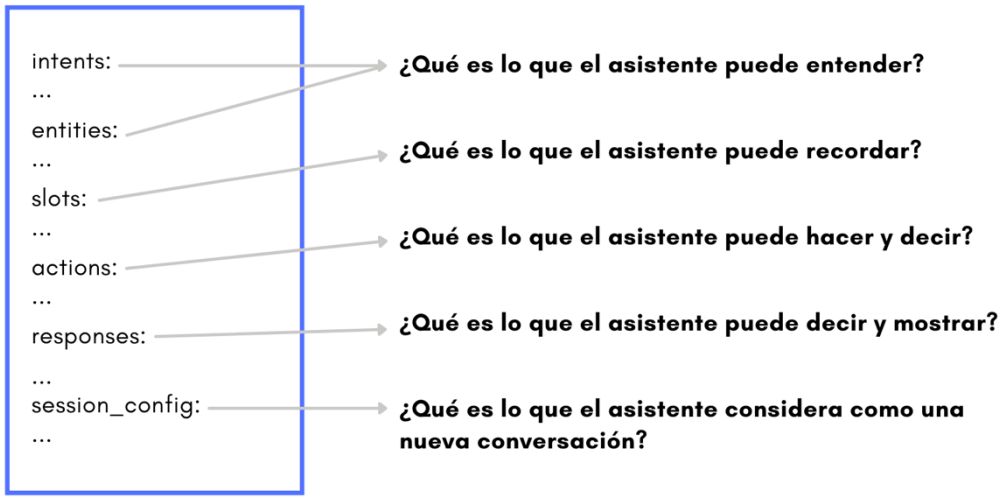


Figura 19 - Descripción del archivo que describe el universo donde opera el asistente.

2.3.6 Endpoints

Gracias a las ventajas que proporciona Rasa debido a su estructura, permite que la comunicación con servicios externos sea simple; similar a acceder a micro servicios. A través de la configuración del archivo `endpoints.yml` se configuró el acceso a las bases de datos y al servidor de acciones apuntando a las rutas correspondientes.
En modo desarrollo, se configuró solamente el **servidor de acciones** para realizar acciones personalizadas como las que se describieron anteriormente. Por otro lado, para el despliegue en modo producción, se configuró aparte del servidor de acciones lo siguientes servicios que permitirán manejar una gran cantidad de mensajes:

- **Event Broker:** Servicio para procesar los datos de las conversaciones de forma asíncrona.
- **Tracker Store:** Servicio para almacenar las conversaciones de los usuarios.
- **Lock Store:** Servicio de bloqueo para garantizar que los mensajes de los usuarios sean procesados en orden correcto.

2.3.7 Canales de comunicación

Para tener un punto de contacto con los usuarios cuando el asistente sea desplegado en modo producción se definió un canal tipo WebSocket como punto de contacto. La librería que utiliza Rasa para este tipo de canal es SocketIO, la cual funciona en tiempo real. Además, se configuró una autentificación a través de JWT para que la conexión sea segura. Las configuraciones se realizan en el archivo `credentials.yml` (Figura 20), donde los dos primeros parámetros representan los eventos usados por Rasa cuando se envía o recibe mensajes. Además, el parámetro session_persintence es colocado como true para mantener las conversaciones de los usuarios en su navegador y que no desaparezcan cuando se recargue la página web. Los dos últimos parámetros representan la configuración de la autenticación a través de JWT, añadiendo el método JWT y una llave de seguridad secreta. Gracias a esta configuración, se garantiza el envío seguro de información.

```
socketio:
user_message_evt: user_uttered
bot_message_evt: bot_uttered
session_persistence: true
jwt_key: my_secret_key
jwt_method: HS256
```

Figura 20 - Configuración del canal de contacto.

2.4 Evaluación

Realizar una evaluación es clave para construir asistentes confiables, pero primero se necesita entrenar un modelo a través de conversaciones y datos correspondientes al NLU. Para realizar el entrenamiento es importante haber definido la estructura del modelo (NLU Pipeline), el cual está compuesto de elementos que procesan los mensajes entrantes y que pueden ser ajustados de acuerdo con la personalización y alcance que se requiera.

2.4.1 Configuración del NLU Pipeline

Como se había mencionado en la Sección 2.1.3, existen componentes para clasificar intenciones, detectar y extraer entidades, seleccionar respuesta, entre otros. La selección de los componentes para el análisis de mensajes, depende del propósito y el ambiente en donde se va a desenvolver el asistente.

Para este estudio, los componentes elegidos para procesar los mensajes de usuario se describen en la Tabla 2 y son añadidos en el archivo `config.yml` como se había señalado anteriormente:

Tabla 2 - Descripción del NLU Pipeline utilizado.

NLU Pipeline	Componentes	Descripción
Tokenizador	WhiteSpaceTokenizer	Usa espacios en blanco como separador de palabras
Extractor de características	RegexFeaturizer	Crea una representación vectorial del mensaje del usuario mediante expresiones regulares.
	LexicalSyntacticFeaturizer	Crea características léxicas y sintácticas para un mensaje para permitir la extracción de entidades.
	CountVectorsFeaturizer	Crea una bag-of-words de mensajes de usuario, intenciones y respuestas.
Clasificador de intenciones	DIETClassifier	Clasifica intenciones
	FallbackClassifier	Clasifica intenciones si las puntuaciones de clasificación de intención de NLU son ambiguas.
Extractor de entidades	DIETClassifier	Extrae de entidades
	EntitySynonymMapper	Mapea sinónimos de las entidades definidas
	DucklingEntityExtractor	Extrae entidades de número y correos electrónicos (basada en patrones)
	SpacyEntityExtractor	Extrae entidades de nombres de personas

2.4.2 Configuración de las políticas para la gestión de diálogo

El asistente utiliza políticas para decidir qué respuesta brindar al usuario en cada paso de la conversación. Como se mencionó anteriormente, existen políticas basadas en reglas y en aprendizaje automático que el asistente puede hacer uso.

Por lo tanto, las políticas seleccionadas para configurar el sistema de gestión de diálogo se detallan en la Tabla 3:

Tabla 3 - Políticas utilizadas en el sistema de gestión de diálogo.

Políticas	Descripción
RulePolicy	Maneja partes de la conversación que siguen un comportamiento fijo
AugmentedMemoizationPolicy	Maneja partes de la conversación basada en las conversaciones de ejemplo del conjunto de entrenamiento
TEDPolicy	Maneja la conversación a través de prediciones basadas en un modelo de aprendizaje automático

La configuración de las políticas es llevada a cabo en el archivo `config.yml` como ya se detalló anteriormente.

2.4.3 Entrenamiento y evaluación del modelo.

Una vez configurado el proceso de entrenamiento, tanto para la NLU como para la gestión de diálogo, es necesario realizar una verificación para que no exista errores e inconsistencias en los datos añadidos en los archivos de la NLU, historias y dominio (`nlu.yml`, `stories.yml`, `domain.yml`). Si la validación de datos tiene errores, el entrenamiento del modelo puede fallar o generar un mal rendimiento. Para realizar la validación de datos, se aplica el siguiente comando:

```
rasa data validate
```

Por lo tanto, el comando para entrenar tanto la NLU como el sistema de gestión de diálogos es el siguiente:

```
rasa train
```

Realizado el entrenamiento, se genera un archivo comprimido (`<timestamp>.tar.gz`) en la carpeta models del proyecto de Rasa, con el cual el asistente ya podrá realizar inferencias en una conversación.

Para evaluar la gestión de diálogo se creó historias de prueba (Archivo: `test_stories.yml`) que permitan evidenciar como se comportará el asistente en ciertas situaciones. Las historias de prueba son parecidas a las historias añadidas en los datos de entrenamiento, pero con una pequeña variación, se debe incluir el mensaje del usuario. El comando para evaluar el modelo de gestión de diálogo es:

```
rasa test core –stories test_stories.yml
```

Por otro lado, para evaluar la NLU se dividió el conjunto de entrenamiento a través de la técnica de validación cruzada. Rasa establece por defecto cinco carpetas. Los datos de entrenamiento se dividen en grupos del mismo tamaño. El rendimiento del modelo se mide tomando una muestra de todas las puntuaciones del modelo para cada conjunto de entrenamiento. El comando para realizar la evaluación es el siguiente:

```
rasa test nlu –cross-validation
```

El análisis y discusión de los resultados correspondientes a la evaluación del modelo NLU para clasificación y extracción de entidades del asistente diseñado serán presentados en la Sección 3.

2.5 Despliegue

Terminada la evaluación del asistente, empieza la fase de despliegue. En este caso, el despliegue fue dividido en dos modos, desarrollo y producción. Cada despliegue tiene un propósito, los cuales serán descritos a continuación:

2.5.1 Despliegue: Modo Desarrollo

El despliegue del asistente en modo desarrollo fue realizado en una computadora con sistema MacOS de 2,4 GHz Quad-Core Intel Core i7, 16GB de RAM y 520GB SSD. El objetivo de este despliegue es verificar las rutas de conversación diseñadas junto con las acciones personalizadas que se han añadido al asistente, las cuales se conectan con servicios externos de la nube. La Figura 21 muestra la arquitectura del asistente

desplegado en modo desarrollo con Docker; una herramienta que crea contenedores portables para aplicaciones de software y que puede ejecutarse en cualquier máquina, sin importar el sistema operativo. Como se había mencionado anteriormente, Rasa está comprendida del sistema de gestión de diálogo (Rasa Core) para brindar una respuesta acertada dependiendo el contexto de la conversación y el NLU (Rasa NLU) que permite clasificar intenciones y extraer entidades. En cuanto, a las acciones personalizadas se incluye el uso de un servidor SMTP para envío de correos a los usuarios que han realizado búsquedas de ropa. Para implementar esta característica fue necesario crear una cuenta de Gmail. Por otro lado, el servicio de AirTable se usa para almacenar los comentarios que escriben los usuarios acerca del asistente y ciertas búsquedas de ropa para un análisis futuro. Para hacer uso de este servicio, fue necesario crear una cuenta y configurar tablas de almacenamiento para guardar los datos. Al igual que Algolia, AirTable se comunica a través de APIs con el asistente para guardar y leer información. Finalmente, el servicio de Algolia permite las búsquedas de ropa de forma rápida y eficiente. La configuración de este servicio se mencionó en la Sección 2.2.1.

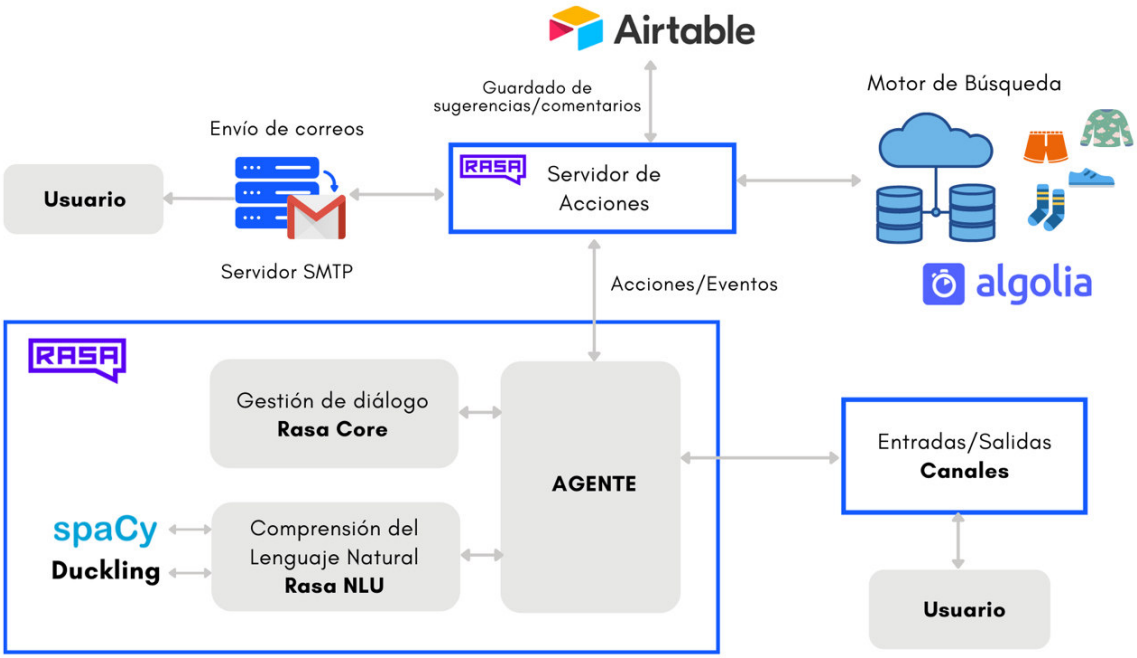


Figura 21 - Arquitectura del asistente en modo desarrollo.

Para interactuar con el asistente en modo local y empezar con la revisión y ajuste de las rutas de conversación se ejecutó el siguiente comando:

```
rasa interactive
```

2.5.2 Despliegue: Modo Producción

Una vez que se ha desplegado el asistente localmente y se haya comprobado que las tareas especificadas en el diseño funcionen correctamente, es momento de que el asistente tenga sus primeras conversaciones con usuarios reales. El despliegue en modo producción del asistente es realizado de igual manera con Docker. Pero, en este caso, se añadieron contenedores adicionales. Los contenedores que se tomaron en cuenta para realizar el despliegue del asistente y que pueda interactuar con usuarios reales se describe en la Tabla 4 y presentan en la Figura 22:

Tabla 4 - Descripción de los contenedores para el despliegue del asistente.

No.	Contenedores	Descripción
1	Rasa (NLU & Core)	Procesa mensajes de usuario entrantes
2	Lock Store	Garantiza que las conversaciones sean procesadas en orden correcto
3	Servidor de acciones	Tareas personalizadas que realiza el asistente
4	Event Broker	Procesar conversaciones de forma asíncrona
5	Duckling	Extractor de entidades de emails y números
6	Rasa X	Herramienta para procesar mensajes y mejorar el asistente
7	SQL Store	Base de datos de Rasa X
8	NGINX	Proxy inverso utilizado para redirigir solicitudes a los diferentes servicios

Con la extensión de Docker conocida como Docker Compose se ejecutó múltiples contenedores al mismo tiempo. El archivo `docker-compose.yml` describe la configuración de los contenedores y los servicios, de tal manera que puedan comunicarse entre sí.

El contenedor revelación que emerge en el despliegue en modo producción es Rasa X. Esta herramienta tiene como objetivo ayudar a los desarrolladores para la mejora continua del asistente. El desafío de crear excelentes asistentes conversacionales con inteligencia artificial es que es imposible anticiparse a todo lo que dirán los usuarios. Por esta razón, Rasa X se enfoca en el análisis y revisión de conversaciones reales que se han generado entre el usuario y el asistente. Este método es conocido como Conversation-Driven Development (CDD, por sus siglas en inglés,), el cual fue estudiado en la Sección 1.4.4.

Aparte de crear toda la arquitectura del asistente con Docker, se utilizó varios servicios adicionales de la nube para el despliegue. Por ejemplo, para la instancia donde se aloja el

asistente se compró un servidor en SSNODES de 8GB de RAM, 160GB SSD y 2 CPU, requerimientos que solicita Rasa para desplegar un asistente. Por otro lado, para almacenar las conversaciones de forma segura que mantiene el asistente con los usuarios se utilizó AWS con su servicio RDS, correspondiente a bases de datos relacionales. Finalmente, los servicios externos detallados anteriormente en el despliegue local fueron conectados: el servidor SMTP para envío de correos, AirTable para guardar sugerencias y ciertas búsquedas y finalmente el servicio de Algolia para buscar la ropa también.

La Figura 22 ilustra la arquitectura del asistente en modo producción.

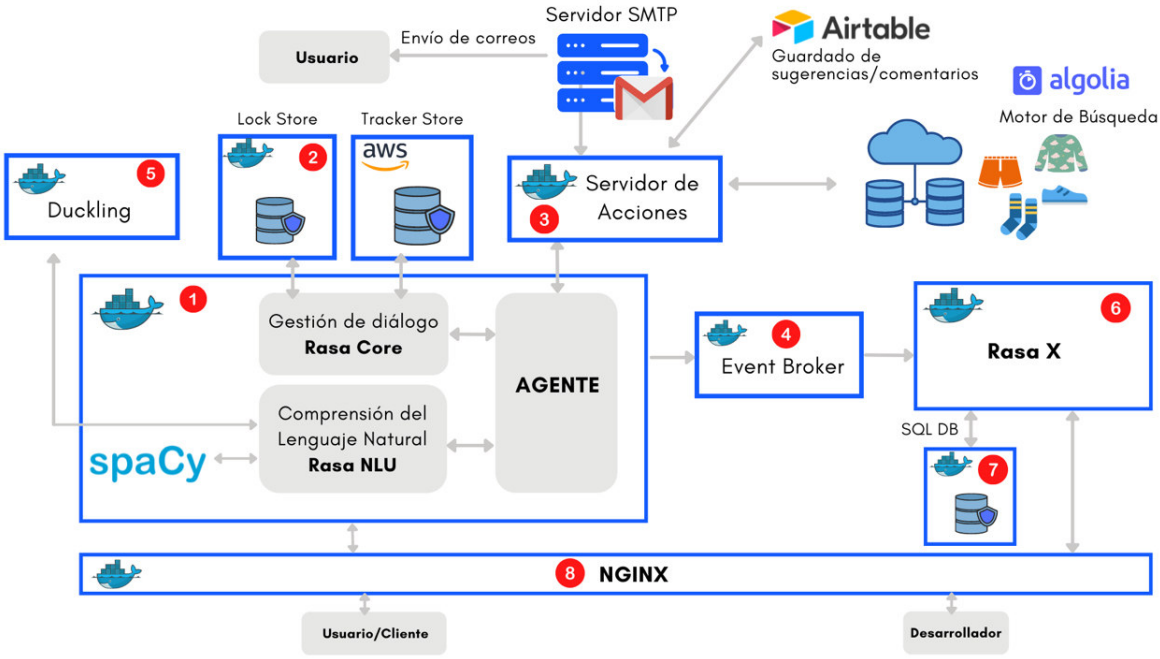


Figura 22 - Arquitectura del asistente en modo producción.

Para que los usuarios puedan interactuar con el asistente conversacional desplegado en modo producción se conectó con una página web estática creada en Gatsby. El framework conocido como Gatsby es un generador de sitios estáticos en base a datos dinámicos La interfaz de la página y del asistente se verá en la sección de Resultados.

3. RESULTADOS Y DISCUSIÓN

En esta sección se detalla y analiza los resultados obtenidos en las distintas evaluaciones que se realizaron al asistente desarrollado. El tamaño del corpus inicial de entrenamiento tiene alrededor de 623 ejemplos repartidos en cada una de las intenciones propuestas (16 intenciones). Estos resultados fueron obtenidos antes de que el asistente sea desplegado y probado con usuarios reales.

La Figura 23 muestra la matriz de confusión del modelo preliminar para la clasificación de intenciones y extracción de entidades.

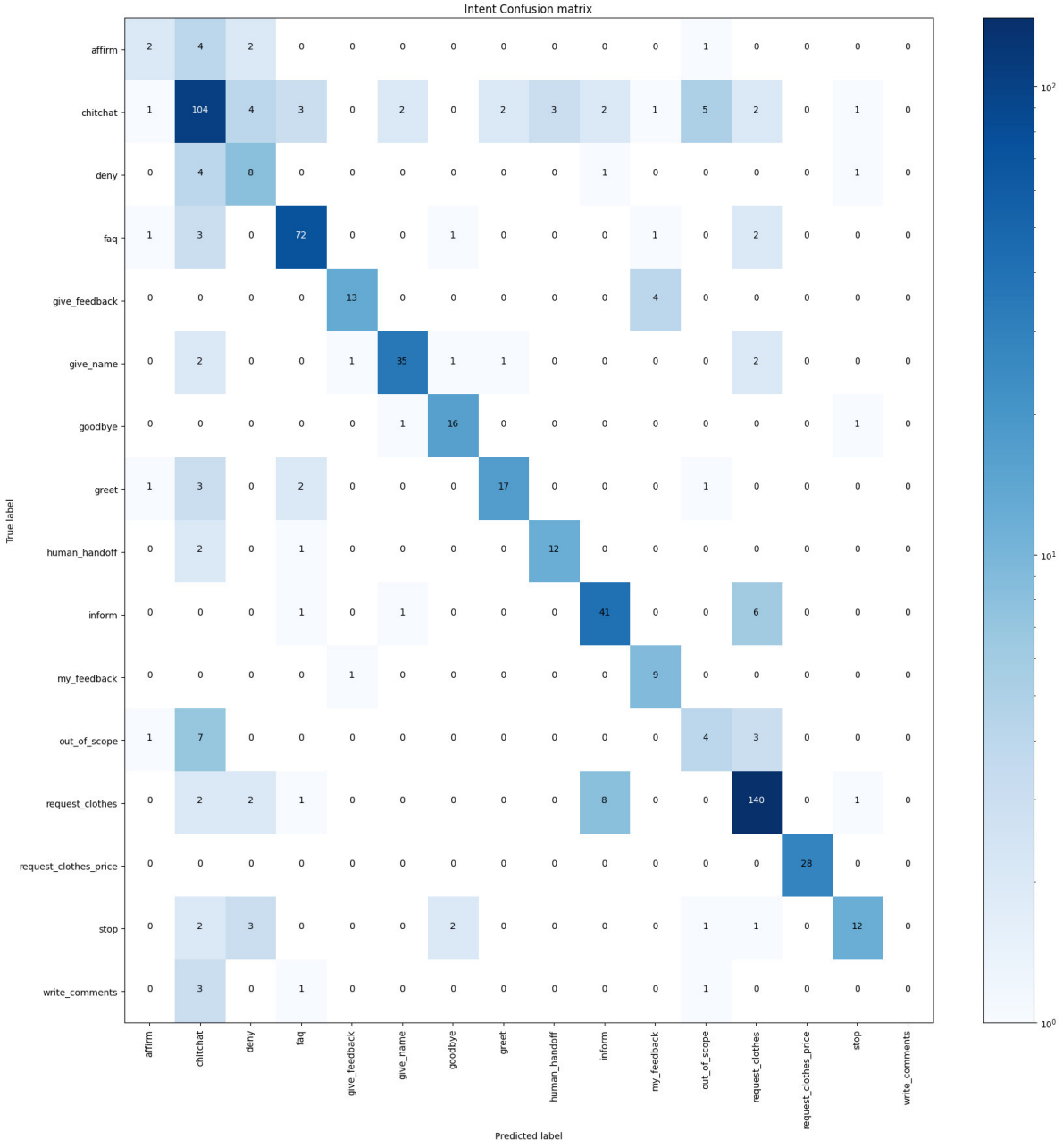


Figura 23 - Matriz de confusión del modelo de clasificación de intenciones y extracción de características.

En base a la Figura 23 se puede indicar que ciertas intenciones (affirm, deny, stop, out_of_scope, write_comments) necesitan más ejemplos para mejorar su detección y no confundirse con otras intenciones. También se puede ver que la intención que contiene mayor cantidad de ejemplos es “request_clothes”, la cual representa la tarea más representativa del asistente para la venta y descubrimiento de ropa, como también “chichat”, la cual maneja entradas de usuario fuera del alcance del asistente. En base a los detalles mencionados anteriormente, se prosiguió a corregirlos y se preparó dos sesiones de prueba para que usuarios reales evalúen las habilidades del asistente. Por otro lado, también es importante mencionar que el asistente fue desplegado en una página web para que interactúe con usuarios reales. La página web fue realizado en Gatsby, un framework que permite crear sitios web estáticos con ReactJS (JavaScript) alimentado por GraphQL. Dentro de esta página web, se mostró el objetivo que tiene el asistente desarrollado y como ayudarle a mejorar su desempeño a través de estas sesiones. El chat widget donde se realiza las conversaciones admite imágenes, botones para respuestas rápidas y carruseles para mostrar contenido. Además, se añadió un ligero retraso entre cada mensaje que envía el asistente para facilitar la legibilidad y una sensación más realista en la interacción. De esta manera, se obtuvo mensajes de usuario para añadirlos al corpus de entrenamiento. La Figura 24 ilustra la vista donde el usuario escribe mensajes al asistente.

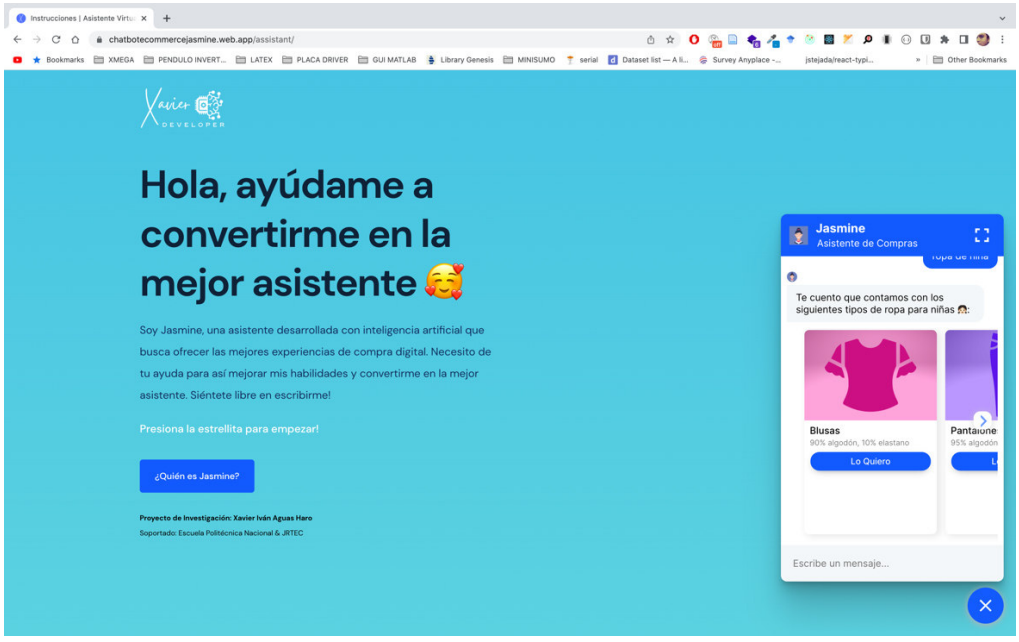


Figura 24 - Vista para interactuar con el asistente conversacional.

Dentro del desarrollo de la página web (Ver Figura 25), se consideró el concepto “Mobile First” para que los usuarios puedan abrir al asistente, ya sea en un navegador de teléfono celular o en una computadora (Diseño Responsive). El Anexo IV y V muestra algunas vistas

adicionales de la página web y ejemplos de conversaciones que ha tenido con un usuario, mostrando así, las habilidades que se plantearon en los objetivos de este trabajo de titulación.

(Enlace: https://chatbotecommercejasmine.web.app/)

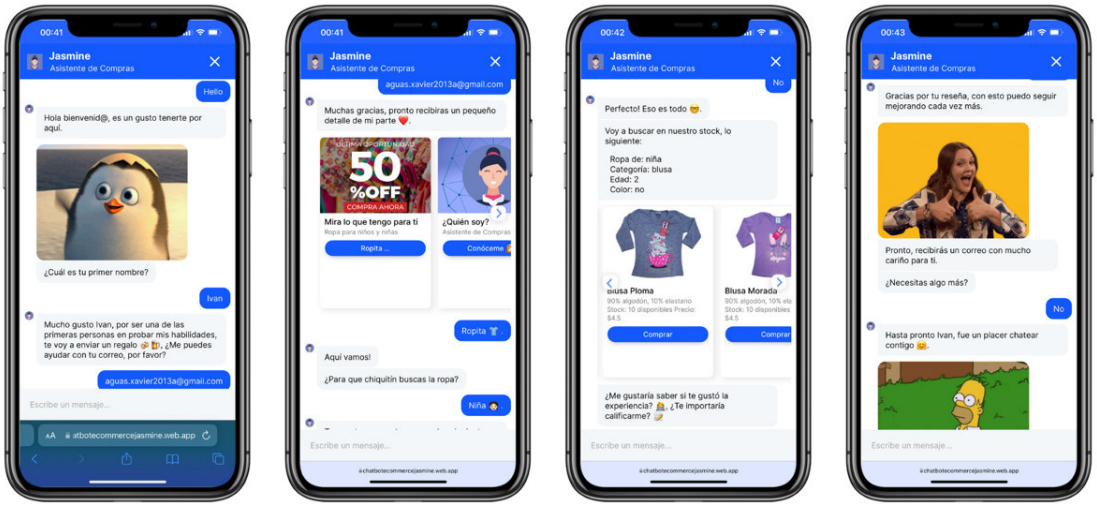


Figura 25 - Ejemplos de conversación entre el usuario y el asistente.

Además, se añadió a la página web, una vista donde se muestra las características y beneficios que trae el asistente desarrollado. El objetivo de esta vista es que más personas se enteren de lo que pueden hacer los asistentes hoy en día y como podrían ayudarles en la automatización de las ventas en línea.

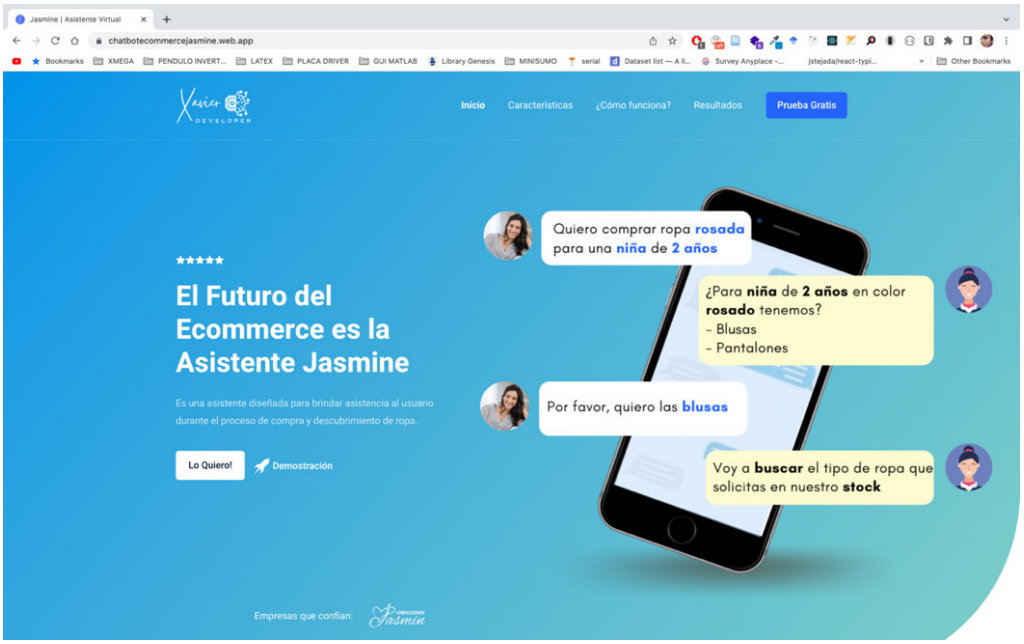


Figura 26 - Vista principal que describe las características del asistente.

Las sesiones fueron realizadas en dos días diferentes obteniendo un registro de 32 chats (20 hombres y 12 mujeres), es decir aproximadamente 210 mensaje recibidos. Las conversaciones tuvieron una duración de 2 a 3 minutos, siendo la sesión 2, el día con más usuarios interactuando con el asistente. La calificación promedio que le dieron al asistente fue de 4 estrellas por parte de las mujeres y una calificación de 3 estrellas por parte de los hombres. La calificación más baja, se debe a que la mayoría de los hombres empezaron a realizar bromas y a hacer preguntas que estaban fuera del dominio del asistente. Por lo tanto, dicho asistente empezó a fallar y a perderse en las conversaciones. El Anexo IV muestra los comentarios de los usuarios de prueba que participaron en las sesiones.

La Figura 27 muestra un resumen de las sesiones realizadas con el objetivo de obtener más datos de entrenamiento y conocer en que etapas de la conversación falla.

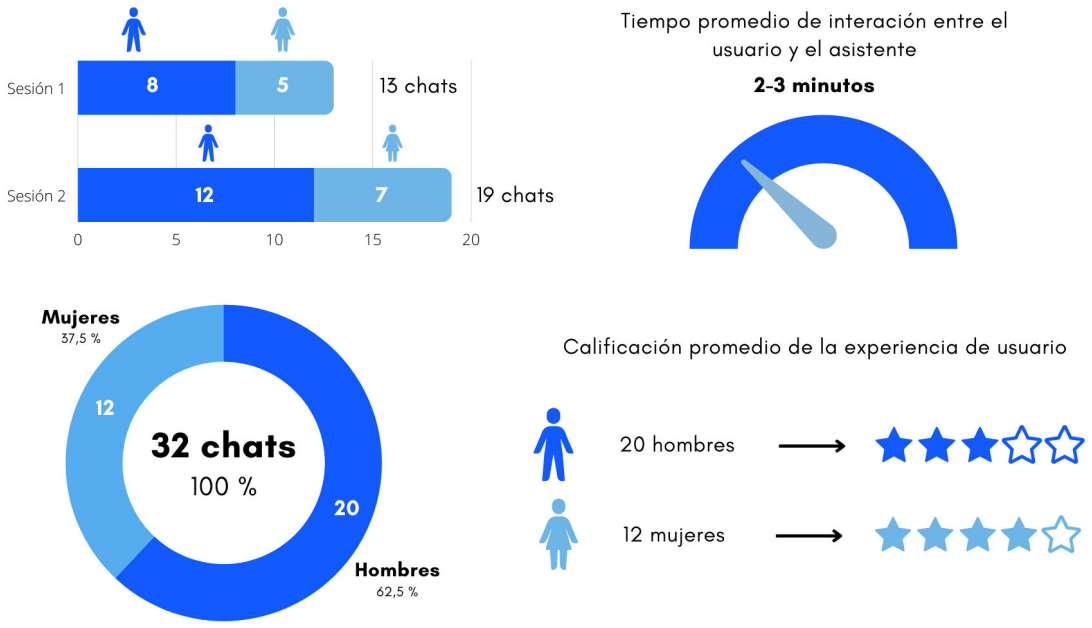


Figura 27- Detalles de las sesiones realizadas con usuarios reales.

Las calificaciones fueron obtenidas siempre que un usuario haya buscado una prenda de vestir y haya terminado el flujo, es decir hasta despedirse; entonces el asistente realiza la pregunta adecuada para obtener dicho dato y almacenarlo. En esta ocasión, el usuario puede dejar un comentario o sugerencia una sola vez. Después de que el usuario haya dejado un comentario el asistente envía un correo electrónico de agradecimiento a través de un servidor SMTP. El correo utilizado pertenece a una cuenta creada en Gmail.

La Figura 28 muestra el correo enviado por parte del asistente después de que el usuario haya terminado el flujo de conversaciones y finalmente haya dejado un comentario.

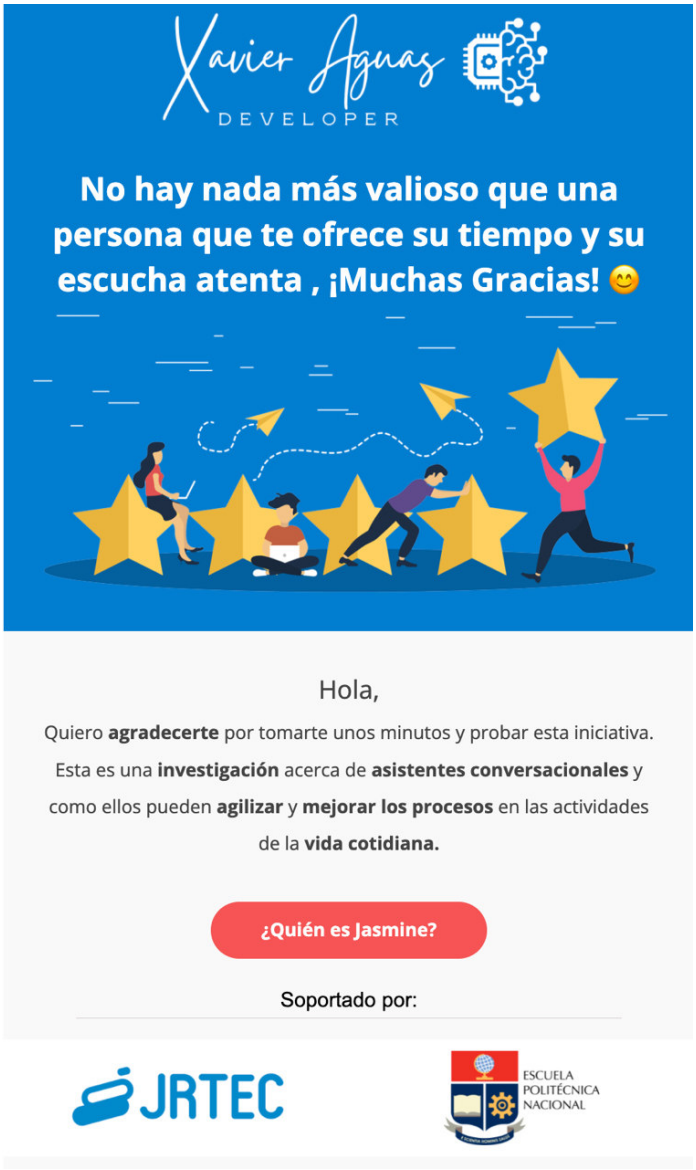


Figura 28 - Cuerpo del correo electrónico de agradecimiento.

Una vez que se ha adquirido varios mensajes de usuario reales, se los añadió al corpus de entrenamiento para mejorar la clasificación y la extracción de entidades, obteniendo los siguientes resultados:

La Figura 29 muestra la matriz de confusión después de haber entrenado nuevamente el modelo considerando los mensajes obtenidos de los usuarios en las sesiones realizadas. Las épocas establecidas para el entrenamiento fueron 200 al igual que en el primer entrenamiento. Dentro del análisis de los mensajes se encontraron mensajes interesantes

y en algunos casos mensajes fuera de lugar, los cuales fueron descartados ya que no pertenecían al dominio del asistente. Se agregaron solamente los mensajes de usuario que el modelo predijo incorrectamente o con poca precisión. Además, se añadieron nuevas intenciones para manejar interjecciones genéricas, como preguntas frecuentes acerca del negocio.

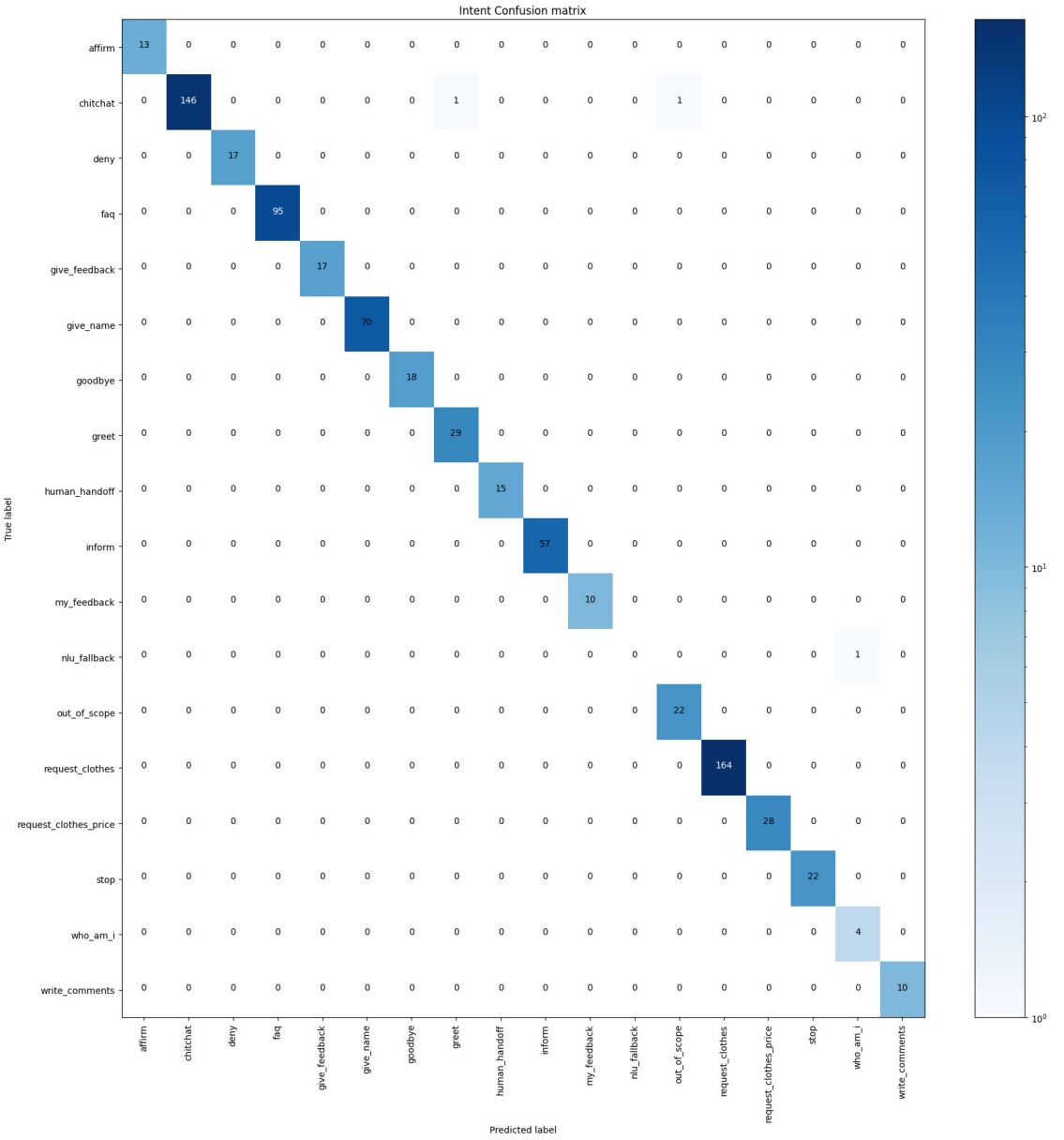


Figura 29 - Matriz de confusión resultante después de las sesiones realizadas.

Por otro lado, a través del histograma podemos visualizar la confianza de todas las predicciones correctas e incorrectas que se realizaron (barras verdes y barras rojas respectivamente). Cada vez que se continúe mejorando el modelo las barras verdes se desplazarán hacia arriba mientras que las rojas se dirigirán hacia la parte de abajo. La

Figura 30 muestra el histograma resultante del modelo entrenado después de las dos sesiones realizadas. Si se desea ver más de los resultados del modelo final, estos son presentados en el Anexo I.

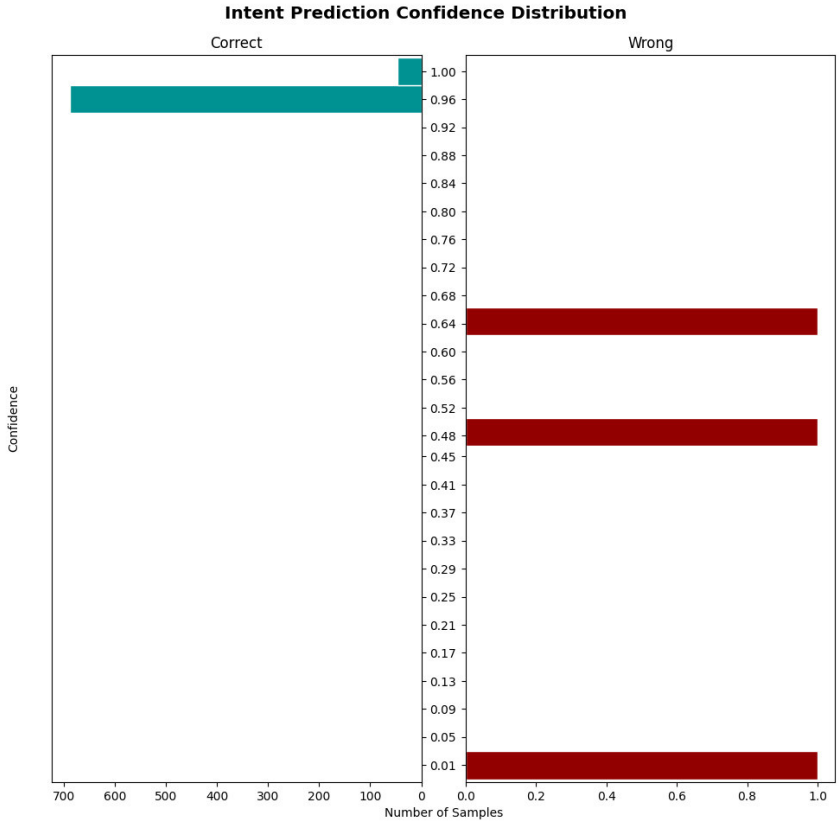


Figura 30 - Histograma final del modelo resultante después de las sesiones realizadas.

Para finalizar, es importante recalcar que la realización de un asistente no termina en una única versión, sino que debe seguir evolucionando y adaptándose a los usuarios constantemente. El desafío de crear excelentes asistentes conversacionales es que es imposible anticipar todas las cosas que podrían decir los usuarios, pero si aumentar la tasa de efectividad en las respuestas.

4. CONCLUSIONES

- Se desarrolló un asistente conversacional para la compra-venta de ropa desarrollado en Rasa, un framework de código abierto, con el objetivo de mostrar cómo este asistente puede aliviar la carga al responder preguntas frecuentes, estar disponible las 24 horas del día y sobre todo que esta investigación sirva como una guía para el desarrollo de asistentes conversacionales inteligentes de bajo costo.

- La técnica de Mago de Oz fue fundamental en el desarrollo del asistente conversacional para obtener ejemplos de conversaciones con usuarios reales. Al simular que había un asistente ya desplegado y funcionando, los usuarios escribían sin limitaciones y se sentían a gusto conversando con el asistente ya que en este caso no recibían respuestas indeseables. Por lo tanto, las sesiones de conversación eran más extensas y había más datos de entrenamiento para la creación del asistente.

- La utilización de servicios web externos permitió al asistente realizar tareas más complejas como por ejemplo utilizar Algolia para encontrar productos en una base de datos más rápido, el uso de un servidor SMTP para envío de correos, usar AirTable para guardar reseñas y saber lo que la gente está buscando.

- La utilización de contenedores (Docker) para el despliegue del asistente en la nube fue de gran ayuda debido a la portabilidad que otorga cuando se cambia de entornos. El desarrollo local se realizó en una computadora, con sistema operativo MacOS y el despliegue en la nube se realizó en Ubuntu.

- Del estudio realizado a Rasa se puede decir que este ecosistema de desarrollo permite crear asistentes mucho más complejos que los enfoques clásicos o los enfoques de aprendizaje por refuerzo. El aprendizaje interactivo que propone Rasa y el uso de redes neuronales tipo “Transformer” permite mejorar la precisión del modelo y la calidad de la respuesta al estar siempre relacionado con el usuario final en cada etapa de mejoramiento. Esto debido a la combinación del procesamiento y comprensión de lenguaje natural que otorga respuestas acertadas ante entradas inesperadas a través del seguimiento de conversaciones y uso del contexto.

- Para la creación de asistentes conversacionales existen diferentes frameworks de desarrollo como DialogFlow o IBM Watson, las cuales permiten crear asistentes con facilidad (low-code), pero el control y personalización no es tan accesible. Por otro lado, Rasa al ser de código abierto brinda control sobre el desarrollo y personalización de los modelos siempre que se tenga un conocimiento elevado sobre procesamiento de lenguaje natural. Para este caso particular, se añadieron distintos componentes de tokenizadores, extractores de características, clasificadores de intenciones y extractores de entidades.

- Una de las tareas que representó un desafío fue la creación de la base de datos de ropa, ya que este proceso no solo conllevo el ingreso de datos en un formato determinado, sino que se realizó actividades complementarias como una sesión fotográfica, edición de fotografía, establecer la cantidad de ropa para la venta y la clasificación de la ropa por tamaños.

- Con el desarrollo de este asistente se puede concluir que un asistente conversacional siempre cometerá errores especialmente en la etapa inicial debido a que no se puede determinar todos los posibles casos que se puedan generar entre el asistente y el usuario, pero incluir un proceso de capacitación y evaluación como el que se ha implementado con usuarios reales (Conversation-Driven Development) permite que el modelo de dicho asistente pueda generalizarse a escenarios del mundo real y que la tasa de efectividad siga aumentando.

- Tener un sistema de control de versiones y despliegue continuo con GitHub Actions permite aplicar la técnica de Conversation-Driven Development de manera satisfactoria. Esto facilitó la inclusión nuevos datos de entrenamiento, desarrollo y mejoramiento de tareas y principalmente realización de pruebas después de haber tenido las sesiones de conversación entre usuarios reales y el asistente. Todo esto sin afectar la disponibilidad del asistente y garantizar que el código no se corrompa en cada actualización.

RECOMENDACIONES

- Un usuario de prueba debe ser cualquier persona que no conozca el funcionamiento del asistente. Personas del equipo de desarrollo no deben participar como usuarios de prueba ya que conocen las habilidades del asistente.

- No sobre indicar a los usuarios de prueba acerca de las tareas que puede realizar el asistente. Solo deben saber sobre el dominio en el que se desarrolla el asistente.

- No utilizar herramientas o plantillas de texto generadoras de datos sintéticos para aumentar ejemplos de entrenamiento. Esto conlleva a que dichos datos no se parezcan a lo que los usuarios realmente escriben, de modo que el rendimiento del asistente será menor.

- Para obtener datos de entrenamiento confiables, lo mejor es involucrar a los usuarios de prueba con el asistente lo antes posible. Crear datos por si mismo no es una buena idea, porque no se puede anticiparse a todo lo que diga el usuario. Por esta razón, las conversaciones reales tienen más relevancia que las conversaciones hipotéticas.

TRABAJO FUTURO

- Para mejorar las habilidades del asistente conversacional se podría implementar un backend que permita al usuario revisar el estado de sus pedidos, el historial de sus compras y sobre todo añadir un sistema de recomendación para ofertarle más productos relacionado a sus preferencias de ropa. En esta instancia, no solo pensar en beneficio del negocio sino en aportarle valor al cliente.

- También, dentro de las mejoras se podría desplegar el asistente en distintos puntos de contacto con posibles clientes. Utilizar el alcance que tienen las redes sociales y maximizar la atracción del negocio con el objetivo de generar más ventas.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Abdellatif, K. Badran, D. Costa, y E. Shihab, «A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering», *IEEE Trans. Softw. Eng.*, pp. 1-1, 2021, doi: 10.1109/TSE.2021.3078384.

[2] E. Paikari y A. Van Der Hoek, «A framework for understanding chatbots and their future», en *2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2018, pp. 13-16.

[3] A. Singh, K. Ramasubramanian, y S. Shivam, «Introduction to microsoft bot, rasa, and google dialogflow», en *Building an Enterprise Chatbot*, Springer, 2019, pp. 281-302.

[4] T. Bocklisch, J. Faulkner, N. Pawlowski, y A. Nichol, «Rasa: Open source language understanding and dialogue management», *ArXiv Prepr. ArXiv171205181*, 2017.

[5] N. M. Deepika, M. M. Bala, y R. Kumar, «Design and implementation of intelligent virtual laboratory using RASA framework», *Mater. Today Proc.*, 2021.

[6] Revista Ekos, «En 2021, el comercio electrónico mantendrá un crecimiento sostenido en Ecuador», *Ekos Negocios*, 2021. https://www.ekosnegocios.com/articulo/en-2021-el-comercio-electronico-mantendra-un-crecimiento-sostenido-en-ecuador (accedido 25 de agosto de 2021).

[7] J. L. M. Loor, A. D. A. Navarro, J. B. V. De Lucca, y D. E. V. Gonzabay, «E-commerce: un factor fundamental para el desarrollo empresarial en el Ecuador», *Rev. Científica ECOCIENCIA*, vol. 5, pp. 1-17, 2018.

[8] W. Astuti, D. P. I. Putri, A. P. Wibawa, Y. Salim, Purnawansyah, y A. Ghosh, «Predicting Frequently Asked Questions (FAQs) on the COVID-19 Chatbot using the DIET Classifier», en *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, abr. 2021, pp. 25-29. doi: 10.1109/EIConCIT50028.2021.9431913.

[9] T. Dinesh, M. R. Anala, T. T. Newton, y G. R. Smitha, «AI Bot for Academic Schedules using Rasa», en *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, 2021, pp. 1-6.

[10] G. Shekhar, R. D'Souza, y K. Fernandes, «AI-Driven Contextual Virtual Teaching Assistant Using RASA», en *Proceedings of the 21st Annual Conference on Information Technology Education*, 2020, pp. 346-346.

[11] E. Strubell, A. Ganesh, y A. McCallum, «Energy and Policy Considerations for Deep Learning in NLP», *ArXiv190602243 Cs*, jun. 2019, Accedido: 3 de abril de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1906.02243

[12] D. Weissenborn, T. Kočiský, y C. Dyer, «Dynamic Integration of Background Knowledge in Neural NLU Systems», *ArXiv170602596 Cs*, ago. 2018, Accedido: 3 de abril de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1706.02596

[13] Y. Windiatmoko, R. Rahmadi, y A. F. Hidayatullah, «Developing Facebook Chatbot Based on Deep Learning Using RASA Framework for University Enquiries», en *IOP Conference Series: Materials Science and Engineering*, 2021, vol. 1077, n.º 1, p. 012060.

[14] M. Zakipour, A. Meghdari, y M. Alemi, «RASA: A low-cost upper-torso social robot acting as a sign language teaching assistant», en *International conference on social robotics*, 2016, pp. 630-639.

[15] S. Studer *et al.*, «Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology», *ArXiv200305155 Cs Stat*, feb. 2021, Accedido: 6 de septiembre de 2021. [En línea]. Disponible en: http://arxiv.org/abs/2003.05155

[16] Pineda, Tania, «El impacto del COVID-19 sobre el comercio electrónico en el Ecuador.», Universidad Técnica de Machala, Ecuador, 2021.

[17] P. B. Brandtzaeg y A. Følstad, «Why people use chatbots», en *International conference on internet science*, 2017, pp. 377-392.

[18] N. Malamas y A. Symeonidis, «Embedding Rasa in edge Devices: Capabilities and Limitations», *Procedia Comput. Sci.*, vol. 192, pp. 109-118, 2021.

[19] K. N. Lam, N. N. Le, y J. Kalita, «Building a Chatbot on a Closed Domain using RASA», en *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, 2020, pp. 144-148.

[20] L. W. Narendra y E. R. Setyaningsih, «Designing a Transactional Smart Assistant in Indonesian using Rasa Framework», en *2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, 2021, pp. 1-6.

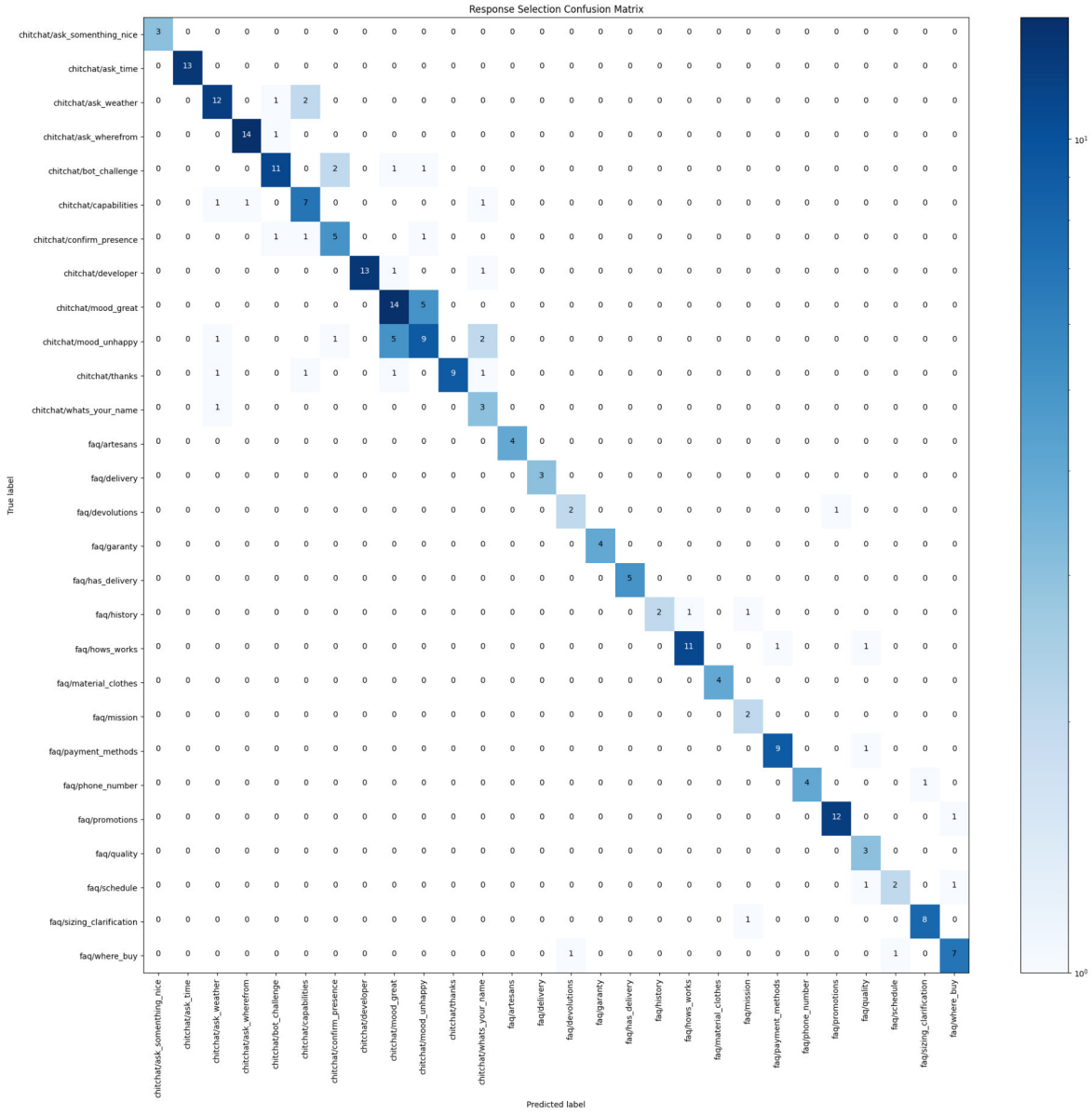
[21] D. G. Ruindungan y A. Jacobus, «Chatbot Development for an Interactive Academic Information Services using the Rasa Open Source Framework», *J. Tek. Elektro Dan Komput.*, vol. 10, n.º 1, pp. 61-68, 2021.

[22] S. Yu, Y. Chen, y H. Zaidi, «A Financial Service Chatbot based on Deep Bidirectional Transformers», *ArXiv200304987 Cs Stat*, feb. 2020, Accedido: 3 de abril de 2022. [En línea]. Disponible en: http://arxiv.org/abs/2003.04987

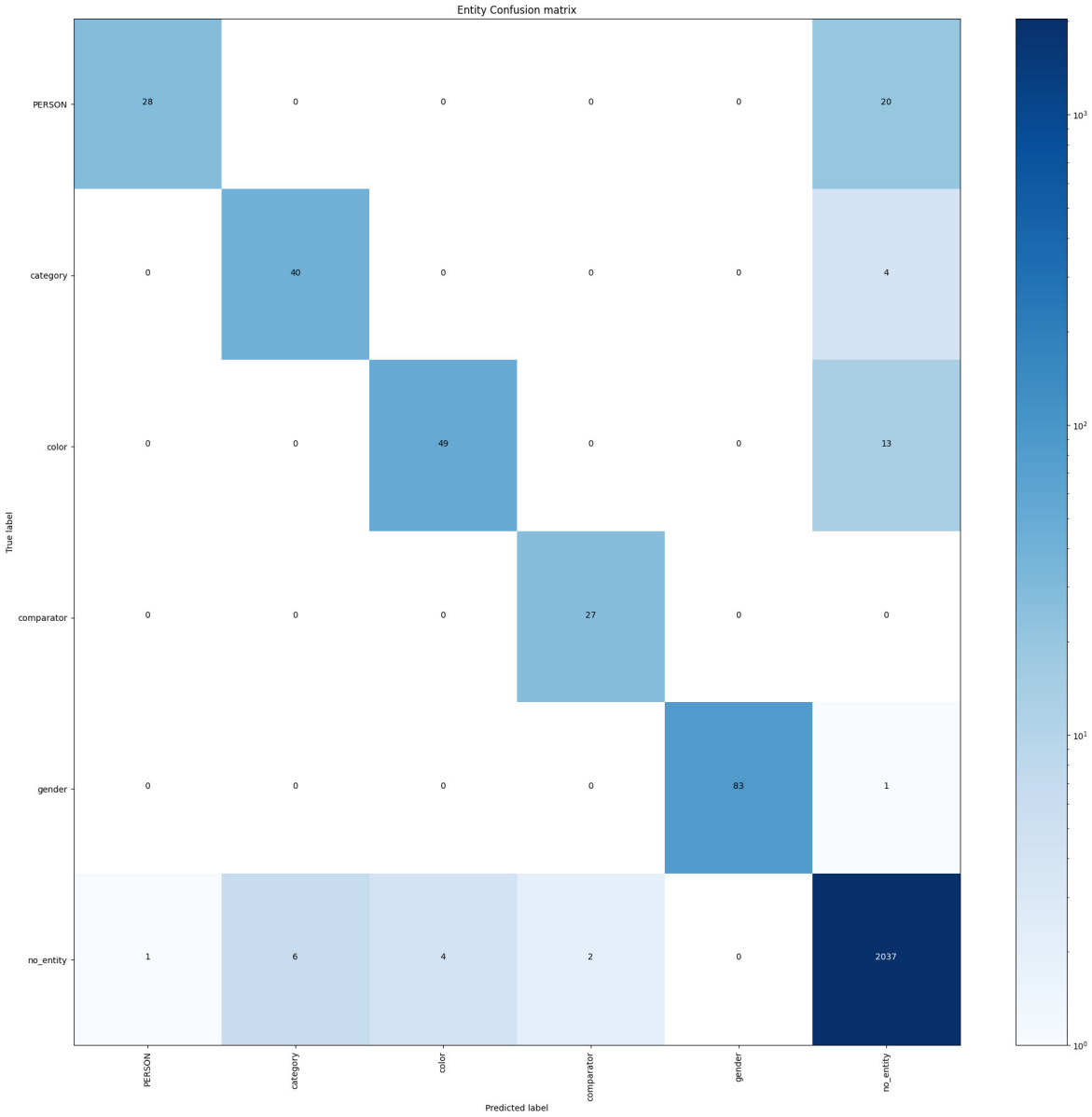
[23] M. Joshi y R. K. Sharma, «An Analytical Study and Review of open Source Chatbot framework, RASA», *Int. J. Eng. Res. Technol.*, vol. 9, n.º 6, jun. 2020, doi: 10.17577/IJERTV9IS060723.

6. ANEXOS

Anexo 1 - Matriz de confusión del modelo de selección de respuestas



MATRIZ DE CONFUSIÓN DEL MODELO DE SELECCIÓN DE RESPUESTAS



Anexo 2 - Fotográfias de la ropa de niña.

Pijamas

Leggings

Blusas

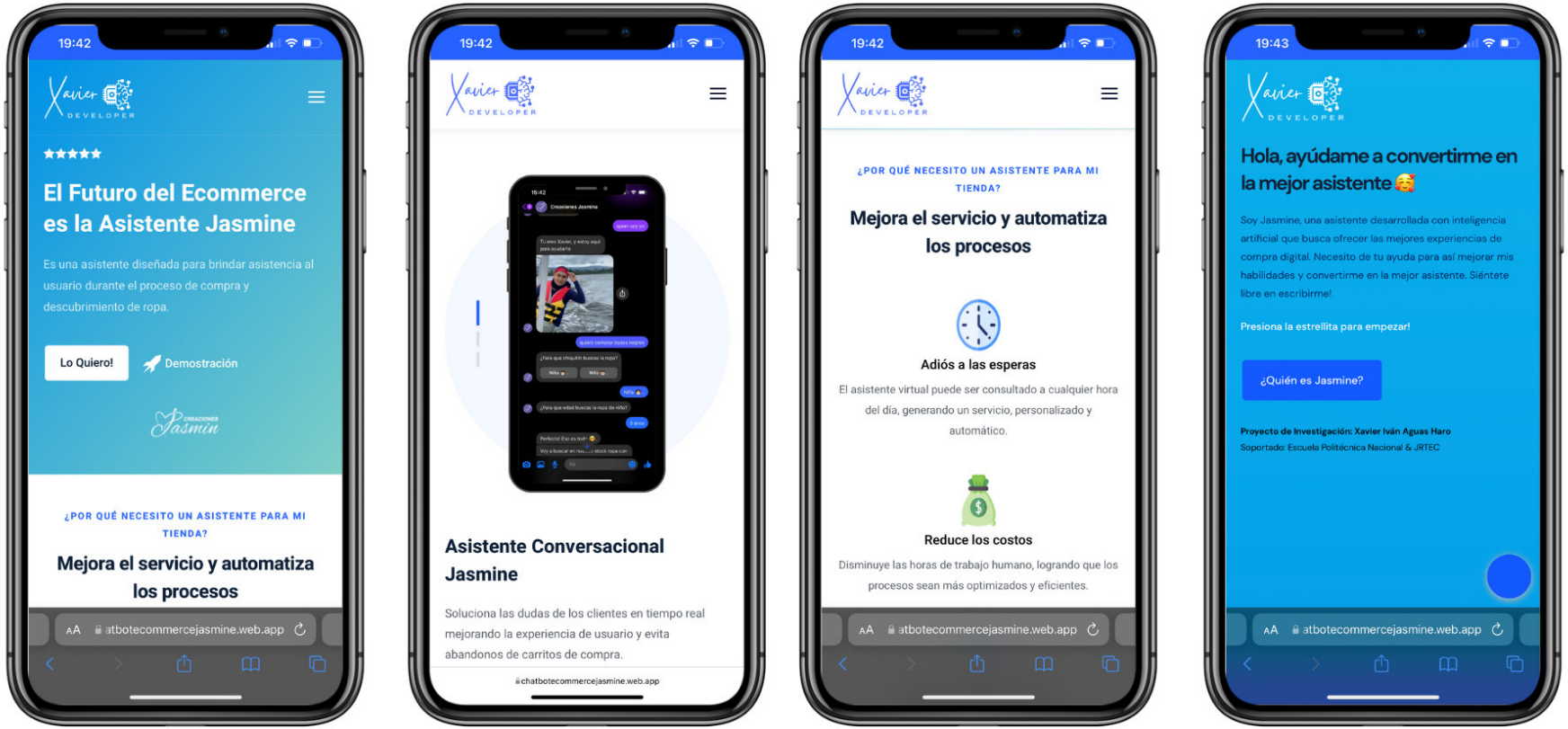
Anexo 3 - Fotográfias de la ropa de niño.

Camisetas

Buzos

Pantalones

Anexo 4 - Vistas de la página informativa sobre el asistente conversacional desarrollado.



Anexo 5 - Algunas vistas del asistente conversacional desarrollado.

Anexo 6 - Tabla de sugerencias/comentarios de las sesiones realizadas.

#	Usuario	Feedback	Mensaje	Sesión
1	Juan	4	Me gusta la interacción y la velocidad de como se carga la pagina	Sesión 1
2	Monica	4	Me gusta la interfaz del asistente	
3	Violeta	3	Necesita mas productos	
4	Lizzy	5	Los gifs son chistosos	
5	Dario	3	Necesitan porner ropa para adultos	
6	Edgar	4	Esta buena la iniciativa pero muy basica	
7	Marcelo	4	El asistente es muy bueno buscando productos	
8	Jeff	3	Se esta perdiendo	
9	Allyson	4	me gusta el asistente porque se adapata a mi telefono	
10	Stalyn	5	Gran iniciativa	
11	El pepe	5	Le hace falta más productos para adultos	
12	Geovanny	4	La pagina web esta cool	
13	Carlos	4	funciona	
14	Kabir	3	El asistente no pudo guardar mi nombre a la primera vez	Sesión 2
15	Jessica	4	Solo venden ropa para niños	
16	Fabian	2	Se pierde	
17	Nelson	4	Me gusta la ropa que presenta el asistente	
18	Jordan	2	El bot no me da la hora	
19	Hugo	3	No responde a ciertas preguntas	
20	Marilyn	5	Me gusta como se muestra la ropa	
21	Arlenys	4	Necesitan mas colores	
22	Jonathan	4	Me gustaria que pueda verlo en facebook	
23	Jenniffer	3	Sigan mejorando	
24	George	1	Se pierde el bot, no funciona	
25	Mary	4	Funciona bien	
26	Luis	4	Me gusta las fotos de la ropa	
27	Marco	3	Siento que le falta algo	
28	QK	4	Funciona para comprar ropa pero no puede responder bien las preguntas	
29	Ramiro	4	good	
30	Jasmina	5	Excelente trabajo	
31	Manu	5	Los gráficos que manda el bot se ven bien	
32	Estefy	4	Quiero que me recomiende cosas	