

Chatbot Para Supermercado Mayorista

Rasa Framework

Alumno: Tomás Agustín Rodriguez

Profesora: Analía Amandi

Ayudante: Santiago Faiella

Unicen Tandil-Facultad de Ciencias Exactas

# Introducción

En este informe se aborda el proceso de desarrollo de un agente inteligente de un supermercado mayorista. El mismo se llevó a cabo a través de la utilización de **Rasa**, un framework de código abierto para el desarrollo de chatbots y asistentes virtuales que utiliza el procesamiento del lenguaje natural (NLP) y el aprendizaje automático para comprender y responder a las conversaciones de los usuarios de manera contextual y personalizada.

El objetivo es lograr una adecuada implementación del asistente virtual de manera que pueda mantener una conversación estable con los clientes del supermercado mayorista, con el fin de satisfacer sus necesidades, asegurando un trato personalizado con cada uno de ellos para mejorar la experiencia.

# Desarrollo

### Registro de clientes

La **funcionalidad básica** del agente es responder las consultas de los clientes, generalmente asociadas al stock del supermercado, pero antes de llegar a esa etapa de la conversación, el agente le da la posibilidad al cliente de crearse una cuenta de cliente personal, esto con el propósito de emplear una **personalización de usuario** para que la información que se le brinde al mismo sea acorde a sus gustos y necesidades.

El procedimiento de registro de clientes consta de la siguiente manera, el agente solicita datos personales al cliente (número de documento, nombre, apellido, fecha de nacimiento y email), los cuales serán cargados en una **base de datos** basada en un archivo **JSON** (JavaScript Object Notation), un formato de intercambio de datos ligero y de fácil lectura que se utiliza comúnmente para representar datos estructurados y que es independiente del lenguaje de programación. En dicho archivo se almacenan los datos de todos los clientes que se encuentren logueados. No obstante, se crea un **archivo.csv** con los campos “nombre”, “marca” y “gusta” sin ninguna tupla, del cual se hará mención acerca de su funcionalidad posteriormente cuando se hable sobre la personalización del usuario, al igual que de los contadores de consultas de categorías de productos almacenados en cada registro de cliente del JSON. Cabe aclarar que el número de documento del cliente sirve como identificador de los registros de la base (JSON) y como nombre del archivo.csv (<número de documento del cliente>.csv).

En caso de que el usuario no desee loguearse, el agente estará a disposición y atenderá las consultas correspondientes que se presentan a continuación.

### Consultas de Stock

Un cliente puede consultar acerca de los productos que se encuentran en stocks, brindando el nombre y la marca del producto de interés, obteniendo como respuesta el nombre, precio y el link que redirige a la página donde pueden comprarse de manera online aquellos productos que estén presentes en el stock del supermercado. El agente se conecta con la API de MercadoLibre, realizando un request con el nombre del producto y la marca a [https://api.mercadolibre.com/sites/MLA/search](https://api.mercadolibre.com/sites/MLA/search%20) para obtener los productos a mostrar.

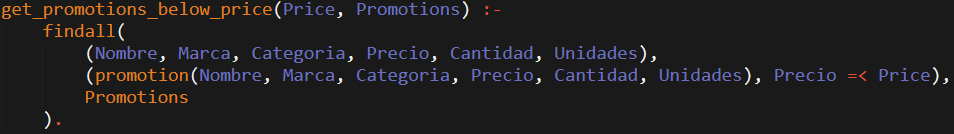
Además, los clientes pueden consultar acerca de promociones asociadas a ciertas categorías como lácteos, galletitas, limpieza, repostería, bebidas, panadería, carnicería y verdulería, o preguntar por productos en promoción que tengan un precio menor o igual a un valor monetario determinado, recibiendo como respuesta una lista de productos con sus respectivos nombres, marcas, unidades necesarias a llevar para la aplicación de la promoción, cantidad (en gramos/mililitros) y precios.

A diferencia de las consultas sobre productos que se hayan en stocks, los productos asociados a promociones se encuentran en una **base de conocimiento** creada con **Prolog**, un lenguaje de programación que se utiliza principalmente para la programación lógica y el procesamiento del lenguaje natural, el cual es empleado en una variedad de aplicaciones, como la inteligencia artificial, la inferencia lógica, la representación del conocimiento y la resolución de problemas basados en reglas.

La base contiene un conjunto de reglas y hechos que permiten representar las promociones y acceder a ellas de diferentes formas. En la *figura 1* se puede apreciar el formato de los hechos con los que se definieron las promociones, y en la *figura 2* y *figura 3* las reglas que permiten obtener la información de los productos definidos.

*figura 1. Formato de los hechos definidos en la base de prolog.*

*figura 2. Regla que retorna promociones en función de una categoría específica.*

*figura 3. Regla que retorna promociones con precio menor o igual a un valor dado.*

### Personalización del usuario

Al comienzo de la conversación el asistente virtual saluda al cliente y le pide amablemente que ingrese su número de documento para que se loguee y así poder cargar los datos que se encuentran en el registro del JSON vinculado al cliente en los slots de rasa (todos sus datos excepto los contadores). Si el cliente no posee una cuenta de cliente personal el asistente le brinda la posibilidad de crearse una como se mencionó anteriormente, y procede a cargar los datos de igual manera.

Una vez cargados los datos, el asistente procede a mostrar un mensaje de bienvenida personalizado con el nombre y apellido del cliente, junto con la **“promoción del día”**. Es aquí donde se usan los contadores que se crearon en el registro del cliente asociados a las categorías consultadas por el usuario. La función de estos contadores es llevar el conteo de la cantidad de veces que el usuario consultó sobre una categoría en especial, para que a la hora de mostrar la “promoción del día” la misma se base en la categoría más frecuentada por el usuario en cuestión. El agente se conecta con la API de mercado libre antes presentada y realiza un request con el nombre de la categoría más consultada para poder visualizar el producto (nombre del producto, precio y link que redirige a la página de compra).

Existe un día especial en el que no solo se muestra la “promoción del día” y es la **fecha de cumpleaños del cliente**, donde el agente genera un código de 6 dígitos alfanuméricos aleatoriamente y presenta un mensaje que felicita al cliente por su cumpleaños junto con el código, informándole que puede utilizarlo para obtener un 10% de descuento en su próxima compra.

En esta instancia (luego de que el usuario vea la “promoción del día” junto con el mensaje de felicitación por su cumpleaños con el código de descuento si corresponde), el cliente puede realizar las consultas de stocks ya explicadas, teniendo en cuenta que, cada vez que consulte sobre promociones de una categoría específica se incrementa el contador de tal categoría de su registro del JSON en una unidad, además de que al consultar sobre la existencia de un producto en stock se añade al archivo.csv relacionado al cliente una tupla con los valores “nombre del producto”, “marca del producto”, “1”. De igual manera, el cliente puede informar al asistente sobre el poco interés que le genera un producto en particular, brindando el nombre y la marca, lo que provoca que se añada una tupla al archivo.csv con los valores “nombre del producto”, “marca del producto”, “0”.

Seguramente usted se pregunte con que finalidad se cargan tales tuplas al archivo.csv, y la respuesta es que las mismas se utilizan para realizar **predicciones** con respecto a productos que pueden ser o no del agrado del cliente. El cliente tiene la capacidad de solicitar sugerencias al chatbot acerca de determinados productos, recibiendo como respuesta si el mismo es o no de su agrado, es decir, si el cliente no suele consultar por bebidas alcohólicas e incluso ha informado que no le interesan productos de tal rubro, lo más probable (por no decir obvio) es que si el cliente solicita una sugerencia acerca de una bebida alcohólica el asistente le informe que no es de su agrado.

Para poder llevar esto a cabo, el agente al recibir una solicitud de sugerencia de un producto (recibiendo el nombre del producto junto con la marca por parte del cliente) genera un modelo de **árbol de decisión** con el que realiza la predicción. Para ello, el agente pre procesa los nombres y las marcas de los productos del archivo.csv, transformándolos a enteros debido a que el algoritmo funciona con enteros como entradas, selecciona las variables (las columnas “nombre” y “marca” son las features y la columna “gusto” es el target), crea el modelo y lo entrena con las variables. Luego, el agente verifica si el nombre y la marca del producto en cuestión se encuentra en el archivo.csv del cliente para obtener el valor entero correspondiente. En caso de que el producto solicitado no se encuentre en el archivo.csv se transforman el nombre del producto y la marca a nuevos valores enteros y se crea un dataframe con ambos valores, el cual usa el modelo del árbol de decisión para realizar la predicción, para así en función del resultado, informarle al cliente si el producto es o no de su agrado.

### Flujos de conversación

Ya habiendo citado las funcionalidades del chatbot, se puede presuponer de cierta manera como es el flujo de conversación. La conversación comienza con el saludo por parte del bot al cliente, el cliente si lo desea se loguea, siempre y cuando sea propietario de una cuenta, caso contrario decide si se creará una o no. Si opta por no crearse una cuenta, el siguiente flujo de la conversación tiene al cliente como emisor de consultas, donde el chatbot se abstendrá simplemente a escucharlas y responderlas a su debida manera. Si procede a crearse una cuenta, luego de cargar los datos que el asistente le solicita, se visualiza la promoción del día o la felicitación por su cumpleaños con el código de descuento si corresponde, y el cliente puede plantear sus necesidades.

Algunos aspectos secundarios a considerar son la capacidad del cliente para despedirse, dar las gracias o remarcar que una respuesta no fue de utilidad, las cuales el bot contestará adecuadamente.

# Aclaraciones

* En el archivo domain.yml se hayan utters que no se utilizan en ningún flujo de conversación que se decidieron no eliminar para su uso en futuras implementaciones.
* El email de los clientes no abarca ninguna funcionalidad, simplemente se decidió no eliminarlo para su uso futuras implementaciones.
* En el archivo.nlu, a la hora de definir valores de entidades con espacios se optó por utilizar ‘\_’ en lugar de ‘ ’ debido a los problemas de implementación referidos al entendimiento del intent por parte del chatbot que conllevaba la presencia de espacios. De todas formas, existen algunos valores en lookup tables con espacios simplemente porque en los mismos el chatbot no tiene problemas para reconocerlos, solamente si la primera palabra no se encuentra repetida en otro valor de lookup table (por ejemplo, si existe “tapa de asado” y “tapa de cuadril” el chatbot no reconocería el corte de carne porque encuentra dos valores distintos a partir de tapa, pero si solo existe “tapa de asado” lo reconocería sin problemas).
* Por lo antes mencionado, se escogió categorías de productos que no contemplen espacios para evitar el uso de ‘\_’ en vez de ‘ ’ y no tener que indicarle a Prolog con ‘’ que el valor de la categoría contiene espacios cuando se consultan promociones por categoría.
* El código que se genera cuando es el cumpleaños de un cliente es ficticio, se trató de simular un descuento de regalo para el cliente debido a que la API de Mercado Libre no permite modificar los precios de los productos.
* La primera “promoción del día” que vea el cliente al registrarse por primera vez no presenta una respuesta personalizada porque no se tienen los datos correspondientes para realizarla, es decir, los contadores de las consultas de las categorías están inicializados en cero, todas tienen el mismo valor.
* Se normaliza el valor de la categoría a lower case al realizar una consulta de promociones por categoría para evitar problemas de compatibilidad con la definición del valor de la categoría en los hechos de Prolog.
* Al momento de crear el árbol se genera un gráfico que muestra el árbol de decisión para poder visualizarlo en caso de que se desee.
* Se seleccionó una profundidad de 3 a la hora de crear el árbol por la poca cantidad de datos de prueba que se cargaron en los archivos.csv de los clientes creados para probar la funcionalidad. Una profundidad de 5 por ejemplo probablemente sea una mejor opción para flujos de datos más grandes.
* Cuando el nombre y la marca del producto que el usuario solicita sugerencia no se encuentran en el archivo.csv vinculado al mismo, los nuevos enteros que se generan son valores que no se encuentran en las variables features correspondientes a cada valor (específicamente se asigna el valor siguiente al más alto de la feature, es decir, si el valor para “nombre” más alto es 6, entonces al nombre del producto solicitado se le asigna el valor 7, ídem con la marca en función del valor más alto de “marca “en feature).
* A criterio del autor de este informe, no se encontró una utilidad esencial para el árbol de decisión planteado, anteriormente se planteó uno que si era útil y que era único para todos los clientes pero que requería cambiar el contexto del problema de cierta manera, por ende, se seleccionaron el nombre y la marca de los productos como features ya que existe cierta coherencia con respecto a las predicciones. Por ejemplo, si un cliente indica al agente que no le gusta una cerveza de la marca X lo más probable es que si el cliente solicita al agente que le sugiera sobre una cerveza de la marca Y, aunque el cliente no haya indicado en el pasado que no es de su interés el agente le conteste que no va a ser de su agrado. Lamentablemente, éste prototipo de árbol de decisión es univoco para cada cliente debido a la subjetividad de los gustos de cada persona.
* A causa de la poca cantidad de datos de prueba en los archivos.csv de los clientes, se decidió entrenar el árbol cada vez que el cliente solicite una sugerencia de producto para predecir con el árbol con más datos cargados (aumenta la tasa de confiabilidad por sobre el modelo).
* Anteriormente se había implementado two\_stage\_fallback para mejorar la experiencia y fluidez de la conversación con el chatbot pero se retiró su implementación debido a que traía problemas asociados al reconocimiento de patrones de texto del chatbot que sin él no ocurrían (sin importar si la confianza sobre el intent era alta el chatbot ejecutaba la acción default\_fallback).
* Como el chatbot tiene problemas para reconocer flujos de stories muy cortas, la mayoría de los flujos (que son pequeños) se encuentran en el archivo rules.yml donde el chatbot los reconoce sin problemas, y los flujos largos permanecen en el archivo stories.yml.
* Se utilizó la API de Mercado Libre como la base de datos que almacena los productos de stock para separar los productos de promoción que se encuentran en la base de conocimiento de Prolog, además de la flexibilidad que otorga tal herramienta a la hora de realizar cualquier tipo de consulta.

# Conclusión

En conclusión, la versión actual del asistente virtual cumple con los objetivos establecidos a principio de la cursada 2023 de la materia Programación Exploratoria. Sin embargo, el potencial del bot es considerable y puede ser explotado aún más con futuras mejoras y extensiones en su implementación. Se puede considerar esta versión del bot como un reflejo de lo que el framework Rasa y sus acciones personalizadas nos permiten modelar.

# Bibliografía

<https://rasa.com/docs/rasa/>  
<https://developers.mercadolibre.com.ar/es_ar/guia-para-producto>