ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

DESARRROLLO DE CHATBOTS CON ENTORNOS DE CÓDIGO ABIERTO

CHATBOTS DEVELOPMENT WITH OPEN SOURCE FRAMEWORKS

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: José María Gutiérrez Siliceo

Junio - 2019

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: José María Gutiérrez Siliceo
Director del TFG: Pablo Pedro Sanchez Espeso

Título: "**DESARRROLLO DE CHATBOTS CON ENTORNOS DE CÓDIGO ABIERTO**"
Title: "CHATBOTS DEVELOPMENT WITH OPEN SOURCE FRAMEWORKS "

Presentado a examen el día: 12 Julio 2019

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Pablo Pedro Sanchez Espeso
Secretario (Apellidos, Nombre): Pablo Prieto Torralbo
Vocal (Apellidos, Nombre): Jesús Ibáñez Diaz

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecer…

A mi familia por el constante apoyo y preocupación,

A mis amigos por la inspiración y motivación,

A Pablo Pedro Sánchez por creer en el proyecto y hacer posible que salga adelante.

Resumen:

Con cada vez más teléfonos móviles, ordenadores, tabletas y tecnología en general en nuestras manos se hace necesario el desarrollo de herramientas que permitan ayudar al usuario con dichas tecnologías, respondiendo a sus dudas y problemas de forma automática. En esta línea los asistentes virtuales o chatbots son una de las herramientas más utilizadas. Estos asistentes se usan a diario por millones de personas para pedirles que les recuerden algo, comprar una pizza a domicilio, sugerir un restaurante cercano, mantener una conversación intranscendente y otras muchas aplicaciones.

El presente trabajo pretende responder a preguntas básicas sobre estos asistentes tales como ¿son fáciles de desarrollar?, ¿cómo funcionan? y ¿Es posible desarrollar estos sistemas con herramientas no propietarias y de bajo coste?
El proyecto introduce la tecnología de asistentes virtuales y muestra como poder desarrollar un chatbot usando tecnologías de gran actualidad como Machine Learning.

Además del estudio de la tecnología, se ha generado un tutorial que permite a cualquier persona poder crear su chatbot utilizando una plataforma Open Source, que utiliza machine learning para responder al usuario. Por último, se ha desarrollado un chatbot (caso de uso) para una supuesta empresa de e-commerce dedicada a la venta de patinetes online, utilizando la misma tecnología que en el tutorial.

A b s t r a c t :

The number of phones, computers, tablets and technology in general in our hands is in constantly growth. And each day thousands of developers creating new systems, apps, new software and hardware etc. Virtual assistants or chatbots don't are less, using them daily by millions of people, to ask them to remind the user something, to ask them to order pizza, to ask for advice for a near restaurant, or simply have a conversation, and a lot of other things.

Comes the idea of this project, How chatbots work? Are them easy to create? Could I do one without being an expert on the field? The answer is yes, and this project works as an introduction to this world of chatbots, shows a tutorial on how to create a chatbot using one of the strongest technologies of the time: Machine Learning.

A tutorial has been done, that teaches on how to do a chatbot using an Open Source framework that uses Machine Learning to answer the user. And also, it has been developed a chatbot for an e-commerce company that sells skateboards online, by using the same technology of the tutorial.

ÍNDICE.

ÍNDICE DE FIGURAS.

1. INTRODUCCIÓN.

1.1 Motivación.

Viviendo en una época en la que la tecnología evoluciona rápidamente y es ubicua (está por todos lados), los humanos tenemos la necesidad de interactuar con los sistemas de dicha tecnología de una forma sencilla e intuitiva. Para facilitar dicha interacción se han desarrollado sistemas que permiten al usuario mantener una conversación con un programa informático de forma natural. Estos sistemas, o chatbots, serán el objeto de estudio de este proyecto. Como casi todas las personas tienen un teléfono móvil, o mejor dicho un Smart phone, y aunque no todo el mundo lo use, casi todos disponemos en dicho dispositivo de al menos un chatbot. Siri, en el caso de los productos Apple, y Google assistant, en los productos con sistemas Android, son ejemplos típicos de chatbot de uso corriente en dispositivos móviles.

Otra actividad de fuerte crecimiento en la actualidad es el e-commerce, los negocios online. Cada vez más gente decide comprar por internet, y la verdad es que en internet se vende prácticamente todo lo que uno se pueda imaginar. Además, es muy común entrar a la página web de una compañía o comercio online, y que salte en una esquinita de la pantalla, un pequeño robot (bot) que se presenta y nos pregunta cómo nos puede ayudar, comunicándose (chat bot) de la misma forma que lo haría un asistente humano. Dicho robot ha sido integrado en la página para mejorar la experiencia del usuario y facilitarle la búsqueda del producto o servicio que necesita. Gracias al chatbot, el usuario se dirige rápidamente al área del sitio web que almacena la información que busca, reduciendo el tiempo de búsqueda y facilitando las ventas.

Por todo ello es frecuente escuchar en blogs de tecnología que “en unos años el 90% de las transacciones serán realizadas por chatbots”. Dado su prometedor fututo la primera idea de este proyecto fue crear un pequeño negocio de chatbots, con el objetivo de desarrollar estos sistemas e implementarlos en una página web que permitiese ayudar con preguntas frecuentes o realizar alguna acción.

De estos primeros trabajos surge el objetivo general de este proyecto, desarrollar una introducción al mundo de los chatbots para que cualquier persona interesada en el tema pueda desarrollar rápidamente una aplicación. El proyecto estudia las tecnologías utilizadas en chatbots, sus características generales y sus principales aplicaciones. Además incluye un tutorial de cómo realizar un bot desde cero utilizando “machine learning”, así como un ejemplo de creación de un bot real para una empresa ficticia.

1.2 Objetivos.

En este trabajo se explora el mundo de los chatbots. Se parte del estudio de sus conceptos básicos, y se finaliza con la realización de un chatbot real, como si un cliente hubiera solicitado el desarrollo de un asistente para su tienda online.

En primer lugar se estudiará de una manera breve pero completa, las características, arquitectura, plataformas de desarrollo y aplicaciones de los chatbot. Además, se estudiarán algunos de los chatbots comerciales más usados en distintos campos de trabajo. Un aspecto importante en los chatbots es la computación cognitiva, es decir la manera de entender los mensajes que el usuario humano envía al chatbot. En este área, las técnicas de procesamiento de lenguaje natural (NLP) juegan un papel esencial.

En segundo lugar se seleccionará una plataforma concreta con una arquitectura específica para realizar un asistente, y se estudiará a fondo dicha tecnología. Como resultado de dicho trabajo se ha desarrollado un tutorial completo de cómo realizar un chatbot con dicha plataforma. El tutorial incluye muchos detalles, con ejemplos de un bot real.

Y para finalizar el proyecto se desarrollará un chatbot real, asumiendo que somos una empresa de chatbots y se nos ha pedido un bot para un cliente que tiene una tienda online. El bot será un asistente virtual que ayuda al usuario a saber que comprar dentro de la tienda. De este modo ahorrará tiempo, al no tener que buscar en internet la información que necesita para seleccionar el producto que desea, al tiempo que mejora un opinión sobre la tienda. El bot se ha realizado usando la misma plataforma/tecnología que en el tutorial. Dicha tecnología utilizará técnicas de "machine learning".

1.3 Estructura.

Se ha dividido el proyecto en 4 capítulos.

En el primer capítulo se presenta un estudio general sobre los asistentes virtuales, analizando su contexto histórico, características, aplicaciones, plataformas, arquitecturas y uso de la computación cognitiva. También se comentan los chatbots más usados.

El segundo capítulo se centra en la tutoría de chatbots, explicando paso a paso una tecnología "open source" que utiliza "machine learning" para la creación de chatbots: Rasa. El tutorial comienza con la presentación teórica de la arquitectura del charbot y a continuación describe, paso a paso, desde la instalación del entorno hasta el desarrollo del código del bot.

En el siguiente capítulo se desarrolla un chatbot para una supuesta compañía de e-commerce, utilizando la tecnología Rasa explicada en el tutorial. Se incluyen ejemplos del proceso de desarrollo y la versión final.

En el cuarto capítulo se desarrolla un chatbot con el mismo objetivo que el anterior pero creado en una platafroma diferente. Esta es una plataforma online (hosted), donde todo se desarrolla en un interfaz web. Dicha plataforma tiene la gran diferencia de que no hace falta saber nada de programación para su uso.

2. ESTADO DEL ARTE.

Los chatbots son programas software que utilizando algoritmos, reglas y procesamiento de lenguaje natural (NLP: *Natural Language Processing*) tienen como objetivo simular una conversación con un humano. El reto de la tecnología es que el asistente virtual parezca un humano. La interacción con el usuario puede ser mediante mensajes de texto o voz.

2.1 Comienzos.

En 1950, el matemático británico Alan Turing propuso en una publicación científica una forma de responder al interrogante: "¿Pueden pensar las maquinas?". Su método se basa en la implementación de un juego, al que llamó “The Imitation Game”. En dicho ejercicio participan dos personas y una máquina. Una de las personas actúa como juez (o jurado) y se comunica con la máquina y con la otra persona por escrito. El objetivo del juego es que el jurado identifique quién es la máquina y quién la persona. Por supuesto que tanto la otra persona como la maquina no están en la misma habitación que el jurado, y este solo puede comunicarse con ambas mediante preguntas escritas, las cuales son respondidas en igual forma. Turing sostuvo que, si la maquina podía engañar al jurado entonces se podría considerar que dicha maquina piensa. A esta prueba se la llama el Test de Turing. Al día de hoy no ha sido creado un sistema capaz de pasar de forma satisfactoria dicha prueba.

En 1966, Joseph Weinzebaum creó en el MIT el primer chatbot: ELIZA. Dicho sistema estaba más cerca de parecer humano que trabajos anteriores. El algoritmo de ELIZA buscaba palabras clave en las frases del usuario y respondía de manera parecida a un psicólogo, reformulando la frase del usuario. El sistema sirvió de modelo para otros chatbots, que también utilizaban palabras clave y patrones. Entre ellos destacó PARRY, que fue creado por Kenneth Colby en 1972 para impersonar paranoia y esquizofrenia. Dicho sistema sirvió para evaluar conversaciones con pacientes que sufrían paranoia.

En 1990 se creó el Premio Loebner de Inteligencia Artificial, para destacar trabajos que alcanzasen la primera instancia de un *test de Turing.* Por ello se ofreció un premio de 100.000 dólares para el primer chatbot cuyas respuestas fueran indistinguibles de un ser humano. Los primeros años los premios fueron ganados por chatbots que implementaban el funcionamiento básico de ELIZA. Actualmente el premio se sigue otorgando anualmente y destaca los mejores desarrollos de chatbots.

Pocos años después, en 1995, Richard Wallace creo A.L.I.C.E, un chatbot más completo que generaba respuestas mas complejas teniendo en cuenta patrones de entrada y plantillas en bases de datos. Una innovación y diferencia respecto a chatbots anteriores es que está escrito en AIML (Artificial Intelligence Markup Language), una extensión de XML que todavía hoy está en uso. A.L.I.C.E fue vencedor del Premio Loebner en tres ocasiones.

Hoy en día casi todo el mundo conoce chatbots modernos, como Siri de Apple, Alexa de Amazon o Cortana de Microsoft. Estos chatbots sirven como asistentes generales y tienen funcionalidades variadas.

2.2 Características de los chatbots.

Un chatbot es una entidad artificial que tiene el objetivo de mantener conversaciones con seres humanos. En la actualidad existen muchas variedades de chatbots, con distintas funcionalidades, objetivos, complejidades, arquitecturas y prestaciones. Se pueden encontrar desde pequeños juegos o pequeños chatbots de FAQ (Frequently Asked Questions) hasta los sistemas más complejos, como Siri o Alexa.

Las características que diferencian a los chatbots son las siguientes:

- Racionalidad: Si el asistente virutal realiza lo correcto a partir de los datos que recibe del entorno más la frase introducida, y genera la respuesta más apropiada se dice que es racional.
- Autonomía: El chatbot es autónomo si es capaz de adaptarse a cambios, pudiendo tener varios estilos de respuesta dependiendo del usuario, la pregunta realizada, el momento, etc.
- Reactividad: Si el chatbot percibe el entorno y sus cambios afectan en su comportamiento se dice que es reactivo. No se limita solo a dar frases como respuestas, puede proporcionar imágenes, enlaces o distintos recursos como respuesta.
- Pro-actividad: Si el chatbot entiende los cambios en el entorno y depende en la situación que se encuentre es capaz de realizar el mismo las preguntas, o reactivar la conversación cuando parece que ha finalizado.
- Veracidad:Si el chatbot no da información falsa, de dice que tiene la característica de veraz.
- Personalidad: Este aspecto depende del programador y de las características que el mismo quiera darle, como por ejemplo su comportamiento verbal

2.3 Aplicaciones.

Actualmente los chatbots se utilizan en multitud de aplicaciones y entornos. Al principio los sistemas solo trataban de pasar el test de Turing pero pronto pasaron a ser asistentes virtuales, guías, profesores etc. A modo de ejemplo se muestran algunas aplicaciones típicas en distintos campos.

- Académico: Educación, Idiomas, Psicología, Historia.
- Asistentes Virtuales: Atención al Cliente, FAQ (Frequently Asked Questions).
- Empresariales: Estudio de Mercado, Ventas, Marketing, Campañas, Turismo.
- Diversión/Lúdico: Entretenimiento, Redes Sociales, Juegos.

Como ejemplo de aplicación famosa, en el ámbito del aprendizaje de idiomas destaca el chatbot duolingo.

2.4 Arquitectura de los chatbots.

Todo asistente necesita una base de conocimiento, esta se introduce en forma de plantillas, patrones, reglas y datos.

El otro gran pilar es el motor de interferencia, que analiza el mensaje de entrada del usuario, mediante un interprete y controlador, que obtienen y clasifican la intención del mensaje para seleccionar una respuesta sacada de la base de conocimiento.

Con la unión de estos dos bloques se crea la interfaz, donde el usuario introduce un mensaje y el chatbot envia una respuesta de vuelta. A continuación, se muestra en la figura una respresentación de la arquitectura del chatbot.

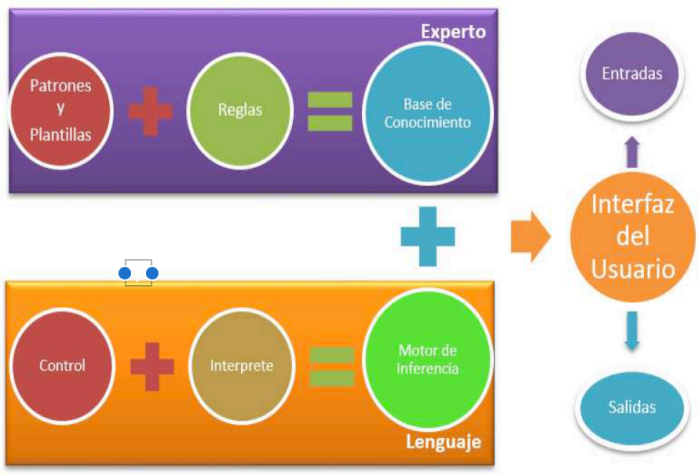

Figura 1 Gráfico de la arquitectura de un chatbot..

Otra representación más básica, menos estructurada es la siguiente: El mensaje del usuario entra en un cerebro, que, con el uso de patrones, plantillas, inteligencia artificial, reglas, predicciones, etc. Saca el mensaje de salida. El gráfico es un pequeño chat.

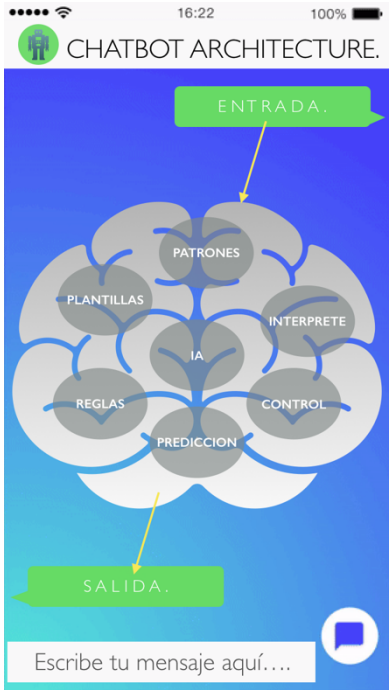

Figura 2 Gráfico de la arquitectura básica de un chatbot.

2.5 Evaluacion de los chatbots.

Desde el punto de vista del usuario, el objetivo del bot es responder a las preguntas del mismo y ser su asistente virtual. Por eso los sistemas se clasifican en base a la usabilidad y satisfacción del usuario con su uso. También expertos en lingüística pueden evaluar la capacidad del chatbot para crear frases correctas gramaticalmente y con sentido. A continuación, se presentan varios métodos de evaluación de uso frecuente.

TEST DE TURING.

Como se ha comentado anteriormente, durante un tiempo, la única manera de evaluar un chatbot era el *Test de Turing*. En dicho test un juez tendrá una conversación con un humano y un ordenador y deberá distinguir cuál es la persona y cuál es el ordenador. En caso que sean indistinguibles para él, la prueba será satisfactoria para la máquina y se puede decir que es inteligente.

No todo el mundo acepta el *test de Turing*. En 1980, John Searle propuso el experimento de la habitación china, donde argumentaba que el experimento de Turing no podía usarse para determinar si una maquina es inteligente o no. *"Imagine un hablante inglés nativo que no conoce el chino encerrado en una habitación llena de cajas de símbolos chinos (una base de datos), junto con un libro de instrucciones para la manipulación de los símbolos (el programa). Imagina que la gente fuera de la sala envía otros símbolos chinos que, sin que la persona en la habitación, son preguntas en chino (la entrada). Además, imagine que, siguiendo las instrucciones del programa del hombre en la habitación es capaz de pasar a los símbolos chinos que son respuestas correctas a las preguntas (de la salida). El programa permite a la persona en el espacio para pasar el Test de Turing para la comprensión de chino, pero él no entiende una palabra de chino." (Cole, 2004)*

El problema es que una maquina como ELIZA puede aprobar la prueba de Turing a través de frases que no había entendido. A los chatbots, "Sin la comprensión no se les puede clasificar realmente como "pensantes" de la misma manera que los humanos". Por lo tanto, Searle concluyó que la prueba de Turing no puede probar que una máquina puede pensar. La propuesta de Searle, como la de Turing, fue tanto respaldada como criticada por muchos. Por lo que no existe un criterio unánime.

PARADISE FRAMEWORK.

Uno de las herramientas más usadas para evaluar los chatbots es PARADISE (PARAdigm for Dialogue System Evaluation). Esta metodología evalúa factores subjetivos como la sencillez al uso, claridad, naturalidad, entorno amigable, comportamiento cuando no entiende correctamente y ganas de usar el sistema de nuevo. Para ello se realizan cuestionarios a los usuarios para que puntúen al bot. Además, PARADISE, intenta cuantificar la efectividad del bot maximizando las tareas realizadas correctamente y minimizando el dialogo.

Maximizar la efectividad de las tareas: Para maximizar la efectividad, el bot debe dar resultados correctos. Los creadores de PARADISE introdujeron el concepto de attribute value matrix (AVM) para medir la efectividad. Proponen que humanos sigan un guion que espera determinados resultados. Estas respuestas correctas se mapean a una llave y los resultados del bot a la matriz AVM. De estos dos se crea la matriz M, que contiene el número de veces que el bot consiguió lo correcto frente a la respuesta incorrecta. Para medir el acierto de las tareas, se creó el coeficiente kappa, basado en la matriz M.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

P(A) es proporción de tiempo donde AVM concuerda con el resultado correcto del escenario, y P(E) es la probabilidad de concordar por casualidad. Un chatbot respondiendo aleatoriamente tendría k=0 donde un humano tendría k=1.

Minimizar el dialogo: Se definen dos tipos de coste de dialogo: costes de calidad, y costes de eficiencia. Los costes de calidad miden el número de re-prompts (veces que imprime por pantalla el mensaje), número de respuestas inapropiadas, precisión de concepto y ratio de vuelta a lo correcto. Los costes de eficiencia miden incluyen: tiempo esperando, numero de ciclos del sistema, numero de ciclos del sistema por tarea y tiempo total esperado por ciclo.

OTROS MÉTODOS DE EVALUACION

Existen muchos otros métodos para medir el trabajo y satisfacción de los usuarios al usar el bot. Por ejemplo, Kuligoswka mide el aspecto visual, ya que cree que si los bots parecen más humanos, incrementa la interacción del humano con el chatbot.

Otras evaluaciones de Kuligowska son la implementación del bot en la página web, la habilidad del bot comunicarse por voz y si tiene diferentes tipos de voz, el conocimiento del bot, la personalización, etc.

2.6 Computación cognitiva.

¿Como nos entienden los chatbots? Es esta una de las primeras preguntas que se hace cualquiera al mantener una pequeña conversación con un chatbot. Según la RAE, la definición de *cognitiva es* que pertenece o es relativo al conocimiento y está relacionado con el proceso de adquisición de conocimiento (cognición) mediante la información recibida de el ambiente y aprendizaje. Se define la computación cognitiva como los sistemas que aprenden a escala, razonan con un propósito e interactúan con humanos de manera natural. Por lo tanto, significa que es informática basada en computadores capaces de sentir adaptarse y aprender tratando de simular el comportamiento humano.

En 2016, en Bhilegaonkar se proponen seis categorías de Tecnologías de computación cognitiva:

- Análisis Predictivo: Sistemas inteligentes que tratan mayoritariamente con datos estructurados. Estos suelen ser en forma de números, por ejemplo, transacciones financieras o datos de ventas. Se utilizan técnicas de análisis matemático y estadístico.
- Reconocimiento de imágenes: Las entradas de imágenes y el procesado de ellas, pasando a forma de números para después de analizar tomar una decisión, lleva por detrás muchos algoritmos y procesos. Con Computer visión (Vision por computadora) se puede conseguir detectar objetos, reconcomiendo de objetos, sacar contornos de formas, procesamiento a imágenes 3D, etc.
- Reglas de Negocio: La lógica de negocio está estructurada usando reglas que expresan construcciones simples tales como declaraciones de si/entonces. Los sistemas expertos fueron programados usando este tipo de reglas para emular la capacidad de tomar decisiones de un experto humano. Por ejemplo, si es una solicitud de crédito si es aceptada o rechazada dependerá de un umbral conocido por la computadora.
- Inteligencia Artificial: Aprendizaje automático, es un concepto de lo más avanzado y actual. Implican la creación de un modelo a partir de datos o modelos. Las formas y enfoques incluyen redes neuronales, redes neuronales profundas, clasificadores bayesianos etc.
- Procesamiento de Lenguaje Natural (NLP): Natural Language Processor se trata del reconocimiento de voz, análisis del texto y derivar el significado del mensaje. Separan el texto en frases, palabras, sujeto, predicado, verbo etc para entender el mensaje.
- Procesamiento Complejo de Eventos (CEP): Los CEP tratan datos en tiempo real de un conjunto diverso de fuentes y tipos.

2.7 Plataformas de desarrollo.

AIML.

El lenguaje de programación AIML(Artificial Intelligence Markup Language), mencionado en la creación del bot ALICE es una adaptación de una gramática XML. Este lenguaje describe una clase de objetos llamados “AIML Objects” y estos describen parcialmente el comportamiento de los programas de ordenador que los procesan. Estos objetos se componen de unidades llamadas temas y categorías, que contienen datos analizados o sin analizar.

Es un lenguaje de scripts que define una base de conocimiento de pregunta respuesta, que se utiliza como software para chatbots de texto. En este lenguaje una interacción entre el bot y el usuario se define en la unidad (category), y las posibles expresiones del usuario se definen en patrones (patterns) y la respuesta del bot se definen en plantillas (templates).

IBM WATSON.

Watson Conversation es una plataforma donde crear una aplicación que comprende la entrada de lenguaje natural y utiliza el aprendizaje automático para responder a los usuarios de manera que simula una conversación entre dos humanos.

Los usuarios interactúan con el asistente a través de la interfaz elegida por el usuario, como puede ser telegram, un servidor local o cualquier aplicación de terceros compatible. Luego la aplicación envia la entrada del usaurio al servicio de Watson, y esta se conecta a un espacio de trabajo con el contenedor para la gestión del dialogo y los datos del chatbot.

La aplicación proporciona un entorno gráfico fácil de usar para crear flujos de conversación. La creación de la primera conversación usando IBM Watson implica los siguientes pasos: Entrenar a Watson para que entienda la entrada de las expresiones. Se le entrenara con ejemplos de flujos. Identificar los términos que pueden variar en las entradas, y crear respuestas a las preguntas del usuario.

DIALOGFLOW.

Es una plataforma para que desarrolladores y no desarrolladores diseñen e integren interfaces de usuario conversacionales inteligentes en aplicaciones web y móvil mediante la compresión del lenguaje natural que ofrece la plataforma. Su objetivo es que su proceso de creación o funcionalidad de la plataforma sea lo más simple posible.

El procesamiento de lenguaje natural y Machine Learnign de Dialogflow son los grandes diferenciadores, ya que se basan en el volumen de diaologos procesados. Dialogflow procesa millones de solicitudes de diálogos al día, y no necesita proporcionar muchos ejemplos inicialmente ya que aprovecha los datos existentes. Ya que es de google y utiliza la cloud y la tecnología Machine Learning de google.

Figura 3 Logo Dialogflow de Google.

RASA STACK.

Rasa es un entorno de código abierto (Open Source) para construir asistentes virtuales o chatbots con inteligencia artificial. Rasa tiene dos grandes módulos, Rasa NLU y Rasa Core. El módulo NLU se utiliza para entender los mensajes del usuario y el módulo Core se emplea para archivar las conversaciones y saber que función realizar después del usuario. Recientemente se ha presentado Rasa X, que es una herramienta para ayudar a mejorar el bot y mejora la experiencia ya que tiene una interfaz simple y sencilla de usar.

Figura 4 Logo Rasa.

FLOW XO:

Con Flow Xo, se puede crear un chabot sin conocimiento alguno de código. Todo se hace desde su página web ya que tienen una interfaz muy intuitiva. Usando un editor de arrastrar y soltar para construir flows de dialogo es rápido y sencillo crear un chatbot. Se puede integrar con aplicaciones externas como Telegram, Facebook Messenger etc. Además es muy adaptado y estable. Por último comentar que el entorno es gratis en primera instancia, pero una vez el negocio o el uso del chatbot crezce, comienza a ser un servicio de pago.

Figura 5 Logo Flow XO.

2.8 Procesamiento del Lenguaje Natural.

El objetivo del procesamiento de lenguaje natural es coger la entrada, ya sea texto o voz, y producir una representación estructurada del lenguaje natural. Vamos a analizar varias maneras de extraer información semántica y su significado, con el fin de crear estructuras gramáticas que puedan ser procesadas.

Reconocimiento de Dialogo.

Una manera de extraer significado del lenguaje es determinar la función del texto/frase (por ejemplo, si es una pregunta, una afirmación, una propuesta etc). Esto es llamado Dialogue Act Recognition (Reconocimiento de la función del dialogo). Estos sistemas etiquetan las frases con el significado de su función. A continuación, se muestran algunos ejemplos.

SPEAKER	FUNCIÓN DEL MENSAJE	MENSAJE
A	INTRO	HOLA!?
B	INTRO	HOLA HANNAH!
B	AFIRMACIÓN	SOY JOSE
B	PREGUNTA	¿QUÉ TAL?
A	INTRO	HOLA JOSE!
A	AFIRMACIÓN	MUY BIEN
A	PREGUNTA	¿Y TU?
B	AFIRMACIÓN	BIEN TAMBIEN.

Figura 6 Tabla clasificando actos de dialogo.

Identificación de Intención.

Las intenciones del dialogo suelen tener una función específica, es decir no quieres saber si la frase es una pregunta o una respuesta, sino por ejemplo si es buscar un restaurante o cancelar una reserva en un programa de buscar restaurantes.

Suele usarse por bots que empiezan con ¿en que puede ayudarte?, y utilizan la identificación de intención para redirigir al usuario a alguna de las predefinidas opciones de redirección.

Extracción de información.

La principal función del NLU no es solo entender la función de la frase, sino entender el significado de la misma. Para extraer significado del texto, se transforma o analiza gramaticalmente el mismo.

Para ello el primer paso es dividir el texto en tokens, que son las palabras, puntos, números que componenen la frase. La tokenizacion es difícil ya que siempre hay palabras compuestas (ej Nueva YorK), o abreviaturas etc, que deberían ser tokenizadas como un simple token y no más. Estos tokens se pueden analizar con distintas técnicas, que se explican a continuación:

Bolsa de Palabras: Se ignora la estructura de la frase, la sintaxis, el orden de las palabras y únicamente se cuenta el número de veces que se repite cada palabra. Se separan los elementos de unión, como los artículos “el” y “la”. A continuación las conjugaciones de verbos o similares se agrupan en lemas. A este proceso se le denomina lematización. A continuación, se comparan con la base de datos del bot y se buscan similitudes. Esta estrategia facilita el análisis, ya que no requiere conocimiento sobre la sintaxis, pero al mismo tiempo puede no ser suficientemente precisa para entender los problemas más complicados.

Expresiones regulares: Las frases pueden ser iguales que una plantilla guardada en el la base de datos del bot. Por ejemplo “my nombre es”, es una frase común que el bot sabe que, si aparece esa frase, tiene que responder una respuesta predeterminada.

Reconocimiento de Entidades: Diferencia las palabras en nombres de personas, lugares, grupos, localizaciones, fechas etc. Esto sirve para saber que función debe realizar el bot, por ejemplo, para realizar una reserva, si el bot obtiene la entidad de localización sabrá donde tiene que reservar.

Análisis sintáctico y gramático: Las frases se tratan como estructuras de datos que representan verbos, nombres, sujetos, predicados, pronombres, etc. Las palabras de la frase se separan y vectorizan (se introducen en matrices) y, mediante algoritmos de machine learning, se pueden obtener predicciones sobre el significado de la frase.

Esta sección ha presentado algunas de las técnias que se utilizan para el procesamiento de lenguaje natural. Ademas se han presentado los avances más nuevos, como Deep learning.

2.9 Chatbots comerciales.

Los asistentes virtuales son de uso común, aunque algunos todavía no se ha dado cuenta de su existencia. A continuación se presentan y evaluan con distintas aplicaciones algunos de los chatbots más populares :

Asistentes virtuales personales:

- Apple Siri: Es un asistente virtual específico para productos de Apple. El sistema tiene acceso a las aplicaciones de Apple, como Mail, Contactos, Maps, Mensajes, Safari, etc. El asistente es capaz de leer correos, mensajes, poner música, buscar restaurantes etc.
- Microsoft Cortana: Cortana es un asistente de Windows, y puede buscar el tiempo, en el calendario, leer el correo Outlook o ayudar al usuario a solucionar sus ducas y preguntas.
- Google Assistant: Es una extensión de la primera versión que apareció para teléfonos google, como “OK Google”. Con esta aplicación, se puede controlar el móvil por comandos de voz, por ejemplo.
- Amazon Alexa: Alexa puede accder al tiempo, televisión, radio, música y varias aplicaciones de terceros, que es lo que la hace interesante, como Uber, JustEat, Spotify etc. Cada vez se está usando más y sobre en los altavoces portátiles de Amazon.

Bots de aplicación especifica:

- Bots de transporte: Instalocate conoce y puede informar sobre el tiempo real en el recorrido de los aviones, saber si hay retrasos etc.

- Citas: No están muy avanzados. Estos sistemas intentan parecerse a tinder pero uno en concreto se llama Foxsy.
- Meditación: MeditateBot ayuda al usuario y le enseña ejercicios de respiración y de posiciones tipo Yoga.
- Bots del tiempo: Poncho, permite pedir el tiempo y se conecta a Weather Channel y lo da. No están muy avanzados ya que las aplicaciones del tiempo ya son bastantes buenas como para preferir un bot.

3. RASA:TUTORIAL.

Es difícil que un asistente virtual o chatbot, entienda exactamente lo que se quiere decir, ya que hay muchas maneras distintas preguntar lo mismo. Sin embargo el reto es que los chatbots se adapten a cualquier circustancia y que no tengan una única estrategia.

Rasa es una compañía que trabaja en inteligencia artificial conversacional. Su producto es Rasa Stack, la combinación de dos Open Source Libraries: Rasa NLU and Rasa Core. Rasa NLU tiene como objetivo entender el lenguaje natural (Natural Language Understanding), que servirá para que el bot entienda que es lo que el usuario dice. Y Rasa Core es un manager de diálogos. Por lo tanto, cuando el bot sabe que ha dicho el usuario gracias a NLU, Rasa Core se encarga de saber que responder o cómo actuar, gracias a técnicas de Machine Learning, que pueden determinar qué acción realizar. De nuevo ambas aplicaciones están basada en Machine Learning, por lo que el sistema aprende de las respuestas que proporciona el usuaro, no siendo un bot clásico que sigue un camino especifico o siempre igual.

¿Por qué escoger Rasa? Algunas de las ventajas que nos ofrece Rasa respecto a otras plataformas para construir Chatbots son:

- Funciona localmente en la máquina del usuario, y se puede usar donde se precise. Las otras plataformas son “Hosted”, corren en los servidores del proveedor.

- Los datos son del usuario, ya que no se envían los datos a ninguna empresa. Por lo que no pueden usar tus datos o copiar tus ideas.

- Y es Hackable: el usuario puede personalizarlo como quiera, crear programas por encima de rasa, añadirle modificaciones etc.

El diagrama adjunto muestra los pasos básicos que el Chatbot sigue para responder a un mensaje.

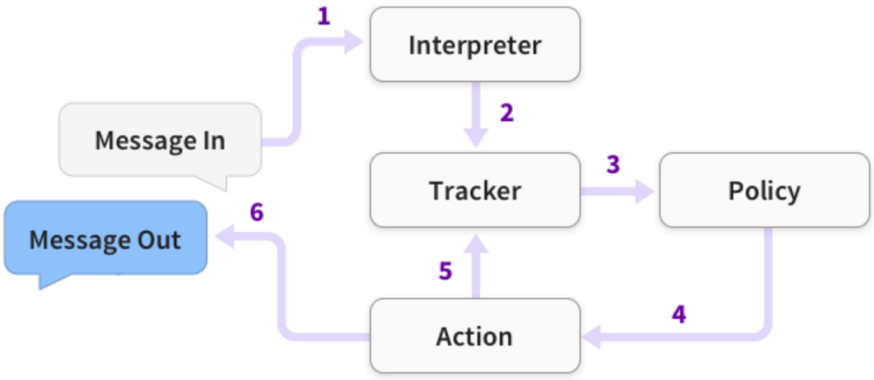

Figura 7 Diagrama de la arquitectura Rasa.

1. El mensaje de entrada del usuario es recibido y enviado al “Interpreter”, el cual clasifica la frase y extrae las intenciones y las entidades obtenidas en el mensaje. Esto lo maneja el NLU.
2. EL “Tracker” es el objeto que lleva a la conversación. Recibe información de que un mensaje nuevo ha entrado.
3. La “Policy” recibe el estado del tracker.
4. La “Policy” decide que acción realizar.
5. La acción es recibida y almacenada por el tracker.
6. La respuesta/acción se envía al usuario.

A continuación comienza el tutorial que responde a la pregunta: ¿Cómo construir y entender un chatbot que utiliza machine learning?

Se va a dividir este tutorial en las tres partes que forman el paquete Rasa con una descripccion teorica y práctica:

- Natural Language Understanding
- Gestionador de diálogos, Rasa Core.
- Rasa X.

3.1 RASA NLU: NATURAL LANGUAGE UNDERSTANDING.

Como ya se ha dicho, RASA NLU es un Open Source Framework cuyo objetivo es la comprensión del lenguaje natural. El sistema captura cualquier tipo de entrada de texto y lo transforma en texto estructurado. Para entenderlo, dentro de NLU los datos estructurados inluyen dos aspectos: Intenciones (intent), que es lo que el usuario quiere decir o está preguntando, y entidades (entities) que son las partes de información del texto que ayudaran al chatbot a dar respuestas más específicas. Por ejemplo:

Usuario: ¿Algún restaurante en el Sardinero?

Intención: Buscar restaurante
Entidad: Sardinero: Dirección.

El primer objetivo es extraer el texto de la entrada del usuario, identificar la intención y las entidades si las hay.

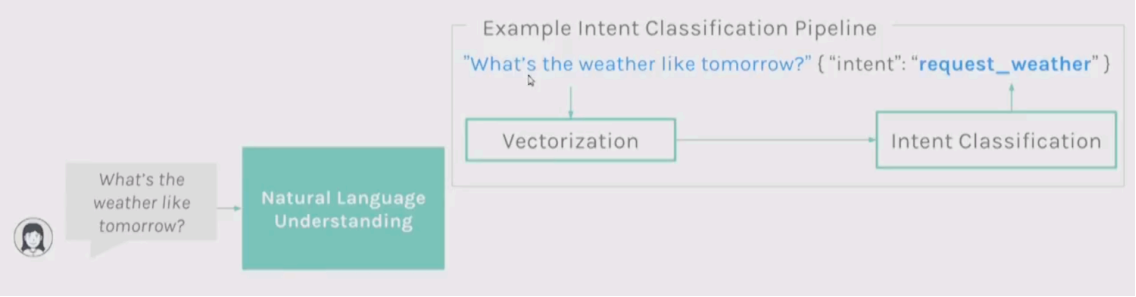

Figura 8 Esquema del funcionamiento de NLU.

Se entiende de la imagen superior que, del texto de entrada del usuario, lo primero que hace NLU es vectorizar el mismo mediante tokens. De esta manera cada palabra será un token. A continuiación se comparan los tokens con los datos que ya tiene el bot y se identifica la intención. ¿Cómo funciona esta vectorización? Una vez se tienen los tokens, se hace una media de estos vectores para cada palabra, de manera que se obtenga una representación de la frase que se usará para entrenar el bot. Como algoritmo de aprendizaje se usa la técnica SVM (Support Vector Machines). Algo a tener en cuenta es que NLU intentara asignar una intención a la frase, pero deberá escoger de las que ya tiene en su base de datos, no creara una nueva.

Para extraer las entidades: Observando la figura 9 primero se pasan a tokens las palabras de la frase. A continuación, se usa "Part of Speech Tagger" un proceso que analiza la frase gramáticamente, y asigna a cada palabra una etiqueta, si es un verbo, sustantivo, adjetivo etc. Se pasa al Chunker, que sirve para unir tokens que estaban separados pero son palabras compuestas o entidades como una dirección. Luego "Named Entity Recognition" y "Entity Extraction" son la función de extraer la entidad, en este caso es tomorrow : date.

Para extraer estas entidades existen dos técnicas que utiliza Rasa NLU.

- Un clasificador binario, donde primero analiza si hay alguna entidad en la frase, y después la clasifica.
- "Direct structured prediction" que se basa en analizar con "conditional random fields" campos aleatorios condicionales que son un modelado usado en machine learning para reconocer patrones y estructuras. En este caso observa las palabras de alrededor de la posible entidad (las palabras contiguas), y predice si es una entidad o no y la extrae. Es la manera que mejor funciona y la que más se usa. No es la más complicada, pero realiza la función.

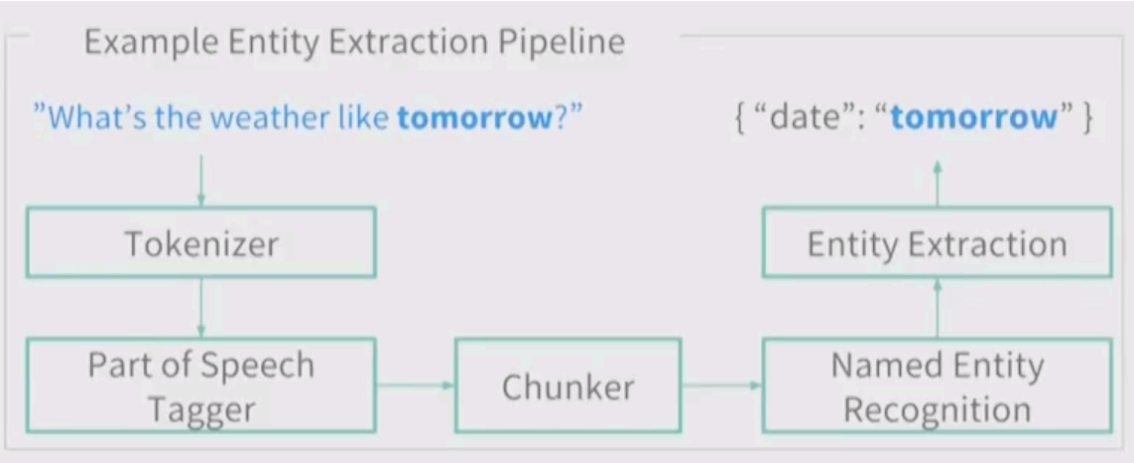

Figura 9 Esquema explicando la extracción de entidades.

3.1.1 INSTALACIÓN.

RASA está en constante desarrollo y cuenta con más de 300 desarrolladores colaborando en el proyecto. Por esta razón, la información en GitHub y la propia página web de RASA cambian regularmente. Por todo ello, es posible que el método de instalación descrito en esta sección haya cambiado cuando se esté leyendo este documento.

El procedimiento de instalación utiliza la herramienta de Python "pip" para instalar paquetes. La versión más reciente instala Rasa NLU y Rasa Core a la vez. Introduciendo en terminal el siguiente comando.

```
pip install rasa-x --extra-index-url https:/pypi.rasa.com/simple
```

Si se quiere instalar la versión en desarrollo, se introducen los siguientes comandos en la terminal.

```
git clone https://github.com/RasaHQ/rasa.git
cd rasa
pip install -r requirements.txt
pip install -e .
```

Una de las librerías importantes que instala rasa es Spacy, una librería open source para el procesamiento de lenguaje natural.

Rasa NLU tiene varios componentes para reconocer intenciones y entidades que introducen dependencias adicionales. Escoger el Pipeline adecuado te permitirá adaptar más o menos tu chatbot o que el sistema funcione mejor.

La respuesta a esta decisión es entender cuál es el caso. Los dos Pipelines más importantes y usados son "*supervised_embeddings"* y *"pretrained_embeddings_spacy".* La mayor diferencia entre ellos dos es que el *"pretrained_embeddings_spacy"* usa palabras-vectores pre entrenadas de GloVe o fastText. Al contrario, el otro Pipeline no usa ninguna palabra pre entrenada, pero se ajusta específicamente a tu conjunto de datos.

La ventaja de usar *"pretrained_embeddings_spacy"* es que, si tienes algún ejemplo de entrenamiento como "Quiero comprar manzanas", y Rasa debería predecir la intención: "querer_peras", el modelo ya sabe que las palabras peras y manzanas son muy similares ya que ya conoce esas palabras al usar este pipeline. Esto lo hace muy útil, especialmente si se tienen pocos datos.

La ventaja de usar *"supervised_embeddings*", es que las palabras-vectores de tu modelo serán específicas y modificadas para tus datos. Por ejemplo, para palabras distintas pero que significan cosas parecidas, o al mezclar dos idiomas que se hace mucho en España y cada vez más, por ejemplo, "cash" y "efectivo", al no usar nada especifico tu pipeline se adaptará, funcionará y entenderá lo que quieras. Se recomienda usar cuando hay muchos

datos. Esto es debido a que se introducen todos los datos por parte del creador y nada esta ya pre guardado.

En Rasa NLU, los mensajes se van procesando en una secuencia de componentes. Estos se van ejecutando uno tras otro. Hay componentes como se vio antes en la teoría de NLU para extraer la entidad, clasificar la entidad y más. Cada componente procesa la entrada y crea una salida que la podrán usar los componentes sucesivos. Hay algunos componentes que solo producen información para que otros componentes la usen y otros que generan ya atributos de salida. Se muestra en la siguiente captura la salida a la frase "I am looking for Chinese food".


```
{
    "text": "I am looking for Chinese food",
    "entities": [
        {"start": 8, "end": 15, "value": "chinese", "entity": "cuisine", "extractor": "CRFEntityExtrac
    ],
    "intent": {"confidence": 0.6485910906220309, "name": "restaurant_search"},
    "intent_ranking": [
        {"confidence": 0.6485910906220309, "name": "restaurant_search"},
        {"confidence": 0.1416153159565678, "name": "affirm"}
    ]
}
```


Figura 10 Resltados de salida a la frase "I am looking for chinese food".

Para usar el Pipeline "Supervised_embeddings" se necesita instalar tensorflow y alguna librería más, asi que si se quiere utilizar estas versiones, se debe ejecutar el siguiente comando.

```
pip install rasa
```

3.1.2 CODIGO.

Una vez hemos instalado RASA, estamos listos para empezar a utilizar el sistema. En primer lugar, como toda técnica de machine learning necesitamos datos de entrenamiento. Para los chatbots es un poco complicado este aspecto ya que no suele haber datos de entrenamiento disponibles. Dependiendo del área para el que se desarrolla el chatbot puede que no haya ningún dato disponible. Cuando se habla de datos, nos referimos a conversaciones entre dos o mas personas. Una buena idea para empezar a preparar datos, es hacer que el tú el usuario eres el chatbot, y hacerte preguntas a ti mismo y como las contestarías(imaginar conversaciones que pudiera tener el chatbot si lo usarías). Lo mismo se puede hacer con ayuda de algún compañero para saber otras maneras de preguntar lo mismo y no solo a tu manera.

A continuación, se muestran tres conversaciones simples que utilizaremos para crear los datos de entrenamiento del chatbot:

U: User
B: ChatBot

Conversación 1:
U: Hey
B: Hello, how are you doing?
U: Pretty good
B: Great, carry on!

Conversación 2:
U: Hello there
B: Hello, how are you doing?
U: I am a little bit sad
B: Oh, that's bad, here is some music to cheer you up: (link).
U: Thanks
B: Did that help?
U: Yes
B: Goodbye then.

Conversación 3:
U: Whats up!
B: Hello, how are you doing?
U: I am not so good.
B: Oh, that's bad, here is some music to cheer you up: (link).
U: Thanks
B: Did that help?
U: no
B: Goodbye then
U: Bye Bye.

A continuación, hay que crear el fichero con los datos. El formato más utilizado y sencillo es markdown. Para ello creamos una intención "intent" que describa mejor la función de la frase. Para empezar, "hey" es un saludo por lo que la llamaremos greet. EL siguiente mensaje del usuario es "pretty good", que quiere decir que está bien, por lo que la intent es "mood_great". En la conversación dos se observa una manera distinta de saludar con "Hello there" por lo que se añade a la intent "greet". Otro ejemplo de intención es la afirmación o negación, muy habitual y que todos los chatbots tienen, "deny" y "affirm". Continuando de esta manera pasando cada mensaje de todas las conversaciones a una intención se rellena una base de datos de entrenamiento. A la derecha se observa el resultado.

```
## intent: greet
- hey
- hello there
- whats up

## intent: goodbye
- bye bye
- bye
- goodbye
- ciao

## intent: mood_great
- pretty good
- great
- good!

## intent: mood_sad
- not so good actually
- bad
- I am sad
- sad
- not feeling good

## intent: affirm
- Yes

## intent: deny
- No

## intent: thanks
- Thanks
- Thank you
- Thx
- Cheers bro
```

Figura 11 Datos NLU

Con esta cantidad de datos puede que no sea suficiente, por lo que se necesitarían más ejemplos o distintas

maneras de decir lo mismo. En la siguiente página se muestra una versión un poco más elaborada con estos datos.

El objetivo de Rasa NLU es predecir la intención del mensaje del usuario cuando se ha utilizado una nueva forma de expresar lo mismo.

```
38
39 ## intent:greet
40 - hey
41 - hello
42 - hi
43 - good morning
44 - good evening
45 - hey there
46
47 ## intent:goodbye
48 - bye
49 - goodbye
50 - see you around
51 - see you later
52
53 ## intent:affirm
54 - yes
55 - indeed
56 - of course
57 - that sounds good
58 - correct
59
60 ## intent:deny
61 - no
62 - never
63 - I don't think so
64 - don't like that
65 - no way
66 - not really
67
68 ## intent:mood_great
69 - perfect
70 - very good
71 - great
72 - amazing
73 - wonderful
74 - I am feeling very good
75 - I am great
76 - I'm good
77
78 ## intent:mood_unhappy
79 - sad
80 - very sad
81 - unhappy
82 - bad
83 - very bad
84 - awful
85 - terrible
86 - not very good
87 - extremely sad
88 - so sad
```

Figura 12 Datos NLU 2

El siguiente paso es definir el modelo de configuración del NLU. Es el momento de seleccionar el pipeline que se va a utilizar. La siguiente configuración especifica el lenguaje y pipeline que el modelo deberá usar.

Nombre del archivo: config.yml

```
language: en
pipeline: supervised_embeddings
```

3.2 RASA CORE. GESTION DE DIALOGOS.

¿Por qué utilizar un motor de gestión de Diálogos en un Chatbot? ¿No es más sencillo escribir if/else y no complicarse?

Rasa Core utiliza un modelo de machine learning que se entrena con conversaciones, para saber cómo actuar/responder al usuario. Hasta hace poco tiempo, los chatbots en general eran un gran número de estructuras if/else, creando grandes máquinas de estados. Esto cambio con la aparición de machine learning. A continuación se explica cómo funciona Rasa Core con esta filosofia.

Figura 13 Típica respuesta de un chatbot en cuanto se ha perdido en la conversación.

Las primeras de las razones por la que utilizar Rasa Core, es que al utilizar machine learning para escoger las respuestas de los datos de entrenamiento, no solo tiene un camino predeterminado que seguir que al salirse de él no sabe cómo actuar. Con esto se entiende que con máquinas de estados se deberían escribir todas las opciones posibles para que el chatbot actué, escribir reglas etc. Con machine learning lo aprendería al aportar datos de entrenamiento.

El modo de actuar de Rasa Core, es utilizar la intención y entidades extraídas por NLU más los datos de la acción previa, para predecir la siguiente acción. Estas predicciones utilizan SVM o una Recurrent Neural Network para predecir como actuar. Las Recurrent Neural Networks utilizan, como ya hemos dicho, la entidad e intención del mensaje actual más las acciones previas o llamadas de alguna API. Una vez superado ese paso, es importante añadir una máscara de acción para reducir las posibilidades de acción que ha de predecir. Por ejemplo, en vez de escoger entre un grupo de 10 acciones, se puede definir una máscara para que solo tenga que escoger entre 3 acciones. Una vez hecho, se re-normaliza la solución para volver a tener valores de porcentajes adecuados. Lo siguiente que realiza el chatbot, es crear una respuesta de muestra, que depende que

sea, si una llamada a otra función, llamada a una API, o el propio mensaje de salida final. Pasa por un último clasificador de acción y sale el mensaje final al usuario.

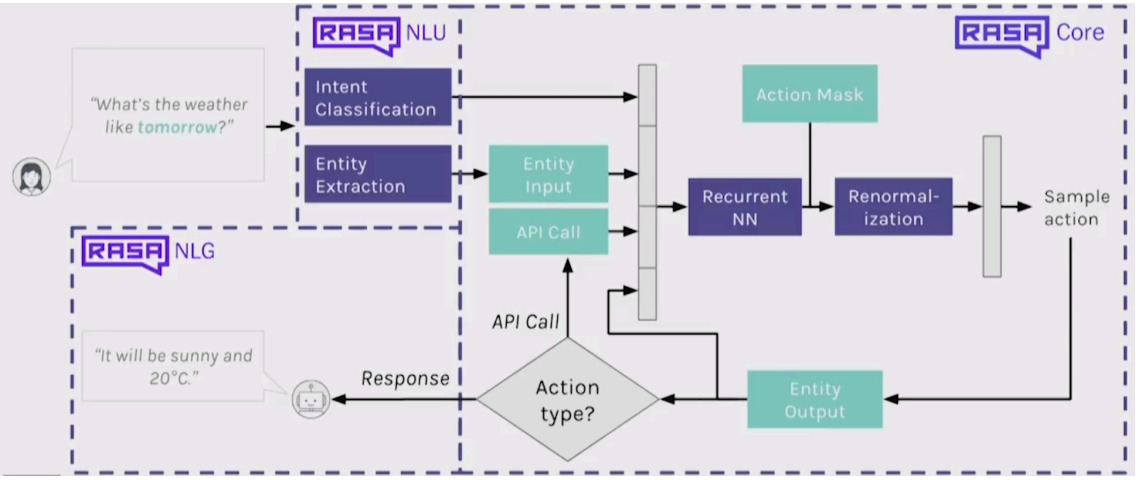

Figura 14 Esquema Arquitectura Rasa Core.

3.2.1 INSTALACIÓN Y CODIGO.

La instalación se realizó en un paso anterior (cuando se instaló Rasa NLU) ya que las versiones actuales instalan todo el paquete en un solo paso.

El ejemplo de dato de entrenamiento para el sistema de gestión de diálogos Rasa Core, se llama story. El formato es sencillo: empieza como las intent, con dos ## y seguidos del nombre de la story. No es necesario indicar el nombre de la story pero si se incluye, servirá para depurar más adelante el código desarrollado. Los mensajes mandados por el usuario comienzan con (*) y las acciones del bot comienzan con (-). A continuación, se muestran ejemplos de stories de las tres conversaciones presentadas anteriormente en NLU.

```
101
102 ## story_great
103 * greet
104   - utter_greet
105 * mood_great
106   - utter_happy
107
108 ## story_sad1
109 * greet
110   - utter_greet
111 * mood_unhappy
112   - utter_cheer_up
113   - utter_did_that_help
114 * affirm
115   - utter_happy
116
117 ## story_sad2
118 * greet
119   - utter_greet
120 * mood_unhappy
121   - utter_cheer_up
122   - utter_did_that_help
123 * deny
124   - utter_goodbye
125
126 ## story_goodbye
127 * goodbye
128   - utter_goodbye
```

Figura 15 Datos Rasa Core (stories)

Se observa que, las stories son los caminos que ha podido seguir la conversación cuantos más tenga mejor entrenado estará el chatbot. La primera story es un saludo del usuario, seguido del saludo del bot (pregunta que tal) así como la respuesta del usuario (se encuentra genial) y la respuesta del bot. Otro ejemplo es la última story en la figura, que solo es despedirse (*goodbye seguido de utter_goodbye). Pero de esta manera si comienzas la conversación despidiéndote, el chatbot entiende que te estas despidiendo y se despide. En otros bots si no está bien programado si comienzas con un goodbye, puede que te salude o te responda con su mensaje de inicio. Como pasaba con NLU se necesitan muchos datos para entrenar bien a Rasa Core y realizar un buen chatbot.

EL DOMINIO.

El dominio especifica el mundo en el que el chatbot opera. En el mundo del chabtot este universo incluye conjuntos de intents, entities y actions. También contiene plantillas de las respuestas que dará el chatbot al usuario.

Estas acciones, en general, simplemente mandan un mensaje pero se puede crear cualquier acción que se precise, como llamar a otras funciones etc.

```yaml
intents:
- greet
- goodbye
- mood_great
- mood_unhappy
- affirm
- deny
- thanks

actions:
- utter_greet
- utter_happy
- utter_cheer_up
- utter_did_that_help
- utter_goodbye


templates:
utter_greet:
- text: "Hello, how are you doing?"
- text: "Hey, how are you?"

utter_happy:
- text: "Great, carry on"

utter_cheer_up:
- text: "Oh thats bad, here is some music to cheer you up (link)"

utter_did_that_help:
- text: "Did that help?"

utter_goodbye:
- text: "bye bye"
- text: "See you later human"
```

Figura 16 Recorte del archivo domain.yml

3.3 ENTRENAMIENTO Y PRUEBA DEL MODELO.

El siguiente paso es el entrenamiento del modelo con las historias y datos del NLU. Para ello, en la última versión (rasa combinada con rasa X) simplemente introduciendo el siguiente comando en directorio del proyecto se entrenan la dos partes del sistema: Rasa NLU y Rasa Core.

```
rasa train
```

Se explica también a continuación la forma de hacerlo por partes, y la que era hasta hace poco la manera de hacerlo únicamente. En algunos casos se recomienda realizar una Virtual Enviroment(una herramienta para que los proyectos con distintas dependencias, separados isolandolos con un entorno virtual), para no tener problemas con las compatibilidades de las versiones, ya que es fácil que se tenga instalado varias versiones de Python o incluso Rasa que se haya creado al actualizar. Los siguientes pasos se puede obviar la creación de la Virtual Enviroment.

Primero para asegurarse de que se tiene la versión Python 3.6 se puede correr:

```
python3 –-version
```

Seguido se crea un entorno virtual (Virtual Enviroment) utiizando los siguientes comandos:

```
python3 –m venv env
source env/bin/activate
```

A continuación se entrenan los módulos del sistema (NLU y Core) y se crea el servidor local para que ejecute el chatbot. Para ello se utilizan los siguientes comandos:

```
make train-nlu
make train-core
make action-server &
```

Por último, se ejecuta el ultimo comando en la terminal y se comienza a utilizar el chatbot.

```
make cmdline
```

En este momento se puede empezar a hablar con el chatbot. Además se han presentado los conceptos básicos para mejorar y crear nuevos asistentes virtuales.

3.4 RASA X.

Rasa X es una nueva herramienta presentada y lanzada a finales de mayo de 2019, por lo que está en en fase de desarrollo (Early Access). Al no ser una versión final y no es del todo estable, pero es una herramienta que se puede utilizar y probar sin ningún problema.Esta es complementaria a Rasa y de uso totalmente opcional, para aprender de conversaciones reales y mejorar el Chatbot.

Al verlo la primera vez, la impresión que da no es la correcta. No es un servicio "hosted", lo corres localmente en tu máquina. No es una plataforma de point-and-click, donde no hace falta escribir ni realizar código.

Rasa X se ejecuta en tu propia máquina, y lo puedes lanzar en tu propio servidor. Los datos son propios, no se envían a ningún lado, en otros servicios se suelen quedar con los datos, y esto da la ventaja de tener mas privacidad en el proyecto.

¿Por qué construyeron Rasa X? En Rasa piensan que con solo algoritmos, no conseguirán resolver los problemas que plantean los chatbots. Estos sistemas necesitan aprender con conversaciones reales. Otra de las razones que justifican el nuevo módulo es la alta complejidad del desarrollo de la aplicación (construir un chatbot no es una tarea sencilla).

Figura 17 Logo Rasa X.

A continuación se desarrollan un poco las herramientas y funciones que destacan en Rasa X:

- Ver y anotar conversaciones: Filtrar, flagear, y corregir conversaciones que no fueron lo esperado. Por ejemplo, si un mensaje con la intención de pedir comida lo ha detectado como un saludo, la plataforma te permite editarlo de forma sencilla y se guarda en los archivos de datos de NLU y Stories con un simple click.
- Conseguir feedback/consejos de probadores: Se puede compartir tu asistente en la versión actual que se tenga con cualquiera, esto ayuda en conseguir feedback

para mejorarlo y en conseguir nuevos datos de entrenamiento. Ya que cada persona tiene su manera de hablar por lo que se convertirá en un mejor chatbot.

- Versiones y gestiona los modelos: Se puede tener varias versiones de chatbot, y probar rápidamente y si no es lo que esperabas fácilmente volver hacia atrás a la versión anterior.

Con la llegada de Rasa X, se produjo la unión de las dos librerías principales (Rasa NLU y Rasa Core). De esta manera es mucho más sencilla la la gestión de documentos y archivos así como el entrenamiento y la definición de comandos a utilizar. Para entrenar el chatbot se ejecuta el siguiente comando que guardará el modelo el directorio ./models.

```
rasa train
```

Una vez entrenado, para cargarlo y empezar a hablar con el chatbot:

```
rasa shell
```

¿COMO UTILIZAR RASA X?

Para crear un proyecto nuevo o utilizar uno ya creado con Rasa X, se corre el siguiente comando:

```
rasa init
```

Y para iniciar Rasa X en un servidor, en el directorio del proyecto se ejecuta:

```
rasa x
```

La interfaz se muestra en la siguiente imagen. Como se observa en el menú aparecen varios campos:

- Talk to your Bot: Para hablar con el chatbot, probar y corregir modelos. Se pueden flagear/marcar mensajes para más adelante editarlos. Hay dos modos de trabajo: el modo Talk para hablar con el chatbot y el modo interactive learning, donde paso a paso le dices al chatbot que acción realizar (te sugiere la acción y el usuario le dice si es la correcta o no). También se pueden crear "stories o definir nuevos datos para las intenciones del NLU.

- Conversations: Se archivan las conversaciones que el chatbot ha tenido, para revisarlas o incluso corregirlas: También en este apartado se puede compartir el chatbot para que tus amigos o cualquiera lo pruebe y se archiven sus conversaciones y puedas utilizar sus datos de conversaciones reales en tu asistente.

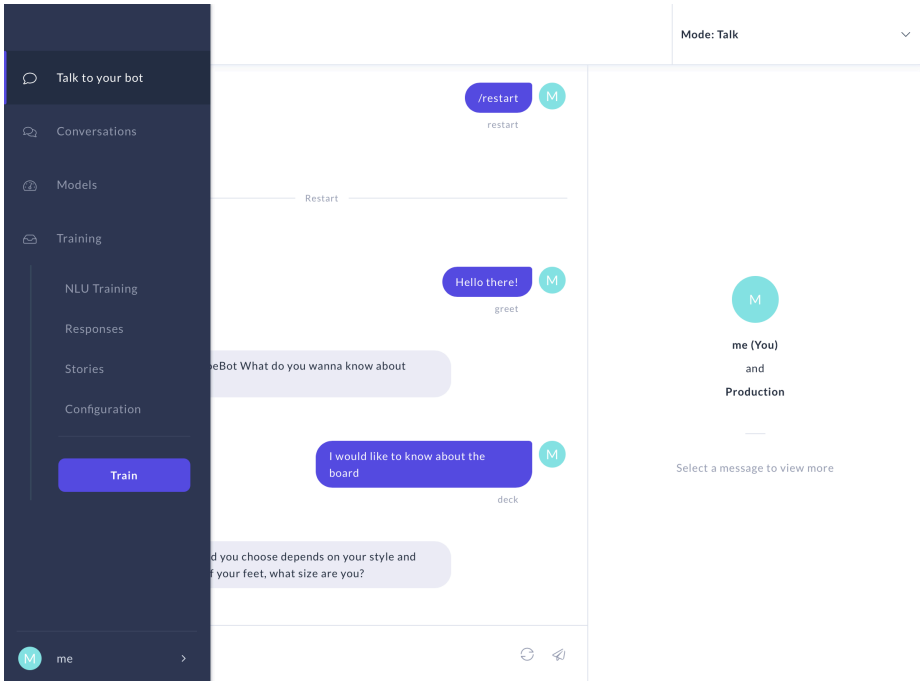

Figura 18 Pestaña talk to your bot.

- **Models:** Al pulsar el botón de entrenar se guarda el modelo en la pestaña modelos. En dicha ventana es posible seleccionar el modelo que se usará en modo production. Dicho modelo está preparado para usarse en la pestaña "Talk to your bot".

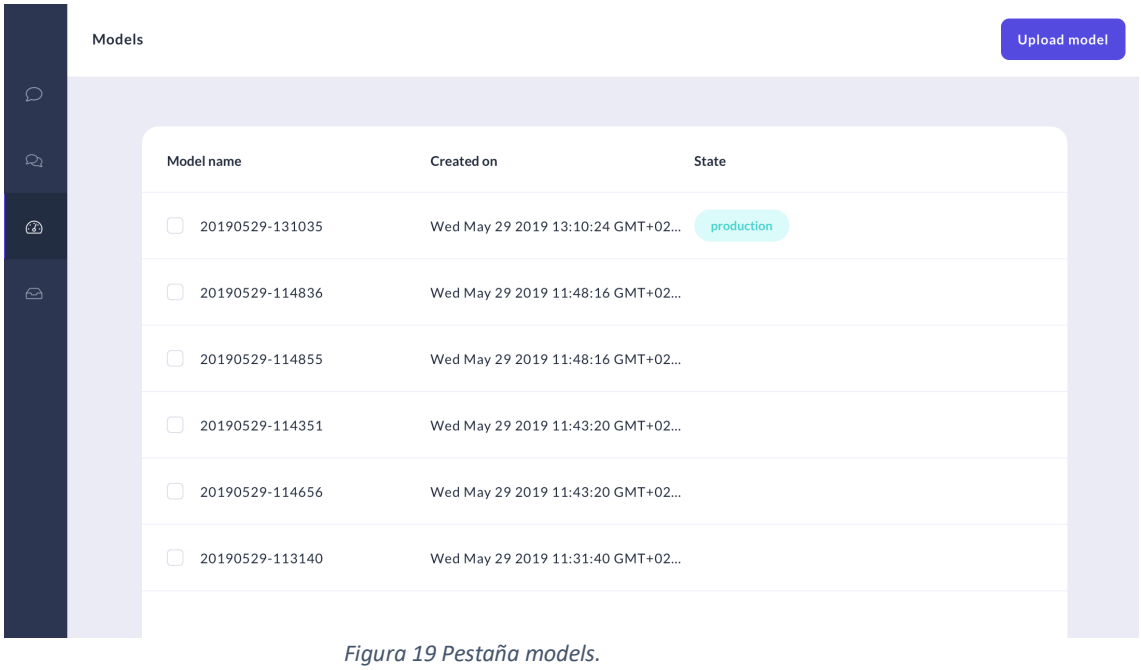

Figura 19 Pestaña models.

- **Training:** En la pestaña training se pueden ver y editar los datos de entrenamiento, tanto de NLU como de las stories de entrenamiento, las plantillas de respuestas y la configuración del pipeline.

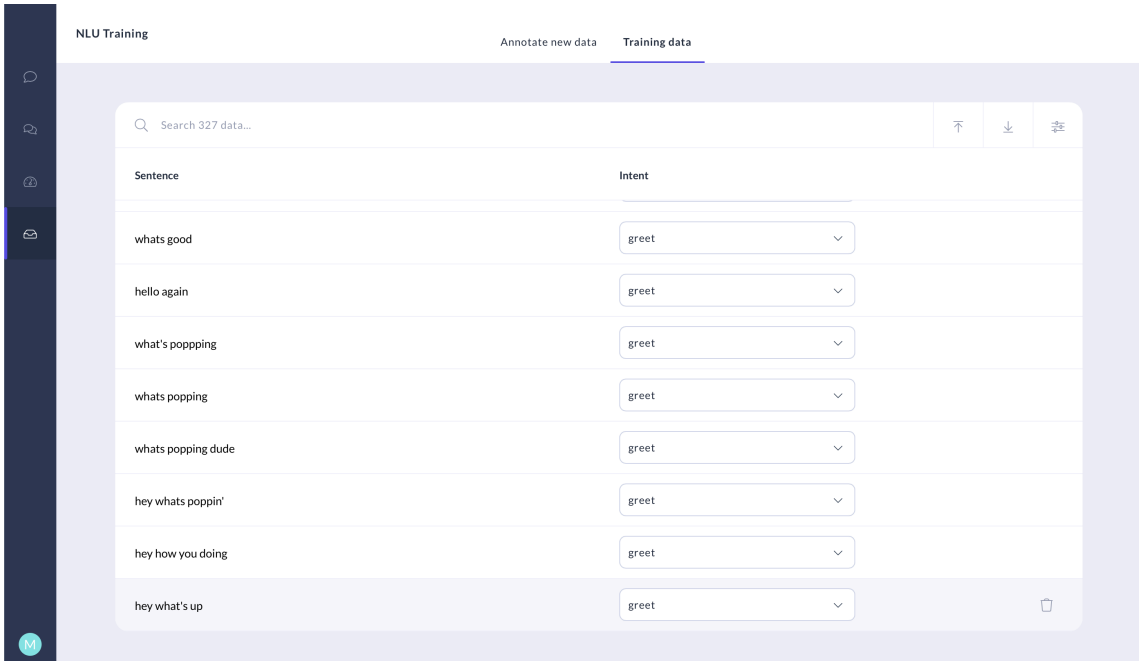

Figura 20 Pestaña Training, mostrando NLU Training Data.

Por último y como aspecto más interesante de Rasa X, ya mencionado antes, una vez se tenga un bot que funcione, se puede compartir con quien se quiera, y de esta manera aprender de verdaderos flujos de conversación, y no solo como el creador crea que van a surgir. Ya que dichos flujos que los usuarios realicen se archivaran y se podrán corregir, guardar como datos de entrenamiento y aprender o generar nuevas ideas que no se habían pensado. Para ello, solo con pulsar en compartir en la pestaña de conversaciones, se puede generar un link y enviárselo a quien se quiera. Los usuarios que lo prueben no necesitaran instalar nada, y el creador podrá ver las conversaciones en la pestaña conversations.

Se tendrá que instalar una única librería para que los usuarios puedan acceder a tu servidor local y hablar con el chatbot. Se llama ngrok y es gratis. Por defecto rasa X corre en el puerto 5002, para hacer este puerto público se necesita insertar el siguiente comando:

```
ngrok http 5002
```

Y está listo para que empieces a crear el chatbot de tus sueños.

4. EJECUCIÓN Y CREACION DE UN CHATBOT REAL.

Aunque la especificación de este chatbot es ficticia, se ha abordado un caso de uso muy cercano a la realidad y que responde a los intereses del autor.

Cada vez es más grande el mundo del patín (skateboarding), siendo por primera vez deporte olímpico en los próximos Juegos Olímpicos de Tokyo 2020. Y las primeras impresiones positivas de un joven (o de un adulto que también les hay que patinan) cuando se acercan a un parque de skate (Skatepark), es que o les gusta e inmediatamente quieren probar a prácticarlo y para ello necesitan un patin. En el caso de los niños se lo piden a sus padres o el mismo padre quiere regalarle uno a su hijo, o incluso el propio adulto quiere uno para el mismo. Y la preguntas mas usuales que se hacen todos son siempre las mismas: ¿Existen distintos tipos de patinetes?, ¿Cuál es mejor para empezar? ¿Qué me debería comprar? ¿Hay distintas medidas de tablas? ¿Qué ruedas son las buenas? Etc.

Claro que la respuesta sencilla, ve a la tienda más cercana y pregunta al dependiente que él te aconsejara, que para eso trabaja en una tienda de patín o deportes. Es muy probable que el encargado no tenga ni idea y que lo único que sabe es cobrar y trabajar en la tienda, pero no es ningún especialista en patinetes. Y como cada vez más y teniendo en cuenta los tiempos en los que estamos, que mas a menudo todo se compra por internet, y todo se puede comprar por internet, está es la opción que la mayoría de gente optaría por hacer.

Y sí, en internet seguro que se puede encontrar un post de un blog que indique las instrucciones a seguir para comprar un patinete para principiantes o no. Ya que muchas veces incluso los skaters más veteranos no saben que es lo indicado, y siguen usando las mismas medidas, mismas ruedas etc, que la primera vez ya que les ha ido bien con ello. También seguro que hay videos en Youtube recomendado y aconsejando sobre los patinetes, ya que otra vez, el mundo del skate es muy numeroso.

Y algo que está en auge son los asistentes virtuales o chatbots que, gracias a los capítulos anteriores, ya sabemos cómo se desarrollan. Cada vez más transacciones y preguntas frecuentes se realizan por medio de chatbots, cuando antes o se hacían por medio de una llamada telefónica o en persona.

De esta idea nace el caso de uso: crear un asistente virtual/chatbot para tiendas online de patinetes. En capítulo es de aplicación general ya que cualquier e-commerce podría implementar el chatbot en su página web o aplicación, si es que la tiene.

La función principal de este chatbot es ser un ayudante para cualquier persona que quiera saber qué patinete adquirir en la tienda, ya sea para el mismo o para sus hijos. Si eres alguien que quiere hacerse con un patinete completo, o cambiar las ruedas, cambiar los ejes, cambiar la tabla etc, este chatbot te servirá como ayudante sin tener que leer un

post de un blog y ver un video de Youtube buscando cual es la parte que te interesa a ti. Con este assitente se puede ir directamente al grano, con preguntas escrita por ti y respondidas por el chatbot propuesto.

4.1 DATOS.

Gracias a los capítulos anteriores sabemos cómo crear el chatbot. Lo primero que se debe hacer es tener una buena base de datos y estructurarla de forma adecuada. De estas tarea se ha encargado el autor de este proyecto, que se considera con bastante experiencia en el mundo del skateboarding. En el caso de no tener ni idea del tema que aborda el chatbot, la alternativa es buscar información en internet o preguntar a algún experto o gente con conocimientos avanzados en el tema.

Una vez se tienen los datos que se quieren introducir en el conocimiento del chatbot, lo siguiente es como estructurar el chatbot. Plantear una pequeña idea de cómo quieres que actue el chatbot. A continuación, se explica detalladamente dicha estructura.

4.2 ESTRUCTURA.

Como casi todos los chatbots, se requiere que para iniciar la conversación se salude al chatbot. Una vez saludado, el chatbot responde con un mensaje de bienvenida, respondiendo al saludo y presentándose. Esta presentación inluye una breve explicación de los aspectos en los que puede ayudar al usuario. En este caso le dice que sabe todo sobre el patín y derivados.

```
## intent:greet
- hello
- hello bro
- hello bot
- hello there
- hi
- hi bro
- hi bot
- hi there
- good morning
- good afternoon
- good evening
- good night
- whats up
- what's up
- whats cooking
- what's cooking
- whats cooking bot
- whats up bro
- how you doing
- how you doing bot
- how you doing bro
- heello bot
- hi bot
- hey
- hey there
- hey bot
- hey bro
- hey whats up
- hey what's up
- hey how you doing
- hey whats poppin'
- whats popping dude
- whats popping
- what's poppping
- hello again
- whats good
- Hello there!
- Hey there!
```

Figura 21 Datos NLU con el saludo al bot.

```
utter_name:
- text: "Hey there! I'm JoeBot What do you wanna know about Skateboarding?"
- text: "Hello there! My name is Joebot, and I know everything about skateboarding."
```

Plantillas de respuesta con la presentación del bot.

De aquí se hace una predicción, o la predicción más probable que se crea. De esta presentación saldrán tres ramas principales, dependiendo del interés del usuario. Un skate está formado por varios módulos:

- Tabla (Board/deck)
- Ruedas (Wheels)
- Ejes (Trucks)
- Rodamientos (Bearings)
- Tornillería (Hardware)
- Lija (Grip)

De estos, los más importantes (y que más pueden influir más a la hora de diferenciar un patín de otros) son los tres primeros: Tabla, ruedas y ejes. Estas van a ser los tres caminos principales que saldrán a raíz del mensaje de presentación o al menos los mas probables:

Pregunta sobre tablas: Se espera del usuario que enuncie preguntas tipo ¿Qué tabla me recomiendas?, ¿Qué tabla me debo comprar?,¿Quiero saber sobre tablas?, etc. De estas preguntas se derivan cuestiones muy básicas, como por ejemplo ¿Qué talla de pie usas? Las tablas se diferencian en el ancho mayoritariamente por lo que dependen del tamaño de pie del patinador lo que orientará la recomendación. Se ha decidido agrupar las tallas de los usuarios en tres tamaños, que se llamaran pie pequeño (smallfeet), pie mediano (mediumfeet), y pie grande (largefeet).

A la hora de realizar y añadir los datos de entrenamiento a nuestro chatbot Rasa, se tendrán en cuenta tres intenciones del usuario y tres respuestas por parte del bot. En las siguientes capturas se muestran los datos de entrenamiento:

```
## intent:deck
- about decks
- what about the deck
- about the board
- about boards
- decks
- boards
- deck
- board
- What size of deck should i ride
- What size of board should i ride
- Want deck do you reccomend me
- what baord do you reccomend me
- It is important the size of the board
- I saw there are sizes on the boards, does it matter
- i want to know what size of board is the right for me
- What kind of board should i buy
- what deck should i ride
- I would like to know about the board
```

Figura 22 Datos NLU con la intención de preguntar sobre tablas.

```
utter_deck:
- text: "The deck or board you choose depends on your style and mainly the size of
your feet, what size are you?"
```

Figura 23 Plantilla de respuesta del bot a la pregunta sobre tablas.

```
## intent:smallfeet
- 7
- 7.5
- 8
- i am 7
- i am 7.5
- i am 8
- i am size 7
- i am size 7.5
- i am sie 8
- i use 7
- i use 7.5
- i use 8
- i wear 7
- i wear 7.5
- i wear 8
- im 7
- im 7.5
- im 8
```

```
## intent:mediumfeet
- 8.5
- 9
- 9.5
- 10
- i am 8.5
- i am 9
- i am 9.5
- i am 10
- i am size 8.5
- i am size 9
- i am sie 9.5
- i am size 10
- i use 8.5
- i use 9
- i use 9.5
- i use 10
- i wear 8.5
- i wear 9
- i wear 9.5
- i wear 10
```

```
## intent:largefeet
- 10.5
- 11
- i am 10.5
- i am 11
- i am size 10.5
- i am size 11
- i use 10.5
- i use 11
- i wear 10.5
- i wear 11
- im 10.5
- im 11
```

Figura 24 Datos NLU a respondiendo a la pregunta sobre tablas. (talla del pie)

```
utter_smallfeet:
- text: "Ok so you should ride between a 7.5 and 8.12, this boards are also specially
used on technical skaters for doing very difficult tricks."

utter_mediumfeet:
- text: "OK so you should ride between and 8.12 and 8.5"

utter_largefeet:
- text: "Ok, so you should ride a deck between 8.5 and above."
```

Figura 25 Plantillas de respuesta a la talla del pie.

```
## story_smalldeck
* deck
 - utter_deck
* smallfeet
 - utter_smallfeet

## story_mediumdeck
* deck
 - utter_deck
* mediumfeet
 - utter_mediumfeet

## story_largedeck
* deck
 - utter_deck
* largefeet
 - utter_largefeet
```

Figura 26 Datos de entrenamiento de Rasa Core, Stories con los distintos flujos de dialogo sobre tablas.

Pregunta sobre Ruedas: Al igual que con las tablas, se esperan preguntas tipo, ¿qué ruedas me recomiendas?, ¿qué ruedas debo comprar?, Quiero saber sobre ruedas, cuéntame, etc. Existen en el mercado dos tipos de ruedas: unas más duras y otras más blandas. La diferencia básica es que unas se agarran más que otras, lo que hace que unas derrapen más fácilmente y otras no. Entonces la pregunta que enunciara el chatbot es que prefiere agarrase al suelo (stick to the ground) o ir derrapando (sliding). A continuación, se muestran los datos de entrenamiento creados para resolver estas preguntas:

```
## intent:wheels
- about wheels
- wheels
- about the wheels
- What size of wheels should i ride
- the size of the wheels it is important
- it is important the size of the wheels
- I want to know about the wheels
- What kind of wheels should i buy
- What wheels are the best
- and what about the wheels
```

Figura 27 Datos NLU, con la intención de preguntar sobre ruedas.

```
utter_wheels:
- text: "There are two types of wheels, hard and soft, What do you preffer to slide
or stick to the ground and cruise around."
```

Figura 28 Plantilla de respuesta, a la pregunta sobre ruedas.

```
## intent:hard_wheels
- slide
- sliding
- to slide
- i preffer to slide
- i like to powerslide
- slippery
- i skate everything
- I preffer to slide
```

```
## intent:soft_wheels
- to stick
- to stuck
- to cruise
- cruise
- cruising
- i preffer to cruise
- i preffer more chill
- i like cruising
- i like to stick to the ground
- i dont like to slide
```

Figura 29 Datos NLU, respondiendo al bot (tipo de ruedas).

```
utter_hardwheels:
- text: "Oh, you are like me, you like go powersliding everywhere, so you should get
hard wheels around 54 mm."

utter_softwheels:
- text: "OH you preffer to be safer and cruise around, you should get soft wheels."
```

Figura 30 Plantillas de respuesta del bot, indicando el tipo de ruedas para el usuario.

```
## story_hardwheels
 * greet
  - utter_name
* wheels
  - utter_wheels
* hard_wheels
  - utter_hardwheels
* thanks
  - utter_noproblem
* goodbye
  - utter_goodbye

## story_hardwheels2
* wheels
  - utter_wheels
* hard_wheels
  - utter_hardwheels
```

```
## story_softwheels
* greet
  - utter_name
* wheels
  - utter_wheels
* soft_wheels
  - utter_softwheels
* thanks
  - utter_noproblem
* goodbye
  - utter_goodbye

## story_softwheels2
* wheels
  - utter_wheels
* soft_wheels
  - utter_softwheels
```

Figura 31 Datos de entrenamiento Rasa Core, a las stories sobre ruedas.

Se aprecia que las stories de la captura superior se repiten dos veces, solo que una es un flujo desde el inicio de la conversación incluyendo el saludo y lo primero que le pregunta seguido del saludo es sobre ruedas. Y la otra existe para que cuando se pregunte en cualquier momento sobre ruedas el chatbot sepa que hacer y no solo si se le pregunta después del saludo. De esta manera en cualquier momento de la conversación si se quiere saber sobre ruedas el chatbot reconocerá la story de las ruedas, y no solo si el flujo empieza en el saludo inicial.

Pregunta sobre ejes: También se esperan por parte del usuario preguntas tipo ¿Qué tipo de ejes me recomiendas?, ¿Son todos los ejes iguales?, ¿Qué pasa con los ejes?, Quiero saber sobre ejes etc. La respuesta a qué tipo de ejes adquirir es más sencilla que las otras dos, con tener los ejes no más anchos que la tabla, es la medida perfecta. Se muestran seguidamente los datos de entrenamiento generados para este dialogo:

```
## intent:trucks
- about trucks
- trucks
- about the trucks
- What size of trucks should i ride
- What size of trucks should i buy
- the size of the trucks it is important
- it is important the size of the trucks
- I want to know about the trucks
- What kind of trucks should i buy
- What trucks are the best
- what about trucks
```

Figura 32 Datos NLU, preguntando sobre ejes.

```
utter_trucks:
- text: "The trucks depends on the size of your deck, so they should be no wider than
your board and also not to narrow."
```

Figura 33 Plantilla respuesta del bot a la pregunta sobre ejes.

Se observa que en los datos de NLU las intenciones señalan muchas maneras de decir lo mismo, pero también ejemplos mas simples como solo la palabra clave para dar paso a la respuesta, o maneras de seguir la conversación. Por ejemplo, si te acaban de responder a la pregunta qué tipo de tabla me recomiendas, y el bot te ha respondido “Anything else you want to know about?”, la nueva pregunta probablemente no vaya a ser de nuevo “i want to know about the trucks”, sino más bien algo más rápido y más real “about trucks”. Este tipo de datos de mas tipo como hablan las personas en la realidad.

Después de estos tres flujos de dialogo, lo más probable es la conversación termine con unas gracias o despidiéndose e incluso con otra pregunta. Estos datos de entrenamiento mostrados a continuación son casi de uso obligatorio, ya que son frases que siempre se usan al igual que los saludos.

```
## story_goodbye
* goodbye
 - utter_goodbye

## story_thanks
* thanks
 - utter_thanks
```

```
## intent:goodbye
- goodbye
- bye
- bye bye
- ciao
- ciao pescao
- see you later
- see you
- see ya
- see you soon
- adios
```

```
## intent:thanks
- thank you
- thanks
- thanks bro
- thx
- thankss
- thank you so much
- thank you very much
- cheers
- cheers bro
- cheers mate
- aprreciate it
- oh thanks man
- oh thanks bro
- oh thank you
- oh thank you bro
- ok thanks
- thanks bot
- thanks so much man
```

Figura 34 Datos NLU y Core con las intenciones y stories de Gracias y Adios. (Thanks and Goodbye).

```
utter_goodbye:
- text: "Talk to you later bro!"
- text: "See ya later dude!!"
```

```
utter_thanks:
- text: "Cheers man!"
- text: "Thank you"
```

Figura 35 Plantillas de respuesta de gracias y adiós. (Se lanzará aleatoriamente una de las dos).

Esas tres serían las tres ramas principales, ya que si el chatbot está ubicado en unq tienda online que vende patines, lo principal es que responda a las dudas sobre que comprar y a las partes que componen el skate. Pero no se quiere que el chatbot sea tan simple y básico. Por ello, se requiere que el chatbot sepa todo lo relacionado con el skateboarding. Pensando en este objetivo, se plantean otras dos ramas de conversación importantes:

¿Qué se quiere hacer una vez tienes el skate? A no ser que solo se quiera la tabla para desplazarse, lo más llamativo del skate son los trucos. Por lo que una de las ramas explora los trucos con el patinete. La otra también es básica: la gente quiere inspiración y motivación por lo que un buen video le sacará a la calle a intentar mejorar. De aquí nace la otra rama, la pregunta sobre que videos me recomiendas, ya que hay millones en Youtube, pero el chatbot recomendará los mejores.

Puede que tu caso sea distinto pero el 99% de los skaters, comienzan aprendiendo tres trucos básicos: Ollie, Shove it y KIckflip. A continuación, se muestra los datos de entrenamiento generados para responder a preguntas sobre que trucos se deben aprender primero.

```
## intent:tricks
- basic tricks
- what are the basics tricks
- what are the basics tricks to learn
- what tricks i should learn first
- tricks for begginers
- i want to know about tricks
- about tricks
- trick tips
- i wanna know about tricks
- what trick is easy
- easy tricks
- easy tricks?
- tricks?
- tricks
- what about the tricks
- what tricks
```

Figura 36 Datos de entrenamiento NLU, preguntando sobre trucos básicos.

```
utter_tricks:
- text: "Good question! There is three basic tricks everyone should learn: Ollie,
Flip, and Shove_it."
```

Figura 37 Plantilla de respuesta a la pregunta sobre trucos.

```
## intent:ollie
- ollie
- i want to know about the ollie
- about the ollie
- what is an ollie
- tell me about the ollie
- how you do an ollie
```

```
## intent:shove_it
- shove it
- i want to know about the shove it
- about the shove it
- what is an shove it
- tell me about the shove it
- how you do a shove it
```

```
## intent:flip
- flip
- i want to know about the flip
- about the flip
- what is an flip
- tell me about the flip
- how you do a flip
```

Figura 38 Datos de entrenamiento NLU, con las intenciones de especificar el truco sobre el que se quiere saber.

```
## story_ollie
* tricks
  - utter_tricks
* ollie
  - utter_ollie
* thanks
  - utter_thanks
```

```
## story_flip
* tricks
  - utter_tricks
* flip
  - utter_flip
* thanks
  - utter_thanks
```

```
## story_shove_it
* tricks
  - utter_tricks
* shove_it
  - utter_shove_it
* thanks
  - utter_thanks
```

Figura 39 Stories para que el bot siga el dialogo de los trucos.

```
utter_ollie:
- text: "This is the first trick everyone learns, consist on jumping with the
skateboard and elevate the four wheels off the ground. It's the most important
trick."

utter_flip:
- text: "This is one of the better looking tricks, and not as easier as the ollie.
Consist on fliping the board 360 degrees on his long axis."

utter_shove_it:
- text: "Second trick almost for everyone, consist on rotating the board 180 degrees,
you can do it without elevating the wheels off the ground at first."
```

Figura 40 Plantillas respuesta del bot a los trucos.

El dialogo respecto a los videos es mucho más sencillo, con una sola respuesta, se puede dar la información necesaria al usuario.

```
## intent:skate_video
- know any good skate video
- know any good skate films
- do you reccomend any skate videos
- skate part
- what skate video you recommend
- what skate part you reccomend
- any videos i should watch
- any inspiring videos
- any good videos
- skate videos?
- skate parts?
- good skate videos
- any good sk8 video part?
- what videos do you recommend me
- what skate videos do you recommend me
```

Figura 41 Datos NLU, con la intención de preguntar sobre videos de skate.

```
utter_skate_video:
- text: "There are plenty of good skate videos and parts, I'll recommend you to watch Pretty Sweet by Spike Jonze, its a really creative video, with good skating too."
```

Figura 42 Plantilla respuesta a la pregunta sobre videos.

Estas serían las ramas principales que se han decidido crear. Se podrían crear muchas más, pero llevaba mucho tiempo generar tantos datos para algo ficticio que no iba a tener un uso real. Aun así, el chatbot tiene más conocimiento, y entiende preguntas más elaboradas, como por ejemplo ¿cómo estas?, ¿quién es tu creador?,¿ cuéntame un chiste? (responde con un chiste aleatorio de Chuck Norris, malísimos la verdad), etc. Tiene algún pequeño secreto también; por ejemplo, si le preguntas cuantos años tiene, te dice que no te lo va a decir. Pero si le insistes te lo dice. Desarrollar esto es muy sencillo con el uso de Rasa Core y sus stories.

```
## intent:age
- how old are you?
- old?
- how old are you
- how young are you
- how many years do you have
- you have age?
- age?
- age
```

```
utter_age:
- text: "Oh I wont tell you, sorry."
```

```
## intent:age2
- please tell me your age
- how old are you?
- please tell me
- tell me please
- so you are not gonna tell me
- please
```

Figura 43 Datos NLU Y plantilla de respuesta a cuantos años tiene el bot.

Creando la siguiente storie, el bot solo responderá con utter_age2, si se le ha repetido la pregunta, o se le ha pedido por favor.

```
## story age
* age
 - utter_age
* age2
 - utter_age2
```

```
utter_age2:
- text: "You insist so i'll tell you, 1 Month old."
```

Figura 44 Storie y plantilla, de la edad del bot.

Con esto se terminan de generar los datos de entrenamiento. Como se ha dicho anteriormente hay más datos, y más stories, pero no se muestran todas en el documento.

4.3 ENTRENAMIENTO.

El siguiente paso, como hemos visto anteriormente en el tutorial, es entrenar el modelo. Para ello nos dirigimos al directorio del proyecto y se introducen los comandos para entrenar el bot. En este caso, se realizó sin el uso de Rasa X, por lo que los comandos son la versión larga, entrenando cada librería por separado: Rasa NLU y Rasa Core. Se muestran a continuación los resultados de los comandos de entrenamiento.

```
make train-nlu
```

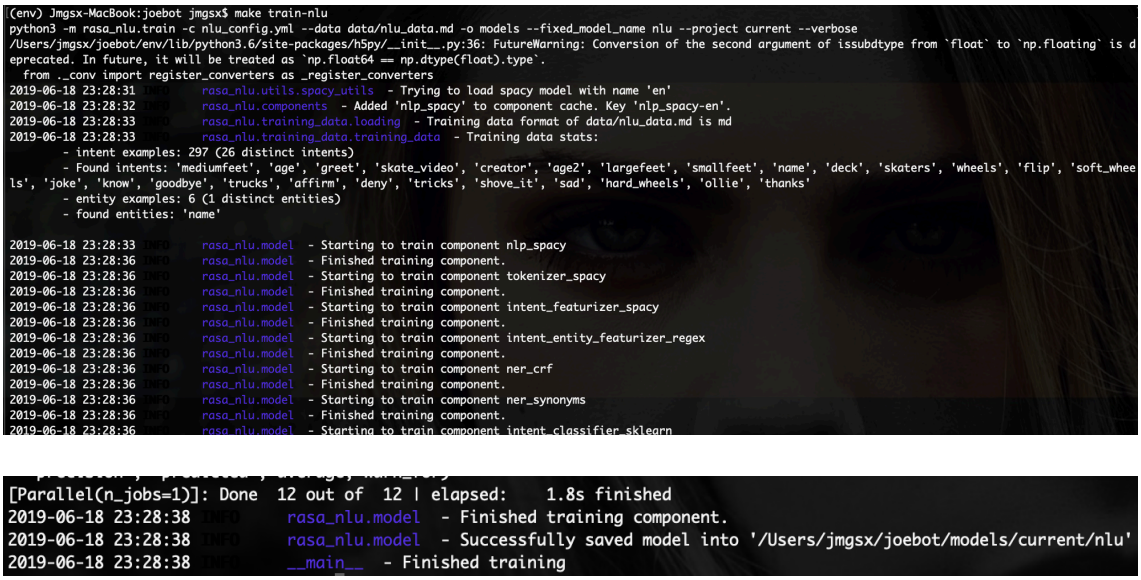

Figura 45 Arriba: Comando make train-nlu y primeros pasos que realiza. Abajo: Final de proceso.

```
make train-core
```

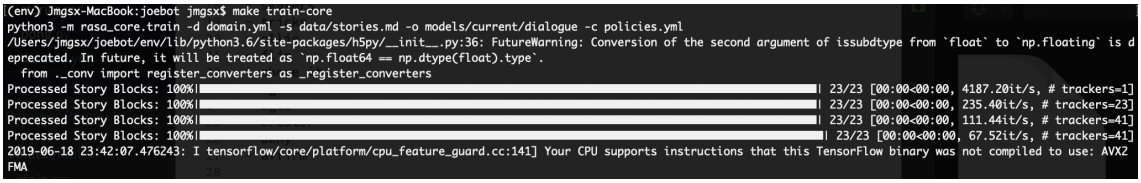

Figura 46 Inicio del comando make train-core.

4.4 PRUEBA DEL PRODUCTO.

Una vez entrenado, solo falta probarlo y empezar a hablar con el chatbot. A continuación, se muestra una pequeña conversación, mostrando el resultado final de chatbot. Se muestra un pequeño saludo en la terminal, y el resto de la conversación en la interfaz de Rasa X, ya que es más bonita y permite demostrar que se ha podido utilizar y probar Rasa X.

```
make cmdline
```

```
(env) Jmgsx-MacBook:joebot jmgsx$ make cmdline
python3 -m rasa_core.run -d models/current/dialogue -u models/current/nlu --endpoints endpoints.yml
/Users/jmgsx/joebot/env/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second
eprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
2019-06-18 23:54:03 INFO     root  - Rasa process starting
2019-06-18 23:54:04 INFO     rasa_nlu.components  - Added 'nlp_spacy' to component cache. Key 'nlp_spacy-en'.
2019-06-18 23:54:11.936202: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions
FMA
2019-06-18 23:54:13 INFO     root  - Rasa Core server is up and running on http://localhost:5005
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  Hey there
Hello there! My name is Joebot, and I know everythinga about skateboarding.
```

Figura 47 Ejemplo de saludo en la terminal.

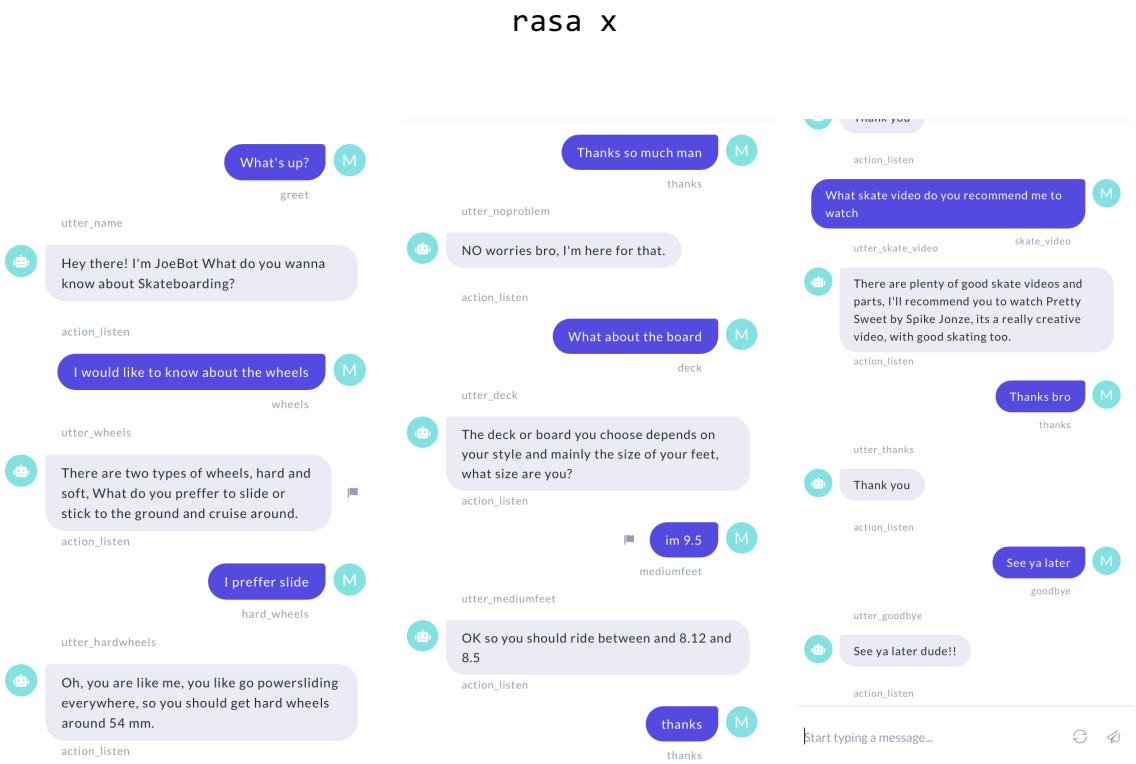

Figura 48 Conversación final con el chatbot comprobando que funciona correctamente.

Los resultados son satisfactorios, ya que se observa que detecta correctamente todos los mensajes con su intención y responde como debe debido a las stories generadas. Es verdad que para ser un chatbot que entienda nuevos mensajes no guardados en los datos de entrenamiento harían falta más datos pero con los proporcionados son suficientes para hacer un buen trabajo

5. FLOW XO

Antes de crear el chatbot y tutorial con Rasa, se evaluaron otras plataformas como FLOW XO. Esta plataforma permite crear chatbots en la web, por lo que no se requiere conocimiento de código alguno. En principio esta posibilidad parecía muy interesante y por ello se procedío a su estudio y evaluación.

La filosofía de desarrollo de FLOW XO se basa en arrastrar y soltar cajitas o acciones en una ventana. Un chatbot creado con FLOW XO está formado por flujos (flows). Un flujo es como se define la conversación entre el chatbot y el usuario. Un flow lo forman disparadores y acciones. El disparador está esperando a que el usuario utilice palabras o frases claves. Y la acción puede ser desde mandar un mensaje de respuesta, generar una pregunta, o mandar una carta.

A continuación, se muetra un ejemplo:

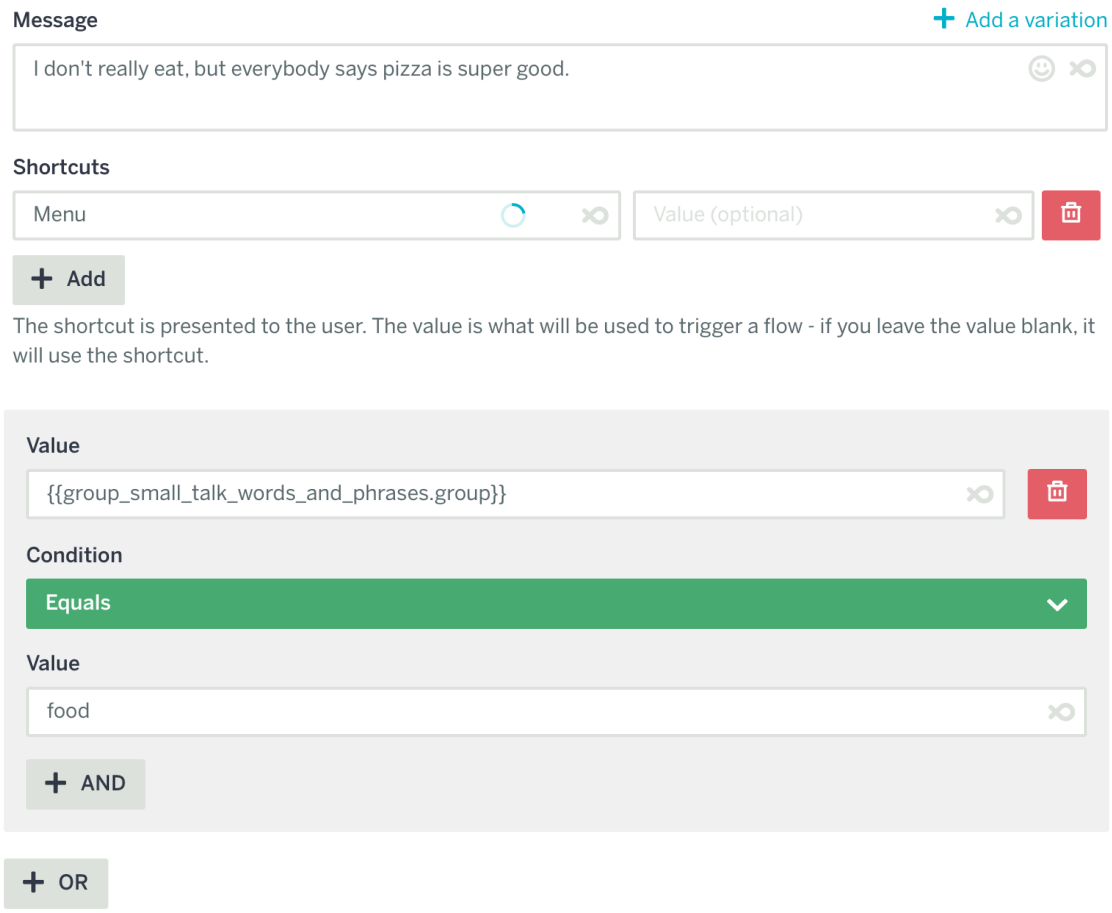

Figura 49 Ejemplo ventana creación de flow.

La acción de este flow es mandar un mensaje, “I dont really eat, but everybody say pizza is super good”. Para activarse, o para mandarse, se observa en la segunda imagen que

cuando la conversación sea igual a “food” se generará este mensaje. Esto no quiere decir que solo vale la palabra clave “food”, “{{group_small_talk_words_and_phrases.group}}” Sino que food es un grupo de datos con palabras clave, y hay varias en el grupo. Por ejemplo, en este caso están guardadas “eat” y “food”. En espera de que el usuario pregunte: “What is your favourite food?” o “Do you eat?”. Si en el mensaje no se encuentra una de las palabras clave la acción no salta, por lo que no se enviará dicho mensaje.

De esta manera, a base de filtros y condiciones “AND” y “OR” se pueden definir situaciones más complejas.

Una vez se conoce el modo de creación de flows, se desarrolló un chatbot igual al de Rasa, que conoce todo sobre el patin. Se muestran a continuación algunas imágenes de los resultados, datos y acciones.

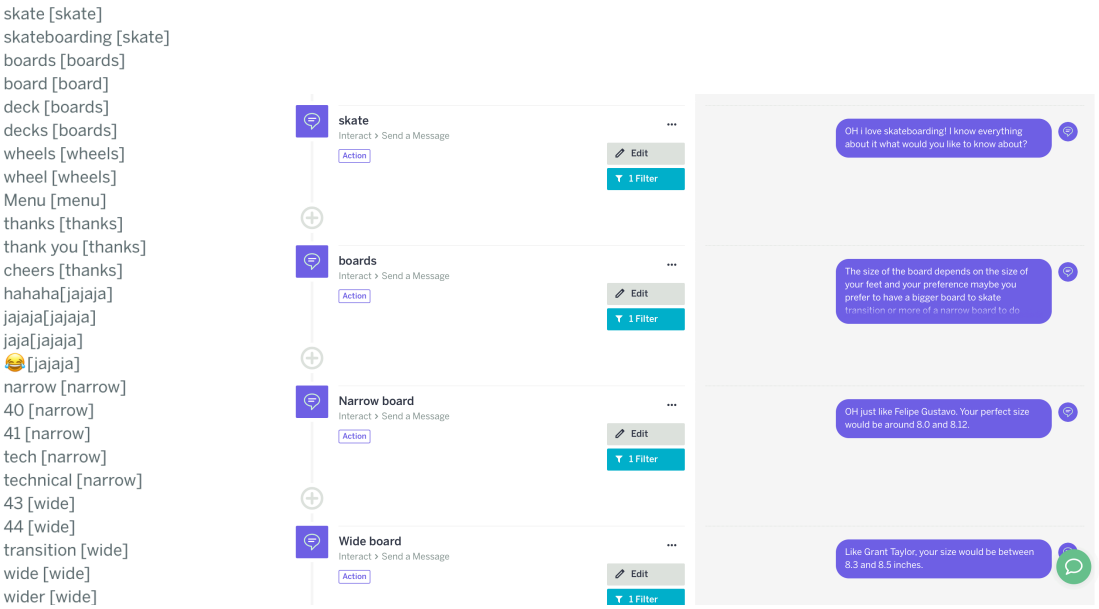

Figura 50 Datos y algunas acciones creadas en la interfaz de FLOW XO.

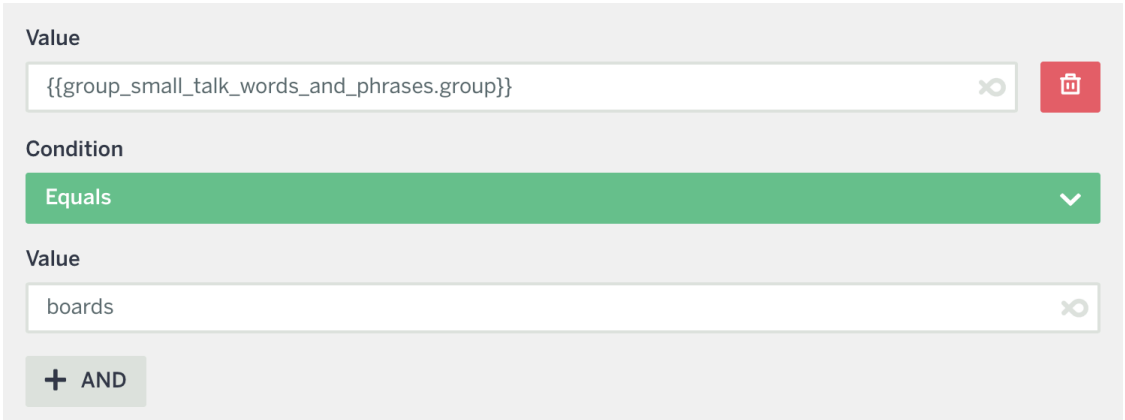

Figura 51 Filtro para la acción boards. (Solo se ejecutará si la frase contiene alguna palabra del grupo de datos boards.)

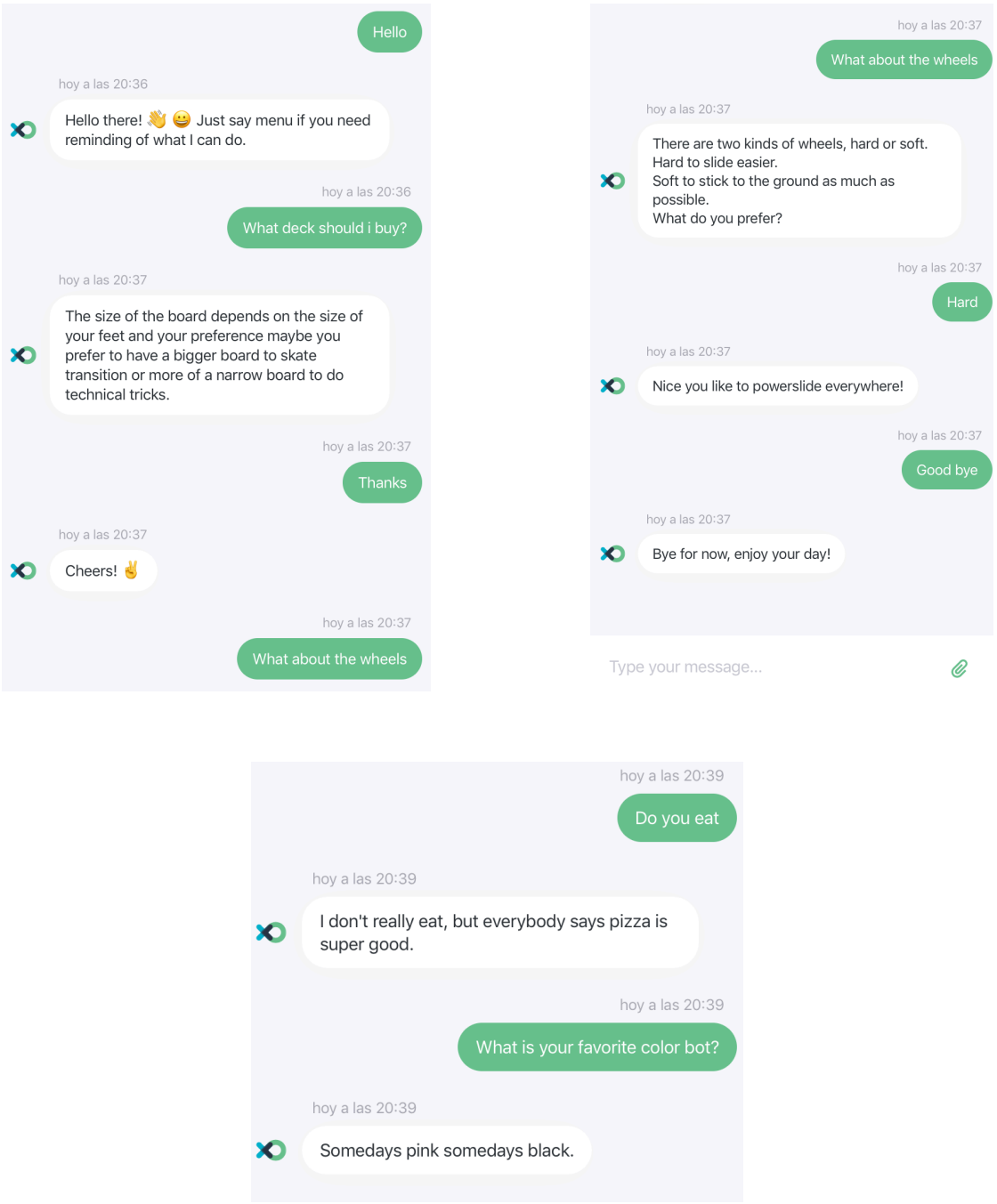

Figura 52 Muestra del chat en funcionamiento en la plataforma FLOW XO.

La conclusión al usar esta plataforma para crear chatbots es que es muy sencilla e intuitiva, pero se puede quedar corta si se quiere un bot más complejo que pueda realizar más funciones o llamar a otras aplicaciones, ya que es un entorno cerrado. Es esta una herramienta perfecta para chatbots de preguntas frecuentes en páginas web de pequeñas compañías o para realizar un chatbot con funciones cerradas y caminos de preguntas y respuestas específicos y esperados, lo que se le denomina “Happy Path”, camino feliz.

6. CONCLUSIÓN.

El Trabajo Fin de Grado muestra los conocimientos y las capacidades obtenidas durante la titulación para poder enfrentarse a un problema o a una especificación real, superando los obstáculos y alcanzando los objetivos marcados. En esta línea, el presente proyecto ha satisfecho los requisitos y espectativas creadas al inicio del mismo, alcanzando los objetivos propuestos.

El germen del proyecto nació durante una conversación entre dos amigos, comentándole el uno al otro que quería aprender a montar chatbots para crear un negocio. El negocio consistía en desarrollar asistentes para empresas que tuvieran página web y/o aplicación móvil. El chatbot podría ser un asistente para búsqueda de información en la página web de la empresa o un bot de preguntas frecuentes, uno de los chatbots más sencillos. De esta manera, los potenciales clientes no tendrían que explorar toda la página web para encontrar lo que desean y, simplemente conversando con el chatbot, podrían encontrar lo que buscan al tiempo que mejoran su impresión y satisfacción con la empresa.

De esta conversación surgio una propuesta de proyecto, que fue aceptada por el director del trabajo. La primera tarea que se definió fue estudiar la technología asocia a los chatbots, dado que es un tema que no se estudia en la titulación. Tras varias semanas de lectura de posts en blogs, papers y plataformas de creación de bots no se avanzó de la forma esperada. Por esta razón, la siguiente tarea fue empezar a escribir un estado del arte que permitiese identificar los principales sistemas de desarrollo. A partir de este estudio se seleccionó FLOW XO, una plataforma online que no necesitaba conocimiento alguno de programación para realizar un chabot que pudiese ser integrado en cualquier página web.

Se creó un primer chatbot en FLOW XO con muchas ganas, pero se descubrió que la plataforma tenía muchas limitaciones. La estrategia de desarrollo era muy específica y guiada por lo que daba poco juego. Por ello se volvió a analizar el estado del arte, con la experiencia adquirida con FLOW XO.

Este trabajo permitió identificar una plataforma muy prometedora, Rasa Stack. Esta tecnología de código abierto utiliza técnicas de Machine Learning para crear el chatbot. Al ser código abierto, se dispone de mayor libertad de desarrollo, pudiendo incluso modificar el propio entorno si se piensa que algo funcionaria mejor de otra manera.

En este punto es donde cogió forma el presente trabajo. Se ha realizado un estudio a fondo de Rasa, para aprender a crear chatbots con su tecnología. No todo fueron cosas buenas con Rasa, ya que hubo que dedicar mucho tiempo a la resolución de problemas de instalación y puesta en funcionamiento. Estos problemas permitieron adquirir experiencia en el uso de foros (como StackOverflow) y sistemas de código abierto (como GitHub). A finales de mayo de este año, Rasa nos sorprendió lanzando la primera versión de Rasa X, que sirvió para ampliar este trabajo.

Gracias al entorno Rasa pudimos desarrollar un chatbot completo, que puede ser utilizado para una aplicación real: asesorar a los clientes en la compra de patinetes. Por último, y con el objetivo de ayudar a otras personas en el desarrallo de chatbots, se ha integrado la experiencia adquirida en una tutoria que facilita el desarrollo de asistentes con esta tecnología.

Como trabajo futuro se podría mejorar enormemente el chatbot presentado o crear otro para una aplicación más compleja. El futuro de los chatbots es positivio y Rasa es una gran herramienta para desarrolar este.

B I B L I O G R A F Í A .

Este trabajo ha sido posible, debido a la búsqueda en internet unicanmente de información. De leer varios papers, tabajos de finde de grado y tesis, los links se encuntran debajo. Y post en blogs, y la documentación de Rasa y videos de youtube hablando sobre los chatbots, desde demostraciones y presentaciones del producto hasta pequeños tutoriales.

PAPERS, TRABAJOS DE FIN DE GRADO, Y TESIS:

Integración de un Chatbot como habilidad de un Robot Social con gestor de diálogos - Juan Carlos Cobos Torres.
http://repositorio.educacionsuperior.gob.ec/handle/28000/1201

Comportamiento Adaptable de Chatbots Dependiente del Contexto - Juan Manuel Rodríguez , Hernán Merlino, Enrique Fernández.
http://revistas.unla.edu.ar/software/article/view/82

CHATBOT: Architecture, Design, & Development. J Chan - University of Pennsylvania School of Engineering and Applied Science
https://s3.amazonaws.com/academia.edu.documents/57035006/CHATBOT_thesis_final.pdf?response-content-disposition=inline%3B%20filename%3DCHATBOT_Architecture_Design_and_Developm.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20190702%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190702T093743Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=3a5dfff76e7ac25406f83d83a60ef4d55c476e3bf2da7a1ae984abcd8dd9518b

Estudio y uso de chatbots para el aprendizaje de inglés. TFG- Sergio Rodríguez Roa
http://oa.upm.es/43327/

Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja. - Toledo Cambizaca, Alcides Francisco.
http://dspace.utpl.edu.ec/handle/20.500.11962/21874

Desarrollo de habilidades cognitivas con aprendizaje móvil: un estudio de casos. - A.I. Ramos, J.A. Herrera y M.S. Ramírez.
https://dialnet.unirioja.es/servlet/articulo?codigo=3167104

LINKS INTERESANTES CON LOS QUE EMPECE A INVESTIGAR Y APRENDER DE ESTE MUNDO DE LOS CHATBOTS:

Chabot creado con Python sobre desastres naturales:
https://planetachatbot.com/disatbot-como-construi-chatbot-telegram-y-python-a67fab1759db

Chatbot con Python:
https://apps.worldwritable.com/tutorials/chatbot/

Librería NLPK toolkit para pyhotn para usar NLP (Natural Language Processor):
https://likegeeks.com/es/tutorial-de-nlp-con-python-nltk/

Natural languge pipline for chatbots:
https://medium.com/@surmenok/natural-language-pipeline-for-chatbots-897bda41482

Python chatbot, Input voice output text:
https://www.youtube.com/watch?v=GjRmy_x6Mn0

Chatscript, open source on C and C++ to create dialogs.
https://medium.freecodecamp.org/chatscript-for-beginners-chatbots-developers-c58bb591da8

Marketing Chatbots:
https://www.youtube.com/watch?v=PXJtFc8DjsE

LINKS DE RASA: Pongo algun ejemplo pero he leido casi al completo la documention.

Algorithms alone won't solve conversational AI - Introducing Rasa X:
https://blog.rasa.com/algorithms-alone-wont-solve-conversational-ai-introducing-rasa-x/

A New Approach to Conversational Software
https://medium.com/rasa-blog/a-new-approach-to-conversational-software-2e64a5d05f2a

Rasa Docs:
https://rasa.com/docs/

Getting started:
https://rasa.com/docs/getting-started/

Installation:
https://rasa.com/docs/rasa/user-guide/installation/

Architecture:
https://rasa.com/docs/rasa/user-guide/architecture/

GitHub Rasa:
https://github.com/rasaHQ/
Deprecating the state machine: building conversational AI with the Rasa stack - Justina Petraitytė
https://www.youtube.com/watch?v=3qgWQ-u1lQo&t=407s

Conversational AI with Rasa Core & NLU - Tom Bocklisch
https://www.youtube.com/watch?v=zRqjH7fT0G0

Building a chatbot with Rasa NLU and Rasa Core
https://www.youtube.com/watch?v=xu6D_vLP5vY&t=648s

Rasa Summit 2018!
https://www.youtube.com/watch?v=DXuhp9kpMwM&t=1330s

Getting started with Rasa: using the Rasa Stack starter-pack
https://www.youtube.com/watch?v=lQZ_x0LRUbI&t=307s

Rasa X Product Announcement
https://www.youtube.com/watch?v=VXvWdrr2yw8&feature=youtu.be

Virtual Enviroment to get Rasa Working
https://github.com/RasaHQ/starter-pack-rasa-stack/issues/56

Flow XO:
https://flowxo.com