Universidad Politécnica

de Madrid

Escuela Técnica Superior de

Ingenieros Informáticos

Grado en Matemáticas e Informática

Trabajo Fin de Grado

Desarrollo de un Chatbot para Identificar Deterioros Cognitivos

Autor: María Merino Pereda

Tutora: Raquel Cedazo León

Madrid, junio 2021

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Matemáticas e Informática
Título: Desarrollo de un Chatbot para Identificar Deterioros Cognitivos
Junio 2021

Autor: María Merino Pereda

Tutor:
Raquel Cedazo León
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

La enfermedad de Alzheimer es un trastorno cognitivo que afecta a una parte significativa de la población mundial. Cada vez son más necesarias las herramientas que permitan diagnosticar esta enfermedad lo antes posible.

Este trabajo propone el desarrollo de un chatbot que identifica los primeros síntomas de la enfermedad del Alzheimer. Se propone un chatbot ya que presenta las características de usabilidad idóneas para que pueda ser accesible para cualquier persona.

El diagnóstico se realiza a través de un test minimental, una de las técnicas más utilizadas en la medicina tradicional. Además, ha sido implementado en una plataforma de desarrollo de chatbots, Rasa Open Source y Rasa X, y desplegado con Google Cloud Platform.

Por otro lado, mediante la automatización del diagnóstico de trastornos cognitivos se podría aliviar en gran medida las consultas de departamentos de neurología y optimizar los recursos del hospital. En definitiva, podría implicar un gran impacto positivo en la sociedad, especialmente en el ámbito de la salud.

Abstract

Alzheimer's Disease is a cognitive disorder that affects a significant part of the world's population. The need of a tool to diagnose cognitive disorders such as Alzheimer's Disease in an autonomous way, is growing exponentially.

The aim of this project is to implement a chatbot that identifies symptoms of Mild Cognitive Disorder and Alzheimer's Disease. The interface provided by chatbots shows great levels of usability, making the product accessible for a very high percentage of the population.

The symptoms are identified throughout a minimental test, which is one of the most popular techniques used by traditional medicine, but it is automatised so that it does not require the presence of a physician. Furthermore, it has been implemented in Rasa Open Source, a very popular framework for chatbot development, and deployed with Google Cloud Platform.

The development of this project can provide a very high benefit for society, for it could deliver a very high-quality diagnosis method at a very reasonable price. It could optimise the resources of hospitals and maximise the number of patients treated.

Tabla de contenidos

1 Introducción

1.1 Motivación

En la actualidad, cerca de 24,3 millones de personas en el mundo padecen demencia, siendo la más común la enfermedad de Alzheimer. Cada año, se diagnostican 4.6 millones de nuevos casos, un caso cada 7 segundos [1], y se estima que el número de casos se duplicará cada 20 años.

En los últimos años, se han desarrollado tratamientos prometedores. Sin embargo, es necesario realizar un diagnóstico precoz de la enfermedad para poder mostrar su eficacia en las fases lo más iniciales posible. Por ello, es cada vez más necesario disponer de herramientas que faciliten el diagnóstico de esta enfermedad.

Por otro lado, los servicios de neurología se encuentran desbordados por el alto número de pacientes que acuden a las consultas con deterioros cognitivos. Por lo tanto, sería muy interesante implementar una herramienta que permita identificar síntomas de enfermedades cognitivas y que se pueda utilizar sin necesidad de acudir a un hospital físicamente.

Teniendo esto en cuenta, podemos concluir que es cada vez más necesario disponer de nuevas herramientas que faciliten el diagnóstico de las enfermedades cognitivas, y en concreto, del Alzheimer. Además, cuanto más automatizadas e independientes sean estas herramientas, mayor será el impacto positivo en la sociedad.

Los últimos avances en tecnología han dado lugar a soluciones digitales más eficientes, de menor coste y mayor calidad, sobre todo en cuanto a la Inteligencia Artificial. Es por ello que una de las soluciones más interesantes en esta área es el desarrollo de chatbots para identificar deterioros cognitivos, como la enfermedad del Alzheimer.

Un chatbot es una entidad artificial capaz de mantener, de forma autónoma, una conversación a través de mensajes. Su funcionamiento se basa en analizar las entradas recibidas con el objetivo de reconocer secuencias para poder devolver respuestas predefinidas [2].

En esta comparación se observa la estructura del texto que se recibe y se busca una respuesta que se ajuste a las variables exclusivas del texto. Por lo tanto,

los chatbots no pueden realizar funciones como contestar a preguntas complejas o realizar actividades compuestas.

En los últimos años, se ha incrementado notablemente el uso de chatbots en el área médica ya que han demostrado tener las características ideales. En primer lugar, poseen la mezcla perfecta entre inmediatez y sincronicidad [3]. La inmediatez la proporciona la rapidez con la que el programa responde a la persona que interactúa con él, mientras que la sincronicidad se obtiene a través de las notificaciones y los recordatorios, que mantienen comprometidos a los usuarios con la aplicación. Además, los chatbots poseen la cualidad de estar siempre disponibles cuando el usuario los necesita, y requieren menos costes y recursos temporales que acudir a un profesional sanitario.

Por otro lado, los chatbots presentan un alto grado de usabilidad para personas mayores [6] o personas con demencia [4] que impulsa la confianza de los usuarios para reforzar una buena comunicación. De esta forma pueden ayudar a un gran número de personas, monitorizando y registrando los datos que proporcionan los usuarios, que más adelante, pueden ser contrastados y tratados mediante algoritmos de inteligencia artificial para realizar predicciones realistas.

Todas estas características proporcionan el marco perfecto de privacidad, personalización, y escalabilidad [5] para aumentar el confort de los usuarios a la hora de interactuar con el chatbot. También poseen instrumentos claves para poder manejar grandes cantidades de datos y proporcionar diagnósticos cada vez más precisos.

Este trabajo propone el desarrollo y la implementación de un chatbot que permita identificar deterioros cognitivos, como la enfermedad del Alzheimer. Este objetivo se conseguirá mediante un test mini mental, que consistirá en un conjunto de preguntas destinadas a evaluar seis aspectos cognitivos diferentes.

Además, se hará a través de una interfaz gráfica accesible para el mayor número de personas, con un grado de usabilidad alto, para que no suponga una barrera tecnológica para personas de edad avanzada o que presenten dificultades a la hora de interactuar con las nuevas tecnologías.

Por último, se mostrará el resultado de la implementación y se analizarán las conclusiones principales a las que se han llegado tras haber desarrollado el proyecto.

1.2 Objetivos

A continuación, se muestra una lista de los objetivos concretos que se pretenden alcanzar con el desarrollo de este trabajo.

- Creación de un chatbot que identifique trastornos cognitivos a través de una dinámica conversacional
- Creación de una interfaz conversacional de un chatbot
- Desarrollo de una interfaz gráfica con un alto grado de usabilidad
- Análisis de los resultados obtenidos
- Desarrollo de una guía de usuario

1.3 Estructura

Este documento proporciona una descripción detallada del desarrollo del proyecto. En este apartado, se comentará la estructura principal del documento completo.

Comienza con un capítulo destino al **estado del arte**, en el que se explica el contexto en el que se realiza el proyecto. Parte de una descripción detallada de algunos de los trabajos previos más importantes sobre el uso de las nuevas tecnologías para el diagnóstico de enfermedades, y la evolución de las diferentes formas de diagnosticar la enfermedad del Alzheimer.

Más adelante, se dedican algunos capítulos a especificar la historia de los chatbots hasta este momento, los sectores principales en los que se están implementando, y una clasificación de diferentes tipos de asistentes virtuales. También se especifican las características de la computación cognitiva y el funcionamiento del Procesamiento de Lenguaje Natural, dos aspectos muy importantes para comprender el funcionamiento de los chatbots.

Por último, se describen algunos de las plataformas más importantes para desarrollar chatbots y para desplegar aplicaciones.

En el capítulo del **desarrollo**, en primer lugar, se explica detalladamente el diseño del chatbot, las diferentes partes que componen la historia diseñada y cómo están conectadas entre ellas.

Más adelante se describe la implementación desde un punto de vista más técnico. Al principio, se describen todos los elementos necesarios para construir un chatbot en el marco Rasa Open Source. A continuación, se explica cómo se ha implementado el diseño de la historia a partir de la implementación de diferentes elementos explicados anteriormente. Por último, se describe el despliegue del chatbot en una máquina virtual de Google Cloud Platform.

En las **conclusiones** se realiza una valoración personal del desarrollo del proyecto. Se analiza el producto final obtenido y las impresiones generales. Además, se proporciona una serie de posibles mejoras a implementar en el futuro.

Por último, el apartado del **impacto**, está destinado a proporcionar una visión de cómo un proyecto como este podría afectar en la sociedad. Se analiza el impacto en el ámbito de la salud y de la economía.

2 Estado del Arte

Los chatbots son entidades artificiales capaces de mantener, de forma autónoma, una conversación a través de mensajes. Su principal objetivo es simular una conversación con un humano y su funcionamiento se basa en analizar las entradas recibidas con el objetivo de reconocer secuencias para poder devolver respuestas predefinidas. Para ello, se utilizan algoritmos, reglas y NLP o Procesamiento del Lenguaje Natural.

En este apartado, se analizan algunos de los aspectos más importantes de los chatbots para comprender mejor su funcionamiento, de cara a comprender mejor el desarrollo de este proyecto y su implementación.

2.1 Trabajos previos

El diagnóstico de enfermedades y deterioros cognitivos, tales como la enfermedad del Alzheimer, se realiza, generalmente, por parte de personal sanitario cualificado. Debido al alto número de pacientes que padecen deterioros cognitivos, se disponen de herramientas neuropsicológicas que faciliten el diagnóstico precoz, pero al mismo tiempo, que se puedan aplicar de forma fácil y rápida.

Actualmente, el test mini mental de Folstein [7] es la técnica más utilizada para identificar síntomas de deterioros cognitivos de los pacientes. Siguiendo la estructura del test, se realizan una serie de preguntas y proporciona una puntación en función de las respuestas de los pacientes. Una puntuación baja puede estar asociada con numerosos deterioros incluyendo diferentes tipos de demencia u otros tipos de trastornos psicológicos.

En esta línea, se han desarrollado numerosos test similares al test de Folstein para detectar de forma precoz la enfermedad de Alzheimer de manera más precisa. El procedimiento es muy similar pero las preguntas están más enfocadas a identificar específicamente a los pacientes con Alzheimer. En el estudio [9] "Descripción de un nuevo test para la detección precoz de la enfermedad Alzheimer" desarrollaron una prueba combinando tareas de lenguaje con tareas de memoria; que fuese capaz de diferenciar tanto a los pacientes con deterioro cognitivo leve (DCL) de los pacientes con principio de enfermedad de Alzheimer, así como a los que no sufrían ninguna de estas enfermedades.

En los últimos años, gracias al desarrollo tecnológico, se han introducido herramientas tecnológicas en el área médica, cuyo objetivo principal es desarrollar aplicaciones capaces de ayudar a los pacientes con pocos recursos para acudir a un profesional sanitario. De esta forma, se introducen técnicas muy económicas y con un alto grado de accesibilidad.

Uno de los estudios más relevantes en este ámbito ha sido el desarrollo de los CANTAB tests [10]. Estos han demostrado sensibilidad a la hora de detectar cambios en las habilidades neuropsicológicas de los pacientes. Incluyen pruebas computarizadas relacionadas con diferentes áreas cognitivas como la memoria, el aprendizaje, toma de decisiones y tiempos de reacción. En la

actualidad, se han convertido en una de las formas más reconocidas de medir capacidades cognitivas de forma computarizada.

A partir de este estudio, han surgido otros basados en intentar mejorar los resultados proporcionados con los test computarizados utilizando Inteligencia Artificial. Un estudio muy relevante en este ámbito [11], denominado *Improving Mild Cognitive Impairment Prediction via Reinforcement Learning and Dialog Situation*, grabó el proceso de los pacientes completando las pruebas computarizadas para aplicar algoritmos de aprendizaje supervisados para intentar mejorar la identificación del Deterioro Cognitivo Leve.

Para ello, utilizaron un simulador conversacional que realizaba el test a los pacientes y aplicaban algoritmos supervisados para analizar los datos y poder introducir nuevas preguntas en las pruebas que identificasen con mayor precisión el Deterioro Cognitivo Leve, así como analizar las respuestas proporcionadas por los pacientes a partir de las cuales se establecía el diagnóstico.

En este ámbito, se han realizado otros estudios para identificar el Alzheimer con modelos de Machine Learning [9]. En el estudio, se propone un método para diagnosticar el Deterioro Cognitivo Leve, y a su vez, intentar calcular la probabilidad de que esto desemboque en la Enfermedad de Alzheimer. Los resultados de este estudio han demostrado el gran potencial que tiene introducir diagnósticos con ayuda computacional en algunas áreas de la biomedicina.

Como resultado de todas estas investigaciones, se puede concluir que el uso de tecnologías puede aportar muchos beneficios en el área de la medicina. Una de las más comunes es el uso de chatbots ya que observando los resultados de estos trabajos, se ha demostrado que presentan las características ideales para la interacción con pacientes. Además, la introducción de algoritmos de procesamiento de datos es muy interesante a la hora de mejorar la predicción de los datos; y se puede apreciar que ambas herramientas se complementan bien y proporcionan un buen resultado.

2.2 Historia de los chatbots

Hay dos principales teorías sobre cómo surgieron los chatbots. La primera de ellas defiende que fue el matemático Alan Turing, quién desarrolló un juego que denominó "The Imitation Game" [12]. En él, participan dos personas y una máquina. Una de las personas actúa como juez y se comunica tanto con la otra persona, como con la máquina por escrito. El objetivo del juego es que el juez determine quién es la maquina y quién la persona.

Turing determinó que, si una máquina era capaz de engañar al juez, entonces se podría considerar inteligente. Esto se denominó Test de Turing [13], y se convirtió en el control de calidad para determinar si un bot tiene la capacidad de generar un comportamiento inteligente. A día de hoy, sigue siendo un estándar en la industria.

Por otro lado, la segunda teoría defiende que el primer chatbot basado en inteligencia artificial fue ELIZA [14]. Lo desarrolló Joseph Weizenbaum en el MIT (Instituto de Tecnología de Massachusetts) en 1966 y su principal objetivo era conectar al hombre con las máquinas a través de una interfaz de comunicación. Se convirtió en la primera máquina con la capacidad de mantener una conversación utilizando el procesamiento del lenguaje natural.

Este primer chatbot, sirvió como modelo para otros como *Parry,* desarrollado por la Universidad de Stanford en 1972, y caracterizado por una actitud emocional. Este destacó ya que, en un estudio realizado, solo el 48% de los psiquiatras profesionales fueron capaces de distinguir entre las respuestas de un humano y las de Parry.

Otros de los chatbots más clásicos son SmarterChild y SHRDLU. Algunos de los más recientes son Racter, Jabberwacky o ALICE [15], que utiliza el lenguaje AIML, una extensión de XML todavía en uso. Hoy en día, miles de personas interactúan diariamente con chatbots modernos como Siri, Cortana o Alexa. Estos actúan como asistentes generales con funciones varias.

En cuanto a la competición de calidad de los chatbots, en 1990 se estableció el Premio Loebner [16], un concurso anual para establecer la inteligencia artificial que más se acerca a la respuesta humana. Desde 2019, se califica en base a la valoración del público. Sin embargo, ningún chatbot ha obtenido nunca una puntuación superior a 70 sobre 100, según las reglas de Turing. Por tanto, ninguno ha sido suficientemente humano todavía.

2.3 Sectores

Los chatbots han ido evolucionando y sufriendo modificaciones para poder ser aplicados en numerosos ámbitos, independientes al ámbito académico. Las nuevas técnicas agregadas a los chatbots han permitido la incorporación de estos en numerosos sectores.

2.3.1 Chatbots en la Salud

En el ámbito médico, podemos encontrar diversos programas que tienen el objetivo de agilizar las reservas de citas médicas, realizar consultas sobre medicamentos o indicaciones acerca del uso de estos. Un ejemplo es Pharmabot [17], que proporciona respuestas a los usuarios que envían sus preguntas sobre medicamentos.

También se utilizan para procesar datos clínicos, gestionar citas para revisiones o entretener a pacientes y familiares si tienen que pasar una temporada larga en un hospital. Otro ejemplo de un chatbot en este contexto, es el desarrollado por la Confederación de Afectados por Enfermedad de Crohn y Ulcerosa (ACCU), para asistir a paciente con enfermedad inflamatoria intestinal.

Por otro lado, en los últimos años se han comenzado a implementar chatbots para diagnosticar. La Asociación Médica Americana estima que al menos el 75% de las visitas presenciales al médico podrían sustituirse por una atención médica a distancia [18]. Además, a raíz de la pandemia causada por el Covid-19, las consultas a través de aplicaciones y chats se han intensificado notablemente.

A pesar de no ser un sustituto de la consulta médica, los chatbots podrían actuar de filtro, averiguando el grado de urgencia de un paciente o incluso derivarlo al especialista más indicado. De hecho, el Servicio Nacional de Salud de Reino Unido (NHS) está probando ya bots médicos a través de aplicaciones [19].

2.3.2 Chatbots en el Ámbito Educativo

Los bots conversacionales tienen un gran potencial en el ámbito educativo debido a su capacidad comunicativa. Se basan en la interacción máquina-estudiante y actúan como mediadores para responder preguntas frecuentes o guiar tareas repetitivas, proporcionando atención de forma ininterrumpida.

Otras de las tareas que engloban los chatbots en este sector es la enseñanza de idiomas. Un ejemplo sería el chatbot "Soy Diego" [20], que puede mantener una conversación en el idioma que el usuario quiere aprender, en este caso el español. También es capaz de detectar errores ortográficos o gramaticales. Otro ejemplo muy conocido es Duolingo [21], diseñado para el aprendizaje de numerosos idiomas.

El objetivo es desarrollar programas que garanticen organización, funcionalidad y viabilidad dentro de las organizaciones educativas. Emplean funciones educativas como personalizar o individualizar hacia los alumnos los modelos de enseñanza. Por ejemplo, 'The Guardian of History' [22] es un programa orientado a enseñar historia a niños de entre diez y doce años.

Por otro lado, también desarrollan funciones administrativas respondiendo cuestiones o solicitudes personales. Un ejemplo de este tipo de chatbots es Ivy, diseñado para gestionar admisiones, servicios financieros y servicios tecnológicos.

2.3.3 Chatbots en la Industria Financiera

El uso de chatbots en este sector se focaliza en que las entidades financieras puedan tener un servicio agilizado con sus clientes para ofrecer una atención personalizada [23]. Muchos bancos ya utilizan estos servicios como el BBVA, Caixabank o Sabadell, ya que les permite ofrecer una alta disponibilidad, independientemente de la hora o de posibles temporadas de vacaciones.

Por otro lado, también se han implementado chatbots de IA en servicios bancarios y financieros con el objetivo de prevención de fraude. Estos permiten monitorear las transacciones de los usuarios y las tendencias de gastos para hacer estimaciones y proporcionar asesoramiento. Sin embargo, también pueden identificar riesgos o amenazas de seguridad, generando alertas para que los bancos tomen medidas si lo consideran necesarios.

2.3.4 Chatbots en el Comercio Electrónico

Los chatbots en el comercio electrónico permiten principalmente responder a las preguntas más frecuentes de los clientes cuando están realizando compras por internet. Además, pueden ayudar a los propietarios de los comercios a saber cuáles son las dudas más frecuentes y así poder solucionar los problemas más comunes de sus clientes.

Se estima que las compras por impulso pueden suponer un 16% de los ingresos de un comercio [24], por lo que la inmediatez a la hora de solucionar dudas es cada vez más importante para los negocios. Los chatbots proporcionan una solución económica y muy efectiva ante esta problemática. Es por ello que el uso de este tipo de chatbots está creciendo, principalmente en China.

Un ejemplo de estos chatbots podría ser el 'Fashion Asistant' de Mango. Se trata sobre un asistente virtual de moda que escoge prendas para los clientes de forma automática al proporcionarle con datos sobre gustos o prendas buscadas.

2.4 Clasificación de Asistentes Virtuales

Podemos distinguir 5 niveles principales que clasifican los asistentes conversacionales inteligentes en función de las capacidades principales que los conforman [25]. Estos niveles también muestran la evolución de los chatbots en el tiempo y las previsiones de la industria en el futuro.

2.4.1 Asistentes Notificantes

Son capaces principalmente de mandar notificaciones simples. Entre ellos podemos distinguir mensajes de texto, notificaciones emergentes o mensajes de WhatsApp.

2.4.2 Asistentes FAQ

Son el asistente conversacional más común en la actualidad. Están caracterizados por tener la capacidad de contestar a las preguntas más frecuentes (Frequently Asked Questions o FAQs), realizadas por los usuarios. Normalmente están construidos en torno a conjuntos de reglas o máquinas de estados.

2.4.3 Asistentes de Contexto

Son capaces de comprender el contexto de las conversaciones, recordando lo que el usuario ha dicho previamente y cómo, cuándo y cómo lo ha hecho. Esto proporciona la capacidad de responder a mensajes de usuario inesperados o diferentes a lo inicialmente planeado. Además, son capaces de aprender y mejorar con los datos de conversaciones anteriores, por lo que mejoran su precisión con el tiempo.

2.4.4 Asistentes Personalizados

En este momento, los modelos desarrollados son casi puramente teóricos. Conforman la siguiente generación de asistentes conversacionales que serán capaces de conocer a los usuarios cada vez mejor, a medida que pasa el tiempo y que los usuarios utilizan el chatbot.

2.4.5 Asistentes Autónomos de Organización

Forman parte de una visión a largo plazo de la industria de los chatbots. Serán asistentes que conozcan a cada usuario personalmente. Tendrán la capacidad de controlar una parte importante de las operaciones de una empresa en diferentes áreas de la empresa como ventas, recursos humanos o finanzas.

2.5 Computación Cognitiva

Este término hace referencia a los nuevos softwares que intentan imitar el funcionamiento del cerebro humano e intentan mejorar los procesos de decisión en el mundo real [26].

Uno de los factores que más caracterizan a los elementos de computación cognitiva es su capacidad de adaptación. Este tipo de programas son capaces de aprender a medida que la información, los requisitos y los objetivos que perciben cambian. Además, están caracterizados por ser interactivos con los usuarios que los consumen para poder satisfacer sus necesidades cómodamente.

Por otro lado, son capaces analizar el contexto que les rodea de forma permanente. De esta forma, extraen la información de los elementos contextuales como los diferentes significados, localización, tareas y objetivos, obteniendo esta información de múltiples fuentes. Además, si reciben información ambigua o no reciben algún dato importante para resolver un problema determinado, pueden obtener la información necesaria, así como guardar información importante para recuperarla en el futuro en caso de necesidad.

Estas características hacen que su aplicación en los campos de análisis de datos y de inteligencia artificial permita el desarrollo de herramientas muy potentes que pueden suponer una gran ventaja en la toma de decisiones. Actualmente, la computación cognitiva se utiliza en numerosas aplicaciones tales como programación en lenguaje natural, redes neuronales, robótica y realidad virtual.

Es por ello que los sistemas de computación cognitiva se están comenzando a utilizar en diversos ámbitos como la educación o la salud. Los asistentes personales son un ejemplo claro de computación cognitiva.

2.6 Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural es un campo de la inteligencia artificial que estudia la interacción entre el lenguaje humano y las computadoras. Trata de diseñar mecanismos que sean eficientes computacionalmente y que permitan simular la comunicación por medio de programas.

Los modelos relacionados con el procesamiento de lenguaje natural se enfocan sobre todo a los aspectos generales cognitivos humanos y a la organización de la memoria. Hasta finales de 1980, estos modelos consistían en un conjunto de reglas complejas diseñadas a mano. Sin embargo, hubo una revolución con la introducción de algoritmos de aprendizaje automático para realizar el procesamiento del lenguaje.

Esta revolución permitió el desarrollo de los primeros sistemas de traducción automática debido a la elaboración de estudios muy relevantes como las teorías lingüísticas de Noam Chomsky y la ley de Moore. Además, se utilizaron los primeros algoritmos de aprendizaje automático como los árboles de decisión.

Generalmente, el lenguaje natural tiene algunas características que dificultan el procesamiento del mismo [27]. En primer lugar, las lenguas naturales son ambiguas a muchos niveles diferentes. A nivel léxico, una misma palabra puede tener varios significados diferentes, por lo que se deberá distinguir el correcto en función del contexto. Además, el uso de anáforas y catáforas o la ironía también dificulta la comprensión y el análisis del mensaje. Por otro lado, a nivel estructural, se requiere una semántica estructurada para construir los árboles sintácticos de forma correcta.

Otra dificultad que podríamos encontrar es la detección de separación entre las palabras. En la lengua hablada no se suele hacer separaciones en forma de pausas entre las palabras. En muchas ocasiones, los sistemas separan las palabras en función de la posibilidad de mantener un sentido lógico. Además, en algunas lenguas escritas como el chino mandarín tampoco existen las separaciones entre las palabras.

Para superar todas estas dificultades con éxito, el procesamiento de lenguaje natural cuenta con varios componentes para analizar las frases recibidas desde puntos de vista diferentes. En primer lugar, se realiza un análisis morfológico en el que se estudia la estructura interna de las palabras. Posteriormente, en el análisis sintáctico se estudia la estructura sintáctica de la oración a partir de

una gramática de la lengua en cuestión. También se realiza un análisis semántico para extraer el significado de la frase, y un análisis pragmático que analiza el texto más allá de los límites de la frase.

El procesamiento de lenguaje natural tiene como objetivo simplificar la comunicación con las máquinas, ya que este resultado proporciona muchos beneficios para la sociedad. Estas técnicas permiten principalmente agilizar algunos procesos que se actualmente realizan de forma manual, automatizándolos. Estos beneficios han provocado la introducción de técnicas de procesamiento de lenguaje natural en numerosas aplicaciones. Algunos ejemplos son la clasificación documental o la traducción de documentos.

Otra de las aplicaciones más extendidas y con más éxito es la interacción conversacional entre humanos y máquinas. El desarrollo de técnicas de procesamiento de lenguaje natural es vital para el funcionamiento de los chatbots. Se utilizan para que el programa pueda obtener el significado de una frase.

2.7 Plataformas de Desarrollo

Con el auge que están sufriendo los chatbots en las industrias más relevantes de la actualidad, cada vez podemos encontrar más plataformas que facilitan el desarrollo de chatbots, algunas enfocadas para perfiles técnicos, y otras orientadas a usuarios no técnicos. En este apartado, se verán algunas de las plataformas de desarrollo de chatbots más importantes, orientadas principalmente a usuarios técnicos.

2.7.1 AIML

AIML es una variación del lenguaje XML desarrollado por Richard Wallace. Supuso la creación de uno de los chatbots más importantes hasta ahora, ALICE, ganador del premio Loebner en 2004. Hoy en día se ofrece como código abierto para cualquiera que quiera implementar un chatbot y se hay varias librerías de AIML disponibles en diversos idiomas.

Los elementos más importantes de este lenguaje son las categorías, los patrones y las plantillas. Las categorías son unidades fundamentales de conocimiento. Además, el chatbot buscara los patrones para poder responder de acuerdo a lo que el usuario ha preguntado. Cuando un patrón coincide con una categoría, entonces se le añade una plantilla a la respuesta del chatbot, específica de cada categoría.

2.7.2 Microsoft Bot Framework

Es un SDK que ofrece un conjunto de funciones y clases en dos lenguajes principales: C# y Node.js. Además, cuenta con un repositorio con ejemplos de chatbots ya creados y un emulador para poder probar el funcionamiento del chatbot de forma sencilla.

Además, Microsoft desarrollo LUIS (Language Understanding Intelligent Service) un servicio de procesamiento de lenguaje natural que incluye una biblioteca de conocimiento de NLP para que los bots lo utilicen como recurso. También cuenta con un conector que permite desplegar el bot en canales como Facebook Messenger, Skype, Teams, Slack y Telegram.

2.7.3 Dialogflow

Es una plataforma que facilita el diseño de una interfaz de usuario de conversación y su integración a aplicaciones externas. Funciona a través de comprensión de lenguaje natural, analizando las entradas de los usuarios en formato de texto o audio, y responde de maneras varias tanto a través de texto como con vez sintética.

Esta plataforma pertenece a Google por lo que ofrece la tecnología de Google Cloud y técnicas de Machine Learning desarrolladas por Google. Esto permite que Dialogflow procese miles de diálogos al día y no sea necesario proporcionar una gran cantidad de ejemplos inicialmente al contar con datos existentes.

2.7.4 Amazon Lex

Se trata de una plataforma creada por Amazon que permite desarrollar y desplegar chatbots. Su interfaz para crear diálogos es más simple que la de otras plataformas como Dialogflow o Microsoft Bot Framework, pero ofrece otras ventajas. Los bots creados con Lex pueden integrarse fácilmente con Alexa, una de los asistentes de voz más populares en la actualidad. Esto permite que los desarrolladores puedan procesar la entrada de voz o de texto de manera intercambiable.

Además, los desarrolladores también tienen acceso a la consola de administración de Amazon Web Services para acceder a Lex, crear y administrar chatbots, y guardarlos en la nube de Amazon.

2.7.5 Rasa

Rasa es un entorno de código abierto para desarrollar asistentes virtuales []. Ofrece herramientas de procesamiento y entendimiento del lenguaje natural, gestión de diálogos e integración. Su funcionamiento se basa en un módulo NLU que estructura el lenguaje natural en intenciones y entidades. Además, proporciona librerías de gestión de diálogos que determinan las respuestas del bot en función del estado de la conversación en ese momento, y del contexto. Este módulo aprende con el uso del chatbot analizando patrones en las conversaciones entre el bot y los usuarios. También cuenta con una interfaz gráfica desde donde se puede validar el chatbot y entrenarlo de forma dinámica.

2.8 Plataformas de Despliegue de Aplicaciones

Actualmente, es poco habitual encontrar aplicaciones desplegadas en servidores propios ya que en los últimos años se han desarrollado exponencialmente las plataformas que ofrecen servicios en la nube pública. Este tipo de plataformas ofrecen una mejora en el procesamiento a un precio económico, así como algunas tecnologías que permiten un ahorro en el trabajo de los desarrolladores.

En este apartado se analizarán algunas de las plataformas de despliegue más importantes de la actualidad, así como las principales tecnologías en las que se apoyan.

2.8.1 Google Cloud Platform

Google Cloud Platform o GCP es una plataforma que ha permitido la unificación en un único sitio de todas las aplicaciones mantenidas por Google como Google Search o Youtube. Consiste en un espacio virtual que permite realizar tareas de almacenamiento y gestión de datos a través de la nube de Google

Una tecnología que ofrece GCP es Kubernetes, que, a su vez, está muy relacionado con Docker. Kubernetes es una plataforma de código abierto que permite administrar cargas de trabajo y servicios. Además, proporciona una amplia variedad de servicios que permiten monitorizar los servicios y lo ocurrido dentro de la aplicación, lo que facilita la localización de errores.

Por otra parte, Docker y Docker Compose son las principales herramientas que permiten la creación y ejecución de contenedores. A través de un fichero de configuración se puede crear un servidor y ejecutar cualquier tipo de aplicación.

2.8.2 Amazon Web Services

Amazon Web Services o AWS es una plataforma de computación en la nube formada por una colección de servicios web, ofrecidos a través de internet por Amazon.com. Está distribuido en 18 regiones, cada una con diversas zonas de disponibilidad o centros de datos que proporcionan los servicios de AWS.

La mayoría de los servicios ofrecidos por AWS no están expuestos a usuarios finales, sino que ofrecen funcionalidades que otros desarrolladores puedan utilizar en sus aplicaciones. Se puede acceder a la plataforma a través de HTTP mediante protocolos REST y SOAP.

Actualmente, algunas aplicaciones importantes como Dropbox o Foursquare están desplegadas en Amazon Web Services.

2.8.3 Microsoft Azure

Microsoft Azure es un servicio de computación en la nube que permite construir, probar, desplegar y administrar aplicaciones y servicios. Proporciona software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS). Además, es compatible con numerosos lenguajes, herramientas y marcos de programación diferentes.

Utiliza un cluster que se encarga de manejar los recursos almacenados y procesamiento para las aplicaciones que se ejecutan sobre Microsoft Azure. Además, ofrece copias de seguridad automáticas con el fin de proteger información importante en el caso de que se produzca algún fallo.

Dentro de la plataforma, Windows Azure es el encargado de alojar aplicaciones y almacenamiento no relacional. Estas aplicaciones pueden estar desarrolladas en .NET, PHP, C++, Ruby o Java. También se proporcionan diversas formas de almacenamiento de datos.

2.8.4 Heroku

Heroku ofrece una plataforma como servicio (PaaS) de computación en la nube. Soporta diferentes lenguajes de programación como Java, Node.js, Python o PHP. La base del sistema operativo es Debian o Ubuntu. Por otra parte, también soporta diferentes bases de datos, entre ellas Cloudant, MongoDB, Redis o PostgreSQL.

Las aplicaciones de ejecutan desde un servidor Heroku mediante Heroku DNS Server para apuntar al dominio de la aplicación. El servidor Git de Heroku maneja los repositorios de las aplicaciones que son subidas por los usuarios.

3 Desarrollo

En este apartado se explica detalladamente el desarrollo completo del proyecto. En primer lugar, se analiza cómo se ha realizado el diseño de la historia que sigue el chatbot, y cómo es el flujo completo de la conversación. Además, se detalla cómo se han establecido los resultados del diagnóstico a partir de la conversación.

En el segundo apartado, se explican todos los elementos necesarios para desarrollar un chatbot en Rasa Open Source y cómo se han utilizado para implementar el diseño explicado anteriormente.

A continuación, se analiza cómo se ha realizado el despliegue para poder entrenar el modelo y compartirlo con usuarios reales. Por último, se explica cómo se han recopilado los datos para habilitar su posterior análisis.

3.1 Diseño del chatbot

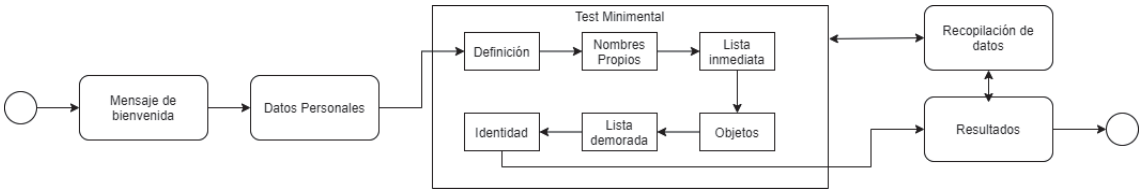
El objetivo de este trabajo es diseñar un chatbot que sea capaz de identificar síntomas de trastornos cognitivos, en especial de la enfermedad de Alzheimer. Es importante que esta enfermedad se diagnostique lo antes posible, por tanto, el chatbot realiza preguntas que están destinadas a identificar los primeros síntomas que se muestran generalmente entre los pacientes.

Entre ellos se encuentran la pérdida de memoria, condiciones del estado de ánimo o bajo rendimiento en pruebas cognitivas. Para identificar estos síntomas, la historia que sigue el chatbot consiste en un test minimental compuesto por una serie de preguntas, que evalúan 6 áreas cognitivas diferentes.

3.1.1 Estructura

En conclusión, el diseño del chatbot está estructurado en cuatro partes diferentes. En la primera parte, se realizan unas preguntas personales para obtener algunos datos relevantes del paciente. A continuación, se procede con el test minimental, que consta de seis partes diferentes. Una vez completado el test, se estiman los resultados, y por último se recogen los datos para su posterior análisis.

El siguiente diagrama muestra la estructura de una forma más visual:



3.1.2 Datos personales

Antes de empezar el test, se realizan tres preguntas de carácter más personal. Se preguntan el nombre (con intención principalmente informativa), la edad y si ha habido casos de enfermedades similares en algún antecesor familiar. Estas dos últimas preguntas establecen un contexto para deducir la probabilidad inicial que tiene el usuario de mostrar algún síntoma de un trastorno cognitivo.

La probabilidad inicial se establece en función de los dos datos recogidos, y es más alta cuanto mayor es el usuario, y se intensifica si la persona indica que sus familiares cercanos han padecido la enfermedad.

3.1.3 Test minimental

Una vez obtenidos los datos personales necesarios, comienza el test minimental. A continuación, se ofrece una explicación detallada de la evaluación de cada área cognitiva que se evalúa.

La primera área cognitiva que se evalúa es la capacidad para **denominar por definición** que tiene el usuario. Para ello, se proporciona la definición de una palabra y el usuario tiene que determinar a qué palabra u objeto se refiere. Se realizan cinco preguntas en este apartado y por cada respuesta correcta, se asigna un punto.

Se proporcionan las siguientes instrucciones: *"Esta pregunta consiste en identificar palabras a través de sus definiciones. Yo le proporcionaré la definición y tendrá que decir a qué objeto se refiere ¡Comencemos!"*

Las preguntas proporcionadas por el test son las siguientes:

Área	Pregunta	Respuesta	Puntuación
	¿Dónde se mira uno si quiere verse a sí mismo?	Un espejo	
	¿Dónde se apoya la cabeza en la cama?	La almohada	
Denominación por definición	¿Dónde se compran los medicamentos?	La Farmacia	/5
	¿Qué alimento producen las abejas?	Miel	
	¿Dónde se conservan fríos los alimentos?	La nevera	

Tabla 1: Denominación por definición

La segunda parte del test, evalúa la capacidad para **denominar nombres propios**. Para ello, se proporcionan las descripciones de 5 personas célebres y el usuario debe contestar con el nombre propio que corresponda. Por cada respuesta correcta, se asigna un punto.

Las instrucciones que se proporcionan son las siguientes: *"Esta pregunta consiste en identificar nombres propios de personas célebres. Se proporcionará una descripción y se espera el nombre de la persona correspondiente. Vamos a comenzar."*

Las preguntas son las siguientes:

Área cognitiva	Preguntas	Respuestas	Puntuación
	¿Cómo se llama el actual presidente del gobierno de España?	Pedro Sánchez	
	¿Quién es el rey de España actual?	Felipe VI	
Denominación por nombres propios	¿Cómo se llama el Papa actual?	Francisco I	/5
	¿Quién es el presidente del Real Madrid?	Florentino Pérez	
	¿Cómo se llama la reina de Inglaterra actual?	Isabel II	

Tabla 2: Denominación por nombres propios

A continuación, se comprueba la **memoria inmediata de una lista de palabras**. Se proporciona una lista de diez palabras e inmediatamente después, se pide formular todas aquellas palabras que se recuerden. Se otorga un punto por cada palabra que se recuerda correctamente.

Las instrucciones que recibe el usuario son: *"A continuación, le voy a proporcionar una lista de 10 objetos y tendrá que memorizar el máximo número posible de ellos."*

"¿Cuántos de los objetos puede repetir?"

La lista de palabras se muestra a continuación.

Área	Pregunta	Puntuación
Recuerdo inmediato de una lista de palabras	Recuerdo de la lista: [Café, Tomate, Arroz, Queso, Sal, Agua, Pan Champú, Atún, Pasta]	/10

Tabla 3: Recuerdo de una lista inmediata

La cuarta pregunta evalúa la capacidad de **denominar objetos**. Para ello, se muestran imágenes de cinco objetos diferentes y el usuario tiene que nombrar el objeto al que hace referencia. Se asigna un punto por cada respuesta correcta.

Las instrucciones que se proporcionan son: "La siguiente pregunta consiste en denominar objetos. Para ello, le voy a enseñar una serie de imágenes y debe responder a qué objeto corresponden."

La tabla a continuación, muestra los objetos que se enseñan:

Área Cognitiva	Pregunta	Respuesta	Puntuación
		Un botón	
		Un zapato	
Denominación de objetos		Un lápiz	/5
		Un reloj	
		Una guitarra	

Tabla 4: Denominación por objetos

La quinta pregunta evalúa el **recuerdo demorado de una lista de palabras**. Para ello, se pide al usuario que enuncie las palabras de la pregunta anterior de las que se acuerda. Se otorga un punto por cada palabra correcta.

Se proporcionan las siguientes instrucciones: *"Ahora deberá decirme los objetos de la lista anterior de los que se acuerda."*

Área	Pregunta	Puntuación
Recuerdo demorado de una lista de palabras	Recuerdo de la lista: [Café, Tomate, Arroz, Queso, Sal, Agua, Pan Champú, Atún, Pasta]	/10

Tabla 5: Lista demorada de palabras

La última pregunta, determina la capacidad de **identificar la identidad** del chatbot. Consiste en comprobar si la persona es capaz de determinar si está manteniendo una conversación con una máquina o con una persona real. Si se responde correctamente se asigna un punto y si no se resta un punto al total de las preguntas anteriores.

Se proporciona lo siguiente: *"Por último, podría decirme quién soy yo, ¿un asistente virtual o una persona real?"*

Área	Pregunta	Puntuación
Identificación de identidad	¿Soy un asistente virtual o una persona real?	{-1,1}

Tabla 6: Identificación de identidad

3.1.4 Resultados

Una vez completado el test minimental, el programa calcula el resultado sumando la totalidad de los puntos obtenidos por el usuario. A continuación, en función del resultado, se determina si el usuario presenta síntomas de un deterioro cognitivo.

Se pueden obtener tres resultados diferentes:

- Si el usuario ha obtenido una puntuación **menor a 21 puntos** sobre el total de 35, se determina que el resultado es muy bajo, y, por lo tanto, puede presentar síntomas de la enfermedad de Alzheimer.
  Se informa al usuario a través del siguiente mensaje: *"Los resultados del test son más bajos de lo esperado. Esto podría llegar a indicar los primeros síntomas de un trastorno cognitivo como la enfermedad de Alzheimer. Los resultados de este test no son definitivos, pero le recomiendo que consulte con personal sanitario cualificado"*
- Si se obtiene una puntuación de **entre 21 y 29**, se determina que el resultado no es lo suficientemente alto, por lo que el usuario podría presentar síntomas de un Deterioro Cognitivo Leve, que eventualmente podría desembocar en la enfermedad de Alzheimer.
  Se comunica el resultado de la siguiente manera: *"Los resultados del test no son tan altos como lo esperado. Esto podría llegar a indicar los primeros síntomas de un trastorno cognitivo como el Deterioro Cognitivo Leve. Los resultados de este test no son definitivos, pero le recomiendo que consulte con personal sanitario cualificado."*

- Finalmente, si el usuario obtiene una puntuación **mayor a 29**, no se detecta ningún síntoma de un deterioro cognitivo. Se comunican los resultados con el siguiente mensaje. *"Los resultados del test son satisfactorios. No muestran ningún síntoma de algún trastorno cognitivo."*

3.2 Implementación:

Una vez completado el diseño de la historia, procedí con la implementación del mismo. El chatbot fue desarrollado con Rasa Open Source, que como ya se ha explicado anteriormente es un marco de desarrollo de chatbots.

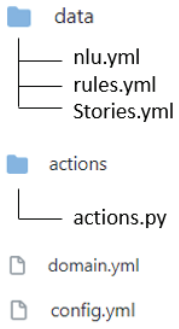
3.2.1 Instalación y creación del proyecto

Para poder instalar Rasa Open Source, y crear un primer proyecto es necesaria la instalación previa de Python. Además, es recomendable la configuración de un entorno virtual con una herramienta como Virtualenv.

Una vez está lista la configuración inicial, para crear un proyecto demo con rasa, habría que acceder al directorio donde queremos crear el proyecto nuevo, y utilizar el siguiente comando:

```
rasa init
```

Este comando crea un proyecto con un ejemplo muy simple de un chatbot desarrollado con rasa. El proyecto creado tiene una estructura similar a la mostrada por la siguiente imagen.



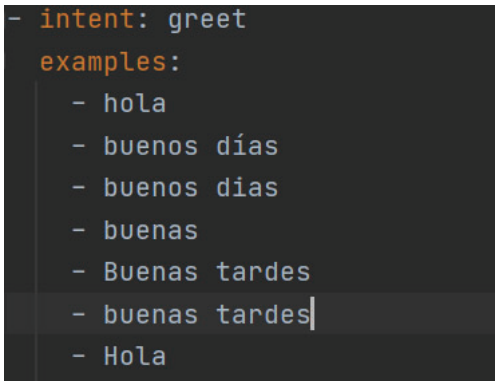
3.2.2 Creación del dataset de entrenamiento

En este apartado se describen todos los elementos necesarios para desarrollar un chatbot en Rasa.

3.2.2.1 Intenciones

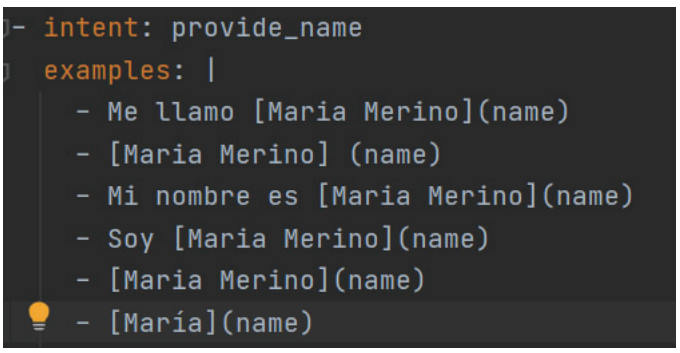
El proyecto generado contiene una carpeta denominada “data” en la que encontramos tres de los ficheros más importantes para el desarrollo del chatbot: nlu.yml, rules.yml y stories.yml.

En el archivo nlu.yml es donde encontraremos los ejemplos de entrenamiento que más adelante usará el modelo generado. Este fichero está compuesto por **intenciones**, que se podría definir como lo que el usuario intenta o desea expresar, y se definen a partir de un conjunto de ejemplos asociados a ellas.



Como podemos apreciar en la imagen, las intenciones se definen proporcionando el nombre que le queremos asignar, y posteriormente un listado de ejemplos que el usuario podría emplear para expresar esa intención.

En este archivo nlu.yml también podemos encontrar y definir **entidades.** Las entidades son piezas de información estructurada, contenida dentro del mensaje del usuario. Nos sirven para estructurar y localizar información de una manera más sencilla.



Las entidades se expresan en los ejemplos de las intenciones y se definen poniendo entre corchetes un ejemplo de la entidad y entre paréntesis el nombre de dicha entidad.

A la hora de definir intenciones y entidades, no es necesario proporcionar todos los posibles ejemplos asociados a una intención en concreto, ya que cuando el chatbot comienza a usarse, se autogeneran más ejemplos a partir de la interacción con usuarios. Sin embargo, es importante proporcionar suficientes ya que el funcionamiento del chatbot depende de la correcta interpretación de estas intenciones.

3.2.2.2 Pipeline

Un **modelo NLU** es lo que usa el chatbot para procesar las respuestas de los usuarios y extraer las intenciones. Los modelos se pueden entrenar para que puedan mejorar sus predicciones sobre las intenciones y las entidades en nuevos mensajes de usuarios desconocidos, incluso cuando el mensaje recibido no concuerda con ninguno de los ejemplos proporcionados.

Los **pipelines** o **tuberías de entrenamiento** es lo que ayuda a crear el modelo. Consiste en una secuencia de pasos que procesan la información para obtener patrones.

Los **Word embeddings** convierten palabras en vectores o representaciones numéricas para poder ser procesados por algoritmos de procesamiento de lenguaje natural. Gracias a esta técnica, los pipelines son capaces de entrenar los modelos.

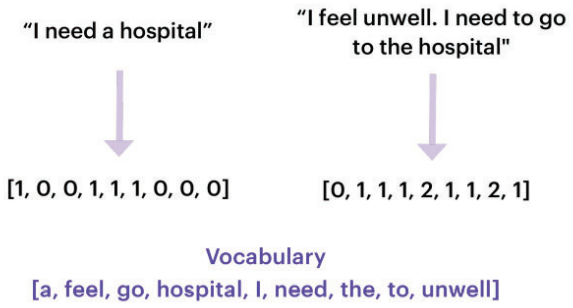
En Rasa, podemos encontrar dos opciones diferentes a la hora de configurar los pipelines. La primera opción es usar un embedding entrenado previamente, que usa la librería de Spacy para entrenar el modelo. Esto ofrece algunas ventajas ya que incluso con poca información de entrenamiento, podemos conseguir un modelo preciso. Sin embargo, al ser una librería ya hecha, no ofrece muchas opciones para personalizar la información en función del chatbot y no cubre palabras específicas o tecnicismos concretos.

La segunda opción, la elegida en este proyecto, es usar un embedding supervisado. Al contrario del primero, se necesita más información de entrenamiento para que el bot funcione de forma correcta. Por otro lado, el

modelo se puede adaptar a palabras específicas del contexto en el que se use el bot ya que se entrena en el contexto real.

Una vez elegida la configuración del pipeline, debemos seleccionar las técnicas que queremos que use le pipeline para convertir las palabras en vectores.

- **Tokenizador**: En primer lugar, establecemos un tokenizador, que recibe frases entreras u oraciones y las devuelve, divididas en trozos más pequeños, para facilitar su procesamiento. En este proyecto, se ha usado el **Whitespace Tokenizer**, que divide las palabras en función de los espacios en blanco que se encuentra.
- **Names Entity Recognition o NER**: El siguiente paso es establecer un NER, que extrae las entidades de los mensajes de los usuarios. El elegido para este proyecto se llama **RegexFeaturizer.** Este componente marca si ha encontrado una expresión relevante en el mensaje o no, y más adelante otro componente lo extrae.
- **Clasificador de intenciones**: Recoge la salida del NER mencionado anteriormente y establece la probabilidad de que un mensaje tenga una intención en concreto. En este proyecto, se han usado varios clasificadores diferentes, que se complementan debido a sus características:
    - **CountVectorsFeaturizer**: No se centra en el orden de las palabras, sino que estima cada cuánto aparecen las palabras de los ejemplos proporcionados, en el mensaje recibido.



    - **LexicalSyntacticFeaturizer:** Crea elementos léxicos y sintácticos de cada mensaje recibido para ayudar a la extracción de entidades.
    - **EntitySynonymMapper:** Modifica las entidades extraídas por otros componentes ya que mapea las entidades que considere que tienen el mismo significado.

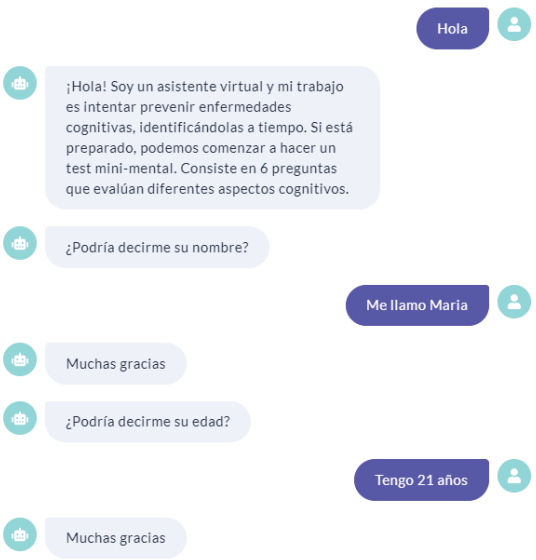
- **DIETClassifier:** Funciona tanto como clasificador de intenciones como extractor de entidades.

3.2.2.3 Historias

Una historia en Rasa es la unidad básica de entrenamiento de diálogo. Las historias se escriben con una estructura definida en el archivo stories.yml, en la carpeta "data". Como podemos ver en la imagen a continuación, su estructura representa un ejemplo de conversación entre el usuario y el chatbot.

```yaml
- story: datos_personales
  steps:
  - action: utter_bienvenida
  - action: utter_ask_name
  - intent: provide_name
  - action: utter_gracias
  - action: utter_ask_age
  - intent: provide_age
  - action: utter_gracias
```

Para definir correctamente la historia, debemos proporcionar un nombre y las acciones que realiza el chatbot al recibir una intención en concreto. En la siguiente imagen podemos ver cómo sería esta historia representada en la realidad.



Las **acciones** son las respuestas que el chatbot puede proporcionar. Se pueden proporcionar varias posibles respuestas para una sola acción. Para definirlas, se proporciona un nombre y pueden estar compuestas por texto plano:

```
utter_ask_name:
- text: Antes de empezar me gustaría preguntarle su nombre
- text: ¿Podría decirme su nombre?
```

También pueden incluir imágenes:

```
- text: Objeto 1
  image: "https://d25rq8gxcq0p71.cloudfront.net/dictionary-images/324/button.jpg"
```

O recuperar valores de entidades guardados anteriormente:

```
utter_slots_values:
- text: |-
    por favor, confirma que los datos son correctos:
     - name: {name}
     - age: {age}
     - antecedentes: {antecedentes}
```

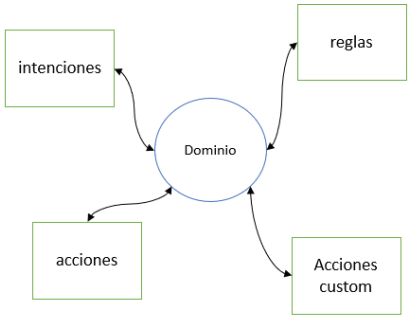
Las **reglas** son elementos muy similares a las historias, pero definen comportamientos que se deben ejecutar por el asistente. Es decir, definen la acción o las acciones que se deben ejecutar al recibir una intención en concreto. Se establecen en el archivo rules.yml.

```
- rule: check prob inicial
  steps:
    - intent: check_prob_inicial
    - action: probabilidad_inicial
```

3.2.2.4 Dominio

Es un aspecto fundamental para la gestión del diálogo. En el archivo domain.yml se define el contexto en el que el chatbot va a funcionar. Especificamos:

- Un listado de las intenciones y las entidades que le proporcionamos al chatbot.
- Las respuestas que el bot puede proporcionar, que pueden ser tanto acciones como acciones personalizadas.
- Los valores importantes que se deben recordar.



3.2.2.5 Slots

Actúan como la memoria del chatbot, son utilizados por el asistente para recordar detalles importantes y aplicarlos apropiadamente para guiar la conversación.

Pueden extraerse tanto de los mensajes de los usuarios como de recursos externos, por ejemplo, una base de datos.

Para definir un slot, se debe proporcionar su nombre y el tipo. Entre los diferentes tipos podemos encontrar texto (text), valores numéricos (float) o listas (list).

```
name:
  type: text
  influence_conversation: false
```

3.2.2.6 Acciones Personalizadas

Las acciones personalizadas se definen en el archivo actions.py en la carpeta actions. Son acciones que contienen código, que puede estar compuesto por, desde una simple respuesta hasta una integración con backend.

Para cada acción personalizada hay que definir en el fichero de Python una clase que contenga dos funciones: *name* y *run.* La función *name* proporciona un nombre a la acción customizada, y le método *run* contiene el código que se ejecutará al llamar a la acción.

En la función run contamos con dos elementos importantes:

- Tracker: nos permite mantener actualizada la información de lo que está pasando en el diálogo: las intenciones estimadas, las entidades extraídas o información importante sobre el contexto.
- Dispatcher: Elemento que permite mandar información de vuelta al usuario.

3.2.2.7 Políticas

Las políticas son componentes que intervienen en el entrenamiento del modelo y condicionan su comportamiento. Su configuración se define en el archivo config.yml. Está definida por un conjunto de nombres de políticas con una serie de parámetros opcionales que podemos establecer.

Las políticas que se han decidido establecer para este proyecto son las siguientes:

- **RulePolicy**: Es una política que maneja las partes de la conversación con comportamiento fijo. Hace las predicciones en función de las reglas que se hayan establecido.
- **TEDPolicy**: Es una arquitectura multifuncional que se encarga de predecir las acciones futuras que el chatbot debe realizar. También puede extraer entidades de las conversaciones.
- **MemorizationPolicy**: Se encarga de memorizar las historias definidas en los datos entrenados. Si la conversación que se está manteniendo concuerda con alguna de las historias proporcionadas es capaz de predecir la siguiente acción.

3.2.3 Implementación del chatbot

Una vez hemos definido todos los elementos necesarios para la implementación del chatbot en el marco de Rasa Open Source, en este apartado se explica detalladamente como se ha implementado el chatbot.

Está estructurado de una forma parecida a la explicación del diseño para facilitar la comprensión de la estructura del chatbot.

3.2.3.1 Datos personales

Como explicado en el diseño de la historia, la primera parte de la historia consiste en tres preguntas que recopilan información sobre el nombre del usuario, su edad y si ha habido casos de trastornos cognitivos en familiares cercanos.

Para ello, he definido cuatro intenciones, una para cada posible respuesta: proporcionar el nombre, la edad, expresar que sí ha habido casos o expresar que no.

Además, he definido las acciones del chatbot necesarias para realizar las tres preguntas y tres acciones personalizadas, dos de ellas para asignar un valor "sí" o "no" a la última pregunta en función de la respuesta del usuario; y una para establecer el valor de la probabilidad inicial que tiene el usuario de presentar síntomas de un trastorno cognitivo en función de los datos obtenidos.

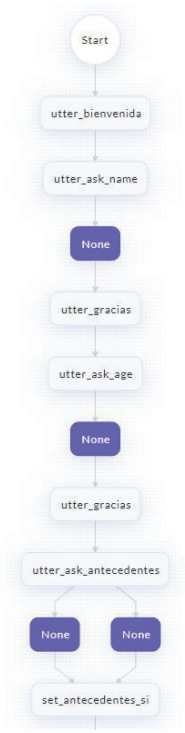
El nombre y la edad se obtienen de la respuesta dada por el usuario a partir de las intenciones *provide_name* y *provide_age*:

```
- intent: provide_name
  examples: |
    - Me llamo [Maria Merino](name)
    - [Maria Merino] (name)
    - Mi nombre es [Maria Merino](name)
    - Soy [Maria Merino](name)
    - [Maria Merino](name)
    - [María](name)
```

```
- intent: provide_age
  examples: |
    - Tengo [21](age) años
    - [21](age)
    - [21](age) años
    - Mi edad es [21]
    - tengo [21](age) años
```

Con todo ello, la historia y la estructura de esta parte de la conversación se ven reflejadas en las siguientes imágenes:

```
version: "2.0"
stories:
- story: interactive_story_1
  steps:
  - intent: greet
  - action: utter_greet
  - action: utter_ask_name
  - intent: provide_name
    entities:
    - name: Maria Merino
  - slot_was_set:
    - name: Maria Merino
  - action: utter_gracias
  - action: utter_ask_age
  - intent: provide_age
    entities:
    - age: '21'
  - slot_was_set:
    - age: '21'
  - action: utter_gracias
  - action: utter_ask_antecedentes
  - intent: antecedentes_si
  - action: set_antecedentes_si
  - action: probabilidad_inicial
```



Este sería un ejemplo en el que el usuario ha contestado que sí ha habido casos de familiares cercanos con trastornos cognitivos, por lo que se ha recibido la intención antecedentes_si, y se ha guardado el valor con la acción personalizada set_antecedente_si. En otro caso, se hubiese percibido la intención antecedentes_no y se hubiese ejecutado set_antecedentes_no.

Por último, podemos ver que tras finalizar las preguntas se ejecuta la acción personalizada probabilidad_inicial, que guarda el valor en el slot *Prob*.

3.2.3.2 Test minimental

Una vez se recogen los datos personales necesarios, se procede con el test minimental. Consta de 6 partes principales, cada una destinada a evaluar un área cognitiva particular.

Las preguntas destinadas a evaluar la denominación por definición, denominación por nombres propios y denominación por objetos siguen una estructura muy similar, por lo que se explicará siguiendo el ejemplo de la denominación por definición.

Todas ellas cuentan con una acción destinada a proporcionar el enunciado de la pregunta:

```
utter_enunciado_definiciones:
- text: Esta pregunta consiste en identificar palabras a través de sus definiciones.
Yo le proporcionaré la definición y tendrá que decir a qué objeto se refiere. ¡Comencemos!
```

Y otra para anunciar que la pregunta ha finalizado:

```
utter_fin_pregunta:
- text: ¡Perfecto! Hemos finalizado esta pregunta. Procedemos con la siguiente.
```

Además, se proporciona una acción para la definición de cada una de las cinco preguntas que conforman la parte correspondiente del test. En el caso de la denominación por definición, la primera pregunta está definida de la siguiente forma:

```
utter_definiciones1:
- text: ¿Dónde se mira uno si quiere verse a si mismo?
```

Y, a continuación, la intención que se espera es o bien la respuesta correcta, o incertidumbre, o una respuesta incorrecta. Para cada una de ellas, se definen tres intenciones diferentes:

```
- intent: definiciones1
  examples: |
    - Un espejo
    - un espejo
    - En el espejo
    - espejo
    - Espejo
    - En un espejo
```

```
- intent: dudas
  examples: |
    - No lo sé
    - No se
    - Y yo que se
    - A saber
    - siguiente
    - no lo se
    - ?
    - no sé
    - Siguiente
    - Siguiente pregunta
```

```
- intent: nlu_fallback
  examples: |
    - un sofa
    - Isable segunda
    - Juan Carlos I
    - a su mujer
    - sabana
    - microondas
    - Juan Carlos
    - un supermercado
```

Además, cada vez que se reconoce la intención destinada a proporcionar una respuesta correcta, se llama a una acción customizada que suma un punto al resultado actual de la pregunta que se esté evaluando.

Se establece un slot para el resultado de cada parte del test. En este caso, el slot recibe el nombre de *res_def* y se actualiza de la siguiente forma:

```python
class SetResultadoDefs(Action):
    # return the name of the action
    def name(self) -> Text:
        return "set_resultado_defs"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        res = tracker.slots.get("res_def")

        resultado = (int(res)+1)
        print(resultado)
        return [SlotSet("res_def", resultado)]
```

Por tanto, la estructura de las preguntas destinadas a evaluar la denominación por definición quedaría de la siguiente forma. Diferenciamos los dos posibles casos: que se obtenga una respuesta correcta y se actualice el resultado, o que no obtenga una respuesta correcta y pase directamente a la siguiente pregunta.

Caso de las respuestas correctas:

```yaml
version: "2.0"
stories:
- story: preg_definiciones
  steps:
  - intent: preg_defs
  - action: utter_enunciado_definiciones
  - action: utter_definiciones1
  - intent: definiciones1
  - action: set_resultado_defs
  - action: utter_definiciones2
  - intent: definiciones2
  - action: set_resultado_defs
  - action: utter_definiciones3
  - intent: definiciones3
  - action: set_resultado_defs
  - action: utter_definiciones4
  - intent: definiciones4
  - action: set_resultado_defs
  - action: utter_definiciones5
  - intent: definiciones5
  - action: set_resultado_defs
  - action: utter_fin_pregunta
  - action: utter_enunciado_propios
  - action: utter_propios1
```

En caso de respuestas incorrectas:

```yaml
version: "2.0"
stories:
- story: preg_definiciones_mal
  steps:
  - intent: preg_defs
  - action: utter_enunciado_definiciones
  - action: utter_definiciones1
  - or:
    - intent: dudas
    - intent: nlu_fallback
  - action: utter_definiciones2
  - or:
    - intent: dudas
    - intent: nlu_fallback
  - action: utter_definiciones3
  - or:
    - intent: dudas
    - intent: nlu_fallback
  - action: utter_definiciones4
  - or:
    - intent: dudas
    - intent: nlu_fallback
  - action: utter_definiciones5
  - or:
    - intent: dudas
    - intent: nlu_fallback
  - action: utter_fin_pregunta
  - action: utter_enunciado_propios
  - action: utter_propios1
```

En ambos casos, podemos apreciar que, una vez finalizada la pregunta, se llama al enunciado de la siguiente parte, que en este caso es denominación por nombres propios, y a la definición de la primera pregunta dentro de esta parte.

Por otra parte, las partes destinadas a evaluar el recuerdo inmediato de una lista de palabras y la destinada a evaluar el recuerdo demorado de una lista de palabras también cuentan con una estructura muy similar.

Por tanto, se explicarán siguiendo el ejemplo del recuerdo inmediato de una lista de palabras.

Ambas preguntas comparten el enunciado:

```yaml
utter_pregunta_lista:
- text: A continuación, le voy a proporcionar una lista de 10 objetos, y tendrá que
  memorizar el máximo número posible. La lista es la siguiente
  Café, Tomate, Arroz, Queso, Sal, Agua, Pan, Champú, Atún, Pasta
```

Además, se ha establecido una acción para realizar la pregunta. En este caso es:

```
utter_ask_lista:
- text: ¿Cuántos  de los objetos puede repetir?
```

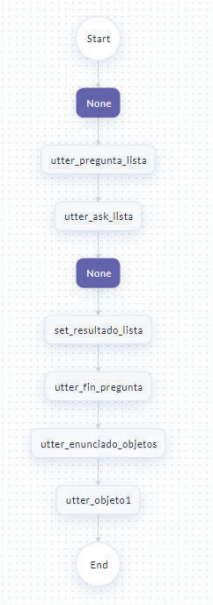
Por último, se ha establecido una intención que comprueba si las palabras de la respuesta recibida forman parte de la lista proporcionada, así como una acción personalizada que cuenta el tamaño de las palabras proporcionadas correctamente, y las almacena en un slot: res_lista_inmediata:

```python
class SetResultadoListaInmediata(Action):
    # return the name of the action
    def name(self) -> Text:
        return "set_resultado_lista"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        res = tracker.slots.get("lista_compra")
        print(res)
        resultado = len(res)
        print(resultado)
        return [SlotSet("res_lista_inmediata", resultado)]
```

Por tanto, la estructura de la historia queda de la siguiente forma:

```
version: "2.0"
stories:
- story: preg_lista_inmediata
  steps:
  - intent: lista_inmediata
  - action: utter_pregunta_lista
  - action: utter_ask_lista
  - intent: recibir_lista_inmediata
  - action: set_resultado_lista
  - action: utter_fin_pregunta
  - action: utter_enunciado_objetos
  - action: utter_objeto1
```



Finalmente, la parte del test destinada a la identificación de identidad está formado por el enunciado de la pregunta:

```
utter_pregunta_chatbot:
- text: Por último, podría decirme quién soy yo, ¿un assistente virtual o una persona real?
```

También se definen dos posibles intenciones en función de la respuesta, con dos acciones customizadas para establecer el resultado. A continuación, se proporcionan los del caso en el que la persona responde correctamente.

```
- intent: chatbot_bien
  examples: |
    - Un assitente virtual
    - un chatbot
    - Un robot
    - Un chatbot
    - un robot
    - chatbot
    - asistente virtual
    - asistente
    - virtual
```

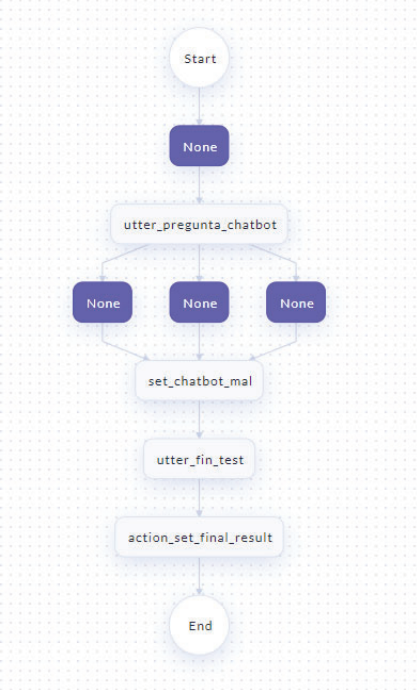
```
class SetChatbotBien(Action):
    # return the name of the action
    def name(self) -> Text:
        return "set_chatbot_bien"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        res = 1

        return [SlotSet("res_chatbot", res)]
```

Por tanto, la estructura de la pregunta queda de la siguiente forma:

```
version: "2.0"
stories:
- story: preg_chatbot_mal
  steps:
  - intent: quien_soy
  - action: utter_pregunta_chatbot
  - intent: chatbot_mal
  - or:
    - intent: dudas
    - intent: chatbot_mal
    - intent: nlu_fallback
  - action: set_chatbot_mal
  - action: utter_fin_test
  - action: action_set_final_result
```



En estas imágenes podemos ver la estructura de la historia si el usuario responde de forma incorrecta a esta pregunta. Además, podemos ver que puesto que es la última pregunta, cuando finaliza se llama a la función action_set_final_results, que es la función que se encarga de establecer el resultado final.

3.2.3.3 Resultados

Una vez completado el test minimental, se calcula el resultado total para establecer el resultado.

Para ello, se ha implementado una acción customizada que recupera los valores de todos los slots mencionados anteriormente y los suma. En función del resultado final muestra un mensaje diferente:

```python
class ActionSetFinalResult(Action):
    def name(self) -> Text:
        return "action_set_final_result"

    async def run(
            self,
            dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any],
    ) -> List[Dict[Text, Any]]:
        prob = tracker.slots.get("prob")
        r1 = tracker.slots.get("res_def")
        r2 = tracker.slots.get("res_propios")
        r3 = tracker.slots.get("res_lista_inmediata")
        r4 = tracker.slots.get("res_lista-despues")
        r5 = tracker.slots.get("res_objetos")
        r6 = tracker.slots.get("res_chatbot")
        res = int(r1) + int(r2) + int(r3) + int(r4) + int(r5) + int(r6) - int(prob)
        dispatcher.utter_message("El resultado es", res)
        if res < 20:
            dispatcher.utter_message(text= "Hemos llegado hasta PROB 2")
            dispatcher.utter_message(response="utter_res2")
        elif 20 < res < 29:
            dispatcher.utter_message(text="Hemos llegado hasta PROB 1")
            dispatcher.utter_message(response="utter_res1")
        elif res > 29:
            dispatcher.utter_message(text= "Hemos llegado hasta PROB 0")
            dispatcher.utter_message(response="utter_res0")
        return []
```

3.3 Despliegue

Una vez diseñado e implementado el chatbot con Rasa Open Source, procedí a realizar el despliegue para poder comenzar a hacer pruebas con usuarios reales. Para ello, en primer lugar, decidí desplegarlo en Rasa X, la interfaz gráfica habilitada por Rasa para probar y corregir los modelos.
Rasa X también ofrece la opción de generar un link para poder compartir el chatbot con usuarios reales en una interfaz diseñada por Rasa.

3.3.1 Despliegue de Rasa X

Para realizar el despliegue del chatbot en Rasa X, en primer lugar, se debe crear una instancia de una máquina virtual en Google Cloud Platform. El sistema operativo elegido es Ubuntu 20.04 LTS con un tamaño de 100GB, y la zona geográfica más parecida, que es "europe-west4 (Netherlands)". Es importante habilitar el tráfico HTTP y HTTPS y el acceso completo a todas las Cloud APIs.

Una vez creada la máquina virtual, hay que copiar el link de acceso, que será parecido a:

```
gcloud beta compute ssh --zone "europe-west1-b" "rasa-x-chatbot" --project "helical-door-314218"
```

Este link, da acceso a la máquina virtual creada previamente, al ejecutarlo desde una ventana de una terminal. En este caso, habría que ejecutarlo desde un sistema operativo Linux, como Ubuntu.

Una vez conectado a Google Cloud, habría que descargarse el archivo de ejecución de Rasa X a través del siguiente comando:

```
curl -sSL -o install.sh https://storage.googleapis.com/rasa-x-releases/0.40.1/install.sh
```

El siguiente comando, permite instalar todos los archivos de las carpetas que se descargan con el comando previo, en un nuevo directorio llamado /etc/rasa:

```
sudo bash ./install.sh
```

Una vez descargados e instalados todos los archivos necesarios, creamos la imagen en los contenedores necesarios y los ejecutamos con Docker Compose.

Primero accedemos al directorio indicado y luego ejecutamos Docker-compose en el background del sistema.

```
curl -sSL -o in cd /etc/rasa
```

```
sudo docker-compose up -d
```

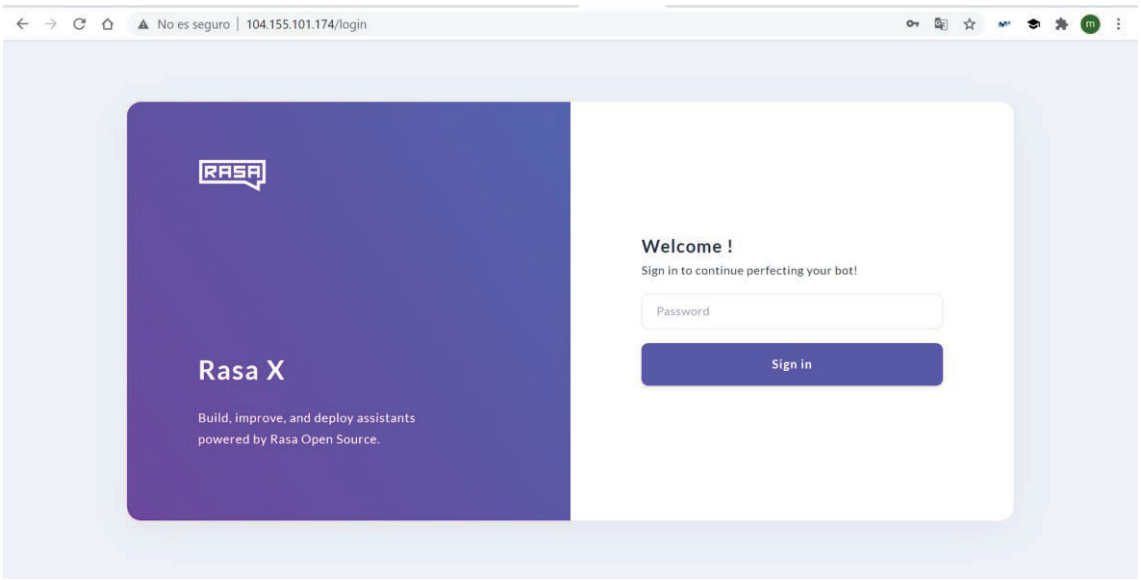
Por último, establecemos la contraseña con la que vamos a acceder posteriormente a la interfaz.

```
cd /etc/rasa
```

```
sudo python3 rasa_x_commands.py create –update admin me <contraseña>
```

3.3.2 Configuración de Rasa X

En este apartado, de explica cómo realizar la conexión entre Rasa X y el chatbot implementado previamente con Rasa Open Source. Para ello, en primer lugar, hay que acceder a la IP proporcionada por la máquina virtual creada anteriormente en Google Cloud, que nos llevará a interfaz de Rasa X:



Se debe introducir la contraseña establecida en el anterior apartado y proceder con la conexión al proyecto. Para ello, en primer lugar, hay que crear un repositorio en Github y subir el proyecto en el que se ha desarrollado el chatbot.

Luego, realizar la conexión mediante el link SSH del proyecto generado e insertamos en Github una llave de despliegue proporcionada por Rasa X.

Una vez conectada la interfaz gráfica a Github, hay que crear una imagen del servidor de acciones personalizadas. Para ello, hay que crear un archivo .github/workflows/action_server.yml, que permite construir una imagen nueva en Docker cada ver que se realizar algún cambio en las acciones personalizadas, y los cambios se suben a GitHub.

```yaml
on:
  push:
    branches:
      - main
    paths:
    - 'actions/**'

jobs:
    build:
        # ...
        steps:
            # ...
            - name: Build an action server
              uses: RasaHQ/rasa-action-server-gha@main
              with:
                docker_image_name: 'rasahq/action-server-example'
                # More details on how to use GitHub secrets:
                # https://docs.github.com/en/actions/configuring-and-managing-workflows/creating-and-storing-encrypted-secrets
                docker_registry_login: ${{ secrets.DOCKER_HUB_LOGIN }}
                docker_registry_password: ${{ secrets.DOCKER_HUB_PASSWORD }}
                # More details about github context:
                # https://docs.github.com/en/actions/reference/context-and-expression-syntax-for-github-actions#github-context
                docker_image_tag: ${{ github.sha }}
            # ...
```

A continuación, debemos empujar los cambios a la máquina virtual de Google Cloud. Para ello, hay que conectarse a la máquina virtual con el comando especificado anteriormente. Una vez conectado, debemos modificar el directorio /etc/rasa, creando un fichero docker-compose.override.yml, que empuje la imagen creada por Docker a la máquina virtual.

```yaml
version: '3.4'
services:
  app:
    image: <image:tag>
```

Finalmente, replegamos los contenedores de Docker-compose con:

```
docker-compose down
```

Y los volvemos a reiniciar, con la nueva imagen que hemos creado con:

```
docker-compose up -d
```

3.3.3 Entrenar y compartir el bot

Una vez configurados los elementos necesarios para que funcione el chatbot en Rasa X, solo faltaría cargar un modelo en la interfaz y entrenarlo. Para ello, entrenamos un modelo del chatbot con el siguiente comando desde una terminar en el directorio donde se encuentre nuestro proyecto:

```
rasa train
```

Con este comando, se generará un archivo .tar que podemos cargar en Rasa X mediante una API HTTP. Una vez cargado el modelo, se puede probar hablando con él desde la interfaz, entrenarlo de forma interactiva, y corregir los posibles fallos que se encuentren. También se pueden revisar las conversaciones realizadas con el bot por otros usuarios.

Finalmente, se puede generar un link desde la interfaz para que compartir el bot de forma que cualquier usuario pueda probarlo y usarlo.

4 Conclusiones y Líneas Futuras

4.1 Conclusiones personales

Tras haber finalizado el diseño y la implementación de este proyecto, hay algunas conclusiones personales que me gustaría compartir

Este proyecto ha sido complicado de desarrollar por diversos motivos. La falta de conocimientos tanto sobre cómo se realiza el diagnóstico del Alzheimer como sobre el funcionamiento de Rasa, que nunca había utilizado, ha hecho que las primeras semanas del proyecto las dedicase a investigar profundamente sobre estos temas, limitando el tiempo que tenía para la implementación.

Sin embargo, ha sido un proyecto muy gratificante de desarrollar. En mi opinión, las nuevas tecnologías han sido creadas para ayudar a las personas y facilitar las tareas cotidianas. Yo pienso que este proyecto es un gran ejemplo de una herramienta con mucho potencial para ofrecer un gran beneficio a la sociedad.

Además, tener la oportunidad de aprender sobre algunas de las tecnologías que más se están desarrollando en el momento como son los chatbots ha sido increíble y ha aprendido muchos conocimientos que podré utilizar en mi futuro profesional.

Por otra parte, el proyecto desarrollado es una primera versión, que pese a ser funcional, hay algunas mejoras que se podrían realizar, y me encantaría poder seguir trabajando en el proyecto, o incluso con ayuda de otras personas, para poder conseguir un producto que poder utilizar en el mundo real.

En conclusión, a pesar de la dificultad del proyecto, ha sido un proyecto muy gratificante de desarrollar y, a pesar de las mejoras que se podrían realizar, estoy muy contenta con el resultado obtenido. Por último, me gustaría agradecer a todas las personas que me han ayudado a desarrollar este Trabajo de Fin de Grado.

4.2 Líneas futuras

Este proyecto presenta una herramienta que cuenta con un potencial muy valioso. Puesto que este desarrollo consiste esencialmente en una primera versión del desarrollo, hay algunos aspectos que se podrían mejorar en futuras versiones, de cara a mejorar su funcionamiento y la usabilidad.

4.2.1 Análisis de los datos:

En primer lugar, este proyecto recoge los datos de los resultados de los usuarios y sería muy interesante estudiar la posibilidad de analizarlos y sacar conclusiones a partir de ellos.

Por otra parte, el diagnóstico que realiza el programa, se establece en función de los datos obtenidos en el test minimental y de los datos personales. Sin embargo, una vez se hayan recogido datos suficientes se podría implementar un algoritmo supervisado de clasificación, como un árbol de clasificación, que clasificase a los pacientes en función de los síntomas que muestran, a partir de los datos recogidos previamente.

La implementación de un modelo de clasificación mejoraría el diagnóstico del chatbot con seguridad, e incluso se podría analizar su precisión a partir de los datos recogidos previamente.

4.2.2 Interfaz web

Por otra parte, pese a que un diseño conversacional proporciona unas buenas características de usabilidad, la interfaz gráfica de este desarrollo está diseñada exclusivamente para mantener una conversación escrita.

La integración de módulos de reconocimiento y sintetizador de voz, proporcionarían unas características de accesibilidad idóneas para casi cualquier usuario.

Por tanto, otra posible mejora para una futura versión podría ser la adaptación de la interfaz web para incluir un diseño tanto escrito como por voz.

4.2.3 Expansión a otros ámbitos

Por otra parte, si el uso de chatbots para realizar diagnósticos presenta buenos resultados, se podría adaptar el uso de chatbots para diagnosticar otro tipo de trastornos cognitivos como el síndrome de Asperger u otros como depresión y ansiedad. Además, se podría ampliar su uso a otras áreas de la medicina, diferentes a la neurología.

Por otra parte, también sería interesante pensar el usar chatbots para tratar enfermedades. Por ejemplo, se ha demostrado que la estimulación cognitiva puede llegar a retrasar los síntomas de trastornos degenerativos como el Alzheimer.

Por tanto, se podrían implementar chatbots para que los pacientes pudiesen realizar estas actividades e incorporar notificaciones para que lo hiciesen de forma periódica.

5 Análisis de Impacto

En los últimos años, la industria de desarrollo de chatbots ha crecido exponencialmente. Esto se debe, en parte, a la gran cantidad de beneficios que herramientas como esta aportan a la sociedad. Mediante el desarrollo de proyectos de este tipo de características, se podría obtener un gran impacto positivo en muchos ámbitos de la sociedad.

5.1 Salud

En primer lugar, puesto que este trabajo está orientado al ámbito de la salud, la implementación de un proyecto como este podría aportar diversas ventajas en este ámbito.

Cada año se diagnostican 4.6 millones de casos nuevos de Alzheimer, por lo que los servicios de neurología de la mayoría de los países se encuentran desbordados. Una herramienta como esta proporciona un nuevo método de diagnóstico de una calidad muy alta a un precio muy razonable.

Además, implementado de una manera óptima podría no necesitar la supervisión directa de un profesional sanitario, lo que se traduciría en la optimización de los recursos de los hospitales para maximizar el número de pacientes tratados.

Por otro lado, puesto un chatbot está en funcionamiento 24 horas al día y es accesible desde casi cualquier sitio, los pacientes podrían realizar estas pruebas periódicamente, para identificar algún trastorno cognitivo, sin tener que acudir a una consulta médica o a un hospital.

Esto proporcionaría muchas ventajas, sobre todo teniendo en cuenta el riesgo que supone para las personas mayores acudir a un hospital en estado de pandemia. Por otro lado, también se podrían realizar diagnósticos de otros trastornos cognitivos diferentes o incluso de enfermedades de otros ámbitos de la medicina.

Además, el uso de los chatbots es gratuito, y solo requiere acceso a internet para poder utilizarlos. Por tanto, podrían ayudar a proporcionar diagnósticos de forma gratuita a una gran parte de la población mundial.

Por lo tanto, Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 se podría cumplir los siguientes objetivos:

- **El objetivo 3.4:** Para 2030, reducir en un tercio la mortalidad prematura por enfermedades no transmisibles mediante la prevención y el tratamiento y promover la salud mental y el bienestar
- **El objetivo 3.d**: "Reforzar la capacidad de todos los países, en particular los países en desarrollo, en materia de alerta temprana, reducción de riesgos y gestión de los riesgos para la salud nacional".
- **El objetivo 3.8**: Lograr la cobertura sanitaria universal, en particular la protección contra riesgos financieros, el acceso a servicios de salud esenciales de calidad y el acceso a medicamentos y vacunas seguros, eficaces, asequibles y de calidad para todos.

Sin embargo, también hay que tener en cuenta que los chatbots en ningún caso podrán superar o sustituir al personal sanitario cualificado, serán simplemente una herramienta más de soporte.

5.2 Crecimiento económico

Por otra parte, el uso de chatbots en organizaciones diversas también podría tener un gran impacto económico. En el caso de un producto como el que propone este proyecto, se podrían optimizar los recursos de los hospitales.

Sin embargo, el uso de chatbots para procesos repetitivos que actualmente se realizan manualmente, podría potencialmente optimizar los recursos de la mayoría de organizaciones.

Además, puesto que es un sector emergente, va a impulsar la creación de una gran cantidad de puestos de empleo destinados al diseño y la implementación de chatbots. Incluso podría llegar a impulsar la creación de nuevas empresas en este sector.

Todos estos factores podrían ayudar a cumplir algunos objetivos de Desarrollo Sostenible como:

- **El objetivo 8.2**: Lograr niveles más elevados de productividad económica mediante la diversificación, modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra.
- **El objetivo 8.3**: Promover políticas orientadas al desarrollo que apoyen las actividades productivas, la creación de puestos de trabajo decentes, el emprendimiento, la creatividad y la innovación, y fomentar la formalización y el crecimiento de las microempresas y las pequeñas y medianas empresas, incluso mediante el acceso a servicios financieros.

Sin embargo, puesto que los chatbots automatizan procesos que actualmente se realizan manualmente, un impacto negativo puede ser la destrucción de algunos puestos de empleo, que pueden ser sustituidos por chatbots. Sin embargo, el desarrollo de nuevas tecnologías, va a suponer la creación de nuevos puestos de empleo más cualificados y sostenibles.

6 Bibliografía

1. CLEUSA P FERRI, MARTIN PRINCE, *Global prevalence of dementia*, https://pubmed.ncbi.nlm.nih.gov/16360788/M
2. M. DAHIYA, *A Tool of Conversation: Chatbot*
3. JUANAN PEREIRA, ÓSCAR DÍAZ, *Using Health Chatbots for Behaviour Change: A Mapping Study*
4. CHRISTINA ATAY, DAVID IRELAND, JACKI LIDDLE, *Can a Smartphone-Based Chatbot Engage Older Community Group Members? The impact of Specialised Content*
5. Klopfenstein, L.C., Delpriori, S., Malatini, S., and Bogliolo, A., *The rise of bots: A survey of conversational interfaces, patterns, and paradigms. Proceedings of the 2017 conference on designing interactive systems.* 555–565. ACM, New York, NY, USA, 2017.
6. Kozinakova, B., *Analysis of chatbot systems focusing on the elderly as users*. Master Thesis, Politecnico de Milano, 2017.
7. Marshal F. Folstein, *The Mini-Mental State Examination*
8. Siqi Liu, Sidong Liu, Weidong Cai, Sonia Pujol, *Early diagnosis of Alzheimer's with deep learning*
9. F. Cuetos-Vega, M. Menéndez-González, T. Calatayud-Noguera, *Descripción de un nuevo test para la detección precoz de la enfermedad de Alzheimer*
10. Robbins T.W., James M., Owen A.M., Sahakian B.J. Cambridge Neuropsychological Test Automated Battery (CANTAB*): A Factor Analytic Study of a Large Sample of Normal Elderly Volunteers*
11. Fengyi Tang, Kaixiang Lin, Ikechukwu Uchendu, Hiroko H. Dodge, Jiayu Zhou. *Improving Mild Cognitive Impairment Prediction via Reinforcement Learning and Dialogue Simulation*
12. *DANIEL CERDAS MENDEZ, Historia de la Inteligencia artificial relacionada con los Chatbots*
13. Alan, M. (1950). Turing. *Computing machinery and intelligence.* https://doi.org/http://dx.doi.org/10.1007/978-1-4020-6710-5_3
14. *ELIZA: a very basic Rogerian Psychotherapist chatbot* https://web.njit.edu/~ronkowit/eliza.html
15. Tomas Zemcik, *A History of Chatbots*
16. DAVID DEL VALLE AGUDO, *Alan Turing y el gran premio Loebner*

17. B. COMENDADOR, BIEN MICHAEL B. FRANCISCO, *Pharmabot: a Pedriatic Generi Medicine Consultant Chatbot*
18. *Telemedicina para combatir la covid-19* https://elpais.com/elpais/2020/06/04/eps/1591275082_845962.html
19. *NHS Chatbot to Provide Scottish Patients Access to Covid-19 Information* https://digit.fyi/nhs-chatbot-to-provide-scottish-patients-access-to-covid-19-information/
20. http://www.soydiego.com/soydiego/
21. https://www.duolingo.com/
22. https://www.theguardian.com/education/historyandhistoryofart
23. Ana Fernández, *Inteligencia artificial en los servicios financieros*
24. *Chatbots en el eCommerce,* https://www.freshcommerce.es/blog/chatbots-ecommerce-ventajas-conseguir-mas-ventas/
25. The Rasa Masterclass Handbook
26. *Computación cognitiva, o cómputo cognitivo*, https://searchdatacenter.techtarget.com/es/definicion/Computacion-cognitiva-o-computo-cognitivo
27. Procesamiento del Lenguaje Natural, https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

7 Anexos

7.1 Anexo 1:

En el siguiente link se puede acceder al repositorio de GitHub en el que está desplegado la totalidad del código desarrollado en este proyecto. El repositorio es de acceso público y la rama principal sobre la que se ha trabajado es “main”.

https://github.com/mariamerino99/chatbot

7.2 Anexo 2:

A través del link proporcionado se puede acceder a la guía de usuario del chatbot, que a su vez contiene un link para poder probar el mismo.

https://drive.google.com/file/d/1fJw7gufeyaNAkW7PyDYIa7tmOe2V9Ive/view?usp=sharing