Bot Inteligente recomendador de juegos

Implementado con Rasa y uso en telegram.



Materia : Investigacion Exploratoria

Alumno: Freire Agustin

Profesora: Analía Amandi

Ayudante: Santiago Faiella

Introduccion

En este informe se explicaran las funcionalidades básicas implementadas en el bot de Rasa, tanto las decisiones de posibles conversaciones, como la implementación de las acciones que se pueden realizar y como se guardan los datos, sin centrarse mucho en como se realiza sino el porqué se tomaron esas decisiones y las características que va a tener el bot en particular.

Desarrollo

Se buscó desarrollar un bot que recomiende juegos, y lo pueda hacer en base a las preferencias del usuario, además hacerlo con conversaciones que sean los más fluidas y humanizadas posibles, manteniendo un contexto cuando sea posible, además de utilizar varias herramientas distintas, tales como Prolog, pyhton , archivos csv entre otras. Este desarrollo fue hecho en un contexto donde se intentaba que todas las cosas necesarias fueran aprendidas mediante práctica e investigación del desarrollador por lo que se logró practicar de cierta forma el aprendizaje autodidacta.

Decisiones

Base de datos:  
Se utilizó como base de datos Prolog, que permite guardar la información de forma encapsulada, en este caso los juegos con el siguiente formato:  
juego(nombre,[categorías],link de compra, imagen, descripción, desarrollador , [palabras clave]).

Esto debido a las facilidades que da para manipular y recuperar esta información con las reglas, tales como devolver juegos con ciertas categorías, devolver datos sobre el juego o hasta devolver juegos en formato csv para poder luego utilizarlo para un Árbol de recomendación de juegos.

Perfil del usuario:

La misma se guarda en archivos csv, se tiene uno para todos los usuarios que guarda su id de telegram , nombre, categorías que le gustan, ultimo juego recomendado y si la conversación estaba empezada o no, todos datos para mejor la fluidez de las conversaciones. También se utilizan otros dos archivos, uno que guarda todos los juegos en formato csv que le gustaron o no al usuario, que se usan para entrenar el árbol de recomendación y otro que guarda de los juegos de la base de datos menos los que ya le fueron recomendados todos los que según la predicción del árbol deberían gustarle, esto se hace para calcularlos al finalizar una sesión y que permita agilizar la recomendación de los juegos.

Métodos de recomendación:

Se cuenta con 2, uno el más primitivo, utilizado para usuarios nuevos o para los cuales todavía no se tiene suficiente información de los juegos que le gustan que se basa simplemente que al gustar o no gustarle un juego se van modificando 3 categorías para el usuario y para recomendarle ese busca en Prolog un juego que tenga esas categorías en su lista de categorías y en caso de que no hayan más para recomendar se van dejando de tener en cuenta ciertas categorías, particularmente las ultimas. El segundo método usa un árbol con inteligencia artificial, al cual se le pasan los juegos que le gustaron o no al usuario como información de entreno y se le pide predecir para todos los juegos menos los que se usaron para entrenar, este método se usa solo cuando ya tiene más de 7 juegos de información sobre si le gustaron o no.

Existen otros métodos pero que no son según las preferencias del usuario sino que el mismo le pide expresamente que sean parecidos algún juego o que sean de alguna categoría en particular.

Slots:

categorías: se usa para guardar las categorías actuales que le gustan al usuario, para el método de recomendación primitivo

FueChau: se usa para guardar si el último mensaje fue un chau, para a la hora de cambiar perfiles, en grupos, se sepa si la conversación estuvo empezada o no.

usarArbol: indica si suficientes ejemplos para usar el árbol de recomendación o no

juegosTipoCSV: guarda todos los juegos que según el último modelo de árbol le deberían gustar al usuario

Juegos: guarda solo el nombre de los juegos que se le fueron recomendando al usuario (para el método más primitivo)

juegosGustan: se guardan en formato csv los juegos que le fueron gustando al usuario

juegosSesionActual: guarda en formato csv los juegos que se le fueron recomendando al usuario, para facilitar ciertas acciones.

juegosNoGustan: se guardan en formato csv los juegos que no le gustaron al usuario

nombre: guarda el nombre por el que quiere ser llamado el usuario, inicialmente el de telegram

id: guarda el usuario de telegram y se usa para los perfiles del usuario

Entidades

- categoría: para cuando el usuario debe indicar una o unas categorías

- juego: cuando el usuario indica el nombre de un juego

- username: para guardar el nombre de usuario si el mismo quiere que se le llame de otra forma

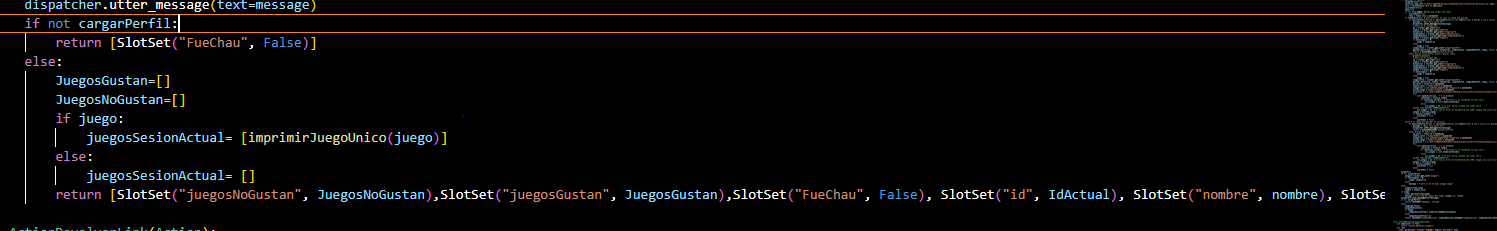
Como debería darse una conversación y cambio de perfil en grupos.

El bot está pensado para que las conversaciones se empiecen con un hola, para cargar el perfil o se cree uno, entonces se genere la conversación y al finalizar se le diga un chau para guardar el perfil y generar el árbol además de predecir los juegos que le pueden gustar. Por lo tanto si no se está en grupos y no se dice chau nunca se van a guardar los datos. En cambio como en grupos hay que hacer el cambio a mano, si se guardan igualmente los perfiles de los usuarios cuando habla otra persona distinta, pero no se genera el árbol solo se guarda el último juego para poder seguir la conversación y se indica si fue chau o no.

Para ver si se cambio se utiliza el siguiente código que esta en casi todas las acciones de Rasa en pyhton, lo cual resulta en mucha repetición de código pero por cómo se dan los return y el manejo de variables haría muy complicado su encapsulamiento.

Lo que hace es que si ve que si el id del mensaje actual es diferente al anterior (del slot) , si existe perfil y mensaje anterior diferente a chau guarda el perfil anterior y carga el nuevo, si el anterior chau solo carga el nuevo, lo cual significa cargar todos los slots con la informacion del perfil además de remplazarlos por los actuales. Ahora si no existía perfil del actual o el actual tenía como último mensaje chau entonces le indico que no estaban hablando y va al mensaje inicial del bot, que es cuando le mandan un hola, además puede tener que guardar el perfil anterior o no.

Esto es al final por si hay que cargar todos los slots o no



Y al inicio va el siguiente codigo que basicamente hacer lo de arribar:



Explicación de las distintas intenciones existentes asi como las acciones que desencadenan y algunos de los ejemplos de entrenamiento:

Intents:

- inicioSesion: detecta saludos o el /start del mensaje inicial, y ejecuta la acción inicial, que primero sigue una lógica de cargar perfil o crearlo parecido a al mostrado anteriormente:



Para crear el perfil se agrega al csv perfiles y se crea un csv llamado data\_(Id del usuario) donde luego se guardan los juegos que le van gustando y otro llamado juegos\_(id del usuario) donde se pondrán los juegos predichos por el árbol, ambos vacíos. Y si existía se carga el nombre del usuario y los juegos predichos.

Luego se fija si hay algún juego predicho del árbol, en ese caso el booleano de usar árbol es true sino es false.

Si verdadero agarra un juego random de esa lista y lo recomienda, sino pide a una función de prolog que le devuelva juegos en base a las categorías( que puede tener si ya tenía perfil sino es vacío por lo tanto recomendaría todos) y después saca de esos juegos los que ya le recomendó, elije uno random y se lo recomienda al usuario, si ya tenía perfil da un mensaje más casual y si no se presenta, además pide con el nombre del juego el link a la imagen del mismo para mostrársela al usuario: 

Y ya para finalizar setea slots que va necesitando:

 if not nuevoUsuario:

            return [SlotSet("FueChau", False), SlotSet("id", Id), SlotSet("nombre", user\_name), SlotSet("juegos", juegos), SlotSet("usarArbol", usarArbol), SlotSet("categorias", categorias), SlotSet("juegosSesionActual", juegosActuales), SlotSet("juegosTipoCSV", csv\_juegos)]

        else:

            JuegosGustan=[]

            JuegosNoGustan=[]

            nombre = nombre\_valor

            return [SlotSet("juegosNoGustan", JuegosNoGustan),SlotSet("juegosGustan", JuegosGustan),SlotSet("FueChau", False), SlotSet("id", Id), SlotSet("nombre", nombre), SlotSet("juegos", juegos), SlotSet("usarArbol", usarArbol), SlotSet("categorias", categorias), SlotSet("juegosSesionActual", juegosActuales), SlotSet("juegosTipoCSV", csv\_juegos)]

Una vez la acción iniciar o que se recomendó un juego pueden suceder varias intenciones pero los más normal son las que piden información extra del juego que son 2, además a partir de ahora todas la acciones tiene la parte de reconocer si hablo otra persona y cambiar usuario por lo tanto no se volverá a explicar pero siempre estar al inicio:

- quiere\_categorias: el usuario le pide que le diga las categorías que tiene el juego anteriormente recomendado, para esto pasa el nombre del juego a prolog y una regla devuelve una lista con las categorías:

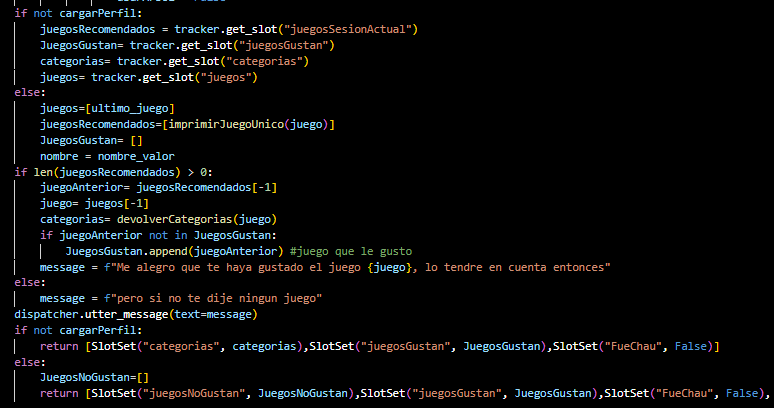


- quiere\_sinopsis: el usuario le indica que le de una sinopsis o descripción del juego anteriormente recomendado, para esto pasa el nombre del juego a prolog y una regla devuelve la descripción:



Ahora el curso esperado es que le diga que el juego le gusto o que no le gusto:

- le\_gusto: el usuario indica que le gusto el juego, por lo que se le dice al usuario que se tendrá en cuenta esa información y se guarda tanto en la lista de juegos que le gustaron, como que se buscan sus categorías en prolog y se asignan en la lista de categorías para la recomendación primitiva, es la acción setear categorías:

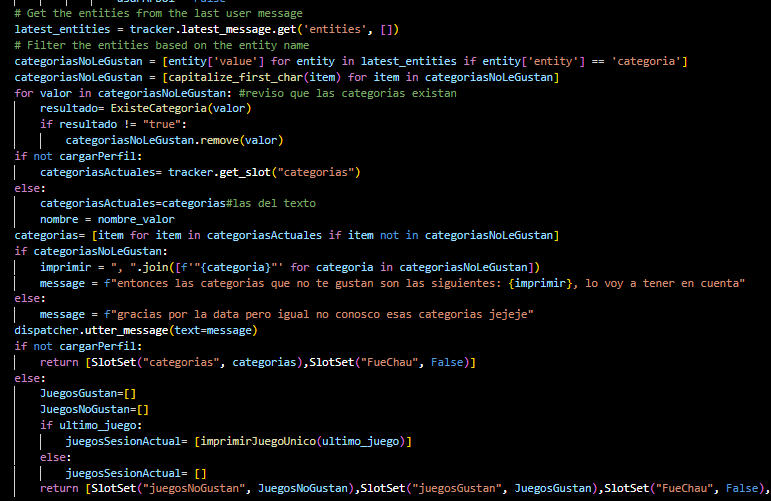


El siguiente curso esperado es el intent - quiere\_link\_compra ya que le gusto se espera que te pida donde comprarlo o el link de compra, pasa eso usa una regla de prolog que le pasa el nombre y le devuelve el link de compra:



Tambien podría pasar que no le guste por lo tanto antes el usuario le diría o Que no le gusto con el intent: - no\_le\_gusto o que le gustan otro tipo de juegos con el intent - le\_gustan\_otro\_tipo ambos se usan para detectar que no le gusta, por lo tanto la acción es preguntar categorías donde se le preguntar cuales no le gustaron y se guarda el juego en no le gustaron:

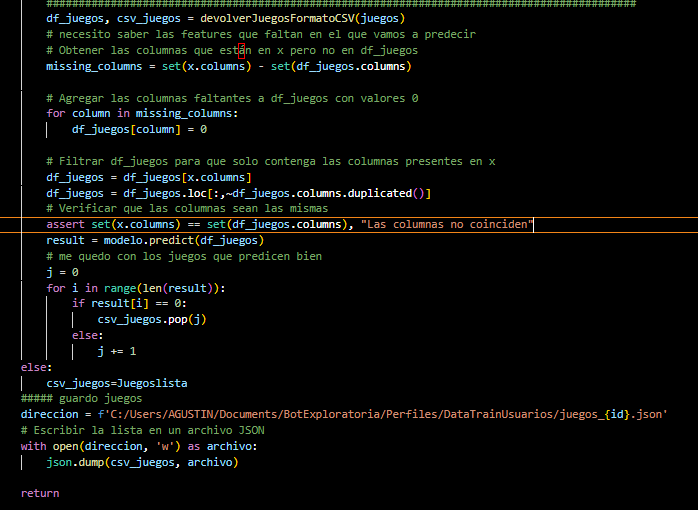


Entonces luego de esto se espera que le indique cual es la categoría mediante el intent - indicar\_categorias : donde se fija en la entidad que categorías le dijo, verificando que sea una existente en con una regla de prolog y de esas se eliminan de la lista de categorías: 

Y luego se espera que se despida, con la intent – goodbye que lo que hace es guardar el perfil, además de los juegos que le gustaron y que no, y generar el modelo del árbol, predecir los juegos y guardarlos en el csv:



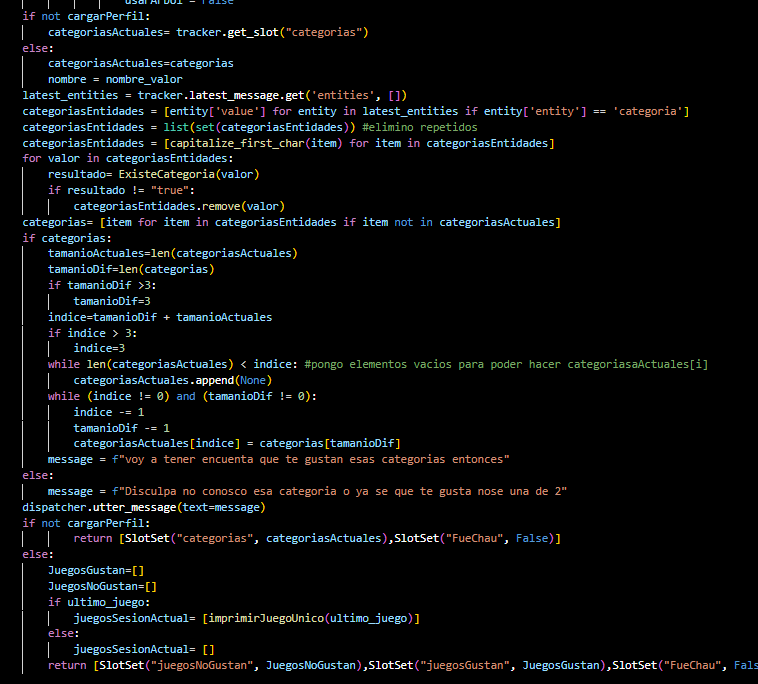




Esta función se la llama desde la acción guardar perfil.

Hay algunas intenciones y acciones también existentes que son adicionales:

- le\_gusta\_categoria se usa para que el usuario pueda decir de por si le gusta una o unas categorías y las mismas se ponen en la lista de categorías, y se sacan de atrás para adelante por cómo se consideran más importantes:



- recomendame\_un\_juego: el usuario puede pedir que se le recomiende un juego, esto lo hace según las preferencias del usuario, si está usando árbol entonces le recomendara de la lista antes calculada y sino buscara juegos con las categorías que le gustan al usuario en prolog, a eso le saca los juegos ya recomendados para no repetir:

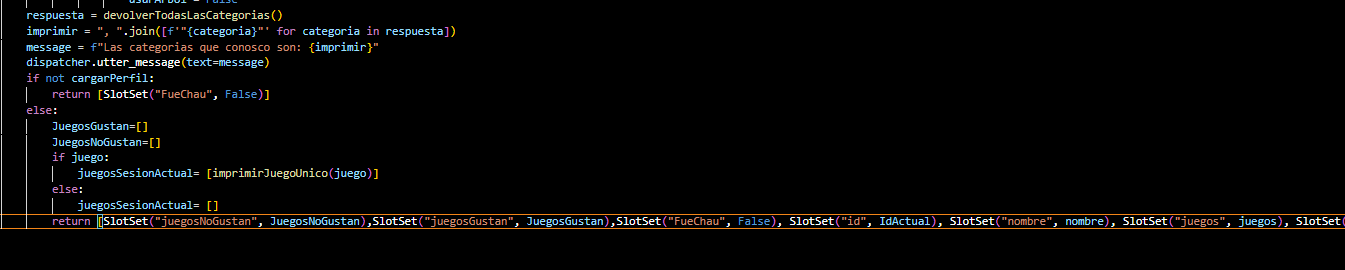


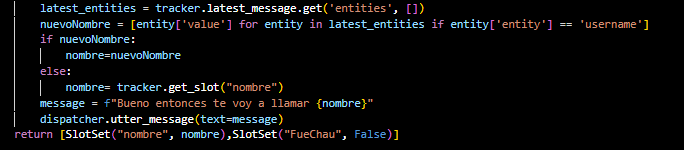
- quiere\_juego\_parecido el usuario al recomendarle un juego puede pedir que el recomiende otro juego parecido al mismo, para eso saca las categorías de prolog y busca un juego con esas categorías, algunas de esas categorías: 

- recomendame\_un\_juego\_en\_base\_a\_categoria: el usuario dice que quiere que se le recomiende un juego en base a una o unas categorías, se detectan con una entidad y se usan en prolog para recuperar los juegos con esas categorías: 

- recomendame\_un\_juego\_en\_base\_a\_juego: parecida la lógica al anterior pero el usuario indica un juego, a este se consiguen sus categorías con prolog, entonces se busca un juego con esas categorías, en tanto este como el otro se busca no repetir juegos: 

- recomendame\_un\_juego\_random: el usuario puede pedir que se le recomiende un juego random, para esto simplemente pide un juego de prolog, sin pasarle categorías, es decir da cualquier juego y de ese se elije uno random que no se haya recomendado antes y se lo da al usuario: 

- quiere\_saber\_todas\_las\_categorias: el usuario pide que se le indique todas las categorías que sabe el bot, esto es una regla de prolog que da todos los elementos de una lista que contienen las categorías registradas: 

- cambiar\_nombre: el usuario puede decirle al bot que quiere que este lo llame de otra forma, por lo que cambia el nombre del usuario que tiene registrado en el perfil, para esto usa una entidad: 

- LuLu: es solo una interacción graciosa ya que el bot dice que sus amigos le dicen LuLu por lo tanto si le decís LuLu te va a decir que no sos su amigo y va a dar una foto de la persona que representa el bot.

- LuisLuis: es otra interacción graciosa, ya que te dice que el bot se llama LuisLuis si le decís por ese nombre te va a decir Yo.

- agradeció: Si el usuario le dice gracias al bot solo te va a decir de nada y va a mostrar una foto.

- nlu\_fallback: se utiliza automáticamente cuando no entiende lo que le dice el usuario, es decir no lo encasilla con ninguna intención, por lo que dice que no entendió el mensaje y que por favor repita el mensaje.

Conclusiones

Para finalizar se puede concluir que el resultado cumple con las expectativas de un bot que puede entender varias intenciones y mensajes, tanto propias de los juegos a recomendar y algunas otras más generales, a su vez el mismo tiene respuestas que pueden considerarse humanizadas o que no dan la sensación de hablar con un bot. Además todo el proceso de aprender las distintas herramientas utilizadas para su creación fue muy enriquecedor e interesante por lo que fue una muy buena experiencia.

Bibliografia:

<https://rasa.com/docs/>

<https://forum.rasa.com/>