



ugr

Universidad
de **Granada**

CreaBot

Desarrollo de un asistente para la implementación y despliegue de sistemas conversacionales mediante la plataforma Rasa

Autor: Pablo Valenzuela Álvarez
Director: David Griol Barres

Grado en Ingeniería Informática

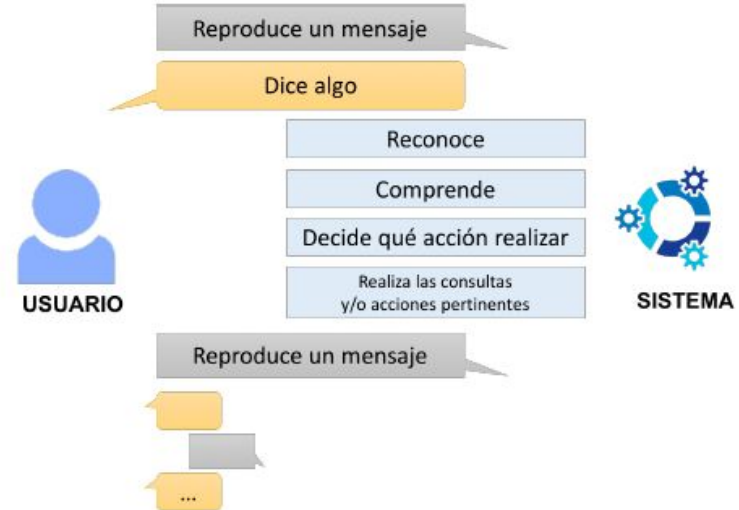
Contenidos

1. Introducción
2. La plataforma Rasa
3. Asistente implementado para desarrollar asistentes conversacionales:
 - 3.1. Utilización de distancias para la medición de la similitud semántica
 - 3.2. Tipos de chatbots que permite crear el asistente
 - 3.3. Configuración del chatbot mediante el fichero actions.py
 - 3.4. Interfaces del asistente y del webchat
 - 3.5. Conexión con NGROK y Telegram
 - 3.6. Chequeo de errores
4. Evaluación y Chequeo de errores
5. Conclusiones y trabajo futuro

1. Introducción

Sistema conversacionales

Sistema capaz de mantener una conversación, admitiendo entrada de datos y proporcionando salida en lenguaje natural.



Características sistemas conversacionales:

- Poder referenciar información dada por el usuario anteriormente.
- Ser capaz de redirigir el diálogo en uno, o unos, de los entornos definidos.
- Pedir la información necesaria para conseguir el objetivo deseado.
- Solicitar una aclaración, si hay confusión en la información aportada por el usuario.

1. Introducción

Enorme número de aplicaciones actuales de estos sistemas:

- Sistemas que muestran información sobre los transportes públicos.
- Sistemas de atención en el ámbito médico.
- Sistemas de banca online.
- Sistemas aplicados al Turismo.
- Aplicaciones accesibles dentro de vehículos.
- Sistemas de accesibilidad para personas con discapacidades.
- Aplicaciones en el campo de la educación.
- Asistente para dispositivos móviles.
- Sistemas de interacción en el hogar y control domótico.
- Robótica y sistemas wearables.
- ...

1. Introducción

Gran avance en estos sistemas gracias a los nuevos dispositivos inteligentes y los avances en Inteligencia Artificial y algoritmos de Deep Learning

Factores

- **Mejora GPUs:** Más capacidad de procesamiento, benefician Redes Neuronales
- **Big Data:** Gran cantidad de datos, favorecen inteligencia
- **Algoritmos Deep Learning:** Usan nuevas arquitecturas GPU
- **Nuevos dispositivos:** Teléfonos inteligentes, más funcionalidades
- **Computación en la nube:** Interacción en entornos domésticos



1. Introducción

Las grandes empresas tecnológicas muestran interés en este sector

Dispositivos y aplicaciones:

- Amazon Echo y Alexa
- Google Home
- Duer (Baidu)
- Siri (Apple)
- Cortana (Microsoft)
- ...



1. Introducción



1. Introducción

Objetivos del proyecto

Desarrollo de un asistente para implementar chatbots de distinta complejidad

1. Despliegue a través de la plataforma de código libre Rasa
2. Compresión de las frases de los usuarios a través de distancias de similitud semántica
3. Desarrollo de interfaces de entrada tanto para el asistente como para el chatbot (entornos web y Telegram)
4. Comprobación del correcto funcionamiento a través del desarrollo de ejemplos prácticos que cubren todas las funcionalidades del asistente

2. La plataforma Rasa



Rasa Open Source es una plataforma de código abierto orientada a crear asistentes virtuales capaces de comunicarse con usuarios mediante texto o voz

2. La plataforma Rasa

Rasa ofrece las funcionalidades necesarias para:

1. Extraer el significado de los mensajes
2. Gestionar conversaciones complejas
3. Aprender automáticamente
4. Conectarse con otras tecnologías actuales

2. La plataforma Rasa

Rasa ofrece las funcionalidades necesarias para:

1. **Extraer el significado de los mensajes**
2. Gestionar conversaciones complejas
3. Aprender automáticamente
4. Conectarse con otras tecnologías actuales



2. La plataforma Rasa

Rasa ofrece las funcionalidades necesarias para:

1. Extraer el significado de los mensajes
2. **Gestionar conversaciones complejas**
3. Aprender automáticamente
4. Conectarse con otras tecnologías actuales

```
entities:  
  - nombre  
  - telefono  
  - ciudad  
slots:  
  nombre:  
    type: text  
    mappings:  
      - type: from_entity  
        entity: nombre  
  telefono:  
    type: text  
    mappings:  
      - type: from_entity  
        entity: telefono  
  ciudad:  
    type: text  
    mappings:  
      - type: from_entity  
        entity: ciudad
```

2. La plataforma Rasa

Rasa ofrece las funcionalidades necesarias para:

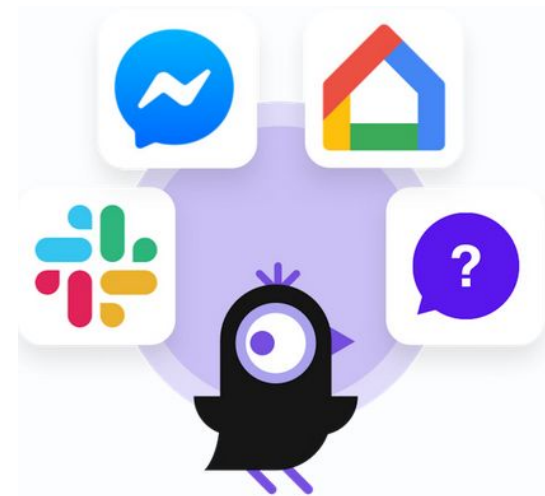
1. Extraer el significado de los mensajes
2. Gestionar conversaciones complejas
3. **Aprender automáticamente**
4. Conectarse con otras tecnologías actuales

```
? Your input -> happy
? Your NLU model classified 'happy' with intent 'mood_unhappy'
? What intent is it? (Use arrow keys)
  <create_new_intent>
    1.00 mood_unhappy
    0.00 greet
    0.00 deny
    0.00 goodbye
    0.00 bot_challenge
    0.00 affirm
  » 0.00 mood_great
```

2. La plataforma Rasa

Rasa ofrece las funcionalidades necesarias para:

1. Extraer el significado de los mensajes
2. Gestionar conversaciones complejas
3. Aprender automáticamente
4. **Conectarse con otras tecnologías actuales**



3. Asistente implementado para desarrollar asistentes conversacionales

- 3.1. Utilización de distancias para la medición de la similitud semántica
- 3.2. Tipos de chatbots que permite implementar el asistente
- 3.3. Configuración del chatbot mediante el fichero actions.py
- 3.4. Interfaces del asistente y del webchat
- 3.5. Conexión con NGROK y Telegram
- 3.6 Verificación de la ausencia de errores

Se han implementado las siguientes distancias para implementar la comprensión semántica en los chatbots

1. Similitud Coseno
2. Okapi BM25
3. Longest Common Subsequence
4. Jaro-Winkler

3.1.1. Similitud Coseno

Calcula el ángulo del coseno entre dos muestras.
Cuanto más cerca de 1, más se parecen las muestras.

$$\cos(\theta) = \frac{a \bullet b}{||a|| \times ||b||}$$

$$= \frac{(x_1, y_1) \bullet (x_2, y_2)}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

$$= \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

```
# coseno #  
vectorizer = TfidfVectorizer()  
  
t1 = vectorizer.fit_transform([text1]).toarray()  
t2 = vectorizer.transform([text2]).toarray()  
cos = cosine_similarity(t1, t2)  
score = cos[0][0]
```

3.1.2 Okapi BM25

Función de ranking basada en la bolsa de palabras.

Devuelve la similitud entre un conjunto de términos y los documentos comparados.

```
tokenized_corpus = [[]]
tokenized_corpus[0] = text1.split()
tokenized_query = text2.split()

bm25 = BM25Okapi(tokenized_corpus)
doc_scores = bm25.get_scores(tokenized_query)

score = abs(doc_scores[0])
```

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgtl}}\right)}$$

3.1.3 Longest Common Subsequence

Trata de encontrar la subsecuencia común más larga en dos secuencias proporcionadas.

```
# LCS #
score = SequenceMatcher(None, text1, text2).ratio()
```



$$\text{ratio} = 2 \cdot M / T$$

	A	C	B	A	D
A	1	1	1	1	1
B	1	1	2	2	2
C	1	2	2	2	2
D	1	2	2	2	3

Tabla 1: Ejemplo de funcionamiento LCS 1

	A	C	B	A	D
A	1	1	1	1	1
B	1	1	2	2	2
C	1	2	2	2	2
D	1	2	2	2	3

Tabla 2: Ejemplo de funcionamiento LCS 2

3.1.4 Jaro-Winkler

Jaro cuanta coincidencias y trasposiciones

$$j_{a,b} = \begin{cases} 0 & \text{si } m = 0, \\ \frac{1}{2} \left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m} \right) & \text{si } m \neq 0 \end{cases}$$

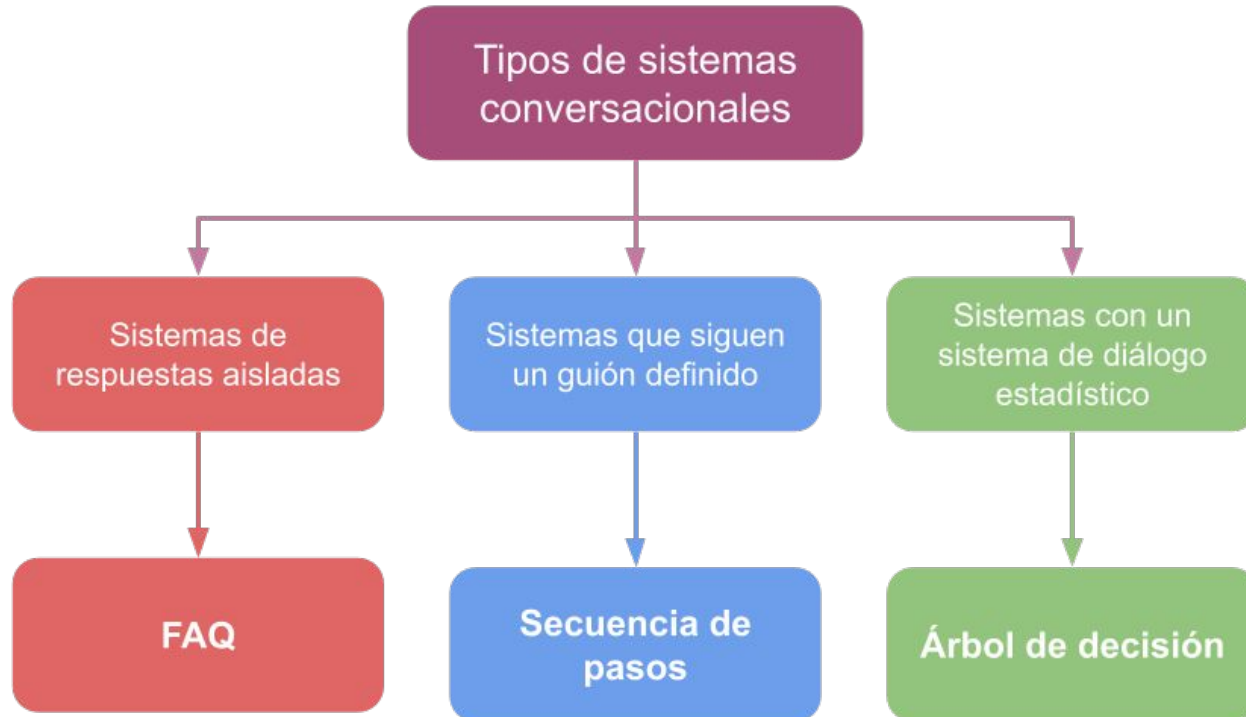


Winkler modificó esta fórmula para dar más peso cuando tengan prefijos comunes.

$$jk_{a,b} = j_{a,b} + lp(1 - j_{a,b})$$

```
# Jaro-Winkler #  
score = jellyfish.jaro_winkler_similarity(text1,text2)
```

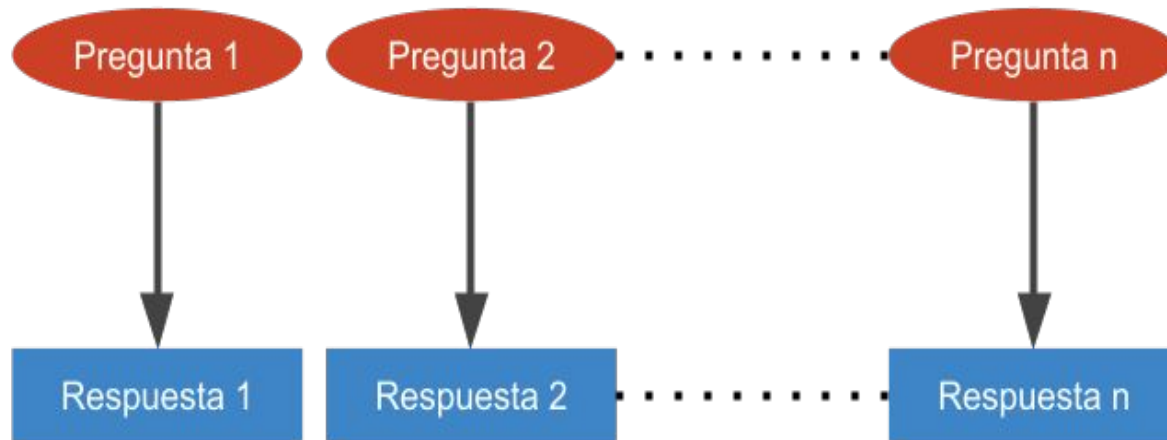
3.2. Tipos de chatbots que permite implementar el asistente



3.2. Tipos de chatbots que permite implementar el asistente

3.2.1 Preguntas Frecuentes (FAQ)

Sistemas que responden a preguntas sueltas y que no tienen relación entre sí.



3.2. Tipos de chatbots que permite implementar el asistente

3.2.1 Preguntas Frecuentes (FAQ)

Plantilla XML

```
<ficheroFAQ>
  <intent>
    <ejemplo>ejemplo</ejemplo>
    <respuesta>respuesta</respuesta>
  </intent>
  <intent>
    <ejemplo>ejemplo</ejemplo>
    <respuesta>respuesta</respuesta>
  </intent>
</ficheroFAQ>
```

Lista de intents

Para cada intent

1. Lista de ejemplos (usuario)
2. Lista de respuestas (chatbot)

3.2. Tipos de chatbots que permite implementar el asistente

3.2.1 Preguntas Frecuentes (FAQ)

Plantilla TXT

```
### primero van los intents con respuestas y ejemplos ###  
  
intent:Preguntal;(Respuesta1|Respuestaa1);{Preguntal|Preguntaa1}  
intent:Preguntal;(Respuesta2|Respuestaa2);{Pregunta2|Preguntaa2}  
intent:Preguntal;(Respuesta3|Respuestaa3);{Pregunta3|Preguntaa3}  
  
### y segundo y último, las stories ###  
  
story:Preguntal  
story:Pregunta2  
story:Pregunta3
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.1 Preguntas Frecuentes (FAQ)

Ejemplo

- ▼ ¿Cuál es el número máximo de destinatarios en un único mensaje?

El número máximo es 150. Si necesita enviar asiduamente mensajes a más destinatarios debe usar una lista de distribución.

- ¿Cuál es el número máximo de correos electrónicos que puedo enviar al día?
- ¿De cuanto espacio dispongo para mi cuenta de correo?
- ¿Puedo consultar la cuota que estoy usando en el correo electrónico?
- ¿Cómo puedo recuperar espacio libre en el correo electrónico una vez sobrepasada la cuota permitida?
- ¿Cual es el tamaño máximo de los archivos adjuntos a enviar con un correo?
- ¿Para que sirve la carpeta BUZONdeEntradaUGR?
- ¿Puedo avisar automáticamente que estoy de vacaciones a mis contactos cuando me manden un correo?
- ¿Que es webmail?

3.2. Tipos de chatbots que permite implementar el asistente

3.2.1 Preguntas Frecuentes (FAQ)

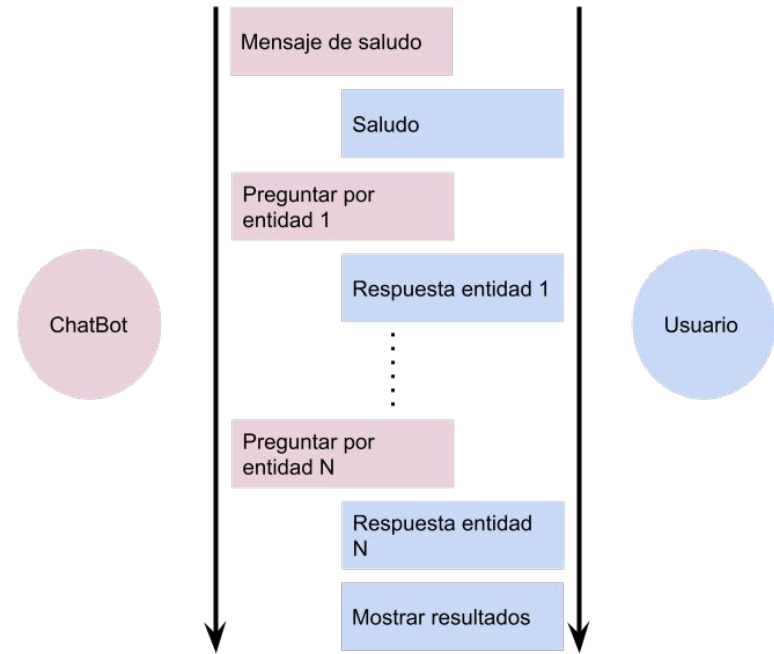
```
1 intent:Pregunta1;(El número máximo es 150. Si necesita enviar asiduamente mensajes a más destinatarios debe usar una lista de distribución);{¿Cuál
2
3 intent:Pregunta2;(A través de las plataformas de Webmail y desde el resto de formas de acceso al correo, el número máximo de correos electrónicos
4
5 intent:Pregunta3;(Las cuentas del tipo @correo.ugr.es tienen espacio para correo: 500 MB);{¿De cuanto espacio dispongo para mi cuenta de correo?}
6
7 intent:Pregunta4;(Está disponible a su izquierda en el programa Webmail. Podrá ver que espacio de su cuenta está usado y los límites de cuota que p
8
9 intent:Pregunta5;(Cuando se sobrepasa la cuota permitida el usuario recibirá un correo avisándole de que ha sobrepasado su cuota y que debe intenta
10
11 intent:Pregunta6;(Para el PAS/PDI de la Universidad, desde Webmail es posible enviar en un correo archivos adjuntos de hasta 10MB de tamaño);{¿Cual
12
13 intent:Pregunta7;(Con objeto de mejorar el rendimiento de su correo electrónico, existe una funcionalidad en las estafetas centrales de correo UGR
14
15 intent:Pregunta8;(Sí, a través de la configuración de webmail, podemos activar una notificación a nuestros contactos.);{¿Puedo avisar automáticamente
16
17 intent:Pregunta9;(Se trata de una herramienta para consultar correo electrónico mediante un navegador de Internet);{¿Que es webmail?}
18
19
20 story:Pregunta1
21 story:Pregunta2
22 story:Pregunta3
23 story:Pregunta4
24 story:Pregunta5
25 story:Pregunta6
26 story:Pregunta7
27 story:Pregunta8
28 story:Pregunta9
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.2 Secuencia de Pasos

Sistemas que siguen un guión predefinido

El sistema pregunta automáticamente por
las entidades



3.2. Tipos de chatbots que permite implementar el asistente

3.2.2 Secuencia de Pasos - Plantilla XML

```
<ficheroSEQ>
  <entidad>
    <nombre>entidad1</nombre>
    <tipo>tipo1</tipo>
  </entidad>
  <entidad>
    <nombre>entidad2</nombre>
    <tipo>tipo2</tipo>
  </entidad>

  <intent>
    <ejemplo>esto es un [ejemplo1](entidad1)</ejemplo>
    <ejemplo>un [ejemplo1](entidad1) es esto</ejemplo>
  </intent>

  <intent>
    <ejemplo>esto es un [ejemplo2](entidad2)</ejemplo>
    <ejemplo>un [ejemplo2](entidad2) es esto</ejemplo>
  </intent>
</ficheroSEQ>
```

Lista de entidades

Lista de intents:

- Lista de ejemplos *

3.2. Tipos de chatbots que permite implementar el asistente

3.2.2 Secuencia de Pasos - Plantilla TXT

```
# primero definimos las entidades
```

```
entities:entidad1;entidad2
```

```
entity:entidad3
```

```
# seguido de los intent con respuestas para las entidades
```

```
intent:pedir_entidad1;{Es [entidad1] (entidad1) | Es otra [entidad1] (entidad1)}
```

```
intent:pedir_entidad2;{Es [entidad1] (entidad2) | Es otra [entidad1] (entidad2)}
```

```
intent:pedir_entidad3;{Es [entidad1] (entidad3) | Es otra [entidad1] (entidad3)}
```

3.2. Tipos de chatbots que permite implementar el asistente

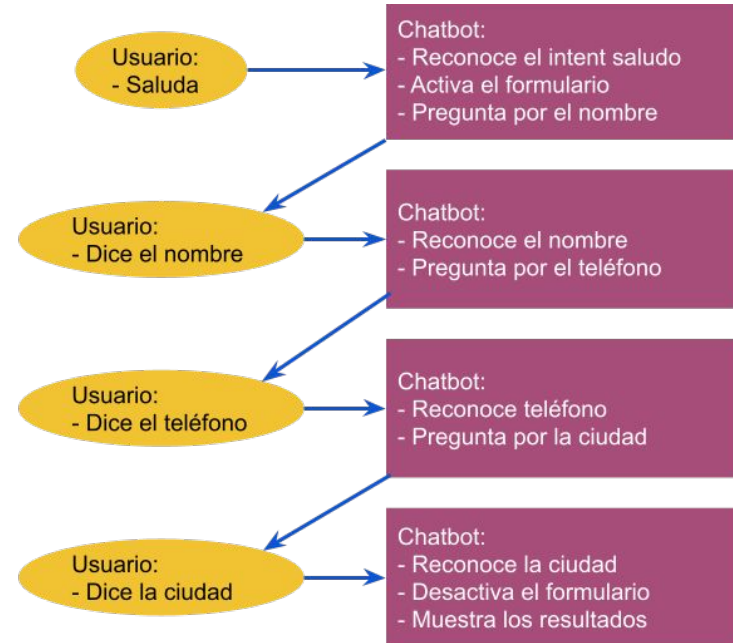
3.2.2 Secuencia de Pasos - Ejemplo

Consta de tres entidades:

- Nombre
- Teléfono
- Ciudad

El formulario se activa al detectar Saludo y pregunta en secuencia por las entidades

Una vez termina muestra los resultados



3.2. Tipos de chatbots que permite implementar el asistente

3.2.2 Secuencia de Pasos - Ejemplo

```
1  
2 entities:nombre;telefono  
3 entity:ciudad  
4  
5 intent:pedir_nombre;{Me llamo [Pablo](nombre)|Me llamo [Ines](nombre)|  
6 intent:pedir_telefono;{Mi telefono es [987645374](telefono)|Mi numero  
7 intent:pedir_ciudad;{Soy de [Granada](ciudad)|Soy de [Huelva](ciudad)|  
8
```


3.2. Tipos de chatbots que permite implementar el asistente

3.2.2 Secuencia de Pasos - Ejemplo

No hacen falta stories

Al detectar el Intent “Saludo”

Activa el formulario

Y pide las entidades
pertenecientes al formulario
automáticamente

```
1  version: "3.0"
2  rules:
3    - rule: Activate Data Form
4      steps:
5        - intent: Saludo
6        - action: data_form
7        - active_loop: data_form
8    - rule: Submit Data Form
9      condition:
10       - active_loop: data_form
11      steps:
12        - action: data_form
13        - active_loop: null
14        - slot_was_set:
15          - requested_slot: null
16        - action: utter_submit
17        - action: utter_slots_values
18
```

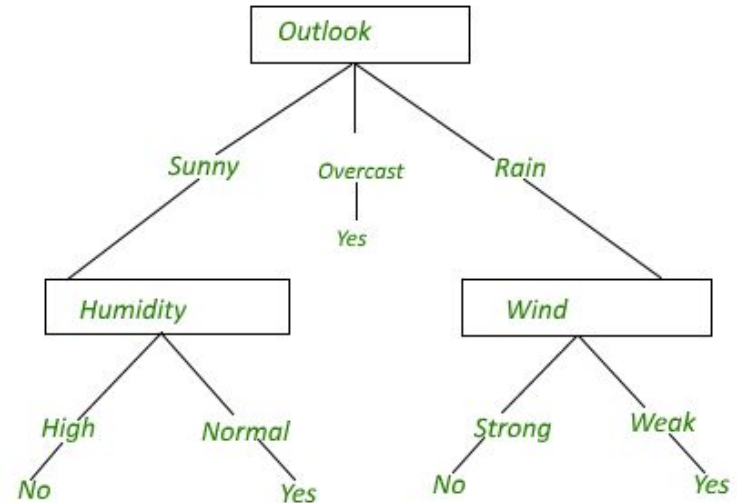
3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión

Siguen un sistema basado en un árbol

Requiere la toma de decisiones

Cada una de ellas genera nuevos caminos y más profundos



3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Plantilla XML

```
<intent>
  <nombre>saludo</nombre>
  <ejemplo>hola</ejemplo>
  <ejemplo>buenas</ejemplo>
</intent>
```

```
<intent>
  <nombre>estado_feliz</nombre>
  <ejemplo>estoy feliz</ejemplo>
  <ejemplo>feliz</ejemplo>
  <ejemplo>contento</ejemplo>
  <ejemplo>alegre</ejemplo>
  <ejemplo>con una sonrisa</ejemplo>
</intent>
```

```
<intent>
  <nombre>estado_triste</nombre>
  <ejemplo>estoy triste</ejemplo>
  <ejemplo>triste</ejemplo>
  <ejemplo>enfadado</ejemplo>
  <ejemplo>llorando</ejemplo>
  <ejemplo>cabreado</ejemplo>
</intent>
```

```
<response>
  <nombre>utter_saludo</nombre>
  <respuesta>¡Hola!</respuesta>
  <respuesta>¡Hey!</respuesta>
  <respuesta>¡Muy Buenas!</respuesta>
</response>
```

```
<response>
  <nombre>utter_preguntar_estado</nombre>
  <respuesta>¿Como te sientes?</respuesta>
  <respuesta>¿Me puedes decir como te sientes?</respuesta>
</response>
```

```
<response>
  <nombre>utter_feliz</nombre>
  <respuesta>¡Que bien!</respuesta>
  <respuesta>¡Fenomenal!</respuesta>
  <respuesta>¡Espléndido!</respuesta>
</response>
```

```
<response>
  <nombre>utter_triste</nombre>
  <respuesta>¡Vaya :(!</respuesta>
  <respuesta>¡Eso no esta bien!</respuesta>
  <respuesta>¡Hay que arreglar eso!</respuesta>
</response>
```

Tenemos Intents:

- Nombre
- Lista de ejemplos

y Responses:

- Nombre
- Lista de respuestas

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Plantilla XML

```
<story>
  <nombre>happy path</nombre>
  <user>saludo</user>
  <bot>utter_saludo</bot>
  <bot>utter_preguntar_estado</bot>
  <user>estado_feliz</user>
  <bot>utter_feliz</bot>
</story>
```

```
<story>
  <nombre>sad path</nombre>
  <user>saludo</user>
  <bot>utter_saludo</bot>
  <bot>utter_preguntar_estado</bot>
  <user>estado_triste</user>
  <bot>utter_triste</bot>
</story>
```

Stories con más elementos:

- Nombre
- User: Intenciones
- Bot: Acciones

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Plantilla TXT

Acepta intents de 3 y 2 elementos

```
1
2 ### primero, si hay, se ponen los intents con respuestas y ejemplos ###
3
4 intent:PreguntaInicial;(Respuesta1|Otra respuesta1);{Preguntal|Otra preguntal}
5
6
7
8 ### seguido, van los intents con respuestas ###
9
10 intent:Respuesta1;{Esto es la respuesta1|Esto es otra respuesta1}
11 intent:Respuesta2;{Esto es la respuesta2|Esto es otra respuesta2}
12
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Plantilla TXT

Se refieren a las preguntas del chatbot

Y a los nodos hoja del árbol

```
14
15 ### ahora, las preguntas a las respuestas anteriores ###
16
17 response:Pregunta;(¿Pregunta?|¿Esto es una pregunta?)
18
19
20
21 ### penultimo, los responses con las respuestas finales (nodos hoja) ###
22
23 response:Opcion1;(Es la opcion1)
24 response:Opcion2;(Es la opcion2)
25
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Plantilla TXT

Las stories cuentan con elementos ilimitados

Admite elementos:

- Normales
- Con paréntesis
- Con *

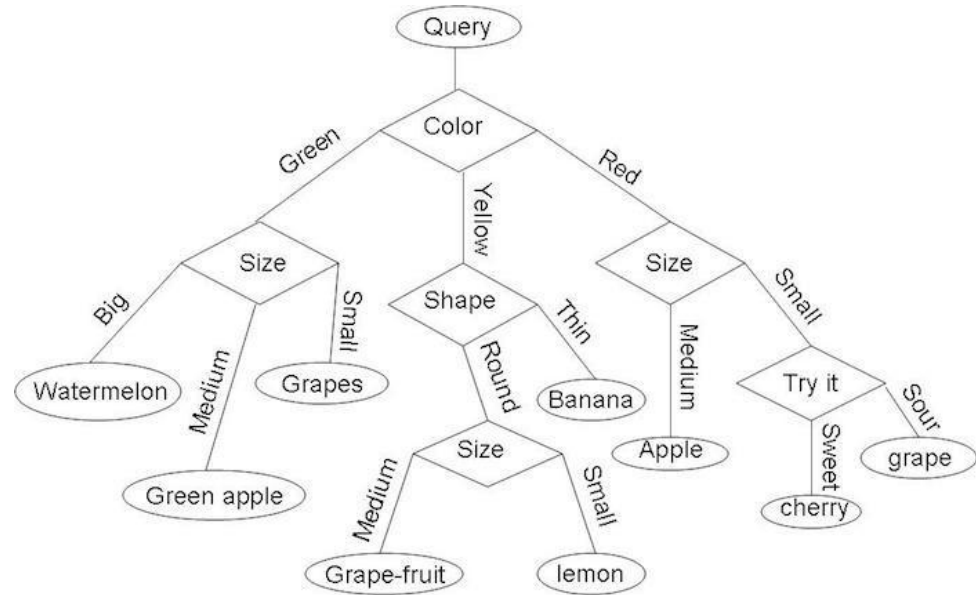
```
27  
28   ### Y por ultimo, las stories ###  
29  
30   story:PreguntaInicial;Pregunta(Respuesta1);*Opcion1  
31   story:PreguntaInicial;Pregunta(Respuesta2);*Opcion2
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Ejemplo TXT

Ejemplo sencillo:

- Contiene 9 ramas
- Toma de decisiones
- Distintos niveles de profundidad



3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Ejemplo TXT

```
15 response:Color;(¿De que color es tu fruta?|¿Que color tiene tu fruta?)
16 response:Tamano;(¿Cual es su tamaño?|¿Que tamaño tiene?)
17 response:Forma;(¿Cual es su forma?|¿Que forma tiene?)
18 response:Sabor;(¿Que sabor tiene?|¿Cual es su sabor?)
19 response:Sandia;(Es una sandia)
20 response:Manzana;(Es una manzana)
21 response:Uvas;(Son uvas)
22 response:Pomelo;(Es un pomelo)
23 response:Limon;(Es un limón)
24 response:Platano;(Es un plátano)
25 response:Cerezas;(Son cerezas)
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Ejemplo TXT

```
1
2 intent:Saludo;(;Hola!|;Saludos!);{Hola,Muy buenas|Buenas|Buenos dias|Buenas noches}
3
4 intent:Color_Verde;{Es verde|Es de color verde|Su color es verde}
5 intent:Color_Amarillo;{Es amarilla|Es de color amarillo|Su color es amarillo}
6 intent:Color_Rojo;{Es roja|Es de color rojo|Su color es rojo}
7 intent:Tamano_Grande;{Es grande|Es de gran tamaño|Su tamaño es grande}
8 intent:Tamano_Medio;{Es medio|Es mediano|Es de tamaño medio|Su tamaño es mediano}
9 intent:Tamano_Pequeno;{Es Pequeño|Es de pequeño tamaño|Su tamaño es pequeño}
10 intent:Forma_Redonda;{Es redondo|Redondeado|Su forma es redonda|Forma redondeada}
11 intent:Forma_Alargada;{Es largo|alargado|Su forma es larga|Forma alargada}
12 intent:Sabor_Dulce;{Sabe dulce|Tiene un sabor dulce|Es dulce}
13 intent:Sabor_Amargo;{Es amargo|Tiene un sabor amargo|Sabe amargo}
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Ejemplo TXT

```
27 story:Saludo;Color (Color_Verde);Tamano (Tamano_Grande);*Sandia
28 story:Saludo;Color (Color_Verde);Tamano (Tamano_Medio);*Manzana
29 story:Saludo;Color (Color_Verde);Tamano (Tamano_Pequeno);*Uvas
30 story:Saludo;Color (Color_Amarillo);Forma (Forma_Redonda);Tamano (Tamano_Medio);*Pomelo
31 story:Saludo;Color (Color_Amarillo);Forma (Forma_Redonda);Tamano (Tamano_Pequeno);*Limon
32 story:Saludo;Color (Color_Amarillo);Forma (Forma_Alargada);*Platano
33 story:Saludo;Color (Color_Rojo);Tamano (Tamano_Medio);*Manzana
34 story:Saludo;Color (Color_Rojo);Tamano (Tamano_Pequeno);Sabor (Sabor_Dulce);*Cerezas
35 story:Saludo;Color (Color_Rojo);Tamano (Tamano_Pequeno);Sabor (Sabor_Amargo);*Uvas
36
```

3.2. Tipos de chatbots que permite implementar el asistente

3.2.3. Modelos de diálogo estadísticos basados en árboles de decisión - Ejemplo TXT

```
- story:
  steps:
    - intent: Saludo
    - action: action_mirar_ejemplos
    - action: utter_Saludo
    - action: utter_Color
    - intent: Color_Amarillo
    - action: action_mirar_ejemplos
    - action: utter_Forma
    - intent: Forma_Redonda
    - action: action_mirar_ejemplos
    - action: utter_Tamano
    - intent: Tamano_Medio
    - action: action_mirar_ejemplos
    - action: utter_Pomelo
```

3.3 Configuración del chatbot mediante el fichero actions.py

Se lleva a cabo la elección del algoritmo que usará el chatbot

```
# segun el algoritmo ...
if opcion == "Jaro-Winkler":
    actions.write("                ratio = otra.score_JW(ejemplo, comparacion)
if opcion == "LCS":
    actions.write("                ratio = otra.score_LCS(ejemplo, comparacion)
if opcion == "Coseno":
    actions.write("                ratio = otra.score_Cosine(ejemplo, comparacion)
if opcion == "BM25":
    actions.write("                ratio = otra.score_BM25(ejemplo, comparacion)
```


Como se añaden ejemplos en los chatbots de Preguntas Frecuentes y de Árbol de Decisión

Donde:

- **intent**: Intent predicho por Rasa
- **contenido**: todo fichero nlu.yml
- **comparación**: ejemplos del fichero nlu.yml. Los que ya contiene el chatbot.
- **ejemplo**: frase dada por el usuario.
- **ratio**: porcentaje de similitud entre ejemplo y comparación
- **limiteMIN**: límite mínimo de similitud
- **limiteMAX**: límite máximo de similitud
- **parecido**: variable boolean. Indicador principal para añadir el ejemplo.
- **maxEjem**: máximo de ejemplos que tiene el intent del chatbot.

```
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137

while not intent in contenido[c]:
    c = c + 1

c = c+2
d = c
k = 0

while d<len(contenido) and not 'intent' in contenido[d]:
    k = k + 1
    comparacion = contenido[d][6:]

    ratio = otra.score_LCS(ejemplo, comparacion)

    if ratio > limiteMAX:
        parecido = True
        break
    elif ratio > limiteMIN and ratio < limiteMAX:
        parecido = False
    else:
        parecido = True

    d = d+1

if parecido == False and k<=maxEjem:
    contenido.insert(c, ' - {}\n'.format(ejemplo))
    nlu.seek(0)
    nlu.writelines(contenido)
    incluido = True
```

Como se añaden ejemplos en el chatbot de Secuencia de Pasos

Hay que añadir funciones
para validar los slots:

- Intent
- Ejemplo
- Entidad
- Valor

```
# funcion para validar cada slot
for e in entities_array:
    actions.write('\n')
    actions.write('    def validate_{}(self, slot_value: Any, dispatcher: CollectingDispatcher,
    actions.write("        intent = tracker.latest_message['intent'].get('name')\n")
    actions.write("        ejemplo = tracker.latest_message.get('text')\n")
    actions.write("        entidad = tracker.latest_message['entities'][0]['entity']\n")
    actions.write("        valor = tracker.latest_message['entities'][0]['value']\n")
    actions.write("        miClase.comprobar_ejemplo(intent, entidad, valor, ejemplo)\n")
    actions.write("        return {{'{}':slot_value}}\n\n".format(e))
```



```
ejemplo = ejemplo.replace(value, '[' + value + ']( ' + entity + ' )')
```

Vista final del archivo action.py del chatbot Secuencia de Pasos

```
class ValidateDataForm(FormValidationAction):  
    def name(self) -> Text:  
        return 'validate_data_form'  
  
    def validate_nombre(self, slot_value: Any, dispatcher: CollectingDispatcher):  
        intent = tracker.latest_message['intent'].get('name')  
        ejemplo = tracker.latest_message.get('text')  
        entidad = tracker.latest_message['entities'][0]['entity']  
        valor = tracker.latest_message['entities'][0]['value']  
        miClase.comprobar_ejemplo(intent, entidad, valor, ejemplo)  
        return {'nombre': slot_value}  
  
    def validate_telefono(self, slot_value: Any, dispatcher: CollectingDispatcher):  
        intent = tracker.latest_message['intent'].get('name')  
        ejemplo = tracker.latest_message.get('text')  
        entidad = tracker.latest_message['entities'][0]['entity']  
        valor = tracker.latest_message['entities'][0]['value']  
        miClase.comprobar_ejemplo(intent, entidad, valor, ejemplo)  
        return {'telefono': slot_value}
```


3.4 Interfaces del asistente y del webchat

Crear de Asistentes con RASA

Selecciona un tipo de asistente
Árbol de decisión

Selecciona un archivo XML
Buscar
Mostrar archivo

Selecciona un método para añadir ejemplos
Jaro-Winkler Recomendado
Límite mínimo
0.5 %
Límite máximo
0.7 %
Número máximo de ejemplos
20

Indica un nombre para el asistente
Y una localización
Elegir

Crear asistente

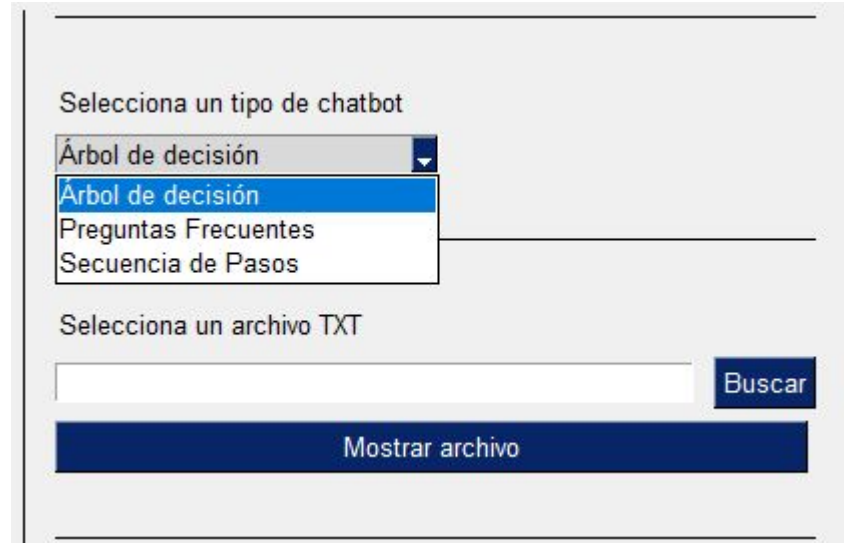
Contenido del archivo seleccionado

Modificar Limpiar

Selección del chatbot y de la plantilla

Los tres tipos implementados

Cargar la plantilla



Selecciona un tipo de chatbot

Árbol de decisión

- Árbol de decisión
- Preguntas Frecuentes
- Secuencia de Pasos

Selecciona un archivo TXT

Buscar

Mostrar archivo

Panel de edición

- Totalmente editable
- Modifica la plantilla
sobrescribiendo el archivo
original
- Limpia el panel

Contenido del archivo seleccionado

```
### primero, si hay, se ponen los intents con respuestas y ejemplos ###  
intent: Preguntainicial;{Respuesta1|Otra respuesta1};{Pregunta1|Otra pregunta1}  
  
### seguido, van los intents con respuestas ###  
intent: Respuesta1;{Esto es la respuesta1|Esto es otra respuesta1}  
intent: Respuesta2;{Esto es la respuesta2|Esto es otra respuesta2}  
  
### ahora, las preguntas a las respuestas anteriores ###  
response: Pregunta;{¿Pregunta?|¿Esto es una pregunta?}  
  
### penultimo, los responses con las respuestas finales (nodos hoja) ###  
response: Opcion1;{Es la opcion1}  
response: Opcion2;{Es la opcion2}  
  
### Y por ultimo, las stories ###  
story: Preguntainicial;Pregunta(Respuesta1);*Opcion1  
story: Preguntainicial;Pregunta(Respuesta2);*Opcion2
```

Modificar Limpiar

Selección de algoritmo y parámetros

Selección de algoritmo

Límite mínimo

Límite máximo

Número de ejemplos



Selecciona un método para añadir ejemplos

Jaro-Winkler ▼ Recomendado

Límite mínimo de coincidencia

0.5 ▼ %

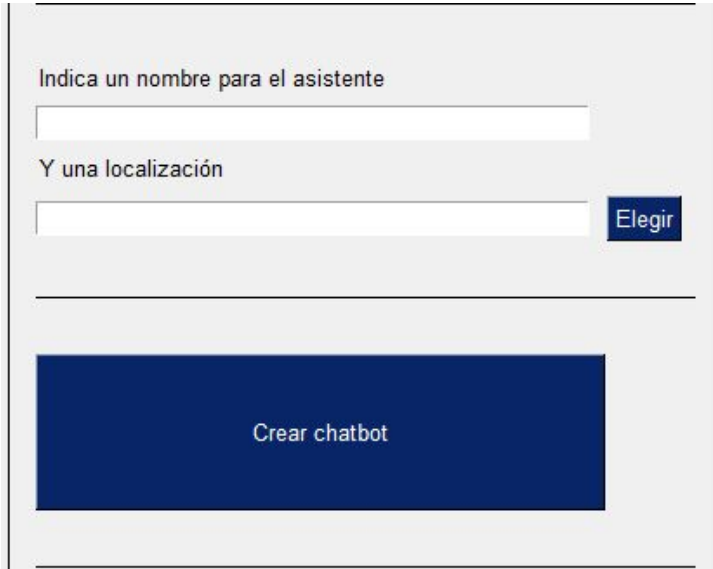
Límite máximo de coincidencia

0.7 ▼ %

Número máximo de ejemplos

20 ▼

Selección de algoritmo y parámetros



Indica un nombre para el asistente

Y una localización

Elegir

Crear chatbot

Nombre del chatbot

Localización

Creación del chatbot

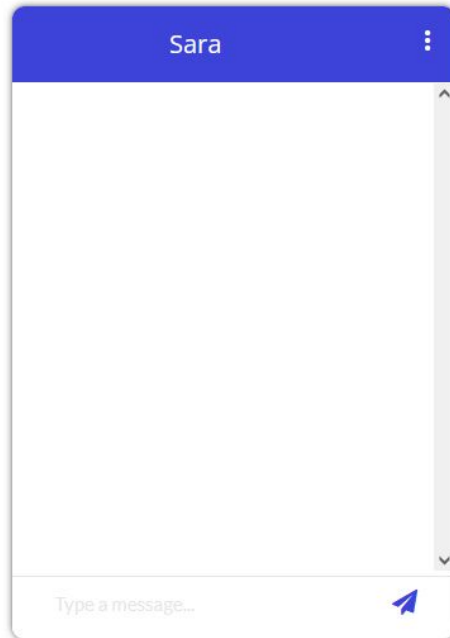
3.4 Interfaces del asistente y del webchat

Hola soy Sara 🙋

Puedo ayudarte si tienes alguna duda

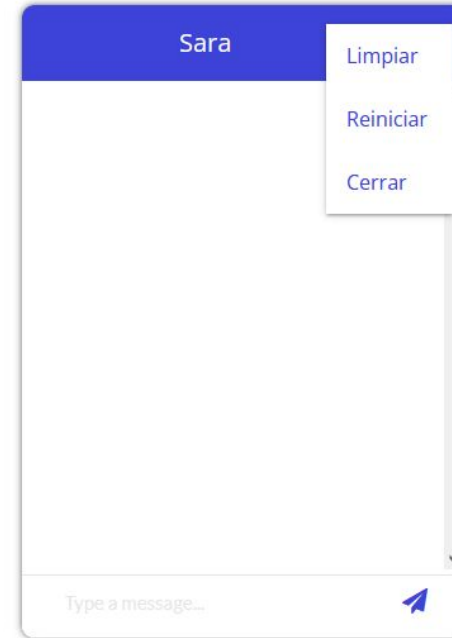


Despliegue



Cuadro de diálogo

Espacio para escribir



← Funciones

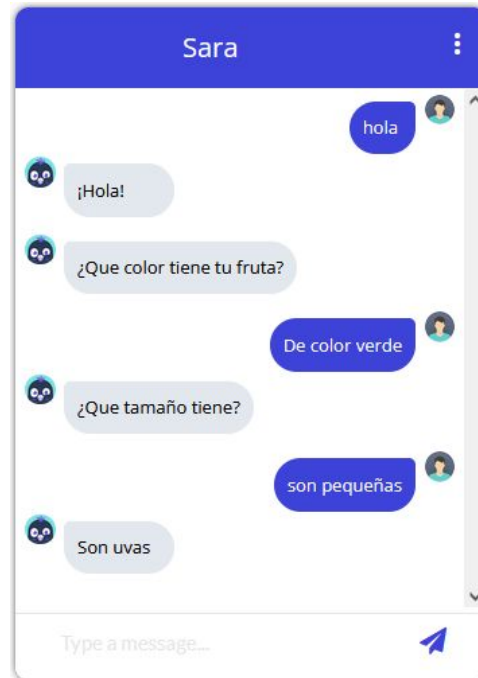
Ejemplo de conversación: Preguntas Frecuentes



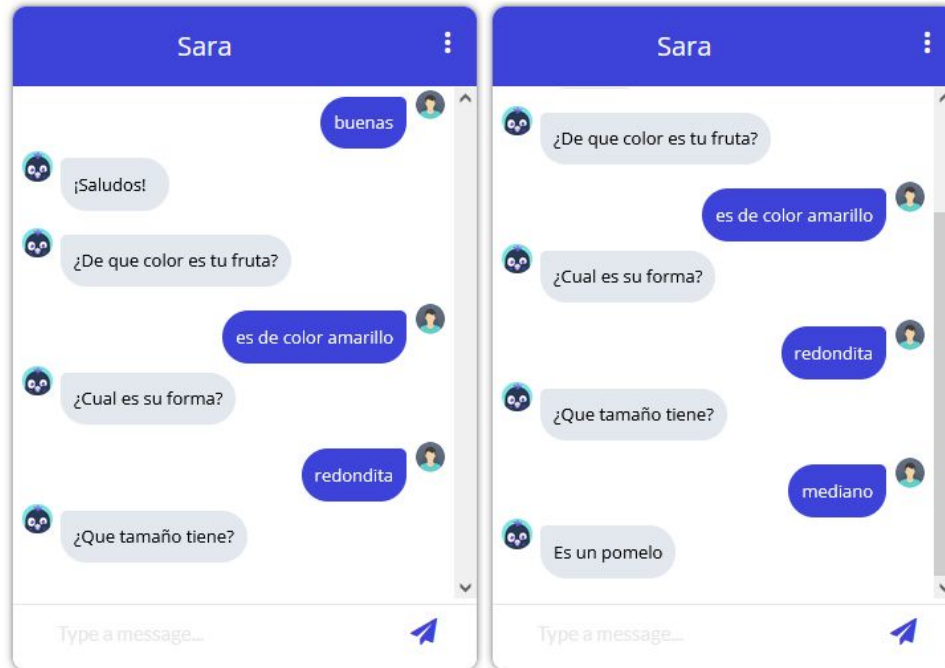
Ejemplo de conversación: Secuencia de Pasos



Ejemplo de conversación: Árbol de decisión 1



Ejemplo de conversación: Árbol de decisión 2



Ejemplo de conversación: Árbol de decisión 3



Despliegue web con NGROK

Crea un enlace público con nuestro servidor

```
Símbolo del sistema - ngrok: http 80
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Account             [REDACTED] (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding           http://eb71-85-136-44-1.ngrok.io -> http://localhost:80
Forwarding           https://eb71-85-136-44-1.ngrok.io -> http://localhost:80

Connections         ttl    opn    rt1    rt5    p50    p90
                   0      0      0.00   0.00   0.00   0.00
```

Despliegue en Telegram

Haciendo uso del enlace generado https por NGROK

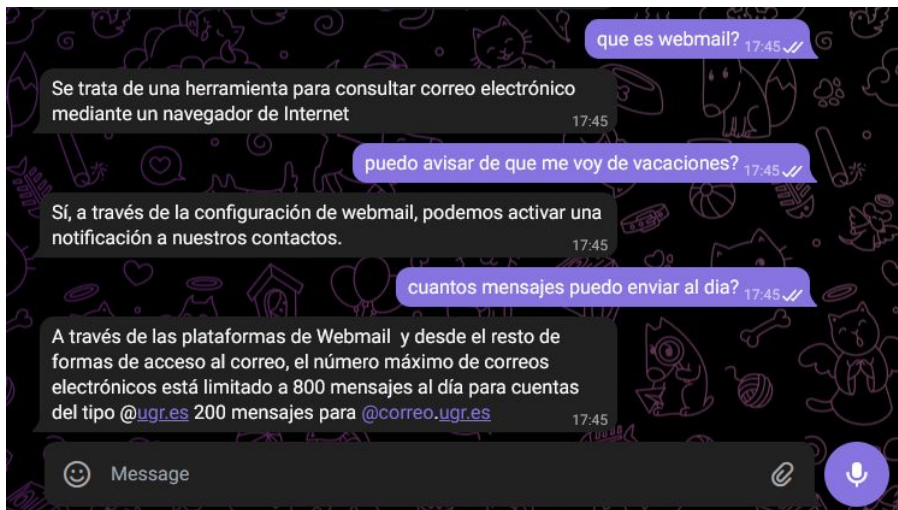
+

```
telegram:  
  access_token: "5524883291:AAHYxMscidoJ25TX_hX_-u1oLoMucaMyKxE"  
  verify: "pvalenz_bot"  
  webhook_url: "https://1d6b-85-136-44-1.ngrok.io/webhooks/telegram/webhook"
```

3.5 Conexión con NGROK y Telegram

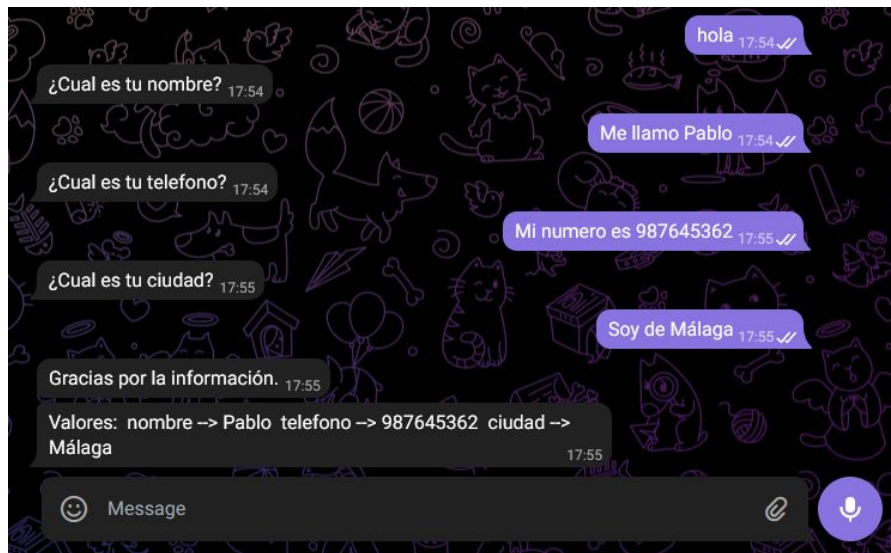
63

Ejemplos de conversación usando Telegram Web



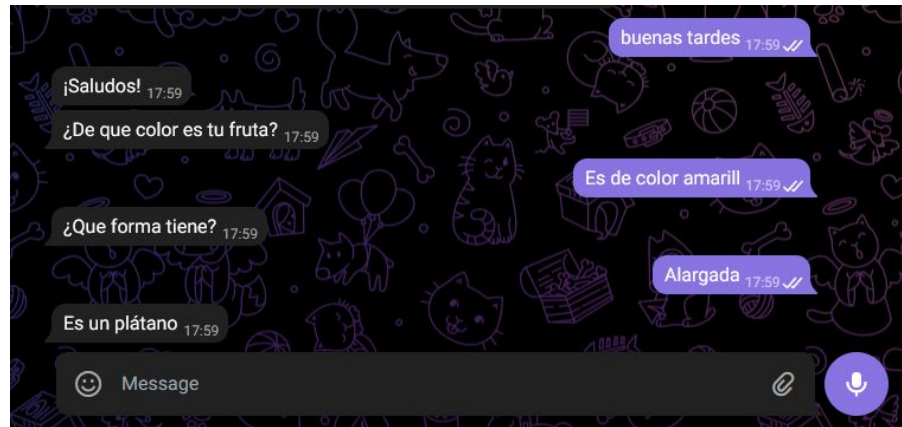
3.5 Conexión con NGROK y Telegram

Ejemplos de conversación usando Telegram Web

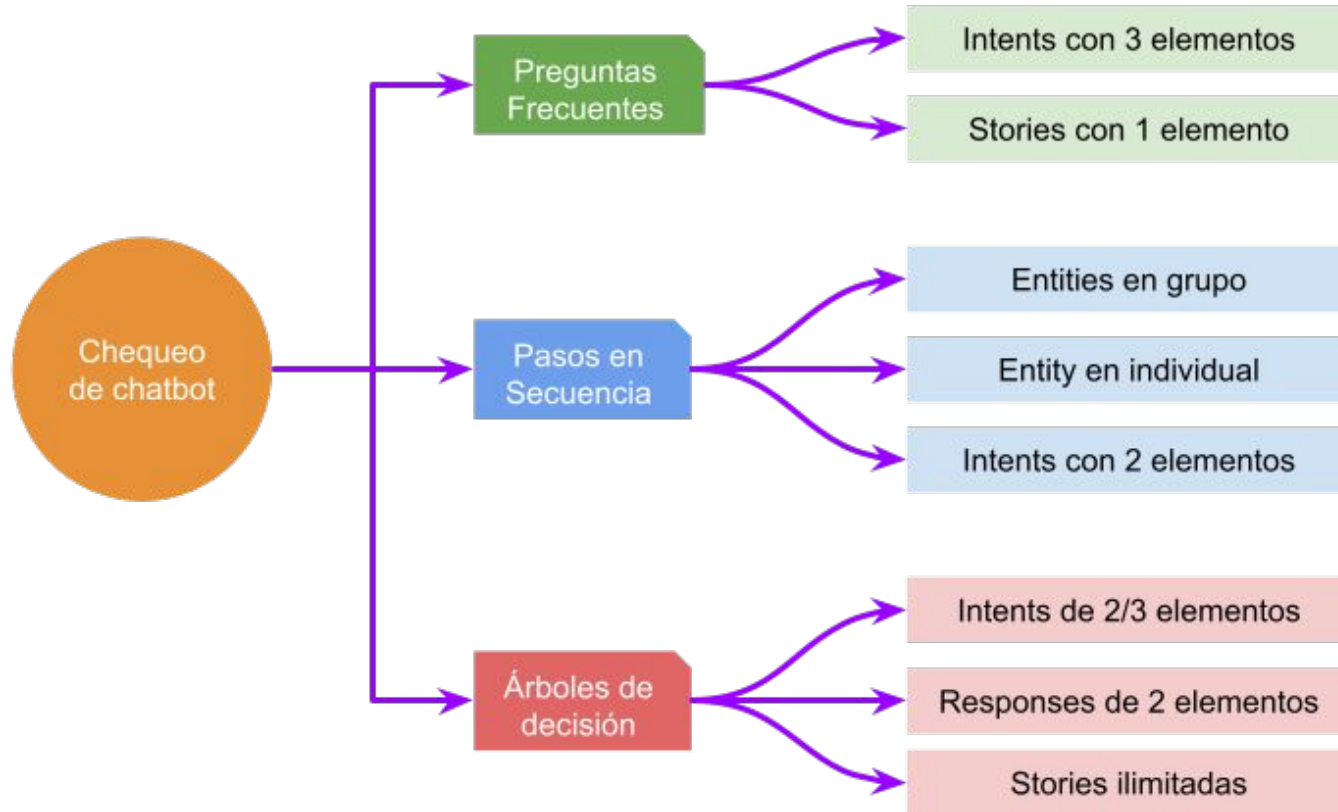


3.5 Conexión con NGROK y Telegram

Ejemplos de conversación usando Telegram Web



4. Evaluación y Chequeo de errores



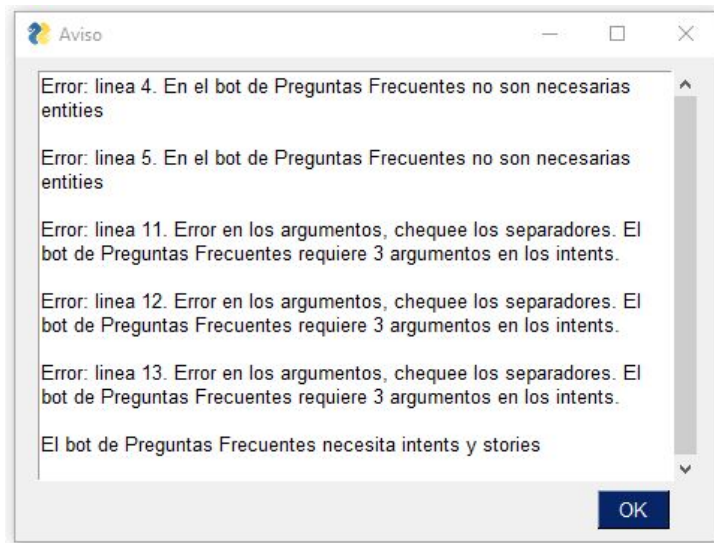
4. Evaluación y Chequeo de errores

Errores al crear un chatbot de
Preguntas Frecuentes con una plantilla
de Secuencia de Pasos

Detecta entidades

Parámetros incorrectos

Detecta que faltan elementos



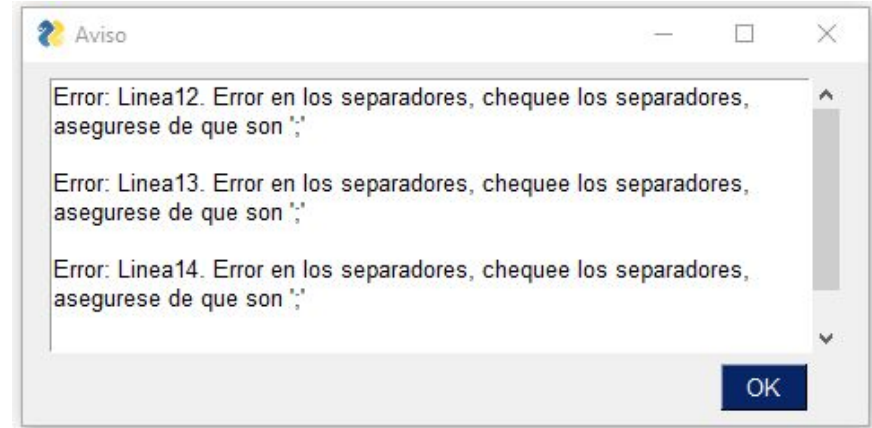
4. Evaluación y Chequeo de errores

Errores al crear un chatbot basado en Árbol de Decisión con una plantilla de Preguntas Frecuentes

Detecta errores con los separadores:

Esta plantilla se corresponde con la vista en la diapositiva 25, y las líneas con los stories.

Recordemos que este chatbot espera más de un elemento en la story, y esta plantilla solo tiene uno.



4. Evaluación y Chequeo de errores

Video demostración de uso:

<https://youtu.be/QZVsPXePDNc>

5. Conclusiones y trabajo futuro

Conclusiones

- Principio
- Sector en auge
- Nuevos retos

Trabajo Futuro

- Modificaciones
- Implementación nuevas versiones



ugr

Universidad
de **Granada**

CreaBot

Desarrollo de un asistente para la implementación y despliegue de sistemas conversacionales mediante la plataforma Rasa

Autor: Pablo Valenzuela Álvarez
Director: David Griol Barres

Grado en Ingeniería Informática