05/03/2024

Desarrollo de chatbots (RASA)

1. Que es RASA?

El Desarrollo de los chatbots es complcada porque se esperaban conversaciones muy estructuradas. Tenían los siguientes problemas:

* Buscaban coincidencias de patrones
* Respuestas predefinidas
* Carecen de aprendizaje profundo

Rasa es un framework para crear chatbots de código abierto, en pyton. Se basa su funcionamiento en:

* NLU: natural language undertanding: analizar el texto para comprender la entrada
* Core: tomar decisiones

Asocia las entradas de los usuasiro a cierto contexto (la etiqueta)

Arquitectura

Rasa tiene una versión free source y luego una pro (de pago) para el desarrollo en equipo

Proyecto

Un proyecto RASA necesita cierta estrcutra de directorios:

* nlu.yml: intenciones de los usuarios. Como por qjemplo saludar. Se puden especificar ejemplos de lo que va a poner un usuario
* stories.yml: indicar cuales son los cursos de acciones (intents y acciones)
* rules.yml: reglas de uso ocasional (comportamiento como chatbot antiguo). Por ejemplo: reconocer insultos y controlarlos directamente
* domain.yml: posibles respuestas que va a dar el asistente
* config.yml: fichero de configuración, donde se define que lógica va a tener el bot
  + Pipeline: las funciones que se utilizan para preprocesar el texto
  + Policies: técnicas para gestionar el flujo de la converscion

12/03/2024

Gestion de desvíos

Contemplar casos donde el usuario se sale de la converacion. RASA incluye dos mecanismos

* Mensajes fueras de contexto: un prompt que se aleja mucho de la conversación acticual. Para manejar estos casos se utilizan unas reglas. Estas reglas se imponen al propio modelo de RASA, tienen mas prioridad. Es recomendable utilizarlo de forma muy puntual.
  + Crear un intent del tipo (out\_of\_scope). Con muchos ejemplos
  + Definir un mensaje en el domain: el cual saldrá cuando se detecte
  + Crear una regla:
* Recuperaciones (fallback): Su funcionamiento consiste en clasificar el mensaje. Si un prompt no se puede realcionar con un intent, salta este mecanismo
  + Agregar FallBackClassifier en el config (con un treshold)
  + Definir una respuesta en el domain
  + Agregar una nueva regla en el rules.yml

Adicionalmente al fallback

**Acciones**

Acción personalizada para ejecutar código Python, pudiendo enlazar la aplicación con APIs, DBs, etc…

RASA además de asignar un intent a determindada entrada del usuario, también busca posibles entidades dentre de este mensaje

Las acciones customizadas se definen en el archivo actions.py

26/03/2024

Slots

Concepto de slots o ranuras: mecanismo para almacenar ciertos datos durante la conversación. Esta información se gurda durante el transcurso de la información y se recuperan en un momento determinado mas adelante.

Ejemplo:

Si un usuario indica la talla de la ropa, esta talla se guarda para luego utilizarla mas adelante. Se puede definir en el archivo nlu.yml con varios intents donde se reconozcan entidades. En el archivo de stories.yml se declara la variable donde se va a almacenar. En el domain.yml hay que declarar también las entities. También debe existir una sección slots con la variable. Finalmenete en el código de la acción se puede utilizar el valor almacenado en el slot

Formularios

Permite manejar varios datos al mismo tiempo.

Pasos a seguir

* Agregar ‘forms’ al domain.yml
* Paso 2
* Paso 3: asociar regla en rules.yml
* Paso 4: activar formulario en stories.yml

Durante el transcurso de un formulario también pueden ocurrir situaciones que desvíen el flujo del formulario:

* Referencias a otras intencioes
* Peticion de cancelación

Referencia a otras intenciones: utilizar reglas

Peticion de cancelación del formulario: requiere de los siguientes pasos

* Respuesta de cancelacion