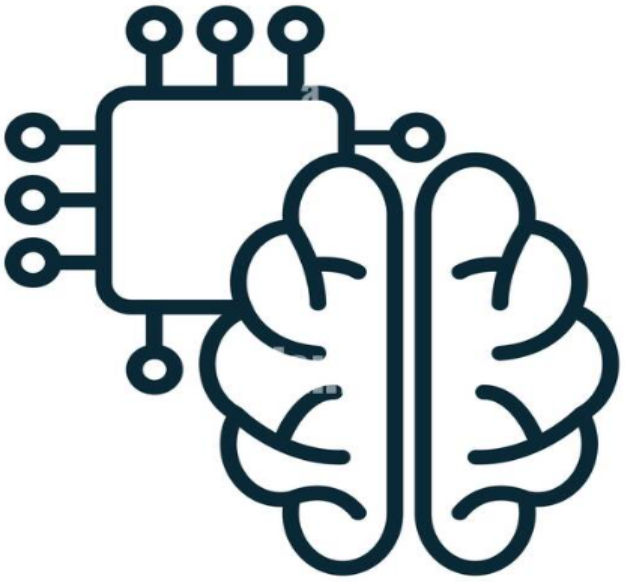
Universidad Nacional del Centro de la
Provincia de Buenos Aires

Facultad de Ciencias Exactas

Ingeniería de Sistemas
Programación Exploratoria

Desarrollo de ChatBot



Profesora a Cargo: Analía Amandi

Alumno:
Martín Vazquez Arispe
martin.vazquez.arispe@gmail.com

Índice

Introducción

La inteligencia artificial (IA) se está introduciendo cada vez más en nuestras vidas, al punto de que existen inteligencias tales que pueden asesorar a un abogado, actuar como repositor en un depósito y/o supermercado, captar aspectos no verbales de una llamada de emergencia para poder determinar la urgencia del acontecimiento, entre otras varias aplicaciones. Es más, esta ciencia es vital para la creación de chatbots capaces de generar conversaciones más “humanas” con los usuarios.

Un chatbot es un método de comunicación digital entre un humano y un robot, y este puede ser tanto físico como virtual. Actualmente, son utilizados por gran cantidad de empresas dado que tienen la virtud de poder atender masivamente a las consultas de los clientes y automatizar la interacción con los mismos. A su vez, pueden tener un trato impersonal o personalizado con el usuario. En este último caso, se identifica al humano para así lograr el trato esperado. Además, pueden tener la capacidad de reconocer y/o capturar el estado de ánimo del consumidor.

Por otra parte, para que un chatbot funcione como tal, tiene que tener como mínimo la capacidad de identificar las intenciones del usuario, determinando previamente cuáles podrían ser dichas intenciones. Como por ejemplo, consultar cuál es el horario en el que abre una sucursal, informarse acerca de las características o disponibilidad de algún producto, solicitar algún servicio, etc.

Sumado a esto, con el propósito de que el chatbot cumpla su objetivo, es necesario entrenarlo para que una vez en funcionamiento pueda reconocer las intenciones del cliente y actuar de manera eficiente frente a ellas. Este proceso de entrenamiento es conocido como machine learning. Otra característica importante de este agente de inteligencia artificial, son las entidades. Estas se encargan de capturar datos relevantes del mensaje para ofrecer una mejor y más acertada respuesta. Por ejemplo, si el usuario quiere saber el horario en que se dicta alguna materia, antes de dar una respuesta es indispensable capturar el nombre de dicha materia para así saber a cuál se está refiriendo.

En este informe se buscará relacionar el funcionamiento del chatbot con el concepto de machine learning, que es una técnica que se encarga de construir los módulos de aprendizaje automatizados para que así progresivamente logren aprender en base a los datos que va analizando. En nuestra vida cotidiana, este método se utiliza en una gran cantidad de servicios brindados por distintas empresas, algunos de ellos son recomendaciones de películas de plataformas digitales, reconocimiento del habla de asistentes virtuales, coches autónomos, etc.

En efecto, este informe desarrollará la presentación de un chatbot que me representa a mi como alumno, aplicando diversas interacciones con el usuario. Estas interacciones se refieren a un trato general en caso de no conocer al usuario, un trato específico si es una persona conocida por el agente y además, cuenta con manejo de contexto. Paralelamente, contiene una base de conocimiento escrita en el lenguaje Prolog.

Descripción General del Chatbot

Para comenzar esta presentación, hay que iniciar hablando de con qué herramienta se llevó a cabo el chatbot. Este fue desarrollado a partir de rasa, que es un framework para la creación de asistentes y chatbots, escrito en python y de código abierto. Para conocer el código en detalle, se deja a continuación el enlace al repositorio: GitHub

Este agente cuenta con la capacidad de responder diversas consultas, empezando desde la más sencilla como el saludo hasta una más compleja, como mi opinión sobre una materia determinada o responder manteniendo el contexto. En esta sección del informe se va a presentar las intenciones y respuestas de diálogos donde no intervienen otras funciones más avanzadas y, además, las stories y rules del agente. Al finalizar esta primera parte, se mostrará como se lo conectó con telegram, para poder tener una mejor interfaz al usarlo y lograr que este pueda desenvolverse en un grupo con otros chatbots.

En relación al saludo, este cuenta con un intent básico donde lo que se busca es reconocer los posibles saludos del usuario, los cuales probablemente sean el inicio de la conversación. Además, se planteó un response o respuesta que se utilizan ante la activación de este intent. A continuación, se muestra las imagenes del código fuente:

```
- intent: saludar
  examples: |
    - hola
    - hola, como andas?
    - que tal?
    - buen dia
    - buenas tardes
    - buenas noches
    - buenos dias
```

Intent: saludar (imagen 1)

```
utter_saludar:
- text: "Hola!, No me dirias tu nombre y dni"
- text: "Buenos Dias!, ¿Cual es tu nro de dni y tu nombre?"
- text: "Hola!, ¿Como te llamas y cual es tu dni?"
- text: "Buenos Dias!, ¿como es tu nombre y cual es tu dni?"
```

Response: saludar (imagen 2)

De igual manera, la despedida tiene su propio intent para reconocer cuando el usuario se retira y un response que, en este caso, se diferencia al del saludo dado que responde de acuerdo al estado de ánimo del usuario (esto, se hablará en la sección computación afectiva) y en el mensaje incluye el nombre de la persona con quien está hablando (este se hablará en al sección de Trato General y Trato Personalizado).

```
intent: adios
examples: |
  - chau
  - hasta luego
  - nos vemos
  - que tengas una linda tarde
  - hasta pronto
  - adios
  - cuidate
  - saludos cordiales
  - un placer
```

Intent: adiós (imagen 3)

```
utter_adios:
- condition:
      - type: slot
        name: estadoAnimo
        value: false
  text: "Hasta la proxima!, que mejore tu dia {nombre}"
- text: "Hasta Luego {nombre}!"
- text: "Nos vemos {nombre}"
```

Response: adiós (imagen 4)

En referencia a los datos personales, este bot cuenta con la capacidad de responder consultas sobre el deporte que realizo, mi edad, mi lugar de nacimiento y las áreas de la informática que me interesan. Estos fueron implementados de igual manera que el saludo y la despedida, con la única diferencia de que el response de las áreas de la informática que más me atraen, tiene una respuesta personalizada para el caso de que la persona con la que esté hablando sea un profesor, ya que si es así responde más formal.

```
# intent Personales

- intent: consultaDeEdad
  examples: |
    - Cuantos años tenes?
    - Que edad tenes?
    - edad?
    - cuanto años tenias?

- intent: hacerDeporte
  examples: |
    - Haces Deporte?
    - Realizas a algún deporte?
    - Deporte?
    - Haces alguna actividad deportiva?
    - Practicas deporte?
    - Haces algun deporte?

- intent: lugarNacimiento
  examples: |
    - Donde naciste?
    - De donde sos?
    - De que ciudad sos?
    - En que ciudad naciste?
    - De donde eres?

- intent: areasInformatica
  examples: |
    - que areas de informatica te interesan?
    - que te gusta de la informatica?
    - de informatica, que areas te gustan?
```

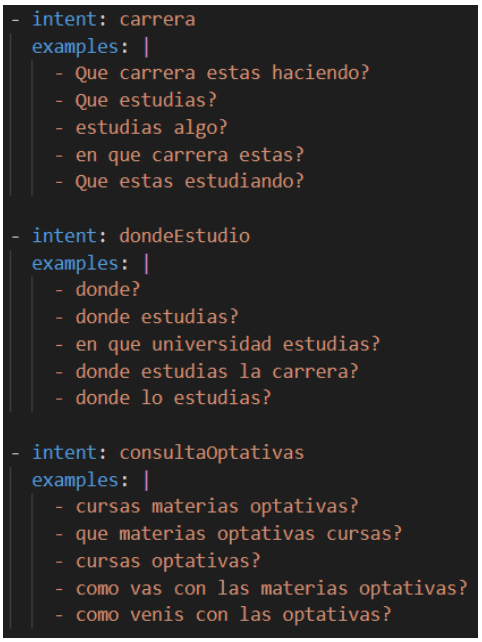
Intens: consultaDeEdad, hacerDeporte, lugarNacimiento y areasInformatica (imagen 5)

```
# Responses Datos personales

  utter_consultaEdad:
  - text: "Tengo 20"
  - text: "20 años"
  - text: "Tengo 20 años"

  utter_consultaDeporte:
  - text: "Hago Natación"
  - text: "Si, Realizo Natación"
  - text: "Practico Natación"

  utter_lugarNacimiento:
  - text: "Nací en Tandil"
  - text: "Soy de Tandil"

  utter_zonaInfor:
  - condition:
      - type: slot
        name: esProfesor
        value: esProfesor
    text: "Las areas que mas me interesan son desarrollo de
                aplicaciones de escritorio y seguridad informatica"
  - text: "Me gustan desarrollo de aplicaciones y seguridad informatica"
```

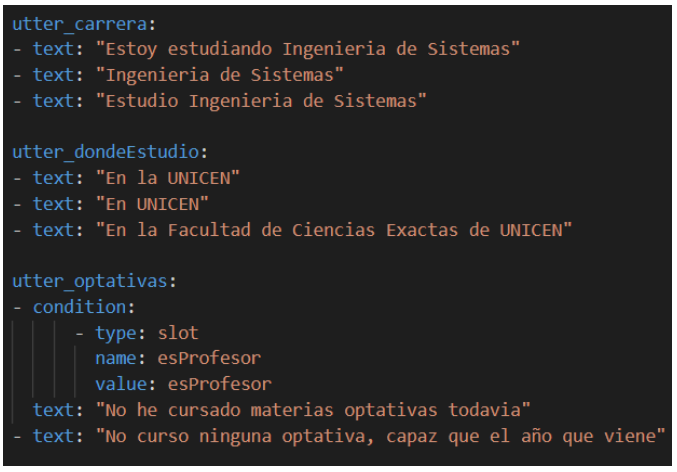
Responses: consultaDeEdad, hacerDeporte, lugarNacimiento y areasInformatica (imagen 6)

Sumado a esto, el bot cuenta con la facultad de responder ciertas preguntas académicas. Desde dónde estudio y qué carrera realizo, hasta mis opiniones sobre materias y la condición de mis materias cursadas o no, en aprobadas o no aprobadas. En esta parte, no se desarrollarán las intenciones en las cuales interviene la base de conocimiento, sino que estas son todas las referidas a materias que curso, materias aprobadas y opiniones sobre materias. Además, en la sección de “Manejo de contexto” se expondrá el intent que consulta cómo voy en la carrera.

A partir de ahora, se exhibirán las intenciones sobre qué carrera estudio, donde estudio y sí estoy cursando materias optativas. Al igual que en los casos anteriores, se implementó mediante intents y sus respectivos responses. De la misma forma que el response de áreas de la informática que me interesan, el response de si curso materias optativas tiene diferentes respuestas en caso de estar hablando con un profesor o no.



Intents: carrera, dondeEstudio y consultaOptativas (imagen 7)



Responses: carrera, dondeEstudio y optativas (imagen 8)

De esta manera queda concluida la presentación de las intenciones menos complejas, (las cuales tienen una única respuesta predeterminada para cada intent). La simplicidad utilizada en estos casos se basa en que no existe ninguna otra respuesta práctica o útil que le añadiera complejidad al intent y, por lo tanto, no tenía sentido complicar su implementación.

Asimismo, para que esto funcione y los intents se conecten con las responses, se tienen que utilizar stories. Estas stories son una representación de una conversación entre un usuario y un asistente de IA, convertida a un formato específico donde las entradas del usuario se expresan como intents (y entidades cuando sea necesario), mientras que las respuestas (resposes) y acciones (actions, métodos escritos en lenguaje python que se utilizan para realizar un trabajo determinado cuando se activa un intent en específico) se expresan con la palabra action. También se utilizan las reglas o rules, que describen fragmentos breves de conversaciones que siempre deben seguir el mismo camino.

El chatbot cuenta con una gran variedad de stories que engloban las conversaciones para las que está preparado. En primer lugar, tiene la story “saludar” la cual tiene el intent “saludar” y su respectivo response. También se encuentra la story “presentarse” que es la que se encarga de reconocer con quién está hablando el agente. Luego, están las referidas a identificar si el usuario es un profesor o no, que en caso de no tener registrada a la persona, se le pregunta si es un profesor. Sumado a esto, contiene una story encargada de reconocer el estado de ánimo del usuario.

```
- story: saludar
  steps:
  - intent: saludar
  - action: utter_saludar
  - checkpoint: check_presentacion

- story: presentarse
  steps:
  - checkpoint: check_presentacion
  - intent: sePresenta
    entities:
      - reconocerNombre
      - nro_dni
  - action: action_extraer_datos
  - checkpoint: check_animo
```

Stories: saludar y presentarse (imagen 9)

```
- story: estadoAnimo
  steps:
  - checkpoint: check_animo
  - or:
    - intent: contestacionesAnimoPositiva
    - intent: contestacionesAnimoNegativa
  - action: action_reconocerEstadoDeAnimo
  - checkpoint: check_profesor

- story: esProfesor
  steps:
  - checkpoint: check_profesor
  - intent: esProfe
  - action: action_cargarProfesor
  - action: utter_profesor

- story: NoesProfesor
  steps:
  - checkpoint: check_profesor
  - intent: noEsProfe
  - action: action_cargarProfesor
  - action: utter_profesor
```

Story: estadoAnimo, esProfesor, NoesProfesor (imagen 10)

Continuando, a estas stories se le suman las que se refieren a datos académicos, las cuales son: "academico_carrera_donde_estudio", "academico_consulta_materiasQueCurso", "academico_opiniones_aprobadas" y "academico_optativas".

```
- story: academico_carrera
  steps:
  - intent: carrera
  - action: utter_carrera

- story: academico_donde_estudio
  steps:
  - intent: dondeEstudio
  - action: action_consultaCarrera

- story: academico_consulta_materiasQueCurso
  steps:
  - intent: materiasQueCurso
  - action: action_cursando
```

Stories: academico_carrera, academico_donde_estudio y academico_consulta_materiasQueCurso (imagen 11)

```
- story: academico_opiniones
  steps:
  - intent: preguntaSobreMateria
    entities:
      - materias
  - action: action_opinionMaterias

- story: academico_aprobadas
  steps:
  - intent: materiasQueAprobaste
  - action: action_materiasAprobadas

- story: academico_optativas
  steps:
  - intent: consultaOptativas
  - action: utter_optativas
```

Stories: academico_opiniones, academico_aprobadas y academico_optativas (imagen 12)

Para finalizar con las stories, (exceptuando las que son de grupo) quedan presentar dos últimas, que son "gustos" y "datosPersonales". La primera representa las áreas de la informática que me interesan y la segunda incluye todos lo referido al deporte que realizo, mi edad y mi ciudad de origen.

```yaml
- story: datosPersonales
  steps:
  - intent: consultaDeEdad
  - action: utter_consultaEdad
  - intent: lugarNacimiento
  - action: utter_lugarNacimiento
  - intent: hacerDeporte
  - action: utter_consultaDeporte
```

Story: datosPersonales (imagen 13)

```yaml
- story: gustos
  steps:
  - intent: areasInformatica
  - action: utter_zonaInfor
```

Story: gustos (imagen 14)

Para proseguir, se mostrarán las rules. Estas son conversaciones cortas, las cuales siguen el flujo de conversación que se les determinó, estrictamente. Por lo tanto, las rules que se especificaron fueron:

"Regla de Despedida" (cuando el usuario se despide), "Regla del Porque" (representa cuando el usuario pregunta el porqué de algo), "Regla estado materias" (consulta sobre si una materia está aprobada, cursando o sin cursar) y por último, "Regla no se entendió el mensaje" (en caso de que la confianza al reconocer el mensaje sea menos del 50% para cada intent, se le pedirá al usuario que ingrese nuevamente el mensaje).

```yaml
- rule: Regla de despedida
  steps:
  - intent: adios
  - action: utter_adios

- rule: Regla del Porque
  steps:
  - intent: preguntaPorque
  - action: action_porque
```

Rules: Regla de despedida y Regla del Porqué (imagen 15)

```yaml
- rule: Regla estado materias
  steps:
  - or:
    - intent: materiaAprobada
    - intent: materiaCursando
  - action: action_estadoMateria

- rule: Regla no se entendio el mensaje
  steps:
  - intent: nlu_fallback
  - action: utter_porfavor_repetir
```

Rules: Regla estado materias y Regla no se entendio el mensaje (imagen 16)

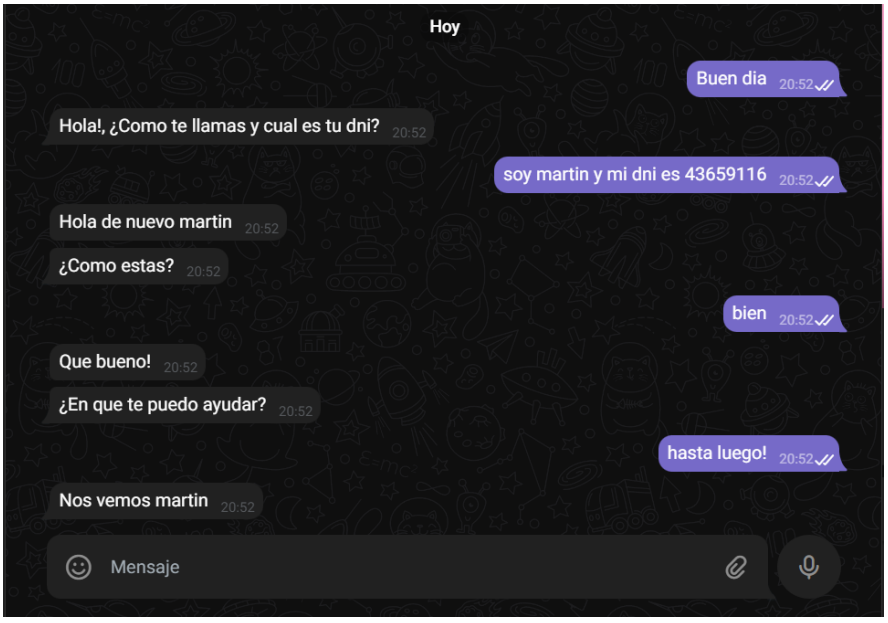
En el caso del "nlu_fallback", el cual sería el intent que "se activa" cuando ningún otro tiene una probabilidad mayor a 0.5. Para este caso, se tiene el response "utter_porfavor_repetir", que es el encargado de pedirle al usuario si puede volver a ingresar el mensaje. Como en casos anteriores, tiene un trato distinto si se da la situación de que la persona con la que está hablando es un profesor.

```yaml
utter_porfavor_repetir:
- condition:
    - type: slot
      name: esProfesor
      value: esProfesor
  text: "Disculpame no te pude entender bien, me lo podrias repetir"
- text: "No te entendi, me repetis"
- text: "me podrias repetir?"
```

Response: porfavor repetir (imagen 17)

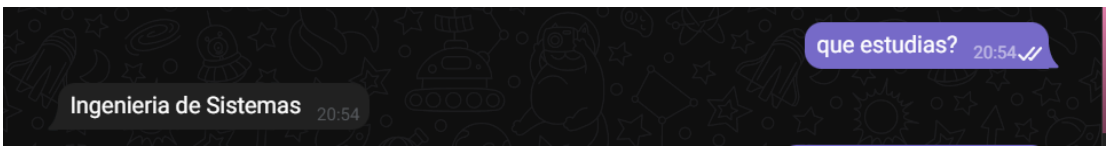
Para concluir esta descripción general del chatbot, se expondrán 3 imágenes del mismo funcionando en diferentes ocasiones:

En el primer caso, se expone la situación donde el usuario inicia la conversación con un saludo. Luego el bot va intentar reconocer con quién está hablando, y puede darse la posibilidad de que no conozca a la persona (caso contrario al de la imagen) y le pregunte si es un profesor o no, para registrarlo. A continuación, le consulta cómo está su día, para tener una idea de cómo es el estado de ánimo del usuario y para finalizar, la persona se despide. Lo que se puede apreciar es que el bot guardó el nombre de esta, para así realizar una despedida más amigable.



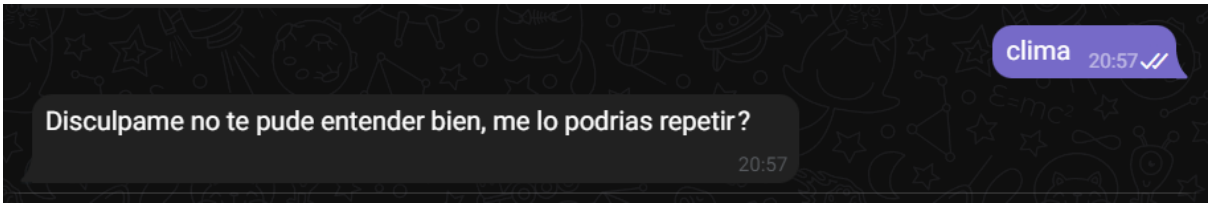
Ejemplo funcionando nro 1: Saludo, presentación del usuario y despedida. (imagen 18)

En el segundo ejemplo, se muestra el caso donde la persona consulta sobre la carrera que realizo, y como se puede apreciar en la imagen, el chatbot responde de manera eficiente a la consulta.



Ejemplo funcionando nro 2: consulta de la carrera (imagen 19)

Finalmente, en el último ejemplo se muestra el caso en el que el usuario ingresa un mensaje para el cual el chatbot no tiene un intent predefinido, entonces este responderá que no lo entendió y le pedirá que lo vuelva a ingresar.



Ejemplo funcionando nro 3: mensaje no entendido (imagen 20)

Aplicación de Machine Learning

"Machine learning" (aprendizaje automático) es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello. Constituye una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos para hacer predicciones. Esta tecnología está presente en un sinfín de aplicaciones como las recomendaciones de Netflix o Spotify, las respuestas inteligentes de Gmail o el habla de Siri y Alexa.

En el caso del chatbot, el mismo cuenta con los intent y las stories, que son los encargados de que pueda reconocer el mensaje y responder de manera adecuada. En el caso de los intent, estos tienen que tener una buena cantidad de ejemplos, para así poder reconocer el mensaje con una alta probabilidad. Para la situación de las stories, estas deben estar bien definidas, así el bot puede seguir correctamente el curso de la conversación y no generar respuestas inesperadas o, en el peor de los casos, no responder.

Para poder visualizar la importancia de que las intenciones tengan una cantidad considerable de ejemplos, se exhibirá a continuación un caso donde la intent está definida con pocos ejemplos y otro a posteriori, con una mayor cantidad de ejemplos.

Para el caso con pocos ejemplos, el chatbot no reconoce la intención (intent: carrera) del mensaje ("que estas estudiando?") con una gran confianza, como se puede apreciar en la imagen (la confianza aparece como confidence).

```
- intent: carrera
  examples: |
    - que estudias?
    - estudias?
    - que carrera haces?
```

Intent con pocos ejemplos (imagen 21)

```
Next message:
que estas estudiando?
{
  "text": "que estas estudiando?",
  "intent": {
    "name": "carrera",
    "confidence": 0.2930168947627612
```

Confianza con la que se reconoció el intent carrera (imagen 22)

Ahora para tener un mejor desempeño, se le agregan más cantidad de ejemplos y se lo entrena nuevamente. Con esto se logra que el bot aumente su confianza a la hora de seleccionar este intent, nuevamente se ingresa el mismo mensaje.

```
- intent: carrera
  examples: |
    - que estudias?
    - estudias?
    - que carrera haces?
    - estudias algo?
    - estas estudiando una carrera?
    - estas estudiando?
```

Intent con mayor cantidad de ejemplos (imagen 23)

```
Next message:
que estas estudiando?
{
  "text": "que estas estudiando?",
  "intent": {
    "name": "carrera",
    "confidence": 0.9980438947677612
```

Confianza con la que se reconoció el intent carrera (imagen 24)

Esto fue un claro ejemplo de la importancia de tener una buena cantidad de posibles entradas por parte del usuario, de modo que el chatbot pueda reconocer el mensaje de manera óptima. Y que gracias al machine learning se aumentó la confianza con la que el bot identifica un mensaje de ese intent.

Conexión con Prolog

Para que el chatbot fuera más inteligente y pueda saber datos académicos sobre las materias, se implementó una base de conocimiento en el lenguaje prolog. Prolog es un lenguaje de programación lógico e interpretado, usado habitualmente en el campo de la inteligencia artificial.

Esta base de conocimiento cuenta con los datos de todas las materias (código, nombre, cuatrimestre y año). Esta información es guardada en hechos para poder verificarlos y recuperarlos.

```
%Primer a?o - Primer Cuatrimestre%
es_una_materia(6111,'introduccion a la programacion 1',1,1).
es_una_materia(6112,'analisis matematico 1',1,1).
es_una_materia(6113,'algebra 1',1,1).
es_una_materia(6114,'quimica',1,1).
```

Declaración de hechos donde se especifican las materias del primer cuatrimestre de primero (imagen 25)

Además de los hechos, para la información de cada materia existen otros que guardan o registran si una asignatura ha sido aprobada o si la estoy cursando. Para esto, cada hecho guarda solamente el código de la materia para hacer más sencilla la comparación.

```
aprobada(6111).
aprobada(6112).
aprobada(6113).
aprobada(6114).
aprobada(6121).
```

Algunos hechos de las materias aprobadas. (imagen 26)

```
cursando(6321).
cursando(6322).
cursando(6323).
cursando(6324).
cursando(6325).
```

Hechos de las materias que estoy cursando. (imagen 27)

A partir de estos hechos, se construyeron las reglas que luego se utilizaron para poder satisfacer las consultas del usuario. El propósito que cumplieron estas reglas fue saber a partir del nombre de una materia, si está aprobada, si la estoy cursando o si todavía no la cursé, y también poder recuperar una lista con esas materias, para presentarla al usuario.

```
%materiasAprobadas%

materias_aprobadas(Lista) :- findall(Nom,(es_una_materia(Cod,Nom,_,_),aprobada(Cod)),Lista).

%materiasCursando%

materias_que_curso(Lista) :- findall(Nom,(es_una_materia(Cod,Nom,_,_),cursando(Cod)),Lista).

%materiasSinCursar%

materias_sin_cursar(Lista) :- findall(Nom,(es_una_materia(Cod,Nom,_,_),not(aprobada(Cod);cursando(Cod))),Lista).
```

Reglas en prolog que devuelven una lista con las materias aprobadas, que estoy cursando o sin cursar. (imagen 28)

```
%materiaAprobada%

materia_aprobada(Nom) :- es_una_materia(Cod,Nom,_,_),aprobada(Cod).

%materia cursando%

materia_en_cursada(Nom) :- es_una_materia(Cod,Nom,_,_),cursando(Cod).

%materia sin cursar%

materia_sin_cursar(Nom) :- es_una_materia(Cod,Nom,_,_),not(aprobada(Cod);cursando(Cod)).
```

Reglas en prolog que verifican si una materia está aprobada, la estoy cursando o sin cursar. (imagen 29)

Se eligió esta disposición en vez de usar solamente las reglas que retornan la lista, debido a que varias actions del chatbot solamente consultan si la materia fue aprobada o no. Esta consulta se le puede dar directamente a prolog, por lo que no es necesaria la lista. En consecuencia, es más correcta la implementación de esta manera ya que no se tiene que recorrer la lista para saber si una asignatura está aprobada o no, y además deja el código más legible.

A continuación, se van a desarrollar las intenciones del chatbot donde interviene prolog. Cabe resaltar que para poder dar una respuesta adecuada, estas contestaciones fueron implementadas mediante actions. Al bot se le puede preguntar por las materias que tengo aprobadas y las que estoy cursando, para lo cual se tiene un intent y una action cada una.

```
- intent: materiasQueCurso
  examples: |
    - que materias estas cursando?
    - que cursas?
    - En que materias te anotaste?
    - Que cursas este cuatrimestre?
    - que materias cursas?
```

intent: materiasQueCurso (imagen 30)

```
- intent: materiasQueAprobaste
  examples: |
    - que materias aprobaste?
    - materias que aprobaste?
    - cuales son las materias que aprobaste?
    - cuales aprobaste?
    - que finales tenes aprobados?
    - cuales finales aprobaste?
```

intent: materiasQueAprobaste (imagen 31)

A raíz de esto, también se le puede consultar si una materia en particular está aprobada o la estás cursando. Sus respectivos intent son muy similares a los anteriores,

con la única diferencia de que interviene la entidad “materias”, para poder identificar la asignatura de la que está hablando el usuario.

```
- intent: materiaCursando
  examples: |
    - cursas [programacion exploratoria](materias)
    - estas cursando ciencias de la computacion 2
    - te anotaste a introduccion a la programacion 1
    - cursas introduccion a la arquitectura de sistemas
    - estas cursando algebra lineal
    - te anotaste a disenio de sistemas de software
    - cursas analisis matematico 2
```

Intent: materiaCursando (imagen 32)

```
- intent: materiaAprobada
  examples: |
    - aprobaste [estructuras de almacenamiento de datos](materias)?
    - tenes aprobada a teoria de la informacion?
    - la aprobaste a bases de datos?
    - aprobaste ciencias de la computacion 1
    - como te fue en arquitectura de computadores y tecnicas digitales
    - tenes aprobada a disenio de compiladores
    - la aprobaste a quimica
    - aprobaste programacion orientada a objetos
    - tenes aprobada programacion exploratoria
    - tenes aprobada algebra lineal
    - la aprobaste a analisis y disenio de algoritmos 2
```

Intent: materiaAprobada (imagen 33)

La conexión de rasa con prolog se hizo mediante la biblioteca “swiplserver”. Luego, mediante funciones implementadas en esta, se le realizan las consultas a prolog. Cabe resaltar que las actions están escritas en lenguaje python.

```python
class ActionMateriasAprobadas(Action):

    def name(self) -> Text:
        return "action_materiasAprobadas"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        with PrologMQI(port=8000) as mqi:
            with mqi.create_thread() as prolog_thread:
                prolog_thread.query_async(r"consult('C:\\Users\\s7tan\\OneDrive\\Desktop\\Universidad\\Programacion Explorato
                prolog_thread.query_async(f"materias_aprobadas(X)", find_all=False)
                result = prolog_thread.query_async_result()[0]['X']
                message='Aprobe estas materias: \n'
                for materia in result:
                    message=message+materia+'\n'
                dispatcher.utter_message(text=f"{message}")
            return[]
```

Action: materiasAprobadas (imagen 34)

La última intención que utiliza prolog, es la que reconoce si le están consultando la opinión acerca de una materia en específico, la cual responde de acuerdo a la materia y al estado de la misma, es decir, si la materia la estoy cursando responde “la estoy cursando, más adelante te digo”, y si todavía no la cursé, contesta “No te puedo decir, la curso más adelante”. Para el caso que no suceda ninguna de las anteriores, buscará la opinión en un

archivo Json donde están guardadas todas las opiniones y si no la encuentra, responderá “No tengo una opinión formada”.

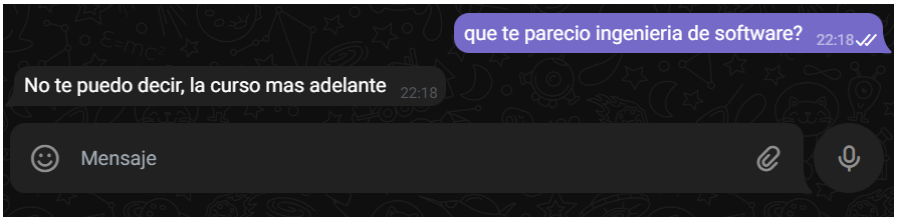
```
- intent: preguntaSobreMateria
  examples: |
    - que te parecio [algebra 1](materias)
    - como fue tu cursada de programacion orientada a objetos
    - que tal es electricidad y magnetismo
    - como es programacion exploratoria
    - que opinion tenes acerca de teoria de la informacion
    - que te parecio sistemas operativos
    - que te parece investigacion operativa
    - que tal es ciencias de la computacion 2
    - que opinion tenes acerca de estructuras de almacenamiento de datos
    - como fue la cursada de metodologias de desarrollo de software
    - que opinion tenes de algebra 1
    - que te parece disenio de compiladores
    - que tal es introduccion a la programacion 1
    - que te parecio arquitectura de computadoras y tecnicas digitales
    - que tal es analisis y disenio de algoritmos 1
    - que te parece [programacion exploratoria](materias)
```

intent: preguntaSobreMateria (imagen 35)

Asimismo, la action se conecta con prolog para determinar si la materia la estoy cursando o todavía no la cursé. Este fue uno de los motivos para la creación de la reglas que devuelven TRUE o FALSE de acuerdo el caso.

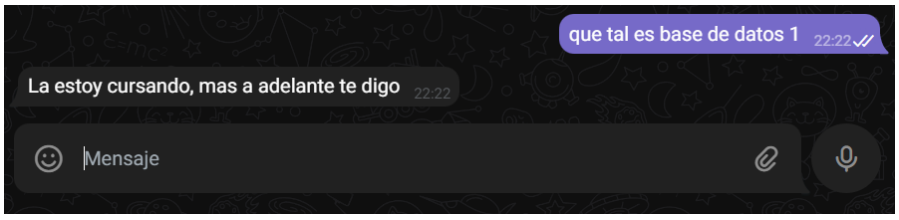
Para concluir este apartado, se exhibirá el funcionamiento de este último intent (opinión de las materias).

Materia aún no cursada:



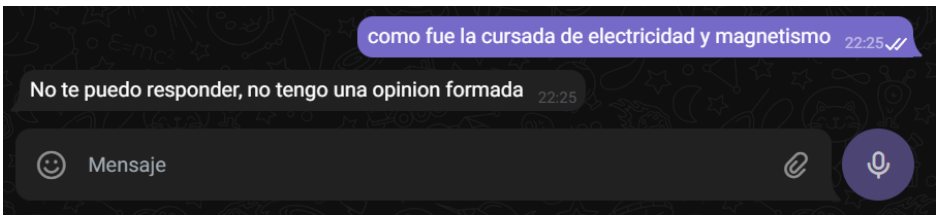
Ejemplo funcionando: opinion materia sin cursar (imagen 36)

Materia cursando actualmente:



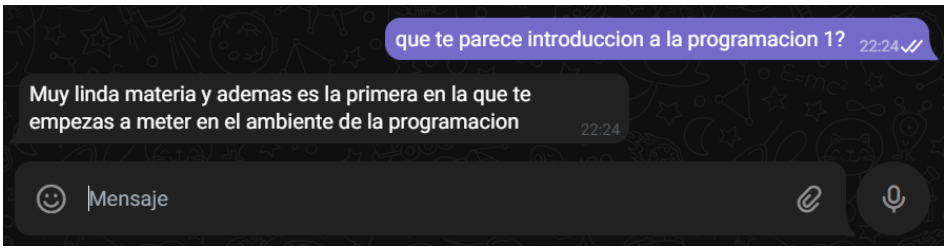
Ejemplo funcionando: opinion materia cursando (imagen 37)

Materia sin opinión cargada:



Ejemplo funcionando: materia sin opinión registrada (imagen 38)

Materia con opinión cargada:



Ejemplo funcionando: materia con opinión registrada (imagen 39)

Manejo de contexto

Una habilidad muy importante de un chatbot es poder mantener el contexto de una conversación y,para esto, rasa provee slots. Estos slots son la memoria del bot, que funcionan mediante clave-valor y son utilizados para almacenar la información proporcionada por el usuario o cargar datos, a partir de un criterio.

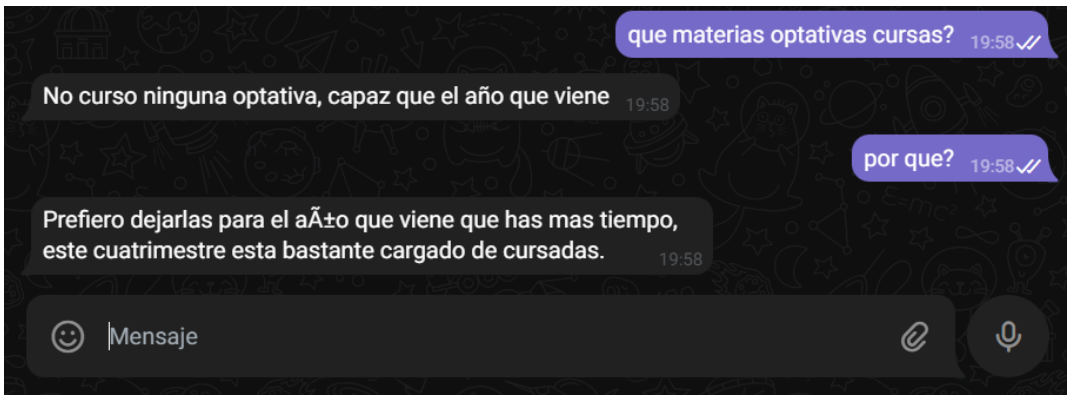
El chabot tiene dos manejos de contexto, el primero involucra la situación de que si el usuario pregunta el “por qué” hacía alguna respuesta del bot, este se la responde. La segunda situación, es que si se viene hablando sobre la carrera y le preguntan, por ejemplo ,“¿cómo vas?”, responde cómo me está yendo en la carrera y si no hay una conversación previa, el chatbot responde como se encuentra de ánimo.

Para el caso de que se le pregunte al bot el por qué de algunas de sus afirmaciones, se declaró un intent con varios ejemplos de cómo preguntar el por qué de algo. Y al ser declarado en una rule ejecuta la action_porque sin inconvenientes. Esta action se encarga de tomar el valor del slot y buscar la respuesta, la cual se encuentra almacenada en un archivo Json. Estas respuestas fueron guardadas en ese archivo para hacer más fácil su búsqueda y no tenerlas separadas. No todas las respuestas del chatbot tienen un por qué registrado, para el caso de que no lo posea, responde que no se ha puesto a pensar sobre eso. A continuación, se presenta la action involucrada y un ejemplo del bot respondiendo.

```python
class ActionPorque(Action):

    def name(self) -> Text:
        return "action_porque"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        opiniones = OperarArchivosJson.cargarArchivoPorque()
        consulta = tracker.get_slot('porque')
        message='No me lo he puesto a pensar'
        if str(consulta) in opiniones:
            message=opiniones[consulta]['porque']
        dispatcher.utter_message(text=str(message))
        return []
```

Action encarga de responder el por qué. (imagen 40)

Ahora, se exhibe una imagen del chatbot funcionando:

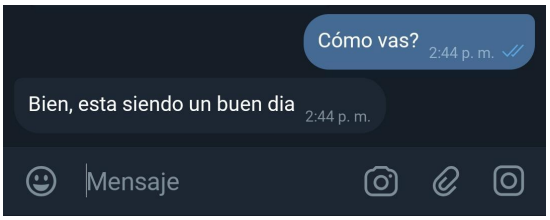


Ejemplo funcionando: manejo de contexto - respuesta de un ¿por qué? (imagen 41)

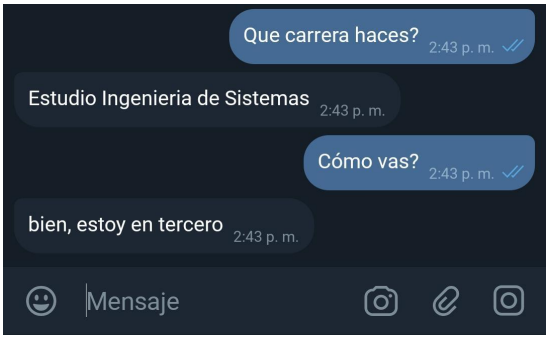
En el segundo caso, ante el reconocimiento del intent consultaCarrera, analiza el contexto de la conversación y realiza una respuesta de acuerdo a eso. Si la conversación viene dada sobre mi carrera universitaria, y se le pregunta, por ejemplo, “¿cómo vas?”, este

va a responder como me está yendo en la carrera. En cambio, si la interacción se viene desarrollando por otro tema totalmente distinto, el chatbot va a responder cuál es su estado de ánimo. El ánimo del chatbot se determina solicitando un número random entre -1 y 1, y en consecuencia si se encuentra en 0 y 1, contesta que se encuentra de buen humor. En cambio, si el número es negativo responderá que no tiene buen humor. La action encargada de esta operación se llama action_consultaCarrera.

Para finalizar, se presentará una imagen del chatbot funcionando para cada caso:



Ejemplo de respuesta a ¿cómo vas? si no se habla de la carrera (imagen 42)



Ejemplo de respuesta a ¿cómo vas? si no se habla de la carrera (imagen 43)

Trato Personal General

El chatbot, para generar una interacción más amigable con el usuario, le consulta su nombre para poder nombrarlo en diferentes puntos de la conversación. De esta forma, se puede causar un efecto positivo en la persona a la hora de interactuar con el agente.

El bot reconoce el nombre mediante una entidad llamada “reconocerNombre”, que tiene asociada una lookup table (tabla de búsqueda) con los posibles valores. De esta manera, la entidad tomará los nombres que se encuentren en esta tabla y, en consecuencia, se quita la posibilidad de que el usuario ingrese cualquier palabra y sea reconocida por la entidad.

```
- lookup: reconocerNombre
  examples: |
    - Martino
    - Juan Ignacio
    - Juan
    - David
    - Leonel
    - Tomas
```

Fragmento de la lookup table (imagen 44)

A continuación, se presentará una imagen del momento en el que la entidad registra el nombre de la persona y una situación donde lo utiliza para responder.

Ejemplo de captura y uso del nombre del usuario (imagen 45)

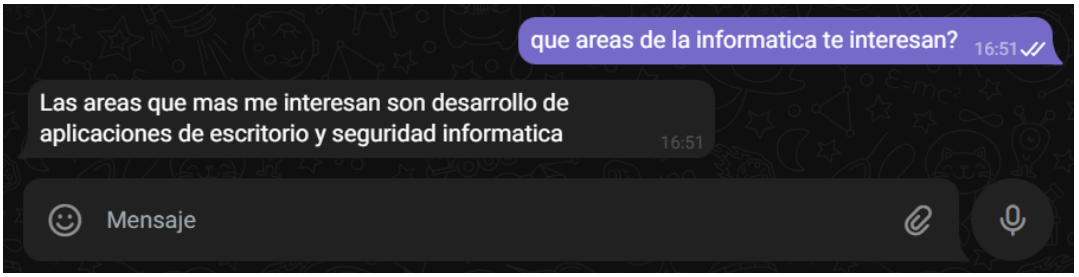
Trato Personal Específico

Para el trato específico, el chatbot cuenta con dos elementos. En primer lugar, reconocer si se encuentra hablando con un profesor o no y, en segundo lugar, si la persona con la que está conversando ya lo ha hecho antes o es la primera vez.

En el caso del primer elemento, para capturar si la persona es un profesor o no, la acción que realiza el chatbot es preguntarle al usuario y que este responda. Ese dato es capturado en un slot, el cual es consultado en varias respuestas del mismo. Con esa información el bot puede hacer una diferenciación a la hora de realizar una respuesta hacia una consulta. Como el agente es una representación mía como alumno, es de gran utilidad saber hacia quién se está dirigiendo, para dar una respuesta acorde.

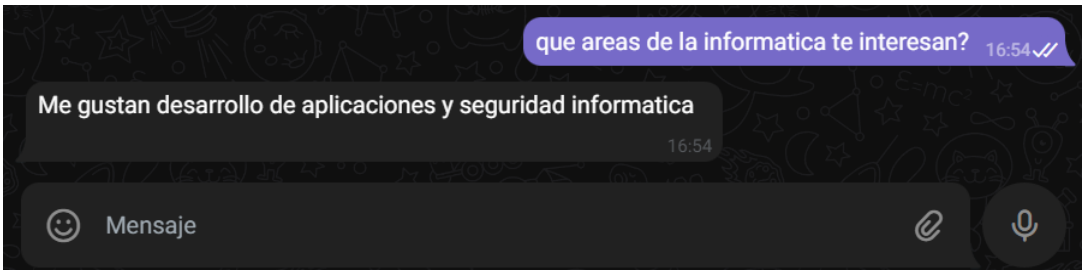
Seguidamente, se muestra un ejemplo de un profesor y de una persona que no lo es:

Como se puede presenciar, el bot responde de una manera más formal al momento de interactuar con un docente.



Ejemplo de respuesta a un profesor (imagen 46)

Para terminar con la primera presentación de trato personal específico, se muestra la respuesta del bot hacia el mismo mensaje que el anterior, pero respondiendo a una persona que no es profesor:



Ejemplo de respuesta a un profesor (imagen 47)

La segunda habilidad del bot es tener la capacidad de recordar con quién estuvo hablando y, de esta forma, a priori ya reconoce si es un profesor o no. Para poder realizar esto, se le solicita al usuario que ingrese su número de documento y se lo busca en un archivo Json, que funciona como una especie de agenda. Si la persona no está registrada, se le pregunta los datos para luego agendarlo y ya tenerlo empadronado para futuras conversaciones. En caso contrario, si el usuario está registrado, se tomarán los datos guardados en el archivo.

Se le solicita el número de DNI al usuario, dado que en el Json se utiliza como una especie de clave, ya que si este fuera otro campo (como por ejemplo, el nombre) pueden existir dos personas con la misma clave y el chatbot no sabría con certeza a quién se está refiriendo. Otra situación por la que se eligió el DNI por encima del número de legajo de un alumno, es que algunos usuarios son profesores o directamente externos a la universidad y, por lo tanto, no hay un número y/o clave para distinguirlos mejor que el del documento.

A continuación, se mostrarán las dos actions encargadas de reconocer a la persona y de capturar los datos en caso de que no esté registrada:

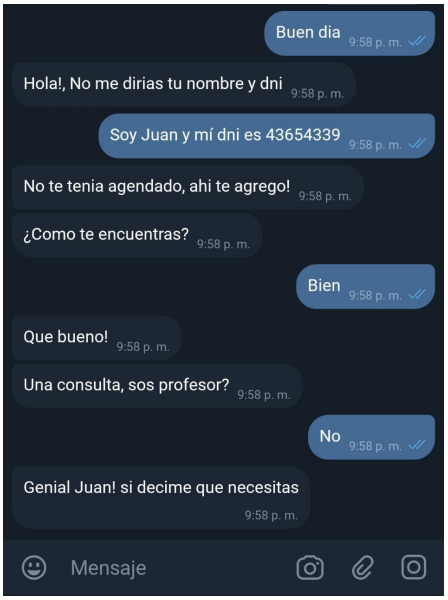
```python
class ActionExtraerDatos(Action):

    def name(self) -> Text:
       return "action_extraer_datos"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        personas = OperarArchivosJson.cargarArchivoPersona()
        dni = next(tracker.get_latest_entity_values('nro_dni'), None)
        print('entre')
        if str(dni) in personas:
            dispatcher.utter_message(text=str('Hola de nuevo ' + personas[dni]['nombre']))
            dispatcher.utter_message(response="utter_consultaComoesta")
            consulProfe=str(personas[dni]['esProfesor'])
            return [SlotSet('esProfesor',consulProfe)]
        else:
            dispatcher.utter_message(text=str('No te tenia agendado, ahi te agrego!'))
            dispatcher.utter_message(response="utter_consultaComoesta")
            return []
```

Action encargada de reconocer si una persona está registrada (imagen 48)

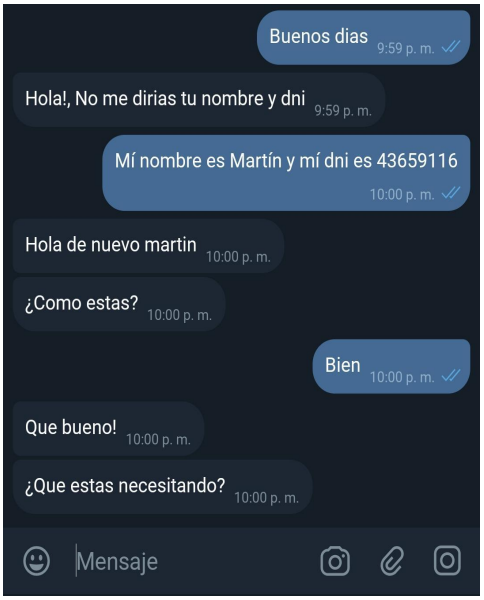
```python
class ActionCargarSiEsProfesor(Action):

    def name(self) -> Text:
        return "action_cargarProfesor"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        datoProfesor = tracker.get_slot('esProfesor')
        dni = tracker.get_slot('dni')
        nombre = tracker.get_slot('nombre')
        personas = OperarArchivosJson.cargarArchivoPersona()
        personas[dni]={}
        personas[dni]['nombre']=str(nombre)
        personas[dni]['esProfesor']=str(datoProfesor)
        OperarArchivosJson.guardarPersona(personas)
        return []
```

Action encargada de registrar al usuario (imagen 49)

Ahora, se presentará cómo responde el bot para ambos casos:



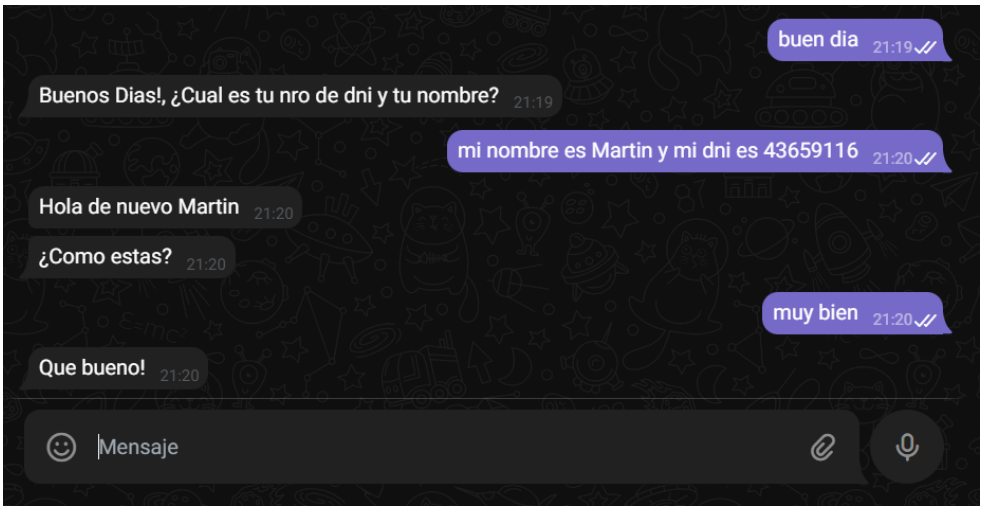
Ejemplo de registrar persona (imagen 50)



Ejemplo de persona registrada (imagen 51)

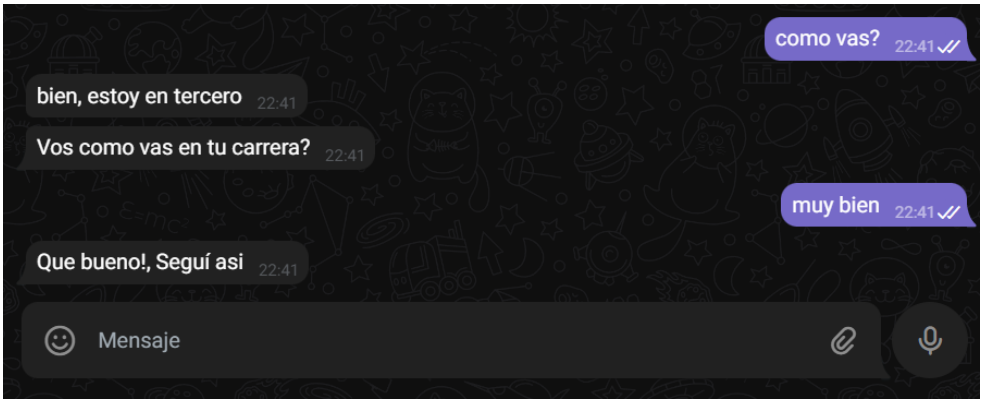
Computación afectiva

La computación afectiva es una rama de estudio y desarrollo de la inteligencia artificial que hace referencia al diseño de sistemas y dispositivos que pueden reconocer, interpretar y procesar emociones humanas. En este trabajo, se tomaron dos formas a la hora de reconocer estas emociones. La primera, se realiza de una forma sencilla preguntándole al usuario cómo se encuentra, y el bot responde en base al mensaje ingresado. Puede responder “qué bueno” o “no te preocupes, todo pasa”, de acuerdo el caso. Además, almacena el estado de ánimo dicho por el usuario, dado que si llega a ser negativo, al despedirse le diga que mejore su día.



Ejemplo funcionando: capturando estado de ánimo (imagen 52)

En el segundo caso, se abordó la situación de que si se está hablando con un alumno, se le consulta cómo va con su carrera y, en base a su respuesta, se realiza una devolución. Antes de continuar con la implementación de esta función, se presenta una imagen del funcionamiento donde el usuario responde cómo va con su carrera.



Ejemplo funcionando: devolución hacia cómo le está yendo en la carrera al usuario (imagen 53)

La implementación en ambos casos, es la misma. Tienen dos intents en común, los cuales abordan una gran cantidad de diferentes contestaciones acerca de cómo se

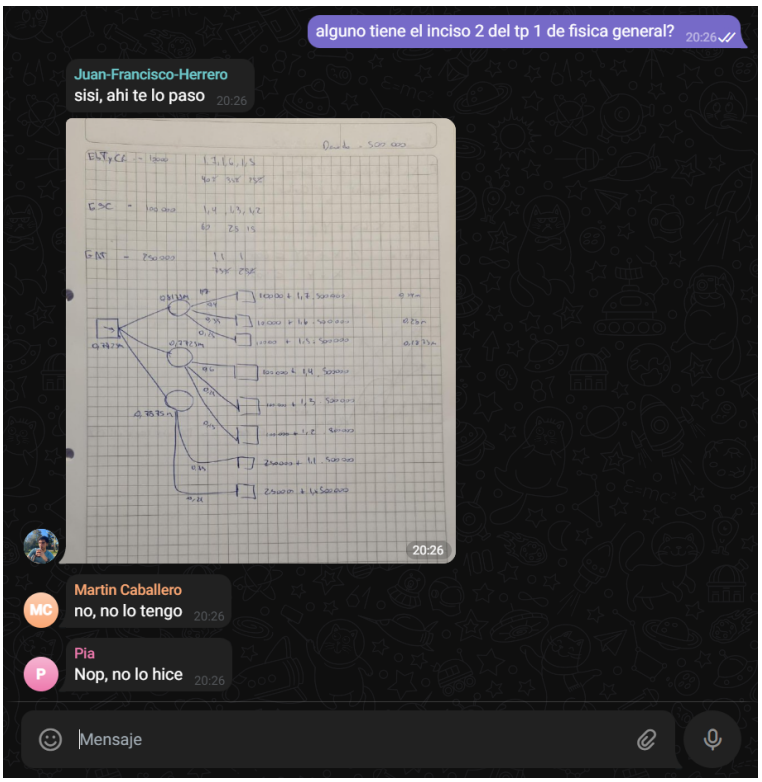
encuentra en ambos casos, y luego se invoca una action que procesa la entrada. Esta action identifica cuál de los dos intents se reconocieron (si el positivo o el negativo) y, además, verifica si se está hablando de la carrera. En caso de que se le pregunte cómo se encuentra, se le setea el valor del slot estadoAnimo en True si está bien, y en False si no. Esto se hace para poder consultar su estado más tarde. Y si reconoce que se refiere a su situación académica, solamente se le hace una devolución. Todo lo mencionado, se implementó en una sola action para poder simplificar la solución y no repetir código, dado que sus comportamientos son muy similares.

Grupal

Para la interacción grupal, decidimos nuevamente realizar un enfoque académico ya que cursamos las mismas materias y estamos en constante comunicación. Los mensajes reales en el grupo de WhatsApp, se deben a la gran mayoría de los casos a pedido de ejercicios de trabajos prácticos de las diferentes materias.

Dado el caso, los chatbots de cada integrante del grupo, se pensaron para que puedan responder ante consultas de ejercicios de cualquiera de los integrantes. La implementación fue similar a otras de este chatbot, utilizando un archivo Json donde se guardan todas las materias, y cada una se subdivide en los trabajos prácticos y luego, cada práctico en incisos. En la situación que no contenga la resolución del enunciado, se responde de forma negativa y en el caso inverso, el bot automáticamente envía la foto del ejercicio que solicitó. Cabe resaltar que la implementación de este comportamiento se llevó adelante mediante una action, que se ejecuta cada vez que el chatbot detecta la solicitud de un ejercicio.

Puede darse la situación de generar un loop infinito de respuestas en el chat, donde cada bot responde las consultas de los otros, luego contestan esas respuestas y así sucesivamente. Para evitar esto, se utilizaron varios intents para que ante la detección de uno de estos no realice ninguna acción. Como por ejemplo, si un chatbot responde "No lo tengo", los otros bot no tienen que responder ese mensaje, sino que ignorarlo. A continuación, se presenta una imagen del funcionamiento.



Ejemplo funcionando: Parte grupal (imagen 54)

Conclusión

Para concluir este informe, es fundamental poner énfasis en el rol que va tomando la inteligencia artificial en nuestro día a día, donde cada vez se ven más asistencias de la IA hacia actividades cotidianas, para lograr la realización de diferentes actividades con menos propensión a algún error, para mejorar la calidad de vida de la sociedad y, en varios casos, para conseguir una mayor seguridad. La IA tiene un gran potencial que puede ser llevado a varios campos para lograr una mejor performance y así poder brindar un mejor servicio.

Los chatbots tienen un futuro prometedor, ya que pueden brindar excelentes servicios a los clientes/usuarios, como por ejemplo realizar la atención al cliente de una empresa. Este trabajo es una ligera visión de lo que puede realizar un chatbot y muestra la flexibilidad de estos para adicionarles nuevas tecnologías. Por todo lo expuesto, los chatbots, son, indudablemente, una herramienta que puede ofrecer grandes servicios y ayudas al ser humano.