UNIVERSIDAD NACIONAL DE ASUNCIÓN

Facultad de Ingeniería
Ingeniería Electrónica

Implementación de un ChatBot utilizando algoritmos de Inteligencia Artificial.

Carlos Buenaventura Ozuna Loncharich

Oscar Bartolome Valdez Sarubbi

San Lorenzo - Paraguay

2023

Miembros del Consejo Directivo
Consejeros Titulares

Prof. Ing. Cirilo Jorge Hernáez Medina (Decano)

Prof. Ing. Rubén Dario Zárate Rojas (Vice-Decana)

Prof. Ing. Amílcar Gaspar Troche Escobar (Docente)

Prof. Ing. Ever Romildo Cabrera Herebia (Docente)

Prof. Ing. Luis Alberto Cardozo Villanueva (Docente)

Prof. Ing. Roberto Arturo Nagy (Docente)

Prof. Ing. Ramón Eugenio Pistilli Scorzara (Docente – C.S.U.)

Ing. Ignacio Velázquez Guachiré (Egresado no docente)

Ing. Carlyle Adolfo Alvarenga González (Egresado no docente)

Hugo Javier Álvarez Ferrerio (Estudiante)

Rodrigo González Jara (Estudiante)

Diego Alberto Martínez Aguilar (Estudiante)

Consejeros Suplentes

Prof. Ing. Gabriel Enrique Fleitas Ferrari (Docente)

Prof. Ing. Roberto Nagy Benítez (Docente)

Prof. Ing. Rubén Darío Zárate Rojas (Docente-CSU)

Ing. Adolfo Esteban Artunduaga Alarcón (Egresado no docente)

Ing. Karina Lorena Ruiz Franco (Egresada no docente)

Rocío María Frutos Echauri (Estudiante)

Ninoska Natalia Candia Villamayor (Estudiante)

José Manuel Benítez Medina (Estudiante)

A quien se dedique

Agradecimientos

A quien se tenga que agradecer

Índice general

1. Introducción **6**

1.1. Objetivo General 7

1.2. Objetivos Específicos 7

1.3. Alcance y limitaciones 7

2. Revisión de literatura **8**

2.0.1. Reseña Histórica 8

2.1. Procesamiento de lenguaje natural 9

2.1.1. Lenguaje 10

2.1.2. Métodos Heurísticos y basado en reglas 11

2.1.3. Métodos basados en Aprendizaje Automático (Machine Learning) 11

2.1.4. Aprendizaje Profundo 12

2.1.5. Redes Neuronales Recurrentes(RNN) 12

2.1.6. Redes Neuronales Convencionales(CNN) 13

2.1.7. Tranformadores 14

2.1.8. Autoencoders 15

2.2. Chatbots 16

2.2.1. Concepto 16

2.2.2. Características 16

2.2.3. Aplicaciones 17

2.2.4. Arquitectura 17

2.2.5. Plataformas de desarrollo 18

3. RASA **22**

3.1. Conceptos Básicos 22

3.2. Cómo se lleva a cabo las conversaciones? 23
3.3. Instalación de RASA 24
3.4. Creación de un Proyecto 24
3.4.1. Archivo nlu 24
3.4.2. Archivo Rules 25
3.4.3. Archivo Stories 26
3.4.4. Archivo config 26
3.4.5. Archivo credentials 26
3.4.6. Archivo domain 27
3.4.7. Archivo endpoints 27
3.5. Componentes 27
3.5.1. Tokenizadores 27
3.5.2. Caracterizadores 28
3.5.3. Clasificadores de Intención(Intent Classiffiers) 29
3.5.4. Extracción de entidades 30
3.5.5. Selectores 31
3.5.6. Predicción de acciones 31
3.6. Patrones de Conversación 33
3.6.1. Chitchat y FAQs 33
3.6.2. Fallbacks 34
3.7. Pruebas 34
3.7.1. Validación de datos 34
3.7.2. Evaluación del desempeño de la NLU 35
3.8. Despliegue del sistema 35
3.8.1. Arquitectura 36

4. IMPLEMENTACION **39**

Índice de figuras

2.1. Aplicaciones de partes de un Lenguaje 10
2.2. Aplicaciones de partes de un Lenguaje 13
2.3. Diagrama de CNN 14
2.4. Diagrama de un Transformador 15
2.5. Representacion de un autoencoder 16
2.6. Arquitectura de los Chatbots 18

3.1. Estrucutra de los archivos 25
3.2. Componentes de un esquema NLU 28
3.3. Tokenización 28
3.4. Caracterización 29
3.5. Clasificación de intenciones y entidades 30
3.6. Extractor Regex 31
3.7. Predicción de acciones. 32
3.8. Arquitectura 36

Índice de tablas

3.1. Caracterizadores. Elaboración Propia 29
3.2. Clasificadores de intenciones. Elaboracion propia 30
3.3. Extractores de entidades. Elaboración propia 30

Índice de anexos

Anexo A: Comandos de Rasa 43
Anexo B: El uso de plantillas 44

CAPÍTULO 1

Introducción

Durante los últimos años, la Inteligencia Artificial (IA) ha tenido un crecimiento exponencial, esto se debe principalmente a la capacidad de cómputo asequible y de alto rendimiento, además de los grandes volúmenes de datos que se encuentran disponibles. [1]

La IA tiene varias áreas de estudio, entre ellas se encuentra el Procesamiento de Lenguaje Natural (NLP, por sus siglas en inglés), encargado de hacer que las computadoras entiendan e interpreten el lenguaje humano con la ayuda de modelos estadísticos, machine learning y deep learning.

Una de las tantas apĺicaciones prácticas del NLP son los chatbots, programas que imitan la conversación humana.Los chatbots son capaces de interactuar con personas y de responder adecuadamente a sus preguntas, son bastante accesibles, eficientes y de alta disponibilidad, permitiendo así que distintas industrias se beneficien de el, entre las que destacan el comercio electrónico, los seguros y el cuidado de la salud . [2]

En este trabajo se presenta un chatbot para el sector de la educación, donde el estudiante pueda realizar sus consultas académicas y la respuesta deberá ser generada de acuerdo a ciertas técnicas de coincidencia de patrones.

1.1. Objetivo General

Desarrollar e implementar un Chatbot utilizando algoritmos de Inteligencia Artificial (IA).

1.2. Objetivos Específicos

- Comparar distintas Tecnologías para abordar el problema.
- Orientar el chatbot a dudas comunes de los estudiantes de la Facultad de Ingeniería en atención al alumno.
- Recopilar, procesar y filtrar preguntas frecuentes para contar con un dataset propio.
- Seleccionar, entrenar y probar el algoritmo utilizando el dataset generado.
- Implementar el chatbot para su uso por estudiantes de la Facultad de Ingeniería
- Agregar una interfaz, propia o a una ya existente para el uso por los alumnos.

1.3. Alcance y limitaciones

Se pretende desarrollar un Chatbot que utilice inteligencia artificial para responder a preguntas frecuentes de los alumnos de la Facultad de Ingeniería de la UNA. El entrenamiento de la IA se llevara a cabo utilizando un dataset propio en conjunto con otros ya existentes. Una previa comparativa entre distintas librerías y plataformas es necesaria para escoger la mejor solución al objetivo planteado.

CAPÍTULO 2

Marco Teórico

2.0.1. Reseña Histórica

Para empezar a adentrarnos a los conceptos del Procesamiento de Lenguaje Natural primero empezaremos a repasar hitos importantes en áreas de conocimiento afines. Los primera aplicación reconocida como una NPL fueron en 1948 que fue un buscador de palabras en el diccionario de desarrollado en Birkbeck College de Londres por Warren Weaver. Luego rápidamente surgió como idea de investigación crear una maquina de traducción automática en varios grupos de en Estados Unidos, Reino Unido, Francia y Rusia. Los primeros grupos se concentraron en traducir texto en Alemán, cuando los textos de la Segunda Guerra Mundial se volvieron obsoletos, pasaron a el Ruso el mayor problema fue que no existían todavía fundamentos formales de al computación, ni mucho menos de NLP. La mayoría de los investigadores eran matemáticos e inmigrantes bilingües que intentaban encontrar relaciones entre ambos lenguajes. La investigación concluyo en que todavía no existían los recursos necesarios para abordar dicho problema, pero se desarrollaron mejoras en herramientas para la traducción asistida. [3] En 1950 Alan Turing desarrolla el test de Turing que es un test para distinguir el nivel de inteligencia de una maquina, propuso que un humano evaluara las conversaciones en un lenguaje natural entre un humano y una máquina diseñada para dar respuestas similares a los humanos. Lo que pudo definir

la inteligencia de un sistema de computo de una forma comparable a la inteligencia humana. En el año 1954 un experimentó de colaboración de la Georgetown University e IBM se hace publica la primera demostración logra la traducción automática de mas de 60 se oraciones en Ruso al Ingles. Las oraciones eran previamente elegidas que trataban de temas políticos, legales, matemáticos y científicos. Luego eran ingresados a la maquina escritos en Ruso con letras romanizadas. El método era principalmente lexicográfico basado en un diccionario de relaciones de palabras de ambos idiomas. [4] En las décadas posteriores se siguieron dando avances con métodos basados en reglas algorítmicas como arboles de decisión para el procesamiento del lenguaje y la traducción automática. A partir de los 2000s la investigación se centro en algoritmos de Machine Learning no supervisados y semi supervisados, por amplia disponibilidad de información no clasificados en literatura e internet, por lo general estos métodos son menos eficientes que los algoritmos supervisados pero la cantidad de datos existentes pueden equiparar estas deficiencias. En la segunda década del nuevo milenio se utilizaron nuevos métodos basados en Aprendizaje de características y Deep Learning lo que nos trae al estado actual del arte que discutiremos en la siguiente sección.

2.1. Procesamiento de lenguaje natural (NPL)

El procesamiento de lenguaje natural o por sus siglas en inglés NPL(de natural language processing) es un campo de las Ciencias de Computación y la Lingüística que trata con métodos para analizar, modelar y entender el lenguaje humano.

2.1.1. Lenguaje

Primero de los conceptos claves para poder adentrarnos a NLP es definir el Lenguaje, un Lenguaje es un sistema de comunicación que involucra una combinación compleja de componentes como, letras, palabras, etc. La Lingüística es el estudio sistemático del lenguaje. Para poder estudiar NLP, es importante entender componentes del lenguaje. [5]

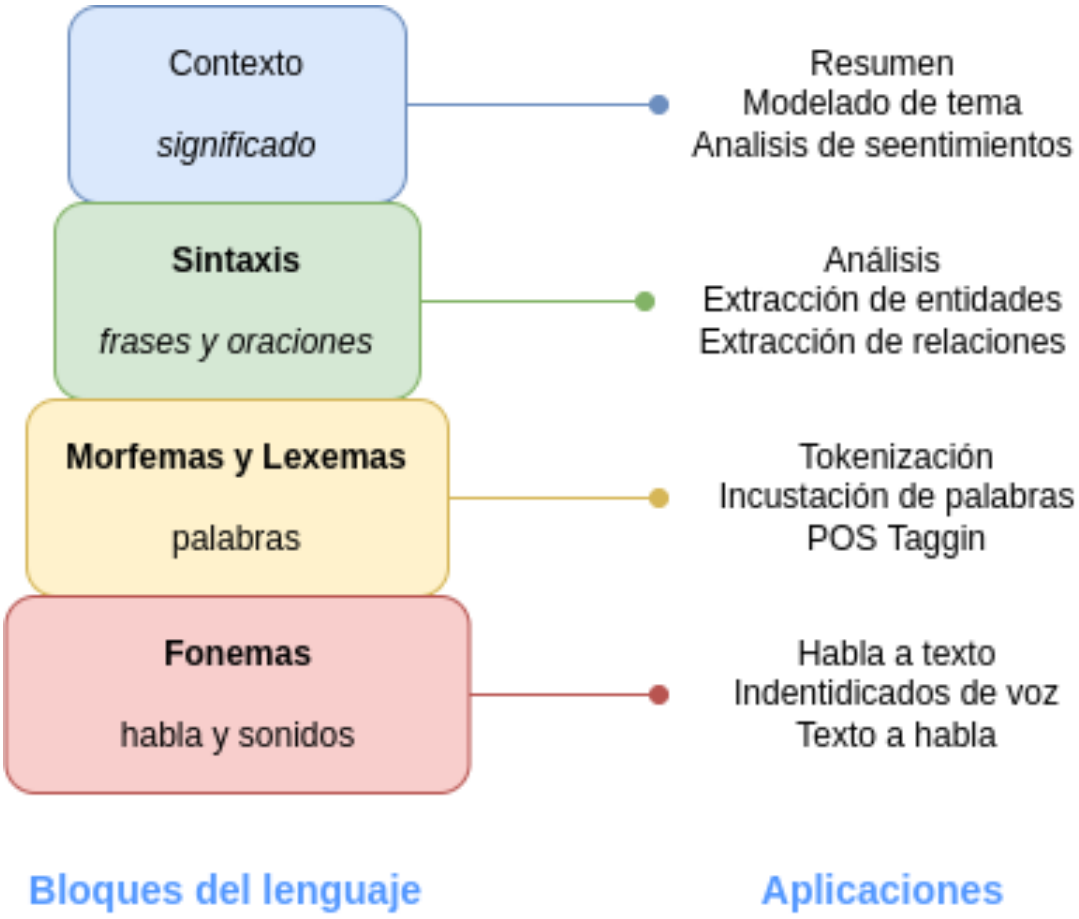

Figura 2.1: Aplicaciones de partes de un Lenguaje [5]

2.1.2. Métodos Heurísticos y basado en reglas

Similarmente a otros métodos primitivos usando AI, los primeros intentos en el diseño de sistemas de NLP fueron en construir reglas de acciones a manualmente por medio de arboles de decisión. Esto requiere que lo desarrolladores tengan experiencia en el dominio del problema para formular las reglas que puedan ser incorporadas al programa.

En estos sistemas también se requieren recursos como diccionarios y tesauros, típicamente compilados y digitalizado. Otra herramienta muy poderosa en la implementación de sistemas basados en reglas son las Expresiones Regulares(Regex por sus siglas en ingles). Una expresión regular es un conjunto de caracteres o patrón que es utilizada para coincidir y encontrar subcadenas en un texto como `^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$` que es utilizado encontrar direcciones de email validas en un texto.

Las reglas y heuristica juegan un rol importande en todo el ciclo de vida de proyectos de NLP incluso hoy en dia. En un por un lado, estos son una buena manera de desarrollar primeras versiones. O también pueden ser de suma importancia en sistemas basados en AI para llenar vacíos o limitaciones de los modelos probabilísticos. [5]

2.1.3. Métodos basados en Aprendizaje Automático (Machine Learning)

El Aprendizaje Automático o ML(por sus siglas en ingles) son aplicados para datos de texto así como también son utilizados en otro tipos de datos, como imágenes, voz y datos estructurados. Métodos de aprendizaje supervisados como clasificación y regresión con usados con mucha frecuencia en NLP. Como por ejemplo para buenas aplicaciones son par clasificar temas de artículos en el caso de calificadores.

Por otro lado las técnicas de regresión suelen ser utilizadas para dar una predicción numérica, como por ejemplo el precio de un stock basado en discusiones en una red social.Similarmente métodos no supervisados pueden ser útiles para agrupar documentos similares.

2.1.4. Aprendizaje Profundo

Aprendizaje profundo o Deep learning(DL) en ingles, es una evolución mas compleja de los algoritmos de ML convencionales, se trata de combinaciones de nodos que emulan el funcionamiento de las redes neuronales del cerebro, sin entrar en mucho detalle procederemos a analizar los métodos basados en DL mas utilizados y exitosos actualmente en el área de NLP.

2.1.5. Redes Neuronales Recurrentes(RNN)

En todos los lenguajes una oración tiene una dirección de lectura, por ejemplo en el Castellano se lee de izquierda a derecha. Entonces un modelo que puede ser útil para leer progresivamente una entrada de texto podría ser muy útil para NLP. Las Redes Neuronales recurrentes o RNNs por sus siglas del ingles están específicamente diseñadas para mantener un procesamiento secuencial y memoria de los pasos anteriores. Esta memoria es temporal y la información es almacenada y actualizada en cada paso de lectura de la RNN.

Las RNNs son poderosa y funcionan muy bien para resolver muchas tareas de NLP, como clasificación de texto, reconocimiento de entidad, tradición automática, etc. También se pueden utilizar para generar texto, por ejemplo para predecir la siguiente palabra de se va a escribir de acuerdo al contexto de lo que ya se escribió.

A pesar de su versatilidad y capacidad, las RNNs sufren de ciertas limitaciones

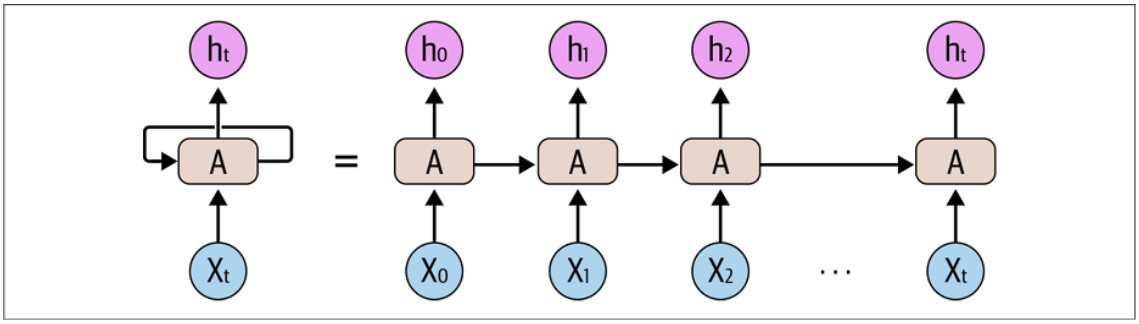

Figura 2.2: Aplicaciones de partes de un Lenguaje
[5]

por contar con memoria temporal, por lo tanto no se desempeñan óptimamente para textos con largos contextos. Pero para estos existen variaciones optimizadas para memoria de largo plazo(LSTMs).

2.1.6. Redes Neuronales Convencionales(CNN)

Las CNNs por sus siglas en Ingles o Redes Neuronales Convoluciones son muy populares y muy frecuentemente usadas para para aplicaciones como clasificación de imágenes, reconocimiento de vídeo, etc. Las CNNs también vieron éxito en NLP, específicamente en clasificación de texto. Se puede reemplazar una palabra de una oración de un texto por un vector palabra, y estos a su vez ser colocados en una matriz para poder ser tratados de manera similar a una imagen.

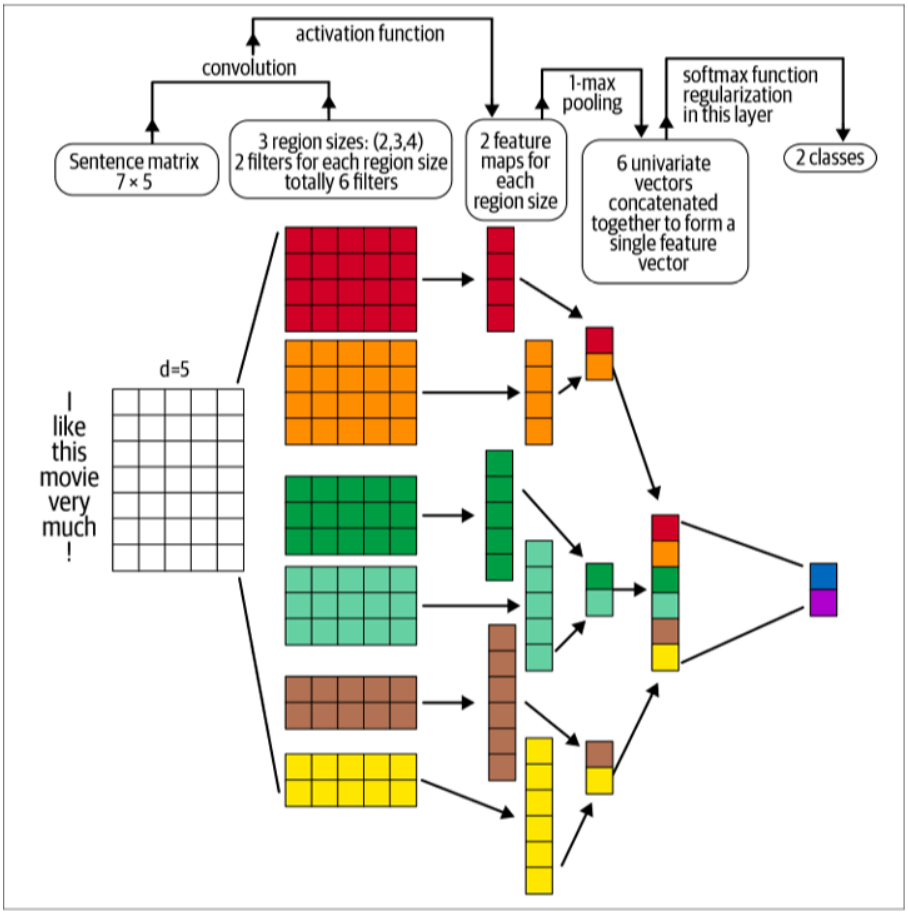

Figura 2.3: Diagrama de CNN
[5]

2.1.7. Tranformadores

Los trasformadores son la ultima innovación en lo que respecta a modelos basados en DL para NLP. Los modelos de transformadores obtuvieron avances en la mayoría de las tareas de NLP en los últimos años.

Estos modelan el contexto textual pero no de manera secuencial. Dada una palabra en una entrada de texto, se prefiere mirar las demás palabras vecinas en el texto y representar cada palabra de acuerdo a el contexto de las demás. Por ejemplo si el contexto habla de finanzas, entonces "banco"probablemente representaría una institución financiera. Por otro lado en el contexto de un rió, estaría relacionado con un montículo de tierra. Recientemente, los transformadores

han sido utilizados para transferir aprendizaje con flujos de pequeñas tareas, La transferencia de aprendizaje es una técnica en IA en la cual lo que se aprendió en resolver un problema es aplicado para resolver problemas similares.

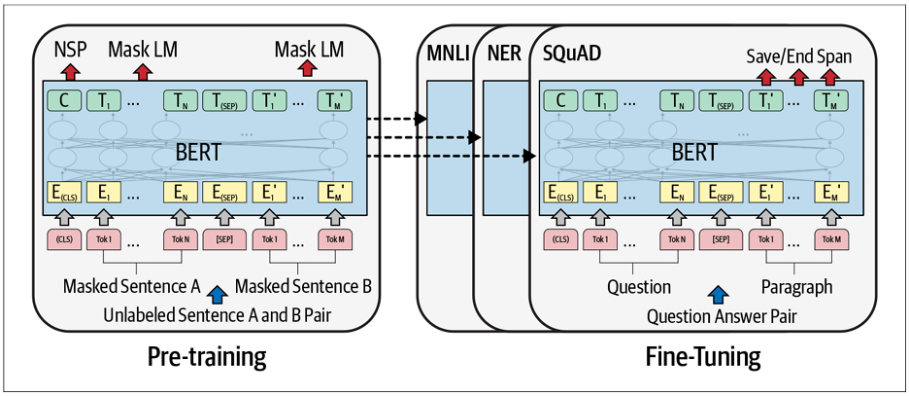

Figura 2.4: Diagrama de un Transformador
[5]

2.1.8. Autoencoders

Los autoencoders son otro tipo de red neuronal utilizadas principalmente para aprender a representar la entrada de forma comprimida en un vector. Por ejemplo, si queremos representar unas palabras en un vector, podemos aprender a mapear el texto en vectores y luego remapear para reconstruir la entrada original. Esto es una forma de aprendizaje no supervisado ya que no se necesitan datos anotados para el entrenamiento.

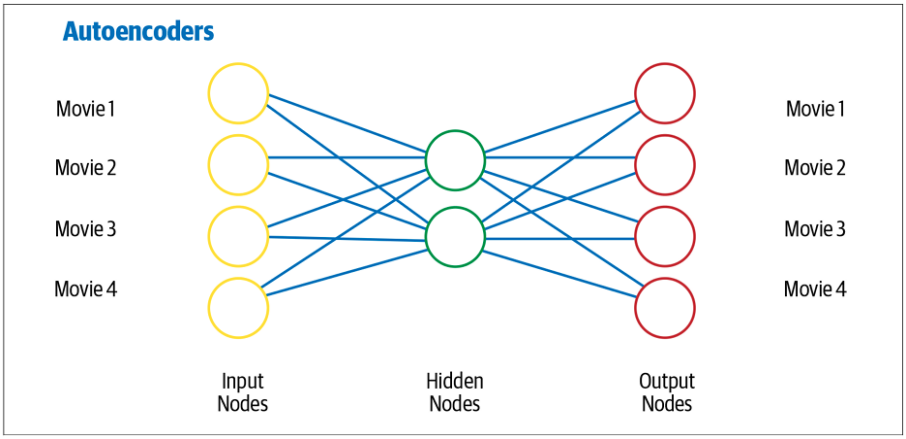

Figura 2.5: Representacion de un autoencoder
[5]

2.2. Chatbots

2.2.1. Concepto

Los chatbots son programas que imitan la conversación humana utilizando la Inteligencia Artificial. [6]

2.2.2. Características

Existen diferentes tipos de chatbots, clasificados según su complejidad, objetivos o funciones, pero todos ellos cuentan con las siguientes cualidades según Nicol Radziwill y Morgan Benton [7]

- Rendimiento: se refiere a la eficiencia en la asignación de funciones y a la robustez que tienen en cuanto a la manipulación y a las entradas inesperadas.
- Funcionalidad: es capaz de interpretar, responder y ejecutar correctamente las tareas demandadas.

- Humanidad: la conversación con el chatbot debe ser natural, lo mas parecida a la humana.
- Ética: genera confianza, respeta y protege la dignidad, y la privacidad de los usuarios.
- Accesibilidad: se encuentra disponible cuando el usuario quiera usarlo, además se refiere a que es capaz de detectar intenciones y significados

2.2.3. Aplicaciones

Se pueden mencionar dos tipos de aplicaciones:

- Asistentes Personales Virtuales: ofrecen servicios a los usuarios a través de texto o voz. Ejemplos: Siri(Apple), Google Assistant, Alexa (Amazon) y Cortana (Microsoft)
- Bots para el consumo específico: sus aplicaciones son muy variadas, puede utilizarse en el transporte, la salud, el clima, entretenimiento o incluso en la educación.

2.2.4. Arquitectura

Para el diseño adecuado de cualquier sistema la mejor solución es dividirla en varias partes o subsistemas de acuerdo a un estándar. En la figura 2.6 se puede ver que existen dos bloques bien definidos, al primero se lo llama 'lado del cliente' que es la parte que interactúa con los usuarios; y el segundo bloque es el 'lado del servidor' encargado de procesar las peticiones del cliente.

El proceso inicia en la interfaz de usuario cuando el cliente realiza una solicitud, ésta es analizada en el bloque componente de comprensión de lenguaje, aquí se extrae toda la información necesaria y se deduce las intenciones del usuario.

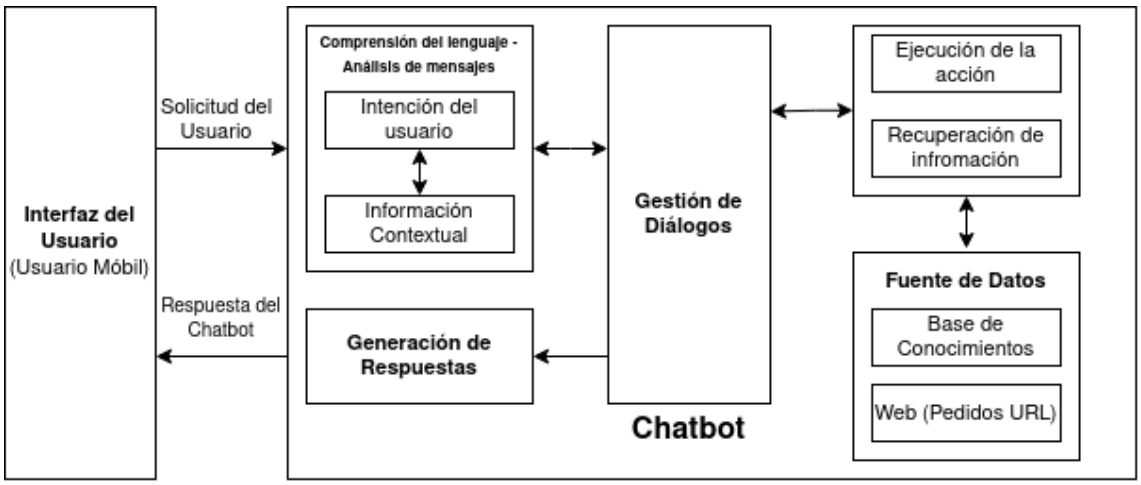

Figura 2.6: Arquitectura de los Chatbots

Una vez que el chatbot llegue a la mejor interpretación que puede, ejecuta las acciones solicitadas o recupera la información de su fuente de datos, que puede ser una base de datos propia o datos externos accedidos a través de APIs.
Posteriormente se generan las respuestas lo más parecidas a las que daría una persona humana, para ello utiliza la información de intención y contexto proveída por el componente de análisis de mensajes del usuario.
El componente de gestión de diálogo se encarga de solicitar información faltante, aclaraciones y hacer preguntas de seguimiento. [8]

2.2.5. Plataformas de desarrollo

- DialogFlow: Es la plataforma de desarrollo de chatbots de Google, permite una fácil integración a aplicaciones móviles y web, también facilita bastante el diseño de la interfaz de usuario.

  Admite como entrada texto y voz, es capaz de responder a los clientes con texto o voz sintética.

  Existen dos versiones, Dialogflw CX utilizado para agentes grandes o muy avanzados y Dialogflow ES que es la versión estándar, ésta cuenta con una versión gratuita.

El precio varia según la versión elegida y el tipo de entrada, Dialogflow ES cobra 0.002 USD por cada solicitud realizada por texto y 0.0065 USD si la entrada es un audio de hasta 15 segundos. [9]

- IBM Watson: IBM Watson permite a los usuarios integrar sus chatbots en cualquier canal, sea web, aplicaciones o incluso una llamada.

  Es capaz de aprender los vocabularios de la industria, términos coloquiales o dialectos regionales, admite entradas de voz y también de texto.

  Está diseñada para aprender sobre el camino, proporciona herramientas para detectar las tendencias y estas ayudan a asignar recursos de forma mas eficiente y eficaz.

  No es necesario escribir ni una sola linea de código ya que utiliza un entorno de 'arrastre y suelte' para construir los diálogos y una vez que lo adaptemos a nuestras necesidades, fácilmente lo incorporamos a la app con 'copiar y pegar'. [10]

  Cuenta con tres versiones: la versión mas básica, Lite, es gratuita pero muy limitada en sus funcionalidades. Luego vienen las versiones de paga, Plus y Enterprise con precios que van desde los 140 USD por mes. [11]

- Amazon Lex: Amazon Lex es la propuesta del gigante tecnológico Amazon, sirve para diseñar, crear, probar e implementar interfaces de conversación en las aplicaciones.

  Reconoce tanto entradas de texto como de voz, gestiona el contexto de las conversaciones de forma nativa y también permite una gran fidelidad en las interacciones de habla telefónica.

  La integración con plataformas se realiza de forma muy sencilla desde la consola de Amazon Lex , permite aplicaciones web, móviles y los servicios propios de Amazon como Amazon Kendra, Amazon Polly o AWS Lambda.

  Los precios varían según el tipo de servicio solicitado, El mas básico 'In-

teracción de respuesta y solicitud' cobra 0,004 USD por solicitud de voz y 0,00075 USD por solicitud de texto. [12]

- RASA Open Source: Es una plataforma de código abierto que proporciona procesamiento de lenguaje natural para convertir los mensajes de los usuarios en intenciones y entidades que los chatbots entienden, permite la gestión de los diálogos basándose en los mensajes de los usuarios y el contexto de la conversación.

  La integración a las aplicaciones mas comunes de mensajes y a los canales de voz se puede hacer de forma muy sencilla con los conectores ya incorporados, para conectar a las demás aplicaciones móviles o web se deben personalizar los conectores.

  RASA también tiene una versión de paga denominada RASA Enterprise, utilizada para el despliegue a escala, su precio varía de acuerdo a las necesidades de la organización y del proyecto. [13]

- Chatterbot: Chatterbot es una librería de Python que facilita la automatización de respuestas mediante distintos algoritmos de aprendizaje automático, esto permite que una instancia de agente mejore su propio conocimiento de las posibles respuestas a medida que interactúa con humanos y otras fuentes de datos informativos

  Cada vez que un usuario introduce una frase, la librería guarda el texto que ha introducido y el texto al que responde la frase. A medida que ChatterBot recibe más entradas, aumenta el número de respuestas que puede dar y la precisión de cada respuesta en relación con la declaración introducida.

  Se puede integrar a las aplicaciones mediante APIs, además ChatterBot tiene soporte directo para la integración con el ORM de Django, esto facilita bastante para crear las páginas conversacionales. [14]

Haciendo una comparación entre todas las herramientas analizadas, vemos que la creación de los chatbots con Dialogflow, IBM Watson y Amazon Lex es mucho mas sencilla, ya que tienen una madurez tecnológica muy alta, gran soporte y las interfaces son muy intuitivas, además no requieren de mucha programación. La mayor desventaja es que las versiones que cuentan con todas las funcionalidades son de paga, y las versiones gratuitas son muy limitadas y básicas, por lo que descartamos estas tres opciones.

En cuanto a Chatterbot y RASA, la curva de aprendizaje puede ser un poco mayor porque no utilizan interfaces gráficas, todo es programado con Python, un lenguaje muy versátil que cuenta con numerosas librerias, esto permite tener mayor control sobre los chatbots porque se pueden manipular todos los ficheros y modificar las configuraciones, Si bien ambos están muy bien documentados, RASA destaca de Chatterbot por su comunidad, cuenta incluso con un foro donde participan los desarrolladores y usuarios dispuestos a brindar ayuda a todo aquel que las necesite.

Teniendo en cuenta que no contamos con experiencia en el desarrollo de chatbots, se valora de sobremanera la comunidad, documentación y tutoriales con los que cuenta RASA, además que permite la integración con distintas plataformas y el lenguaje de programación que utiliza (Python) es de nuestro conocimiento, es por eso que concluimos con que RASA es una buena elección para llevar a cabo este proyecto.

CAPÍTULO 3

RASA Open Source

Rasa es un framework que permite la construcción de forma sencilla de chatbots personalizados, está compuesta de dos librerias de código abierto, Rasa NLU y Rasa Core, denominadas en conjunto Rasa Stack.

3.1. Conceptos Básicos

- **intents (intenciones):** son las categorías, denominadas utterances creadas para lo que el usuario está tratando de transmitir o lograr en una conversación, por ejemplo 'saludos' donde se especifican las distintas formas de saludar. Las intenciones pueden ser divididas en pequeñas subintenciones denominadas 'Retrieval Intent'.
- **entities (entidades):** las entidades son informaciones o palabras clave que pueden ser extraídas de un mensaje para personalizar la conversación.
- **slots:** es un registro de datos que Rasa utiliza para guardar la información proveída por el usuario en el curso de la conversación, un claro ejemplo del uso de este elemento es almacenar el nombre del usuario para personalizar los mensajes.
- **responses (respuestas):** mensajes que los chatbots envían a los usuarios, estos pueden ser dinámicos y con cualquier tipo de contenido como texto,

imágenes, links, etc.

- **forms (formularios):** Un tipo de acción personalizada que pide al usuario varios datos.
- **actions (acciones):** es un paso que toma el bot en la conversación por ejemplo, llamar a una API o enviar una respuesta al usuario. [15]

3.2. Cómo se lleva a cabo las conversaciones?

Para llevar a cabo las conversaciones se utilizan las dos librerías del Rasa Stack.

- **Rasa NLU:** En ella se escriben los archivos de configuración, se elige el pipeline y el modelo de entrenamiento para que deduzca las intenciones y posteriormente pueda extraer las entidades disponibles.

  Puede ser basado en reglas o en redes neuronales, el primero suele ser mas ligero y no necesita de muchos datos aunque no son buenos en tareas antes no vistas, mientras que el segundo necesita de mas capacidad de cómputo y datos para entrenamiento, son mas flexibles que los basados en reglas, ya que pueden aprender cosas que no han visto antes.
- **Rasa Core:** es el gestor de diálogos utilizado para crear modelos que sean capaces de decidir que respuestas o acciones se ejecutarán de acuerdo a las entradas generadas por el usuario.

  También puede ser basado en reglas, que es el enfoque mas tradicional, funciona muy bien en muchos casos pero es difícil de expandir las conversaciones, también puede ser basado en redes neuronales que escoge la siguiente acción basándose en la conversación y en los ejemplos del entrenamiento.

Básicamente, Rasa NLU se encarga de interpretar los mensajes y Rasa Core de

de decidir que acción tomar.

Para asegurarnos de que una conversación funcione Rasa utiliza un proceso denominado ‘conversation-driven development’ que consiste en revisar manualmente las conversaciones para detectar cualquier error cometido, agregar nuevos datos de entrenamiento, volver a entrenar el modelo y probarlo nuevamente [16]

3.3. Instalación

Primeramente crearemos un entorno virtual denominado ’venv’ utilizando Python en una computadora con Linux.

```
python3 -m venv ./venv
```

Luego se activa el entorno virtual.

```
source ./venv/bin/activate
```

Y por último se instala Rasa Open Source utilizando pip (requiere Python 3.7 o 3.8)

```
sudo pip3 install -U –user pip && pip3 install rasa
```

3.4. Creación de un Proyecto

La creación de un nuevo proyecto se realiza con el comando ’rasa init’, éste crea un conjunto de carpetas y archivos así como se muestra en la figura 3.1.

3.4.1. Archivo nlu

En él se encuentran datos estructurados que sirven para entrenar el modelo y luego extraer la información de los mensajes del usuario. Estos datos son las

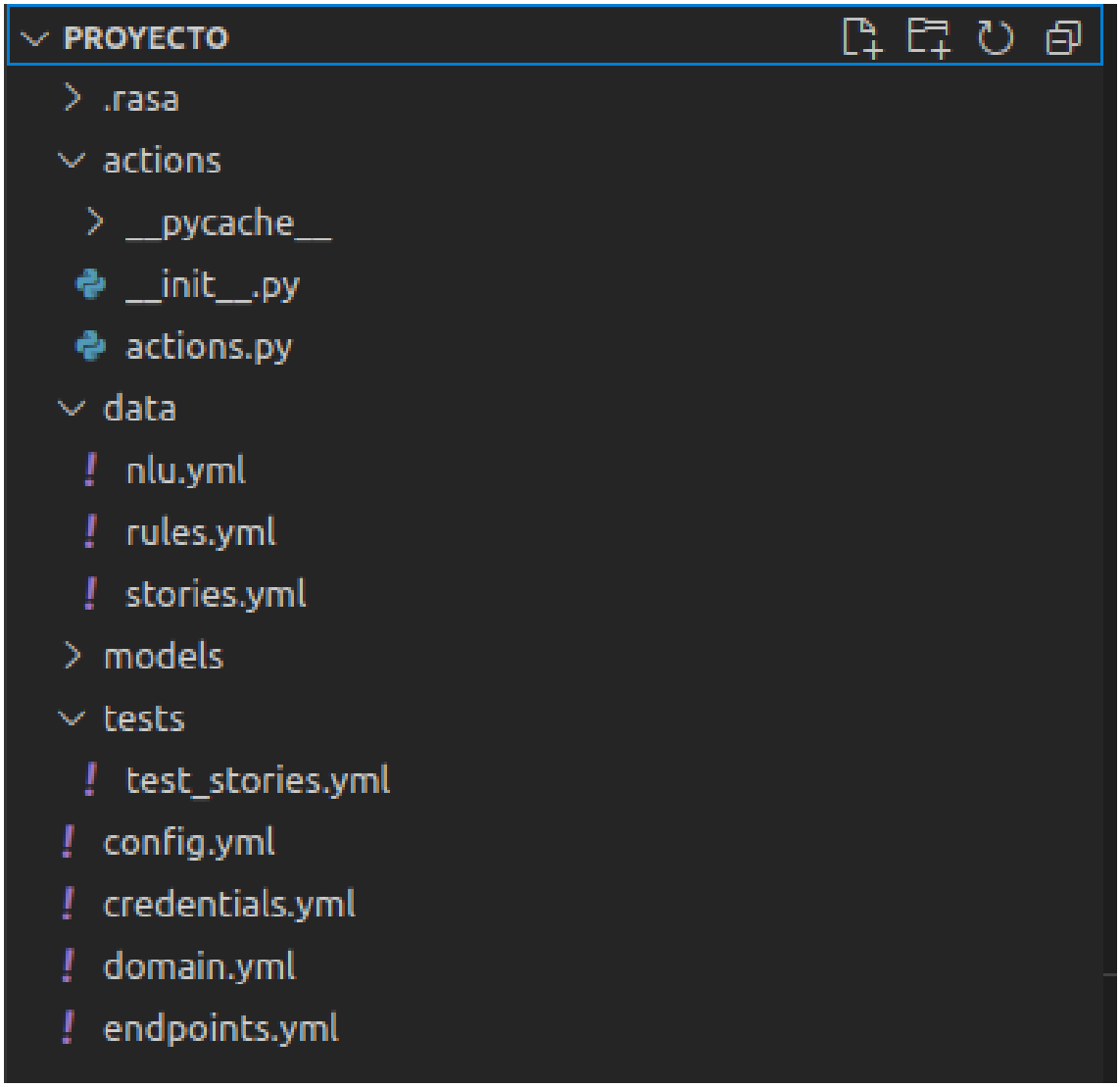

Figura 3.1: Estrucutra de los archivos

intenciones y entidades, también se pueden agregar expresiones regulares y algunas tablas de búsqueda. [17]

3.4.2. Archivo Rules

En este archivo se definen las reglas, que no son mas que tipo de datos de entrenamiento encargados de describir partes de una conversación que siempre sigue el mismo camino. [18]

3.4.3. Archivo Stories

Las historias son un tipo de datos de entrenamiento, se utiliza para entrenar modelos que puedan generalizar las rutas de conversación. Las entradas del usuario son expresadas mediante intents, y entitites si es necesario, mientras que las respuestas del asistente son expresadas mediante actions.
Los patrones que siguen las conversaciones podemos extraer de datos ya existentes o con la herramienta de rasa 'interactive learning'. [19]

3.4.4. Archivo config

En el archivo config se definen el lenguaje y los componentes del pipeline, que forman parte de Rasa NLU y las políticas a ser utilizadas, correspondiente a Rasa NLU.
El pipeline es el encargado de definir la dirección de flujo de datos entre los diferentes componentes, Rasa nos permite configurar cada uno de ellos según nuestras necesidades, de tal forma que podamos realizar las predicciones de las intenciones y la extracción de las entidades. Las políticas forman parte de la gestión de diálogos, encargada de seleccionar la siguiente acción a ser ejecutada. [20]
La configuración del pipeline y las políticas son de suma importancia, por lo que se detallaran sus componentes en la sección 3.5.

3.4.5. Archivo credentials

Aquí se definen los credenciales para las plataformas de voz y chat que el bot utiliza. Rasa cuenta con algunos conectores preestablecidos para los canales mas conocidos como Facebook Messenger, Telegram, Google Hangouts Chat o

una pagina web propia. [21]

3.4.6. Archivo domain

El archivo domain es un archivo de configuración donde se especifican las intenciones, entidades, slots, respuestas, formularios y acciones que el bot debe saber. [22]

3.4.7. Archivo endpoints

Los endpoints son los enlaces a los servicios externos o internos que puede tener Rasa. En el se definen los servidores que corren o en los que están alojadas las acciones personalizadas, al igual que los modelos con los que se cuenta. También es aquí donde se especifican los tracker store, utilizados para guardar las conversaciones, y los event broker, encargados de conectar el bot con otros servicios que procesan los datos que llegan de las conversaciones.

3.5. Componentes

En esta sección estaremos describiendo el funcionamiento de los componentes utilizados en la arquitectura de Rasa, estos componentes son modulares y genéricos para lo que son sistemas de NLU modernos, pueden ser propios del entorno o proveídos por otras librerías de terceros para extender funcionalidades.

3.5.1. Tokenizadores

Antes de poder ser procesada una porción de texto debe ser dividida en porciones mas pequeñas, para esto se suele utilizar un tokenizador (o tokenizer). Este

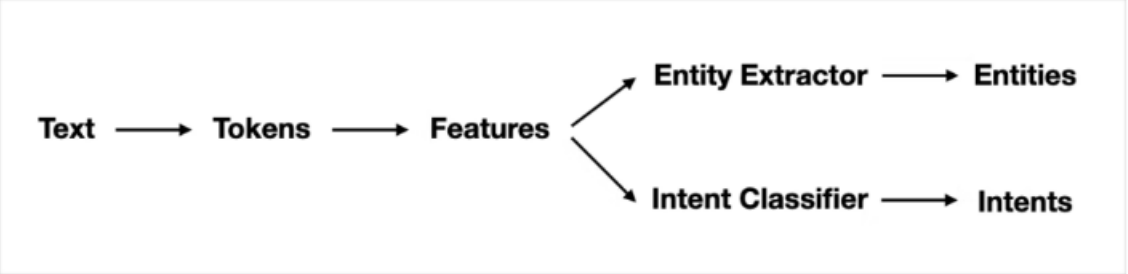

Figura 3.2: Componentes de un esquema NLU

divide el texto en componentes de un vector.

Algunos tokenizadores también agregan información extra a los tokens que pueden ser usados para generar lemas, o sea extraer la palabra que da el significado base a las palabras que pueden ser utilizados por el contador de vectores.

Para el Inglés y el Español usualmente se usa WhiteSpaceTokenizer que separa en tokens cuando se detectan espacios, para los idiomas que no precisen de espacios en blanco para separar las palabras como el Coreano, Japones o Chino, se utilizan MitieTokenizer, en el caso del último tambien es muy frecuente el uso de JiebaTokenizer. [23]

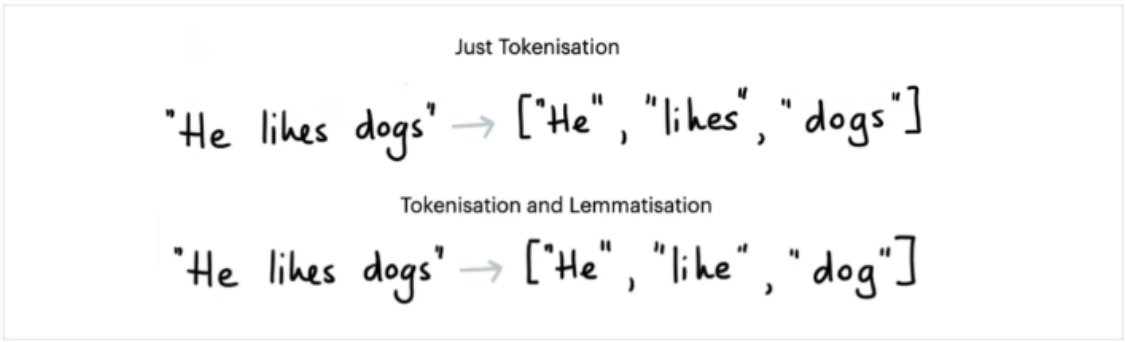

Figura 3.3: Tokenización

3.5.2. Caracterizadores

Los caracterizadores generan pesos numéricos para ser consumidos por los modelos de ML. Existen dos tipos principales de características, las Dispersas(Sparse Features) que usualmente cuentan lo que pueden representar subpalabras o ca-

Caracterizador	Requisitos	Tipo
MitieFeaturizer	MitieNLP	Dense featurizer
SpacyFeaturizer	Dense / Sparse Features	Logistic Regression de scikit-learn
ConveRTFeaturizer	Tokenization	Dense featurizer
LanguageModelFeaturizer	Tokenization	Dense featurizer
CountVectorsFeaturizer	Tokenization	Sparse featurizer
LexicalSyntactitFeaturizer	Tokenization	Sparse featurizer
RegexFeaturizer	Tokenization	Sparse featurizer

Tabla 3.1: Caracterizadores. Elaboración Propia

racterísticas léxicas y las Densas(Dense Features) estos suelen consistir en porciones preentrenadas, para que estos se desempeñen correctamente se debe seleccionar un Tokenizador apropiado. [23] Todos los caracterizadores son presentados en la tabla 3.1

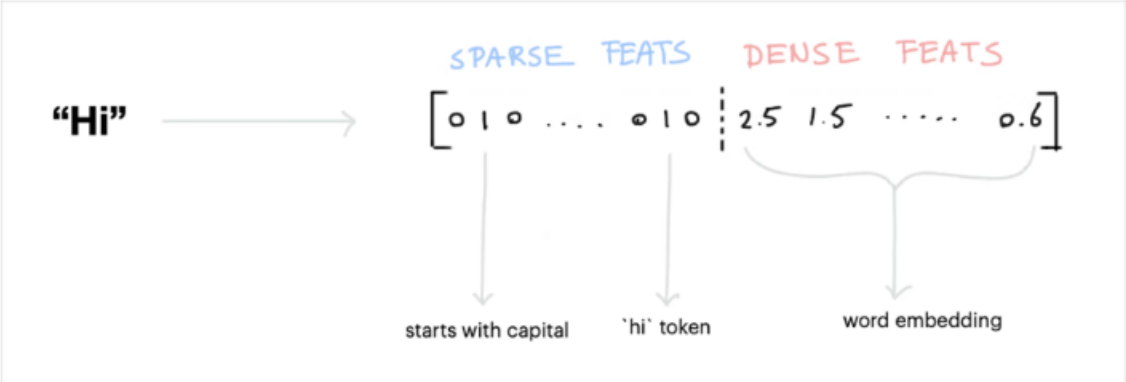

Figura 3.4: Caracterización

3.5.3. Clasificadores de Intención(Intent Classiffiers)

Una vez que se generaron las características para todos los tokens y para toda la oración, podemos pasarlos a un modelo clasificador de intenciones. Rasa por defecto usa el modelo DIET que puede encargarse tanto de la clasificación de la intención y extracción de entidades. También puede aprender tanto de características de Tokens como de oraciones. [23]

Clasificador	Requisitos	Utiliza
MitieIntentClassifier	MitieNLP	Clasificación multiclase con SVM
LogisticRegressionClassifier	Dense / Sparse Features	Logistic Regression de scikit-learn
SklearnIntentClassifier	Dense Features	SVM lineal optimizado con búsqueda en cuadrícula
KeywordIntentClassifier	Ninguno	Comparador de palabras clave
DIETClassifier	Dense Features	Transformadores
FallbackClassifier	Intents	Requiere de un clasificador de intenciones previo

Tabla 3.2: Clasificadores de intenciones. Elaboracion propia

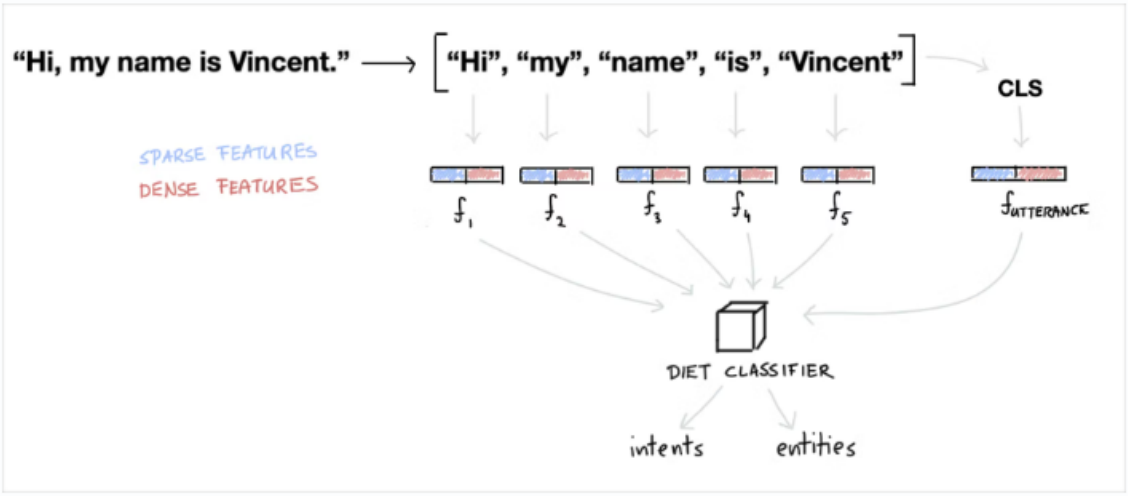

Figura 3.5: Clasificación de intenciones y entidades

3.5.4. Extracción de entidades

Además de DIET, existen otros clasificadores basados en ML que pueden aprender como detectar entidades, estos no son recomendados para todos los casos, también puede ser implementado un extractor basado en Expresiones Regulares(RegexEntityExtractor) [23]

Clasificador	Requisitos	Utiliza
MitieEntityExtractor	MitieNLP	Clasificación multiclase con SVM
SpacyEntityExtractor	SpacyNLP	Modelo estadístico BILOU
CRFEntityExtractor	Tokens	Campo aleatorio condicional (CRF)
DucklingEntityExtractor	Ninguno	Expresiones regulares
DIETClassifier	Dense Features	Transformadores
RegexEntityExtractor	Ninguno	Tablas de búsqueda

Tabla 3.3: Extractores de entidades. Elaboración propia

Figura 3.6: Extractor Regex

3.5.5. Selectores

Los selectores se encargan de predecir la respuesta de un conjunto de respuestas posibles para los retrieval intents, son utilizados posteriormente por el gestor de diálogo para dar la respuesta mas adecuada. Utiliza la misma arquitectura y optimización que el DIETClassifier.

3.5.6. Predicción de acciones

Con el flujo NLU, se detectan las entidades e intenciones. Pero este flujo no predice la siguiente acción en la conversación. Para esto se utiliza el flujo de política. Las políticas aseguran el uso de predicciones de NLU así como también

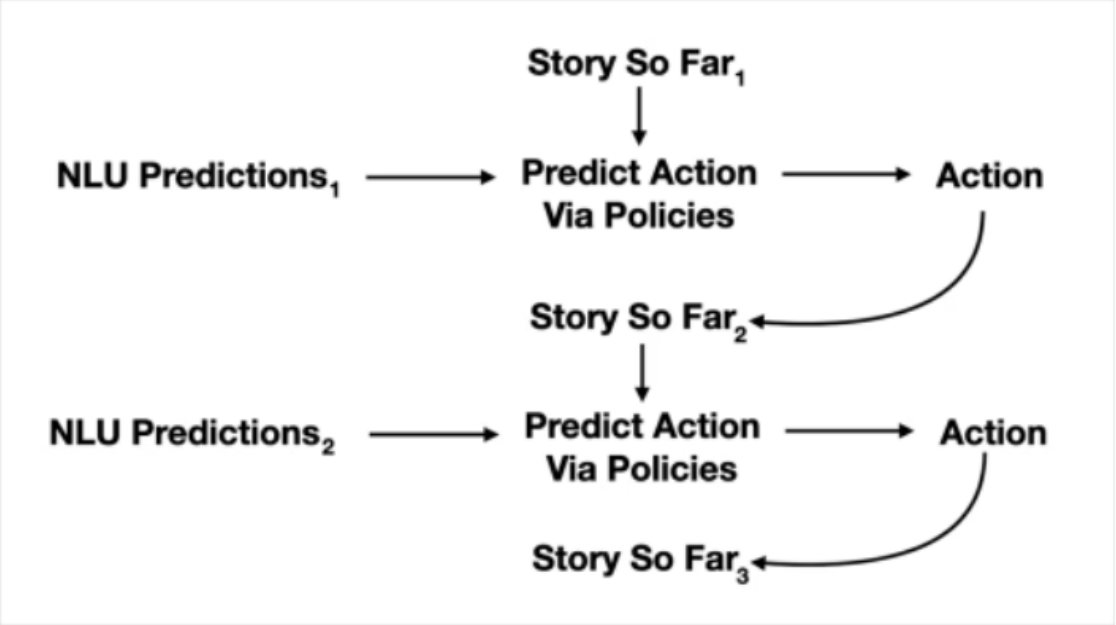

Figura 3.7: Predicción de acciones.

el estado presente de la conversaciones para decidir que acción tomar, pueden ser basado en reglas o en aprendizaje automático.

Políticas basadas en Aprendizaje Automático:

- **TED Policy:** Es un conjunto de algoritmos desarrollados por RASA para la predicción de diálogo y reconocimiento de entidades. Su arquitectura se basa en transformadores que convierten el diálogo actual en un vector de diálogos, para compararlos con otros vectores en busca del mas cercano, a partir de las acciones existentes. [24]
- **UnexpecTED Intent Policy:** Es una política auxiliar, tiene la misma arquitectura que TEDPolicy pero éste aprende cuales son las intenciones mas probables a ser expresadas según el contexto de la conversación. Siempre debe usarse en conjunto con al menos una otra política. [25]
- **Memoization Policy:** Esta política utiliza las historias y acciones de los datos de entrenamiento y las guarda en un diccionario, si la conversación actual no coincide con ningun ejemplo, predice un 0.0. [26]

- **Augmented Memoization Policy:** Tiene las mismas funcionalidades de Memoization Policy, pero además cuenta con un mecanismo que permite olvidar de forma aleatoria algunas partes de la conversación, luego predice las acciones ya con la historia reducida. [27]

Políticas basadas en Reglas:

- **Rule Policy:** Realiza las predicciones basandose en reglas que se tienen en los datos de entramiento.

Con cada interacción, las políticas definidas indican con un nivel de confianza cuál será la siguiente acción a ser tomada, aquella que tiene obtiene el mayor resultado sera la que decida la siguiente acción, en caso de que se prediga con la misma confianza, se tiene en cuenta la siguiente asignación de importancia.

- 6 - RulePolicy
- 3 - MemoizationPolicy o AugmentedMemoizationPolicy
- 2 - UnexpecTEDIntentPolicy
- 1 - TEDPolicy

3.6. Patrones de Conversación

3.6.1. Chitchat y FAQs

Las preguntas frecuentes y los chitchats (conversaciones sobre temas no importantes) son casos donde el asistente siempre tiene que responder de la misma forma, sin importar el contexto de la conversación. El problema se encuentra en que si creamos intenciones y acciones para cada pregunta que se realiza, las historias y reglas serán muy extensas, es por eso que estas preguntas frecuentes las agrupamos en una 'retrieval intent' y se selecciona la respuesta correcta

mediante el componente 'Response Selector'.

Para poder utilizarlo se debe configurar apropiadamente el archivo config.yml. Este componente utiliza la política basada en reglas 'RulePolicy', también caracterizadores y clasificadores de intenciones por lo que debe ubicarse en el pipeline después de estos.

3.6.2. Fallbacks

Para los casos donde el usuario pregunta algo que está fuera del alcance del bot, establecemos una intención llamada 'out of scope' a la que asociamos a una respuesta genérica y creamos una regla en el archivo rules.yml.

Existen casos donde la confianza en la clasificación es muy baja, esto implica que no se pueda predecir con buena confianza si se trata de una intención 'out of scope', para ello Rasa tiene una opción llamada 'Fallback' que permite pedir al usuario que reformule su pregunta para tratar nuevamente de predecir correctamente a que intención se refiere.

Para su utilización se debe agregar el 'FallbackClassifier' en el pipeline, crear respuestas por defecto y actualizar las reglas.

3.7. Pruebas

Rasa cuenta con varias funciones para probar los diálogos, historias, el gestor de diálogos y el procesamiento de mensajes.

3.7.1. Validación de datos

El comando 'rasa data validate' se encarga de verificar que no haya errores e inconsistencias en los datos y configuraciones. Es recomendable ejecutar este

comando antes de entrenar el modelo, ya que si se encuentra algún problema, el entrenamiento también podría fallar.

3.7.2. Evaluación del desempeño de la NLU

Una practica usual al ejecutar aprendizaje automático es dividir aleatoriamente el conjunto de datos en uno de entrenamiento y otro de pruebas. El bot utiliza el primer conjunto para aprender las características necesarias para realizar las predicciones adecuadas, y el segundo conjunto para evaluar el modelo mediante datos que aún no hayan sido vistos antes.
Rasa nos permite dividir los datos mediante el comando 'rasa data split nlu' que por defecto separa los datos de entrenamiento/prueba en un 80/20, luego, para probar que tan bien entrenado se encuentra el modelo utilizamos 'rasa test nlu' especificando cuales son los datos de entrenamiento y prueba de la siguiente forma:
–nlu train_test_split/test_data.yml

Otro método muy completo para evaluar el modelo es mediante validación cruzada o cross validation. Este divide los datos en múltiples subconjuntos llamados 'folds' donde se van alternando los datos de entrenamiento y pruebas

3.8. Despliegue del sistema

Según la documentación de Rasa, el mejor momento para desplegar una versión de prueba es tan rápido cuando se tenga un bot que cumpla los requisitos mínimos de los requisitos de diseño. De esta manera se puede tener pruebas de usuarios reales lo mas rápido posible y poder agregar estos casos.

3.8.1. Arquitectura

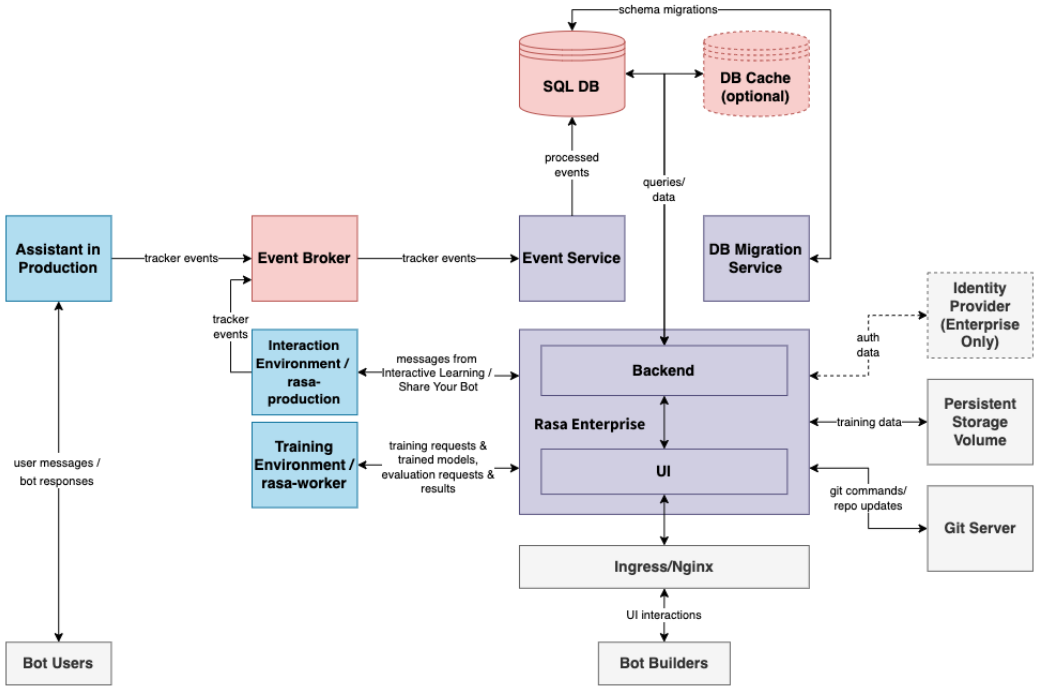

Figura 3.8: Arquitectura

Servicios

El diagrama muestra tres categorías de servicios: Los morados son componentes de principales de Rasa Enterprise, los azules son los servicios principales de Rasa Open Source y los anaranjados son servicios de terceros.

Tanto los componentes de Rasa Open Source e Enterprise tienen bases de datos independientes. Los eventos datos de conversaciones fluyen desde los servicios de Rasa Open Source a Rasa Enterprise a través del event broker.

Servicios de Rasa Enterprise

Los servicios de Rasa Enterprise, no son todos de libre uso, algunas partes se utilizan bajo un modelo de licencias. Pero estos si bien no son indispensables (pueden ser reemplazados por servicios libres de terceros) hacen que el despliegue de el producto sea mucho mas sencillo y así como también su desarrollo. Dicho esto pasamos a explicar los existentes en el diagrama, se tratan de los 3 mínimos servicios para ejecutar Rasa Enterprise estos deben ser todos de la misma versión para asegurar compatibilidad.

- **Servicio de eventos(Event Service):** Consume datos del corredor de eventos(event broker) y los archiva en la base de datos.
- **Servicio de migración de la Base de datos(DB migration service):** Asegura que los esquemas de la base de datos esten actualizados con respecto a la versión actual de Rasa Enterprise.
- **Rasa-X:** Ejecuta las tareas del back-end y front-end. Archiva y recupera datos de conversaciones, entrenamiento y meta-datos, como rótulos de conversaciones y banderas de mensajes en la base de datos de Rasa Enterprise. El front-end usa las entradas del back-end para brindar una interfaz amigable para el usuario.

Servicios de Rasa Open Source

Los servicios de Rasa Open Source se ejecutan de manera totalmente independiente de Rasa Enterprise, por otro lado Rasa Enterprise si depende de Rasa Open Source para manejar los datos de las conversaciones, entrenamiento y ejecución de modelos. En orden para que Rasa Enterprise pueda mostrar conversaciones tomadas por Rasa Open Source este publica los eventos de una

conversación a el mismo event broker al cual Rasa Enterprise esta consumiendo.

Partes

- **rasa-production:** Es el servicio que ejecuta el modelo entrenado, usado para parsear mensajes en intents y predecir acciones en conversaciones con el usuario, en un canal o en UI de Rasa Enterprise.
- **rasa-worker:** Es utilizado para servicios de segundo planos, como para entrenar un nuevo modelo.
- **app:** Es el servidor de acciones personalizadas, ejecutan acciones especificas de la aplicación.

CAPÍTULO 4

IMPLEMENTACIÓN

BIBLIOGRAFÍA

[1] ¿qué es la inteligencia artificial? [En línea]. Disponible: https://www.oracle.com/mx/artificial-intelligence/what-is-ai [Fecha de consulta: Abril 2022]

[2] S. Raj, *Building Chatbots with Python - Using Natural Language Processing and Machine Learning*, 1st ed. Apress, 2019.

[3] P. Hancox, “Sem1a5 - part 1 - a brief history of nlp.” [En línea]. Disponible: https://www.cs.bham.ac.uk/ pjh/sem1a5/pt1/pt1$_h istory.html$

[4] IBM, “Ibm archives: 701 translator,” Jan 2003. [En línea]. Disponible: https://www.ibm.com/ibm/history/exhibits/701/701$_t ranslator.html$

[5] S. Vajjala, B. Majumder, A. Gupta, y H. Surana, *Practical natural language processing : a comprehensive guide to building real-world NLP sysems*. O’reilly Media, Jul 2020.

[6] B. R. Ranoliya, N. Raghuwanshi, y S. Singh, “Chatbot for university related faqs,” *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, vol. 42017-January, pp. 1525–1530, 2017.

[7] N. M. Radziwill y M. C. Benton, “Evaluating quality of chatbots and intelligent conversational agents,” *arXiv preprint arXiv:1704.04579*, 2017.

[8] E. Adamopoulou y L. Moussiades, “An Overview of Chatbot Technology,” in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, y E. Pimenidis, Eds. Cham: Springer International Publishing, 2020, pp. 373–383.

[9] Documentacion de dialogflow | google cloud. [En línea]. Disponible: https://cloud.google.com/dialogflow/docs [Fecha de consulta: Marzo 2022]

[10] IBM Cloud, "Watson Assistant," *IBM Cloud*, 2020. [En línea]. Disponible: https://cloud.ibm.com/catalog/services/watson-assistant

[11] Ibm watson assisant | pricing. [En línea]. Disponible: https://www.ibm.com/products/watson-assistant/pricing [Fecha de consulta: Marzo 2022]

[12] Características de amazon lex: Amazon web services. [En línea]. Disponible: https://aws.amazon.com/es/lex/features/?nc=snloc=2 [Fecha de consulta: Marzo 2022]

[13] Rasa open source | rasa. [En línea]. Disponible: https://rasa.com/open-source/ [Fecha de consulta: Marzo 2022]

[14] About chatterbot - chatterbot 1.0.8 documentation. [En línea]. Disponible: https://chatterbot.readthedocs.io/en/stable/index.html [Fecha de consulta: Marzo 2022]

[15] Rasa glossary. [En línea]. Disponible: https://rasa.com/docs/rasa/stories/ [Fecha de consulta: Abril 2022]

[16] Introduction to rasa – rasa learning center. [En línea]. Disponible: https://learning.rasa.com/conversational-ai-with-rasa/introduction-to-rasa/ [Fecha de consulta: Abril 2022]

[17] Nlu training data. [En línea]. Disponible: https://rasa.com/docs/rasa/nlu-training-data [Fecha de consulta: Abril 2022]

[18] Rules. [En línea]. Disponible: https://rasa.com/docs/rasa/rules/ [Fecha de consulta: Abril 2022]

[19] Stories. [En línea]. Disponible: https://rasa.com/docs/rasa/stories/ [Fecha de consulta: Abril 2022]

[20] Configuration. [En línea]. Disponible: https://rasa.com/docs/rasa/model-configuration/ [Fecha de consulta: Julio 2022]

[21] Connecting to messaging and voice channels. [En línea]. Disponible: https://rasa.com/docs/rasa/messaging-and-voice-channels [Fecha de consulta: Julio 2022]

[22] Domain. [En línea]. Disponible: https://rasa.com/docs/rasa/domain/ [Fecha de consulta: Abril 2022]

[23] V. Warmerdam, "Intents entities: Understanding the rasa nlu pipeline," Mar 2022. [En línea]. Disponible: https://rasa.com/blog/intents-entities-understanding-the-rasa-nlu-pipeline/

[24] X. Kong, G. Wang, y A. Nichol, *Conversational AI with Rasa: Build, test, and deploy AI-powered, enterprise-grade virtual assistants and chatbots*. Packt Publishing, 2021.

[25] Policies | unexpected intent policy. [En línea]. Disponible: https://rasa.com/docs/rasa/policies/unexpected-intent-policy [Fecha de consulta: October 2022]

[26] Policies | memoization policy. [En línea]. Disponible: https://rasa.com/docs/rasa/policies/memoization-policy [Fecha de consulta: October 2022]

[27] Policies | augmented memoization policy. [En línea]. Disponible: https://rasa.com/docs/rasa/policies/augmented-memoization-policy [Fecha de consulta: October 2022]

Anexo A: Comandos de Rasa

Comando	Efecto
rasa init	Crea un nuevo proyecto con un ejemplo de entrenamiento, acciones y archivos de configuracion.
rasa train	Entrena un modelo utilizando datos NLU y lo guarda en ./models.
rasa interactive	Empieza una sesion interactiva de aprendizaje para crear un dataset nuevo de entrenamiento, chateando con el asistente
rasa shell	Carga el modelo entrenado y permite conversar con tu asistente en la linea de comando
rasa run	Inicia un servidor con el modelo entrenado
rasa run actions	Inicia un servidor de acciones utiliando el SDK de Rasa
rasa visualize	Genera una representación visual de las historias
rasas test	Prueba un modelo entrenado de Rasa en cualquier archivo que empiea con test_.
rasas data split nlu	Realiza una separacion 80/20 de los datos de entrernamiento
rasa data convert	Convierte los datos de entrenamiento entre diferentes formatos
rasa data migrate	Migra del dominio 2.0 al formato 3.0
rasa data validate	Controla que los datos del dominio, de la NLU y de la conversación no sean incoherentes
rasa export	Exporta conversaciones de un almacén de seguimiento a un corredor de eventos.
rasa valuates markers	Extrae los marcadores de un almacén de seguimiento existente.
rasa x	Inicia Rasa X en modo local
rasa -h	Muestra los comandos disponibles

Anexo B: El uso de plantillas

Aquí el contenido del anexo B